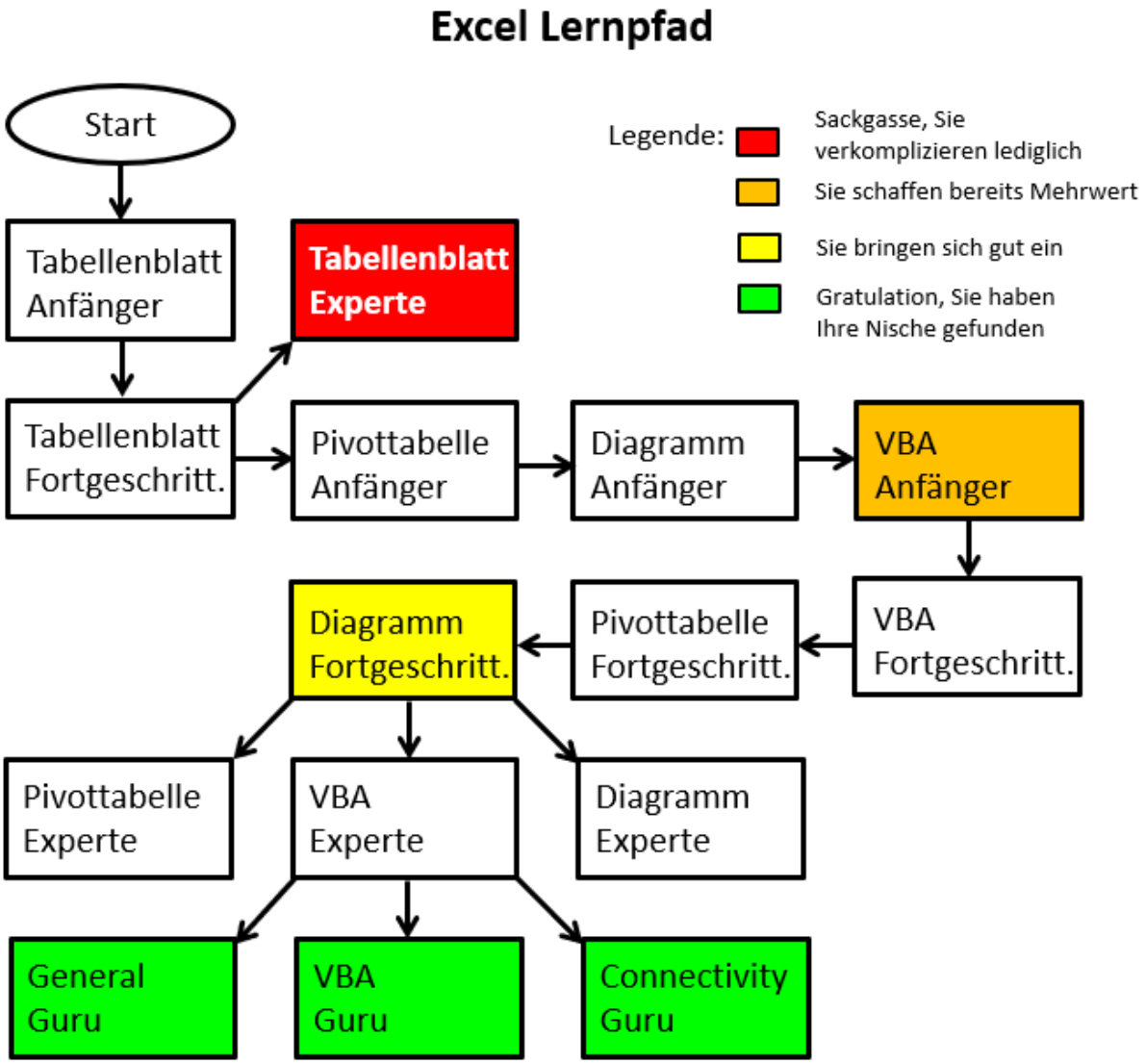
Excel / VBA – Eine Sammlung



 Stand: 2. Februar 2025

Abstract

Dies ist eine Sammlung von Excel VBA Programmen und auch einigen Excelformeln, die mir sinnvoll erschienen.

Bernd Plumhoff, 2. Februar 2025

Excel Lernpfad

Excel Bereiche / Expertenniveau	**Was Sie wissen sollten**	**Womit Sie andere in die Irre führen**	**Bereiche oder Stufen, die Sie kennen oder erreichen können**
Tabellenblattfunktionen / Dummkopf	Nicht alle Zellen rechtsbündig formatieren: das Zellformat Standard zeigt Text linksbündig und Zahlen rechtsbündig; keine fehlerhaften Namensdefinitionen haben		Lernen Sie die Funktionen WENN, TEXT, DATUM, XVERWEIS
VBA / Dummkopf	Funktionen und Subroutinen nicht wie Module benennen		Makros aufzeichnen
Tabellenblattfunktionen / Anfänger:in	1<>“1”, Funktionen: ZEICHEN, DATUM, TEIL, RUNDEN, TEXT, HEUTE, XVERWEIS	Mit Formeln angeben, die Sie nicht verstehen	Lernen Sie die Funktion SUMMENPRODUKT, verstehen Sie volatile Funktionen
VBA / Anfänger:in	Wie man Makros aufzeichnet und Programmteile von anderen kopiert; Integer können nur Werte von -32768 bis 32767 annehmen; OPTION EXPLICIT verwenden		Lernen Sie wie man Makroaufzeichnungen säubert, verwenden Sie eine Namenskonvention
Pivot Tabelle / Anfänger:in	Erstellen Sie einfache Pivot Tabellen		Lernen Sie etwas VBA um Pivot Tabellen automatisch zu befüllen und zu aktualisieren
Diagramm / Anfänger:in	Erstellen Sie einfache Diagramme		Lernen Sie von Peltiers Website https://peltiertech.com/Excel/Charts/chartvba.html
Tabellenblattfunktionen / Fortgeschritten	Verweisen Sie auf Spalten links vom Suchwert mit INDEX(VERGLEICH()); wann Sie Pivot Tabellen oder VBA Code an Stelle von Tabelenblattfunktionen verwenden sollten; vermeiden Sie volatile Formeln	Verwenden Sie INDIREKT oder Formeln die mehr als 2 Zeilen im Formeleditor beanspruchen	Lernen Sie gutes Tabellendesign, lernen Sie etwas VBA
VBA / Fortgeschritten	Entwickeln Sie Programme für bestimmte Anwendungsgebiete, verwenden Sie einfache Datei-Ein- und Ausgaben		Lernen Sie wann eine Pivot Tabelle sinnvoll ist und wann Tabellenblattfunktionen ausreichen

Excel Bereiche / Expertenniveau	Was Sie wissen sollten	Womit Sie andere in die Irre führen	Bereiche oder Stufen, die Sie kennen oder erreichen können
Pivot Tabelle / Fortgeschritten	Aktualisieren Sie sie mit VBA		
Diagramm / Fortgeschritten	Verwenden Sie fortgeschrittene Diagramm Eigenschaften		Lernen Sie noch mehr von Peltiers Website https://peltiertech.com/Excel/Charts/chartvba.html
Tabellenblattfunktionen / Experte	Messen Sie die Rechenzeiten mit https://jkp-ads.com/articles/performanceclass.aspx	Verwenden Sie Tabellenblattfunktionen um ihrer selbst willen, auch wenn sie quadratische Laufzeiten aufweisen	Lernen Sie bitte VBA, wirklich
VBA / Experte	Add-in Verwendung wie Fincad, benutzen Sie Klassen wo immer es Sinn macht, progammieren Sie allgemeinen wiederverwendbaren Code		Lernen Sie wie Klassen aufgebaut und programmiert werden - siehe SystemState (Seite 15)und Logging (Seite 18)
Diagramm / Experte	Aktualisieren Sie Diagramme automatisch mit Tabellenblattfunktionen oder via VBA,		
VBA / Wizard	Programmieren Sie Klassen wie SystemState (Seite 15)und Logging (Seite 18)		Bleiben Sie nicht bei simpler Datei-Ein- und Ausgabe hängen. Lernen Sie wie man Daten mit Datenbanken austauscht
Excel & Datenbanken / Wizard	Nutzen Sie Datenbanken effizient		
General / Wizard	Add-in Entwicklung	Veröffentlichen Sie nicht Ihren Quellcode, kümmern Sie sich nicht um Dokumentation oder Nachfolge = Stellen Sie nicht sicher, dass Andere Ihre Lösungen verwenden können, wenn Sie gegangen sind	

Die Grafik auf der Titelseite zeigt ein Beispiel für einen möglichen Excel Lernpfad.

Inhaltsverzeichnis

Die Excel / VBA Programmierumgebung

Abstract

Mit Visual Basic for Applications (VBA kann man in der Tabellenkalkulation Excel Aufgaben automatisieren und spezielle Funktionalitäten programmieren, die im Funktionsumfang von Excel nicht enthalten sind.

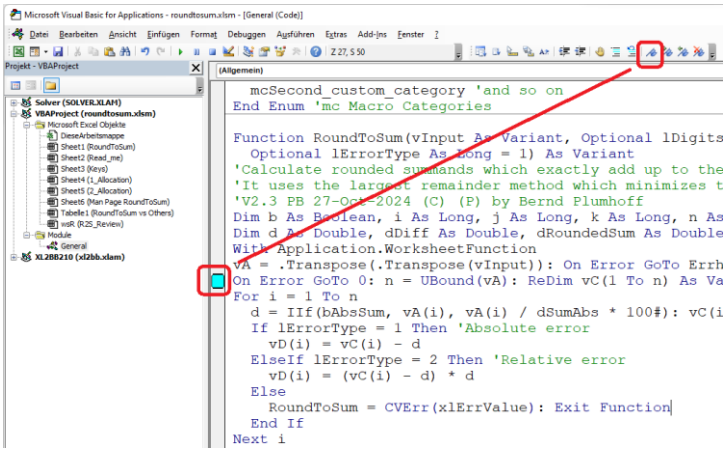
Hier zeige ich Programme, die ich nützlich fand, als ich Betriebsprozesse planen, umsetzen und ausführen musste.

Grundlegendes

Während des Editierens

Mit `Option Explicit` erzwingen Sie in einem VBA Modul die Deklaration von allen verwendeten Variablen. Wer dies nicht verwendet, sollte nicht programmieren.
Wenn der Cursor in der VBA Programmierungebung auf einer Variablen oder einem Objeckt steht, springen Sie mit SHIFT und der Funktionstaste F2 (SHIFT + F2) direkt zur Deklaration dieser Variablen (nur Dim, nicht ReDim). Mit STRG + SHIFT + F2 springen Sie zurück.

Im VBA Editor können Sie mit dem Symbol „blaue Flagge“ Textstellen markieren, zu denen Sie mit den angrenzenden Zeigersymbolen schnell springen können:



Mit dem Programmbefehl `Stop` oder wenn Sie auf den grauen Bereich gleich links neben einer Programmzeile klicken, können Sie einen Stopppunkt (engl. breakpoint) setzen:

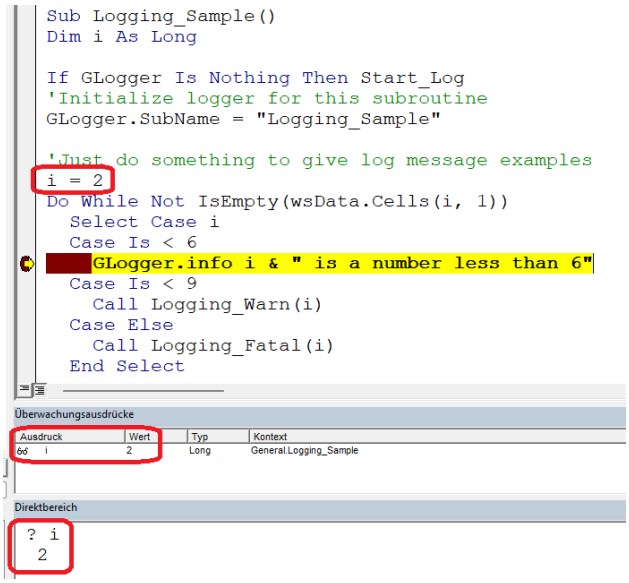
```
Sub Logging_Sample()
Dim i As Long

If GLogger Is Nothing Then Start_Log
'Initialize logger for this subroutine
GLogger.SubName = "Logging_Sample"

'Just do something to give log message examples
i = 2
Do While Not IsEmpty(wsData.Cells(i, 1))
  Select Case i
  Case Is < 6
    Stop
    GLogger.info i & " is a number less than 6"
  Case Is < 9
    Call Logging_Warn(i)
  Case Else
    Call Logging_Fatal(i)
  End Select
  i = i + 1
Loop
```

Wenn das Programm später ausgeführt wird, dann stoppt es an dieser Stelle, und Sie können z. B. den Inhalt von Variablen oder Tabellenblättern prüfen.

Während der Programmausführung

Breakpoints

Wenn ein VBA Programm ausgeführt wird, können Sie es durch Drücken der ESC Taste unterbrechen. Auch der Befehl `Stop` oder ein Stopppunkt unterbricht das Programm, wenn es an die entsprechende Stelle gelangt, und kennzeichnet die aktuelle Programmzeile gelb:



Sie können nun mit dem Cursor zur Variablen i an der Stelle „i = 2“ gehen (nicht klicken, lediglich darüber schweben, „hovern“). Es wird ein kleines Fenster i = 2 gezeigt. Sie können auch i auswählen, die rechte Maustaste drücken und eine Überwachung für i hinzufügen. Dann wird der Wert von i im Fenster Überwachungsausdrücke gezeigt.

Mit STRG + g kommen Sie nach einer Programmunterbrechung in den Direktbereich (engl. Immediate Window) . Hier können Sie einzelne Programmbefehle eingeben wie z. B. „Print i“ (oder kurz „? I“ – das Fragezeichen ist das Kürzel für den Befehl Print).

Sie können nach einer Programmunterbrechung das Programm mit der Funktionstaste F5 weiterlaufen lassen. Sie können es aber mit F8 auch weiter schrittweise Befehl-für-Befehl ausführen. Mit SHIFT + F8 an Stelle von F8 verzweigt das Programm nicht in Unterprogramme, sondern führt diese als einen Befehl aus.

Fehlerbehandlung

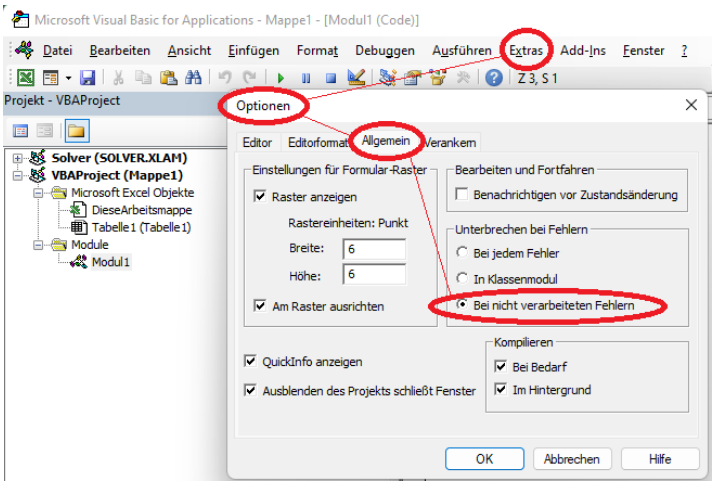
Ihr Excel VBA Programm muss Im Fehlerfall nicht notwendigerweise stoppen. Sie können Fehler abfangen mit `On Error GoTo <Label>` oder Sie ignorieren alle Fehler einfach mit `On Error Resume Next`.

Eine gute Einführung bietet Chip Pearson's Seite
http://www.cpearson.com/excel/errorhandling.htm

Ein etwas komplexeres Praxisbeispiel finden Sie in meiner benutzerdefinierten Funktion sbMiniPivot. Da wir die genaue Dimension des Ausgabebereiches nicht kennen können, starten wir mit einer kleineren Vorgabe, die wir bei Bedarf erhöhen und am Ende des Programms mit `ReDim Preserve` auf die genau benötigte Größe reduzieren – `ReDim Preserve` kann die letzte Dimension eines Arrays verändern, ohne die bisherigen Elemente zu löschen:

```
…
lvdim = UBound(vInput(0))
If lvdim > 100 Then lvdim = 100 'Let us start with small dimension
On Error GoTo ErrHdl
…
'Reduce result array to used area
If k > 0 Then ReDim Preserve vR(0 To UBound(vInput) + liscount, 1 To k)
…
Exit Function

ErrHdl:
If Err.Number = 9 Then
  If i > lvdim Then
    'Here we normally get if we breach Ubound(vR,2)
    'So we need to increase last dimension
    lvdim = 10 * lvdim
    If lvdim > UBound(vInput(0)) Then lvdim = UBound(vInput(0))
    ReDim Preserve vR(0 To UBound(vInput) + liscount, 1 To lvdim)
    Resume 'Back to statement which caused error
  End If
End If
'Other error - terminate
On Error GoTo 0
Resume
```

Ein wichtiger Hinweis: Stellen Sie sicher, dass Ihre Optionen in VBA auf "Unterbrechen bei nicht verarbeiteten Fehlern" gesetzt sind!



Falls “Unterbrechen bei jedem Fehler” gewählt ist, wird Ihr VBA Fehlerbehandlung “On Error …” einfach ignoriert - bei jedem Fehler wird Excel mit einer Fehlermeldung gestoppt.

“Unterbrechen in Klassenmodul” ist eine vernünftige Option, wenn Sie Fehler in Ihren Klassenmodulen suchen oder sie testen. Falls diese Option nicht gewählt ist, wird Excel im Normalfall einen Fehler nicht im Klassenmodul zeigen, sondern mit einer Fehlermeldung bei dem aufrufenden Code zurückkehren. Ist diese Option gesetzt, wird Excel bei einem Fehler in einem Klassenmodul stoppen und dort mit einer Fehlermeldung anhalten.

Leider verfügt Excel nicht wie MS Access über eine Funktion wie `Application.GetOption("Error Trapping")`. Mark Lundberg veröffentlichte den Trick

```
Application.SendKeys "%{F11}%TO+{TAB}{RIGHT 2}%E~%{F4}"
```

aber ich empfehle hier normale, natürliche Achtsamkeit.

Gute Programmierpraxis

Seien Sie ein guter Programmierer

Das Wichtigste an einem guten VBA Programm ist, dass es ein gutes Programm ist, und nicht voller VBA Tricks. Wenn Sie keine assoziativen Arrays und keine Klassen kennen, dann müssen sie auf dem Boden herumkriechen und bekommen Ihre Anwendung nie zum Fliegen.

Gutes Excel und VBA Wissen

Sie sollten Excel and VBA gut beherrschen. Mit dem VBA Befehl *Enum* können Sie z. B. Tabellenspalten Namen geben. Dies vereinfacht Programmänderungen - oder wollen Sie alle hart kodierten Verweise auf Spalten rechts von einer neu eingefügten oder gelöschten Spalte anpassen?

Programmierkonventionen

Lernen Sie Namenskonventionen und Kodierkonventionen. Durch ihre Anwendung machen Sie Ihre Programme lesbarer, leichter zu warten, und verlässlicher und effizienter.
Beispiellinks:
https://de.wikibooks.org/wiki/VBA_in_Excel/_Namenskonventionen
https://learn.microsoft.com/de-de/dotnet/visual-basic/programming-guide/program-structure/coding-conventions

Säubern Sie Makroaufzeichnungen

Selbstverständlich nehme ich ein Makro auf, wenn ich einen VBA Befehl vergessen habe. Aber wenn Sie aufgezeichneten ungesäuberten Spaghetticode verwenden, muss dies Ihr Nachfolger aufräumen.

Dokumentieren Sie Ihr Programm ausreichend

Erklären Sie was ein kundiger Dritter wissen muss, aber schreiben Sie keine Romane über Trivialitäten. Gute Dokumentation kommt mit dem Programm - nicht danach. Nur Dummköpfe und Menschen, die sich unentbehrlich machen wollen, dokumentieren nicht.

Testen Sie Ihr Programm gut

Anwendungen ab einer gewissen Größe benötigen Testprogramme oder sogar eine Serie von Regressionstests.

Protokollieren Sie Ihre Programmausführung

Mit einer Logger Klasse (Programmablauf dokumentieren – *Logging* Klasse) können Sie einem Revisor zeigen, wer Ihr Programm mit welchen Parametern ausführte und ob Ihr Programm problemlos durchlief.

Optimieren Sie Ihr Programm

Die Qualität Ihrer Anwendung wird maßgeblich durch ihr Design und ihre Struktur bestimmt. Je komplexer die Aufgabe, desto besser sollte Ihr Fachwissen und Ihre Erfahrung sein. Durch das Messen der Laufzeiten einzelner Programmteile (engl. Profiling) können Sie erkennen, wo noch Verbesserungen möglich sind.

Jan Karel Pieterse hat eine hilfreiche Klasse für das Profiling erstellt:
https://jkp-ads.com/Articles/performanceclass.asp

Systemstatus sichern und zurückschreiben – *SystemState* Klasse

Mein ehemaliger Kollege Jon T. entwickelte die kleinste mir bekannte sinnvolle VBA Klasse: Mit *SystemState* kann man Systemstatusvariablen leicht speichern und für eigene Zwecke optimieren. Um die Programmausführung zu beschleunigen, schreibt man normalerweise zu Beginn eines VBA Makros

```
Application.Calculation = xlCalculationManual
Application.ScreenUpdating = False
```

und am Ende des Makros

```
Application.Calculation = xlCalculationAutomatic
Application.ScreenUpdating = True
```

Mit dem Klassenmodul *SystemState* schreibt man am Start lediglich

```
Dim state As SystemState
Set state = New SystemState
'Bitte beachten: Dies kann NICHT mit "Dim state as New SystemState"
abgekürzt werden!
```

und am Ende

```
Set state = Nothing 'Nicht einmal nötig - dies wird automatisch gemacht
```

Systemstatus Variablen

Die Klasse *SystemState* sichert und restauriert die folgenden Systemstatus Variablen:

Variable	Status	Kommentar / Zu optimieren durch …
Calculation	xlCalculationAutomatic, xlCalculationManual, xlCalculationSemiautomatic	Bestimmt ob nach jeder Zelländerung eine Neuberechnung durchgeführt wird. Auf xlCalculationManual setzen.
Cursor	xlDefault, xlBeam, xlNorthwestArrow, xlWait	Dies ist lediglich eine Anzeige. Am besten nicht anfassen - es sei denn, man möchte den Debug Modus z. B. mit einem Sanduhr-Zeiger beginnen.
DisplayAlerts	Wahr, Falsch	Auf Falsch (engl. False) setzen um Systemrückfragen abzuschalten.
EnableAnimations	Wahr, Falsch	Ab Excel Version 2016 kann man hiermit Excel's Bildschirmanimationen abschalten.
EnableEvents	Wahr, Falsch	Auf Falsch (engl. False) setzen um Ereignisprozeduren an der Ausführung zu hindern.
Interactive	Wahr, Falsch	Am besten nicht anfassen - es sei denn, man möchte unbedingt alle Tastatureingaben verhindern.
PrintCommunication	Wahr, Falsch	Auf Falsch (engl. False) setzen um Seiteneinstellungen zu ändern ohne auf Rückantwort vom Drucker zu warten.
ScreenUpdating	Wahr, Falsch	Auf Falsch (engl. False) setzen um Bildschirmaktualisierungen während der Programmausführung abzuschalten.
StatusBar	Falsch, "Eine beliebige Benutzerinformation"	Der Text wird in der Statuszeile (unterste Bildschirmzeile) gezeigt. Auf Falsch (engl. False) setzen um die Anzeige zu löschen.

SystemState Programmcode

Bitte kopieren Sie den folgenden Programmcode in ein Klassenmodul mit dem Namen *SystemState*, nicht in ein normales Modul.

```
'This class has been developed by my former colleague Jon T.
'I adapted it to newer Excel versions. Any errors are mine for sure.
'(C) (P) by Jon T., Bernd Plumhoff 3-Nov-2024 PB V1.5
'
'The class is called SystemState.
'It can of course be used in nested subroutines.
'
'This module provides a simple way to save and restore key excel
'system state variables that are commonly changed to speed up VBA code
'during long execution sequences.
'
'Usage:
'    Save() is called automatically on creation and Restore() on destruction
'    To create a new instance:
'        Dim state as SystemState
'        Set state = New SystemState
'    Warning:
'        "Dim state as New SystemState" does NOT create a new instance
'
'
'    Those wanting to do complicated things can use extended API:
'
'    To save state:
'       Call state.Save()
'
'    To restore state and in cleanup code: (can be safely called multiple times)
'       Call state.Restore()
'
'    To restore Excel to its default state (may upset other applications)
'       Call state.SetDefaults()
'       Call state.Restore()
'
'    To clear a saved state (stops it being restored)
'       Call state.Clear()
'
Private Type m_SystemState
  Calculation As XlCalculation
  Cursor As XlMousePointer
  DisplayAlerts As Boolean
  EnableAnimations As Boolean   'From Excel 2016 onwards
  EnableEvents As Boolean
  Interactive As Boolean
  PrintCommunication As Boolean 'From Excel 2010 onwards
  ScreenUpdating As Boolean
  StatusBar As Variant
  m_saved As Boolean
End Type

'
'Instance local copy of m_State?
'
Private m_State As m_SystemState

'
'Reset a saved system state to application defaults
'Warning: restoring a reset state may upset other applications
'
Public Sub SetDefaults()
  m_State.Calculation = xlCalculationAutomatic
  m_State.Cursor = xlDefault
  m_State.DisplayAlerts = True
  m_State.EnableAnimations = True
  m_State.EnableEvents = True
  m_State.Interactive = True
  On Error Resume Next 'In case no printer is installed
  m_State.PrintCommunication = True
  On Error GoTo 0
  m_State.ScreenUpdating = True
  m_State.StatusBar = False
  m_State.m_saved = True 'Effectively we saved a default state
End Sub

'
'Clear a saved system state (stop restore)
'
Public Sub Clear()
    m_State.m_saved = False
End Sub

'
```

```
'Save system state
'
Public Sub Save(Optional SetFavouriteParams As Boolean = False)
  If Not m_State.m_saved Then
    m_State.Calculation = Application.Calculation
    m_State.Cursor = Application.Cursor
    m_State.DisplayAlerts = Application.DisplayAlerts
    m_State.EnableAnimations = Application.EnableAnimations
    m_State.EnableEvents = Application.EnableEvents
    m_State.Interactive = Application.Interactive
    On Error Resume Next 'In case no printer is installed
    m_State.PrintCommunication = Application.PrintCommunication
    On Error GoTo 0
    m_State.ScreenUpdating = Application.ScreenUpdating
    m_State.StatusBar = Application.StatusBar
    m_State.m_saved = True
  End If
  If SetFavouriteParams Then
    Application.Calculation = xlCalculationManual
    Application.DisplayAlerts = False
    Application.EnableAnimations = False
    Application.EnableEvents = False
    On Error Resume Next 'In case no printer is installed
    Application.PrintCommunication = False
    On Error GoTo 0
    Application.ScreenUpdating = False
    Application.StatusBar = False
  End If
End Sub

'
'Restore system state
'
Public Sub Restore()
  If m_State.m_saved Then
    'We check now before setting Calculation because setting
    'Calculation will clear cut/copy buffer
    If Application.Calculation <> m_State.Calculation Then
      Application.Calculation = m_State.Calculation
    End If
    Application.Cursor = m_State.Cursor
    Application.DisplayAlerts = m_State.DisplayAlerts
    Application.EnableAnimations = m_State.EnableAnimations
    Application.EnableEvents = m_State.EnableEvents
    Application.Interactive = m_State.Interactive
    On Error Resume Next 'In case no printer is installed
    Application.PrintCommunication = m_State.PrintCommunication
    On Error GoTo 0
    Application.ScreenUpdating = m_State.ScreenUpdating
    If m_State.StatusBar = "FALSE" Then
      Application.StatusBar = False
    Else
      Application.StatusBar = m_State.StatusBar
    End If
  End If
End Sub

'
'By default save when we are created
'
Private Sub Class_Initialize()
  Call Save(SetFavouriteParams:=True)
End Sub

'
'By default restore when we are destroyed
'
Private Sub Class_Terminate()
  Call Restore
End Sub
```

Programmablauf dokumentieren – *Logging* Klasse

Diese Logger Klasse bietet Logging mit den Berichtsstufen INFO, WARN, FATAL und EVER an. Die Programminformationen werden sowohl in einem Tabellenblatt als auch in einer Datei festgehalten.

Die Anwendung dieser Logger Klasse ist nicht schwer: Das allgemeine Modul Logger_Factory und das Klassenmodul Logger kopieren wir aus der unten bereitgestellten Beispieldatei in die eigene Anwendung, anschließend definieren wir die Konstante `Public Const AppVersion` zum Beispiel mit dem Wert “Meine Anwendung Version 1.0” im Hauptmodul, und nun können wir mit

```
GLogger.info "Info Meldung ..."
GLogger.warn "Warn Meldung ..."
GLogger.fatal "Fehler Meldung ..."
GLogger.ever "Nichtunterdrückbare Standard Meldung ..."
```

eigene Logmeldungen erzeugen und automatisch im Tabellenblatt Workflow und in der Logdatei “Meine Anwendung Version 1.0_Logfile_yyyymmdd.log” im Unterverzeichnis Logs speichern.

Ich erhielt den ursprünglichen Programmcode 2009 von Cliff G. und erweiterte ihn später. Cliff verwendete dieses Programm hauptsächlich zum Debuggen. Ich finde es auch sehr sinnvoll, um Programmläufe zu Revisionszwecken zu protokollieren, oder damit ein Programm dem Benutzer ggf. detailliert seine einzelnen Ausführungsschritte erläutert. Weiter fügte ich Versionsinformationen und System- oder Excel-Parameter hinzu, um rasch wichtige Unterschiede zwischen verschiedenen Benutzerumgebungen zu ermitteln. Mit diesem Logger messe ich auch grob die Laufzeiten von SQL Abfragen:

```
'Glogger is declared in module LoggerFactory and set in Sub Start_Log()
Dim dtStamp As Date
'...
dtStamp = Now
'Retrieve data from database here
Glogger.info "SQL xxx ran " & Format(Now - dtStamp, "n:ss") & " [m:ss]"
```

Für und Wider

Dieses Logging Programm bietet meines Erachtens die sinnvollste sekundäre Funktionalität für jede VBA Anwendung. Man kann:

- nachvollziehbar testen
- ein Programm alle seine Ausführungsschritte nachvollziehbar erklären lassen
- feststellen, ob mehrere Anwender eine Anwendung gleichzeitig nutzen
- erkennen, ob ein Anwenderproblem auf einer unterschiedlichen Umgebung beruht
- auch sporadische Anwendungsfehler systematisch eingrenzen
- über einen längeren Zeitraum hinweg auch Revisoren überzeugend die korrekte fehlerfreie Nutzung nachweisen (einzelne Logdateien können zwar manipuliert werden, aber eine größe Menge von Logdateien wirkt dennoch hinreichend überzeugend)
- die Laufzeit von VBA (Unter-)Routinen grob bestimmen
- die Durchlaufzeiten von gesamten Prozessen messen

Der letzte obige Punkt wird Betriebs- und Personalräte hellhörig machen:

- wenn man Durchlaufzeiten von ganzen Prozessen misst, könnte man die Leistung einzelner Mitarbeiter ermitteln, vergleichen und ggf. gegen sie verwenden.

Dies wäre ein klarer Verstoß gegen die DSGVO (Datenschutz-Grundverordnung), siehe https://dsgvo-gesetz.de/

Ich habe dieses Logging nie gegen meine Mitarbeiter oder Anwender für eine Leistungsmessung verwendet, sondern lediglich für Nachschulungen genutzt, wenn ich fehlerhafte Nutzungen erkannte. Aber dies kann selbstverständlich nicht als Argument für eine unbedenkliche Nutzung dienen.

Eine Zustimmung von Betriebs- und Personalräten kann m. E. immer erreicht werden, wenn man auf die Freiwilligkeit dieser Selbstaufschreibung hinweist:

- jeder Anwender kann das Logging vor einem Programmlauf ein- oder ausschalten
- jeder Anwender kann die Log-Dateien zu jeder Zeit im Nachhinein löschen

Ich setzte und setze in Europa in mehreren Ländern (UK, Deutschland) bei mehreren Gesellschaften (Banken, Versicherungen, IT Providern) dieses Logging ohne jede Beanstandung erfolgreich ein.

Parameter

Public (öffentliche) Konstanten

AppVersion - Diese Zeichenkette sollte den Programmnamen und seine Version enthalten, z. B.:

```
Public Const AppVersion As String = "... Version x"
```

Dann wird "… Version x" als Versionsinformation für dieses Programm protokolliert.

Compilerkonstanten

Separate_Log_Files_for_each_User - True erstellt für jeden Benutzer eigene Logdateien, False führt zu einer täglichen Logdatei für alle Benutzer

Use_Logger_auto_Open_Close - True verwendet die Subroutinen auto_open und auto_close im Modul LoggerFactory, False nicht.

Logging_on_Screen - In LoggerFactory und Logger auf True setzen um Nachrichten auch im Tabellenblatt Workflow zu zeigen. Hinweis: der interne VBA Name von Worksheet muss ‚wsW' lauten

Logging_cashed - In LoggerFactory und Logger auf True setzen um das Logging zu beschleunigen. Log Nachrichten werden dann erst am Ende des Programlaufs in eine Datei geschrieben. Hierfür muss auch Logging_on_Screen auf True gesetzt werden.

Log_WMI_Info - In LoggerFactory auf True setzen um interessante Windows Management Instrumentation (WMI) Informationen auszugeben wie z. B. Prozessor-, Speicher-, Laufwerks- und Betriebssystem-Angaben.

Show_Reference_Details - True zeigt alle Details, False zeigt lediglich die Beschreibung.

Logging Variablen

LogFilePath - Vollständiger Pfadname der Logdatei

SubName - Muss am Anfang jeder Subroutine gesetzt werden um den Sub Namen korrekt im Log zu protokollieren

LogLevel - Die Logging Berichtsstufen:

```
1 - Alle Log Nachrichten protokollieren: INFO, WARN, FATAL, and EVER
2 - Alle Log Nachrichten mit Ausnahme von Stufe INFO protokollieren
3 - Nur FATAL und EVER Log Nachrichten protokollieren
4 - Lediglich EVER Log Nachrichten protokollieren
5 - Kein Logging
```

LogScreenRow - Startzeile für das Logging in Tabellenblatt Workflow (gewöhnlich 3)

Beispielausgabe

Die Logs werden standardmäßig im Tabellenblatt Workflow und in der Logdatei im Unterverzeichnis Logs ausgegeben:

```
EVER: BERND-CAPTIVA\earso 12.11.2024 03:57:15 [Start_Log] - Logging started with Logging_Version_13
EVER: BERND-CAPTIVA\earso 12.11.2024 03:57:15 [Start_Log] - Microsoft Windows 11 Home 10.0.22000 (64-Bit)
and Excel 2024 (64-Bit)
INFO: BERND-CAPTIVA\earso 12.11.2024 03:57:15 [Start_Log] - Application ThousandsSeparator '.',
DecimalSeparator ',', use system separators
INFO: BERND-CAPTIVA\earso 12.11.2024 03:57:15 [Start_Log] - App.Internl ThousandsSeparator '.',
DecimalSeparator ',', ListSeparator ';'
INFO: BERND-CAPTIVA\earso 12.11.2024 03:57:15 [Start_Log] - App.Internl xlCountryCode '49',
xlCountrySetting '49'
INFO: BERND-CAPTIVA\earso 12.11.2024 03:57:15 [Start_Log] - VBAProject References: Visual Basic For
Applications, Microsoft Excel 16.0 Object Library, OLE Automation, Microsoft Office 16.0 Object Library
EVER: BERND-CAPTIVA\earso 12.11.2024 03:57:18 [End_Log] - Logging finished with Logging_Version_13
```

Module

Die Logs werden standardmäßig im Tabellenblatt Workflow und in der Logdatei im Unterverzeichnis Logs ausgegeben:

Normal

LoggerFactory enthält Konstanten, öffentliche Variablen, Standard Logger Einstellungen und optionale Auto-Open and Auto-Close Subroutinen.

Hinweis: Die Prozedur *Start_Log* benötigt (ruft auf) die Prozeduren *ApplicationVersion* und *getOperatingSystem* (https://www.devhut.net/vba-determine-the-installed-os/) sowie das Modul *LibFileTools* (https://github.com/cristianbuse/VBA-FileTools), um auch Verzeichnisse unter OneDrive und Sharepoint zu unterstützen.

```
'This general module is named LoggerFactory. Together with class module Logger it offers logging
functionality.
'Requires LibFileTools: https://github.com/cristianbuse/VBA-FileTools
'Version When         Who             What
'      1 Once upon .. Cliff G.        Initial version
'     14 01-Feb-2025  Bernd Plumhoff  sPath corrected
#Const Separate_Logfiles_for_each_User = False
#Const Use_Logger_auto_Open_Close = True 'Enable auto_open and auto_close subs in here
#Const Logging_on_Screen = True          'IMPORTANT: Also change this constant in class module Logger! We
like to see recent run's loggging messages on screen in tab Workflow
#Const Logging_cashed = False            'IMPORTANT: Also change this constant in class module Logger!
Write logging messages into file at program end to speed this up
#Const Log_WMI_Info = False              'True shows interesting Windows Management Instrumentation (WMI)
data
#Const Show_Reference_Details = False    'True: Show all details; False: Just show description
Public GLogger As Logger                 'Global logfile object - variable scope is across all modules
Public GsThisLogFilePath As String
' Constant log levels
Public Const INFO_LEVEL As Integer = 1
Public Const WARN_LEVEL As Integer = 2
Public Const FATAL_LEVEL As Integer = 3
Public Const EVER_LEVEL As Integer = 4 'For logging messages which cannot be switched off
Public Const DISABLE_LOGGING As Integer = 5
'The application-specific defaults
Const DEFAULT_LOG_FILE_PATH As String = "" 'Force error if not set [Bernd 12-Aug-2009]
Const DEFAULT_LOG_LEVEL As Integer = INFO_LEVEL

Public Function getLogger(sSubName As String) As Logger
  Dim oLogger As New Logger
  oLogger.SubName = sSubName
  'Defaults to the specified values - but may be overridden before used
  oLogger.LogLevel = DEFAULT_LOG_LEVEL
  oLogger.LogFilePath = DEFAULT_LOG_FILE_PATH
  Set getLogger = oLogger
End Function

#If Use_Logger_auto_Open_Close Then
  Sub auto_open()
  'Version Date        Programmer Change
  '9       12-Sep-2021 Bernd      Code outsorced to Start_Log so that user does not need to use auto_open
  Start_Log
  End Sub

  Sub auto_close()
  'Version Date        Programmer Change
  '3       12-Sep-2021 Bernd      Code outsorced to End_Log so that user does not need to use auto_close
  End_Log
  End Sub
#End If '#If Use_Logger_auto_Open_Close

Sub Start_Log()
'Version Date        Programmer Change
'3       02-Nov-2023 Bernd      Log interesting Windows Management Instrumentation (WMI) infos
'4       27-Feb-2024 Bernd      Show_Reference_Details added
'5       12-Nov-2024 Bernd      Using LibFileTools https://github.com/cristianbuse/VBA-FileTools
Dim i As Long
Dim s As String, sDel As String, sPath As String
#If Log_WMI_Info = True Then
  Dim oWMISrvEx As Object 'SWbemServicesEx
  Dim oWMIObjSet As Object 'SWbemServicesObjectSet
  Dim oWMIObjEx As Object 'SWbemObjectEx
  Dim oWMIProp As Object 'SWbemProperty
  Dim sWQL As String 'WQL Statement
  Dim v As Variant
#End If
'LibFileTools are necessary just for the next 5 lines:
sPath = GetLocalPath(ThisWorkbook.Path)
If Right(sPath, 1) <> "\" Then sPath = sPath & "\"
sPath = sPath & "Logs\"
If Not IsFolder(sPath) Then
  CreateFolder (sPath)
End If
If GLogger Is Nothing Then Set GLogger = New Logger
#If Separate_Logfiles_for_each_User Then
  'If AppVersion is not defined please define it in your main module like:
  'Public Const AppVersion As String = "Application Version ..."
  GLogger.LogFilePath = sPath & Environ("Userdomain") & _
    "_" & Environ("Username") & "_" & AppVersion & "_" & "Logfile_" & _
    Format(Now, "YYYYMMDD") & ".txt"
#Else
  GLogger.LogFilePath = sPath & AppVersion & "_" & _
    "Logfile_" & Format(Now, "YYYYMMDD") & ".txt"
#End If
GLogger.LogLevel = 1
#If Logging_on_Screen Then
  GLogger.LogScreenRow = 3
  wsW.Range("E2:E4").ClearContents
  wsW.Range("5:65535").Delete
#End If
'Initialize logger for this subroutine
```

```
With Application
GLogger.SubName = "Start_Log"
GLogger.ever "Logging started with " & AppVersion
#If Log_WMI_Info = True Then
  Set oWMISrvEx = GetObject("winmgmts:root/CIMV2")
  For Each v In Array("BaseService", "Processor", "PhysicalMemoryArray", "LogicalDisk", "OperatingSystem")
    'Not: "NetworkAdapterConfiguration", "VideoController", "OnBoardDevice", "Printer", "Product"
    Set oWMIObjSet = oWMISrvEx.ExecQuery("Select * From Win32_" & v)
    For Each oWMIObjEx In oWMIObjSet
      s = v & ": "
      For Each oWMIProp In oWMIObjEx.Properties_
        If Not IsNull(oWMIProp.Value) Then
          If Not IsArray(oWMIProp.Value) Then
            Select Case v
            Case "BaseService"
              If InStr("'SystemName'", "'" & oWMIProp.Name & "'") > 0 Then
                GLogger.ever oWMIProp.Name & "='" & Trim(oWMIProp.Value) & "'"
                GoTo Next_v
              End If
            Case "Processor"
              If
InStr("'Name'Description'NumberOfEnabledCore'AddressWidth'DataWidth'CurrentClockSpeed'LoadPercentage'", _
                "'" & oWMIProp.Name & "'") > 0 Then
                If IsNumeric(oWMIProp.Value) Then
                  s = s & oWMIProp.Name & "=" & Format(oWMIProp.Value, "#,##0") & ", "
                Else
                  s = s & oWMIProp.Name & "='" & Trim(oWMIProp.Value) & "', "
                End If
              End If
            Case "PhysicalMemoryArray"
              If InStr("'MaxCapacityEx'", _
                "'" & oWMIProp.Name & "'") > 0 Then s = s & oWMIProp.Name & "=" & Format(oWMIProp.Value,
"#,##0") & ", "
            Case "LogicalDisk"
              If InStr("'DeviceID'ProviderName'Size'FreeSpace'", _
                "'" & oWMIProp.Name & "'") > 0 Then
                If IsNumeric(oWMIProp.Value) Then
                  s = s & oWMIProp.Name & "=" & Format(oWMIProp.Value, "#,##0") & ", "
                Else
                  s = s & oWMIProp.Name & "='" & Trim(oWMIProp.Value) & "', "
                End If
              End If
            Case "OperatingSystem"
              If
InStr("'FreePhysicalMemory'FreeVirtualMemory'FreeSpaceInPagingFiles'MaxProcessMemorySize'InstallDate'", _
                "'" & oWMIProp.Name & "'") > 0 Then s = s & oWMIProp.Name & "=" & Format(oWMIProp.Value,
"#,##0") & ", "
            End Select
          End If
        End If
      Next oWMIProp
      If Len(s) > Len(v & ": ") Then GLogger.ever Left(s, Len(s) - 2)
    Next oWMIObjEx
Next_v:
  Next v
#End If
#If Win64 Then
  s = "64"
#Else
  s = "32"
#End If
GLogger.ever getOperatingSystem() & " and " & ApplicationVersion() & _
  " (" & s & "-Bit)" '& .Version & .Build & " (" & .CalculationVersion & ")"
GLogger.info "Application ThousandsSeparator '" & .ThousandsSeparator & _
  "', DecimalSeparator '" & .DecimalSeparator & "', " & _
  IIf(Not (Application.UseSystemSeparators), "do not ", "") & "use system separators"
GLogger.info "App.Internl ThousandsSeparator '" & .International(xlThousandsSeparator) & _
  "', DecimalSeparator '" & .International(xlDecimalSeparator) & "', ListSeparator '" & _
  .International(xlListSeparator) & "'"
GLogger.info "App.Internl xlCountryCode '" & .International(xlCountryCode) & _
  "', xlCountrySetting '" & .International(xlCountrySetting) & "'"
End With
With ThisWorkbook.VBProject.References 'In case of error tick box Trust access to the VBA project object
  'model under File / Options / Trust Center / Trust Center Settings / Macro Settings
  s = "VBAProject References: "
  On Error Resume Next
  For i = 1 To .Count
    #If Show_Reference_Details Then
      GLogger.info s
      s = ""
      s = s & .Item(i).Description
      s = s & ", FullPath: '" & .Item(i).fullPath & "'"
      s = s & ", Guid: " & .Item(i).GUID
      s = s & ", BuiltIn: " & .Item(i).BuiltIn
      s = s & ", IsBroken: " & .Item(i).IsBroken
      s = s & ", Major: " & .Item(i).Major
      s = s & ", Minor: " & .Item(i).Minor
    #Else
      s = s & sDel & .Item(i).Description
      sDel = ", "
```

```
    #End If
  Next i
  GLogger.info s
End With
'Now two examples of environment variables which might not exist for all Windows / Excel installations.
'Use Sub List_Environ_Variables below to see which variables exist on your system.
s = ""
s = Environ("CRC_VDI-TYPE") 'If this does not exist we will not log anything
If s <> "" Then GLogger.info "CRC_VDI-TYPE: '" & s & "'"
s = ""
s = Environ("ORACLE_HOME_X64") 'If this does not exist we will not log anything
If s <> "" Then GLogger.info "Oracle Client: '" & s & "'"
On Error GoTo 0
End Sub

Sub End_Log()
'Change History:
'Version Date        Programmer Change
'1       12-Sep-2021 Bernd      Initial version so that user does not need to use auto_close. He can
manually call this sub.
If GLogger Is Nothing Then Call auto_open
GLogger.SubName = "End_Log"
'If AppVersion is not defined please define it in your main module like: Public Const AppVersion As String
= "Application Version ..."
GLogger.ever "Logging finished with " & AppVersion
#If Logging_cashed Then
  Set GLogger = Nothing 'Necessary, or Class_Terminate() won't be called for GLogger because it's Public
#End If
End Sub
```

Ein Beispielmodul General welches zeigt wie man den Logger nutzen kann:

```
Option Explicit
'Version When         Who             What
'     11 03-Nov-2023  Bernd Plumhoff  Log interesting Windows Management Instrumentation (WMI) infos
'     12 17-Feb-2024  Bernd Plumhoff  Show_Reference_Details added
'     13 12-Nov-2024  Bernd Plumhoff  Using LibFileTools https://github.com/cristianbuse/VBA-FileTools

Public Const AppVersion As String = "Logging_Version_13"

Sub Logging_Sample()
Dim i As Long

If GLogger Is Nothing Then Start_Log
'Initialize logger for this subroutine
GLogger.SubName = "Logging_Sample"

'Just do something to give log message examples
i = 2
Do While Not IsEmpty(wsData.Cells(i, 1))
    Select Case i
    Case Is < 6
        GLogger.info i & " is a number less than 6"
    Case Is < 9
        Call Logging_Warn(i)
    Case Else
        Call Logging_Fatal(i)
    End Select
    i = i + 1
Loop

#If Logging_cashed Then
Set GLogger = Nothing 'Necessary, or Class_Terminate() won't be called for GLogger since it's Public
#End If

End Sub

'You do not need extra subroutines to log warn messages or fatal messages.
'They are just examples of additional subroutines which do some logging.
Sub Logging_Warn(i As Long)
    'Initialize logger for this subroutine
    GLogger.SubName = "Logging_Warn"
    GLogger.warn i & " is 6, 7, or 8"
End Sub

Sub Logging_Fatal(i As Long)
    'Initialize logger for this subroutine
    GLogger.SubName = "Logging_Fatal"
    GLogger.fatal i & " is greater 8"
End Sub
```

Klassenmodule

Logger enthält die Logging Funktionalität:

```
'This class module is named Logger. Together with class module LoggerFactory it offers logging
functionality.
'Version When         Who             What
'      1 Once upon .. Cliff G.        Initial version
'     14 01-Feb-2025  Bernd Plumhoff  Same version as LoggerFactory
#Const Logging_on_Screen = True 'IMPORTANT: Also change this constant in module LoggerFactory! We like to
see recent run's loggging messages on screen in tab Workflow
#Const Logging_cashed = False   'IMPORTANT: Also change this constant in module LoggerFactory! Write
logging messages into file at program end to speed this up
Const INFO_LEVEL_TEXT As String = "INFO:"
Const WARN_LEVEL_TEXT As String = "#WARN:"
Const FATAL_LEVEL_TEXT As String = "##FATAL:"
Const EVER_LEVEL_TEXT As String = "EVER:"
Private sThisSubName As String
Private iThisLogLevel As Integer
#If Logging_on_Screen Then
  Private iThisLogRow As Integer
  Public Property Let LogScreenRow(iLogRow As Integer)
    iThisLogRow = iLogRow
  End Property

  Public Property Get LogScreenRow() As Integer
    LogScreenRow = iThisLogRow
  End Property
#End If

Public Property Let LogFilePath(sLogFilePath As String)
  GsThisLogFilePath = sLogFilePath
End Property

Public Property Get LogFilePath() As String
  LogFilePath = GsThisLogFilePath
End Property

Public Property Let SubName(sSubName As String)
  sThisSubName = sSubName
End Property

Public Property Get SubName() As String
  SubName = sThisSubName
End Property

Public Property Let LogLevel(iLogLevel As Integer)
  iThisLogLevel = iLogLevel
End Property

Public Property Get LogLevel() As Integer
  LogLevel = iThisLogLevel
End Property

Public Sub info(sLogText As String)
  If Me.LogLevel = LoggerFactory.INFO_LEVEL Then
    Call WriteLog(LoggerFactory.INFO_LEVEL, sLogText)
  End If
End Sub

Public Sub warn(sLogText As String)
  If Me.LogLevel < LoggerFactory.FATAL_LEVEL Then
    Call WriteLog(LoggerFactory.WARN_LEVEL, sLogText)
  End If
End Sub

Public Sub fatal(sLogText As String)
  If Me.LogLevel <= LoggerFactory.FATAL_LEVEL Then
    Call WriteLog(LoggerFactory.FATAL_LEVEL, sLogText)
  End If
End Sub

Public Sub ever(sLogText As String)
  If Me.LogLevel <= LoggerFactory.EVER_LEVEL Then
    Call WriteLog(LoggerFactory.EVER_LEVEL, sLogText)
  End If
End Sub

Private Sub WriteLog(iLogLevel As Integer, sLogText As String)
  Dim FileNum As Integer, LogMessage As String, sDateTime As String, sLogLevel As String
  Select Case iLogLevel
  Case LoggerFactory.INFO_LEVEL
    sLogLevel = INFO_LEVEL_TEXT
  Case LoggerFactory.WARN_LEVEL
    sLogLevel = WARN_LEVEL_TEXT
  Case LoggerFactory.FATAL_LEVEL
```

```
    sLogLevel = FATAL_LEVEL_TEXT
  Case LoggerFactory.EVER_LEVEL
    sLogLevel = EVER_LEVEL_TEXT
  Case Else
    sLogLevel = "!INVALID LOG LEVEL!"
  End Select
  sDateTime = CStr(Now())
  LogMessage = sLogLevel & " " & Environ("Userdomain") & "\" & Environ("Username") & " " & _
    sDateTime & " [" & Me.SubName & "] - " & sLogText
  #If Not Logging_cashed Then
    FileNum = FreeFile
    Open Me.LogFilePath For Append As #FileNum
    Print #FileNum, LogMessage
    Close #FileNum
  #End If
  #If Logging_on_Screen Then
    wsW.Cells(iThisLogRow, 5) = LogMessage
    iThisLogRow = iThisLogRow + 1
  #End If
End Sub

Private Sub Class_Initialize()
  #If Logging_cashed And Not Logging_on_Screen Then
    Err.Raise Number:=vbObjectError + 513, Description:="Logging_cashed requires Logging_on_Screen"
  #End If
End Sub

Private Sub Class_Terminate()
  #If Logging_cashed Then
    Dim i As Long, FileNum As Integer, LogMessage As String
    FileNum = FreeFile
    Open Me.LogFilePath For Append As #FileNum
    For i = 3 To iThisLogRow - 1
      LogMessage = wsW.Cells(i, 5).Text
      Print #FileNum, LogMessage
    Next i
    Close #FileNum
  #End If
End Sub
```

Exkurs: Logger für PowerShell – *Write-Log*

PowerShell ist eine Microsoft Windows basierte objektorientierte Script- oder Programmiersprache. Sie hat eine eigene Befehlszeilenschnittstelle. Eine nutzbare Log-Funktion:

```
<#
.SYNOPSIS
    Write-Log -Message [string] -Severity EVER|INFO|WARN|FATAL

.DESCRIPTION
    This script/function enables logging within PowerShell scripts.

    Global variables which control features of this script are:
    [string]$global:LogFilePath    - The text file into which all messages are getting written
    [string]$global:SubName        - The name of the script calling this Log script
    [int]$global:LogLevel          - The log levels:
                                                                                                1 logs
everything,
                                                                                                2 omits
INFOs,
                                     3 omits INFOs and WARNings,
                                                                                                4 only
logs severity EVER messages
    [bool]$global:WriteToHost      - $FALSE = Only write into logfile; $TRUE = Also write to Host

.OUTPUTS
           Write-Log writes into a logfile with name [string]$global:LogFilePath.
           if the user has not given this, the standard filename "$PSScriptRoot\Logs\LogFile.txt" is used.
    The contents of a log file will look like:
    INFO: BERND-PC\Bernd 2020-02-16 12:19:36 [zip_and_copy.ps1] - Log started
    INFO: BERND-PC\Bernd 2020-02-16 12:19:36 [zip_and_copy.ps1] - Log finished

           In case the user has set [bool]$global:WriteToHost to $TRUE, Write-Log also repeats all log
           messages on screen (Host).

.PARAMETER Message
    The message you want to log.

.PARAMETER Severity
    The severity of the message. This can be 'INFO', 'WARN', 'FATAL', or 'EVER'.
    If global variable LogLevel is 1 then messages with all severities will be logged.
    If LogLevel is 2 then INFO messages will not be logged.
```

```
    If LogLevel is 3 then INFO and WARN messages will not be logged.
    If LogLevel is 4 then INFO, WARN and FATAL messages will not be logged.

.EXAMPLE
    # Initialize global logging variables and make log script known
    $global:SubName = $MyInvocation.MyCommand.Name
    $global:LogLevel = 1
    $global:LogFilePath = "$PSScriptRoot\Logs\LogFile_" + $SubName + "_" +
        (Get-Date -UFormat "%Y%m%d") + ".txt"
    $global:WriteToHost = $TRUE
    . "$PSScriptRoot\Write-Log.ps1"

.EXAMPLE
    # How to call
    Write-Log -Message 'Log started' -Severity INFO
    If(-Not (Test-path $ArchiveFile)) {Write-Log -Message "$ArchiveFile was not created" -Severity FATAL}

.NOTES
    Source (EN): http://www.sulprobil.de/write-log_en/
    Source (DE): http://www.berndplumhoff.de/write-log_de/
    Version Date        Who                 What
    1.0     16-Feb-2020 Bernd Plumhoff      Initial Version
    1.1     13-Jul-2020 Bernd Plumhoff      EVER instead of ALWAYS, and optional Write-Host
#>

#Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope CurrentUser -Force
#Clear
#Requires -Version 4
Set-StrictMode -Version Latest

function Write-Log {
    [CmdletBinding()]
    param(
        [Parameter()]
        [ValidateNotNullOrEmpty()]
        [string]$Message,
        [Parameter()]
        [ValidateNotNullOrEmpty()]
        [ValidateSet('INFO','WARN','FATAL','EVER')]
        [string]$Severity = 'INFO'
    )
    switch ($Severity)
    {
        'INFO'        { If ([int]$global:LogLevel -gt 1) {Exit} }
        'WARN'        { If ([int]$global:LogLevel -gt 2) {Exit} }
        'FATAL'       { If ([int]$global:LogLevel -gt 3) {Exit} }
        'EVER'        { <# Will always be logged! #> }
    }
    if (-Not (Test-Path -IsValid $global:LogFilePath)) { $global:LogFilePath =
"$PSScriptRoot\Logs\LogFile.txt" }
    [string]$Output = $Severity + ": " + [System.Security.Principal.WindowsIdentity]::GetCurrent().Name +
        " " + (Get-Date -UFormat "%Y-%m-%d %T") + " [" + $global:SubName + "] - " + $Message
    Write-Output $Output | Out-File -append $global:LogFilePath
           If (Test-Path variable:global:WriteToHost) {
                   If ($global:WriteToHost) { Write-Host $Output }
           }
}
```

Bereichsnamen anzeigen – sbNamedRanges

Es ist gute Programmierpraxis, wichtigen Zellen oder Zellbereichen Namen zuzuweisen. Bereichsnamen können auch Formeln enthalten. Sie können für einzelne Tabellenblätter oder die gesamte Arbeitsmappe gelten. Wenn Sie alle Bereichsnamen auflisten wollen, zum Beispiel:

Bereichsnamen		
Name	Gültig in	Bezug
A_global_Name	Arbeitsmappe	=HEUTE()
Deutsch_Bereich	Deutsch	=Deutsch!$A$1
English_Named_Range	English	=English!$A$1

sbNamedRanges Programmcode

```
Function sbNamedRanges(Optional bRefersToLocal As Boolean = True) As String()
'List all named ranges.
'(C) (P) by Bernd Plumhoff 01-Feb-2025 PB V3.1
Dim i           As Long
Dim s           As String
Dim wb          As Workbook

Set wb = ThisWorkbook
ReDim sR(1 To wb.Names.Count, 1 To 3) As String
For i = 1 To wb.Names.Count
  If TypeOf wb.Names(i).Parent Is Worksheet Then
    s = wb.Names(i).Parent.Name
    sR(i, 1) = Mid(wb.Names(i).Name, Len(s) + 2, 2048)
    sR(i, 2) = s
  Else
    sR(i, 1) = wb.Names(i).Name
    If bRefersToLocal Then
      sR(i, 2) = "Arbeitsmappe"
    Else
      sR(i, 2) = "Workbook"
    End If
  End If
  If bRefersToLocal Then
    sR(i, 3) = wb.Names(i).RefersToLocal
  Else
    sR(i, 3) = wb.Names(i).RefersTo
  End If
Next i
sbNamedRanges = sR
End Function
```

Excel Version anzeigen – *ApplicationVersion*

Microsoft beschloss, mit Excel 2016 den Wert der Funktion *Application.Version* nicht mehr über 16 hinaus zu erhöhen. Diese benutzerdefinierte Funktion *ApplicationVersion* korrigiert dies.

ApplicationVersion Programmcode

```
Function ApplicationVersion(Optional bShowBuild365 As Boolean = True) As String
'Returns MS Excel's version - with a little kludge
'(C) (P) by Bernd Plumhoff 20-Oct-2024 PB V0.61
Dim n As Integer
With Application
n = Val(.Version)
Select Case n
Case 16
  ApplicationVersion = "Excel 2016"
  On Error Resume Next 'We know what we are doing
  'Excel 365 and 2024 (LTSC) introduced ValueToText
  n = Val(.ValueToText(19))
  If n = 19 Then
    If .Build = "17932" Then
      ApplicationVersion = "Excel 2024"
    Else
      If bShowBuild365 Then
        'When all of them are 365 you might want to know the build.
        ApplicationVersion = "Excel 365 (Build " & .Build & ")"
      Else
        ApplicationVersion = "Excel 365"
      End If
    End If
  Else
    'Excel 2021 (LTSC) introduced RandArray
    n = .RandArray(1, 1, 18, 18, True)(1)
    If n = 18 Then
      ApplicationVersion = "Excel 2021"
    Else
      'Excel 2019 introduced TextJoin
      n = Val(.TextJoin(" ", True, "17"))
      If n = 17 Then ApplicationVersion = "Excel 2019"
    End If
  End If
  On Error GoTo 0
Case 15
  ApplicationVersion = "Excel 2013"
Case 14
  ApplicationVersion = "Excel 2010"
Case 12
  ApplicationVersion = "Excel 2007"
Case 11
  ApplicationVersion = "Excel 2003"
Case 10
  ApplicationVersion = "Excel 2002"
Case 9
  ApplicationVersion = "Excel 2000"
Case 8
  ApplicationVersion = "Excel 97"
Case 7
  ApplicationVersion = "Excel 7/95"
Case 5
  ApplicationVersion = "Excel 5"
Case Else
  ApplicationVersion = "[Error]"
End Select
End With
End Function
```

Anzahl der Dimensionen eines Arrays – *ArrayDim*

Wie ermitteln Sie die Anzahl der Dimensionen eines Arrays?

ArrayDim Programmcode

```
Function ArrayDim (v As Variant) As Long
'Returns number of dimensions of an array or 0 for
'an undimensioned array or -1 if no array at all.
'(C) (P) by Bernd Plumhoff 10-May-2010 PB V0.1
Dim i As Long
ArrayDim = -1
If Not IsArray(v) Then Exit Function
On Error Resume Next
'Err.Clear 'Not necessary
Do While IsNumeric(UBound(v, i + 1))
  If Err.Number <> 0 Then Exit Do
  i = i + 1
Loop
ArrayDim = i
End Function
```

Zellinformationen ausgeben – *sbGetCell*

In älteren Excel Versionen konnte man mit dem Excel4 Makro *ZELLE.ZUORDNEN* Zellinformationen ausgeben. Zum Beispiel konnte man den Namen HatFormel mit dem Wert

```
=ZELLE.ZUORDNEN(48;INDIREKT("ZS(-1)";FALSCH))
```

im Namensmanager definieren. Wenn Sie dann =HatFormel in der Zelle rechts neben einer gewünschten Zelle eingegeben hätten, dann würde diese anzeigen, ob die gewünschte Zelle eine Formel enthält (“WAHR”) oder nicht (“FALSCH”).

Sie können mit VBA ähnliche Informationen ausgeben lassen:

sbGetCell Programmcode

```
Function sbGetCell(r As Range, s As String) As Variant
'Bernd Plumhoff V0.33 30-Oct-2022
With Application.WorksheetFunction
Application.Volatile
Select Case s
Case "AbsReference", "1"
    'Absolute style reference like $A$1
    If Application.Caller.Parent.Parent.Name = r.Worksheet.Parent.Name And _
        Application.Caller.Parent.Name = r.Worksheet.Name Then
        sbGetCell = r.Address
    Else
        If InStr(r.Worksheet.Parent.Name & r.Worksheet.Name, " ") > 0 Then
            sbGetCell = "'[" & r.Worksheet.Parent.Name & "]" & r.Worksheet.Name & "'!" & r.Address
        Else
            sbGetCell = "[" & r.Worksheet.Parent.Name & "]" & r.Worksheet.Name & "!" & r.Address
        End If
    End If
Case "RowNumber", "2"
    'Row number in the top cell reference
    sbGetCell = r.Row
Case "ColumnNumber", "3"
    'Column number of the leftmost cell in reference
    sbGetCell = r.Column
Case "Type", "4"
    'Same as TYPE(reference)
    sbGetCell = -IsEmpty(r) - .IsNumber(r) - .IsText(r) * 2 - .IsLogical(r) _
                * 4 - .IsError(r) * 16 - IsArray(r) * 64
Case "Contents", "5"
    'Contents of reference
    sbGetCell = r.Value
Case "FormulaLocal", "ShowFormula", "6"
    'Cell formula
    sbGetCell = r.FormulaLocal
Case "NumberFormat", "7"
    'Number format of cell
    sbGetCell = r.NumberFormatLocal
Case "HorizontalAlignment", "8"
    'Number indicating the cell's horizontal alignment
    Select Case r.HorizontalAlignment
    Case xlGeneral
        sbGetCell = 1
    Case xlLeft
        sbGetCell = 2
    Case xlCenter
        sbGetCell = 3
    Case xlRight
        sbGetCell = 4
    Case xlFill
        sbGetCell = 5
    Case xlJustify
        sbGetCell = 6
    Case xlCenterAcrossSelection
        sbGetCell = 7
    Case xlDistributed
        sbGetCell = 8
    Case Else
        Debug.Assert False 'Should not get here
    End Select
Case "LeftBorderStyle", "9"
    'Number indicating the left-border style assigned to the cell
    Select Case r.Borders(1).LineStyle
    Case xlLineStyleNone
```

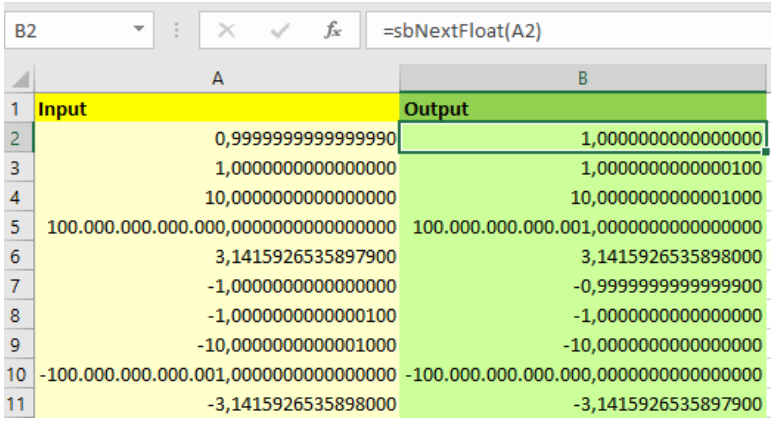
```
            sbGetCell = 0
        Case xlHairline
            sbGetCell = IIf(r.Borders(1).Weight = xlMedium, 2, 7)
        Case xlDot
            sbGetCell = 4
        Case xlDashDotDot
            sbGetCell = IIf(r.Borders(1).Weight = xlMedium, 12, 11)
        Case xlDashDot
            sbGetCell = IIf(r.Borders(1).Weight = xlMedium, 10, 9)
        Case xlDash
            sbGetCell = IIf(r.Borders(1).Weight = xlMedium, 8, 3)
        Case xlSlantDashDot
            sbGetCell = 13
        Case xlDouble
            sbGetCell = 6
        Case Else
            sbGetCell = CVErr(xlErrValue)
        End Select
    Case "RightBorderStyle", "10"
        'Number indicating the right-border style assigned to the cell
        Select Case r.Borders(2).LineStyle
        Case xlLineStyleNone
            sbGetCell = 0
        Case xlHairline
            sbGetCell = IIf(r.Borders(2).Weight = xlMedium, 2, 7)
        Case xlDot
            sbGetCell = 4
        Case xlDashDotDot
            sbGetCell = IIf(r.Borders(2).Weight = xlMedium, 12, 11)
        Case xlDashDot
            sbGetCell = IIf(r.Borders(2).Weight = xlMedium, 10, 9)
        Case xlDash
            sbGetCell = IIf(r.Borders(2).Weight = xlMedium, 8, 3)
        Case xlSlantDashDot
            sbGetCell = 13
        Case xlDouble
            sbGetCell = 6
        Case Else
            sbGetCell = CVErr(xlErrValue)
        End Select
    Case "TopBorderStyle", "11"
        'Number indicating the top-border style assigned to the cell
        Select Case r.Borders(3).LineStyle
        Case xlLineStyleNone
            sbGetCell = 0
        Case xlHairline
            sbGetCell = IIf(r.Borders(3).Weight = xlMedium, 2, 7)
        Case xlDot
            sbGetCell = 4
        Case xlDashDotDot
            sbGetCell = IIf(r.Borders(3).Weight = xlMedium, 12, 11)
        Case xlDashDot
            sbGetCell = IIf(r.Borders(3).Weight = xlMedium, 10, 9)
        Case xlDash
            sbGetCell = IIf(r.Borders(3).Weight = xlMedium, 8, 3)
        Case xlSlantDashDot
            sbGetCell = 13
        Case xlDouble
            sbGetCell = 6
        Case Else
            sbGetCell = CVErr(xlErrValue)
        End Select
    Case "BottomBorderStyle", "12"
        'Number indicating the bottom-border style assigned to the cell
        Select Case r.Borders(4).LineStyle
        Case xlLineStyleNone
            sbGetCell = 0
        Case xlHairline
            sbGetCell = IIf(r.Borders(4).Weight = xlMedium, 2, 7)
        Case xlDot
            sbGetCell = 4
        Case xlDashDotDot
            sbGetCell = IIf(r.Borders(4).Weight = xlMedium, 12, 11)
        Case xlDashDot
            sbGetCell = IIf(r.Borders(4).Weight = xlMedium, 10, 9)
        Case xlDash
            sbGetCell = IIf(r.Borders(4).Weight = xlMedium, 8, 3)
        Case xlSlantDashDot
            sbGetCell = 13
        Case xlDouble
            sbGetCell = 6
        Case Else
            sbGetCell = CVErr(xlErrValue)
        End Select
    Case "Pattern", "13"
        'Number indicating cell pattern
        sbGetCell = r.Interior.Pattern
    Case "IsLocked", "14"
        'True if cell is locked
        sbGetCell = r.Locked
```

```
Case "FormulaHidden", "HiddenFormula", "15"
    'True if cell formula is hidden
    sbGetCell = r.FormulaHidden
Case "Width", "CellWidth", "16"
    'Cell width. If array-entered into two cells of a row,
    'second value is true if width is standard
    sbGetCell = Array(r.ColumnWidth, r.UseStandardWidth) 'Not width!
Case "Height", "RowHeight", "17"
    'Cell height
    sbGetCell = r.RowHeight
Case "FontName", "18"
    'Cell font name
    sbGetCell = r.Font.Name
Case "FontSize", "19"
    'Cell font size
    sbGetCell = r.Font.Size
Case "IsBold", "20"
    'Cell is formatted bold?
    sbGetCell = r.Font.Bold
Case "IsItalic", "21"
    'Cell is formatted in Italics?
    sbGetCell = r.Font.Italic
Case "IsUnderlined", "22"
    'Cell is formatted as underlined?
    sbGetCell = (r.Font.Underline = xlUnderlineStyleSingle Or _
                 r.Font.Underline = xlUnderlineStyleSingleAccounting Or _
                 r.Font.Underline = xlUnderlineStyleDouble Or _
                 r.Font.Underline = xlUnderlineStyleDoubleAccounting)
Case "IsStruckThrough", "23"
    'Cell characters are struck through?
    sbGetCell = r.Font.Strikethrough
Case "FontColorIndex", "24"
    'Cell font color of first character, 1-56, 0 = automatic
    sbGetCell = r.Font.ColorIndex
Case "IsOutlined", "25", "IsShaddowed", "26"
    'Cell font is outlined or shaddowed? (Not supported by Excel)
    sbGetCell = False
Case "PageBreak", "27"
    '0 = no break, 1 = row, 2 = column, 3 = row and column
    sbGetCell = -(r.EntireRow.PageBreak <> xlPageBreakNone) - 2 * (r.EntireColumn.PageBreak <>
xlPageBreakNone)
Case "RowLevelOutline", "28"
    'Row level outline
    sbGetCell = r.EntireRow.OutlineLevel
Case "ColumnLevelOutline", "29"
    'Row level outline
    sbGetCell = r.EntireColumn.OutlineLevel
Case "IsSummaryRow", "30"
    'Row is a summary row?
    sbGetCell = r.EntireRow.Summary
Case "IsSummaryColumn", "31"
    'Column is a summary column?
    sbGetCell = r.EntireColumn.Summary
Case "WorkbookSheetName", "32"
    'Workbook name like [Book1.xls]Sheet1 or Book1.xls if
    'workbook and single sheet have
    'identical names
    If r.Worksheet.Parent.Name = r.Worksheet.Name & ".xls" And _
        Application.Worksheets.Count = 1 Then
        sbGetCell = r.Worksheet.Parent.Name
    Else
        sbGetCell = "[" & r.Worksheet.Parent.Name & "]" & _
        r.Worksheet.Name
    End If
Case "IsWrapped", "33"
    'Cell text is formatted as wrapped?
    sbGetCell = r.WrapText
Case "LeftBorderColorIndex", "34"
    'Left border color index
    sbGetCell = r.Borders.Item(1).ColorIndex
Case "RightBorderColorIndex", "35"
    'Right border color index
    sbGetCell = r.Borders.Item(2).ColorIndex
Case "TopBorderColorIndex", "36"
    'Top border color index
    sbGetCell = r.Borders.Item(3).ColorIndex
Case "BottomBorderColorIndex", "37"
    'Bottom border color index
    sbGetCell = r.Borders.Item(4).ColorIndex
Case "ShadeForeGroundColor", "38", "PatternBackGroundColor", "64"
    'ShadeForeGroundColor
    sbGetCell = r.Interior.PatternColorIndex
Case "ShadeBackGroundColor", "39", "PatternForeGroundColor", "63"
    'ShadeBackGroundColor
    sbGetCell = r.Interior.ColorIndex
Case "TextStyle", "40"
    'Style of the cell, as text
    sbGetCell = r.Style.NameLocal
Case "FormulaWOT", "41"
    'Returns the formula in the active cell without translating it (useful for international macro sheets)
```

```
    sbGetCell = r.Formula
'Case "HDistWinToLCell", "42"
'    'Horizontal distance, measured in points, from the left edge of the active window to the left edge of
the cell
'    sbGetCell = r. 'Does not work yet
Case "HasNote", "46"
    'True if cell contains a text note
    sbGetCell = Len(r.NoteText) > 0
Case "HasSound", "47"
    'True if cell has a sound note. Not supported.
    sbGetCell = False
Case "HasFormula", "48"
    'True if cell contains a formula
    sbGetCell = r.HasFormula
Case "IsArray", "49"
    'True if cell is part of an array formula
    sbGetCell = r.HasArray
Case "VerticalAlignment", "50"
    '1 = Top, 2 = Center, 3 = Bottom, 4 = Justified, 5 = Distributed
    sbGetCell = -(r.VerticalAlignment = xlVAlignTop) - 2 * (r.VerticalAlignment = xlVAlignCenter) _
                - 3 * (r.VerticalAlignment = xlVAlignBottom) - 4 * (r.VerticalAlignment = _
                xlVAlignJustify) - 5 * (r.VerticalAlignment = xlVAlignDistributed)
Case "VerticalOrientation", "51"
    '0 = Horizontal, 1 = Vertical, 2 = Upward, 3 = Downward
    sbGetCell = -(r.Orientation = xlVertical) - 2 * (r.Orientation = xlUpward) - _
                3 * (r.Orientation = xlDownward)
Case "IsStringConst", "IsStringConstant", "52"
    'Text alignment char "'" if cell is a string constant,
    'empty string "" if not
    sbGetCell = r.PrefixCharacter
Case "AsText", "53"
    'Cell displayed as text with numbers formatted and symbols included
    sbGetCell = r.Text
Case "PivotTableViewName", "54"
    'PivotTableViewName
    sbGetCell = r.PivotTable.Name
'Case "PivotTableViewPosition", "55"
'    'PivotTableViewPosition
'    sbGetCell = r.PivotField.Position 'Not correct yet
Case "PivotTableViewFieldName", "56"
    'PivotTableViewFieldName
    sbGetCell = r.PivotField.Name
Case "IsSuperscript", "57"
    'Cell text is formatted as superscript?
    sbGetCell = r.Font.Superscript
Case "FontStyleText", "58"
    'FontStyleText
    sbGetCell = r.Font.FontStyle
Case "UnderlineStyle", "59"
    'Underline style, 1 = none, 2 = single, 3 = double, 4 = single accounting, 5 = double accounting
    Select Case r.Font.Underline
    Case xlUnderlineStyleNone
        sbGetCell = 1
    Case xlUnderlineStyleSingle
        sbGetCell = 2
    Case xlUnderlineStyleDouble
        sbGetCell = 3
    Case xlUnderlineStyleSingleAccounting
        sbGetCell = 4
    Case xlUnderlineStyleDoubleAccounting
        sbGetCell = 5
    Case Else
        sbGetCell = CVErr(xlErrValue)
    End Select
Case "IsSubscript", "60"
    'Cell text is formatted as subscript?
    sbGetCell = r.Font.Subscript
Case "PivotTableItemName", "61"
    'PivotTableItemName
    sbGetCell = r.PivotItem.Name
Case "WorksheetName", "62"
    'Worksheet name like [Book1.xls]Sheet1
        sbGetCell = "[" & r.Worksheet.Parent.Name & "]" & _
                    r.Worksheet.Name
Case "IsAddIndentAlignment", "65"
    'Only Far East Excel Versions
    sbGetCell = False 'Not supported here
Case "WorkbookName", "66"
    'Workbook name like Book1.xls
    sbGetCell = r.Worksheet.Parent.Name
Case "IsHidden"
    'Cell hidden?
    sbGetCell = r.EntireRow.Hidden Or r.EntireColumn.Hidden
Case Else
    sbGetCell = CVErr(xlErrValue)
End Select
End With
End Function
```

Nächste Gleitkommazahl – *sbNextFloat*

Wenn Sie zu einer Gleitkommazahl die nächstgelegene ermitteln wollen:



	A	B
1	**Input**	**Output**
2	0,9999999999999990	1,0000000000000000
3	1,0000000000000000	1,0000000000000100
4	10,0000000000000000	10,0000000000001000
5	100.000.000.000.000,0000000000000000	100.000.000.000.001,0000000000000000
6	3,1415926535897900	3,1415926535898000
7	-1,0000000000000000	-0,9999999999999900
8	-1,0000000000000100	-1,0000000000000000
9	-10,0000000000001000	-10,0000000000000000
10	-100.000.000.000.001,0000000000000000	-100.000.000.000.000,0000000000000000
11	-3,1415926535898000	-3,1415926535897900

sbNextFloat Programmcode

```
Function sbNextFloat(d As Double, Optional bUp As Boolean = True) As Double
'Returns the smallest double which is greater than the input (bUp = True)
'or the greatest double which is smaller than the input value (bUp = False).
'(C) (P) by Bernd Plumhoff 03-Oct-2010 PB V0.11
sbNextFloat = d - (2# * bUp + 1#) * CDbl("1e" & Right(Format(d, "." & _
              String(15, "0") & "E+000"), 4) - 15)
End Function
```

Aufruf anderer Windows Programme am Beispiel *sbZip*

Via VBA kann man andere Windows Programme aufrufen. Beispiel: Wenn Sie eine Datei oder ein Dateiverzeichnis (auch einschließlich aller Unterverzeichnisse) komprimieren wollen, können Sie dies direkt mit VBA erledigen oder das frei erhältliche Programm 7zip verwenden.

Das VBA Programm sbZip benutzt den hier vorgestellten Logger (siehe Programmablauf dokumentieren – *Logging* Klasse), um die Standardausgabe und Standardfehlerausgabe von 7zip zu protokollieren und LibFileTools.

sbZip – Programmcode

```
Public Const AppVersion As String = "sbZip_Version_9"

Sub sbZip(ByVal vSourceFullPathName As Variant, _
  ByVal vDestinationZipFullPathName As Variant, _
  Optional bCreate As Boolean = True, _
  Optional bUse7zip As Boolean = True)
'Create zip file vDestinationZipFullPathName and insert zipped file or folder vSourceFullPathName.
'Version When         Who    What
'      1 24-Nov-2020  EotG   Original downloaded from https://exceloffthegrid.com/vba-cod-to-zip-unzip/
'      6 17-Dec-2020  Bernd  ByVal to enforce variants, single file feature and parameter bCreate added
'      7 25-Apr-2024  Bernd  lRepeat to avoid endless loops and parameter 16 for CopyHere to avoid
'                            confirmation prompt. No error checking.
'      8 12-Sep-2024  Bernd  Use a valid empty zip template if it exists.
'                            Workaround in case the print sequence fails.
'      9 29-Dec-2024  Bernd  New option bUse7zip and using Logger and LibFileTools
'                            https://github.com/cristianbuse/VBA-FileTools.
Dim iFile             As Integer
Dim lItems            As Long
Dim lRepeat           As Long
Dim sLine             As String
Dim sShellCmd         As String
```

```
Dim oExec             As Object
Dim oOutput           As Object
Dim oShell            As Object

If GLogger Is Nothing Then Call Start_Log
If bCreate And Not IsFile(CStr(vDestinationZipFullPathName)) Then
  If Not DeleteFile(CStr(vDestinationZipFullPathName)) Then
    GLogger.warn "Could not delete file '" & vDestinationZipFullPathName & "'"
  End If
End If
If bUse7zip Then
  If IsFile("C:\Program Files\7-Zip\7z.exe") Then
    Set oShell = CreateObject("WScript.Shell")
    sShellCmd = "C:\Program Files\7-Zip\7z.exe a """ & vDestinationZipFullPathName & _
      """ """ & vSourceFullPathName & """"
    Set oExec = oShell.exec(sShellCmd)
    Set oOutput = oExec.StdOut
    Do While Not oOutput.AtEndOfStream
      sLine = oOutput.ReadLine
      If sLine <> "" Then GLogger.info "STDOUT " & sLine
    Loop
    Set oOutput = oExec.StdErr
    Do While Not oOutput.AtEndOfStream
      sLine = oOutput.ReadLine
      If sLine <> "" Then GLogger.warn "STDERR " & sLine
    Loop
    Do While oExec.Status = 0
      Application.Wait (Now + TimeValue("0:00:01"))
    Loop
    GLogger.info vSourceFullPathName & "' zipped into '" & vDestinationZipFullPathName & "'"
  Else
    GLogger.fatal "C:\Program Files\7-Zip\7z.exe doesn't exist. Cannot zip '" & _
      vSourceFullPathName & "'"
  End If
Else
  If IsFile(GetLocalPath(ThisWorkbook.Path) & "Zip_Template.zip") Then
    'Workaround in case print sequence in Else clause does not work
    CopyFile ThisWorkbook.Path & "\Zip_Template.zip", CStr(vDestinationZipFullPathName)
    If Not IsFile(CStr(vDestinationZipFullPathName)) Then
      GLogger.warn "Could not copy template file '" & vDestinationZipFullPathName & "'"
    End If
  Else
    iFile = FreeFile
    Open vDestinationZipFullPathName For Output As #iFile
    Print #iFile, Chr$(80) & Chr$(75) & Chr$(5) & Chr$(6) & String(18, 0)
    Close #iFile
  End If
  On Error Resume Next
  lItems = oShell.Namespace(vDestinationZipFullPathName).Items.Count
  On Error GoTo 0

  Set oShell = CreateObject("Shell.Application")
  If GetAttr(vSourceFullPathName) = vbDirectory Then
    oShell.Namespace(vDestinationZipFullPathName).CopyHere _
      oShell.Namespace(vSourceFullPathName).Items, 16
    lRepeat = 0
    On Error Resume Next
    Do Until oShell.Namespace(vDestinationZipFullPathName).Items.Count = _
      lItems + oShell.Namespace(vSourceFullPathName).Items.Count Or lRepeat > 5
      Application.Wait (Now + TimeValue("0:00:01"))
    lRepeat = lRepeat + 1
    Loop
    On Error GoTo 0
  Else
    oShell.Namespace(vDestinationZipFullPathName).CopyHere vSourceFullPathName, 16
    lRepeat = 0
    On Error Resume Next
    Do Until oShell.Namespace(vDestinationZipFullPathName).Items.Count = _
      lItems + 1 Or lRepeat > 3
      Application.Wait (Now + TimeValue("0:00:01"))
    lRepeat = lRepeat + 1
    Loop
    On Error GoTo 0
  End If
End If
End Sub
```

Excel Don’ts – Was man mit Excel besser sein lässt

“Wenn ich mein Leben noch einmal leben könnte, würde ich die gleichen Fehler machen. Aber ein bisschen früher, damit ich mehr davon habe.” [Marlene Dietrich]

Kritik sollte konstruktiv sein. Im Allgemeinen versuche ich positiv zu zeigen, was man wie am besten machen könnte. Aber in mehr als 35 Jahren Excel Praxis sah ich einige Beispiele, die besser hätten vermieden werden sollen.

Eine Tabelle mit Beispielen die man besser vermeidet

Bitte sehen Sie diese Liste als intellektuelle Herausforderung an und nehmen Sie sie *cum grano salis*.

#	Bitte vermeiden	Warum?
1	Excel Add-in Freeware wie z. B. Morefunc	Sie sollten für ein Add-in immer eine Lizenz haben oder wenigstens den Quellcode, damit Sie sich selbst helfen können, wenn der Ersteller verschwindet. Andernfalls besteht auch ein Sicherheitsrisiko. Bei Morefunc ist der Entwickler offensichtlich verschwunden, es ist kein Quellcode verfügbar, viele Funktionen laufen nicht mehr korrekt seit Excel 2007. Ich denke, es ist unverantwortlich, die Nutzung noch zu empfehlen.
2	Tabellenblattfunktionen die Sie nicht verstehen	Wie können Sie sicher sein, dass sie korrekt ist? Nach Zeilen- oder Spalten-Einfügungen oder -Löschungen? Wie können Sie sie an künftige Anforderungen anpassen? Bei einer benutzerdefinierten VBA Funktion müssen Sie lediglich deren Funktion und Parameter kennen. Dies nennt man Datenkapselung.
3	Option *Genauigkeit wie angezeigt* in den erweiterten Excel Optionen	Die Änderung des Anzeigeformats einer Eingabe- oder Zwischenberechnungs-Zelle kann Ihre Ergebnisse ruinieren. Für immer.
4	Option *Iterative Berechnung aktivieren* in den Excel Formeln Optionen	Sie können keine unbeabsichtigten Zirkelbezüge mehr erkennen. Manche Dummköpfe nutzen diese Option auch absichtlich, um Zirkelbezüge zu verbergen.
5	Option *1904-Datumswerte verwenden* in den erweiterten Excel Optionen, um negative Zeiten anzeigen zu können	Fast alle verwenden die standardmäßigen 1900-Datumswerte. Falls Sie davon abweichen wollen, merken Sie sich schon einmal den Wert 1462, um ihn zu addieren oder zu subtrahieren, wenn Sie zwischen den Datumssystemen wechseln müssen.
6	Testen Sie Gleitkommazahlen nie mit = auf Gleichheit. Verwenden Sie stattdessen etwas wie ABS(a - b) < 1E-13	=(43.1-43.2)+1=0.9 ergibt nicht WAHR wie Sie vielleicht vermuten. Siehe Microsoft’s Artikel unter https://learn.microsoft.com/en-us/office/troubleshoot/excel/floating-point-arithmetic-inaccurate-result. Die beste Erklärung liefert m. E. Goldberg’s Artikel https://www.itu.dk/~sestoft/bachelor/IEEE754_article.pdf
7	Bestimmte Zellen mit SUMMENPRODUKT bedingt summieren oder zählen und für alle Bedingungen auflisten	SUMMENPRODUKT ist eine mächtige Funktion, aber für eine Zahl- oder Summenstatistik für alle Ausprägungen einer Liste sind Pivot Tabellen oder PowerQuery besser geeignet.
8	Tabellenblattfunktion INDIREKT	Diese Funktion ist volatil, d.h. sie wird jedes Mal neu berechnet, wenn Sie F9 drücken. Wenn Sie INDIREKT verwenden, haben Sie Ihre Excel Tabelle völlig falsch aufgebaut.
9	Tabellenblattfunktion AGGREGAT	Wenn Sie ausgeblendete Zeilen oder Fehlerwerte in Ihrer Berechnung ignorieren, wird kein Revisor Ihre Excel Tabelle akzeptieren können.

Zahlensysteme, Formate und Umwandlungen

Abstract

Als Programmierer hat man häufig mit Zahlensystemen und deren Darstellung bzw. Umwandlung zu tun. Hier stelle ich einige Programme vor, die ich im Laufe der Zeit kennen- und nutzen lernte.

Umwandlungen und Berechnungen von Zahlen

Zahlen in Worten ausgeben – *sbInWorten*

Manchmal müssen Sie Zahlen in Worten ausgeben, z. B. in Euro/Cent oder in Dollars/Cents oder in britischen Pfund Sterling/Pence. 12,31 würde zum Beispiel als “Zwölf Euro und Einunddreißig Cent” ausgegeben werden.

Hinweis: Im Web existiert eine Vielzahl von *SpellNumber*- und *InWorten*-Versionen. Leider sind die meisten davon fehlerhaft. Testen Sie am besten mit den hier genannten Beispielzahlen:

	A	B	C
1	**Spell numbers:**		
2			
3	**Number**	**Spell Number**	**In Worten**
4	1.000.000.000.000.000,00	>>>>> Error (Absolute amount > 999999999999999)! <<<<<	>>>>> Fehler (Absolutbetrag > 999999999999999)! <<<<<
5	0,123	Zero Dollars and Twelve Cents (rounded)	Null Euro und Zwölf Cent (gerundet)
6	-1,00	Minus One Dollar and Zero Cents	Minus Ein Euro und Null Cent
7	20,123	Twenty Dollars and Twelve Cents (rounded)	Zwanzig Euro und Zwölf Cent (gerundet)
8	-20,123	Minus Twenty Dollars and Twelve Cents (rounded)	Minus Zwanzig Euro und Zwölf Cent (gerundet)
9	1,01	One Dollar and One Cent	Ein Euro und Ein Cent
10	1.000.001,01	One Million One Dollars and One Cent	Eine Million und Ein Euro und Ein Cent

sbInWorten / sbSpellNumber Programmcode

```
Private sNWord(0 To 28) As String
Private sHWord(1 To 4) As String

Function sbInWorten(ByVal sNumber As String) As String
   sbInWorten = sbSpellNumber(sNumber, "German", "EUR")
End Function

Function sbSpellNumber(ByVal sNumber As String, _
          Optional sLang As String = "English", _
          Optional sCcy As String = "USD") As String
'Template was Microsoft's limited version:
'https://support.microsoft.com/de-de/help/213360/
'how-to-convert-a-numeric-value-into-english-words-in-excel
'This version informs the user about its limits.
'(C) (P) by Bernd Plumhoff  02-Mar-2018 PB V1.0

Dim Euros As String, cents As String
Dim Result As String, Temp As String
Dim DecimalPlace As Integer, Count As Integer
Dim Place(1 To 6) As String
Dim dNumber As Double
Dim prefix As String, suffix As String

Select Case sLang
Case "English"
   Place(1) = ""
   Place(2) = " Thousand "
   Place(3) = " Million "
   Place(4) = " Billion "
   Place(5) = " Trillion "
```

```
    Place(6) = " Mantissa not wide enough for this number "
    sHWord(1) = ">>>>> Error (Absolute amount > 999999999999999)! <<<<<"
    sHWord(2) = " (rounded)"
    sHWord(3) = "Minus "
    sHWord(4) = "and"
    sNWord(0) = "zero"
    sNWord(1) = "one"
    sNWord(2) = "two"
    sNWord(3) = "three"
    sNWord(4) = "four"
    sNWord(5) = "five"
    sNWord(6) = "six"
    sNWord(7) = "seven"
    sNWord(8) = "eight"
    sNWord(9) = "nine"
    sNWord(10) = "ten"
    sNWord(11) = "eleven"
    sNWord(12) = "twelve"
    sNWord(13) = "thirteen"
    sNWord(14) = "fourteen"
    sNWord(15) = "fifteen"
    sNWord(16) = "sixteen"
    sNWord(17) = "seventeen"
    sNWord(18) = "eighteen"
    sNWord(19) = "nineteen"
    sNWord(20) = "twenty"
    sNWord(21) = "thirty"
    sNWord(22) = "fourty"
    sNWord(23) = "fifty"
    sNWord(24) = "sixty"
    sNWord(25) = "seventy"
    sNWord(26) = "eighty"
    sNWord(27) = "ninety"
    sNWord(28) = "hundred"
Case "German"
    Place(1) = ""
    Place(2) = " Tausend "
    Place(3) = " Millionen "
    Place(4) = " Milliarden "
    Place(5) = " Billionen "
    Place(6) = " Die Mantisse ist nicht groß genug für diese Zahl "
    sHWord(1) = ">>>>> Fehler (Absolutbetrag > 999999999999999)! <<<<<"
    sHWord(2) = " (gerundet)"
    sHWord(3) = "Minus "
    sHWord(4) = "und"
    sNWord(0) = "null"
    sNWord(1) = "ein"
    sNWord(2) = "zwei"
    sNWord(3) = "drei"
    sNWord(4) = "vier"
    sNWord(5) = "fünf"
    sNWord(6) = "sechs"
    sNWord(7) = "sieben"
    sNWord(8) = "acht"
    sNWord(9) = "neun"
    sNWord(10) = "zehn"
    sNWord(11) = "elf"
    sNWord(12) = "zwölf"
    sNWord(13) = "dreizehn"
    sNWord(14) = "vierzehn"
    sNWord(15) = "fünfzehn"
    sNWord(16) = "sechzehn"
    sNWord(17) = "siebzehn"
    sNWord(18) = "achtzehn"
    sNWord(19) = "neunzehn"
    sNWord(20) = "zwanzig"
    sNWord(21) = "dreißig"
    sNWord(22) = "vierzig"
    sNWord(23) = "fünfzig"
    sNWord(24) = "sechzig"
    sNWord(25) = "siebzig"
    sNWord(26) = "achtzig"
    sNWord(27) = "neunzig"
    sNWord(28) = "hundert"
End Select

'Empty string = 0
If "" = sNumber Then
    sNumber = "0"
End If
dNumber = sNumber + 0#
'If we cannot cope with it, tell the user!
If Abs(dNumber) > 999999999999999# Then
    sbSpellNumber = sHWord(1)
    Exit Function
End If

'If we have to round we present a suffix "(rounded)"
If Abs(dNumber - Round(dNumber, 2)) > 1E-16 Then
    dNumber = Round(dNumber, 2)
```

```
   suffix = sHWord(2)
End If

'Negative numbers get a prefix "Minus"
If dNumber < 0# Then
   prefix = sHWord(3)
   dNumber = -dNumber
   sNumber = Right(sNumber, Len(sNumber) - 1)
End If

sNumber = Trim(Str(sNumber))
If Left(sNumber, 1) = "." Then
   sNumber = "0" & sNumber
End If
DecimalPlace = InStr(sNumber, ".")
If DecimalPlace > 0 Then
   cents = GetTens(Left(Mid(sNumber, DecimalPlace + 1) & "00", 2), _
             sLang, sCcy)
   sNumber = Trim(Left(sNumber, DecimalPlace - 1))
End If

Count = 1
Do While sNumber <> ""
   Temp = GetHundreds(Right(sNumber, 3), sLang, sCcy)
   If Temp <> "" Then
       If Euros <> "" And sLang = "German" Then
           Euros = Temp & Place(Count) & " " & _
                   sHWord(4) & " " & Euros
       Else
           Euros = Temp & Place(Count) & Euros
       End If
   End If
   If Len(sNumber) > 3 Then
       sNumber = Left(sNumber, Len(sNumber) - 3)
   Else
       sNumber = ""
   End If
   Count = Count + 1
Loop

Select Case sCcy
Case "EUR"
   Select Case Euros
       Case ""
           Euros = sNWord(0) & " Euros"
       Case sNWord(1)
           Euros = sNWord(1) & " Euro"
       Case Else
           Euros = Euros & " Euros"
   End Select

   Select Case cents
       Case ""
           cents = " " & sHWord(4) & " " & sNWord(0) & " Cents"
       Case sNWord(1)
           cents = " " & sHWord(4) & " " & sNWord(1) & " Cent"
       Case Else
           cents = " " & sHWord(4) & " " & cents & " Cents"
   End Select
Case "GBP"
   Select Case Euros
       Case ""
           Euros = sNWord(0) & " Pounds"
       Case sNWord(1)
           Euros = sNWord(1) & " Pound"
       Case Else
           Euros = Euros & " Pounds"
   End Select
   Select Case cents
       Case ""
           cents = " " & sHWord(4) & " " & sNWord(0) & " Pence"
       Case sNWord(1)
           cents = " " & sHWord(4) & " " & sNWord(1) & " Penny"
       Case Else
           cents = " " & sHWord(4) & " " & cents & " Pence"
   End Select
Case "USD"
   Select Case Euros
       Case ""
           Euros = sNWord(0) & " Dollars"
       Case sNWord(1)
           Euros = sNWord(1) & " Dollar"
       Case Else
           Euros = Euros & " Dollars"
   End Select

   Select Case cents
       Case ""
           cents = " " & sHWord(4) & " " & sNWord(0) & " Cents"
       Case sNWord(1)
```

```
            cents = " " & sHWord(4) & " " & sNWord(1) & " Cent"
        Case Else
            cents = " " & sHWord(4) & " " & cents & " Cents"
    End Select
End Select

Temp = UCase(Replace(Euros & cents, "  ", " "))
Select Case sLang
Case "English"
    Temp = Application.WorksheetFunction.Proper(Temp)
    Temp = Replace(Temp, " And ", " and ")
Case "German"
    Temp = Application.WorksheetFunction.Proper(Temp)
    Temp = Replace(Temp, "Ein Millionen", "Eine Million")
    Temp = Replace(Temp, "Ein Milliarden", "Eine Milliarde")
    Temp = Replace(Temp, "Ein Billionen", "Eine Billion")
    Temp = Replace(Temp, "Dollars", "Dollar")
    Temp = Replace(Temp, "Cents", "Cent")
    Temp = Replace(Temp, "Pounds", "Pfund")
    Temp = Replace(Temp, "Pound", "Pfund")
    Temp = Replace(Temp, "Euros", "Euro")
    Temp = Replace(Temp, "Pence", "Pennies")
    Temp = Replace(Temp, " Und ", " und ")
End Select
sbSpellNumber = prefix & Temp & suffix
End Function

Private Function GetHundreds(ByVal sNumber, _
            Optional sLang As String = "English", _
            Optional sCcy As String = "USD") As String
Dim Result As String

If Val(sNumber) = 0 Then Exit Function
    sNumber = Right("000" & sNumber, 3)
    If Mid(sNumber, 1, 1) <> "0" Then
        Result = GetDigit(Mid(sNumber, 1, 1)) _
                & sNWord(28)
        If Mid(sNumber, 2, 2) <> "00" Then
            Result = Result & sHWord(4)
        End If
    End If
    If Mid(sNumber, 2, 1) <> "0" Then
        Result = Result & GetTens(Mid(sNumber, 2), sLang, sCcy)
    ElseIf Mid(sNumber, 3, 1) <> "0" Then
        Result = Result & GetDigit(Mid(sNumber, 3))
    End If
    GetHundreds = Result
End Function

Private Function GetTens(TensText As String, _
            Optional sLang As String = "English", _
            Optional sCcy As String = "USD")
Dim Result As String

Result = ""
If Val(Left(TensText, 1)) = 1 Then   '10-19...
    If Val(TensText) > 9 And Val(TensText) < 20 Then
        GetTens = sNWord(Val(TensText))
    End If
    Exit Function
Else                                 '20-99...
    If Val(Left(TensText, 1)) > 1 And _
        Val(Left(TensText, 1)) < 10 Then
        Result = sNWord(18 + Val(Left(TensText, 1)))
    Else
        Result = GetDigit(Right(TensText, 1))
    End If
    If Right(TensText, 1) <> "0" And Left(TensText, 1) <> "0" Then
        Select Case sLang
        Case "German"
            Result = GetDigit(Right(TensText, 1)) & _
                        sHWord(4) & Result
        Case "English"
            Result = Result & GetDigit(Right(TensText, 1))
        End Select
    End If
End If
GetTens = Result
End Function

Private Function GetDigit(Digit As String) As String
If Val(Digit) < 10 Then
    GetDigit = sNWord(Val(Digit))
Else
    GetDigit = ""
End If
End Function
```

Umwandlungen zwischen dem Dezimalsystem und dem Binärsystem

Wie lautet die binäre Darstellung (Bitlänge = 256) der Dezimalzahl -872362346234627834628734627834627834628? Die eingebaute Excel Funktion DEZINBIN hilft hier nicht weiter. Sie kann nur Eingaben von -512 bis 511 verarbeiten. Wenn Sie die korrekte Antwort
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111110101101111101101010001111110011101111100101111100100001000011101011001001010011001101000100
1100111101010000101010100101111001111100
erhalten wollen, benutzen Sie bitte die unten gezeigte Funktion *sbDec2Bin*.

Bitte bachten: Nachkommastellen werden lediglich für positive Zahlen unterstützt. Zum Beispiel lautet die Dezimalzahl 0.5 im Binärformat 0.1.

Funktion	Parameter 1	Parameter 2	Ergebnis	Kommentar
sbDec2Bin	2005		1111101010	
sbDec2Bin	2005	11	#Wert!	Mit 11 Bit lässt sich 2005 nicht darstellen
sbDec2Bin	11,5		1011,1	
sbDec2Bin	2,25		10,01	
sbBin2Dec	1111		15	
sbBin2Dec	1111	4	-1	4 Bit: das erste Bit ist das Vorzeichen
sbBin2Dec	11,11		3,75	

sbDec2Bin, sbBin2Dec, sbDivBy2, sbBinNeg und sbDecAdd Programmcode

```
Function sbDec2Bin(ByVal sDecimal As String, _
               Optional lBits As Long = 32, _
               Optional blZeroize As Boolean = False) As String
'Convert a decimal number into its binary equivalent.
'(C) (P) by Bernd Plumhoff 18-Dec-2021 PB V0.4
Dim sDec As String, sFrac As String
Dim sD As String, sB As String
Dim blNeg As Boolean
Dim i As Long, lPosDec As Long, lLenBinInt As Long
lPosDec = InStr(sDecimal, Application.DecimalSeparator)
If lPosDec > 0 Then
   If Left(sDecimal, 1) = "-" Then 'So far we cannot handle
                                   'negative fractions, will come later
       sbDec2Bin = CVErr(xlErrValue)
       Exit Function
   End If
   sDec = Left(sDecimal, lPosDec - 1)
   sFrac = Right(sDecimal, Len(sDecimal) - lPosDec)
   lPosDec = Len(sFrac)
Else
   sDec = sDecimal
   sFrac = ""
End If
sB = ""
If Left(sDec, 1) = "-" Then
   blNeg = True
   sD = Right(sDec, Len(sDec) - 1)
Else
   blNeg = False
   sD = sDec
End If
Do While Len(sD) > 0
   Select Case Right(sD, 1)
       Case "0", "2", "4", "6", "8"
           sB = "0" & sB
       Case "1", "3", "5", "7", "9"
           sB = "1" & sB
       Case Else
           sbDec2Bin = CVErr(xlErrValue)
           Exit Function
   End Select
   sD = sbDivBy2(sD, True)
   If sD = "0" Then
       Exit Do
   End If
Loop
If blNeg And sB <> "1" & String(lBits - 1, "0") Then
   sB = sbBinNeg(sB, lBits)
End If
'Test whether string representation is in range and correct
'If not, the user has to increase lbits
lLenBinInt = Len(sB)
If lLenBinInt > lBits Then
   sbDec2Bin = CVErr(xlErrNum)
   Exit Function
Else
   If (Len(sB) = lBits) And (Left(sB, 1) <> -blNeg & "") Then
       sbDec2Bin = CVErr(xlErrNum)
       Exit Function
   End If
End If

If blZeroize Then sB = Right(String(lBits, "0") & sB, lBits)

If lPosDec > 0 And lLenBinInt + 1 < lBits Then
   sB = sB & Application.DecimalSeparator
   i = 1
   Do While i + lLenBinInt < lBits
       sFrac = sbDecAdd(sFrac, sFrac) 'Double fractional part
       If Len(sFrac) > lPosDec Then
           sB = sB & "1"
           sFrac = Right(sFrac, lPosDec)
           If sFrac = String(lPosDec, "0") Then
               Exit Do
           End If
       Else
           sB = sB & "0"
       End If
       i = i + 1
   Loop
   sbDec2Bin = sB
Else
   sbDec2Bin = sB
End If
End Function
```

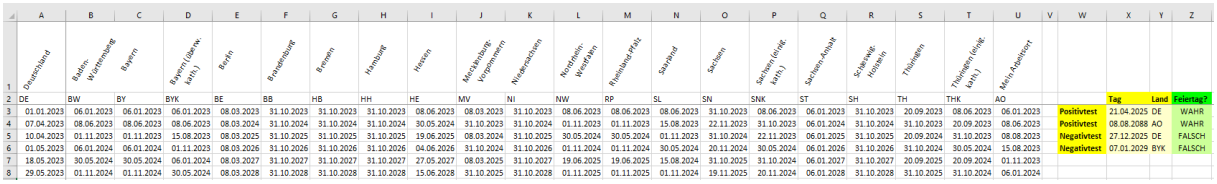
```
Function sbBin2Dec(sBinary As String, _
    Optional lBits As Long = 32) As String
'Converts a binary number into its decimal equivalent.
'(C) (P) by Bernd Plumhoff 18-Dec-2021 PB V0.4
Dim sBin As String
Dim sB As String
Dim sFrac As String
Dim sD As String
Dim sR As String
Dim blNeg As Boolean
Dim i As Long
Dim lPosDec As Long

lPosDec = InStr(sBinary, Application.DecimalSeparator)
If lPosDec > 0 Then
   If (Left(Right(String(lBits, "0") & sBinary, lBits), 1) = "1") And _
       Len(sBin) >= lBits Then 'So far we cannot handle Right(String(lBits, "0") & sB, lBits)
                   'negative fractions, will come later
       sbBin2Dec = CVErr(xlErrValue)
       Exit Function
   End If
   sBin = Left(sBinary, lPosDec - 1)
   sFrac = Right(sBinary, Len(sBinary) - lPosDec)
   lPosDec = Len(sFrac)
Else
   sBin = sBinary
   sFrac = ""
End If

Select Case Sgn(Len(sBin) - lBits)
   Case 1
       sbBin2Dec = CVErr(xlErrNum)
       Exit Function
   Case 0
       If Left(sBin, 1) = "1" Then
           sB = sbBinNeg(sBin, lBits)
           blNeg = True
       Else
           sB = sBin
           blNeg = False
       End If
   Case -1
       sB = sBin
       blNeg = False
End Select
sD = "1"
sR = "0"
For i = Len(sB) To 1 Step -1
   Select Case Mid(sB, i, 1)
       Case "1"
           sR = sbDecAdd(sR, sD)
       Case "0"
           'Do nothing
       Case Else
           sbBin2Dec = CVErr(xlErrNum)
           Exit Function
   End Select
   sD = sbDecAdd(sD, sD) 'Double sd
Next i

If lPosDec > 0 Then 'now the fraction
   sD = "0" & Application.DecimalSeparator & "5"
   For i = 1 To lPosDec
       If Mid(sFrac, i, 1) = "1" Then
           sR = sbDecAdd(sR, sD)
       End If
       sD = sbDivBy2(sD, False)
   Next i
End If

If blNeg Then
   sbBin2Dec = "-" & sR
Else
   sbBin2Dec = sR
End If
End Function

Function sbDivBy2(sDecimal As String, blInt As Boolean) As String
'Divide positive sDecimal by two, blInt = TRUE returns integer only
'(C) (P) by Bernd Plumhoff 18-Dec-2021 PB V0.4
Dim i As Long, lPosDec As Long
Dim sDec As String, sD As String
Dim lCarry As Long

If Not blInt Then
   lPosDec = InStr(sDecimal, Application.DecimalSeparator)
   If lPosDec > 0 Then
       sDec = Left(sDecimal, lPosDec - 1) & _
              Right(sDecimal, Len(sDecimal) - lPosDec) 'Without decimal point
       'lposdec already defines location of decimal point
```

```
   Else
       sDec = sDecimal
       lPosDec = Len(sDec) + 1 'Location of decimal point
   End If
   If ((1 * Right(sDec, 1)) Mod 2) = 1 Then
       sDec = sDec & "0"  'Append zero so that integer algorithm
                          'below calculates division exactly
   End If
Else
   sDec = sDecimal
End If

lCarry = 0
For i = 1 To Len(sDec)
   sD = sD & Int((lCarry * 10 + Mid(sDec, i, 1)) / 2)
   lCarry = (lCarry * 10 + Mid(sDec, i, 1)) Mod 2
Next i

If Not blInt Then
   If Right(sD, Len(sD) - lPosDec + 1) <> _
       String(Len(sD) - lPosDec + 1, "0") Then   'frac part is non-zero
       i = Len(sD)
       Do While Mid(sD, i, 1) = "0"
           i = i - 1  'Skip trailing zeros
       Loop
       sD = Left(sD, lPosDec - 1) & Application.DecimalSeparator & _
            Mid(sD, lPosDec, i - lPosDec + 1) 'Insert decimal point again
   End If
End If

i = 1
Do While i < Len(sD)
   If Mid(sD, i, 1) = "0" Then
       i = i + 1
   Else
       Exit Do
   End If
Loop
If Mid(sD, i, 1) = Application.DecimalSeparator Then
   i = i - 1
End If
sbDivBy2 = Right(sD, Len(sD) - i + 1)
End Function

Function sbBinNeg(sBin As String, _
               Optional lBits As Long = 32) As String
'Negate sBin: take the 2's-complement, then add one
'(C) (P) by Bernd Plumhoff 18-Dec-2021 PB V0.4
Dim i As Long, sB As String

If Len(sBin) > lBits Or sBin = "1" & String(lBits - 1, "0") Then
   sbBinNeg = CVErr(xlErrValue)
   Exit Function
End If

'Calculate 2's-complement
For i = Len(sBin) To 1 Step -1
   Select Case Mid(sBin, i, 1)
       Case "1"
           sB = "0" & sB
       Case "0"
           sB = "1" & sB
       Case Else
           sbBinNeg = CVErr(xlErrValue)
           Exit Function
   End Select
Next i

sB = String(lBits - Len(sBin), "1") & sB

'Now add 1
i = lBits
Do While i > 0
   If Mid(sB, i, 1) = "1" Then
       Mid(sB, i, 1) = "0"
       i = i - 1
   Else
       Mid(sB, i, 1) = "1"
       i = 0
   End If
Loop

'Finally strip leading zeros
i = InStr(sB, "1")
If i = 0 Then
   sbBinNeg = "0"
Else
   sbBinNeg = Right(sB, Len(sB) - i + 1)
End If
End Function
```

```
Function sbDecAdd(sOne As String, sTwo As String) As String
'Sum up two positive string decimals.
'(C) (P) by Bernd Plumhoff 18-Dec-2021 PB V0.4
Dim lStrLen As Long
Dim s1 As String, s2 As String
Dim sA As String, sB As String, sR As String
Dim d As Long, lCarry As Long, lPosDec1 As Long, lPosDec2 As Long
Dim sF1 As String, sF2 As String

lPosDec1 = InStr(sOne, Application.DecimalSeparator)
If lPosDec1 > 0 Then
   s1 = Left(sOne, lPosDec1 - 1)
   sF1 = Right(sOne, Len(sOne) - lPosDec1)
   lPosDec1 = Len(sF1)
Else
   s1 = sOne
   sF1 = ""
End If
lPosDec2 = InStr(sTwo, Application.DecimalSeparator)
If lPosDec2 > 0 Then
   s2 = Left(sTwo, lPosDec2 - 1)
   sF2 = Right(sTwo, Len(sTwo) - lPosDec2)
   lPosDec2 = Len(sF2)
Else
   s2 = sTwo
   sF2 = ""
End If

If lPosDec1 + lPosDec2 > 0 Then
   If lPosDec1 > lPosDec2 Then
       sF2 = sF2 & String(lPosDec1 - lPosDec2, "0")
   Else
       sF1 = sF1 & String(lPosDec2 - lPosDec1, "0")
       lPosDec1 = lPosDec2
   End If
   sF1 = sbDecAdd(sF1, sF2) 'Add fractions as integer numbers
   If Len(sF1) > lPosDec1 Then
       lCarry = 1
       sF1 = Right(sF1, lPosDec1)
   Else
       lCarry = 0
   End If
   Do While lPosDec1 > 0
       If Mid(sF1, lPosDec1, 1) <> "0" Then
           Exit Do
       End If
       lPosDec1 = lPosDec1 - 1
   Loop
   sF1 = Left(sF1, lPosDec1)
Else
   lCarry = 0
End If

lStrLen = Len(s1)
If lStrLen < Len(s2) Then
   lStrLen = Len(s2)
   sA = String(lStrLen - Len(s1), "0") & s1
   sB = s2
Else
   sA = s1
   sB = String(lStrLen - Len(s2), "0") & s2
End If

Do While lStrLen > 0
   d = 0 + Mid(sA, lStrLen, 1) + Mid(sB, lStrLen, 1) + lCarry
   If d > 9 Then
       sR = (d - 10) & sR
       lCarry = 1
   Else
       sR = d & sR
       lCarry = 0
   End If
   lStrLen = lStrLen - 1
Loop
If lCarry > 0 Then
   sR = lCarry & sR
End If

If lPosDec1 > 0 Then
   sbDecAdd = sR & Application.DecimalSeparator & sF1
Else
   sbDecAdd = sR
End If

End Function
```

Feiertage ermitteln – *IstFeiertag*

Wollen Sie herausfinden, ob ein Datum ein deutscher Feiertag ist? Oder ob es ein Feiertag in einem Bundesland ist?



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	Deutschland	Baden-Württemberg	Bayern	Bayern (überw. kath.)	Berlin	Brandenburg	Bremen	Hamburg	Hessen	Mecklenburg-Vorpommern	Niedersachsen	Nordrhein-Westfalen	Rheinland-Pfalz	Saarland	Sachsen	Sachsen (einig. kath.)	Sachsen-Anhalt	Schleswig-Holstein	Thüringen	Thüringen (einig. kath.)	Mein Arbeitsort					
2	DE	BW	BY	BYK	BE	BB	HB	HH	HE	MV	NI	NW	RP	SL	SN	SNK	ST	SH	TH	THK	AO			Tag	Land	Feiertag?
3	01.01.2023	06.01.2023	06.01.2023	06.01.2023	08.03.2023	31.10.2023	31.10.2023	31.10.2023	08.06.2023	08.03.2023	31.10.2023	08.06.2023	08.06.2023	08.06.2023	31.10.2023	08.06.2023	06.01.2023	31.10.2023	20.09.2023	08.06.2023	06.01.2023		Positivtest	21.04.2025	DE	WAHR
4	07.04.2023	08.06.2023	08.06.2023	08.06.2023	08.03.2024	31.10.2024	31.10.2024	31.10.2024	30.05.2024	31.10.2023	31.10.2024	01.11.2023	01.11.2023	15.08.2023	22.11.2023	31.10.2023	06.01.2024	31.10.2024	31.10.2023	20.09.2023	08.06.2023		Positivtest	08.08.2088	AO	WAHR
5	10.04.2023	01.11.2023	01.11.2023	15.08.2023	08.03.2025	31.10.2025	31.10.2025	31.10.2025	19.06.2025	08.03.2024	31.10.2025	30.05.2024	30.05.2024	01.11.2023	31.10.2024	22.11.2023	06.01.2025	31.10.2025	20.09.2024	31.10.2023	08.08.2023		Negativtest	27.12.2025	DE	FALSCH
6	01.05.2023	06.01.2024	06.01.2024	01.11.2023	08.03.2026	31.10.2026	31.10.2026	31.10.2026	04.06.2026	31.10.2024	31.10.2026	01.11.2024	01.11.2024	30.05.2024	20.11.2024	30.05.2024	06.01.2026	31.10.2026	31.10.2024	30.05.2024	15.08.2023		Negativtest	07.01.2029	BYK	FALSCH
7	18.05.2023	30.05.2024	30.05.2024	06.01.2024	08.03.2027	31.10.2027	31.10.2027	31.10.2027	27.05.2027	08.03.2025	31.10.2027	19.06.2025	19.06.2025	15.08.2024	31.10.2025	31.10.2024	06.01.2027	31.10.2027	20.09.2025	20.09.2024	01.11.2023					
8	29.05.2023	01.11.2024	01.11.2024	30.05.2024	08.03.2028	31.10.2028	31.10.2028	31.10.2028	15.06.2028	31.10.2025	31.10.2028	01.11.2025	01.11.2025	01.11.2024	19.11.2025	20.11.2024	06.01.2028	31.10.2028	31.10.2025	31.10.2024	06.01.2024					

Hier sind alle Feiertage von 2023 bis 2100 gezeigt. Sie können die Liste manuell ändern, z. B. Feiertage hinzufügen oder entfernen. Diese Liste kann mit dem unten angebotenen Programm Feiertage_generieren.xlsm automatisch erzeugt werden, wobei aber historische Feiertagsänderungen (z. B. Abschaffung des Buß- und Bettages seit 1995 mit Ausnahme Sachsens) und Gebietsreformen (z. B. Wiedervereinigung 1990) nicht berücksichtigt sind.

Für Schulferien (besser: schulfreie Tage) können die Feiertage erweitert werden: Man fügt alle einzelnen Ferientage zu den entsprechenden Bundeslandspalten hinzu und benennt die benutzerdefinierte Funktion *IstFeiertag* um in *IstSchulfreierTag*.

IstFeiertag Programmcode

```
Enum Spalten
    col_LBound = 0
    col_DE  'Deutschland
    col_BW  'Baden_Württemberg
    col_BY  'Bayern
    col_BYK 'Bayern (überw. kath.)
    col_BE  'Berlin
    col_BB  'Brandenburg
    col_HB  'Bremen
    col_HH  'Hamburg
    col_HE  'Hessen
    col_MV  'Mecklenburg-Vorpommern
    col_NI  'Niedersachsen
    col_NW  'Nordrhein-Westfalen
    col_RP  'Rheinland-Pfalz
    col_SL  'Saarland
    col_SN  'Sachsen
    col_SNK 'Sachsen (einig. kath.)
    col_ST  'Sachsen-Anhalt
    col_SH  'Schleswig-Holstein
    col_TH  'Thüringen
    col_THK 'Thüringen (einig. kath.)
    col_AO  'Mein Arbeitsort
    col_UBound
End Enum

Function IstFeiertag(dt As Date, _
    Optional Land As String = "DE") As Variant
'Prüft, ob ein Datum ein Feiertag ist, mit Land = "DE"
'ob es ein bundeseinheitlicher Feiertag ist, sonst
'ob bundeseinheitlich oder vom entsprechenden Bundesland
'oder der individuelle "mein Arbeitsort".
'(C) (P) by Bernd Plumhoff 18-Jan-2024 PB V0.2
Static Feiertage    As Variant
Static LetzteZeile  As Variant
Dim d               As Date
Dim i               As Long
Dim j               As Long
Dim s               As String
Dim ws              As Worksheet

With Application.WorksheetFunction

If dt = 0# Or Land = "" Then
    IstFeiertag = CVErr(xlErrNull)
    Exit Function
End If
```

```
Set ws = Sheets("Feiertage")

If IsEmpty(Feiertage) Then
    LetzteZeile = ws.Cells(2, 1).End(xlDown).Row
    Set Feiertage = Range(ws.Cells(3, 1), _
        ws.Cells(LetzteZeile, col_UBound - 1))
End If

i = 1
s = ws.Cells(2, i)
Do While s <> ""
    If Land = s Then Exit Do
    i = i + 1
    s = ws.Cells(2, i)
Loop

If s = "" Then
    IstFeiertag = CVErr(xlErrName)
    Exit Function
End If

IstFeiertag = False

'Bundesweiter Feiertag?
j = 1
d = Feiertage(j, 1)
Do While j < LetzteZeile - 2
    If dt = d Then
        IstFeiertag = True
        Exit Function
    End If
    j = j + 1
    d = Feiertage(j, 1)
Loop

'Bundesland Feiertag?
If i > 1 Then
    j = 1
    d = Feiertage(j, i)
    Do While j < LetzteZeile - 2
        If dt = d Then
            IstFeiertag = True
            Exit Function
        End If
        j = j + 1
        d = Feiertage(j, i)
    Loop
End If

End With

End Function
```

Feiertagsliste_erstellen Programmcode

```
Const StartJahr = 2023
Const EndJahr = 2100

Enum Spalten
    col_LBound = 0
    col_DE  'Deutschland
    col_BW  'Baden_Württemberg
    col_BY  'Bayern
    col_BYK 'Bayern (überw. kath.)
    col_BE  'Berlin
    col_BB  'Brandenburg
    col_HB  'Bremen
    col_HH  'Hamburg
    col_HE  'Hessen
    col_MV  'Mecklenburg-Vorpommern
    col_NI  'Niedersachsen
    col_NW  'Nordrhein-Westfalen
    col_RP  'Rheinland-Pfalz
    col_SL  'Saarland
    col_SN  'Sachsen
    col_SNK 'Sachsen (einig. kath.)
    col_ST  'Sachsen-Anhalt
    col_SH  'Schleswig-Holstein
    col_TH  'Thüringen
    col_THK 'Thüringen (einig. kath.)
    col_AO  'Mein Arbeitsort
    col_UBound
End Enum
```

```
Sub Feiertagsliste_erstellen()
'Generiert Feiertage für die Jahre StartJahr bis EndJahr.
'(C) (P) by Bernd Plumhoff 18-Jan-2024 PB V0.2

Dim Advent4                             As Date
Dim Easter                              As Date

Dim i                                   As Long
Dim r(col_LBound + 1 To col_UBound - 1) As Long

Dim state                               As SystemState

Set state = New SystemState

wsGen.Range("3:100000").Delete
For i = col_LBound + 1 To col_UBound - 1
    r(i) = 3
Next i

For i = StartJahr To EndJahr

    Advent4 = DateSerial(i, 12, 25) - Weekday(DateSerial(i, 12, 25), 2)
    Easter = EasterUSNO(i)

    'Deutschland
    wsGen.Cells(r(col_DE), col_DE) = DateSerial(i, 1, 1)    'Neujahr
    r(col_DE) = r(col_DE) + 1
    wsGen.Cells(r(col_DE), col_DE) = Easter - 2             'Karfreitag
    r(col_DE) = r(col_DE) + 1
    wsGen.Cells(r(col_DE), col_DE) = Easter + 1             'Ostermontag
    r(col_DE) = r(col_DE) + 1
    wsGen.Cells(r(col_DE), col_DE) = DateSerial(i, 5, 1)    'Maifeiertag
    r(col_DE) = r(col_DE) + 1
    wsGen.Cells(r(col_DE), col_DE) = Easter + 39            'Himmelfahrt
    r(col_DE) = r(col_DE) + 1
    wsGen.Cells(r(col_DE), col_DE) = Easter + 50            'Pfingstmontag
    r(col_DE) = r(col_DE) + 1
    wsGen.Cells(r(col_DE), col_DE) = DateSerial(i, 10, 3)   'Tag der deutschen Einheit
    r(col_DE) = r(col_DE) + 1
    wsGen.Cells(r(col_DE), col_DE) = DateSerial(i, 12, 25)  '1. Weihnachtstag
    r(col_DE) = r(col_DE) + 1
    wsGen.Cells(r(col_DE), col_DE) = DateSerial(i, 12, 26)  '2. Weihnachtstag
    r(col_DE) = r(col_DE) + 1

    'Baden-Württemberg
    wsGen.Cells(r(col_BW), col_BW) = DateSerial(i, 1, 6)    'Hl. Drei Könige
    r(col_BW) = r(col_BW) + 1
    wsGen.Cells(r(col_BW), col_BW) = Easter + 60            'Fronleichman
    r(col_BW) = r(col_BW) + 1
    wsGen.Cells(r(col_BW), col_BW) = DateSerial(i, 11, 1)   'Allerheiligen
    r(col_BW) = r(col_BW) + 1

    'Bayern
    wsGen.Cells(r(col_BY), col_BY) = DateSerial(i, 1, 6)    'Hl. Drei Könige
    r(col_BY) = r(col_BY) + 1
    wsGen.Cells(r(col_BY), col_BY) = Easter + 60            'Fronleichman
    r(col_BY) = r(col_BY) + 1
    wsGen.Cells(r(col_BY), col_BY) = DateSerial(i, 11, 1)   'Allerheiligen
    r(col_BY) = r(col_BY) + 1

    'Bayern (überwiegend katholische Bevölkerung)
    wsGen.Cells(r(col_BYK), col_BYK) = DateSerial(i, 1, 6)  'Hl. Drei Könige
    r(col_BYK) = r(col_BYK) + 1
    wsGen.Cells(r(col_BYK), col_BYK) = Easter + 60          'Fronleichman
    r(col_BYK) = r(col_BYK) + 1
    wsGen.Cells(r(col_BYK), col_BYK) = DateSerial(i, 8, 15) 'Mariä Himmelfahrt
    r(col_BYK) = r(col_BYK) + 1
    wsGen.Cells(r(col_BYK), col_BYK) = DateSerial(i, 11, 1) 'Allerheiligen
    r(col_BYK) = r(col_BYK) + 1

    'Berlin
    wsGen.Cells(r(col_BE), col_BE) = DateSerial(i, 3, 8)    'Int. Frauentag
    r(col_BE) = r(col_BE) + 1

    'Brandenburg
    wsGen.Cells(r(col_BB), col_BB) = DateSerial(i, 10, 31)  'Reformationstag
    r(col_BB) = r(col_BB) + 1

    'Bremen
    wsGen.Cells(r(col_HB), col_HB) = DateSerial(i, 10, 31)  'Reformationstag
    r(col_HB) = r(col_HB) + 1

    'Hamburg
    wsGen.Cells(r(col_HH), col_HH) = DateSerial(i, 10, 31)  'Reformationstag
    r(col_HH) = r(col_HH) + 1

    'Hessen
    wsGen.Cells(r(col_HE), col_HE) = Easter + 60            'Fronleichman
    r(col_HE) = r(col_HE) + 1
```

```
    'Mecklenburg-Vorpommern
    wsGen.Cells(r(col_MV), col_MV) = DateSerial(i, 3, 8)    'Int. Frauentag
    r(col_MV) = r(col_MV) + 1
    wsGen.Cells(r(col_MV), col_MV) = DateSerial(i, 10, 31)  'Reformationstag
    r(col_MV) = r(col_MV) + 1

    'Niedersachsen
    wsGen.Cells(r(col_NI), col_NI) = DateSerial(i, 10, 31)  'Reformationstag
    r(col_NI) = r(col_NI) + 1

    'Nordrhein-Westfalen
    wsGen.Cells(r(col_NW), col_NW) = Easter + 60            'Fronleichman
    r(col_NW) = r(col_NW) + 1
    wsGen.Cells(r(col_NW), col_NW) = DateSerial(i, 11, 1)   'Allerheiligen
    r(col_NW) = r(col_NW) + 1

    'Rheinland-Pfalz
    wsGen.Cells(r(col_RP), col_RP) = Easter + 60            'Fronleichman
    r(col_RP) = r(col_RP) + 1
    wsGen.Cells(r(col_RP), col_RP) = DateSerial(i, 11, 1)   'Allerheiligen
    r(col_RP) = r(col_RP) + 1

    'Saarland
    wsGen.Cells(r(col_SL), col_SL) = Easter + 60            'Fronleichman
    r(col_SL) = r(col_SL) + 1
    wsGen.Cells(r(col_SL), col_SL) = DateSerial(i, 8, 15)   'Mariä Himmelfahrt
    r(col_SL) = r(col_SL) + 1
    wsGen.Cells(r(col_SL), col_SL) = DateSerial(i, 11, 1)   'Allerheiligen
    r(col_SL) = r(col_SL) + 1

    'Sachsen
    wsGen.Cells(r(col_SN), col_SN) = DateSerial(i, 10, 31)  'Reformationstag
    r(col_SN) = r(col_SN) + 1
    wsGen.Cells(r(col_SN), col_SN) = Advent4 - 32           'Buß- und Bettag
    r(col_SN) = r(col_SN) + 1

    'Sachsen (einige katholische Gemeinden)
    wsGen.Cells(r(col_SNK), col_SNK) = Easter + 60          'Fronleichman
    r(col_SNK) = r(col_SNK) + 1
    wsGen.Cells(r(col_SNK), col_SNK) = DateSerial(i, 10, 31) 'Reformationstag
    r(col_SNK) = r(col_SNK) + 1
    wsGen.Cells(r(col_SNK), col_SNK) = Advent4 - 32         'Buß- und Bettag
    r(col_SNK) = r(col_SNK) + 1

    'Sachsen-Anhalt
    wsGen.Cells(r(col_ST), col_ST) = DateSerial(i, 1, 6)    'Hl. Drei Könige
    r(col_ST) = r(col_ST) + 1

    'Schleswig-Holstein
    wsGen.Cells(r(col_SH), col_SH) = DateSerial(i, 10, 31)  'Reformationstag
    r(col_SH) = r(col_SH) + 1

    'Thüringen
    wsGen.Cells(r(col_TH), col_TH) = DateSerial(i, 9, 20)   'Weltkindertag
    r(col_TH) = r(col_TH) + 1
    wsGen.Cells(r(col_TH), col_TH) = DateSerial(i, 10, 31)  'Reformationstag
    r(col_TH) = r(col_TH) + 1

    'Thüringen (einige katholische Gemeinden)
    wsGen.Cells(r(col_THK), col_THK) = Easter + 60          'Fronleichman
    r(col_THK) = r(col_THK) + 1
    wsGen.Cells(r(col_THK), col_THK) = DateSerial(i, 9, 20) 'Weltkindertag
    r(col_THK) = r(col_THK) + 1
    wsGen.Cells(r(col_THK), col_THK) = DateSerial(i, 10, 31) 'Reformationstag
    r(col_THK) = r(col_THK) + 1

    'Mein Arbeitsort (nehmen wir einmal an, Augsburg)
    wsGen.Cells(r(col_AO), col_AO) = DateSerial(i, 1, 6)    'Hl. Drei Könige
    r(col_AO) = r(col_AO) + 1
    wsGen.Cells(r(col_AO), col_AO) = Easter + 60            'Fronleichman
    r(col_AO) = r(col_AO) + 1
    wsGen.Cells(r(col_AO), col_AO) = DateSerial(i, 8, 8)    'Hohes Friedensfest Augsburg
    r(col_AO) = r(col_AO) + 1
    wsGen.Cells(r(col_AO), col_AO) = DateSerial(i, 8, 15)   'Mariä Himmelfahrt
    r(col_AO) = r(col_AO) + 1
    wsGen.Cells(r(col_AO), col_AO) = DateSerial(i, 11, 1)   'Allerheiligen
    r(col_AO) = r(col_AO) + 1

Next i
End Sub

Public Function EasterUSNO(YYYY As Long) As Long
'Source: http://www.cpearson.com/excel/easter.aspx
End Function
```

Zahl vollständig nicht-wissenschaftlich darstellen – *sbNum2Str*

Wenn Sie eine nicht-wissenschaftliche Zahlendarstellung mit allen signifikanten Ziffern und allen vorangestellten und nachstehenden Nullen benötigen, können Sie diese Funktion *sbNum2Str* verwenden:

B2 =sbNum2Str(A2)

	A	B
1	**Input**	**sbNum2Str**
2	3,141592654	3,14159265358979
3	-1	-1
4	-1,00E+307	- 100000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000 00000000000000000000000
5	-1,00E-307	- 0,0000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000 000000000000000000000000000000000000000000000000000000000 000000000000000000000001
6	-0,001	-0,001
7	-123	-123
8	-0,1	-0,1

sbNum2Str Code

```
Function sbNum2Str(d As Double) As String
'Returns string with number representation with all
'significant digits and leading or trailing zeros, i.e.
'1E+3 will be returned as 1000
'1E-3 will be 0.001
'Pi() will be 3.14159265358979
'(C) (P) by Bernd Plumhoff 15-Nov-2010 PB V0.20
Dim v
Dim lExp As Long, lLenMant As Long
Dim sDot As String 'decimal separator
Dim sMant As String 'new mantissa

If d < 0# Then
    sbNum2Str = "-" & sbNum2Str(-d)
    Exit Function
End If
sDot = Application.DecimalSeparator
'Split scientific representation into mantissa and exponent
v = Split(Format(d, _
        "0." & String(15, "#") & "E+0"), "E")
If Left(v(0), 1) = "0" Then
    sbNum2Str = "0"
    Exit Function
End If
lExp = CLng(v(1))    'get exponent
v = Split(v(0), sDot)
If lExp < 0 Then
    sbNum2Str = "0" & sDot & String(-lExp - 1, "0") & _
              v(0) & v(1)
Else
    lLenMant = Len(v(1))
    If Len(v(1)) > lExp Then
        sMant = v(0) & v(1)
        sbNum2Str = Left(sMant, lExp + 1) & sDot & _
                  Right(sMant, Len(sMant) - lExp - 1)
    Else
        sbNum2Str = v(0) & v(1) & String(lExp - lLenMant, "0")
    End If
End If
End Function
```

Nummer eines Monatsnamens – *sbMonatsZahl*

Falls Sie die Nummer für einen Monatsnamen ermitteln wollen:

English		Deutsch		Schweizer Deutsch	
Month	Zahl	Monat	Zahl	Monet	Zahl
January	1	Januar	1	Jänner	1
February	2	Februar	2	Hornig	2
March	3	März	3	Merze	3
April	4	April	4	Abrele	4
May	5	Mai	5	Mäie	5
June	6	Juni	6	Brachet	6
July	7	Juli	7	Heuet	7
August	8	August	8	Augschte	8
September	9	September	9	Herbschtmonet	9
October	10	Oktober	10	Wiimonet	10
November	11	November	11	Wintermonet	11
December	12	Dezember	12	Chrischtmonet	12

sbMonatsZahl Programmcode

```
Function sbMonatsZahl(sMonat As String) As Integer
'Gibt die Monatsnummer für einen Monatsnamen zurück.
'(C) (P) by Bernd Plumhoff  19-Nov-2022 PB V0.2
Dim s As String, c1 As String, c2 As String, c3 As String, c4 As String

s = Left(LCase(sMonat) & String(4, " "), 4)
c1 = Left(s, 1)
Select Case c1
Case "a"
    c2 = Mid(s, 2, 1)
    Select Case c2
    Case "b", "p"
        sbMonatsZahl = 4 'Abrele, April, Aprilius
    Case "u"
        sbMonatsZahl = 8 'August, Augschte, Augusti
    Case Else
        sbMonatsZahl = CVErr(xlErrNum)
    End Select
Case "b"
    sbMonatsZahl = 6 'Brachet, Brachmond
Case "c", "d"
    sbMonatsZahl = 12 'Chrischtmonet, Dezember, December, Decembris, Christmond
Case "e"
    sbMonatsZahl = 8 'Erntemond
Case "f"
    sbMonatsZahl = 2 'Februar, February, Feber
Case "h"
    c2 = Mid(s, 2, 1)
    Select Case c2
    Case "a"
        sbMonatsZahl = 1 'Hartung
    Case "e"
        c3 = Mid(s, 3, 1)
        Select Case c3
        Case "r"
            sbMonatsZahl = 9 'Herbschtmonet, Herbstmond
        Case "u"
            sbMonatsZahl = 7 'Heuet, Heuert, Heumond
        Case Else
            sbMonatsZahl = CVErr(xlErrNum)
        End Select
    Case "o"
        sbMonatsZahl = 2 'Hornig, Hornung
    Case Else
        sbMonatsZahl = CVErr(xlErrNum)
    End Select
Case "j"
    c2 = Mid(s, 2, 1)
    Select Case c2
    Case "a", "ä", "e"
        sbMonatsZahl = 1 'Januar, January, Jänner, Jenner
    Case "u"
        c3 = Mid(s, 3, 1)
        Select Case c3
```

```
        Case "l"
            c4 = Mid(s, 4, 1)
            Select Case c4
            Case "e", "i", "y"
                sbMonatsZahl = 7 'Juli, July, Juley
            Case "m"
                sbMonatsZahl = 12 'Julmond
            Case Else
                sbMonatsZahl = CVErr(xlErrNum)
            End Select
        Case "n"
            sbMonatsZahl = 6 'Juni, June, Juno
        Case Else
            sbMonatsZahl = CVErr(xlErrNum)
        End Select
    Case Else
        sbMonatsZahl = CVErr(xlErrNum)
    End Select
Case "l"
    sbMonatsZahl = 3 'Lenzmond
Case "m"
    c2 = Mid(s, 2, 1)
    Select Case c2
    Case "a"
        c3 = Mid(s, 3, 1)
        Select Case c3
        Case "i", "y"
            sbMonatsZahl = 5 'Mai, May
        Case "r"
            sbMonatsZahl = 3 'March, Marty, Martii
        Case Else
            sbMonatsZahl = CVErr(xlErrNum)
        End Select
    Case "ä"
        c3 = Mid(s, 3, 1)
        Select Case c3
        Case "i"
            sbMonatsZahl = 5 'Mäie
        Case "r"
            sbMonatsZahl = 3 'März
        Case Else
            sbMonatsZahl = CVErr(xlErrNum)
        End Select
    Case "e"
        sbMonatsZahl = 3 'Merze
    Case Else
        sbMonatsZahl = CVErr(xlErrNum)
    End Select
Case "n"
    sbMonatsZahl = 11 'November, Nebelmond
Case "o"
    c2 = Mid(s, 2, 1)
    Select Case c2
    Case "c", "k"
        sbMonatsZahl = 10 'Oktober, October, Oktobris
    Case "s"
        sbMonatsZahl = 4 'Ostermond
    Case Else
        sbMonatsZahl = CVErr(xlErrNum)
    End Select
Case "s"
    c2 = Mid(s, 2, 1)
    Select Case c2
    Case "a"
        sbMonatsZahl = 4 'Saating
    Case "c"
        c3 = Mid(s, 3, 1)
        Select Case c3
        Case "h"
            c4 = Mid(s, 4, 1)
            Select Case c4
            Case "e"
                sbMonatsZahl = 9 'Scheiding
            Case "n"
                sbMonatsZahl = 1 'Schneemond
            Case Else
                sbMonatsZahl = CVErr(xlErrNum)
            End Select
        Case Else
            sbMonatsZahl = CVErr(xlErrNum)
        End Select
    Case "e"
        sbMonatsZahl = 9 'September, Septembris
    Case Else
        sbMonatsZahl = CVErr(xlErrNum)
    End Select
Case "w"
    sbMonatsZahl = 9 'September
    c2 = Mid(s, 2, 1)
    Select Case c2
```

```
    Case "e"
        sbMonatsZahl = 10 'Weinmond
    Case "i"
        c3 = Mid(s, 3, 1)
        Select Case c3
        Case "i"
            sbMonatsZahl = 10 'Wiimonet
        Case "n"
            sbMonatsZahl = 11 'Wintermonet
        Case Else
            sbMonatsZahl = CVErr(xlErrNum)
        End Select
    Case "o"
        sbMonatsZahl = 5 'Wonnemond
    Case Else
        sbMonatsZahl = CVErr(xlErrNum)
    End Select
Case Else
    sbMonatsZahl = CVErr(xlErrNum)
End Select
End Function
```

Die Berechnung der Kreiszahl π

Wie viele Dezimalstellen der Kreiszahl π werden für die Praxis gebraucht?

Der Hamburger Mathematikprofessor Hermann Schubert zeigte 1889, wieviel Stellen nicht mehr benötigt werden: Man stelle sich eine Kugel vor, in deren Mitte die Erde liegt. Der Radius sei so groß wie die Entfernung der Erde vom Sirius: 8,8 Lichtjahre. Diese Kugel sei mit Mikroben gefüllt, von denen viele Millionen in einen Kubikmillimeter passen. Nun fädele man alle Mikroben in dieser Kugel an einem straffen Faden auf und lasse zwischen je zwei Mikroben einen Platz von 8,8 Lichtjahren. Diesen Faden nehme man als Durchmesser eines Kreises. Multipliziert man diese Länge mit π (auf 100 Dezimalstellen genau), so weicht das Ergebnis vom exakten Umfang des Kreises weniger als ein Millionstel Millimeter ab.

Dieses Beispiel zeigt, daß die Berechnung von π auf 100 Stellen oder mehr vollkommen nutzlos ist.

Warum wird dann versucht, π auf immer mehr Stellen zu berechnen?

In jüngster Zeit möchte man erfahren, ob die Folge der Dezimalziffern gewissen Gesetzen genügt oder nicht. Auch ist nicht bekannt, wie häufig alle Ziffern in der Dezimaldarstellung vorkommen.

Bis in die Mitte des 17. Jahrhunderts wurde π nach der Methode des Archimedes berechnet. Er berechnete die Fläche des Einheitskreises näherungsweise mit Polygonflächen. James Gregory entdeckte 1671 die Potenzreihe

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - + \ldots$$

für -1 <= x <= 1.

Mit dieser Reihenentwicklung und der Gleichung

$$(*)\ \arctan x + \arctan y = \arctan \frac{x+y}{1-xy}$$

wird π in der jüngsten Zeit berechnet.

Herleitung von (*):

Sei $-\frac{\pi}{2} < \alpha + \beta < \frac{\pi}{2}$.

Das Additionstheorem für den Tangens lautet $\tan(\alpha + \beta) = \frac{\tan\alpha + \tan\beta}{1 - \tan\alpha \tan\beta}$.

Wird $\alpha = \arctan x, \beta = \arctan y$ gesetzt, so ergibt sich für xy ≠ 1

$\tan(\alpha + \beta) = \frac{x+y}{1-xy}$

so dass $\alpha + \beta = \arctan x + \arctan y = \arctan \frac{x+y}{1-xy}$ ist.

Eine Tabelle zur Geschichte der Näherungen von π

Jahr	Stellen	Name
-225	2	Archimedes
1579	9	Viete
1593	15	Roomen
1610	35	Ceulen
1699	72	Sharp
1719	112	de Lagny
1794	140	von Vega
1794	200	Dahse
1873	527	Shanks
945-48	808	Smith & Wrench
1949	2.037	Reitwiesner
954-55	3.089	Nicholson & Jeenel
1958	10.000	Genuys
1959	16.167	Genuys
1961	100.000	Shanks & Wrench
1966	250.000	Gilloud & Fillatoire
1967	500.000	Gilloud & Dichampt

Pi Programmcode

```
Option Explicit

Const n = 1050
Const z1 = 554
Const z2 = 285
Const z3 = 211
Dim m1            As Integer
Dim m2            As Integer
Dim m3            As Integer
Dim a(n)          As Integer
Dim u_b(n)        As Integer
Dim c(n)          As Integer
Dim d(n)          As Integer
Dim p(n)          As Integer
Dim i             As Integer
Dim j             As Integer
Dim k             As Integer
Dim r             As Integer
Dim u             As Integer
Dim v             As Integer
Dim x             As Integer
Dim y             As Integer

Sub addiere()
u = 0
For j = n To 0 Step -1
  p(j) = p(j) + a(j) + u
  u = p(j) \ 10
  p(j) = p(j) - 10 * u
Next j
End Sub

Sub subtrahiere()
u = 1
For j = n To 0 Step -1
  p(j) = p(j) + 9 - a(j) + u
  u = p(j) \ 10
  p(j) = p(j) - 10 * u
Next j
End Sub
```

```
Sub pi()

' pi auf 1000 Stellen ausgeben

For i = 0 To n
  a(i) = 0
  u_b(i) = 0
  c(i) = 0
  d(i) = 0
  p(i) = 0
Next i
m1 = 8
m2 = 57
m3 = 239
'Der erste Summand jeder Reihe wird ermittelt
'b(I) = 24 \ 8
u_b(0) = 24 \ 8
p(0) = u_b(0)
c(0) = 8
r = 0   'r enthält immer den Rest der Division
'c(I) = 8\57
For i = 0 To n
  a(i) = (10 * r + c(i)) \ m2
  r = 10 * r + c(i) - a(i) * m2
Next i
For i = 0 To n
  c(i) = a(i)
Next i
addiere
d(0) = 4
r = 0
'd(I) = 4\239
For i = 0 To n
  a(i) = (10 * r + d(i)) \ m3
  r = 10 * r + d(i) - a(i) * m3
Next i
For i = 0 To n
  d(i) = a(i)
Next i
addiere

' Nun wird die Reihe von 24 arctan 1\8 berechnet
v = -1  'Das Vorzeichen des Summanden
m1 = m1 * m1
k = 3
For i = 1 To z1
  r = 0
  For j = 0 To n
    a(j) = (10 * r + u_b(j)) \ m1
    r = 10 * r + u_b(j) - a(j) * m1
  Next j
  For j = 0 To n
    u_b(j) = a(j)
  Next j
  r = 0
  For j = 0 To n
    a(j) = (10 * r + u_b(j)) \ k
    r = 10 * r + u_b(j) - a(j) * k
  Next j
  If v = 1 Then addiere Else subtrahiere
  k = k + 2
  v = 0 - v
Next i

' Nun wird die Reihe von 8 arctan 1\57 berechnet
v = -1
m2 = m2 * m2
k = 3
For i = 1 To z2
  r = 0
  For j = 0 To n
    a(j) = (10 * r + c(j)) \ m2
    r = 10 * r + c(j) - a(j) * m2
  Next j
  For j = 0 To n
    c(j) = a(j)
  Next j
  r = 0
  For j = 0 To n
    a(j) = (10 * r + c(j)) \ k
    r = 10 * r + c(j) - a(j) * k
  Next j
  If v = 1 Then addiere Else subtrahiere
  k = k + 2
  v = 0 - v
Next i
' Nun wird die Reihe von 4 arctan 1\239 berechnet
v = -1
k = 3
For i = 1 To z3
```

```
  r = 0
  For j = 0 To n
    a(j) = (10 * r + d(j)) \ m3
    r = 10 * r + d(j) - a(j) * m3
  Next j
  r = 0
  For j = 0 To n
    d(j) = (10 * r + d(j)) \ m3
    r = 10 * r + a(j) - d(j) * m3
  Next j
  r = 0
  For j = 0 To n
    a(j) = (10 * r + d(j)) \ k
    r = 10 * r + d(j) - a(j) * k
  Next j
  If v = 1 Then addiere Else subtrahiere
  k = k + 2
  v = 0 - v
Next i

x = 1
y = 1

Open ThisWorkbook.Path & "/pi.txt" For Output As #1
Print #1, "Pi = " & p(0) & ".";
For i = 1 To 1000
  Print #1, Format(p(i), "&");
  x = x + 1
  If x > 3 Then
    Print #1, " ";
    x = 1
  End If
  y = y + 1
  If y > 42 Then
    Print #1, " "
    Print #1, "       ";
    y = 1
  End If
Next i
Close #1

End Sub
```

Pi Ausgabe auf 1000 Stellen

```
Pi = 3.141 592 653 589 793 238 462 643 383 279 502 884 197 169
       399 375 105 820 974 944 592 307 816 406 286 208 998 628
       034 825 342 117 067 982 148 086 513 282 306 647 093 844
       609 550 582 231 725 359 408 128 481 117 450 284 102 701
       938 521 105 559 644 622 948 954 930 381 964 428 810 975
       665 933 446 128 475 648 233 786 783 165 271 201 909 145
       648 566 923 460 348 610 454 326 648 213 393 607 260 249
       141 273 724 587 006 606 315 588 174 881 520 920 962 829
       254 091 715 364 367 892 590 360 011 330 530 548 820 466
       521 384 146 951 941 511 609 433 057 270 365 759 591 953
       092 186 117 381 932 611 793 105 118 548 074 462 379 962
       749 567 351 885 752 724 891 227 938 183 011 949 129 833
       673 362 440 656 643 086 021 394 946 395 224 737 190 702
       179 860 943 702 770 539 217 176 293 176 752 384 674 818
       467 669 405 132 000 568 127 145 263 560 827 785 771 342
       757 789 609 173 637 178 721 468 440 901 224 953 430 146
       549 585 371 050 792 279 689 258 923 542 019 956 112 129
       021 960 864 034 418 159 813 629 774 771 309 960 518 707
       211 349 999 998 372 978 049 951 059 731 732 816 096 318
       595 024 459 455 346 908 302 642 522 308 253 344 685 035
       261 931 188 171 010 003 137 838 752 886 587 533 208 381
       420 617 177 669 147 303 598 253 490 428 755 468 731 159
       562 863 882 353 787 593 751 957 781 857 780 532 171 226
       806 613 001 927 876 611 195 909 216 420 198 9
```

Die Berechnung der Eulerschen Zahl *e*

Zur Berechnung der Eulerschen Zahl *e* verwendet man die bekannte Summenformel

$$e = \sum_{i=0}^{\infty} \frac{1}{i!}$$

Also ist

$$e - 2 = \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \ldots$$

Jede reelle Zahl 0 <= x < 1 kann man darstellen als (möglicherweise unendlichen) “variablen Basis”-Bruch $.a_1a_2a_3\ldots$, wobei für alle i gilt: 0 <= a(i) < 1. Die dargestellte Zahl ist

$$x = \sum_{i=1}^{\infty} \frac{1}{(i+1)!}$$

Die reelle Zahl *e* - 2 wird demnach dargestellt durch den unendlichen Bruch .111… . Das Problem der Berechnung von *e* reduziert sich damit auf die Umwandlung der obigen Darstellung in Dezimaldarstellung. Je mehr Stellen man für die Dezimaldarstellung ermitteln will, desto “länger” muss der “variable Basis”-Bruch sein, den man umwandelt.

Beispiel: Berechnung von e auf 1000 Stellen. Ab welchem Index n kann man den Reihenrest vernachlässigen? Sei y(n) die (n+1)-te Partialsumme der unendlichen Reihe

$$\sum_{i=0}^{\infty} \frac{1}{i!}$$

Es gilt (Quelle: Fichtenholz Band I, Nr. 37):

$$0 < e - y_n < \frac{1}{n!n}$$

Für dieses Beispiel muss gelten:

$$e - y_n < \frac{1}{n!n} < 10^{-1000}$$

Dafür wähle man n = 500.

e Programmcode

```
Option Explicit

Sub e()

'e auf 1000 Stellen ausgeben

Dim a(500) As Long
Dim c      As Long
Dim d      As Long
Dim i      As Long
Dim j      As Long
Dim x      As Integer
Dim y      As Integer

Open ThisWorkbook.Path & "/e.txt" For Output As #1
Print #1, "e = 2.";

For i = 1 To 500
  a(i) = 1
Next i

For i = 1 To 1000
  c = 0
  For j = 500 To 1 Step -1
    d = 10 * a(j) + c
    c = Fix(d / (j + 1))
    a(j) = d - c * (j + 1)
  Next j
  x = x + 1
  If x > 3 Then
    Print #1, " ";
    x = 1
  End If
  y = y + 1
  If y > 42 Then
    Print #1, " "
    Print #1, "      ";
    y = 1
  End If
  Print #1, Format(c, "&");
Next i

Close #1

End Sub
```

e Ausgabe auf 1000 Stellen

```
e = 2.718 281 828 459 045 235 360 287 471 352 662 497 757 247
      093 699 959 574 966 967 627 724 076 630 353 547 594 571
      382 178 525 166 427 427 466 391 932 003 059 921 817 413
      596 629 043 572 900 334 295 260 595 630 738 132 328 627
      943 490 763 233 829 880 753 195 251 019 011 573 834 187
      930 702 154 089 149 934 884 167 509 244 761 460 668 082
      264 800 168 477 411 853 742 345 442 437 107 539 077 744
      992 069 551 702 761 838 606 261 331 384 583 000 752 044
      933 826 560 297 606 737 113 200 709 328 709 127 443 747
      047 230 696 977 209 310 141 692 836 819 025 515 108 657
      463 772 111 252 389 784 425 056 953 696 770 785 449 969
      967 946 864 454 905 987 931 636 889 230 098 793 127 736
      178 215 424 999 229 576 351 482 208 269 895 193 668 033
      182 528 869 398 496 465 105 820 939 239 829 488 793 320
      362 509 443 117 301 238 197 068 416 140 397 019 837 679
      320 683 282 376 464 804 295 311 802 328 782 509 819 455
      815 301 756 717 361 332 069 811 250 996 181 881 593 041
      690 351 598 888 519 345 807 273 866 738 589 422 879 228
      499 892 086 805 825 749 279 610 484 198 444 363 463 244
      968 487 560 233 624 827 041 978 623 209 002 160 990 235
      304 369 941 849 146 314 093 431 738 143 640 546 253 152
      096 183 690 888 707 016 768 396 424 378 140 592 714 563
      549 061 303 107 208 510 383 750 510 115 747 704 171 898
      610 687 396 965 521 267 154 688 957 035 035 4
```

Literatur

Nievergelt, Farrer, Reingold: Computer Approaches to Mathematical Problems; Prentice Hall, Inc. 1974, p. 191 + 192, p. 198 - 202.

Ullman: Fundamental Concepts of Programming Systems; Addison-Wesley Publishing Company 1976, p. 48 + 49.

Fichtenholz; Differential- und Integralrechnung Band I + II; VEB Deutscher Verlag der Wissenschaften 1981, Nr. 37 + 50 + 410.

Zahlenfolge kürzer darstellen – *sbParseNumSeq*

sbParseNumSeq Gliedere eine Zahlenfolge auf und gib eine kürzere Darstellung zurück: 1,2,3,5,6,7 wird zurückgegeben als 1-3,5-7. Falls *bWithSingleDouble* = TRUE, dann wird 1,3,5,6,8,10 zurückgegeben als 1-5(single),6-10(double).

	A	B	C
1	**Input**	**bWithSingleDouble**	**Output**
2	1,2,3,4,8,11,12,16,17,18	WAHR	1-4,8,11-12,16-18
3	3,5,6,7,9,12,13,14,15,20,101	WAHR	3,5-7,9,12-15,20,101
4	1,3,5,7,9,11	WAHR	1-11(single)
5	1	WAHR	1
6	2,4,6,8,10,12,14,16,17	WAHR	2-16(double),17
7	1,5,6,7,9,12,13,14,15,16	WAHR	1,5-7,9,12-16
8	2,3,4,5,6,7,10,17,22,23,24,25,26,77	WAHR	2-7,10,17,22-26,77
9	3,4,7,13,15,16,17	WAHR	3-4,7,13,15-17
10	1,2,3,4,5,6,7,13,15,17	WAHR	1-7,13-17(single)
11	3,5,7,13,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78,80	WAHR	3-7(single),13,22-80(double)
12	2,3,4,7,12,14,17,18	WAHR	2-4,7,12-14(double),17-18
13	1,3,4,5,7,9,11,12,14,16	WAHR	1,3-4,5-11(single),12-16(double)
14	3,4,5,7,9,11	WAHR	3-4,5-11(single)
15	1,2,3,4,8,11,12,16,17,18	FALSCH	1-4,8,11-12,16-18
16	3,5,6,7,9,12,13,14,15,20,101	FALSCH	3,5-7,9,12-15,20,101
17	1,3,5,7,9,11	FALSCH	1,3,5,7,9,11
18	1	FALSCH	1
19	2,4,6,8,10,12,14,16,17	FALSCH	2,4,6,8,10,12,14,16-17
20	1,5,6,7,9,12,13,14,15,16	FALSCH	1,5-7,9,12-16
21	2,3,4,5,6,7,10,17,22,23,24,25,26,77	FALSCH	2-7,10,17,22-26,77
22	3,4,7,13,15,16,17	FALSCH	3-4,7,13,15-17
23	1,2,3,4,5,6,7,13,15,17	FALSCH	1-7,13,15,17
24	3,5,7,13,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78,80	FALSCH	3,5,7,13,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78,80
25	2,3,4,7,12,14,17,18	FALSCH	2-4,7,12,14,17-18
26	1,3,4,5,7,9,11,12,14,16	FALSCH	1,3-5,7,9,11-12,14,16
27	3,4,5,7,9,11	FALSCH	3-5,7,9,11

sbParseNumSeq Programmcode

```
Function sbParseNumSeq(s As String, _
  Optional bWithSingleDouble As Boolean = True) As String
'Parse a comma-separated number sequence and return a
'shortened representation:
'1,2,3,5,6,7 will result in 1-3,5-7.
'If bWithSingleDouble = TRUE then
'1,3,5,6,8,10 will result in 1-5(single),6-10(double).
'(C) (P) by Bernd Plumhoff 08-Sep-2024 PB V0.1
Dim i           As Long
Dim j           As Long
Dim k           As Long
Dim m           As Long
Dim sDel        As String
Dim suffix      As String
Dim r           As String
Dim v           As Variant

v = Split(s, ",")
j = UBound(v)
ReDim seq(0 To j, 0 To 2) As Long
For i = 0 To j - 1
  k = v(i + 1)
  If k = v(i) + 1 Then
    m = i + 1
    Do While m < j
      If v(m) + 1 = CLng(v(m + 1)) Then
        m = m + 1
      Else
        Exit Do
      End If
    Loop
    seq(i, 0) = 1
    seq(i, 1) = m - i
  ElseIf bWithSingleDouble And k = v(i) + 2 Then
    m = i + 1
    Do While m < j
      If v(m) + 2 = CLng(v(m + 1)) Then
        m = m + 1
      Else
        Exit Do
      End If
    Loop
    seq(i, 0) = 2
    seq(i, 2) = m - i
  End If
Next i
For i = 0 To j
  If seq(i, 0) = 0 Then
    r = r & sDel & v(i)
  Else
    k = seq(i, seq(i, 0))
    m = seq(i + k, seq(i + k, 0))
    If k > 0 And k >= m Then
      suffix = ""
      If seq(i, 0) = 2 Then
        If v(i) Mod 2 = 0 Then
          suffix = "(double)"
        Else
          suffix = "(single)"
        End If
      End If
      r = r & sDel & v(i) & "-" & v(i + k) & suffix
      i = i + k
    ElseIf k >= 2 Then
      suffix = ""
      If seq(i, 0) = 2 Then
        If v(i) Mod 2 = 0 Then
          suffix = "(double)"
        Else
          suffix = "(single)"
        End If
      End If
      r = r & sDel & v(i) & "-" & v(i + k - 1) & suffix
      i = i + k - 1
    Else
      r = r & sDel & v(i)
    End If
  End If
  sDel = ","
Next i
sbParseNumSeq = r
End Function
```

Rationale Zahlen = Brüche

Ermittle die nächstliegende rationale Zahl zu einer Gleitkommazahl – *sbNRN*

Welche rationale Zahl ist eine gute Näherung von π (3,1415926…)? Geben Sie ein: in Zelle A1 '=pi()', in Zelle B1 den maximal gewünschten Nenner (zum Beispiel 10), und als Matrixformel (mit STRG + SHIFT + ENTER) in den Zellen C1:D1 '=*sbNRN*(A1;B1)'. Als Ergebnis erscheint in C1:D1 22 und 7. Dies bedeutet: 22/7 ist die nächstliegende rationale Zahl (Bruch) zu π mit einem Nenner nicht größer als 10. Mit 1000 in B1 würde man 355/113 erhalten.

Dieser Algorithmus findet nicht immer die nächstliegende rationale Zahl zu einer gegebenen Gleitkommazahl mit einem gewünschten maximalen Nenner und der vordefinierten maximalen Fehlerschranke 1# / (2# * CDbl(lMaxDen) ^ 2#). Die gute Nachricht ist jedoch, dass er dann einen #ZAHL! Fehler zurückgeben würde. In einem solchen Fall geben Sie bitte eine größere individuelle maximale Fehlerschranke vor.

Die ursprüngliche Absicht des Autoren Oliver Aberth bestand in der Unterstützung exakter Berechnungen von Brüchen, zum Beispiel bei der Lösung von linearen Gleichungssystemen mit rationalen Koeffizienten.

Input			Result		Quality Measure			
dFloat	lMaxDen	dMaxErr	pK	qK	Absolute Error	# of "?"	TEXT Representation	Comment
3,14159265358979	1		3	1	0,141592654	1	22/7	
3,14159265358979	10		22	7	0,001264489	1	22/7	
3,14159265358979	112	0,001264489	333	106	8,32196E-05	3	355/113	
3,14159265358979	1000		355	113	2,66764E-07	3	355/113	
3,14159265358979	33214	2,66764E-07	103993	33102	5,77891E-10	5	312689/99532	
3,14159265358979	66316	5,77891E-10	104348	33215	3,31628E-10	5	312689/99532	
3,14159265358979	99531	3,31628E-10	208341	66317	1,22356E-10	5	312689/99532	
3,14159265358979	100000		312689	99532	2,91434E-11	5	312689/99532	
3,14159265358979	364912	2,91434E-11	833719	265381	8,71525E-12	6	1146408/364913	
3,14159265358979	1360119	8,71525E-12	1146408	364913	1,61071E-12	6	1146408/364913	
3,14159265358979	1725032	1,61071E-12	4272943	1360120	4,04121E-13	7	5419351/1725033	
3,14159265358979	25510581	4,04121E-13	5419351	1725033	2,22045E-14	7	5419351/1725033	
3,14159265358979	78256778	2,22045E-14	80143857	25510582	4,44089E-16	8	5419351/1725033	Accuracy limit of TEXT reached
3,14159265358979	100000000		245850922	78256779	0	8	5419351/1725033	Accuracy limit of TEXT reached

Bemerkung: Die letzte Zeile in der obigen Grafik sagt uns nicht, dass wir den Kreis erfolgreich quadriert haben. Wir haben lediglich (meines) Excel's Genauigkeitsgrenze erreicht.

Die Bruchdarstellungen der TEXT Funktion wurden zum Vergleich angezeigt. Beispiel: =TEXT(PI();"?/?") = "22/7". Microsoft hat diese Darstellung nicht für die 64-Bit Version erweitert. Genauer als PI() = "5419351/1725033" kann nicht gezeigt werden. Genauer wäre mit 64-Bit PI() = "245850922/78256779", aber dann ist der absolute Fehler ist natürlich längst kleiner als 1e-15.

<u>Grenzen der Berechnung</u>

Excel kann Dezimalzahlen von -9,99999999999999E+307 bis 9,99999999999999E+307 darstellen. Die 64-Bit Version von Excel kann ganze Zahlen des Typs LongLong von -9223372036854775808 bis 9223372036854775807 darstellen, was in etwa dem Umfang -1E+10 bis 1E+10 entspricht. Es ist offensichtlich, dass Aberth's Algorithmus in Excel nicht alle verfügbaren Dezimalzahlen hinreichend genau als Bruch ermitteln kann.

Name

sbNRN - Berechne die nächstliegende rationale Zahl zu einer gegebenen Gleitkommazahl mit einem gegebenen maximalen Nenner

Synopsis

sbNRN(*dFloat*, *lMaxDen*, [*dMaxErr*])

Beschreibung

sbNRN berechnet die nächstliegende rationale Zahl zu der gegebenen Gleitkommazahl *dFloat* mit dem maximalen Nenner *lMaxDen* und der optionalen maximalen absoluten Fehlerschranke *dMaxErr*.

Parameter

dFloat - Die Gleitkommazahl für die die nächstliegende rationale Zahl gefunden werden soll

lMaxDen - Die Obergrenze für den Nenner

dMaxErr - Optional - Die Obergrenze für den absoluten Fehler (absolute Differenz zwischen der eingegebenen Gleitkommazahl und der auszugebenden rationalen Zahl)

Literatur

Oliver Aberth, A method for exact computation with rational numbers, JCAM, vol 4, no. 4, 1978

Oliver Aberth, Introduction to Precise Numerical Methods, ISBN 0-12-373859-8

George Chrystal, Algebra an Elementary Text-Book, Part II, Chapter 32, p. 423 ff, 1900

Peter Henrici, A Subroutine for Computations with Rational Numbers, JACM, vol 3, no. 1, 1956

Exkurs

Falls Sie lediglich die Relation zur Zehnerpotenz benötigen, verwenden Sie die folgende Formel:

```
=WENNFEHLER(-A2*10^(LÄNGE(-A2)-SUCHEN(",";-A2))&":"&10^(LÄNGE(-A2)-SUCHEN(",";-A2));-A2&":1")
```

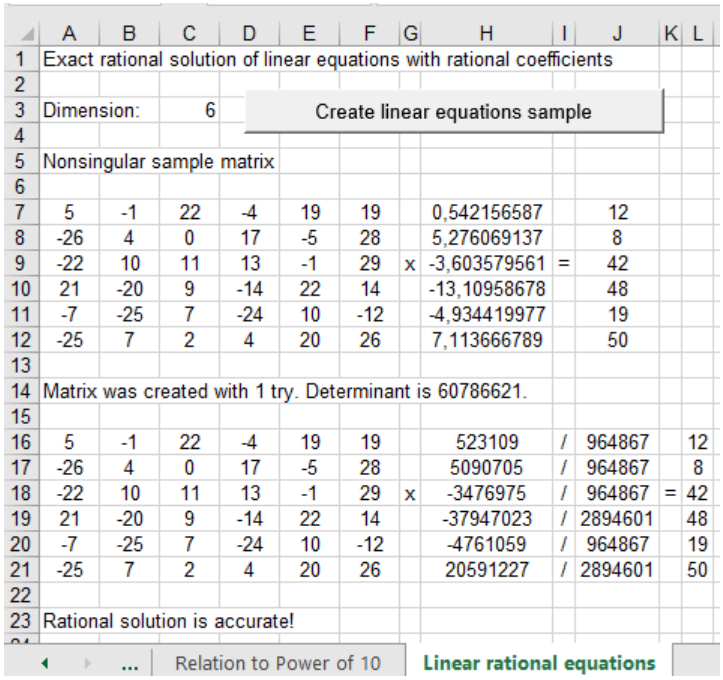
	A	B	C
1	**Eingabe**	**Ausgabe**	**Formel in B**
2	1E-14	1:100000000000000	=WENNFEHLER(--A2*10^(LÄNGE(--A2)-SUCHEN(",";--A2))&":"&10^(LÄNGE(--A2)-SUCHEN(",";--A2));--A2&":1")
3	0,00001	1:100000	=WENNFEHLER(--A3*10^(LÄNGE(--A3)-SUCHEN(",";--A3))&":"&10^(LÄNGE(--A3)-SUCHEN(",";--A3));--A3&":1")
4	0,1	1:10	=WENNFEHLER(--A4*10^(LÄNGE(--A4)-SUCHEN(",";--A4))&":"&10^(LÄNGE(--A4)-SUCHEN(",";--A4));--A4&":1")
5	0,2	2:10	=WENNFEHLER(--A5*10^(LÄNGE(--A5)-SUCHEN(",";--A5))&":"&10^(LÄNGE(--A5)-SUCHEN(",";--A5));--A5&":1")
6	0,22	22:100	=WENNFEHLER(--A6*10^(LÄNGE(--A6)-SUCHEN(",";--A6))&":"&10^(LÄNGE(--A6)-SUCHEN(",";--A6));--A6&":1")
7	0,0001234	1234:10000000	=WENNFEHLER(--A7*10^(LÄNGE(--A7)-SUCHEN(",";--A7))&":"&10^(LÄNGE(--A7)-SUCHEN(",";--A7));--A7&":1")
8	0	0:1	=WENNFEHLER(--A8*10^(LÄNGE(--A8)-SUCHEN(",";--A8))&":"&10^(LÄNGE(--A8)-SUCHEN(",";--A8));--A8&":1")
9	1	1:1	=WENNFEHLER(--A9*10^(LÄNGE(--A9)-SUCHEN(",";--A9))&":"&10^(LÄNGE(--A9)-SUCHEN(",";--A9));--A9&":1")
10	10	10:1	=WENNFEHLER(--A10*10^(LÄNGE(--A10)-SUCHEN(",";--A10))&":"&10^(LÄNGE(--A10)-SUCHEN(",";--A10));--A10&":1")
11	100	100:1	=WENNFEHLER(--A11*10^(LÄNGE(--A11)-SUCHEN(",";--A11))&":"&10^(LÄNGE(--A11)-SUCHEN(",";--A11));--A11&":1")
12	3,141592654	314159265358979:100000000000000	=WENNFEHLER(--A12*10^(LÄNGE(--A12)-SUCHEN(",";--A12))&":"&10^(LÄNGE(--A12)-SUCHEN(",";--A12));--A12&":1")
13	1,1	11:10	=WENNFEHLER(--A13*10^(LÄNGE(--A13)-SUCHEN(",";--A13))&":"&10^(LÄNGE(--A13)-SUCHEN(",";--A13));--A13&":1")
14	12,12	1212:100	=WENNFEHLER(--A14*10^(LÄNGE(--A14)-SUCHEN(",";--A14))&":"&10^(LÄNGE(--A14)-SUCHEN(",";--A14));--A14&":1")
15	123,123	123123:1000	=WENNFEHLER(--A15*10^(LÄNGE(--A15)-SUCHEN(",";--A15))&":"&10^(LÄNGE(--A15)-SUCHEN(",";--A15));--A15&":1")
16	1E+200	1E+200:1	=WENNFEHLER(--A16*10^(LÄNGE(--A16)-SUCHEN(",";--A16))&":"&10^(LÄNGE(--A16)-SUCHEN(",";--A16));--A16&":1")
17	1E-200	1E-200:1	=WENNFEHLER(--A17*10^(LÄNGE(--A17)-SUCHEN(",";--A17))&":"&10^(LÄNGE(--A17)-SUCHEN(",";--A17));--A17&":1")
18	-0,1	-1:10	=WENNFEHLER(--A18*10^(LÄNGE(--A18)-SUCHEN(",";--A18))&":"&10^(LÄNGE(--A18)-SUCHEN(",";--A18));--A18&":1")
19	-3,141592654	-314159265358979:100000000000000	=WENNFEHLER(--A19*10^(LÄNGE(--A19)-SUCHEN(",";--A19))&":"&10^(LÄNGE(--A19)-SUCHEN(",";--A19));--A19&":1")
20	-123,123	-123123:1000	=WENNFEHLER(--A20*10^(LÄNGE(--A20)-SUCHEN(",";--A20))&":"&10^(LÄNGE(--A20)-SUCHEN(",";--A20));--A20&":1")
21	-12,12	-1212:100	=WENNFEHLER(--A21*10^(LÄNGE(--A21)-SUCHEN(",";--A21))&":"&10^(LÄNGE(--A21)-SUCHEN(",";--A21));--A21&":1")
22	-0,00004	-4:100000	=WENNFEHLER(--A22*10^(LÄNGE(--A22)-SUCHEN(",";--A22))&":"&10^(LÄNGE(--A22)-SUCHEN(",";--A22));--A22&":1")

sbNRN Programmcode

```
Option Explicit

#If Win64 Then
Function sbNRN(dFloat As Double, lMaxDen As LongLong, _
               Optional dMaxErr As Double = -1#) As Variant
#Else
Function sbNRN(dFloat As Double, lMaxDen As Long, _
               Optional dMaxErr As Double = -1#) As Variant
#End If
'Computes nearest rational number to dFloat with a maximal denominator
'lMaxDen and a maximal absolute error dMaxErr and returns result as a
'variant Nominator / Denominator.
'See: Oliver Aberth, A method for exact computation with rational numbers,
'     JCAM, vol 4, no. 4, 1978
'Bernd Plumhoff V1.21 09-Oct-2020

Dim dB As Double
#If Win64 Then
Dim lA As LongLong, lSgn As LongLong
Dim lP1 As LongLong, lP2 As LongLong, lP3 As LongLong
Dim lQ1 As LongLong, lQ2 As LongLong, lQ3 As LongLong
#Else
Dim lA As Long, lSgn As Long
Dim lP1 As Long, lP2 As Long, lP3 As Long
Dim lQ1 As Long, lQ2 As Long, lQ3 As Long
#End If

If dMaxErr = -1# Then dMaxErr = 1# / (2# * CDbl(lMaxDen) ^ 2#)
lSgn = Sgn(dFloat): dB = Abs(dFloat)
lP1 = 0: lP2 = 1: lQ1 = 1: lQ2 = 0

Do While lMaxDen > lQ2
    lA = Int(dB)
    lP3 = lA * lP2 + lP1: lQ3 = lA * lQ2 + lQ1
#If Win64 Then
    If Abs(dB - CDbl(lA)) < 1# / CLngLng("9223372036854775807") Then
#Else
    If Abs(dB - CDbl(lA)) < 1# / 2147483647# Then
#End If
        Exit Do
    End If
    dB = 1# / (dB - CDbl(lA))
    lP1 = lP2: lP2 = lP3: lQ1 = lQ2: lQ2 = lQ3
Loop

If lQ3 > lMaxDen Then
    lQ3 = lQ2: lP3 = lP2
    If lQ2 > lMaxDen Then
        lQ3 = lQ1: lP3 = lP1
    End If
End If

'If absolute error exceeds 1/2Q^2 then Aberth's lemma p. 286 might not apply.
'But the user can override this and check the result himself.
If Abs(dFloat - lSgn * lP3 / lQ3) > dMaxErr Then
    sbNRN = CVErr(xlErrNum)
Else
    sbNRN = Array(lSgn * lP3, lQ3)
End If

End Function
```

Lineare Gleichungssysteme mit rationalen Koeffizienten

Lineare Gleichungssysteme der Form A * x = b mit der regulären quadratischen Matrix A und dem Ergebnisvektor b haben eine eindeutige Lösung, da die Determinante von A ungleich Null ist. Sind die Koeffizienten von A und b rationale Zahlen, so ist auch die Lösung rational.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Exact rational solution of linear equations with rational coefficients											
2												
3	Dimension:		6		Create linear equations sample							
4												
5	Nonsingular sample matrix											
6												
7	5	-1	22	-4	19	19		0,542156587		12		
8	-26	4	0	17	-5	28		5,276069137		8		
9	-22	10	11	13	-1	29	x	-3,603579561	=	42		
10	21	-20	9	-14	22	14		-13,10958678		48		
11	-7	-25	7	-24	10	-12		-4,934419977		19		
12	-25	7	2	4	20	26		7,113666789		50		
13												
14	Matrix was created with 1 try. Determinant is 60786621.											
15												
16	5	-1	22	-4	19	19		523109	/	964867		12
17	-26	4	0	17	-5	28		5090705	/	964867		8
18	-22	10	11	13	-1	29	x	-3476975	/	964867	=	42
19	21	-20	9	-14	22	14		-37947023	/	2894601		48
20	-7	-25	7	-24	10	-12		-4761059	/	964867		19
21	-25	7	2	4	20	26		20591227	/	2894601		50
22												
23	Rational solution is accurate!											

... | Relation to Power of 10 | **Linear rational equations**

Beispiel Programmcode

```
Option Explicit

Sub Generate_linear_equations_and_solve()
'Calculates next rational numbers to the double-precision solution of
'given linear equations. Accuracy of rational solution is then reported.
'(C) (P) by Bernd Plumhoff 05-Jun-2024 PB V0.1

Dim bAccurate          As Boolean
Dim abserr             As Double
Dim d                  As Double
Dim det                As Double
Dim i                  As Long
Dim iter               As Long
Dim j                  As Long
Dim k                  As Long
Dim loc                As Long
Dim n                  As Long
Dim state As SystemState

With Application.WorksheetFunction
Set state = New SystemState
n = Range("Matrix_Dimension")
If n < 2 Or n > 52 Then '52 is max dimension of MInverse
  Call MsgBox("Dimension must be between 2 and 52!", vbOKOnly, "Error")
  Exit Sub
End If
loc = Range("Sample_Matrix").Row + 2
wsLEQ.Rows(loc & ":" & 1000000).Delete
ReDim m(1 To n, 1 To n) As Variant
det = 0#
iter = 0
Do While det = 0# And iter < 20
  iter = iter + 1
  For i = 1 To n
    For j = 1 To n
      m(i, j) = Int(Rnd * 10 * n) - 5 * n
```

```
    Next j
  Next i
  det = .MDeterm(m)
Loop
If det = 0# Then
  Call MsgBox("Determinant was still 0 after 20 tries. Check the algorithm, please.", vbOKOnly, "Error")
  Exit Sub
End If
Range(wsLEQ.Cells(loc, 1), wsLEQ.Cells(loc + n - 1, n)) = m
ReDim b(1 To n) As Variant
For i = 1 To n
  b(i) = Int(Rnd * 10 * n)
  wsLEQ.Cells(loc + i - 1, n + 2) = "X" & i
Next i
wsLEQ.Cells(loc + (n - 1) \ 2, n + 1) = "x"
Range(wsLEQ.Cells(loc, n + 4), wsLEQ.Cells(loc + n - 1, n + 4)) = .Transpose(b)
wsLEQ.Cells(loc + (n - 1) \ 2, n + 3) = "="

ReDim mInv(1 To n, 1 To n) As Variant
mInv = .MInverse(m)
ReDim X(1 To n) As Variant
X = .MMult(mInv, .Transpose(b))
Range(wsLEQ.Cells(loc, n + 2), wsLEQ.Cells(loc + n - 1, n + 2)) = X
wsLEQ.Rows(loc & ":" & loc + n - 1).HorizontalAlignment = xlCenter
wsLEQ.Cells(loc + n + 1, 1) = "Matrix was created with " & iter & IIf(iter = 1, " try", " tries") & _
  ". Determinant is " & det & "."
'Just a check whether the design was ok. Commented out after successful check:
'ReDim bCheck(1 To n) As Variant
'bCheck = .MMult(m, x) 'For n=7 we could also manually have entered into cell M7: =MMULT(A7:G13;I7:I13)
'Range(wsLEQ.Cells(loc, n + 5), wsLEQ.Cells(loc + n - 1, n + 5)) = bCheck

ReDim xR(1 To n) As Variant
ReDim xCheck(1 To n, 1 To 1) As Variant
For i = 1 To n
  d = X(i, 1)
  xR(i) = sbNRN(d, 100000000#, 0.00000001)
  xCheck(i, 1) = xR(i)(0) / xR(i)(1)
Next i
'Now show the rational solution and whether it is accurate.
ReDim bCheck(1 To n) As Variant
bCheck = .MMult(m, xCheck)

Range(wsLEQ.Cells(loc + n + 3, 1), wsLEQ.Cells(loc + 2 * n + 2, n)) = m
wsLEQ.Cells(loc + n + 3 + (n - 1) \ 2, n + 1) = "x"
abserr = 0#
For i = 1 To n
  wsLEQ.Cells(loc + i + n + 2, n + 2) = xR(i)(0)
  wsLEQ.Cells(loc + i + n + 2, n + 3) = "/"
  wsLEQ.Cells(loc + i + n + 2, n + 4) = xR(i)(1)
  abserr = abserr + Abs(X(i, 1) - xR(i)(0) / xR(i)(1))
Next i
wsLEQ.Cells(loc + n + 3 + (n - 1) \ 2, n + 5) = "="
Range(wsLEQ.Cells(loc + n + 3, n + 6), wsLEQ.Cells(loc + 2 * n + 2, n + 6)) = bCheck
wsLEQ.Rows(loc + n + 3 & ":" & loc + 2 * n + 2).HorizontalAlignment = xlCenter
wsLEQ.Cells(loc + 2 * n + 4, 1) = "Rational solution is " & _
  IIf(abserr < 0.000000000001, "fairly accurate", "off by " & abserr) & "."
Range(wsLEQ.Cells(1, 1), wsLEQ.Cells(1, n)).EntireColumn.ColumnWidth = 5
Range(wsLEQ.Cells(1, n + 1), wsLEQ.Cells(1, n + 6)).EntireColumn.AutoFit
If n < 10 Then
  bAccurate = True
  ReDim b2(1 To n) As String
  Dim rT As String
  ReDim r2(1 To n) As String
  For i = 1 To n
    b2(i) = -b(i)
    r2(i) = "0"
    For j = 1 To n
      rT = xR(j)(0)
      b2(i) = sbMult(b2(i), xR(j)(1))
      'Debug.Print i, j, "rT = " & rT, "b2(" & i & ") = " & b2(i)
      For k = 1 To n
        If j <> k Then
          rT = sbMult(rT, xR(k)(1))
          'Debug.Print i, j, k, "rT = " & rT
        End If
      Next k
      r2(i) = sbPlus(r2(i), sbMult(m(i, j), rT))
      'Debug.Print i, j, "r2(" & i & ") = " & r2(i)
    Next j
    r2(i) = sbPlus(r2(i), b2(i))
    'Debug.Print i, "r2(" & i & ") = " & r2(i)
    If r2(i) <> "0" Then bAccurate = False
  Next
```

```
  If bAccurate Then wsLEQ.Cells(loc + 2 * n + 4, 1) = "Rational solution is accurate!"
End If

End With
End Sub

Function sbMult(ByVal a As String, ByVal b As String) As String
'Multiplication of big integers.
'(C) (P) by Bernd Plumhoff 05-Jun-2024 PB V0.1
Dim c As String, s As String, r As String
Dim i As Long, j As Long, k As Long, carry As Long, m As Long
With Application.WorksheetFunction
If Left(a, 1) = "-" And Left(b, 1) = "-" Then
  a = Right(a, Len(a) - 1)
  b = Right(b, Len(b) - 1)
End If
If Left(a, 1) = "-" Then
  s = "-"
  a = Right(a, Len(a) - 1)
End If
If Left(b, 1) = "-" Then
  s = "-"
  b = Right(b, Len(b) - 1)
End If
j = Len(a) + Len(b) + 1
a = Right(String(j, "0") & a, j)
b = Right(String(j, "0") & b, j)
r = "0"
carry = 0
For i = j To 1 Step -1
  c = String(j - i, "0")
  For m = j To 1 Step -1
    k = CLng(Mid(a, i, 1)) * CLng(Mid(b, m, 1)) + carry
    c = CStr(k Mod 10) & c
    carry = k \ 10
  Next m
  r = sbPlus(r, c)
Next i
For i = 1 To j - 1
  If Mid(r, i, 1) <> "0" Then Exit For
Next i
If i <= j Then r = Right(r, j - i + 1) '
sbMult = IIf(r = "0", "0", s & r)
End With
End Function

Function sbPlus(ByVal a As String, ByVal b As String) As String
'Addition of big integers.
'(C) (P) by Bernd Plumhoff 05-Jun-2024 PB V0.1
Dim bNega As Boolean, bNegb As Boolean
Dim c As String, s As String
Dim i As Long, j As Long, k As Long, carry As Long, negcarry As Long
With Application.WorksheetFunction
bNega = False
bNegb = False
If Left(a, 1) = "-" And Left(b, 1) = "-" Then
  s = "-"
  a = Right(a, Len(a) - 1)
  b = Right(b, Len(b) - 1)
End If
If Left(a, 1) = "-" Then
  bNega = True
  a = Right(a, Len(a) - 1)
  If Len(b) < Len(a) Or (Len(b) = Len(a) And b < a) Then
    bNega = False
    bNegb = True
    s = "-"
  End If
End If
If Left(b, 1) = "-" Then
  bNegb = True
  b = Right(b, Len(b) - 1)
  If Len(b) > Len(a) Or (Len(b) = Len(a) And b > a) Then
    bNega = True
    bNegb = False
    s = "-"
  End If
End If
j = .Max(Len(a), Len(b)) + 1
a = Right(String(j, "0") & a, j)
b = Right(String(j, "0") & b, j)
c = ""
carry = 0
```

```
For i = j To 1 Step -1
  k = IIf(bNega, 10 - CLng(Mid(a, i, 1)), CLng(Mid(a, i, 1))) + _
      IIf(bNegb, 10 - CLng(Mid(b, i, 1)), CLng(Mid(b, i, 1))) + _
      carry + negcarry
  c = CStr(k Mod 10) & c
  carry = k \ 10
  negcarry = bNega + bNegb
Next i
For i = 1 To j - 1
  If Mid(c, i, 1) <> "0" Then Exit For
Next i
If i <= j Then c = Right(c, j - i + 1)
sbPlus = IIf(c = "0", "0", s & c)
End With
End Function
```

Anteilsveränderung als Bruch

Manchmal müssen Sie Anteilsveränderungen leicht verständlich darstellen. Das können Sie mit Brüchen zeigen.

Beispiel: Eine Erbengemeinschaft besteht aus den Herren Müller, Meier und Schulz. Meier verstirbt ohne Erben, sein Anteil wird aufgeteilt. Schulz stirbt auch, sein Anteil geht zu 2/3 an seine Witwe und zu 1/3 an sein einziges Kind. Bitte beachten Sie, dass Sie diese Anteile mit =1/3 * 2/3 respektive =1/3 * 1/3 eingeben müssen, wobei das erste 1/3 den Originalanteil von Schulz darstellt.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Anteil alt	Als Bruch	Anteil alt normiert	Als Bruch	Anteil neu	Als Bruch	Anteil neu normiert	Als Bruch	Größter Nenner	Nenner gleich	Veränderung	Als Bruch	Größter Nenner	Nenner gleich
2	*Gesamt*	*100,00%*	*1*	*100,00%*	*1*	*66,67%*	*2/3*	*100,00%*	*1*	*6*	*6/6*			*6*	
3	Müller	33,33%	1/3	33,33%	1/3	33,33%	1/3	50,00%	1/2	2	3/6	16,67%	1/6	6	1/6
4	Meier	33,33%	1/3	33,33%	1/3							-33,33%	- 1/3	3	-2/6
5	Schulz	33,33%	1/3	33,33%	1/3							-33,33%	- 1/3	3	-2/6
6	Schulz Witwe					22,22%	2/9	33,33%	1/3	3	2/6	33,33%	1/3	3	2/6
7	Schulz Kind					11,11%	1/9	16,67%	1/6	6	1/6	16,67%	1/6	6	1/6
8															
9															
10															
11															
12															
13															
14	Hinweis:	Nur in diese gefärbten Zellen Werte eingeben!													

Die Formeln

Diese Aufgabe lässt sich mit Excel Formeln lösen. indem Sie Excel's interne Bruchdarstellung als Textformat verwenden:

	A	B	C	D	E	F	G
1	Name	Anteil alt	Als Bruch	Anteil alt normiert	Als Bruch	Anteil neu	Als Bruch
2	*Gesamt*	*=SUMME(B3:B12)*	*=WENN(B2:B12=0;"";TEXT(B2:B12;"# ???/???"))*	*=SUMME(D3:D12)*	*=WENN(D2:D12=0;"";TEXT(D2:D12;"# ???/???"))*	*=SUMME(F3:F12)*	*=WENN(F2:F12=0;"";TEXT(F2:F12;"# ???/???"))*
3	Müller	=1/3		=B3:B12/B2		=1/3	

	H	I	J	K
1	Anteil neu normiert	Als Bruch	Größter Nenner	Nenner gleich
2	*=SUMME(H3:H12)*	*=WENN(H2:H12=0;"";TEXT(H2:H12;"# ???/???"))*	*=MAX(J3:J12)*	*=WENN(H2:H12<>0;RUNDEN(H2:H12*J2;0) & "/" & J2;"")*
3	=F3:F12/F2		=WERT("0" & RECHTS(I3:I12;LÄNGE(I3:I12)-WENNFEHLER(SUCHEN("/";I3:I12);0)))	

	L	M	N	O
1	Veränderung	Als Bruch	Größter Nenner	Nenner gleich
2	*=H2:H12-D2:D12*	*=WENN(L2:L12=0;"";TEXT(L2:L12;"# ???/???"))*	*=MAX(N3:N12)*	*=WENN(L2:L12<>0;RUNDEN(L2:L12*N2;0) & "/" & N2;"")*
3			=WERT("0" & RECHTS(M3:M12;LÄNGE(M3:M12)-WENNFEHLER(SUCHEN("/";M3:M12);0)))	

Sie können aber auch die benutzerdefinierte Funktion sbNRN verwenden.

Der Trick mit den 17 Kamelen

Ein Vater hinterließ seinen drei Söhnen 17 Kamele, sein letzter Wille besagte, dass der älteste Sohn die Hälfte der Kamele erhalten solle, der mittlere Sohn ein Drittel und der jüngste ein Neuntel. Dies war nicht leicht, aber ein weiser Mann half den Söhnen: er fügte sein Kamel hinzu, der älteste Sohn nahm 18/2 = 9 Kamele, der zweite 18/3 = 6, der jüngste 18/9 = 2 und der weise Mann nahm sein eigenes zurück und entschwand.

Nun können wir erkennen, was an dem letzten Willen des Vaters falsch war:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Anteil alt	Als Bruch	Anteil alt normiert	Als Bruch	Anteil neu	Als Bruch	Anteil neu normiert	Als Bruch	Größter Nenner	Nenner gleich	Veränderung	Als Bruch	Größter Nenner	Nenner gleich
2	*Gesamt*	*94,44%*	*17/18*	*100,00%*	*1/1*	*100,00%*	*1/1*	*100,00%*	*1/1*	*18*	*18/18*			*153*	
3	Ältester Sohn	50,00%	1/2	52,94%	9/17	50,00%	1/2	50,00%	1/2	2	9/18	-2,94%	-1/34	34	-5/153
4	Zweiter Sohn	33,33%	1/3	35,29%	6/17	33,33%	1/3	33,33%	1/3	3	6/18	-1,96%	-1/51	51	-3/153
5	Jüngster Sohn	11,11%	1/9	11,76%	2/17	11,11%	1/9	11,11%	1/9	9	2/18	-0,65%	-1/153	153	-1/153
6	Weiser Mann					5,56%	1/18	5,56%	1/18	18	1/18	5,56%	1/18	18	9/153

Die Anteile der Söhne ergeben zusammen keine 100%, sondern lediglich 94,44%. Damit erhalten sie am Ende 1/34, 1/51 und 1/153 zu wenig. Der Vater hätte dem jüngsten Sohn besser ein Sechstel der Kamele vermacht, dann hätten die Anteile zusammen 100% ausgemacht. Die Anteile hätten zwar keine ganzen Kamele ergeben, aber die Söhne hätten Ausgleichszahlungen vereinbaren können.

Monatsanteil

Wenn Sie die Anzahl ganzer Monate zwischen zwei Daten benötigen, können Sie die Funktion *DATEDIF* verwenden, aber was machen Sie, wenn Sie den korrekten Anteil von Monaten und zusätzlich Tagen brauchen?

Auf diese Weise kann man es korrekt machen:

```
=DATEDIF(A2,B2,"m")+IF(DAY(B2)>=DAY(A2),(DAY(B2)-DAY(A2))/(DAY(DATE(YEAR(B2),MONTH(B2)+1,0))),(DAY(DATE(YEAR(A2),MONTH(A2)+1,0))-DAY(A2))/(DAY(DATE(YEAR(A2),MONTH(A2)+1,0)))+DAY(B2)/(DAY(DATE(YEAR(B2),MONTH(B2)+1,0))))
```

	A	B	C	D	E
1	Start Date	End Date	Difference in months	...in days	Comment
2	01-Jan-2009	01-Feb-2009	1	31	One full calendar month
3	01-Jan-2009	16-Jan-2009	0.483870968	15	15 days in January 2009
4	16-Jan-2009	01-Feb-2009	0.519585253	16	15 days in January 2009 plus 1 day in February 2009
5	28-Feb-2000	28-Mar-2000	1	29	One full calendar month

Hinweis: Mit der Funktion *MONATSENDE* kann man die Formel abkürzen zu

```
=DATEDIF(A2;B2;"m")+WENN(TAG(B2)>=TAG(A2);(TAG(B2)-TAG(A2))/TAG(MONATSENDE(B2;0));(TAG(MONATSENDE(A2;0))-TAG(A2))/TAG(MONATSENDE(A2;0))+TAG(B2)/TAG(MONATSENDE(B2;0)))
```

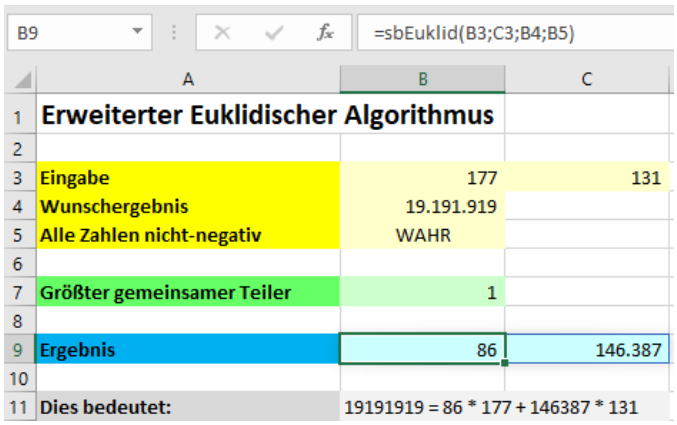
Falls Sie das Ergebnis als rationale Zahl darstellen müssen, nutzen Sie bitte die Funktion *sbNRN*.

Linearkombination Ganzer Zahlen

Erweiterter Euklidischer Algorithmus – *sbEuklid*

Sie wollen eine Zahl als nicht-negative Linearkombination zweier natürlicher Zahlen darstellen?

Dies kann man mit dem erweiterten Euklidischen Algorithmus erreichen:



	A	B	C
1	**Erweiterter Euklidischer Algorithmus**		
2			
3	**Eingabe**	177	131
4	**Wunschergebnis**	19.191.919	
5	**Alle Zahlen nicht-negativ**	WAHR	
6			
7	**Größter gemeinsamer Teiler**	1	
8			
9	**Ergebnis**	86	146.387
10			
11	**Dies bedeutet:**	19191919 = 86 * 177 + 146387 * 131	

Hinweis: Wenn das Wunschergebnis ein Vielfaches des größten gemeinsamen Teilers der eingegebenen Zahlen ist, dann existiert immer eine ganzzahlige Lösung, aber nicht notwendigerweise eine nicht-negative. Beispiel: Sie können mit den Eingaben 5 und 3 das Wunschergebnis 1 ganzzahlig darstellen, weil 1 der GGT von 5 und 3 ist: 1 = 2 * 5 + (-3) * 3. Aber es geht nicht mit ausschließlich nicht-negativen Faktoren.

Ein weiterer Hinweis: Es können mehrere nicht-negative Lösungen existieren, mit *bAllNonNegative* = WAHR gibt der u. g. Algorithmus die kleinste Summe der Ausgabewerte zurück. Wenn eine ganzzahlige Lösung existiert, dann gibt es abzählbar viele Lösungen.

sbEuklid Programmcode

Die möglichen Fehlercodes des Programms bedeuten:

- #NV! - Es existiert keine Lösung
- #WERT! - *bAllNonNegative* = WAHR aber nicht alle Eingaben sind nicht-negativ
- #ZAHL! - *bAllNonNegative* = WAHR aber es existiert keine nicht-negative Lösung

```
Function sbEuklid(lInput1 As Long, _
                  lInput2 As Long, _
                  Optional lDesiredResult As Long, _
                  Optional bAllNonNegative As Boolean = False) As Variant
'Extended Euklidean Algorithm which calculates two factors f1 and f2
'so that lDesiredResult = f1 * lInput1 + f2 * lInput2. If lDesiredResult
'is not given, the greatest common divisor (GCD) of lInput1 and lInput2
'will be calculated. If bAllNonNegative is True then we try to achieve a
'non-negative result of all inputs and outputs with minimal Sum(f1+f2).
'Error return values can be:
'xlErrNA    - There is no solution
'xlErrValue - bAllNonNegative = True but not all inputs are non-negative
'xlErrNum   - bAllNonNegative = True but there is no non-negative solution
'(C) (P) by Bernd Plumhoff 20-May-2024 PB V0.4
Dim lDiv                    As Long
Dim lGCD                    As Long
Dim lRest                   As Long
Dim lT1                     As Long
Dim lT2                     As Long
Dim vR                      As Variant
Dim vT                      As Variant

With Application '.WorksheetFunction 'Test with, release without
  lGCD = .Gcd(lInput1, lInput2)
  If IsMissing(lDesiredResult) Then lDesiredResult = lGCD
  lRest = lDesiredResult Mod lGCD
  If lRest <> 0 Then
    sbEuklid = CVErr(xlErrNA) 'There is no solution
  Else
    If bAllNonNegative And (lInput1 < 0 Or lInput2 < 0 Or lDesiredResult < 0) Then
      sbEuklid = CVErr(xlErrValue) 'bAllNonNegative but not all inputs are non-negative
    Else
      'See https://www.arndt-bruenner.de/mathe/scripts/erweitertereuklid.htm
      vR = [{1, 0; 0, 1}]
      vT = [{0, 1; 1, 0}]
      lT1 = lInput1
      lT2 = lInput2
      Do
        lDiv = lT1 \ lT2
        lRest = lT1 Mod lT2
        lT1 = lT2
        lT2 = lRest
        vT(2, 2) = -lDiv
        vR = .MMult(vR, vT)
      Loop While lRest <> 0
      vR = .MMult(Array(lDesiredResult \ lGCD, 0), .Transpose(vR))
      Debug.Assert lDesiredResult = vR(1) * lInput1 + vR(2) * lInput2 'Just assuring
      sbEuklid = vR
      If bAllNonNegative Then
        If lInput1 > lInput2 Then
          lT1 = lDesiredResult \ lInput1 + 1
          Do While lT1 > 0
            lT1 = lT1 - 1
            If (lDesiredResult - lInput1 * lT1) Mod lInput2 = 0 Then GoTo Success1
          Loop
          GoTo ErrorExit
Success1: vR(1) = lT1
          vR(2) = (lDesiredResult - lInput1 * lT1) \ lInput2
        Else
          lT2 = lDesiredResult \ lInput2 + 1
          Do While lT2 > 0
            lT2 = lT2 - 1
            If (lDesiredResult - lInput2 * lT2) Mod lInput1 = 0 Then GoTo Success2
          Loop
          GoTo ErrorExit
Success2: vR(2) = lT2
          vR(1) = (lDesiredResult - lInput2 * lT2) \ lInput1
        End If
        sbEuklid = vR
      End If
    End If
  End If
  'Debug.Assert lDesiredResult = vR(1) * lInput1 + vR(2) * lInput2 'Just testing
Exit Function
ErrorExit:
  sbEuklid = CVErr(xlErrNum)
End With

End Function
```

Uhrzeiten

Arbeitszeit zwischen 2 Zeitpunkten – *sbTimeDiff*

Name

sbTimeDiff() - Berechne die Zeit zwischen zwei Zeitpunkten, aber zähle lediglich die spezifizierten Zeiten pro Wochentag oder Feiertag minus Pausen, falls die tägliche Arbeitszeit definierte Grenzwerte überschreitet.

Synopsis

sbTimeDiff(*dtFrom*, *dtTo*, *vwh* [, *vHolidays*] [, *vBreaks*])

Beschreibung

Berechnet die Zeit zwischen zwei Zeitpunkten, aber zähle lediglich die spezifizierten Zeiten pro Wochentag oder Feiertag minus Pausen, falls die tägliche Arbeitszeit definierte Grenzwerte überschreitet.

Optionen

dtFrom - Datum und Uhrzeit ab wann zu zählen ist

dtTo - Datum und Uhrzeit bis wann zu zählen ist

vwh - 8 mal 2 Matrix, definiert die Startzeiten und Endzeiten pro Wochentag und für Feiertage, die erste Zeile für Montage, die achte für Feiertage

vHolidays - Optional. Liste der Feiertage (ganzzahlige Datumswerte). Für die Tage in dieser Liste werden nicht die Wochentagszeiten von vwh genommen, sondern die Feiertagszeiten in der achten Zeile

vBreaks - Optional. N x 2 Matrix, die die aggregierten täglichen Arbeitszeiten aufsteigend mit den zugehörigen Pausenzeiten darstellt, die zu subtrahieren sind, wenn die entsprechende Zeit an einem Tag gearbeitet wurde

Beispiel

C9: =sbTimeDiff(A9,B9,$B$12:$C$19,,$B$22:$C$23)

	A	B	C	D	E	F	G
1	sbTimeDiff Examples						UK Holidays
2	From	To	Result	Formula	Comment		Fri 19-Apr-2019
3	Tue 24-Dec-2019 16:00:00	Sun 29-Dec-2019 10:00:00	34:00:00	=sbTimeDiff(A3,B3,$B$12:$C$19)	No holidays		Mon 22-Apr-2019
4	Tue 24-Dec-2019 16:00:00	Sun 29-Dec-2019 10:00:00	18:00:00	=sbTimeDiff(A4,B4,$B$12:$C$19,$G$2:$G$128)	With holidays		Mon 06-May-2019
5	Sun 05-Apr-2020 12:59:00	Sun 05-Apr-2020 19:32:00	0:01:00	=sbTimeDiff(A5,B5,$B$12:$C$19)	No holidays		Mon 27-May-2019
6	Sun 05-Apr-2020 11:30:00	Sun 05-Apr-2020 16:30:00	1:30:00	=sbTimeDiff(A6,B6,$B$12:$C$19,$G$2:$G$128)	With holidays		Mon 26-Aug-2019
7	Sun 05-Apr-2020 06:29:00	Mon 06-Apr-2020 08:32:00	2:32:00	=sbTimeDiff(A7,B7,$B$12:$C$19)	No holidays		Wed 25-Dec-2019
8	Tue 24-Dec-2019 16:00:00	Sun 29-Dec-2019 10:00:00	18:00:00	=sbTimeDiff(A8,B8,$B$12:$C$19,$G$2:$G$128)	With holidays		Thu 26-Dec-2019
9	Tue 15-Sep-2020 05:30:00	Fri 18-Sep-2020 00:31:00	27:45:00	=sbTimeDiff(A9,B9,$B$12:$C$19,,$B$22:$C$23)	With breaks, no hols		Wed 01-Jan-2020
10							Fri 10-Apr-2020
11	Working Hours	Start	End				Mon 13-Apr-2020
12	Monday	8:00	18:00				Mon 04-May-2020
13	Tuesday	8:00	18:00				Mon 25-May-2020
14	Wednesday	8:00	18:00				Mon 31-Aug-2020
15	Thursday	8:00	18:00				Fri 25-Dec-2020
16	Friday	8:00	18:00				Mon 28-Dec-2020
17	Saturday	10:00	12:00				Fri 01-Jan-2021
18	Sunday	11:00	13:00				Fri 02-Apr-2021
19	Holidays	12:00	14:00				Mon 05-Apr-2021
20							Mon 03-May-2021
21	Breaks	Limit	Duration				Mon 31-May-2021
22	First	06:00	00:30				Mon 30-Aug-2021
23	Second	09:00	00:15				Mon 27-Dec-2021

sbTimeDiff Programmcode

```
Enum mc_Macro_Categories
    mcFinancial = 1
    mcDate_and_Time
    mcMath_and_Trig
    mcStatistical
    mcLookup_and_Reference
    mcDatabase
    mcText
    mcLogical
    mcInformation
    mcCommands
    mcCustomizing
    mcMacro_Control
    mcDDE_External
    mcUser_Defined
    mcFirst_custom_category
    mcSecond_custom_category 'and so on
End Enum 'mc_Macro_Categories

Function sbTimeDiff(dtFrom As Date, dtTo As Date, _
    vwh As Variant, _
    Optional vHolidays As Variant, _
    Optional vBreaks As Variant) As Date
'Returns time between dtFrom and dtTo but counts only
'dates and hours given in table vwh: for example
'09:00   17:00  'Monday
'09:00   17:00  'Tuesday
'09:00   17:00  'Wednesday
'09:00   17:00  'Thursday
'09:00   17:00  'Friday
'00:00   00:00  'Saturday
'00:00   00:00  'Sunday
'00:00   00:00  'Holidays
'This table defines hours to count for each day of the
'week (starting with Monday, 2 columns) and for holidays.
'Holidays given in vHolidays overrule week days.
'If you define a break table with break limits greater zero
'then the duration of each break exceeding the applicable
'time for this day will be subtracted from each day's time,
'but only down to the limit time, table needs to be sorted
'by limits in increasing order:
'Break table example
'Limit Duration (title row is not part of the table)
'6:00  0:30
'9:00  0:15
'
'(C) (P) by Bernd Plumhoff 28-Aug-2020 PB V1.3
Dim dt2 As Date, dt3 As Date, dt4 As Date, dt5 As Date
```

```
Dim i As Long, lTo As Long, lFrom As Long
Dim lWDFrom As Long, lWDTo As Long, lWDi As Long
Dim objHolidays As Object, objBreaks As Object, v As Variant

With Application.WorksheetFunction
sbTimeDiff = 0#
If dtTo <= dtFrom Then Exit Function
Set objHolidays = CreateObject("Scripting.Dictionary")
If Not IsMissing(vHolidays) Then
    For Each v In vHolidays
        objHolidays(v.Value) = 1
    Next v
End If
If Not IsMissing(vBreaks) Then
    vBreaks = .Transpose(.Transpose(vBreaks))
    Set objBreaks = CreateObject("Scripting.Dictionary")
    For i = LBound(vBreaks, 1) To UBound(vBreaks, 1)
        objBreaks(CDate(vBreaks(i, 1))) = CDate(vBreaks(i, 2))
    Next i
End If
lFrom = Int(dtFrom): lWDFrom = Weekday(lFrom, vbMonday)
lTo = Int(dtTo): lWDTo = Weekday(lTo, vbMonday)
If lFrom = lTo Then
    lWDi = lWDTo: If objHolidays(lTo) Then lWDi = 8
    dt3 = lTo + CDate(vwh(lWDi, 2))
    If dt3 > dtTo Then dt3 = dtTo
    dt2 = lTo + CDate(vwh(lWDi, 1))
    If dt2 < dtFrom Then dt2 = dtFrom
    If dt3 > dt2 Then
        dt2 = dt3 - dt2
    Else
        dt2 = 0#
    End If
    If Not IsMissing(vBreaks) Then
        dt2 = sbBreaks(dt2, objBreaks)
    End If
    sbTimeDiff = dt2
    Set objHolidays = Nothing
    Set objBreaks = Nothing
    Exit Function
End If
lWDi = lWDFrom: If objHolidays(lFrom) Then lWDi = 8
If dtFrom - lFrom >= CDate(vwh(lWDi, 2)) Then
    dt2 = 0#
Else
    dt2 = lFrom + CDate(vwh(lWDi, 1))
    If dt2 < dtFrom Then dt2 = dtFrom
    dt2 = lFrom + CDate(vwh(lWDi, 2)) - dt2
    If Not IsMissing(vBreaks) Then
        dt2 = sbBreaks(dt2, objBreaks)
    End If
End If
lWDi = lWDTo: If objHolidays(lTo) Then lWDi = 8
If dtTo - lTo <= CDate(vwh(lWDi, 1)) Then
    dt4 = 0#
Else
    dt4 = lTo + CDate(vwh(lWDi, 2))
    If dt4 > dtTo Then dt4 = dtTo
    dt4 = dt4 - lTo - CDate(vwh(lWDi, 1))
    If Not IsMissing(vBreaks) Then
        dt4 = sbBreaks(dt4, objBreaks)
    End If
End If
dt3 = 0#
For i = lFrom + 1 To lTo - 1
    lWDi = Weekday(i, vbMonday)
    If objHolidays(i) Then lWDi = 8
    dt5 = CDate(vwh(lWDi, 2)) - CDate(vwh(lWDi, 1))
    If Not IsMissing(vBreaks) Then
        dt5 = sbBreaks(dt5, objBreaks)
    End If
    dt3 = dt3 + dt5
Next i
Set objHolidays = Nothing
Set objBreaks = Nothing
sbTimeDiff = dt2 + dt3 + dt4
End With
End Function

Private Function sbBreaks(ByVal dt As Date, objBreaks As Object) As Date
'Subtract break durations from dt as long as it exceeds the break limit,
'but not below break limit.
'(C) (P) by Bernd Plumhoff 22-Mar-2020 PB V1.00
Dim dtTemp As Date
Dim k As Long
k = 0
Do While k <= UBound(objBreaks.keys)
    If dt > objBreaks.keys()(k) + objBreaks.items()(k) - dtTemp Then
        dt = dt - objBreaks.items()(k)
        dtTemp = dtTemp + objBreaks.items()(k)
```

```
    ElseIf dt > objBreaks.keys()(k) - dtTemp Then
        dt = objBreaks.keys()(k) - dtTemp
        Exit Do
    End If
    k = k + 1
Loop
sbBreaks = dt
End Function

Sub DescribeFunction_sbTimeDiff()

'Run this only once, then you will see this description in the function menu

Dim FuncName As String
Dim FuncDesc As String
Dim Category As String
Dim ArgDesc(1 To 5) As String

FuncName = "sbTimeDiff"
FuncDesc = "Returns time between dtFrom and dtTo but counts only " & _
            "time given in table vwh. Holidays given in vHolidays " & _
            "overrule week days, all breaks given in vBreaks are " & _
            "subtracted if corresponding time has been worked"
Category = mcDate_and_Time
ArgDesc(1) = "Start date and time where to count from"
ArgDesc(2) = "End date and time to count to"
ArgDesc(3) = "Range or array which defines which time to count during the week starting from Monday, " & _
            "8 by 2 matrix defining start time and end time for each weekday (8th row for holidays)"
ArgDesc(4) = "Optional list of holidays which overrule week days, define time to count in 8th row of vwh"
ArgDesc(5) = "Optional. N x 2 matrix specifying working limit times (sorted in ascending order) and break"
& _
             " durations to subtract if corresponding time for a day has been worked (but not below limit
time)"

Application.MacroOptions _
    Macro:=FuncName, _
    Description:=FuncDesc, _
    Category:=Category, _
    ArgumentDescriptions:=ArgDesc

End Sub
```

Arbeitszeit zu einem Zeitpunkt addieren – *sbTimeAdd*

Name

sbTimeAdd() - Addiere eine positive Zeit zu einem Datum mit Uhrzeit, wobei lediglich spezifizierte Zeitintervalle pro Wochentag und Feiertag berücksichtigt werden und auch eine definierte tägliche Pausenzeit abgezogen wird, falls die entsprechende tägliche definierte Arbeitszeit überschritten wird.

Synopsis

sbTimeAdd(*dt*, *dh*, *vwh* [, *vHolidays*] [, *dtBreakLimit*] [, *dtBreakDuration*])

Beschreibung

Addiere eine positive Zeit zu einem Datum mit Uhrzeit, wobei lediglich spezifizierte Zeitintervalle pro Wochentag und Feiertag berücksichtigt werden und auch eine definierte tägliche Pausenzeit abgezogen wird, falls die entsprechende tägliche definierte Arbeitszeit überschritten wird.

Optionen

dt - Datum und Uhrzeit auf die die Zeit dh addiert werden soll

dh - Zeit als Datentyp Double die auf *dt* addiert werden soll

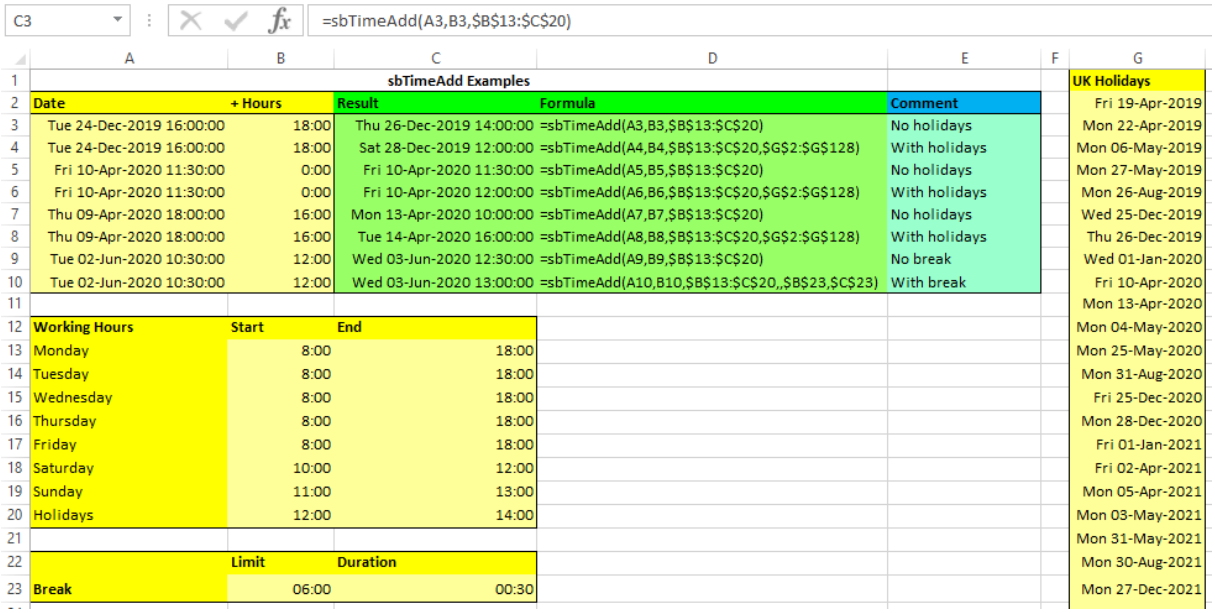
vwh - 8 mal 2 Matrix, definiert die Startzeiten und Endzeiten pro Wochentag und für Feiertage, die erste Zeile für Montage, die achte für Feiertage

vHolidays - Optional. Liste der Feiertage (ganzzahlige Datumswerte). Für die Tage in dieser Liste werden nicht die Wochentagszeiten von *vwh* genommen, sondern die Feiertagszeiten in der achten Zeile

dtBreakLimit - Optional. Tägliche Arbeitszeit ab der die Pausenzeit *dtBreakDuration* abgezogen werden muss

dtBreakDuration - Optional. Pausenzeit die von der aggregierten täglichen Arbeitszeit *dtBreakLimit* abgezogen muss, falls diese erreicht wird

Beispiel



C3 =sbTimeAdd(A3,B3,$B$13:$C$20)

	A	B	C	D	E	F	G
1	sbTimeAdd Examples						UK Holidays
2	Date	+ Hours	Result	Formula	Comment		Fri 19-Apr-2019
3	Tue 24-Dec-2019 16:00:00	18:00	Thu 26-Dec-2019 14:00:00	=sbTimeAdd(A3,B3,$B$13:$C$20)	No holidays		Mon 22-Apr-2019
4	Tue 24-Dec-2019 16:00:00	18:00	Sat 28-Dec-2019 12:00:00	=sbTimeAdd(A4,B4,$B$13:$C$20,$G$2:$G$128)	With holidays		Mon 06-May-2019
5	Fri 10-Apr-2020 11:30:00	0:00	Fri 10-Apr-2020 11:30:00	=sbTimeAdd(A5,B5,$B$13:$C$20)	No holidays		Mon 27-May-2019
6	Fri 10-Apr-2020 11:30:00	0:00	Fri 10-Apr-2020 12:00:00	=sbTimeAdd(A6,B6,$B$13:$C$20,$G$2:$G$128)	With holidays		Mon 26-Aug-2019
7	Thu 09-Apr-2020 18:00:00	16:00	Mon 13-Apr-2020 10:00:00	=sbTimeAdd(A7,B7,$B$13:$C$20)	No holidays		Wed 25-Dec-2019
8	Thu 09-Apr-2020 18:00:00	16:00	Tue 14-Apr-2020 16:00:00	=sbTimeAdd(A8,B8,$B$13:$C$20,$G$2:$G$128)	With holidays		Thu 26-Dec-2019
9	Tue 02-Jun-2020 10:30:00	12:00	Wed 03-Jun-2020 12:30:00	=sbTimeAdd(A9,B9,$B$13:$C$20)	No break		Wed 01-Jan-2020
10	Tue 02-Jun-2020 10:30:00	12:00	Wed 03-Jun-2020 13:00:00	=sbTimeAdd(A10,B10,$B$13:$C$20,,$B$23,$C$23)	With break		Fri 10-Apr-2020
11							Mon 13-Apr-2020
12	Working Hours	Start	End				Mon 04-May-2020
13	Monday	8:00	18:00				Mon 25-May-2020
14	Tuesday	8:00	18:00				Mon 31-Aug-2020
15	Wednesday	8:00	18:00				Fri 25-Dec-2020
16	Thursday	8:00	18:00				Mon 28-Dec-2020
17	Friday	8:00	18:00				Fri 01-Jan-2021
18	Saturday	10:00	12:00				Fri 02-Apr-2021
19	Sunday	11:00	13:00				Mon 05-Apr-2021
20	Holidays	12:00	14:00				Mon 03-May-2021
21							Mon 31-May-2021
22		Limit	Duration				Mon 30-Aug-2021
23	Break	06:00	00:30				Mon 27-Dec-2021

sbTimeAdd Programmcode

```
Enum mc_Macro_Categories
    mcFinancial = 1
    mcDate_and_Time
    mcMath_and_Trig
    mcStatistical
    mcLookup_and_Reference
    mcDatabase
    mcText
    mcLogical
    mcInformation
    mcCommands
    mcCustomizing
    mcMacro_Control
    mcDDE_External
    mcUser_Defined
    mcFirst_custom_category
    mcSecond_custom_category 'and so on
End Enum 'mc_Macro_Categories

Function sbTimeAdd(dt As Date, dh As Double, _
    vwh As Variant, _
    Optional vHolidays As Variant, _
    Optional dtBreakLimit As Date, _
```

```
    Optional dtBreakDuration As Date) As Date
'Returns end date from start date dt and positive duration
'dh in hours (and minutes and seconds) but counts only
'time as given in table vwh: for example
'09:00   17:00  'Monday
'09:00   17:00  'Tuesday
'09:00   17:00  'Wednesday
'09:00   17:00  'Thursday
'09:00   17:00  'Friday
'00:00   00:00  'Saturday
'00:00   00:00  'Sunday
'00:00   00:00  'Holidays
'This table defines hours to count for each day of the
'week (starting with Monday, 2 columns) and for holidays.
'You can also define a break limit and a break duration.
'If the working hour for a day is exceeding the limit
'then the duration will be subtracted from its time.
'(C) (P) by Bernd Plumhoff 02-Feb-2019 PB V0.7
Dim dt1 As Date, dt2 As Date
Dim ldt1 As Long, lWDi As Long, v As Variant
Dim objHolidays As Object, objBreaks As Object

If dh < 0# Then
    sbTimeAdd = CVErr(xlErrValue)
    Exit Function
End If
If Not IsMissing(vHolidays) Then
    Set objHolidays = CreateObject("Scripting.Dictionary")
    For Each v In vHolidays
        objHolidays(Int(v.Value)) = 1
    Next v
End If
ldt1 = Int(dt)
lWDi = Weekday(ldt1, vbMonday)
If Not IsMissing(vHolidays) Then
    If objHolidays(ldt1) Then
        lWDi = 8
    End If
End If
dt1 = ldt1 + CDate(vwh(lWDi, 1)) 'start time of this day
If dt1 < dt Then dt1 = dt
dt2 = ldt1 + CDate(vwh(lWDi, 2)) 'end time of this day
If dt2 < dt1 Then dt2 = dt1
Do While Round2Sec(dt1 + dh - (dh >= dtBreakLimit) * _
    dtBreakDuration) > Round2Sec(dt2)
    'go ahead as long as our duration exceeds this day
    If dt1 < ldt1 + CDate(vwh(lWDi, 2)) Then
        dh = dh - dt2 + dt1 - (dh >= dtBreakLimit) * dtBreakDuration
    End If
    ldt1 = ldt1 + 1
    lWDi = Weekday(ldt1, vbMonday)
    If Not IsMissing(vHolidays) Then
        If objHolidays(ldt1) Then
            lWDi = 8
        End If
    End If
    dt1 = ldt1 + CDate(vwh(lWDi, 1)) 'start time of this day
    dt2 = ldt1 + CDate(vwh(lWDi, 2)) 'end time of this day
Loop
sbTimeAdd = dt1 + dh - (dh >= dtBreakLimit) * dtBreakDuration
End Function

Function Round2Sec(dt As Date) As Date
Round2Sec = Int(0.5 + dt * 24 * 60 * 60) / 24 / 60 / 60
End Function

Sub DescribeFunction_sbTimeAdd()

'Run this only once, then you will see this description in the function menu

Dim FuncName As String
Dim FuncDesc As String
Dim Category As String
Dim ArgDesc(1 To 6) As String

FuncName = "sbTimeAdd"
FuncDesc = "Add positive hours to a timepoint but count only time as specified for week days" & _
           " and for holidays increased by break time if working time exceeds specified time"
Category = mcDate_and_Time
ArgDesc(1) = "Start date and time where to count from"
ArgDesc(2) = "Hours to add"
ArgDesc(3) = "Range or array which defines which time to count during the week starting from Monday, " & _
             "8 by 2 matrix defining start time and end time for each weekday (8th row for holidays)"
ArgDesc(4) = "Optional list of holidays which overrule week days, define time to count in 8th row of vwh"
ArgDesc(5) = "Optional. Daily working time limit. If exceeded dtBreakDUration will be subtracted from total
time"
ArgDesc(6) = "Optional. Break time. Will be subtracted from total time if daily working time exceeds
dtBreakLimit"

Application.MacroOptions _
```

```
    Macro:=FuncName, _
    Description:=FuncDesc, _
    Category:=Category, _
    ArgumentDescriptions:=ArgDesc
```

Uhrzeit für eine andere Zeitzone umwandeln – *ConvertTime*

Julian Hess und Patrick Honorez entwickelten ein sehr gutes Konvertierprogramm für MS Access und für MS Excel, um die Zeit einer Zeitzone in eine andere zu übersetzen:

https://stackoverflow.com/questions/3120915/get-timezone-information-in-vba-excel/20489651#20489651

Bitte beachten Sie, dass hierfür ein ordentlich installiertes MS Outlook Voraussetzung ist.

Prüfziffern

Prüfziffern sollen die manuelle Fehleingabe oder die fehlerhafte Übertragung von numerischen Daten erschweren.

Berechne oder prüfe eine Europäische Artikelnummer – *sbEAN*

Falls Sie eine Europäische Artikel Nummer (EAN) berechnen oder prüfen müssen:

	A	B	C
1	**Eingabe**	**Ausgabe**	**Kommentar**
2	1234567	0	Nur die EAN-8 Prüfziffer
3	1234567	12345670	Ganze EAN-8
4	12345670	WAHR	EAN-8 ist korrekt
5	12345678	FALSCH	EAN-8 ist falsch
6	123456789012	8	Nur die EAN-13 Prüfziffer
7	123456789012	1234567890128	Ganze EAN-13
8	1234567890128	WAHR	EAN-13 ist korrekt
9	1234567890129	FALSCH	EAN-13 ist falsch
10	12345678901234567	5	Nur die EAN-18 / NVE / SSCC Prüfziffer
11	12345678901234567	123456789012345675	Ganze EAN-18 / NVE / SSCC
12	123456789012345675	WAHR	EAN-18 ist korrekt
13	123456789012345676	FALSCH	EAN-18 ist falsch
14	9023671254823	0	Nur die EAN-14 / GTIN Prüfziffer
15	9023671254823	90236712548230	Ganze EAN-14 / GTIN
16	90236712548230	WAHR	EAN-14 / GTIN ist korrekt
17	90236712548231	FALSCH	EAN-14 / GTIN ist falsch
18	1234567890123456789	#WERT!	Falsche Eingabelänge
19	123456789O12	#ZAHL!	Ist keine Zahl (Zeichen 'O')

sbEAN Programmcode

```
Option Explicit

Function sbEAN(s As String, _
  Optional bFullEAN As Boolean = True, _
  Optional bEAN14 As Boolean = False) As Variant
'Calculate or check EAN check digit. Works for EAN-8,
'EAN-13, EAN-14 / GTIN, and for EAN-18 / NVE / SSCC.
'If EAN is given without check digit, it is calculated
'and returned (full EAN if bFullEAn is True or just the
'check digit if False). If full EAN is entered the
'result of the check (True or False) will be returned.
'(C) (P) by Bernd Plumhoff 31-Mar-2024 PB V0.3
Dim i As Long, d As Long, m As Long, w As Long
Dim bCheck As Boolean
m = Len(s)
For i = 1 To m
  w = Asc(Mid(s, i, 1))
  If w < 48 Or w > 57 Then
    sbEAN = CVErr(xlErrNum)
    Exit Function
  End If
Next i
If bEAN14 Then
  If m = 13 Then
    bCheck = False
  ElseIf m = 14 Then
    bCheck = True
    m = m - 1 'Calculate checksum without check digit
  Else
    sbEAN = CVErr(xlErrValue)
    Exit Function
  End If
Else
  Select Case m
  Case 7, 12, 17
    bCheck = False
  Case 8, 13, 18
    bCheck = True
    m = m - 1 'Calculate checksum without check digit
  Case Else
    sbEAN = CVErr(xlErrValue)
    Exit Function
  End Select
End If
w = 3
For i = m To 1 Step -1
    d = d + Mid(s, i, 1) * w
    w = 4 - w 'Alternate between 3 and 1
Next i
d = (10 - d Mod 10) Mod 10
If bCheck Then
  sbEAN = Right(s, 1) = d
ElseIf bFullEAN Then
  sbEAN = s & d
Else
  sbEAN = d
End If
End Function
```

Einfache Mathematik in Formeln

Wie man Tabellenblattformeln analysiert

Mit der Funktionstaste F9 können Sie Tabellenblattformeln von innen nach außen oder von links nach rechts auswerten:

Beispiel (aus dem nachfolgenden Kapitel

Datumsformeln testen):

Eingabe in Zelle A2: 07.01.1901

Tabellenblattformel: `=QUOTIENT(TAG(MONATSENDE(A2;0))+13-WOCHENTAG(MONATSENDE(A2;0);2);7)`

Wir selektieren das erste Auftreten von `MONATSENDE(A2;0)` und drücken F9.

Wir erhalten `=QUOTIENT(TAG(397)+13-WOCHENTAG(MONATSENDE(A2;0);2);7).`

Nun selektieren wir den Teil `TAG(397)` und drücken wieder F9.

Es ergibt sich `=QUOTIENT(31+13-WOCHENTAG(MONATSENDE(A2;0);2);7).`

Wir selektieren den Teil `MONATSENDE(A2;0)` und drücken F9.

Wir erhalten `=QUOTIENT(31+13-WOCHENTAG(397;2);7).`

Wir selektieren den Teil `WOCHENTAG(397;2)` und drücken F9.

Es ergibt sich `=QUOTIENT(31+13-4;7).`

Nun selektieren wir den Teil `31+13-4` und drücken F9.

Heraus kommt `=QUOTIENT(40;7)`. Und danach das Ergebnis `5`, da die Funktion QUOTIENT ein ganzzahliges Ergebnis ohne Rest berechnet.

Viele Aufgaben in Excel lassen sich mit trivialer Mathematik lösen. Es folgen einige Beispiele.

Datumsformeln testen

Wie stelle ich sicher, dass eine Excel Datumsformel korrekt ist? Ich teste sie!

So geht es am besten:

- Ich generiere alle Tage des relevanten Datumsbereiches in Spalte A. Zum Beispiel 1-Jan-1901 bis 31-Dez-2099.
- Ich gebe alle in Frage kommenden Formeln nebeneinander in die Spalten B und folgende ein und kopiere sie nach unten.
- Ich vergleiche alle Ergebnisse mit der Referenzformel in Spalte B.

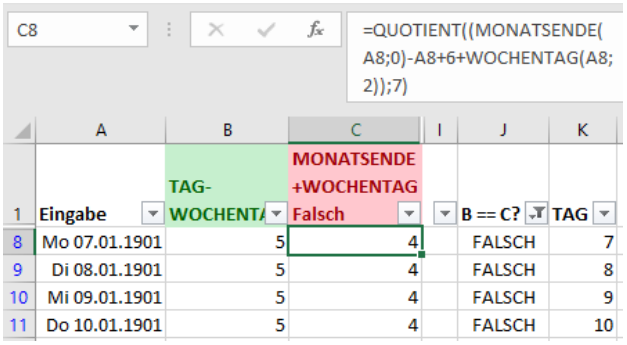
<u>Beispiel</u>

Wieviele ISO-Kalenderwochen berührt ein gegebener Monat?

Die Formeln:

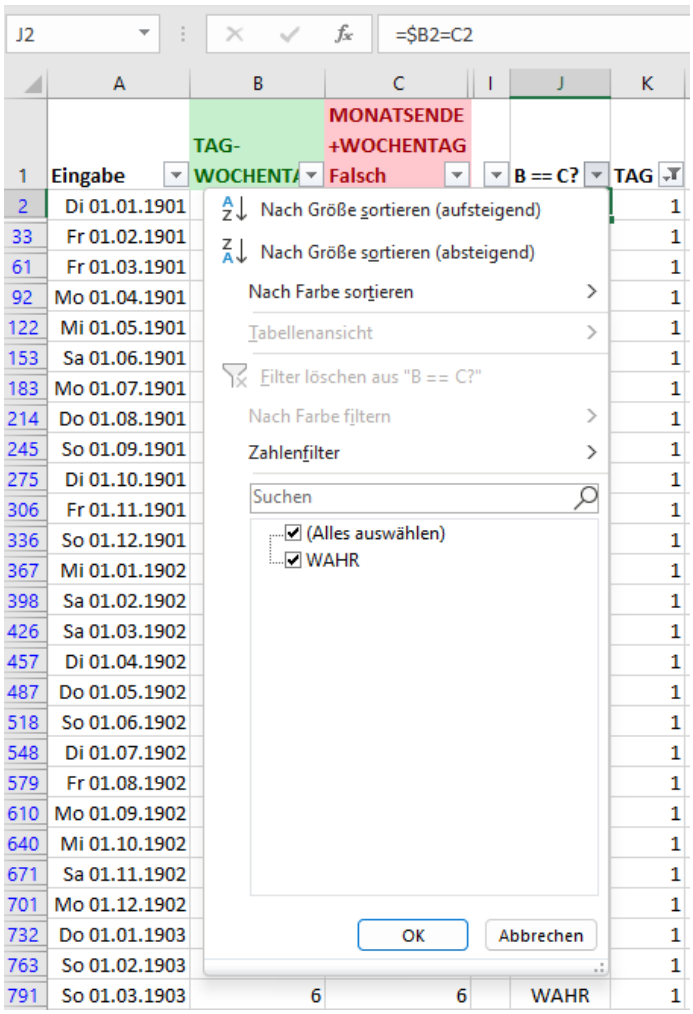
	B	C	J	K
1	**TAG-WOCHENTAG**	**MONATSENDE+WOCHENTAG Falsch**	**B == C?**	**TAG**
2	=QUOTIENT(TAG(MONATSENDE(A2;0))+13-WOCHENTAG(MONATSENDE(A2;0);2);7)	=QUOTIENT((MONATSENDE(A2;0)-A2+6+WOCHENTAG(A2;2));7)	=$B2=C2	=TAG(A2)

Kopieren Sie Zeile 2 nach unten. Wenn Sie einen Filter in Zeile 1 setzen und Spalte J auf den Wert FALSCH filtern, können Sie schnell erkennen, für welche Datumswerte Spalte C nicht mit der Referenzformel in Spalte B übereinstimmt:



	A	B	C	I	J	K
1	Eingabe	TAG-WOCHENTA	MONATSENDE+WOCHENTAG Falsch		B == C?	TAG
8	Mo 07.01.1901	5	4		FALSCH	7
9	Di 08.01.1901	5	4		FALSCH	8
10	Mi 09.01.1901	5	4		FALSCH	9
11	Do 10.01.1901	5	4		FALSCH	10

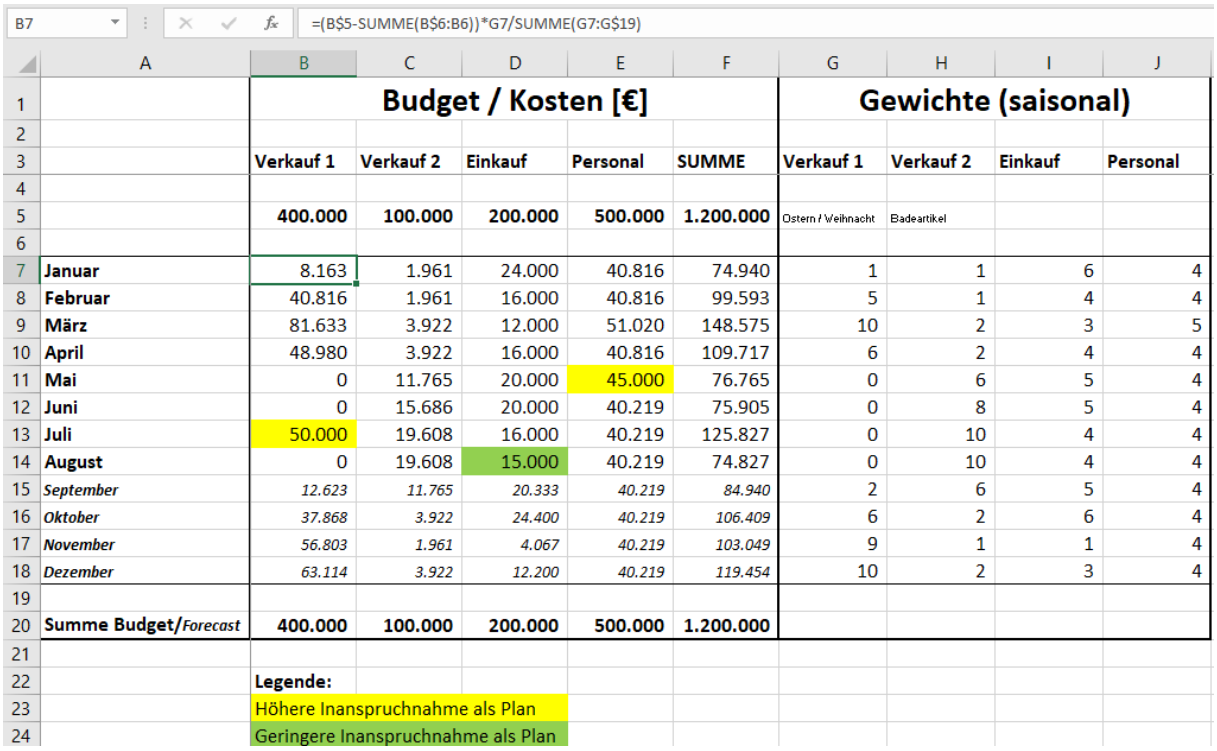
Wenn Sie Spalte K nach Tag 1 (Monatserster) filtern, dann sehen Sie dass die Formel in Spalte C am Monatsanfang (aber leider auch nur dann) korrekt ist:


J2
=$B2=C2
Eingabe
TAG-WOCHENTA
MONATSENDE +WOCHENTAG Falsch
B == C?
TAG
Nach Größe sortieren (aufsteigend)
Nach Größe sortieren (absteigend)
Nach Farbe sortieren
Tabellenansicht
Filter löschen aus "B == C?"
Nach Farbe filtern
Zahlenfilter
Suchen
(Alles auswählen)
WAHR
OK
Abbrechen

Budgetkontrolle

“If you want creativity, take a zero off your budget. If you want sustainability, take off two zeros.” [Jaime Lerner]

Dies ist eine simple Budgetplanung und -kontrolle.



B7 | =(B$5-SUMME(B$6:B6))*G7/SUMME(G7:G$19)

	A	B	C	D	E	F	G	H	I	J
1		**Budget / Kosten [€]**					**Gewichte (saisonal)**			
2										
3		**Verkauf 1**	**Verkauf 2**	**Einkauf**	**Personal**	**SUMME**	**Verkauf 1**	**Verkauf 2**	**Einkauf**	**Personal**
4										
5		**400.000**	**100.000**	**200.000**	**500.000**	**1.200.000**	Ostern / Weihnacht	Badeartikel		
6										
7	**Januar**	8.163	1.961	24.000	40.816	74.940	1	1	6	4
8	**Februar**	40.816	1.961	16.000	40.816	99.593	5	1	4	4
9	**März**	81.633	3.922	12.000	51.020	148.575	10	2	3	5
10	**April**	48.980	3.922	16.000	40.816	109.717	6	2	4	4
11	**Mai**	0	11.765	20.000	45.000	76.765	0	6	5	4
12	**Juni**	0	15.686	20.000	40.219	75.905	0	8	5	4
13	**Juli**	50.000	19.608	16.000	40.219	125.827	0	10	4	4
14	**August**	0	19.608	15.000	40.219	74.827	0	10	4	4
15	*September*	*12.623*	*11.765*	*20.333*	*40.219*	*84.940*	2	6	5	4
16	*Oktober*	*37.868*	*3.922*	*24.400*	*40.219*	*106.409*	6	2	6	4
17	*November*	*56.803*	*1.961*	*4.067*	*40.219*	*103.049*	9	1	1	4
18	*Dezember*	*63.114*	*3.922*	*12.200*	*40.219*	*119.454*	10	2	3	4
19										
20	**Summe Budget/*Forecast***	**400.000**	**100.000**	**200.000**	**500.000**	**1.200.000**				
21										
22		**Legende:**								
23		Höhere Inanspruchnahme als Plan								
24		Geringere Inanspruchnahme als Plan								

In die Zellen B5:E5 geben Sie die jährlichen Planbudgets Ihrer Abteilungen oder Kostenstellen ein. In die Zellen G7:J18 gehören die saisonalen Gewichte der Abteilungen. Als Beispiele sind Festartikel (Ostern und Weihnachten) sowie Badeartikel gezeigt. Der Einkauf hat 2 Monate vorher entsprechenden Aufwand. Die allgemeine Formel `=(B$5-SUMME(B$6:B6))*G7/SUMME(G7:G$19)` in Zelle B7 kopiert man anfänglich in alle Zellen des Bereiches B7:F18. Bitte beachten Sie, dass die Zeilen 6 und 19 absichtlich leer blieben.

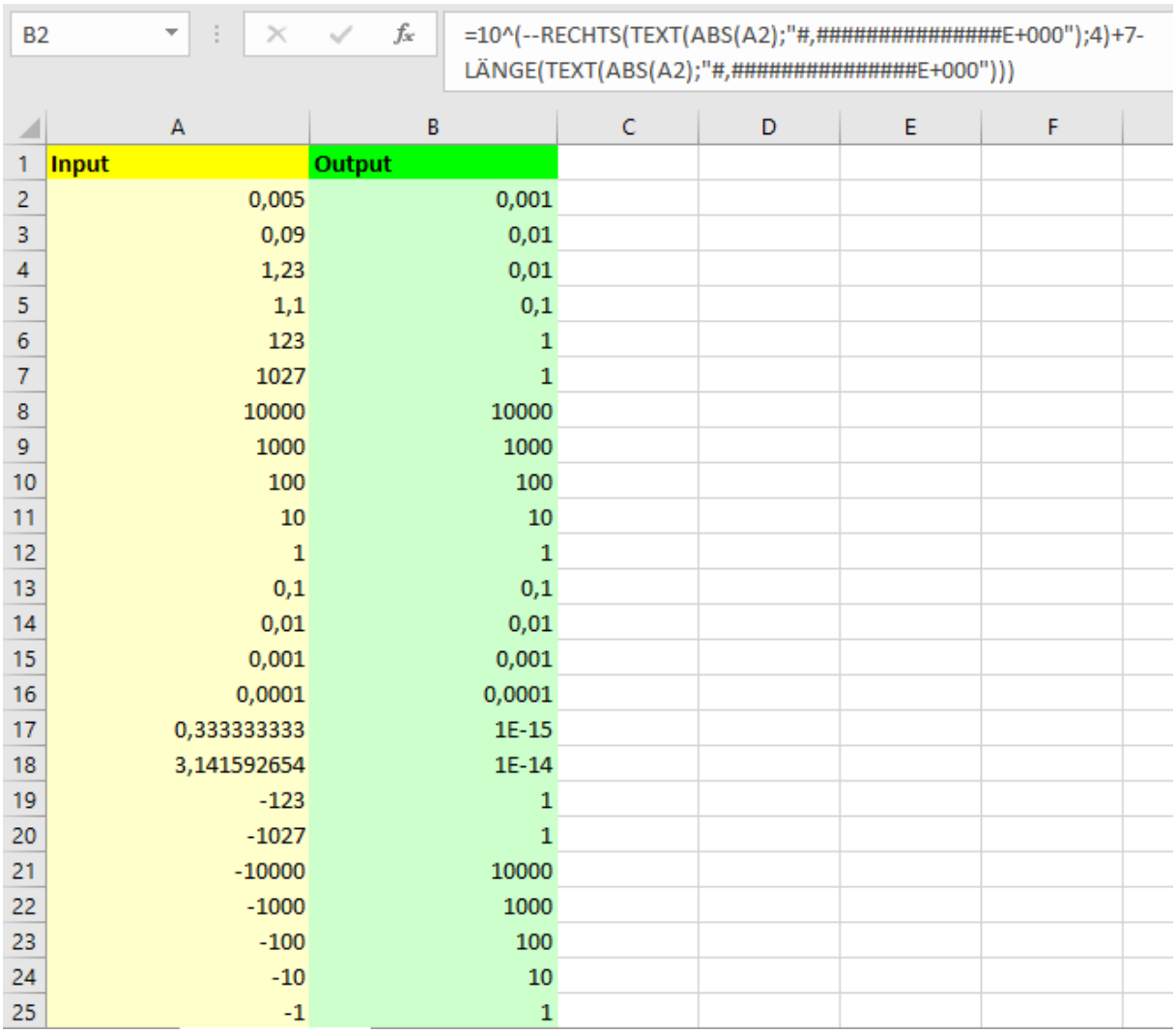
Im Laufe des Jahres überschreibt man die Plankosten mit den festgestellten Istkosten. Das Restbudget des Jahres passt sich dann automatisch an, um die Summe der Plankosten für jede Abteilung einzuhalten.

Sobald diese Restwerte für eine Abteilung unrealistisch werden, sollte man sie mit realistischen Prognosewerten überschreiben und z. B. unten eine Abweichungsinfomation ausgeben.

Geringste Signifikante Ziffer Erhöhen

Wie können Sie die geringste signifikante Ziffer einer Zahl erhöhen?

Beispiel: Gegeben sei die Zahl 123,4567. Sie benötigen eine Funktion, um 0,0001 zu ermitteln und auf die gegebene Zahl zu addieren.



	A	B	C	D	E	F
1	**Input**	**Output**				
2	0,005	0,001				
3	0,09	0,01				
4	1,23	0,01				
5	1,1	0,1				
6	123	1				
7	1027	1				
8	10000	10000				
9	1000	1000				
10	100	100				
11	10	10				
12	1	1				
13	0,1	0,1				
14	0,01	0,01				
15	0,001	0,001				
16	0,0001	0,0001				
17	0,333333333	1E-15				
18	3,141592654	1E-14				
19	-123	1				
20	-1027	1				
21	-10000	10000				
22	-1000	1000				
23	-100	100				
24	-10	10				
25	-1	1				

Lösung:

```
=10^(--RECHTS(TEXT(ABS(A1);"#,################E+000");4)+7-
LÄNGE(TEXT(ABS(A1);"#,################E+000")))
```

Linearer Breakdown

“I’m not such a big fan of having a linear answer to things.” [Adam Driver]

Wie kann man jährliche Daten

	A	B
1	**Year**	**Annual Data**
2	2019	29.063.000,00
3	2020	31.388.000,00
4	2021	24.826.000,00
5	2022	27.436.000,00

in monatliche Daten linear stückweise umwandeln

	A	B	C	D
7	**Month Ends**	**Monthly Data**	**Rounded preserving the sum**	**StDev(Data)**
8	31. Dez 2018	1.971.555,79		296.897,83
9	31. Jan 2019	2.040.842,08	2.040.842,08	
10	28. Feb 2019	2.110.128,37	2.110.128,37	
11	31. Mrz 2019	2.179.414,66	2.179.414,66	
12	30. Apr 2019	2.248.700,95	2.248.700,95	
13	31. Mai 2019	2.317.987,23	2.317.987,23	
14	30. Jun 2019	2.387.273,52	2.387.273,53	
15	31. Jul 2019	2.456.559,81	2.456.559,81	
16	31. Aug 2019	2.525.846,10	2.525.846,10	
17	30. Sep 2019	2.595.132,39	2.595.132,38	
18	31. Okt 2019	2.664.418,67	2.664.418,68	
19	30. Nov 2019	2.733.704,96	2.733.704,96	
20	31. Dez 2019	2.802.991,25	2.802.991,25	
21	31. Jan 2020	2.774.172,08	2.774.172,08	
22	29. Feb 2020	2.745.352,92	2.745.352,92	
23	31. Mrz 2020	2.716.533,75	2.716.533,75	
24	30. Apr 2020	2.687.714,58	2.687.714,58	
25	31. Mai 2020	2.658.895,42	2.658.895,42	
26	30. Jun 2020	2.630.076,25	2.630.076,25	
27	31. Jul 2020	2.601.257,08	2.601.257,08	
28	31. Aug 2020	2.572.437,92	2.572.437,92	
29	30. Sep 2020	2.543.618,75	2.543.618,75	
30	31. Okt 2020	2.514.799,58	2.514.799,58	
31	30. Nov 2020	2.485.980,42	2.485.980,42	
32	31. Dez 2020	2.457.161,25	2.457.161,25	
33	31. Jan 2021	2.397.418,49	2.397.418,49	
34	28. Feb 2021	2.337.675,74	2.337.675,74	
35	31. Mrz 2021	2.277.932,98	2.277.932,98	
36	30. Apr 2021	2.218.190,22	2.218.190,22	
37	31. Mai 2021	2.158.447,47	2.158.447,47	
38	30. Jun 2021	2.098.704,71	2.098.704,71	
39	31. Jul 2021	2.038.961,96	2.038.961,96	
40	31. Aug 2021	1.979.219,20	1.979.219,20	
41	30. Sep 2021	1.919.476,44	1.919.476,44	
42	31. Okt 2021	1.859.733,69	1.859.733,69	
43	30. Nov 2021	1.799.990,93	1.799.990,93	
44	31. Dez 2021	1.740.248,17	1.740.248,17	
45	31. Jan 2022	1.824.261,27	1.824.261,27	
46	28. Feb 2022	1.908.274,38	1.908.274,38	
47	31. Mrz 2022	1.992.287,48	1.992.287,48	
48	30. Apr 2022	2.076.300,58	2.076.300,58	
49	31. Mai 2022	2.160.313,68	2.160.313,68	
50	30. Jun 2022	2.244.326,78	2.244.326,78	
51	31. Jul 2022	2.328.339,88	2.328.339,89	
52	31. Aug 2022	2.412.352,99	2.412.352,98	
53	30. Sep 2022	2.496.366,09	2.496.366,09	
54	31. Okt 2022	2.580.379,19	2.580.379,19	
55	30. Nov 2022	2.664.392,29	2.664.392,29	
56	31. Dez 2022	2.748.405,39	2.748.405,39	

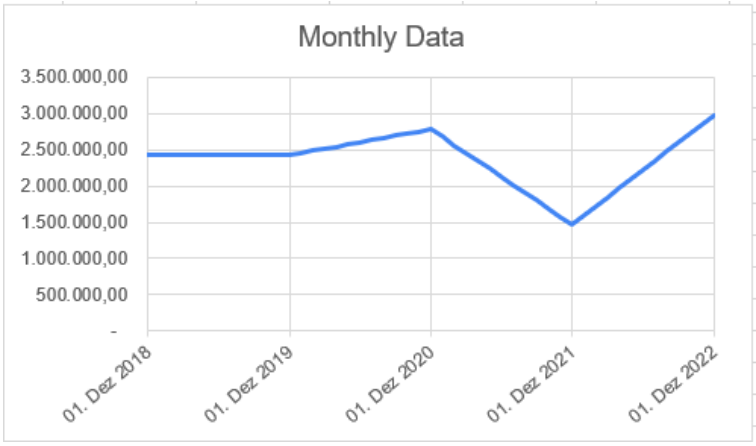
?

Geben Sie in A2:B5 Ihre jährlichen Daten ein. In A8:A56 geben Sie Ihre Monatsenddaten chronologisch sortiert ein. Für den Anfang geben Sie in B8 ein: `=B2/12`. In B9 geben Sie

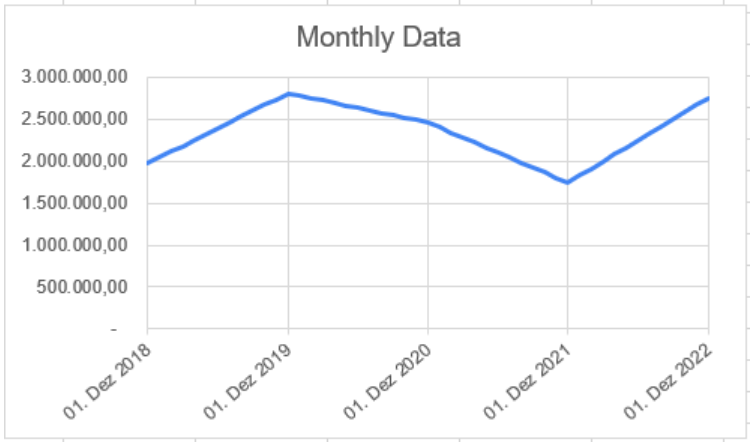
```
=SVERWEIS(DATUM(JAHR($A9)-
1;12;31);A$8:B8;2)+(MONAT($A9))*(SVERWEIS(JAHR($A9);$A$2:$B$5;2)-
```

SVERWEIS(DATUM(JAHR($A9)-1;12;31);A$8:B8;2)*12)/78 ein und kopieren nach unten. In D8 geben Sie =STABW.N(B9:B56) ein. Falls Ihre monatlichen Daten cent-genau die Summe der jährlichen Daten ausmachen müssen, geben Sie in C9 ein =RUNDEN(B9;2) und in D10: =RUNDEN(SUMME(B$9:B10);2)-SUMME(C$9:C9) und kopieren wieder nach unten. Dies ist lediglich eine pragmatische Rundungsmethode ohne VBA. Falls Sie diese Rundung korrekt durchführen wollen, verwenden Sie bitte RoundToSum.

Nun erhalten Sie die stückweise lineare Ausgabe:



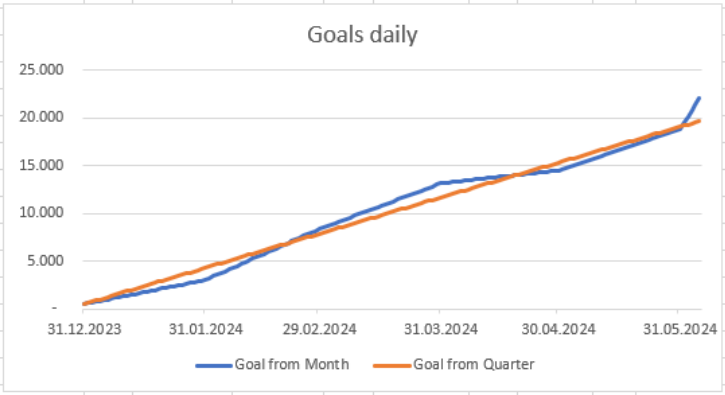
Der Startwert in Zelle B8 bietet einen Freiheitsgrad. Bei Eingabe von etwa 2.000.000 erhalten Sie:



Ein weiteres Beispiel ist der Breakdown von quartärlichen oder monatlichen Daten auf tägliche Daten mit dieser Formel in L2:

```
=SVERWEIS(DATUM(JAHR($K3);MONAT($K3);0);K$2:L2;2)+(TAG($K3)) *
(SVERWEIS(DATUM(JAHR($K3);MONAT($K3);1);$A$2:$C$7;3)
- SVERWEIS(DATUM(JAHR($K3);MONAT($K3);0);K$2:L2;2) *
SVERWEIS(DATUM(JAHR($K3);MONAT($K3);1);$A$2:$C$7;2))
/ (SVERWEIS(DATUM(JAHR($K3);MONAT($K3);1);$A$2:$C$7;2) *
(1+SVERWEIS(DATUM(JAHR($K3);MONAT($K3);1);$A$2:$C$7;2))/2)
```

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Month**	**Days**	**Goal**	**% of Total**		**Quarter**	**Days**	**Total**			**Date**	**Goal from Month**	**Goal from Quarter**
2	01.01.2024	31	56.000	3,5%		01.01.2024	91	560000	35%		31.12.2023	500	500
3	01.02.2024	29	168.000	10,5%		01.04.2024	66	1040000	65%		01.01.2024	582	623
4	01.03.2024	31	336.000	21,0%							02.01.2024	663	746
5	01.04.2024	30	416.000	26,0%							03.01.2024	745	869
6	01.05.2024	31	520.000	32,5%							04.01.2024	827	992
7	01.06.2024	5	104.000	6,5%							05.01.2024	908	1.115
8	**Total**	**157**	**1.600.000**	**100,0%**							06.01.2024	990	1.237
9											07.01.2024	1.072	1.360
10	**Forecast**										08.01.2024	1.153	1.483
11	**Month**	**Days**	**Goal**	**% of Total**							09.01.2024	1.235	1.606
12	Jan	31	56.000	2,6%							10.01.2024	1.317	1.729
13	Feb	29	168.000	7,9%							11.01.2024	1.398	1.852
14	Mar	31	336.000	15,8%							12.01.2024	1.480	1.975
15	Apr	30	416.000	19,6%							13.01.2024	1.561	2.098
16	May	31	520.000	24,5%							14.01.2024	1.643	2.221
17	Jun	30	624.000	29,4%							15.01.2024	1.725	2.344
18	**Total**	**182**	**2.120.000**	**100,0%**							16.01.2024	1.806	2.467
19											17.01.2024	1.888	2.589
20											18.01.2024	1.970	2.712
21											19.01.2024	2.051	2.835
22											20.01.2024	2.133	2.958
23											21.01.2024	2.215	3.081
24											22.01.2024	2.296	3.204
25											23.01.2024	2.378	3.327
26											24.01.2024	2.460	3.450
27											25.01.2024	2.541	3.573
28											26.01.2024	2.623	3.696
29											27.01.2024	2.705	3.819
30											28.01.2024	2.786	3.941
31											29.01.2024	2.868	4.064
32											30.01.2024	2.950	4.187
33											31.01.2024	3.031	4.310
34											01.02.2024	3.215	4.433



Falls dieser Ansatz Ihnen nicht ausreicht und Sie saisonale Gewichte ihrer Daten zur Verfügung haben, können Sie auch die bessere Methode Budgetkontrolle verwenden.

Minimum Truck Load Problem

Angenommen, ein Lieferant verkauft seine Ware nur in gewissen Mindestmengen, um die Warenlieferungen wirtschaftlich zu gestalten. So könnte z. B. die erste Mindestmenge eine volle Wagenladung sein, und ab dem zweiten Wagen immer mindestens eine halbe Wagenladung.

Beispiel: Eine volle Wagenladung besteht aus 10 Waren. Dies ist die minimale Bestellmenge. Danach muss immer mindestens eine halbe Wagenladung bestellt werden. Es soll verhindert werden, dass ein Truck mit weniger als einer halben Ladung fährt. Dies bedeutet, dass man nur diese Anzahl von Waren bestellen kann: 0, 15, 16, 17, 18, 19, 20, 25, 26, 27, 28, 29, 30, 35 …

Welche Formel berechnet zu einer Menge von benötigten Waren immer die entsprechende Mindestbestellmenge? Die allgemeine Lösung lautet:

	A	B
1	Truck Kapazität [Stück]	10
2	Minimale Bestellung [Trucks]	1
3	Minimale Beladung [%]	50%
4		
5	Eingabe [Stück]	Ausgabe [Akzeptierte Stückzahl]
6	0	0
7	1	10
8	2	10
9	3	10
10	4	10
11	5	10
12	6	10
13	7	10
14	8	10
15	9	10
16	10	10
17	11	19
18	12	19
19	13	19
20	14	19
21	15	19
22	16	19
23	17	19
24	18	19
25	19	19
26	20	20
27	21	29
28	22	29
29	23	29
30	24	29
31	25	29
32	26	29
33	27	29
34	28	29
35	29	29

Tabellenblattformeln		
Bereich	Formel	
B6	B6	=AUFRUNDEN(WENN(A6:A35<=0; 0; WENN(A6:A35<=$B$1*$B$2; $B$1*$B$2; $B$1*GANZZAHL(A6:A35/$B$1) + WENN(REST(A6:A35; $B$1) > 0; MAX(REST(A6:A35; $B$1); $B$1*$B$3); 0))); 0)

Nachstehende Nullen Zählen

Sie wollen die nachstehenden Nullen einer ganzen Zahl zählen? So kann es gemacht werden:

	A	B
1	Nachstehende Nullen zählen	
2	Eingabe	Ausgabe
3	300	2
4	6.000	3
5	1.200.000.000.000	11
6	-300	2
7	-6.000	3
8	-1.200.000.000.000	11

Tabellenblattformeln		
Bereich	Formel	
B3:B5	B3	=(A3<>0) * SUMME(--(A3=RUNDEN(A3; -SEQUENZ(15))))
B6:B9	B6	=LÄNGE(ABS(A6)) - LÄNGE(SUMME(TEIL(ABS(A6); LÄNGE(ABS(A6)) + 1 - SEQUENZ(LÄNGE(ABS(A6))); 1) * 10 ^ (LÄNGE(ABS(A6)) - SEQUENZ(LÄNGE(ABS(A6)))))

Die Formel `=(A3<>0) * SUMME(--(A3=RUNDEN(A3; -SEQUENZ(15))))` in Zelle B3 ist deutlich eleganter und kürzer.

REFA Zeitklassen

REFA – Verband für Arbeitsgestaltung, Betriebsorganisation und Unternehmensentwicklung e. V. ist Deutschlands älteste Organisation für Arbeitsgestaltung, Betriebsorganisation und Unternehmensentwicklung. REFA bietet Unterricht für Arbeitsorganisation an ähnlich Six Sigma oder Lean Management.

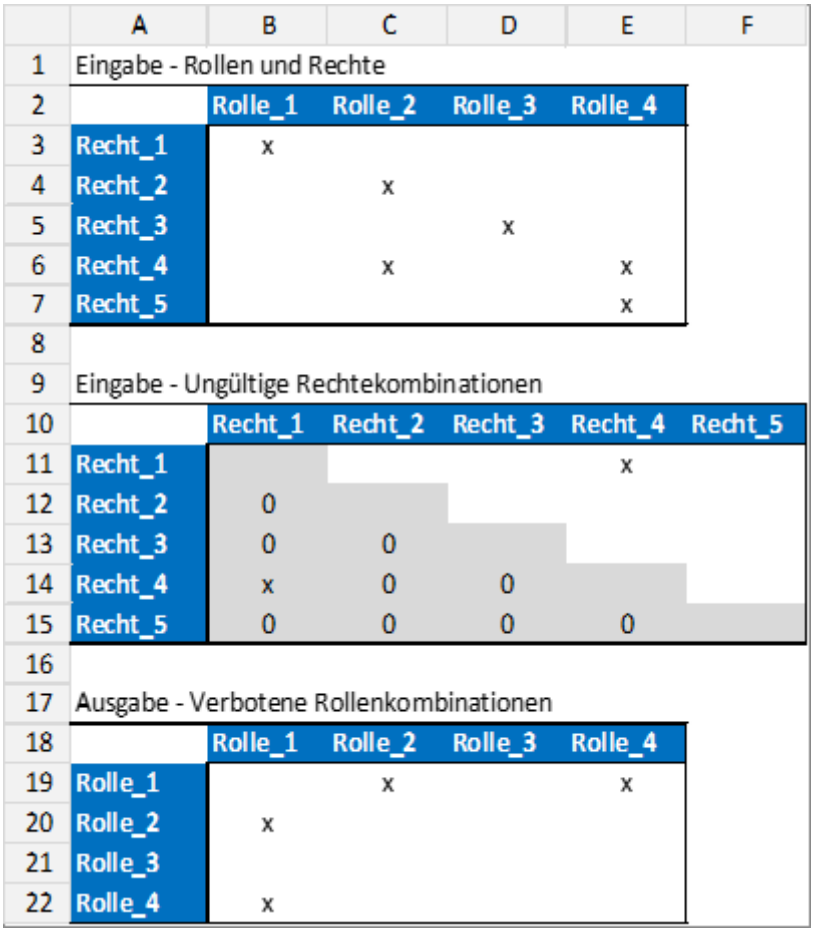
Wenn Sie Ihre monatlichen Hotline Anrufe analysieren möchten, können Sie diesen Ansatz für eine Zeitklassenschätzung verwenden:

	A	B	C	D	E	F
1	**Zeitklassenreihe nach REFA**					
2						
3	**Beispiel Hotline**				**Comment**	**Mögliche Warn- und Fehlerhinweise**
4	**Relativer Vertrauensbereich [%]**		2		Vertrauensbereich üblicherweise zwischen 0,1 und 5%	
5	**Dauer der Abrechnungsperiode [min]**		10080	A typical work month	Abrechnungsperiode größer als größte vorkommende Grundzeit?	
6	**Kleinste vorkommende Grundzeit [min]**		2	Shortest Hotline call		
7	**Größte vorkommende Grundzeit [min]**		60	Longest Hotline call	Größte vorkommende Grundzeit größer als kleinste?	
8	**Klassenuntergrenze der 1 Klasse [min]**		1		Klassenuntergrenze der 1. Klasse 50 - 80% der kleinsten vorkommenden Grundzeit?	
9						
10	**Warn- und Fehlerhinweise**		Alles ok			
11	Erklärung: Wenn man eine methodische Untersuchung z. B. von Hotlinezeiten vornehmen will (welche Themen wurden in welcher Zeit behandelt), dann sagt dieses Beispiel für einen Monat Zeitaufschreibung bei einem Vertrauensbereich von 1%, dass die Hotline-Mitarbeiter jedes Gespräch grob abschätzen sollen, in welcher der vier Zeitklassen es liegt und welches Thema behandelt wurde.					
12	**Hilfsspalte**	**Zeitklassennummer**	**Klassenbreite**	**Klassenuntergrenze**	**Klassenmitte**	**Klassenobergrenze**
13	4,861392064	1	10	1	6	11
14	8,974179072	2	18	11	20	29
15	13,01564799	3	26	29	42	55
16	17,04345008	4	34	55	72	89

Tabellenblattformeln		
Bereich	Formel	
F4	**F4**	=WENN(ODER($C$4<0,1;$C$4>5);"Vertrauensbereich liegt üblicherweise zwischen 0,1 und 5%.";"")
F5	**F5**	=WENN($C$5<$C$7;"Abrechnungsperiode ist kleiner als größte vorkommende Grundzeit.";"")
F7	**F7**	=WENN($C$7<=$C$6;"Größte vorkommende Grundzeit ist nicht größer als kleinste.";"")
F8	**F8**	=WENN(ODER($C$8<0,5*$C$6;$C$8>0,8*$C$6);"Die Klassenuntergrenze der 1. Klasse sollte 50 - 80% der kleinsten vorkommenden Grundzeit betragen.";"")
C10	**C10**	=WENN($F$4&$F$5&$F$6&$F$7&$F$8="";"Alles ok";$F$4&$F$5&$F$6&$F$7&$F$8)
A13:A22	**A13**	=($C$4%)^2*$C$5/2+(($C$4%)^2*$C$5/2)^2+($C$4%)^2*$C$5*D13)^0,5
B13	**B13**	=SEQUENZ(10)
C13	**C13**	=RUNDEN(A13:A22*2;0)
D13	**D13**	=$C$8
D14	**D14**	=RUNDEN(D13+2*A13;0)
E13	**E13**	=D13:D22+RUNDEN(A13:A22;0)
F13	**F13**	=D13:D22+RUNDEN(2*A13:A22;0)

Rollen und Rechte

Bei einer gegebenen Tabelle mit Rollen und Rechten, die mit den Rollen verknüpft sind, sowie einer Tabelle von ungültigen Rechtekombinationen, d.h. die angibt, welche Rechte nicht zusammen vergeben werden dürfen, wie können Sie nun die ungültigen Rollenkombinationen ermitteln, d.h. die Tabelle die zeigt, welche Rollen nicht miteinander gekoppelt werden dürfen?



	A	B	C	D	E	F
1	Eingabe - Rollen und Rechte					
2		Rolle_1	Rolle_2	Rolle_3	Rolle_4	
3	Recht_1	x				
4	Recht_2		x			
5	Recht_3			x		
6	Recht_4		x		x	
7	Recht_5				x	
8						
9	Eingabe - Ungültige Rechtekombinationen					
10		Recht_1	Recht_2	Recht_3	Recht_4	Recht_5
11	Recht_1				x	
12	Recht_2	0				
13	Recht_3	0	0			
14	Recht_4	x	0	0		
15	Recht_5	0	0	0	0	
16						
17	Ausgabe - Verbotene Rollenkombinationen					
18		Rolle_1	Rolle_2	Rolle_3	Rolle_4	
19	Rolle_1		x		x	
20	Rolle_2	x				
21	Rolle_3					
22	Rolle_4	x				

Beispiel:

Rolle 1 beinhaltet Recht 1, Rolle 2 enthält die Rechte 2 und 4, und Rolle 4 enthält die Rechte 4 und 5.

Wenn die Rechte 1 und 4 nicht zusammen vergeben werden dürfen, dann ist die interessante Schlussfolgerung, dass die Rolle 1 weder mit der Rolle 2 noch mit der Rolle 4 zusammen vergeben werden darf.

Die graue Fläche in der zweiten Tabelle spiegelt die Werte der weißen Zellen in derselben Tabelle.

Tabellenblattformeln		
Bereich	Formel	
B12	**B12**	=MTRANS(C11:F11)
C13	**C13**	=MTRANS(D12:F12)
D14	**D14**	=MTRANS(E13:F13)
E15	**E15**	=F14
B12	**B12**	=MTRANS(C11:F11)
B19	**B19**	=WIEDERHOLEN("x"; VORZEICHEN(MMULT(--(MTRANS(B3:E7)="x");MMULT(--(B11:F15="x"); --(B3:E7="x")))))

Rundungstricks

	A	B	C
1	**Rundungstricks**	**Eingabe**	**Ausgabe**
2	Zu einer ganzen Zahl runden	3,141592654	3
3	Auf 2 Stellen runden	3,141592654	3,14
4	Auf den nächsten Tausender runden	2020	2000
5	Auf 50 Cent runden	5,672	5,5
6	Abrunden wenn 35 Cent oder weniger sonst aufrunden	2,36	3
7	Auf 1/64 runden	2,1718	2,171875
8	Zeige einen Bond Preis im US T-Note Format (siehe Bloomberg ©)	100,578125	100-18,5
9	Zeige einen Bond Preis im US T-Note Format (siehe Bloomberg ©), Version 2	100,578125	100-18½
10	Runde eine Zeit auf eine ganze Stunde	26.04.2010 16:37:12	26.04.2010 17:00:00
11	Runden eine Zeit auf 10 Minuten	26.04.2010 16:37:12	26.04.2010 16:40:00
12	Runde auf 1/8 aber zeige 1/4, 1/2, 3/4 anstatt 2/8, 4/8, 6/8	-2,5	-2 1/2

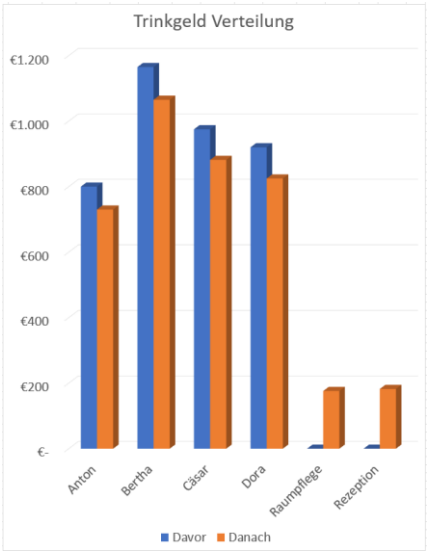
Tabellenblattformeln		
Bereich	Formel	
C2	**C2**	=RUNDEN(B2;0)
C3	**C3**	=RUNDEN(B3;2)
C4	**C4**	=RUNDEN(B4;-3)
C5	**C5**	=RUNDEN(B5 * 2;0) / 2
C6	**C6**	=OBERGRENZE(B6 - 0,35;1)
C7	**C7**	=RUNDEN(B7 * 64;0) / 64
C8	**C8**	=GANZZAHL(B8) & WIEDERHOLEN("-" & REST(B8;1) * 32; VORZEICHEN(REST(B8;1)))
C9	**C9**	=GANZZAHL(B9) & WIEDERHOLEN("-" & GANZZAHL(REST(B9;1) * 32) & WIEDERHOLEN(ZEICHEN(187 + GANZZAHL((REST(B9;1) - GANZZAHL(REST(B9;1) * 32) / 32) * 128)); GANZZAHL((REST(B9;1) - GANZZAHL(REST(B9;1) * 32) / 32) * 128) > 0); GANZZAHL(REST(B9;1) * 128) > 0)
C10	**C10**	=RUNDEN(B10 * 24;0) / 24
C11	**C11**	=RUNDEN(B11 * 24 * 6;0) / 24 / 6
C12	**C12**	=WAHL(2 + VORZEICHEN(B12);"-";"0";"") & WENN(GANZZAHL(ABS(B12)) <> 0;GANZZAHL(ABS(B12)) & " ";"") & WAHL(1 + RUNDEN((ABS(B12) - GANZZAHL(ABS(B12))) * 8;0);"";"1/8";"1/4";"3/8";"1/2";"5/8";"3/4";"7/8")

Bitte beachten: Zelle D5: =RUNDEN(B5 * 2;0) / 2 ist besser als =RUNDEN(B5 / 5;1) * 5, weil die interne binäre Darstellung von Fließkommazahlen bei der Multiplikation bzw. Division mit/durch 2 weniger Rechenungenauigkeiten aufweist. Zelle C8: Dieses Ergebnis ist die maximale Zahl im T-Note Format, die kleiner oder gleich der eingegebenen Zahl in Zelle B9 ist.

Trinkgeld Verteilung

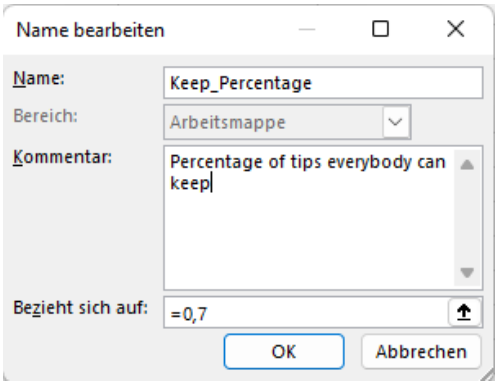
“Flattery is nice but tips will suffice.”

Sie wollen alle Trinkgelder Ihrer Mitarbeitenden fair aufteilen? Jeder soll 70% aller direkt erhaltenen Trinkgelder behalten, aber 30% sollen auf Basis gearbeiteter Stunden verteilt werden, damit auch Reinigungskräfte und Rezeptionsangestellte etwas bekommen?



Tag	Gesamt		Anton		Bertha		Cäsar		Dora		Raumpflege		Rezeption	
	€	Std.	€	Std.	€	Std.	€	Std.	€	Std.	€	Std.	€	Std.
1	115,00 €	31,0	10,00 €	1,0	35,00 €	7,0	40,00 €	8,0	30,00 €	6,0		5,0		4,0
2	65,00 €	22,0	10,00 €	1,0	10,00 €	2,0	5,00 €	1,0	40,00 €	8,0		3,0		7,0
3	140,00 €	19,0	- €	0,0	50,00 €	5,0	40,00 €	8,0	50,00 €	5,0		1,0		0,0
4	165,00 €	32,0	20,00 €	2,0	35,00 €	7,0	80,00 €	8,0	30,00 €	3,0		4,0		8,0
5	205,00 €	41,0	15,00 €	3,0	80,00 €	8,0	40,00 €	8,0	70,00 €	7,0		7,0		8,0
6	110,00 €	22,0	- €	0,0	20,00 €	4,0	70,00 €	7,0	20,00 €	2,0		1,0		8,0
7	140,00 €	22,0	30,00 €	6,0	70,00 €	7,0	- €	0,0	40,00 €	4,0		2,0		3,0
8	60,00 €	18,0	10,00 €	2,0	40,00 €	8,0	5,00 €	1,0	5,00 €	1,0		5,0		1,0
9	185,00 €	33,0	50,00 €	5,0	15,00 €	3,0	40,00 €	4,0	80,00 €	8,0		6,0		7,0
10	160,00 €	28,0	30,00 €	3,0	40,00 €	8,0	80,00 €	8,0	10,00 €	1,0		3,0		5,0
11	195,00 €	25,0	60,00 €	6,0	50,00 €	5,0	80,00 €	8,0	5,00 €	1,0		3,0		2,0
12	200,00 €	25,0	25,00 €	5,0	80,00 €	8,0	80,00 €	8,0	15,00 €	3,0		0,0		1,0
13	65,00 €	13,0	40,00 €	4,0	- €	0,0	25,00 €	5,0	- €	0,0		1,0		3,0
14	85,00 €	23,0	15,00 €	3,0	20,00 €	4,0	40,00 €	8,0	10,00 €	2,0		5,0		1,0
15	165,00 €	25,0	70,00 €	7,0	25,00 €	5,0	50,00 €	5,0	20,00 €	2,0		5,0		1,0
16	90,00 €	22,0	40,00 €	4,0	30,00 €	6,0	15,00 €	3,0	5,00 €	1,0		5,0		3,0
17	90,00 €	27,0	20,00 €	4,0	- €	0,0	30,00 €	3,0	40,00 €	8,0		6,0		6,0
18	135,00 €	22,0	50,00 €	5,0	35,00 €	7,0	20,00 €	2,0	30,00 €	3,0		0,0		5,0
19	125,00 €	21,0	40,00 €	8,0	70,00 €	7,0	- €	0,0	15,00 €	3,0		3,0		0,0
20	100,00 €	21,0	15,00 €	3,0	70,00 €	7,0	5,00 €	1,0	10,00 €	1,0		1,0		8,0
21	140,00 €	27,0	15,00 €	3,0	60,00 €	6,0	5,00 €	1,0	60,00 €	6,0		8,0		3,0
22	105,00 €	18,0	50,00 €	5,0	5,00 €	1,0	35,00 €	7,0	15,00 €	3,0		1,0		1,0
23	60,00 €	14,0	- €	0,0	60,00 €	6,0	- €	0,0	- €	0,0		1,0		7,0
24	75,00 €	22,0	35,00 €	7,0	20,00 €	4,0	- €	0,0	20,00 €	4,0		7,0		0,0
25	130,00 €	33,0	35,00 €	7,0	25,00 €	5,0	40,00 €	4,0	30,00 €	3,0		8,0		6,0
26	180,00 €	28,0	30,00 €	3,0	50,00 €	5,0	30,00 €	6,0	70,00 €	7,0		7,0		0,0
27	110,00 €	34,0	10,00 €	2,0	40,00 €	8,0	20,00 €	4,0	40,00 €	8,0		6,0		6,0
28	155,00 €	25,0	15,00 €	3,0	60,00 €	6,0	60,00 €	6,0	20,00 €	2,0		6,0		2,0
29	130,00 €	27,0	30,00 €	3,0	20,00 €	4,0	- €	0,0	80,00 €	8,0		5,0		7,0
30	85,00 €	18,0	10,00 €	2,0	15,00 €	3,0	20,00 €	4,0	40,00 €	4,0		0,0		5,0
31	95,00 €	18,0	20,00 €	4,0	35,00 €	7,0	20,00 €	2,0	20,00 €	4,0		0,0		1,0
Summe	3.860,00 €	756,0	800,00 €	111,0	1.165,00 €	163,0	975,00 €	130,0	920,00 €	118,0	- €	115,0	- €	119,0
Behalten	2.702,00 €		560,00 €		815,50 €		682,50 €		644,00 €		- €		- €	
Abgeben		1158,0		240,00 €		349,50 €		292,50 €		276,00 €		- €		- €
Bekommen	1.158,00 €		170,02 €		249,67 €		199,13 €		180,75 €		176,15 €		182,28 €	
Gesamt	3.860,00 €		730,02 €		1.065,17 €		881,63 €		824,75 €		176,15 €		182,28 €	

Die Eingabe des Prozentsatzes, den jeder behalten kann, sowie die verwenden Formeln:

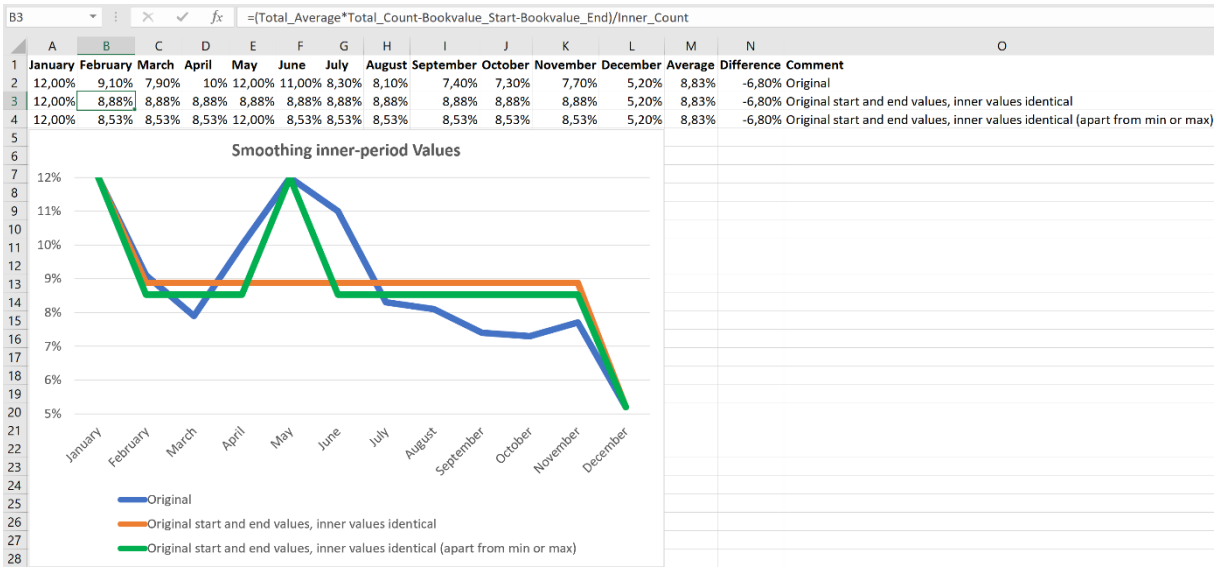


Tabellenblattformeln		
Bereich	Formel	
B3:B34;B36;B38;B40	**B3**	=SUMME(D3;F3;H3;J3;L3;N3)
C3:C34;C36;C38;C49	**C3**	=SUMME(E3;G3;I3;K3;M3;O3)
B34:O34	**B34**	=SUMME(B3:B33)
D36;F36;H36;J36;L36;N36	**D36**	=D34*Keep_Percentage
E37;G37;I37;K37;M37;O37	**E37**	=D34-D36
D38;F38;H38;J38;L38;N38	**D38**	=$C$37*E34/$C$34
D40;F40;H40;J40;L40;N40	**D40**	=D36+D38

Unterjährige Werte Glätten

Sie wollen unterjährige Werte glätten und dabei die Periodenendwerte und den Gesamtdurchschnitt beibehalten?

Zwei simple Möglichkeiten:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	January	February	March	April	May	June	July	August	September	October	November	December	Average	Difference	Comment
2	12,00%	9,10%	7,90%	10%	12,00%	11,00%	8,30%	8,10%	7,40%	7,30%	7,70%	5,20%	8,83%	-6,80%	Original
3	12,00%	8,88%	8,88%	8,88%	8,88%	8,88%	8,88%	8,88%	8,88%	8,88%	8,88%	5,20%	8,83%	-6,80%	Original start and end values, inner values identical
4	12,00%	8,53%	8,53%	8,53%	12,00%	8,53%	8,53%	8,53%	8,53%	8,53%	8,53%	5,20%	8,83%	-6,80%	Original start and end values, inner values identical (apart from min or max)

Definieren Sie diese Namen:

Name	Bezieht sich auf
All_Values	=Sheet1!$A$2:$K$2
Bookvalue_End	=Sheet1!$L$2
Bookvalue_Start	=Sheet1!$A$2
Inner_Count	=(Total_Count-2)
Inner_Values	=Sheet1!$B$2:$K$2
Total_Average	=MITTELWERT(All_Values)
Total_Count	=SPALTE(Bookvalue_End)-SPALE(Bookvalue_Start)+1

Wenn alle unterjährigen Werte identisch sein sollen, geben Sie ein:

```
A3: =Bookvalue_Start
B3: =(@Total_Average*@Total_Count-Bookvalue_Start-Bookvalue_End)/@Inner_Count
```

und nach rechts kopieren bis K3

```
L3: =Bookvalue_End
```

Wenn alle unterjährigen Werte (mit Ausnahme eines oder mehrerer Extremwerte) identisch sein sollen, geben Sie ein:

```
A3: =Bookvalue_Start
B3: =WENN(ABS(@Total_Average-MAX(Inner_Values))>=ABS(@Total_Average-
MIN(Inner_Values));WENN(B$2=MAX(Inner_Values);B$2;(@Total_Average*@Total_Count-Bookvalue_Start-
Bookvalue_End-MAX(Inner_Values)*ZÄHLENWENN(Inner_Values;MAX(Inner_Values)))/(@Inner_Count-
ZÄHLENWENN(Inner_Values;MAX(Inner_Values))));WENN(B$2=MIN(Inner_Values);B$2;(@Total_Average*@Total_Count-
Bookvalue_Start-Bookvalue_End-MIN(Inner_Values)*ZÄHLENWENN(Inner_Values;MIN(Inner_Values)))/(@Inner_Count-
ZÄHLENWENN(Inner_Values;MIN(Inner_Values)))))
```

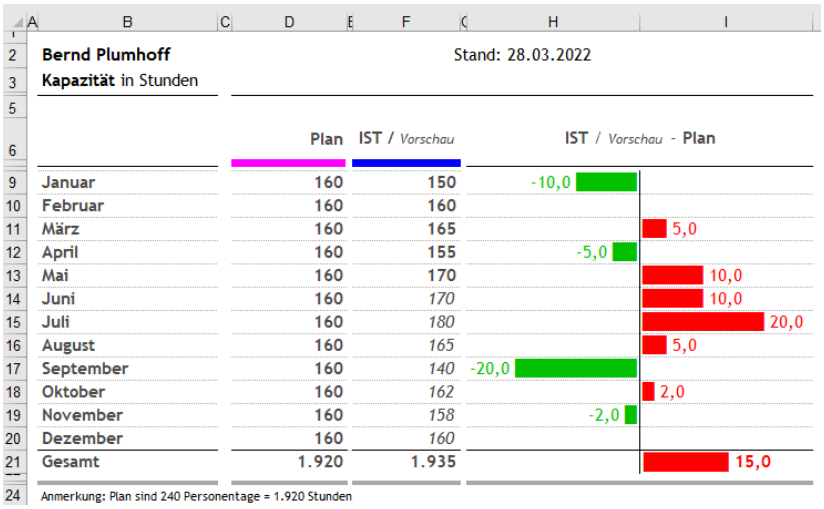
und nach rechts kopieren bis K3

```
L3: =Bookvalue_End
```

Zellenbasiertes Diagramm

Dietmar P. stellte ein zellenbasiertes Diagramm bei der Controller Akademie vor. Ich änderte und erweiterte es etwas, z. B. mit automatischer Skalierung und zwei neuen Parametern:

- Ein Nummernformat, um eine allgemeine Formatänderung zu ermöglichen.
- Einen Wahrheitswert-Parameter, um ggf. die Farben Grün und Rot vertauschen zu können. Wenn man einen Berater beauftragt, möchte man vielleicht ein Unterschreiten des Budgets grün anzeigen, aber im Falle von Verkaufszahlen sollte es besser rot sein.



	B	D	F	H	I
2	**Bernd Plumhoff**		Stand: 28.03.2022		
3	**Kapazität** in Stunden				
6		**Plan**	**IST** / *Vorschau*	**IST** / *Vorschau* - **Plan**	
9	Januar	160	150	-10,0	
10	Februar	160	160		
11	März	160	165		5,0
12	April	160	155	-5,0	
13	Mai	160	170		10,0
14	Juni	160	*170*		10,0
15	Juli	160	*180*		20,0
16	August	160	*165*		5,0
17	September	160	*140*	-20,0	
18	Oktober	160	*162*		2,0
19	November	160	*158*	-2,0	
20	Dezember	160	*160*		
21	Gesamt	1.920	1.935		15,0
24	Anmerkung: Plan sind 240 Personentage = 1.920 Stunden				

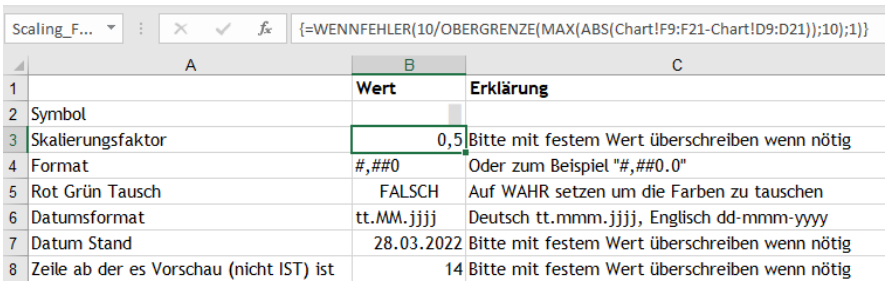
Formel in H9:

```
=WIEDERHOLEN(TEXT(F9-D9;Format)&" ";F9-D9<0)&WIEDERHOLEN(Symbol;(F9-D9<0)*ABS(RUNDEN(F9-
D9;0))*Scaling_Factor)
```

In I9:

```
=WIEDERHOLEN (Symbol;(F9-D9>0)*ABS(RUNDEN (F9-D9;0))*Scaling_Factor)&WIEDERHOLEN(" "&TEXT(F9-D9;Format);F9-
D9>0)
```

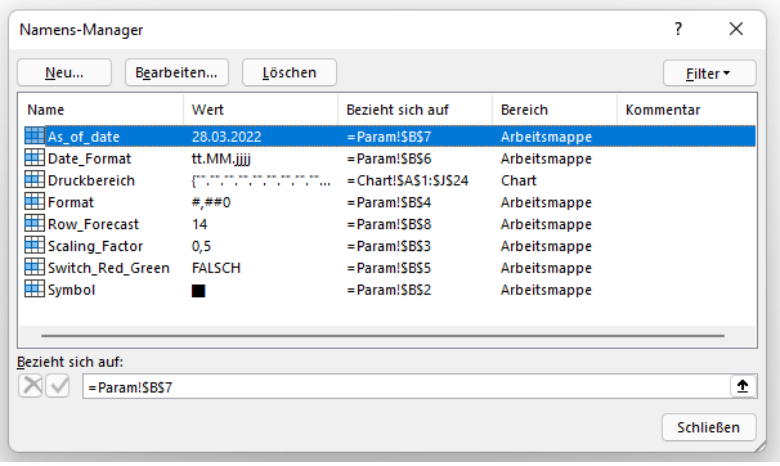
Das Tabellenblatt Param enthält alle Parameter:



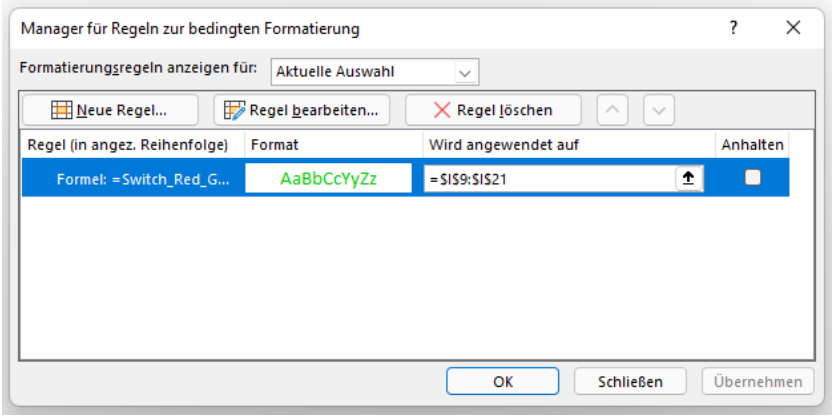
Scaling_F... | {=WENNFEHLER(10/OBERGRENZE(MAX(ABS(Chart!F9:F21-Chart!D9:D21));10);1)}

	A	B	C
1		**Wert**	**Erklärung**
2	Symbol		
3	Skalierungsfaktor	0,5	Bitte mit festem Wert überschreiben wenn nötig
4	Format	#,##0	Oder zum Beispiel "#,##0.0"
5	Rot Grün Tausch	FALSCH	Auf WAHR setzen um die Farben zu tauschen
6	Datumsformat	tt.MM.jjjj	Deutsch tt.mmm.jjjj, Englisch dd-mmm-yyyy
7	Datum Stand	28.03.2022	Bitte mit festem Wert überschreiben wenn nötig
8	Zeile ab der es Vorschau (nicht IST) ist	14	Bitte mit festem Wert überschreiben wenn nötig

Diese Parameter haben Namen wie im Namensmanager gezeigt:



Eine mögliche Vertauschung von Grün und Rot wurde mit einem bedingten Format implementiert, z. B. in Spalte I:



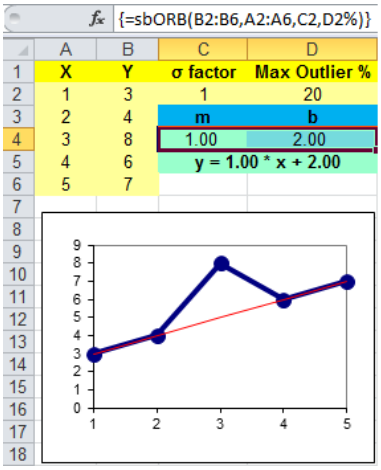
Einfache VBA Programme

Abstract

Mit einfachen VBA Programmen kann man Lösungen für fehlende Excel Funktionalitäten erzeugen.

Ausreißer Werte eliminieren – sbORB

Echtdaten enthalten manchmal extreme Werte oder Ausreißer (engl. Outlier), die Sie ignorieren oder löschen möchten:



sbORB Programmcode

```
Function sbORB(rY As Range, rX As Range, _
  Optional dSigmaFactor As Double = 3#, _
  Optional dMaxOutlierPercentage As Double = 0.5) As Variant
'sbORB() = outlier resistant beta returns a beta and
'an alpha where y = beta * x + alpha is most accurate
'for (almost) all given x in rX and y in rY.
'"Almost" means that we successively (one by one) throw out outliers
'which have a distance of > dSigmaFactor * STDEV_of_all_Distances
'from the least square (LS) proxy.
'(C) (P) by Bernd Plumhoff 24-Jun-2007 PB V0.9
Dim vLinEst As Variant 'store LinEst() result of recent LS proxy during iterations
Dim dm2 As Double 'ortogonal slope to recent LS proxy
Dim dc As Double 'Constant c in: y2=m2*x2+c which is ortogonal to LS proxy through a given point
Dim dx2 As Double 'x2 in: y2 = m2 * x2 + c which is ortogonal to LS proxy through a given point
Dim dy2 As Double 'y2 in: y2 = m2 * x2 + c which is ortogonal to LS proxy through a given point
Dim i As Long, j As Long
Dim lcount As Long 'holds current number of live points
Dim lcount_orig As Long 'original (starting) number of points
Dim lcount_old As Long 'holds number of live points of previous iteration
Dim daverage As Double 'average of distances to LS proxy of current iterations' live points
Dim dstdev As Double 'Stdev of distances to LS proxy of current iterations' live points
Dim dDistMax As Double
Dim lDistMaxIdx As Long

lcount = rX.Rows.Count
If rX.Columns.Count > lcount Then
  lcount = rX.Columns.Count
End If
lcount_orig = lcount
lcount_old = lcount + 1

ReDim dDist(1 To lcount) As Double 'store distances of live points to recent LS proxy (line)
ReDim dX(1 To lcount) As Double
ReDim dY(1 To lcount) As Double    'store coordinates of "live" points during iterations

'read data row-wise or column-wise
```

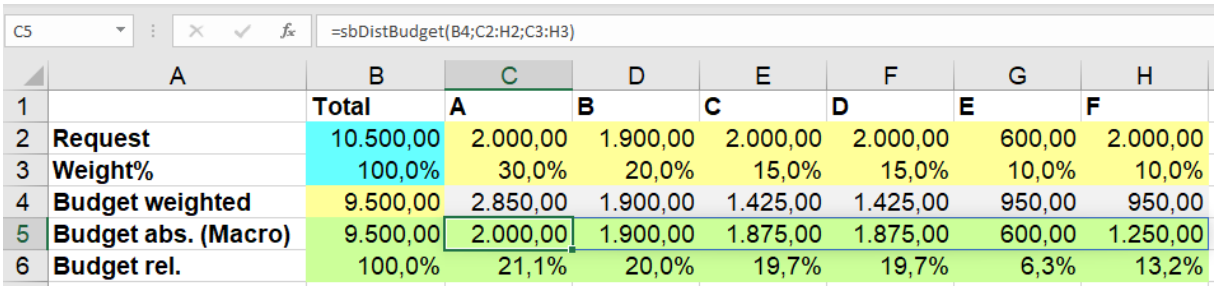
```
If rX.Rows.Count > rX.Columns.Count Then
  For i = 1 To lcount
    dX(i) = rX.Cells(i, 1)
    dY(i) = rY.Cells(i, 1)
  Next i
Else
  For i = 1 To lcount
    dX(i) = rX.Cells(1, i)
    dY(i) = rY.Cells(1, i)
  Next i
End If

Do
  lcount_old = lcount
  ReDim Preserve dDist(1 To lcount) As Double 'Store distances of live points to last LS proxy
  ReDim Preserve dX(1 To lcount) As Double
  ReDim Preserve dY(1 To lcount) As Double 'Store coordinates of "live" points during iterations
  vLinEst = Application.WorksheetFunction.LinEst(dY, dX, True, True)
  dDistMax = 0#
  lDistMaxIdx = 1
  For i = 1 To lcount
    'Calculate distances of live points to recent LS proxy
    dm2 = -1# / vLinEst(1, 1)
    dc = dY(i) - dX(i) * dm2
    dx2 = (dc - vLinEst(1, 2)) / (vLinEst(1, 1) - dm2)
    dy2 = dm2 * dx2 + dc
    dDist(i) = Sqr((dX(i) - dx2) * (dX(i) - dx2) + (dY(i) - dy2) * (dY(i) - dy2))
    'remember largest distance and its index
    If dDist(i) > dDistMax Then
      dDistMax = dDist(i)
      lDistMaxIdx = i
    End If
  Next i
  'calculate average and standard deviation of live points' distances to LS proxy
  daverage = Application.WorksheetFunction.Average(dDist)
  dstdev = Application.WorksheetFunction.StDev(dDist)
  'kill points with distance > dSigmaFactor * dstdev 'Attention: might erase too many points
'  j = 1
'  For i = 1 To lcount
'    If dDist(i) <= dstdev * dSigmaFactor Then
'      dX(j) = dX(i)
'      dY(j) = dY(i)
'      j = j + 1
'    Else
'      Debug.Print "Lcount: " & lcount & ". Throwing out (" & dX(i) & ";" & dY(i) & ")"
'    End If
'  Next i
'  lcount = j - 1
  'kill point with largest distance > dSigmaFactor * dstdev
  If dDist(lDistMaxIdx) >= dstdev * dSigmaFactor Then
    Debug.Print "Lcount: " & lcount & ". Throwing out (" & dX(lDistMaxIdx) & _
                ";" & dY(lDistMaxIdx) & ")"
    dX(lDistMaxIdx) = dX(lcount)
    dY(lDistMaxIdx) = dY(lcount)
    lcount = lcount - 1
  End If
Loop While lcount_old > lcount And lcount / lcount_orig > 1# - dMaxOutlierPercentage

If lcount < lcount_old Then
  vLinEst = Application.WorksheetFunction.LinEst(dY, dX, True, True)
End If

sbORB = vLinEst

End Function
```

Budgetplanung – sbDistBudget

“Givers have to set limits because takers rarely do.” [Irma Kurtz]

Nehmen Sie, Ihr Unternehmen befindet sich mitten in seinem jährlichen Planungsprozess für Einnahmen und Ausgaben. Sie sind der Bereichsleiter von 6 Abteilungen (A bis F). Ihre Abteilungsleiter haben ein Budget von 2000, 1900, 2000, 2000, 600 und 2000 € beantragt, aber Ihnen wurde lediglich ein Gesamtbudget in Höhe von 9500 € zur Verfügung gestellt. Ihre Abteilungen tragen gewichtet 30%, 20%, 15%, 15%, 10% und 10% zu den Unternehmenseinnahmen bei.

Wie würden Sie Ihr Budget verteilen? Sicherlich würden Sie niemanden mehr geben als beantragt wurde:



	A	B	C	D	E	F	G	H
1		Total	A	B	C	D	E	F
2	Request	10.500,00	2.000,00	1.900,00	2.000,00	2.000,00	600,00	2.000,00
3	Weight%	100,0%	30,0%	20,0%	15,0%	15,0%	10,0%	10,0%
4	Budget weighted	9.500,00	2.850,00	1.900,00	1.425,00	1.425,00	950,00	950,00
5	Budget abs. (Macro)	9.500,00	2.000,00	1.900,00	1.875,00	1.875,00	600,00	1.250,00
6	Budget rel.	100,0%	21,1%	20,0%	19,7%	19,7%	6,3%	13,2%

sbDistBudget Programmcode

```
Function sbDistBudget(dBudget As Double, _
  vRequest As Variant, _
  vWeight As Variant) As Variant
'Distribute a budget fairly upon Ubound(vRequest)
'requestors according to their weight vWeight(i)
'but do not give them more than they requested.
'Iterative solution.
'(C) (P) by Bernd Plumhoff 03-Dec-2012 PB V0.22
Dim dSumRequest As Double
Dim dSumWeight As Double
Dim dSumDist As Double
Dim dBudgetRest As Double
Dim dMinRest As Double
Dim i As Long, lc As Long, lgtNull As Long
With Application
lc = vRequest.Count
If lc <> vWeight.Count Then
  sbDistBudget = CVErr(xlErrValue)
  Exit Function
End If
ReDim dWeight(1 To lc) As Double
ReDim vR(1 To lc) As Variant 'Result vector
ReDim vT(1 To lc) As Variant 'Temp vector
dSumRequest = .Sum(vRequest)
If dSumRequest <= dBudget Then
  'Easy case: budget >= requests
  For i = 1 To lc
    vR(i) = vRequest(i)
  Next i
  sbDistBudget = vR
  Exit Function
End If
'Initialize budget distribution
dBudgetRest = dBudget
For i = 1 To lc
  dWeight(i) = vWeight(i)
Next i
'Distribute budget
```

```
Do While dBudget > dSumDist
  dSumWeight = .Sum(dWeight)
  If dSumWeight > 0# Then
    For i = 1 To lc
      vT(i) = dWeight(i) * dBudgetRest / dSumWeight
      If vT(i) + vR(i) >= vRequest(i) Then
        vT(i) = vRequest(i) - vR(i)
        dWeight(i) = 0#
      End If
      vR(i) = vR(i) + vT(i)
    Next i
  Else
    lgtNull = 0
    dMinRest = dBudgetRest
    For i = 1 To lc
      vT(i) = .Max(vRequest(i) - vR(i), 0#)
      If vT(i) > 0# Then
        lgtNull = lgtNull + 1
        If dMinRest > vT(i) Then
          dMinRest = vT(i)
        End If
      End If
    Next i
    If lgtNull = 0 Then Exit Do
    If dMinRest > dBudgetRest / lgtNull Then
      dMinRest = dBudgetRest / lgtNull
    End If
    For i = 1 To lc
      If vT(i) > 0# Then
        vR(i) = vR(i) + dMinRest
        vT(i) = dMinRest
      End If
    Next i
  End If
  dBudgetRest = dBudgetRest - .Sum(vT)
  dSumDist = .Sum(vR)
Loop
End With
sbDistBudget = vR
End Function
```

Collatz Länge Berechnen - sbCollatz

“Mathematics is not yet ready for such problems.” [Paul Erdös]

Die Collatz Vermutung für natürliche Zahlen:

- Eine gerade Zahl wird halbiert
- Eine ungerade Zahl wird verdreifacht und um 1 erhöht

Wenn Sie die oben genannten Regeln wiederholt anwenden, gelangen Sie stets zur Zahl 1.

Beispiel: Wenn Sie mit 5 beginnen, erhalten Sie die Folge 5, 16, 8, 4, 2, 1, die eine Collatz Länge von 6 besitzt.

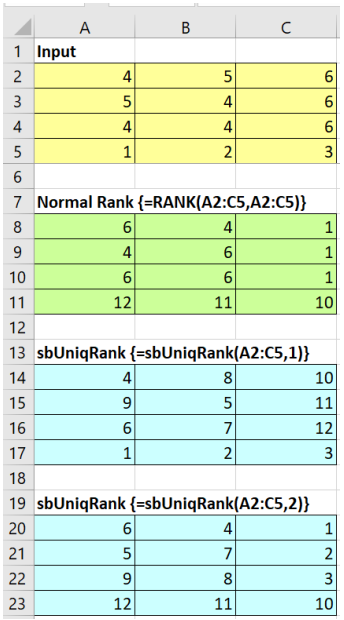
sbCollatz Programmcode

```
Function sbCollatz(s As String) As Long
'Calculates the Collatz length of a positive integer =
'returns count of iterations until result is 1.
'Excel is not the best tool to implement this but here we are:
'(C) (P) by Bernd Plumhoff 17-Jul-2022 PB V0.2
Dim b As Boolean, c As Integer
Dim i As Long, j As Long, k As Long, m As Long, n As Long, p As Long
n = Len(s)
m = n + 20 'We assume 20 additional digits will suffice
ReDim t(1 To m) As Integer
For i = 1 To n
    t(m - n + i) = Mid(s, i, 1)
Next i
b = False
For j = 1 To m - 1
  If t(j) <> 0 Then Exit For
Next j
k = 1
If j = m And t(m) < 2 Then
  t(m) = 1
  b = True
End If
Do While Not b
  k = k + 1
  Select Case t(m)
  Case 0, 2, 4, 6, 8
    'Divide by 2
    c = 0
    For j = 1 To m
      p = 5 * (t(j) Mod 2)
      t(j) = t(j) \ 2 + c
      c = p
    Next j
  Case 1, 3, 5, 7, 9
    'Multiply by 3 and add 1
    c = 1
    For j = m To 1 Step -1
      p = 3 * t(j) + c
      t(j) = p Mod 10
      c = p \ 10
    Next j
    Debug.Assert c = 0 'If we fail here the number of additional digits was too small
  Case Else
    Debug.Assert False
  End Select
  For j = 1 To m - 1
    If t(j) <> 0 Then Exit For
  Next j
  If j = m And t(m) = 1 Then b = True
  'If you like you can print out t() here
Loop
sbCollatz = k
End Function
```

Eindeutigen Rang auch bei Duplikaten vergeben – sbUniqRank

Sie benötigen eine RANG Funktion die einen eindeutigen Rang auch im Falle von Duplikaten vergibt? Ein möglicher Ansatz:

Drücken Sie die Tasten ALT + F11, fügen Sie ein neues Modul ein und kopieren Sie den unten gezeigten Programmcode in das neue Modul. Dann kehren Sie zu Ihrem Tabellenblatt zurück, wählen die Zellen A12:C15 aus und geben =sbUniqRank(A2:C5) mit STRG + SHIFT + ENTER als Matrixformel ein.



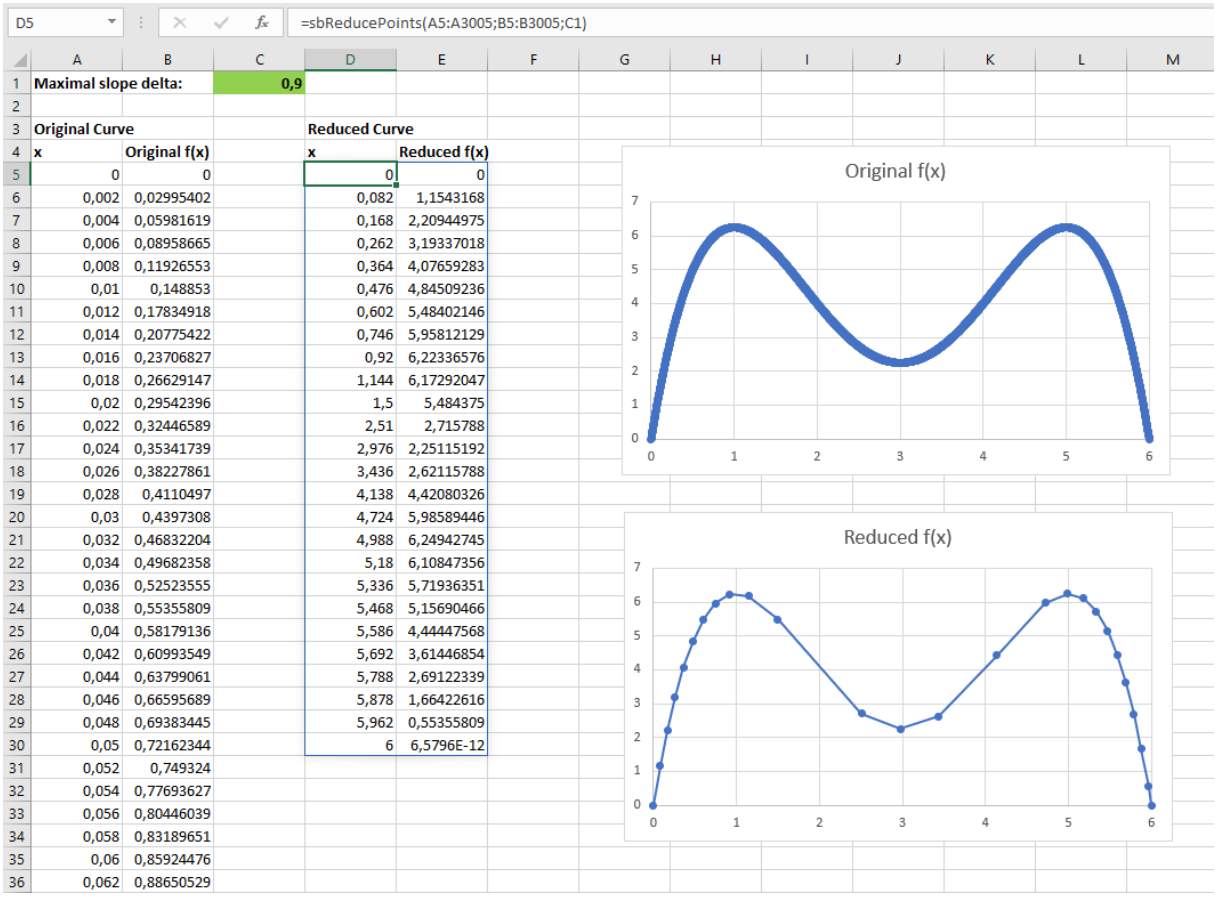
	A	B	C
1	**Input**		
2	4	5	6
3	5	4	6
4	4	4	6
5	1	2	3
6			
7	**Normal Rank {=RANK(A2:C5,A2:C5)}**		
8	6	4	1
9	4	6	1
10	6	6	1
11	12	11	10
12			
13	**sbUniqRank {=sbUniqRank(A2:C5,1)}**		
14	4	8	10
15	9	5	11
16	6	7	12
17	1	2	3
18			
19	**sbUniqRank {=sbUniqRank(A2:C5,2)}**		
20	6	4	1
21	5	7	2
22	9	8	3
23	12	11	10

sbUniqRank Programmcode

```
Function sbUniqRank(r As Range, _
    Optional vCountFrom As Variant = 1, _
    Optional bJustNumeric As Boolean = True, _
    Optional lOrder As Long = 0) As Variant
'Array function to rank a range with unique ranks.
'vCountFrom determines from where you count in case of duplicates:
'1 = first rows (1 to count), then columns (1 to count), i. e. top left to top right (tltr)
'2 = starting with top right to top left, then downwards (trtl)
'...
'8 = starting with bottom right to top right, then to the left (brtr)
'If bJustNumeric is True then Rank will be used to rank, if False then Countif will be used.
'lOrder is like Rank's order: 0 = Descending, 1 = Ascending
'(C) (P) by Bernd Plumhoff 25-Oct-2018 PB V0.6
Dim obj As Object
Dim bSwap As Boolean
Dim i As Long, i1 As Long, i2 As Long, i3 As Long
Dim j As Long, j1 As Long, j2 As Long, j3 As Long
Dim sComp As String
Dim vI As Variant, vR As Variant
vI = r: vR = vI
Set obj = CreateObject("Scripting.Dictionary")
Select Case vCountFrom
  Case 1, "tltr", "olor"
    i1 = 1: i2 = UBound(vI, 1): i3 = 1: j1 = 1: j2 = UBound(vI, 2): j3 = 1: bSwap = False
  Case 2, "trtl", "orol"
    i1 = 1: i2 = UBound(vI, 1): i3 = 1: j1 = UBound(vI, 2): j2 = 1: j3 = -1: bSwap = False
  Case 3, "blbr", "ulur"
    i1 = UBound(vI, 1): i2 = 1: i3 = -1: j1 = 1: j2 = UBound(vI, 2): j3 = 1: bSwap = False
  Case 4, "brbl", "urul"
    i1 = UBound(vI, 1): i2 = 1: i3 = -1: j1 = UBound(vI, 2): j2 = 1: j3 = -1: bSwap = False
  Case 5, "tlbl", "olul"
    i1 = 1: i2 = UBound(vI, 2): i3 = 1: j1 = 1: j2 = UBound(vI, 1): j3 = 1: bSwap = True
  Case 6, "bltl", "ulol"
    i1 = 1: i2 = UBound(vI, 2): i3 = 1: j1 = UBound(vI, 1): j2 = 1: j3 = -1: bSwap = True
  Case 7, "trbr", "orur"
    i1 = UBound(vI, 2): i2 = 1: i3 = -1: j1 = 1: j2 = UBound(vI, 1): j3 = 1: bSwap = True
  Case 8, "brtr", "uror"
    i1 = UBound(vI, 2): i2 = 1: i3 = -1: j1 = UBound(vI, 1): j2 = 1: j3 = -1: bSwap = True
  Case Else
    sbUniqRank = CVErr(xlErrValue)
    Exit Function
End Select
sComp = ">": If lOrder = 1 Then sComp = "<"
If bSwap Then
  'column - wise
  For i = i1 To i2 Step i3
    For j = j1 To j2 Step j3
      If bJustNumeric Then
        vR(j, i) = Application.WorksheetFunction.Rank(vI(j, i), r, lOrder) _
                 + obj.Item(vI(j, i))
      Else
        vR(j, i) = Application.WorksheetFunction.CountIf(r, _
                 sComp & vI(j, i)) + obj.Item(vI(j, i)) + 1
      End If
      obj.Item(vI(j, i)) = obj.Item(vI(j, i)) + 1
    Next j
  Next i
Else
  'row - wise
  For i = i1 To i2 Step i3
    For j = j1 To j2 Step j3
      If bJustNumeric Then
        vR(i, j) = Application.WorksheetFunction.Rank(vI(i, j), r, lOrder) _
                 + obj.Item(vI(i, j))
      Else
        vR(i, j) = Application.WorksheetFunction.CountIf(r, _
                 sComp & vI(i, j)) + obj.Item(vI(i, j)) + 1
      End If
      obj.Item(vI(i, j)) = obj.Item(vI(i, j)) + 1
    Next j
  Next i
End If
sbUniqRank = vR
End Function
```

Eliminiere Punkte eines Graphen mit kleiner Steigungsänderung – *sbReducePoints*

Falls Sie die Anzahl von Kurvenpunkten auf solche mit signifikanter Steigungsänderung reduzieren möchten:



D5 =sbReducePoints(A5:A3005;B5:B3005;C1)

	A	B	C	D	E
1	**Maximal slope delta:**		0,9		
2					
3	**Original Curve**			**Reduced Curve**	
4	**x**	**Original f(x)**		**x**	**Reduced f(x)**
5	0	0		0	0
6	0,002	0,02995402		0,082	1,1543168
7	0,004	0,05981619		0,168	2,20944975
8	0,006	0,08958665		0,262	3,19337018
9	0,008	0,11926553		0,364	4,07659283
10	0,01	0,148853		0,476	4,84509236
11	0,012	0,17834918		0,602	5,48402146
12	0,014	0,20775422		0,746	5,95812129
13	0,016	0,23706827		0,92	6,22336576
14	0,018	0,26629147		1,144	6,17292047
15	0,02	0,29542396		1,5	5,484375
16	0,022	0,32446589		2,51	2,715788
17	0,024	0,35341739		2,976	2,25115192
18	0,026	0,38227861		3,436	2,62115788
19	0,028	0,4110497		4,138	4,42080326
20	0,03	0,4397308		4,724	5,98589446
21	0,032	0,46832204		4,988	6,24942745
22	0,034	0,49682358		5,18	6,10847356
23	0,036	0,52523555		5,336	5,71936351
24	0,038	0,55355809		5,468	5,15690466
25	0,04	0,58179136		5,586	4,44447568
26	0,042	0,60993549		5,692	3,61446854
27	0,044	0,63799061		5,788	2,69122339
28	0,046	0,66595689		5,878	1,66422616
29	0,048	0,69383445		5,962	0,55355809
30	0,05	0,72162344		6	6,5796E-12
31	0,052	0,749324			
32	0,054	0,77693627			
33	0,056	0,80446039			
34	0,058	0,83189651			
35	0,06	0,85924476			
36	0,062	0,88650529			

Hier bestand der ursprüngliche Graph der Funktion f(x) = -0,25*(A5-3)^4 + 2*(A5-3)^2 + 2,25 aus 3001 Punkten. Durch den Aufruf *sbReducePoints*(A5:A3005; B5:B3005; 0,9) konnte die Anzahl der Punkte auf 26 reduziert werden.

Falls dieser simple Ansatz der Steigungsänderung nicht ausreicht, empfiehlt sich der Douglas-Peucker-Algorithmus.

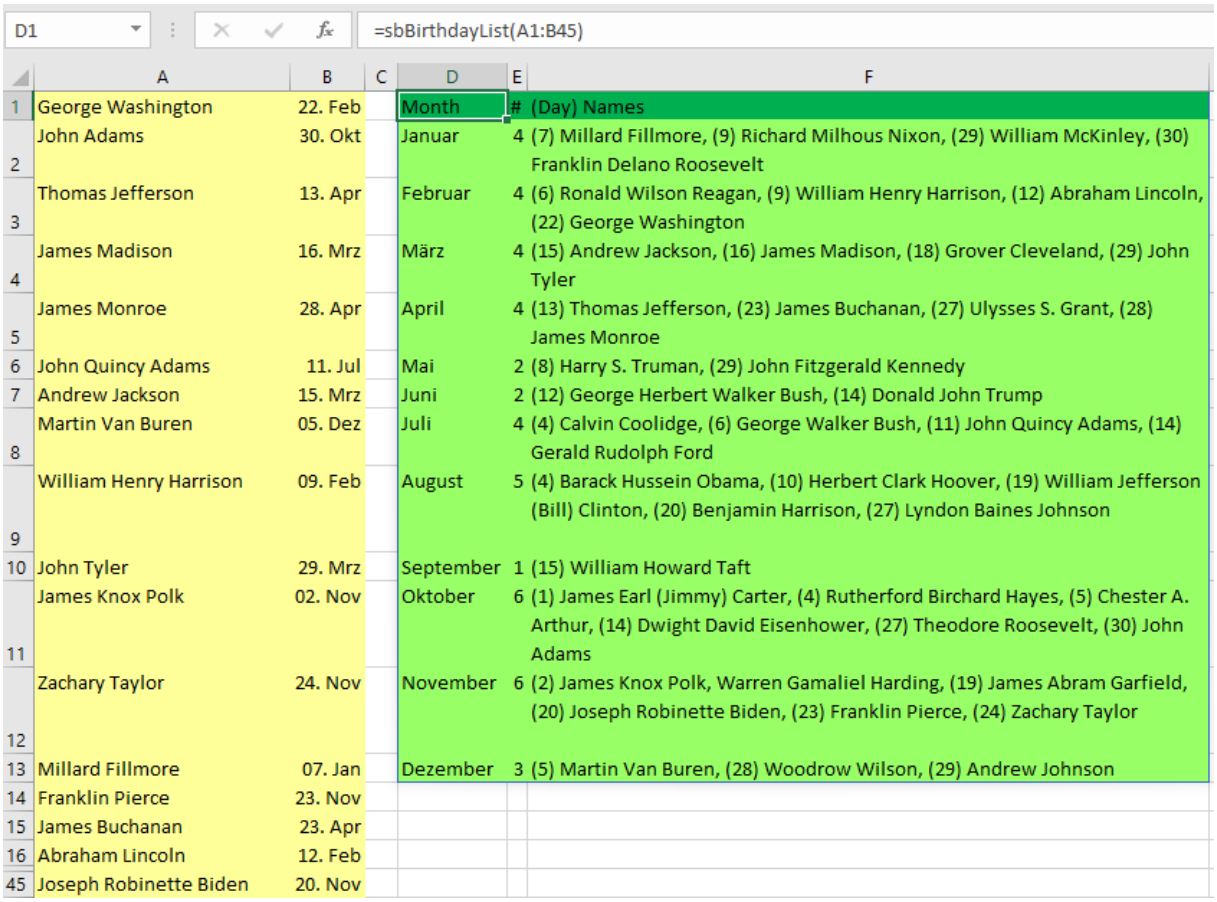
sbReducePoints Programmcode

```
Function sbReducePoints(rX As Range, rY As Range, _
    Optional dMaxSlopeDelta As Double = 0.001) As Variant
'sbReducePoints eliminates points from a given set
'in case the slopes between these points do not differ
'too much.
'(C) (P) by Bernd Plumhoff 29-Mar-2023 PB V0.1

Dim bNewSlope               As Boolean

Dim dSlope12                As Double
Dim dSlope13                As Double
Dim dSlope23                As Double

Dim i                       As Long
Dim k                       As Long
Dim lcount                  As Long

With Application.WorksheetFunction

lcount = rX.Rows.Count
If rX.Columns.Count > lcount Then
  lcount = rX.Columns.Count
End If

ReDim dX(1 To lcount) As Double
ReDim dY(1 To lcount) As Double

'read data row-wise or column-wise
If rX.Rows.Count > rX.Columns.Count Then
  For i = 1 To lcount
    dX(i) = rX.Cells(i, 1)
    dY(i) = rY.Cells(i, 1)
  Next i
Else
  For i = 1 To lcount
    dX(i) = rX.Cells(1, i)
    dY(i) = rY.Cells(1, i)
  Next i
End If

ReDim vR(1 To 2, 1 To lcount) As Variant

vR(1, 1) = dX(1)
vR(2, 1) = dY(1)
vR(1, 2) = dX(2)
vR(2, 2) = dY(2)
k = 2
bNewSlope = True
For i = 3 To lcount
  If bNewSlope Then dSlope12 = (vR(2, k) - vR(2, k - 1)) / (vR(1, k) - vR(1, k - 1))
  dSlope13 = (dY(i) - vR(2, k - 1)) / (dX(i) - vR(1, k - 1))
  dSlope23 = (dY(i) - vR(2, k)) / (dX(i) - vR(1, k))
  If Abs(dSlope13 - dSlope12) > dMaxSlopeDelta Or _
    Abs(dSlope13 - dSlope23) > dMaxSlopeDelta Then
    k = k + 1
    bNewSlope = True
  Else
    bNewSlope = False
  End If
  vR(1, k) = dX(i)
  vR(2, k) = dY(i)
Next i

ReDim Preserve vR(1 To 2, 1 To k) As Variant

If rX.Rows.Count > rX.Columns.Count Then
  sbReducePoints = .Transpose(vR)
Else
  sbReducePoints = vR
End If

End With

End Function
```

Geburtstagsliste – sbBirthdayList

“Nice to be here? At my age, it’s nice to be anywhere.” [George Burns]

Sie wollen wissen, wann es wieder Zeit für Kuchen in Ihrem Team ist? Dann erstellen Sie eine Geburtstagsliste:



D1 =sbBirthdayList(A1:B45)

	A	B	C	D	E	F
1	George Washington	22. Feb		Month	#	(Day) Names
2	John Adams	30. Okt		Januar	4	(7) Millard Fillmore, (9) Richard Milhous Nixon, (29) William McKinley, (30) Franklin Delano Roosevelt
3	Thomas Jefferson	13. Apr		Februar	4	(6) Ronald Wilson Reagan, (9) William Henry Harrison, (12) Abraham Lincoln, (22) George Washington
4	James Madison	16. Mrz		März	4	(15) Andrew Jackson, (16) James Madison, (18) Grover Cleveland, (29) John Tyler
5	James Monroe	28. Apr		April	4	(13) Thomas Jefferson, (23) James Buchanan, (27) Ulysses S. Grant, (28) James Monroe
6	John Quincy Adams	11. Jul		Mai	2	(8) Harry S. Truman, (29) John Fitzgerald Kennedy
7	Andrew Jackson	15. Mrz		Juni	2	(12) George Herbert Walker Bush, (14) Donald John Trump
8	Martin Van Buren	05. Dez		Juli	4	(4) Calvin Coolidge, (6) George Walker Bush, (11) John Quincy Adams, (14) Gerald Rudolph Ford
9	William Henry Harrison	09. Feb		August	5	(4) Barack Hussein Obama, (10) Herbert Clark Hoover, (19) William Jefferson (Bill) Clinton, (20) Benjamin Harrison, (27) Lyndon Baines Johnson
10	John Tyler	29. Mrz		September	1	(15) William Howard Taft
11	James Knox Polk	02. Nov		Oktober	6	(1) James Earl (Jimmy) Carter, (4) Rutherford Birchard Hayes, (5) Chester A. Arthur, (14) Dwight David Eisenhower, (27) Theodore Roosevelt, (30) John Adams
12	Zachary Taylor	24. Nov		November	6	(2) James Knox Polk, Warren Gamaliel Harding, (19) James Abram Garfield, (20) Joseph Robinette Biden, (23) Franklin Pierce, (24) Zachary Taylor
13	Millard Fillmore	07. Jan		Dezember	3	(5) Martin Van Buren, (28) Woodrow Wilson, (29) Andrew Johnson
14	Franklin Pierce	23. Nov				
15	James Buchanan	23. Apr				
16	Abraham Lincoln	12. Feb				
45	Joseph Robinette Biden	20. Nov				

sbBirthdayList Programmcode

```
Function sbBirthdayList(r As Range) As Variant
'Create monthly birthday list.
'(C) (P) by Bernd Plumhoff 15-Sep-2010 PB V0.10
Dim vR(1 To 13, 1 To 3) As Variant
Dim i As Long, j As Long
Dim sNames(101 To 1231) As String

'Fill temporary array
For i = 1 To r.Rows.Count
  If IsDate(r.Cells(i, 2)) Then
    j = Month(r.Cells(i, 2))
    vR(j + 1, 2) = vR(j + 1, 2) + 1 'Increasing DOB counter for month
    j = j * 100 + Day(r.Cells(i, 2))
    If sNames(j) <> "" Then sNames(j) = sNames(j) & ", "
    sNames(j) = sNames(j) & r.Cells(i, 1)
  End If
Next i

'Fill output area
vR(1, 1) = "Month"
vR(1, 2) = "#"
vR(1, 3) = "(Day) Names"
For i = 1 To 12
  vR(i + 1, 1) = Format(DateSerial(1900, i, 1), "MMMM")
  vR(i + 1, 3) = ""
  For j = 1 To 31
    If sNames(i * 100 + j) <> "" Then
```

```
      If vR(i + 1, 3) <> "" Then vR(i + 1, 3) = vR(i + 1, 3) & ", "
      vR(i + 1, 3) = vR(i + 1, 3) & "(" & j & ") " & sNames(i * 100 + j)
    End If
  Next j
Next i

sbBirthdayList = vR

End Function
```

Akumuliertes Handelsblatt – sbAccumulatedTradeBlotter

Wenn Sie ein akumuliertes Handelsblatt (ein kumuliertes Handelsprotokoll) erzeugen wollen:

I3 =G3*F3

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			Input Data									Output Data			
		Bought/	Order-			Position Size/		Conversion	Traded	Traded Value		Create Output Data		Position Size/	Traded Value
2	Currency	Sold	ID	Trade Date	Value Date	Amount	Price	Rate	Value	in EUR		Currency	Long/Short	Amount	in EUR
3	EURGBP	Sold	1	14/01/2011	17/01/2011	26,100	0.89	1.17137265	23,317.14	27,313.07		EURGBP	Enter short	26100	27,313.07
4	EURCHF	Sold	2	14/01/2011	17/01/2011	26,100	1.46	0.72345243	38,160.77	27,607.50		EURCHF	Enter short	26100	27,607.50
5	EURGBP	Bought	3	14/01/2011	17/01/2011	26,100	0.89	1.17045492	23,309.96	27,283.25		EURGBP	Exit short - flat	0	0.00
6	EURGBP	Sold	4	14/01/2011	17/01/2011	26,100	0.89	1.17137265	23,317.14	27,313.07		EURGBP	Enter short	26100	27,313.07
7	EURCHF	Bought	5	14/01/2011	17/01/2011	26,100	1.46	0.71429161	38,155.45	27,254.12		EURCHF	Exit short - flat	0	0.00
8	EURGBP	Bought	6	14/01/2011	17/01/2011	26,100	0.89	1.16996567	23,311.29	27,273.41		EURGBP	Exit short - flat	0	0.00
9	EURGBP	Sold	7	14/01/2011	17/01/2011	26,100	0.89	1.17137265	23,317.14	27,313.07		EURGBP	Enter short	26100	27,313.07
10	EURCHF	Bought	8	14/01/2011	17/01/2011	26,100	1.46	0.72345243	38,151.46	27,600.76		EURCHF	Enter long	26100	27,600.76
11	EURCHF	Sold	9	14/01/2011	17/01/2011	26,100	1.46	0.71446362	38,160.77	27,264.48		EURCHF	Exit long - flat	0	0.00
12	EURGBP	Bought	10	14/01/2011	17/01/2011	26,100	0.89	1.16996567	23,311.29	27,273.41		EURGBP	Exit short - flat	0	0.00
13	EURGBP	Sold	11	14/01/2011	17/01/2011	26,100	0.89	1.17137265	23,317.14	27,313.07		EURGBP	Enter short	26100	27,313.07
14	EURCHF	Bought	12	14/01/2011	17/01/2011	26,100	1.46	0.72345243	38,154.12	27,602.69		EURCHF	Enter long	26100	27,602.69
15	EURGBP	Bought	13	14/01/2011	17/01/2011	26,100	0.89	1.1738292	23,312.62	27,365.03		EURGBP	Exit short - flat	0	0.00
16	EURGBP	Bought	14	14/01/2011	17/01/2011	26,100	0.89	1.17137265	23,311.29	27,306.21		EURGBP	Enter long	26100	27,306.21
17	EURCHF	Bought	15	14/01/2011	17/01/2011	26,100	1.46	0.72345243	38,148.79	27,598.84		EURCHF	Enter long	52200	55,201.53
18	EURGBP	Sold	16	14/01/2011	17/01/2011	26,100	0.89	1.1740558	23,315.81	27,374.07		EURGBP	Exit long - flat	0	0.00
19	EURCHF	Sold	17	14/01/2011	17/01/2011	26,100	1.46	0.71241701	38,154.12	27,181.64		EURCHF	Exit long	26100	28,019.88
20	EURCHF	Sold	18	14/01/2011	17/01/2011	26,100	1.46	0.73571458	38,154.12	28,070.54		EURCHF	Exit long - flat	0	0.00

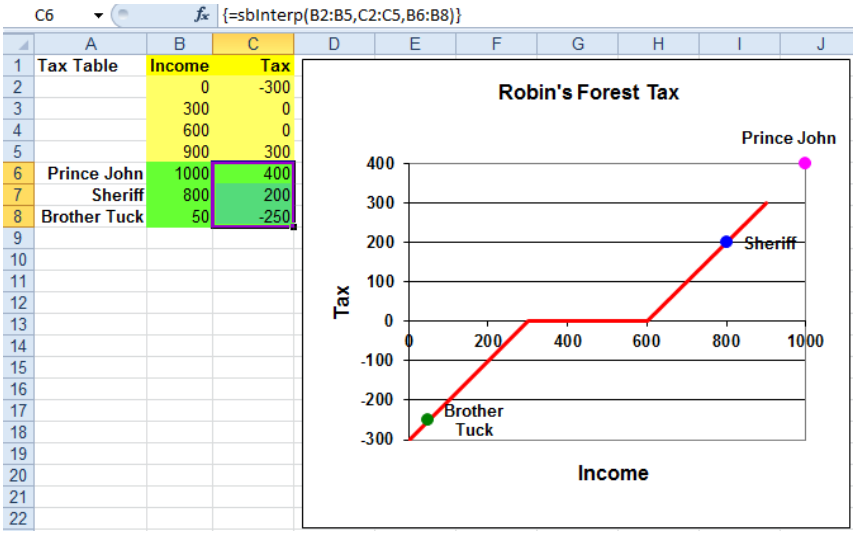
sbAccumulatedTradeBlotter Programmcode

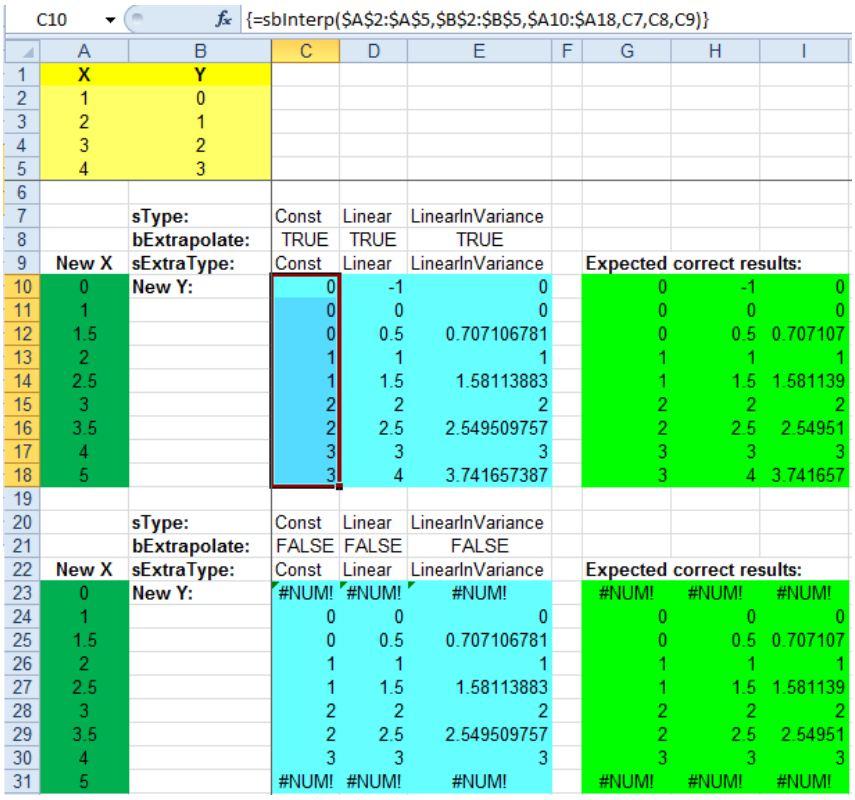
```
Enum trade_sheet_columns
  ts_ccy = 1
  ts_bought_sold
  ts_order_id
  ts_trade_date
  ts_value_date
  ts_amount
  ts_price
  ts_fxrate
  ts_traded_value
  ts_traded_value_EUR
  ts_EMPTY
  ts_output_ccy
  ts_output_long_short
  ts_output_amount_accumulated
  ts_output_traded_value_EUR
End Enum 'trade_sheet_columns

Private Enum xlCI 'Excel Color Index
: xlCIBlack = 1: xlCIWhite: xlCIRed: xlCIBrightGreen: xlCIBlue '1 - 5
: xlCIYellow: xlCIPink: xlCITurquoise: xlCIDarkRed: xlCIGreen '6 - 10
: xlCIDarkBlue: xlCIDarkYellow: xlCIViolet: xlCITeal: xlCIGray25 '11 - 15
: xlCIGray50: xlCIPeriwinkle: xlCIPlum: xlCIIvory: xlCILightTurquoise '16 - 20
: xlCIDarkPurple: xlCICoral: xlCIOceanBlue: xlCIIceBlue: xlCILightBrown '21 - 25
: xlCIMagenta2: xlCIYellow2: xlCICyan2: xlCIDarkPink: xlCIDarkBrown '26 - 30
: xlCIDarkTurquoise: xlCISeaBlue: xlCISkyBlue: xlCILightTurquoise2: xlCILightGreen '31 - 35
: xlCILightYellow: xlCIPaleBlue: xlCIRose: xlCILavender: xlCITan '36 - 40
: xlCILightBlue: xlCIAqua: xlCILime: xlCIGold: xlCILightOrange '41 - 45
: xlCIOrange: xlCIBlueGray: xlCIGray40: xlCIDarkTeal: xlCISeaGreen '46 - 50
: xlCIDarkGreen: xlCIGreenBrown: xlCIBrown: xlCIDarkPink2: xlCIIndigo '51 - 55
: xlCIGray80 '56
End Enum

Sub Calculate_Accumulated_Blotter()
'Source (EN): http://www.sulprobil.de/sbaccumulatedtradeblotter_en/
'Source (DE): http://www.berndplumhoff.de/sbaccumulatedtradeblotter_de/
'(C) (P) by Bernd Plumhoff 15-Jan-2011 PB V0.1
Dim lRow As Long
Dim lColorIdx As Long
Dim dSign As Double
Dim dSum As Double
Dim vColors As Variant
Dim oCcyPairAcc As Object    'Stores accumulated amounts of ccy pairs
Dim state As SystemState     'Runtime optimisation - see class SystemState

Set state = New SystemState 'Runtime optimisation - see class SystemState
Set oCcyPairAcc = CreateObject("Scripting.Dictionary")

vColors = Array(xlCIRed, xlCIBlue, xlCIBrown, xlCIDarkGreen, _
          xlCIDarkYellow, xlCIOrange, xlCIDarkTeal, xlCIPlum)
lColorIdx = 0
```

```
Sheets("CurrencyPairs").Select       'Sheet to work on
lRow = 3                     'Start row
Do While Not IsEmpty(Cells(lRow, ts_ccy))
  If lRow Mod 10 = 0 Then Application.StatusBar = _
     "Calculate_Accumulated_Blotter: Processing row " & lRow & " ..."
  Cells(lRow, ts_output_ccy) = Cells(lRow, ts_ccy)
  If oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text & "|FontColor") = 0# Then
    oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text & "|FontColor") = vColors(lColorIdx)
    lColorIdx = lColorIdx + 1
    If lColorIdx > UBound(vColors) Then
      Call MsgBox("Row " & lRow & ": Not enough colors defined", vbOKOnly, "Error")
      Exit Sub
    End If
  End If
  Select Case Cells(lRow, ts_bought_sold)
  Case "Bought"
    dSign = 1#
  Case "Sold"
    dSign = -1#
  Case Else
    Call MsgBox("Row " & lRow & ": Illegal Bought/Sold Keyword """ & _
      Cells(lRow, ts_bought_sold) & """ in column " & ts_bought_sold, vbOKOnly, "Error")
    Exit Sub             'Stop at first error
  End Select
  dSum = oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text) + dSign * Cells(lRow, ts_amount)
  Select Case Sgn(dSum)
  Case 1#
    If dSign = 1# Then
      Cells(lRow, ts_output_long_short) = "Enter long"
    Else
      Cells(lRow, ts_output_long_short) = "Exit long"
    End If
    oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text & "|EUR") = _
        oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text & "|EUR") + _
        dSign * Cells(lRow, ts_fxrate) * Cells(lRow, ts_traded_value)
  Case 0#
    If dSign = 1# Then
      Cells(lRow, ts_output_long_short) = "Exit short - flat"
    Else
      Cells(lRow, ts_output_long_short) = "Exit long - flat"
    End If
    oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text & "|EUR") = 0#
  Case -1#
    If dSign = 1# Then
      Cells(lRow, ts_output_long_short) = "Exit short"
    Else
      Cells(lRow, ts_output_long_short) = "Enter short"
    End If
    oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text & "|EUR") = _
        oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text & "|EUR") + _
        dSign * Cells(lRow, ts_fxrate) * Cells(lRow, ts_traded_value)
  End Select
  Cells(lRow, ts_output_amount_accumulated) = Abs(dSum)
  oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text) = dSum
  Cells(lRow, ts_output_traded_value_EUR) = Abs(oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text & "|EUR"))
  Range(Cells(lRow, ts_output_ccy), Cells(lRow, ts_output_traded_value_EUR)).Font.ColorIndex = _
    oCcyPairAcc.Item(Cells(lRow, ts_ccy).Text & "|FontColor")
  lRow = lRow + 1
Loop
Set oCcyPairAcc = Nothing
End Sub
```

Interpolieren – sbInterp

Falls Sie eine Menge von bekannten (x, y) Paaren haben und y-Werte zu anderen gegebenen x-Werten finden müssen, müssen Sie interpolieren. Dies entspricht dem Füllen von Lücken in einer Tabelle:





	A	B	C	D	E	F	G	H	I
1	X	Y							
2	1	0							
3	2	1							
4	3	2							
5	4	3							
6									
7		sType:	Const	Linear	LinearInVariance				
8		bExtrapolate:	TRUE	TRUE	TRUE				
9	New X	sExtraType:	Const	Linear	LinearInVariance		Expected correct results:		
10	0	New Y:	0	-1	0		0	-1	0
11	1		0	0	0		0	0	0
12	1.5		0	0.5	0.707106781		0	0.5	0.707107
13	2		1	1	1		1	1	1
14	2.5		1	1.5	1.58113883		1	1.5	1.581139
15	3		2	2	2		2	2	2
16	3.5		2	2.5	2.549509757		2	2.5	2.54951
17	4		3	3	3		3	3	3
18	5		3	4	3.741657387		3	4	3.741657
19									
20		sType:	Const	Linear	LinearInVariance				
21		bExtrapolate:	FALSE	FALSE	FALSE				
22	New X	sExtraType:	Const	Linear	LinearInVariance		Expected correct results:		
23	0	New Y:	#NUM!	#NUM!	#NUM!		#NUM!	#NUM!	#NUM!
24	1		0	0	0		0	0	0
25	1.5		0	0.5	0.707106781		0	0.5	0.707107
26	2		1	1	1		1	1	1
27	2.5		1	1.5	1.58113883		1	1.5	1.581139
28	3		2	2	2		2	2	2
29	3.5		2	2.5	2.549509757		2	2.5	2.54951
30	4		3	3	3		3	3	3
31	5		#NUM!	#NUM!	#NUM!		#NUM!	#NUM!	#NUM!

sbInterp Programmcode

```
Function sbInterp(vX As Variant, vY As Variant, _
           vT As Variant, _
           Optional ByVal sType As String = "Linear", _
           Optional bExtrapolate As Boolean = True, _
           Optional ByVal sExtraType As String) As Variant
'Interpolates y-values for target values vT with known
'y-values vY and known x-values vX with type sType.
'sType can be:
'Const or C
'Linear or L
'LinearInVariance or LIV
'Extrapolation will be done if bExtrapolate is TRUE.
'Extrapolation type sExtraType defaults to sType if empty.
'Values in vX must be in ascending order. #VALUE! error
'indicates illegal sType, #NUM! error indicates that
'extrapolation has been switched off and #N/A tells you
'that x-values are not given in increasing order, or
'y-value count differs from x-value count.
'(C) (P) by Bernd Plumhoff 25-Dec-2023 PB V0.7
Dim i As Long, iX As Long, iY As Long, iT As Long, k As Long
Dim vTk, vXi
Dim sT As String 'Type of inter- or extrapolation
Dim sEType As String 'Extrapolation type
With Application
On Error Resume Next
iX = vX.Count
iX = UBound(vX)
iY = vY.Count
iY = UBound(vY)
iT = vT.Count
iT = UBound(vT)
On Error GoTo 0
If iX <> iY Then
  sbInterp = CVErr(xlErrNA)
  Exit Function
End If
k = 0
ReDim vX1(1 To iX) As Variant, vY1(1 To iX) As Variant
For i = 1 To iX
  If vX(i) <> "" And vY(i) <> "" Then
    k = k + 1
    vX1(k) = vX(i)
    vY1(k) = vY(i)
  End If
Next i
iX = k
ReDim Preserve vX1(1 To iX) As Variant
ReDim Preserve vY1(1 To iX) As Variant
If iX < 2 Then
  sType = "Const"
  sExtraType = "Const"
Else
  For k = 2 To iX
    If vX1(k) <= vX1(k - 1) Then
      sbInterp = CVErr(xlErrNA)
      Exit Function
    End If
  Next k
End If
ReDim vR(1 To iT) As Variant
If sExtraType = "" Then
  sEType = sType 'Same as interpolation type
Else
  sEType = sExtraType
End If
For k = 1 To iT
  i = 0
  vTk = 0
  vXi = 0
  On Error Resume Next
  i = .Match(vT(k), vX1, 1)
  vTk = vT(k)
  vXi = vX1(i)
  On Error GoTo 0
  If Not bExtrapolate And _
    (i = 0 Or (i = iX And vTk <> vXi)) Then
    vR(k) = CVErr(xlErrNum)
  Else
    sT = sType 'Set to interpolation type
    If i = 0 Then
      i = 1
      sT = sEType 'Set to extrapolation type
    End If
    If i = iX Then
      i = i - 1
      If vTk <> vXi Then
```

```
        sT = sEType 'Set to extrapolation type
      End If
      If sT = "C" Or sT = "Const" Then i = i + 1
    End If
    Select Case sT
    Case "C", "Const"
      vR(k) = .Index(vY1, i)
    Case "L", "Linear"
      vR(k) = .Index(vY1, i) + (vTk - .Index(vX1, i)) _
              * (.Index(vY1, i + 1) - .Index(vY1, i)) _
              / (.Index(vX1, i + 1) - .Index(vX1, i))
    Case "LIV", "LinearInVariance"
      On Error Resume Next
      vR(k) = Sqr(.Index(vY1, i) ^ 2# + (vTk - .Index(vX1, i)) _
              * (.Index(vY1, i + 1) ^ 2# - .Index(vY1, i) ^ 2#) _
              / (.Index(vX1, i + 1) - .Index(vX1, i)))
      On Error GoTo 0
    Case Else
      sbInterp = CVErr(xlErrValue)
      Exit Function
    End Select
  End If
Next k
If TypeName(vT) = "Range" Then
  If vT.Rows.Count > vT.Columns.Count Then
    vR = .Transpose(vR)
  End If
ElseIf TypeName(.Caller) = "Range" Then
  If .Caller.Rows.Count > .Caller.Columns.Count Then
    vR = .Transpose(vR)
  End If
End If
sbInterp = vR
End With
End Function
```

Erzeuge alle Kombinationen der Subsets k von n

Dieser Algorithmus erzeugt alle Kombinationen von k Elementen aus einer Gesamtmenge mit n Elementen.

Während die Excel Funktion KOMBINATIONEN(n; k) die Anzahl dieser Kombinationen anzeigt, listet der hier vorgestellte Algorithmus alle einzeln, damit sie ggf. untersucht werden können. Das vorgestellte Programm ist schnell – zum Vergleich siehe unten einen interessanten Link für weitere Formelansätze.

<u>Beispiel</u>

Eingabe:

	A	B
1	n	5
2	k	3

Ausgabe:

	A	B	C	D	E
1	1	1	1	0	0
2	1	0	1	1	0
3	0	1	1	1	0
4	1	1	0	1	0
5	1	0	0	1	1
6	0	1	0	1	1
7	0	0	1	1	1
8	1	0	1	0	1
9	0	1	1	0	1
10	1	1	0	0	1

Literatur

Reingold, Nievergelt, Deo: Combinatorial Algorithms, 1977, Algorithm 5.9, p. 186, ISBN 0-13-152447-X

Ein interessanter Link (engl.):
https://techcommunity.microsoft.com/discussions/excelgeneral/efficient-approach-to-generate-list-of-combinations-with-no-repetition/3786730

Combinations_with_k_subsets_of_n Programmcode

```
Public r As Long 'Output row

'Generates all combinations of array n with subsets k.
'See Reingold, Nievergelt, Deo: Combinatorial Algorithms, 1977, Algorithm 5.9, p. 186, ISBN 0-13-152447-X
'Version 0.2 12-Jul-2024

Sub Combinations_with_k_subsets_of_n(n As Long, k As Long)
Dim i As Long, j As Long, m As Long, t As Long, u As Long

With Application.WorksheetFunction
wsO.Cells.ClearContents
ReDim g(1 To n + 1) As Long
ReDim tau(1 To n + 1) As Long
ReDim res(1 To .Combin(n, k), 1 To n) As Variant
r = 1
For j = 1 To k
  g(j) = 1
  tau(j) = j + 1
Next j
For j = k + 1 To n + 1
  g(j) = 0
  tau(j) = j + 1
Next j
t = k
tau(1) = k + 1
i = 0
Do While i <> n + 1
  'Call Visit(g) instead of the next 4 rows if you need to analyze g().
  For u = 1 To UBound(g) - 1
    res(r, u) = g(u)
  Next u
  r = r + 1
  i = tau(1)
  tau(1) = tau(i)
  tau(i) = i + 1
  If g(i) = 1 Then
    If t <> 0 Then
      g(t) = 1 - g(t)
    Else
      g(i - 1) = 1 - g(i - 1)
    End If
    t = t + 1
  Else
    If t <> 1 Then
      g(t - 1) = 1 - g(t - 1)
```

```
    Else
      g(i - 1) = 1 - g(i - 1)
    End If
    t = t - 1
  End If
  g(i) = 1 - g(i)
  If t = i - 1 Or t = 0 Then
    t = t + 1
  Else
    t = t - g(i - 1)
    tau(i - 1) = tau(1)
    If t = 0 Then
      tau(1) = i - 1
    Else
      tau(1) = t + 1
    End If
  End If
Loop
Range(wsO.Cells(1, 1), wsO.Cells(r - 1, n)) = res
End With
End Sub

Sub Visit(g As Variant)
'Print current permutation in immediate window and on sheet Output.
'You can analyze the permutation or do other things as well.
Dim i As Long
For i = 1 To UBound(g) - 1
  wsO.Cells(r, i) = g(i)
  Debug.Print g(i);
Next i
Debug.Print
r = r + 1
End Sub

Sub test()
wsI.[H1] = Now
Call Combinations_with_k_subsets_of_n(wsI.[B1], wsI.[B2])
wsI.[H2] = Now
End Sub
```

Lookup Varianten

“The hardest thing of all is to find a black cat in a dark room, especially if there is no cat.” [Confucius]
Abstract

Hier stelle ich einige VERWEIS (LOOKUP) Varianten vor, die ich hilfreich finde:

Lookup Programmcodes

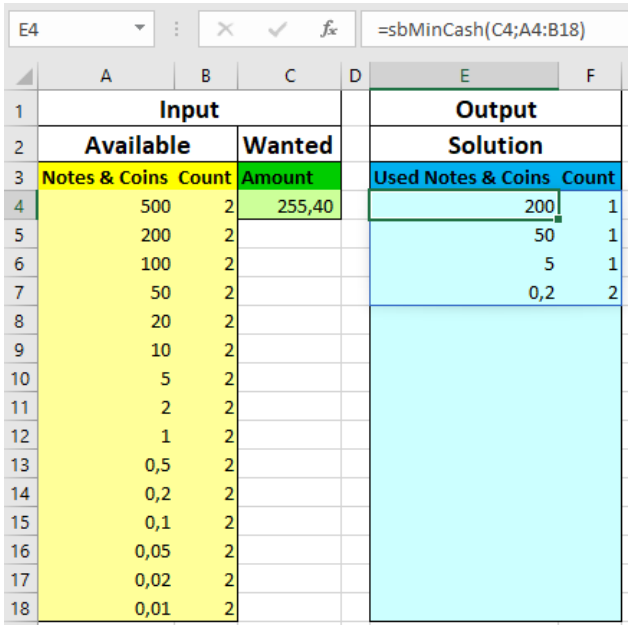
```
Function sbLookup(vLookupValue As Variant, _
            rTableArray As Range, _
            Optional ByVal lOccurrence As Long = 1, _
            Optional lColumnOffset As Long, _
            Optional lRowOffset As Long) As Variant
'Reverse("moc.LiborPlus.www") PB 09-May-2010 V0.10
'Looks up lOccurrence'th occurrence of vLookupValue in rTableArray
'and returns found cell offset by lRowOffset rows and lColumnOffset
'columns. If lOccurrence is negative the search is done bottom-up
'(i.e. -1 finds the last value, -2 last but one, etc.).
'This function was inspired by the "Ultimate" Excel Lookup Function OzgridLookup:
'http://www.ozgrid.com/VBA/ultimate-excel-lookup-function.htm

Dim i As Long
Dim rFound As Range
Dim iSearchDir As Integer

If lOccurrence >= 0 Then
    iSearchDir = xlNext
Else
    iSearchDir = xlPrevious
    lOccurrence = -lOccurrence
End If

With rTableArray
    If rTableArray.Cells(1, 1) = vLookupValue And lOccurrence = 1 Then
        sbLookup = .Cells(1, 1)(1, lColumnOffset + 1)
        Exit Function
```

```
    Else
        Set rFound = .Cells(1, 1)
        For i = 1 To lOccurrence
            Set rFound = rTableArray.Find(What:=vLookupValue, After:=rFound, _
                    LookIn:=xlValues, LookAt:=xlWhole, SearchOrder:=xlRows, _
                    SearchDirection:=iSearchDir)
        Next i
    End If
End With

sbLookup = rFound.Offset(lRowOffset, lColumnOffset)

End Function


Function sbClosest(dSearchVal As Double, _
    rLookupRange As Range, _
    Optional dLower As Double = 0#, _
    Optional dUpper As Double = 0#) As Variant
'Looks for the closest value to dSearchVal in
'rLookupRange which is greater or equal to dSearchVal
'+ dLower and less or equal to dSearchVal + dUpper.
'Returns that value and the address of it. xlErrNum
'indicates that no relevant data was found.
'Reverse("moc.LiborPlus.www") V0.10 16-Oct-2010 PB
Dim dMin As Double, v, vR(1 To 2)
dMin = 1E+308
For Each v In rLookupRange
    If (dLower = 0# And dUpper = 0#) Or _
        (v >= dSearchVal + dLower And _
        v <= dSearchVal + dUpper) Then
            If Abs(v - dSearchVal) < dMin Then
                vR(1) = v
                vR(2) = v.Address(False, False)
                dMin = Abs(v - dSearchVal)
            End If
    End If
Next v
If dMin = 1E+308 Then
    sbClosest = CVErr(xlErrNum)
Else
    sbClosest = vR
End If
End Function


Function sbLookupAddress(vLookupValue As Variant, _
            rTableArray As Range, _
            Optional ByVal lOccurrence As Long = 1, _
            Optional lColumnOffset As Long, _
            Optional lRowOffset As Long) As String
'Reverse("moc.LiborPlus.www") PB 26-Aug-2010 V0.10
'Looks up lOccurrence'th occurrence of vLookupValue in rTableArray and
'returns address of found cell offset by lRowOffset rows and lColumnOffset
'columns. If lOccurrence is negative the search is done bottom-up
'(i.e. -1 finds the last value, -2 last but one, etc.).

Dim i As Long
Dim rFound As Range, rLast As Range
Dim iSearchDir As Integer

If lOccurrence >= 0 Then
    iSearchDir = xlNext
Else
    iSearchDir = xlPrevious
    lOccurrence = -lOccurrence + 1
End If

With rTableArray
    If rTableArray.Cells(1, 1) = vLookupValue Then lOccurrence = lOccurrence - 1
    If lOccurrence = 0 Then
        sbLookupAddress = .Cells(1, 1)(1, lColumnOffset + 1).Address(False, False)
        Exit Function
    Else
        Set rFound = .Cells(1, 1)
        Set rFound = rTableArray.Find(What:=vLookupValue, After:=rFound, _
                LookIn:=xlValues, LookAt:=xlWhole, SearchOrder:=xlRows, _
                SearchDirection:=iSearchDir)
        Set rLast = rFound
        Do
            lOccurrence = lOccurrence - 1
            If lOccurrence = 0 Then
                sbLookupAddress = rFound.Offset(lRowOffset, _
                                  lColumnOffset).Address(False, False)
                Exit Function
            End If
            Set rFound = rTableArray.Find(What:=vLookupValue, After:=rFound, _
                    LookIn:=xlValues, LookAt:=xlWhole, SearchOrder:=xlRows, _
                    SearchDirection:=iSearchDir)
        Loop While rLast.Address <> rFound.Address
```

```
        sbLookupAddress = CVErr(xlErrValue)
    End If
End With

End Function


Function vlookupall(sSearch As String, rRange As Range, _
    Optional lLookupCol As Long = 2, Optional sDel As String = ",") As String
'Vlookupall searches in first column of rRange for sSearch and returns
'corresponding values of column lLookupCol if sSearch was found. All these
'lookup values are being concatenated, delimited by sDel and returned in
'one string. If lLookupCol is negative then rRange must not have more than
'one column.
'Reverse("moc.LiborPlus.www") PB 16-Sep-2010 V0.20
Dim i As Long, sTemp As String
If lLookupCol > rRange.Columns.Count Or sSearch = "" Or _
    (lLookupCol < 0 And rRange.Columns.Count > 1) Then
    vlookupall = CVErr(xlErrValue)
    Exit Function
End If
vlookupall = ""
For i = 1 To rRange.Rows.Count
    If rRange(i, 1).Text = sSearch Then
        If lLookupCol >= 0 Then
            vlookupall = vlookupall & sTemp & rRange(i, lLookupCol).Text
        Else
            vlookupall = vlookupall & sTemp & rRange(i).Offset(0, lLookupCol).Text
        End If
        sTemp = sDel
    End If
Next i
End Function


Function vlookupallarr(sSearch As String, rRange As Range, _
    Optional lLookupCol As Long = 2) As Variant
'Vlookupall searches in first column of rRange for sSearch and returns
'corresponding values of column lLookupCol if sSearch was found. All
'values looked up are being returned in a vertical array.
'If lLookupCol is negative then rRange must not have more than
'one column.
'Reverse("moc.LiborPlus.www") PB 12-Jul-2012 V0.10
Dim i As Long, j As Long
If lLookupCol > rRange.Columns.Count Or sSearch = "" Or _
    (lLookupCol < 0 And rRange.Columns.Count > 1) Then
    vlookupallarr = CVErr(xlErrValue)
    Exit Function
End If
ReDim v(1 To rRange.Rows.Count)
For i = 1 To rRange.Rows.Count
    If rRange(i, 1).Text = sSearch Then
        j = j + 1
        If lLookupCol >= 0 Then
            v(j) = rRange(i, lLookupCol).Text
        Else
            v(j) = rRange(i).Offset(0, lLookupCol).Text
        End If
    End If
Next i
i = Application.Caller.Rows.Count
ReDim Preserve v(1 To i)
For j = j + 1 To i
    v(j) = ""
Next j
vlookupallarr = Application.WorksheetFunction.Transpose(v)
End Function


Function lookup2(vSV As Variant, vSA As Variant, vRA As Variant) As Variant
'Similar to lookup() but it looks up the biggest value in vSA which is less-equal than vSV
'vSA has to be sorted, lowest first!!
'Remember that lookup() looks up the smallest value in the search-array which is
'greater-equal than search-value.
Dim i As Long
i = 1
Do While i <= vSA.Count
    If vSV <= vSA(i) Then
        lookup2 = vRA(i)
        Exit Function
    End If
    i = i + 1
Loop
lookup2 = "OUT OF RANGE"
End Function
```

Minimale Anzahl von Scheinen und Münzen für einen Geldbetrag – sbMinCash

Wenn Sie die kleinste Menge an Geldscheinen und Münzen ermitteln wollen, um auf einen Betrag zu kommen, hilft dieses Programm:



	A	B	C	D	E	F
1	Input				Output	
2	Available		Wanted		Solution	
3	Notes & Coins	Count	Amount		Used Notes & Coins	Count
4	500	2	255,40		200	1
5	200	2			50	1
6	100	2			5	1
7	50	2			0,2	2
8	20	2				
9	10	2				
10	5	2				
11	2	2				
12	1	2				
13	0,5	2				
14	0,2	2				
15	0,1	2				
16	0,05	2				
17	0,02	2				
18	0,01	2				

Mögliche Fehlerwerte sind:

#ZAHL! - Der Eingangsbetrag oder der Wert eines Scheins oder Münze hat mehr als 2 Nachkommastellen
#WERT! - Der Wert eines Scheins oder einer Münze ist negativ oder ungültig
#NV - Es existiert keine Lösung

Eine bekannte Ausnahme: Falls die Werte der Geldscheine oder Münzen ungünstig sind, dann ist die ausgegebene Stückelung nicht minimal. Beispiel: Mit den Münzwerten 1, 6, und 10 wird der Eingabebetrag 13 mit der Stückelung 10, 1, 1, 1 und nicht 6, 6, 1 ausgegeben. Die gute Nachricht ist jedoch: Dies tritt nicht im normalen Fall mit Werten wie 1000, 500, 200, 100, 50, 20, 10, 5, 2, 1, 0.50, 0.20, 0.10, 0.05, 0.02, und 0.01 auf.

Aber falls Sie prüfen müssen ob sbMinCash die minimale Anzahl von Münzen und Scheinen ermittelt hat - Sie müssen alle Eingabewerte mit 100 multiplizieren falls Sie Nachkommastellen haben, da dieser Algorithmus ganzzahlig ist, und er berücksichtigt keine Begrenzungen bei den Scheinen und Münzen:

sbMinCoins Programmcode

```
Function minCoins(V As Long, coins As Variant) As Variant
'Source: https://www.geeksforgeeks.org/find-minimum-number-of-coins-that-make-a-change/
'Adapted to VBA by Bernd Plumhoff  15-Jan-2023 PB V0.1
Dim i As Integer, j As Integer, sub_res As Integer
Dim vCoins As Variant
With Application.WorksheetFunction
If V <> Int(V) Then
    minCoins = CVErr(xlErrNum)
    Exit Function
End If
vCoins = .Transpose(.Transpose(coins))
ReDim tabl(0 To V + 1) As Long
tabl(V + 1) = UBound(vCoins)
tabl(0) = 0
For i = 1 To V
  tabl(i) = 65536 'Do not use type Long. 2147483647 you cannot wait for.
Next i
For i = 1 To V
  For j = 1 To UBound(vCoins, 1)
    If vCoins(j, 1) <= i Then
      sub_res = tabl(i - vCoins(j, 1))
      If sub_res <> 65536 And _
        sub_res + 1 < tabl(i) Then
        tabl(i) = sub_res + 1
      End If
    End If
  Next j
Next i

If tabl(V) = 65536 Then
  minCoins = CVErr(xlErrNA)
Else
  minCoins = tabl(V)
End If
End With
End Function
```

sbMinCash Programmcode

```
Function sbMinCash(dAmount As Variant, vNotesCoins As Variant) As Variant
'Returns the minimum number of given notes and coins to make up dAmount
'in a two-dimensional vertical array, for example for dAmount = 255.40 and
'vNotesCoins = {500.2;200.2;100.2;50.2;20.2;10.2;5.2;2.2;1.2;0,5.2;0,2.2;0,1.2;0,05.2;0,02.2;0,01.2}
'the result would be:
'200   1
' 50   1
'  5   1
'  0.2 2
'If you omit the second dimension of vNotesCoins - that is, if you do not provide
```

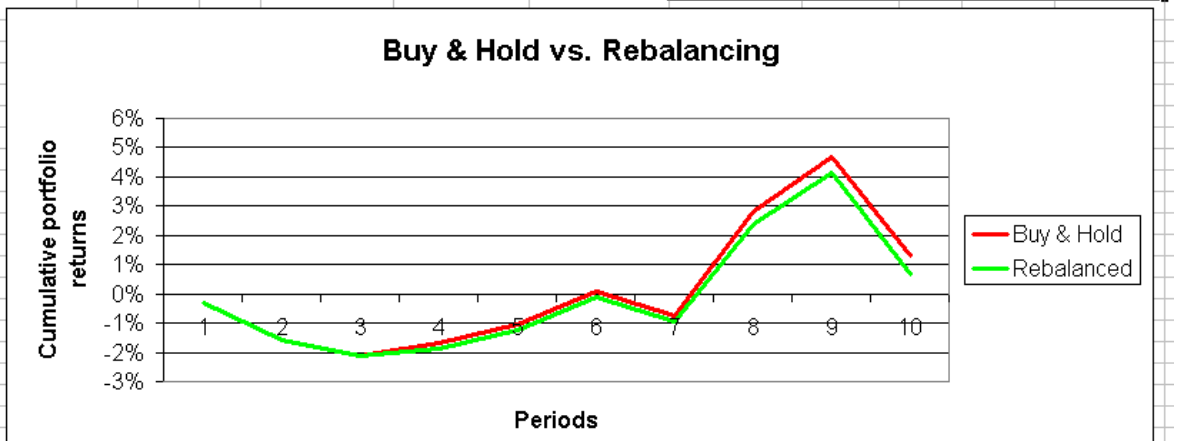
```
'the number of available banknotes and coins (just their face value) then the
'program would assume unlimited supply.
'Return error values:
'xlErrNum -   dAmount or a face values in vNotesCoins have more than 2 decimal places
'xlErrValue - A face value in vNotesCoins is negative
'xlErrNull -  No value in vNotesCoins given
'xlErrNA -    There is no solution
'Known limitation: In some less fortunate cases such as banknotes with
'amounts 1, 6, and 10 the cash amount 13 will result in 10, 1, 1, 1,
'and not in 6, 6, 1.
'(C) (P) by Bernd Plumhoff  15-Jan-2023 PB V0.5
Dim lAmount100 As Long 'dAmount x 100 to be able to apply integer calc
Dim i As Long, j As Long, k As Long

With Application.WorksheetFunction
If dAmount * 100# <> Int(dAmount * 100#) Then
    sbMinCash = CVErr(xlErrNum)
    Exit Function
End If
vNotesCoins = .Transpose(.Transpose(vNotesCoins))
ReDim lNC100(1 To UBound(vNotesCoins, 1)) As Long

'Fill integer array with 100 x non-empty notes and coins set
i = 1
j = 1
Do While i <= UBound(vNotesCoins, 1)
  If vNotesCoins(i, 1) >= 0 Then
    lNC100(j) = Int(vNotesCoins(i, 1) * 100# + 0.5)
    If lNC100(j) / 100# <> vNotesCoins(i, 1) Then
      sbMinCash = CVErr(xlErrNum)
      Exit Function
    End If
    j = j + 1
  Else
    sbMinCash = CVErr(xlErrValue)
    Exit Function
  End If
  i = i + 1
Loop
If j = 1 Then
  sbMinCash = CVErr(xlErrNull)
  Exit Function
End If

ReDim Preserve lNC100(1 To j - 1) As Long
ReDim vR(0 To 1, 1 To j - 1) As Variant
'Sort notes and coins, highest value first
For i = 1 To UBound(lNC100, 1) - 1
  For j = i + 1 To UBound(lNC100, 1)
    If lNC100(i) < lNC100(j) Then
      k = lNC100(i)
      lNC100(i) = lNC100(j)
      lNC100(j) = k
    End If
  Next j
Next i

lAmount100 = Int(dAmount * 100# + 0.5)
j = 1
i = 1
Do While lAmount100 > 0 And lNC100(i) > 0
  k = lAmount100 \ lNC100(i)
  If UBound(vNotesCoins, 2) > 1 Then
    If k > vNotesCoins(i, 2) Then
      k = vNotesCoins(i, 2)
    End If
  End If
  If k > 0 Then
    vR(0, j) = lNC100(i) / 100#
    vR(1, j) = k
    j = j + 1
    lAmount100 = lAmount100 - k * lNC100(i)
  End If
  i = i + 1
  If i > UBound(lNC100, 1) Then Exit Do
Loop
If lAmount100 <> 0 Then
  sbMinCash = CVErr(xlErrNA)
Else
  ReDim Preserve vR(0 To 1, 1 To j - 1)
  sbMinCash = .Transpose(vR)
End If
End With
End Function
```

Neugewichtung der Assets eines Portfolios – sbRebalancedReturn

Die Neugewichtung eines Portfolios ist der Prozess, alle Assetklassen eines Portfolios hinsichtlich ihrer Gewichtung wieder wie gewünscht auszurichten. Sie bringen Ihr Portfolio wieder zur gewünschten Mischung von Aktien, Bonds, Cash und anderen Produkten zurück, wenn sie nicht mehr mit Ihren Plänen oder Limiten übereinstimmen.

H3 {=RebalancedReturn(B3:D12,B1:D1,J1,L1)}

	A	B	C	D	E	F	G	H	I	J	K	L
1	Weights:	33.33%	33.33%	33.33%	Buy & Hold	Cum Buy & Hold	Cum rebalanced	Rebalanced	Frequency:	4	Tolerance:	3%
2	Period	A1	A2	A3	Portfolio return	Portfolio return	Portfolio return	Portfolio return	A1	A2	A3	Rebalanced?
3	1	-0.99%	-3.00%	3.00%	-0.33%	-0.33%	-0.33%	-0.33%	33.11%	32.44%	34.45%	TRUE
4	2	2.60%	-4.00%	-2.29%	-1.23%	-1.56%	-1.56%	-1.23%	34.39%	31.53%	34.08%	FALSE
5	3	1.45%	-5.00%	1.47%	-0.58%	-2.14%	-2.14%	-0.58%	35.10%	30.13%	34.78%	FALSE
6	4	5.00%	-3.00%	-1.16%	0.45%	-1.69%	-1.86%	0.28%	34.90%	32.24%	32.85%	TRUE
7	5	1.16%	0.06%	0.64%	0.66%	-1.03%	-1.24%	0.62%	33.51%	33.15%	33.34%	TRUE
8	6	5.00%	2.52%	-4.21%	1.13%	0.10%	-0.13%	1.11%	34.80%	33.61%	31.59%	FALSE
9	7	1.22%	0.22%	-4.21%	-0.83%	-0.73%	-0.96%	-0.83%	35.52%	33.97%	30.51%	FALSE
10	8	6.00%	1.16%	2.75%	3.55%	2.82%	2.40%	3.36%	36.43%	33.24%	30.33%	FALSE
11	9	3.00%	0.75%	1.42%	1.86%	4.68%	4.13%	1.72%	33.75%	33.01%	33.23%	TRUE
12	10	-2.70%	-4.51%	-3.21%	-3.38%	1.30%	0.66%	-3.47%	34.02%	32.66%	33.32%	FALSE



sbRebalancedReturn Programmcode

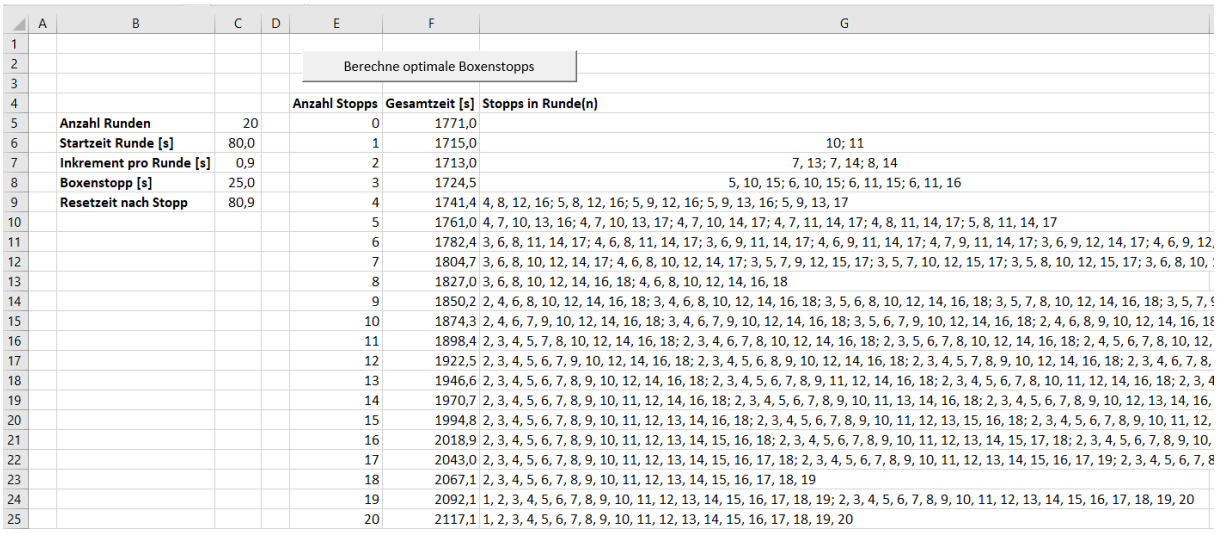
```
Const CMaxDouble = 1.79769313486231E+308
Function sbRebalancedReturn(rARM As Range, _
    rIWV As Range, _
    Optional ByVal lRF As Long = 0, _
    Optional dDT As Double = CMaxDouble) As Variant
'RebalancedReturn calculates balanced returns for a
'portfolio with given
'rARM - asset return matrix (columns show different
'       assets, rows show returns per asset over time)
'rIWV - initial weight vector for the assets
'lRF  - rebalancing frequency (in time steps = rows)
'       If lRF > 0 then each lRF time step rebalancing
'       will take place
'       If lRF = 0 then no rebalancing will take place
'       If lRF < 0 then each -lRF time step after last
'       rebalance portfolio will be rebalanced again
'dDT  - drift tolerance %, if any asset has drifted by
'       by more than dDT (relative measure) then the
'       portfolio will be rebalanced AND the internal
'       rebalancing frequency count will be reset
'The output matrix shows portfolio returns % in first
'column, then end-of-period asset weights and finally
'boolean output values in last column, showing whether
'a rebalance happened.
'This function has been inspired by Andreas Steiner's
'similar function.
'(C) (P) by Bernd Plumhoff 19-Mar-2011 PB V0.2
Dim i As Long, j As Long, k As Long, n As Long, m As Long
Dim bDrifted As Boolean, bForceRB As Boolean

n = rARM.Rows.Count 'Number of observations
m = rARM.Columns.Count 'Number of assets

If m <> rIWV.Columns.Count Or _
```

```
  rIWV.Rows.Count <> 1 Then
  sbRebalancedReturn = CVErr(xlErrValue)
  Exit Function
End If

ReDim w0(1 To m) As Double, x(1 To m) As Double
ReDim r(1 To n, 1 To m) As Double

If lRF = 0 Then lRF = n
If lRF < 0 Then
  lRF = -lRF
  bForceRB = True
Else
  bForceRB = False
End If

ReDim vR(1 To n, 1 To m + 2)
For i = 1 To m
  x(i) = rIWV(i)
  w0(i) = x(i)
  For j = 1 To n
    r(j, i) = rARM(j, i)
  Next j
Next i

k = 1
'Model rebalancing tolerance
For i = 1 To n
  If bDrifted And bForceRB Then k = i
  'Calculate period start weights
  vR(i, m + 2) = (i - k) Mod lRF = 0 Or bDrifted
  If vR(i, m + 2) Then
    For j = 1 To m
      x(j) = w0(j)
    Next j
  Else
    For j = 1 To m
      x(j) = vR(i - 1, 1 + j)
    Next j
  End If
  'Calculate portfolio return
  For j = 1 To m
    vR(i, 1) = vR(i, 1) + x(j) * r(i, j)
  Next j
  'Calculate period end weights & check for drift
  bDrifted = False
  For j = 1 To m
    vR(i, 1 + j) = x(j) * (1# + r(i, j)) / (1# + vR(i, 1))
    bDrifted = bDrifted Or Abs(vR(i, 1 + j) - w0(j)) > dDT
  Next j
Next i
sbRebalancedReturn = vR
End Function
```

Optimale Boxenstopps

Sie managen ein Autorennteam und wollen die optimalen Boxenstopps für ein Rennen planen?

Beispiel:



Berechne optimale Boxenstopps

				Anzahl Stopps	Gesamtzeit [s]	Stopps in Runde(n)
Anzahl Runden	20		0	1771,0		
Startzeit Runde [s]	80,0		1	1715,0	10; 11	
Inkrement pro Runde [s]	0,9		2	1713,0	7, 13; 7, 14; 8, 14	
Boxenstopp [s]	25,0		3	1724,5	5, 10, 15; 6, 10, 15; 6, 11, 15; 6, 11, 16	
Resetzeit nach Stopp	80,9		4	1741,4	4, 8, 12, 16; 5, 8, 12, 16; 5, 9, 12, 16; 5, 9, 13, 16; 5, 9, 13, 17	
			5	1761,0	4, 7, 10, 13, 16; 4, 7, 10, 13, 17; 4, 7, 10, 14, 17; 4, 7, 11, 14, 17; 4, 8, 11, 14, 17; 5, 8, 11, 14, 17	
			6	1782,4	3, 6, 8, 11, 14, 17; 4, 6, 8, 11, 14, 17; 3, 6, 9, 11, 14, 17; 4, 6, 9, 11, 14, 17; 4, 7, 9, 11, 14, 17; 3, 6, 9, 12, 14, 17; 4, 6, 9, 12,	
			7	1804,7	3, 6, 8, 10, 12, 14, 17; 4, 6, 8, 10, 12, 14, 17; 3, 5, 7, 9, 12, 15, 17; 3, 5, 7, 10, 12, 15, 17; 3, 5, 8, 10, 12, 15, 17; 3, 6, 8, 10,	
			8	1827,0	3, 6, 8, 10, 12, 14, 16, 18; 4, 6, 8, 10, 12, 14, 16, 18	
			9	1850,2	2, 4, 6, 8, 10, 12, 14, 16, 18; 3, 4, 6, 8, 10, 12, 14, 16, 18; 3, 5, 6, 8, 10, 12, 14, 16, 18; 3, 5, 7, 8, 10, 12, 14, 16, 18; 3, 5, 7,	
			10	1874,3	2, 4, 6, 7, 9, 10, 12, 14, 16, 18; 3, 4, 6, 7, 9, 10, 12, 14, 16, 18; 3, 5, 6, 7, 9, 10, 12, 14, 16, 18; 2, 4, 6, 8, 9, 10, 12, 14, 16, 18	
			11	1898,4	2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 18; 2, 3, 4, 6, 7, 8, 10, 12, 14, 16, 18; 2, 3, 5, 6, 7, 8, 10, 12, 14, 16, 18; 2, 4, 5, 6, 7, 8, 10, 12,	
			12	1922,5	2, 3, 4, 5, 6, 7, 9, 10, 12, 14, 16, 18; 2, 3, 4, 5, 6, 8, 9, 10, 12, 14, 16, 18; 2, 3, 4, 5, 7, 8, 9, 10, 12, 14, 16, 18; 2, 3, 4, 6, 7, 8,	
			13	1946,6	2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18; 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 14, 16, 18; 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 16, 18; 2, 3, 4	
			14	1970,7	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16, 18; 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 18; 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16,	
			15	1994,8	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 18; 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 18; 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,	
			16	2018,9	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18; 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18; 2, 3, 4, 5, 6, 7, 8, 9, 10,	
			17	2043,0	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18; 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19; 2, 3, 4, 5, 6, 7, 8	
			18	2067,1	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19	
			19	2092,1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19; 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	
			20	2117,1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20	

Optimale_Boxenstopps Programmcode

```
Sub optimale_boxenstopps()
'Berechnet optimale Boxenstopps für ein Autorennteam.
'(C) (P) by Bernd Plumhoff 01-Jan-2023 PB V0.2

Dim i                           As Long
Dim j                           As Long
Dim k                           As Long
Dim m                           As Long
Dim t                           As Long
Dim lRunden                     As Long

Dim dStartzeit                  As Double
Dim dRundenZeit                 As Double
Dim dResetZeit                  As Double
Dim dZeitTotal                  As Double
Dim dZeitBest                   As Double
Dim dInkrement                  As Double
Dim dBoxenstopp                 As Double

Dim sBoxenstopps                As String
Dim sComma                      As String
Dim sSemiColon                  As String

Dim state                       As SystemState 'Siehe https://www.berndplumhoff.de/systemstate_de/

Set state = New SystemState

lRunden = Range("Anzahl_Runden")
ReDim lIdx(1 To lRunden) As Long
dStartzeit = Range("Startzeit")
dResetZeit = Range("Resetzeit")
dInkrement = Range("Inkrement")
dBoxenstopp = Range("Boxenstopp")
Columns("E:G").ClearContents
Range("E4:G4").FormulaArray = Array("Anzahl Stopps", "Gesamtzeit [s]", "Stopps in Runde(n)")
```

```
For t = 0 To lRunden 'Anzahl der Boxenstopps
  dZeitBest = 1E+300
  ReDim c(1 To t + 2) As Long
  For j = 1 To t
    c(j) = j - 1
  Next j
  c(t + 1) = lRunden
  c(t + 2) = 0
  Do
    dZeitTotal = 0#
    dRundenZeit = dStartzeit
    For i = 1 To lRunden
      dZeitTotal = dZeitTotal + dRundenZeit
      For m = 1 To t
        If i = c(m) + 1 Then
          dZeitTotal = dZeitTotal + dBoxenstopp
          dRundenZeit = dResetZeit
          Exit For
        End If
      Next m
      If m > t Then dRundenZeit = dRundenZeit + dInkrement
    Next i
    If (dZeitBest > dZeitTotal) Or (Abs(dZeitBest - dZeitTotal) < 0.000000001) Then
      If dZeitBest > dZeitTotal Then
        dZeitBest = dZeitTotal
        sBoxenstopps = ""
        sSemiColon = ""
      End If
      sComma = ""
      sBoxenstopps = sBoxenstopps & sSemiColon
      For m = 1 To t
        sBoxenstopps = sBoxenstopps & sComma & c(m) + 1
        sComma = ", "
      Next m
      sSemiColon = "; "
    End If
    j = 1
    Do While c(j) + 1 = c(j + 1)
      c(j) = j - 1
      j = j + 1
    Loop
    c(j) = c(j) + 1
  Loop Until j > t
  Cells(t + 5, 5) = t
  Cells(t + 5, 6) = dZeitBest
  Cells(t + 5, 7) = sBoxenstopps
Next t

Columns("E:G").EntireColumn.AutoFit
If Columns("G:G").ColumnWidth > 70 Then Columns("G:G").ColumnWidth = 70

End Sub
```

Rundensystem für Turnier Jeder Gegen Jeden - sbRoundRobin

Falls Sie ein Turnier organisieren müssen, bei dem jeder einmal gegen jeden anderen Spieler spielen muss, können Sie dieses Programm verwenden. Es implementiert das sogenannte Rutschsystem:

Illustratives Beispiel für 8 oder 9 Spieler:

	8	7	6	5
	Schwarz	Weiß	Schwarz	Weiß
9	Tisch 1	2	3	4
	Weiß	Schwarz	Weiß	Schwarz
	Platz 1	2	3	4

Bei gerader Spieleranzahl bleibt Spieler 1 an Tisch 1 / Platz 1 und wechselt jedes Spiel die Farbe, alle anderen Spieler rutschen entgegen dem Uhrzeigersinn einen Platz weiter und von Platz 8 nach Platz 2.

Bei ungerader Spieleranzahl rutschen alle Spieler entgegen dem Uhrzeigersinn einen Platz weiter, von Platz 8 geht es nach 9 mit einer Runde Pause, und von Platz 9 dann nach 1.

Ein Beispiel für 6 Spieler:

	Tisch 1	Tisch 2	Tisch 3
Runde 1	1 - 6	5 - 2	3 - 4
Runde 2	5 - 1	4 - 6	2 - 3
Runde 3	1 - 4	3 - 5	6 - 2
Runde 4	3 - 1	2 - 4	5 - 6
Runde 5	1 - 2	6 - 3	4 - 5

Das unten gezeigte VBA Programm – jedoch nicht der Tabellenblattfunktionsansatz in Kapitel „**Fehler! Verweisquelle konnte nicht gefunden werden.**“ auf Seite **Fehler! Textmarke nicht definiert.** – generiert auch diese Art von Paarungstabelle:

	Spieler 1	Spieler 2	Spieler 3	Spieler 4	Spieler 5	Spieler 6
Spieler 1	X	Runde 5, Tisch 1, Weiß	Runde 4, Tisch 1, Schwarz	Runde 3, Tisch 1, Weiß	Runde 2, Tisch 1, Schwarz	Runde 1, Tisch 1, Weiß
Spieler 2	Runde 5, Tisch 1, Schwarz	X	Runde 2, Tisch 3, Weiß	Runde 4, Tisch 2, Weiß	Runde 1, Tisch 2, Schwarz	Runde 3, Tisch 3, Schwarz
Spieler 3	Runde 4, Tisch 1, Weiß	Runde 2, Tisch 3, Schwarz	X	Runde 1, Tisch 3, Weiß	Runde 3, Tisch 2, Weiß	Runde 5, Tisch 2, Schwarz
Spieler 4	Runde 3, Tisch 1, Schwarz	Runde 4, Tisch 2, Schwarz	Runde 1, Tisch 3, Schwarz	X	Runde 5, Tisch 3, Weiß	Runde 2, Tisch 2, Weiß
Spieler 5	Runde 2, Tisch 1, Weiß	Runde 1, Tisch 2, Weiß	Runde 3, Tisch 2, Schwarz	Runde 5, Tisch 3, Schwarz	X	Runde 4, Tisch 3, Weiß
Spieler 6	Runde 1, Tisch 1, Schwarz	Runde 3, Tisch 3, Weiß	Runde 5, Tisch 2, Weiß	Runde 2, Tisch 2, Schwarz	Runde 4, Tisch 3, Schwarz	X

Weiterführende Literatur

Suksompong, W. (2018, April 11). Scheduling Asynchronous Round-Robin Tournaments. https://arxiv.org/pdf/1804.04504.pdf

Abel, Finizio, Greig, Lewis (2003). Generalized whist tournament designs. https://www.researchgate.net/publication/222140264_Generalized_whist_tournament_designs

Abel, Finizio, Greig, Morales (2008). Existence of (2, 8) GWhD(v) and (4, 8) GWhD(v) with v ≡ 0, 1 (mod 8). https://www.researchgate.net/profile/Malcolm_Greig2/publication/257554633_Existence_of_2_8_GWhDv_and_4_8_GWhDv_with_v_equiv_01_mod_8/links/56f56a5f08ae7c1fda2ee68f.pdf

Richard A. DeVenezia's Homepage:
https://www.devenezia.com/downloads/round-robin/index.html

Direkt verwendbare Turniertabellen:
https://www.printyourbrackets.com/roundrobin.html

sbRoundRobin Programmcode

```
Const CFirstOutputRow = 10

Sub sbRoundRobin()
'Creates a round robin tournament.
'(C) (P) by Bernd Plumhoff  19-May-2023 PB V0.4

Dim bPause           As Boolean

Dim c                As Long
Dim c1               As Long 'Colours, 1 = White (Home game), 2 = Black (Away game)
Dim f                As Long 'Player who has to pause
Dim i                As Long
Dim j                As Long
Dim k                As Long
Dim n                As Long 'Number of players
Dim p                As Long 'Number of players who can play
Dim r                As Long 'Number of rounds
Dim t                As Long 'Temporary storage during moves

Dim state            As SystemState

'Initialize
Set state = New SystemState
n = Range("Number_of_Players")
c = Range("Player1_Game1")
wsR.Range(CFirstOutputRow & ":" & 16382 + CFirstOutputRow).EntireRow.Delete

If n < 2 Then
    wsR.Cells(CFirstOutputRow, 1) = "'Spieleranzahl muss 2 oder höher sein!"
    Exit Sub
End If
If n > 16383 Then
    wsR.Cells(CFirstOutputRow, 1) = "'Spieleranzahl muss 16383 oder geringer sein!"
    Exit Sub
End If
If c < 1 Or c > 2 Then
    wsR.Cells(CFirstOutputRow, 1) = "'Die Farbe des Spielers 1 in Spiel 1 muss 1 (Weiß) oder 2 (Schwarz)
sein!"
    Exit Sub
End If

wsT.Cells.EntireRow.Delete

ReDim vR(1 To n + 1, 1 To n / 2 + 2) As Variant
ReDim vT(1 To n + 1, 1 To n + 1) As Variant

For i = 1 To n
    vT(1 + i, 1) = "Spieler " & i
    vT(1, 1 + i) = "Spieler " & i
    vT(1 + i, 1 + i) = "'X"
Next i

c1 = c

If n Mod 2 = 0 Then
    bPause = False
    p = n
    r = n - 1
Else
    bPause = True
    p = n - 1
    r = n
End If
ReDim a(1 To p) As Long
For i = 1 To p
    a(i) = i
Next i
j = 0
If bPause Then
    f = n
    vR(1, 2) = "Frei"
    j = 1
End If
```

```
For i = 1 To p / 2
    vR(1, i + j + 1) = "Tisch " & i
Next i

For i = 1 To r

    'Output of of current game pairings
    vR(1 + i, 1) = "'Runde " & i
    j = 2
    If bPause Then
        vR(1 + i, j) = f & " pausiert"
        j = j + 1
    End If
    If c1 = 1 Then
        vR(1 + i, j) = "'" & a(1) & " - " & a(UBound(a))
        vT(1 + a(1), 1 + a(UBound(a))) = "Runde " & i & ", Tisch 1, Weiß"
        vT(1 + a(UBound(a)), 1 + a(1)) = "Runde " & i & ", Tisch 1, Schwarz"
    Else
        vR(1 + i, j) = "'" & a(UBound(a)) & " - " & a(1)
        vT(1 + a(1), 1 + a(UBound(a))) = "Runde " & i & ", Tisch 1, Schwarz"
        vT(1 + a(UBound(a)), 1 + a(1)) = "Runde " & i & ", Tisch 1, Weiß"
    End If
    j = j + 1
    For k = 2 To UBound(a) / 2
        If (c + k) Mod 2 = 0 Then
            vR(1 + i, j) = "'" & a(k) & " - " & a(UBound(a) - k + 1)
            vT(1 + a(k), 1 + a(UBound(a) - k + 1)) = "Runde " & i & ", Tisch " & k & ", Weiß"
            vT(1 + a(UBound(a) - k + 1), 1 + a(k)) = "Runde " & i & ", Tisch " & k & ", Schwarz"
        Else
            vR(1 + i, j) = "'" & a(UBound(a) - k + 1) & " - " & a(k)
            vT(1 + a(k), 1 + a(UBound(a) - k + 1)) = "Runde " & i & ", Tisch " & k & ", Schwarz"
            vT(1 + a(UBound(a) - k + 1), 1 + a(k)) = "Runde " & i & ", Tisch " & k & ", Weiß"
        End If
        j = j + 1
    Next k

    'Move on to next round
    If bPause Then
        t = f
        f = a(UBound(a))
        j = 2
    Else
        c1 = 3 - c1 'Switch colour for player 1
        t = a(UBound(a))
        j = 3
    End If
    For k = UBound(a) To j Step -1
        a(k) = a(k - 1)
    Next k
    a(j - 1) = t

Next i

wsR.Range(wsR.Cells(CFirstOutputRow, 1), wsR.Cells(CFirstOutputRow + n, 2 + n / 2)) = vR
wsT.Range(wsT.Cells(1, 1), wsT.Cells(n + 1, n + 1)) = vT
wsT.Cells.EntireColumn.AutoFit

End Sub
```

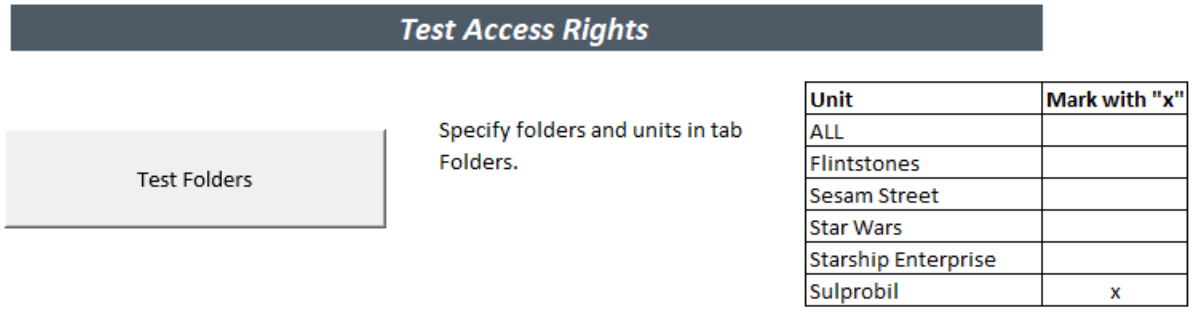
Zugriffsrechte Prüfen

Sie arbeiten in einer relativ komplexen Umgebung? In der Sie Lese- und Schreibrechte auf Dutzende von Verzeichnissen benötigen? Sie müssen diese Zugriffe bei Ihrer EDV Abteilung bestellen und dann wiederholt prüfen, ob diese Rechte zugewiesen wurden?

Dann kann dieses Program Ihnen helfen. Zuerst spezifizieren Sie alle notwendigen Zugriffsrechte, ggf. für mehrere Teams:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Folder [full path without trailing "\", no drive mapping]	Flintstones	Read	Write	Sesam Street	Read	Write	Star Wars	Read	Write	Starship Enterprise	Read	Write	Sulprobil	Read	Write	End
2	C:\Program Files	x	x	x				x	x	x							
3	C:\Windows	x	x	x				x	x	x							
4	C:\Windows\Temp	x	x	x				x	x	x							
5	C:\Program Files				x	x	x	x	x	x							
6	C:\Windows				x	x	x										
7	C:\Windows\Temp				x	x	x										
8	C:\Program Files										x	x	x				
9	C:\Windows										x	x	x				
10	C:\Windows\Temp										x	x	x				
11	C:\Program Files													x	x	x	
12	C:\Windows													x	x	x	
13	C:\Windows\Temp													x	x	x	

Dann lassen Sie dieses Programm laufen:



Nun können Sie sehen, welche Zugriffsrechte Sie haben:

Message
EVER: Sulprobil\Bernd 11.01.2023 10:44:49 [Start_Log] - Logging started with Test_Access_Rights_Version_22, Windows (64-bit) NT 10.00 / Microsoft Windows 11 Home 10.0.22000 and Microsoft Excel [Excel 2021/365] (64-bit) 16.014332 (191029)
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [Start_Log] - Application ThousandsSeparator '.', DecimalSeparator ',', use system separators
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [Start_Log] - App.Internl ThousandsSeparator '.', DecimalSeparator ',', ListSeparator ';'
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [Start_Log] - App.Internl xlCountryCode '49', xlCountrySetting '49'
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [Start_Log] - VBAProject References: Visual Basic For Applications, Microsoft Excel 16.0 Object Library, OLE Automation, Microsoft Office 16.0 Object Library, Microsoft Forms 2.0 Object Library, Microsoft ActiveX Data Objects 6.1 Library, Microsoft Outlook 16.0 Object Library, Microsoft Scripting Runtime
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [TestFolders] - Testing access to folders now
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [TestFolders] - Unit Sulprobil has value 'x'
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [TestFolders] - Can access (read) folder 'C:\Program Files'
##FATAL: Sulprobil\Bernd 11.01.2023 10:44:49 [TestFolders] - Cannot access (write) folder 'C:\Program Files'. Error number: 75
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [TestFolders] - Can access (read) folder 'C:\Windows'
##FATAL: Sulprobil\Bernd 11.01.2023 10:44:49 [TestFolders] - Cannot access (write) folder 'C:\Windows'. Error number: 75
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [TestFolders] - Can access (read) folder 'C:\Windows\Temp'
##FATAL: Sulprobil\Bernd 11.01.2023 10:44:49 [TestFolders] - Cannot access (write) folder 'C:\Windows\Temp'. Error number: 53
INFO: Sulprobil\Bernd 11.01.2023 10:44:49 [TestFolders] - Testing access to folders finished

Testfolders Programmcode

```
Public Const AppVersion As String = "Test_Access_Rights_Version_22" 'Each log will show which version it
has been created with

Sub TestFolders()
'Test folder access.
'(C) (P) by Bernd Plumhoff  11-Jan-2023 PB V22

Dim bRead As Boolean, bWrite As Boolean
Dim FileNumber As Integer
Dim i As Long, j As Long
Dim s As String, sTry As String
Dim state As SystemState
Dim oUnit As Object
Dim v As Variant

Set state = New SystemState
If GLogger Is Nothing Then Call auto_open
GLogger.SubName = "TestFolders"
GLogger.info "Testing access to folders now"
Main.Calculate
Set oUnit = CreateObject("Scripting.Dictionary")
For Each v In Range("Units_Selected")
    s = Main.Range(v.Address).Offset(0, 1).Text
    oUnit(CStr(v)) = s
    If s = "x" Then GLogger.info "Unit " & v & " has value 'x'"
Next v
On Error GoTo ErrHdl
i = 2
s = wsF.Cells(i, 1)
Do While s <> ""
    Application.StatusBar = "Testing " & s
    bRead = False: bWrite = False
    If oUnit("ALL") = "x" Then
        bRead = True
        bWrite = True
    Else
        j = 2
        Do While wsF.Cells(1, j) <> "End"
            If oUnit(wsF.Cells(1, j).Text) = "x" Then
                If wsF.Cells(i, j) = "x" Then
                    If wsF.Cells(i, j + 1) = "x" Then bRead = True
                    If wsF.Cells(i, j + 2) = "x" Then bWrite = True
                End If
            End If
            j = j + 3
        Loop
    End If
    If bRead Then
        'Folder readable? Let us check this by ChDir into it
        sTry = "read"
        ChDir (s)
        GLogger.info "Can access (" & sTry & ") folder '" & s & "'"
    End If
    If bWrite Then
        'Folder writeable? Try to create Remove_me.txt here
        sTry = "write"
        FileNumber = FreeFile
        Open s & "\Remove_me.txt" For Output As #FileNumber
```

```
        Write #FileNumber, "This is just a write test. This file should" & _
            "get deleted again automatically. If it does not," & _
            " please do it manually. Thank you."
        Close #FileNumber
        Kill s & "\Remove_me.txt"
        GLogger.info "Can access (" & sTry & ") folder '" & s & "'"
    End If
LabelNext:
    i = i + 1
    s = wsF.Cells(i, 1)
Loop

GLogger.info "Testing access to folders finished"
Exit Sub

ErrHdl:
Select Case Err.Number
Case 52
    'Dir(s, vbDirectory) went wrong
    GLogger.fatal "Cannot access (" & sTry & ") folder '" & s & "'" & _
        IIf(sTry = "read" And bWrite, " - write access expected", "")
    Resume LabelNext 'Back to next row
Case 76
    'ChDir (s) was not possible
    GLogger.fatal "Cannot access (" & sTry & ") folder '" & s & "'" & _
        IIf(sTry = "read" And bWrite, " - write access expected", "")
    Resume LabelNext 'Back to next row
Case Else
    GLogger.fatal "Cannot access (" & sTry & ") folder '" & s & _
        "'. Error number: " & Err.Number & _
        IIf(sTry = "read" And bWrite, " - write access expected", "")
    Resume LabelNext 'Back to next row
End Select

End Sub
```

Urlaubstage Optimal Nutzen

Jedes Jahr stellt sich für die Berufstätigen unter uns die Frage, wann wir unseren Urlaub nehmen sollten.

Dieses Programm zeigt Ihnen, mit wieviel Urlaubstagen Sie welche Gesamtzahl an freien Tagen erreichen können:

Beispiel: Mit den angegebenen Feiertagen können Sie 2024 mit 7 Urlaubstagen 16 Tage vom 14. Dezember 2024 bis 29. Dezember 2024 frei nehmen. Mehr kann man mit 7 Tagen Urlaub 2024 nicht erreichen, wie die zweitbeste Lösung beweist (da sind es lediglich 14 freie Tage).

E3 =sbOptimalVacationDays(C3;C6;C9;Bank_Holidays)

	A	C	E	F	G	H	I	J	K
1	Bank Holidays	Input	Output	Best			Second Best		
2	Mo 01.01.2024	# of Holidays	# of Holidays	# of free days	First Day	Last Day	# of free days	First Day	Last Day
3	Sa 06.01.2024	30	1	6	Di 24.12.2024	So 29.12.2024	6	Sa 21.12.2024	Do 26.12.2024
4	Fr 29.03.2024		2	9	Sa 21.12.2024	So 29.12.2024	7	Fr 20.12.2024	Do 26.12.2024
5	Mo 01.04.2024	Start Date	3	10	Fr 20.12.2024	So 29.12.2024	8	Do 19.12.2024	Do 26.12.2024
6	Mi 01.05.2024	01.01.2024	4	11	Do 19.12.2024	So 29.12.2024	10	Fr 29.03.2024	So 07.04.2024
7	Do 09.05.2024		5	12	Mi 18.12.2024	So 29.12.2024	11	Fr 29.03.2024	Mo 08.04.2024
8	Mo 20.05.2024	End Date	6	13	Di 17.12.2024	So 29.12.2024	13	Sa 14.12.2024	Do 26.12.2024
9	Do 30.05.2024	31.12.2024	7	16	Sa 14.12.2024	So 29.12.2024	14	Fr 13.12.2024	Do 26.12.2024
10	Do 15.08.2024		8	17	Fr 13.12.2024	So 29.12.2024	16	Sa 18.05.2024	So 02.06.2024
11	Do 03.10.2024		9	18	Do 12.12.2024	So 29.12.2024	17	Sa 18.05.2024	Mo 03.06.2024
12	Fr 01.11.2024		10	19	Mi 11.12.2024	So 29.12.2024	18	Sa 18.05.2024	Di 04.06.2024
13	Di 24.12.2024		11	20	Di 10.12.2024	So 29.12.2024	20	Sa 07.12.2024	Do 26.12.2024
14	Mi 25.12.2024		12	23	Sa 07.12.2024	So 29.12.2024	21	Fr 06.12.2024	Do 26.12.2024
15	Do 26.12.2024		13	24	Fr 06.12.2024	So 29.12.2024	24	Sa 27.04.2024	Mo 20.05.2024
16	Di 31.12.2024		14	25	Do 05.12.2024	So 29.12.2024	25	Do 09.05.2024	So 02.06.2024
17	Mi 01.01.2025		15	26	Mi 04.12.2024	So 29.12.2024	26	Do 09.05.2024	Mo 03.06.2024
18	Mo 06.01.2025		16	27	Di 03.12.2024	So 29.12.2024	27	Sa 30.11.2024	Do 26.12.2024
19	Fr 18.04.2025		17	30	Sa 30.11.2024	So 29.12.2024	30	Sa 04.05.2024	So 02.06.2024
20	Mo 21.04.2025		18	31	Fr 29.11.2024	So 29.12.2024	31	Sa 04.05.2024	Mo 03.06.2024
21	Do 01.05.2025		19	33	Mi 01.05.2024	So 02.06.2024	32	Sa 27.04.2024	Di 28.05.2024
22	Do 29.05.2025		20	34	Mi 01.05.2024	Mo 03.06.2024	34	Di 30.04.2024	So 02.06.2024
23	Mo 09.06.2025		21	37	Sa 27.04.2024	So 02.06.2024	35	Fr 26.04.2024	Do 30.05.2024
24	Do 19.06.2025		22	38	Sa 27.04.2024	Mo 03.06.2024	38	Fr 26.04.2024	So 02.06.2024
25	Fr 15.08.2025		23	39	Sa 27.04.2024	Di 04.06.2024	39	Fr 26.04.2024	Mo 03.06.2024
26	Fr 03.10.2025		24	40	Mi 01.05.2024	So 09.06.2024	40	Sa 27.04.2024	Mi 05.06.2024
27	Sa 01.11.2025		25	41	Mi 01.05.2024	Mo 10.06.2024	41	Di 30.04.2024	So 09.06.2024
28	Mi 24.12.2025		26	44	Sa 27.04.2024	So 09.06.2024	44	Sa 20.04.2024	So 02.06.2024
29	Do 25.12.2025		27	45	Sa 27.04.2024	Mo 10.06.2024	45	Fr 26.04.2024	So 09.06.2024
30	Fr 26.12.2025		28	46	Sa 27.04.2024	Di 11.06.2024	46	Fr 26.04.2024	Mo 10.06.2024
31	Mi 31.12.2025		29	47	Mi 01.05.2024	So 16.06.2024	47	Sa 27.04.2024	Mi 12.06.2024
32			30	48	Mi 01.05.2024	Mo 17.06.2024	48	Di 30.04.2024	So 16.06.2024

sbOptimalVacationDays Programmcode

```
Function sbOptimalVacationDays(lNumberOfVacationDays As Long, _
    dtStartDate As Date, _
    dtEndDate As Date, _
    vBankHolidays As Variant) As Variant
'This function informs you for all days from 1..lNumberOfVacationDays which exact days would result in the
'longest (as well as second longest) series of subsequent free days with given vBankHolidays and assuming
'that weekends (Saturdays and Sundays) are also free.
'(C) (P) Bernd Plumhoff v0.2 07-Jan-2024
Dim dt      As Date
Dim dtMax   As Date
Dim i       As Long
Dim j       As Long
ReDim v(1 To lNumberOfVacationDays, 1 To 7) As Variant
With Application.WorksheetFunction
If lNumberOfVacationDays < 1 Then
  sbOptimalVacationDays = CVErr(xlErrValue)
  Exit Function
End If
If dtStartDate > dtEndDate Then
  sbOptimalVacationDays = CVErr(xlErrNum)
  Exit Function
End If
For i = 1 To lNumberOfVacationDays: v(i, 1) = i: Next i
For dt = dtStartDate To dtEndDate
  For i = 1 To lNumberOfVacationDays
    dtMax = .WorkDay(.WorkDay(dt - 1, 1, vBankHolidays), i, vBankHolidays) - 1
    If dtMax > dtEndDate Then Exit For
      j = dtMax - dt + 1
      If j >= v(i, 2) Then
        v(i, 7) = v(i, 4): v(i, 6) = v(i, 3)
        v(i, 5) = v(i, 2): v(i, 2) = j
        v(i, 3) = dt: v(i, 4) = dtMax
      End If
  Next i
Next dt
sbOptimalVacationDays = v
End With
End Function
```

VBA Programme für Fortgeschrittene

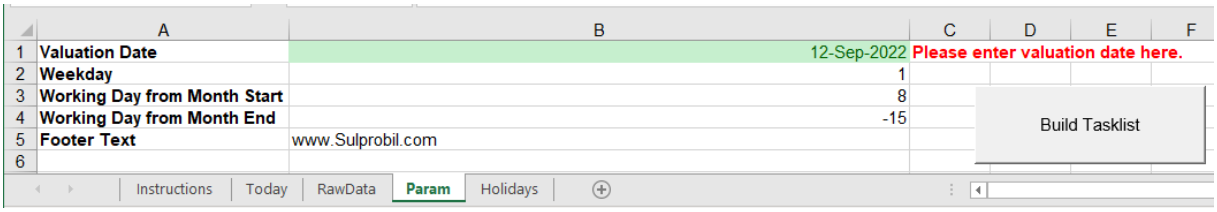
Abstract

In vielen Fällen benötigt man erweitere technische, betriebswirtschaftliche oder mathematische Kenntnisse. Dieses Kapitel gibt dazu Beispiele

Aufgabenliste – *sbTaskList*

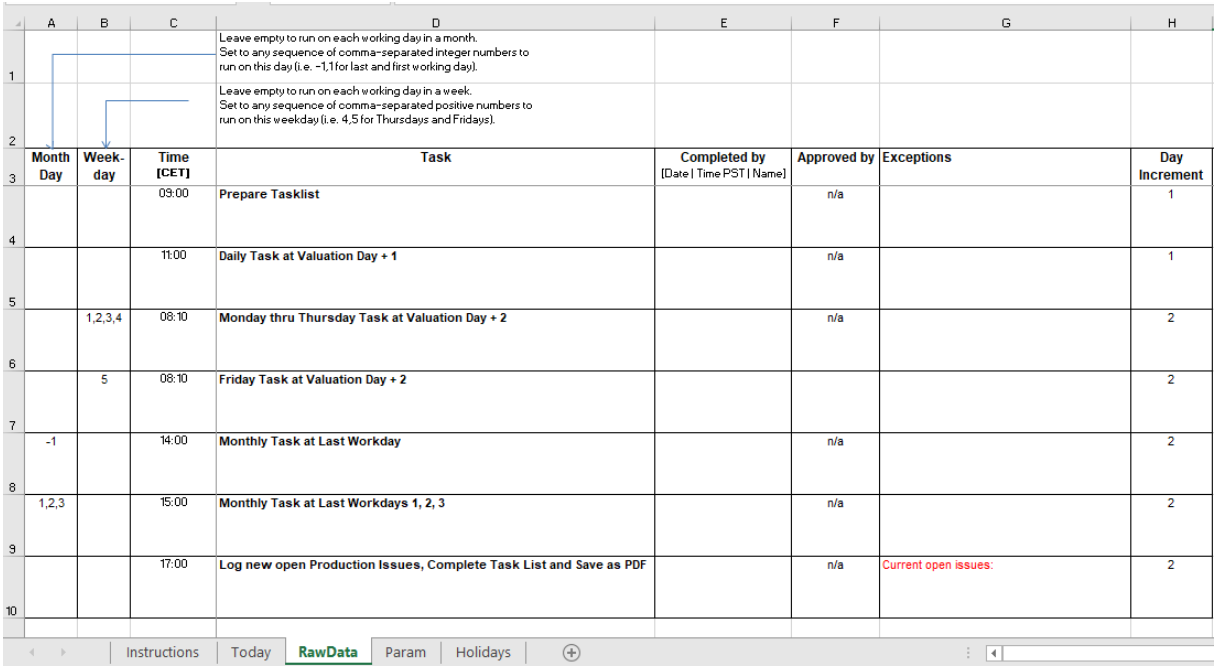
Falls Ihr Team viele verschiedene manuelle Aufgaben erledigen muss, nicht alle täglich sondern auch an unterschiedlichen Wochentagen oder Arbeitstagen im Monat, dann könnte diese Aufgabenliste sie unterstützen.

Im Arbeitsblatt Param geben Sie Ihren Teamnamen oder eine andere Referenz ein, die im Fußbereich jeder Seite erscheinen soll, und auch den Arbeitstag:



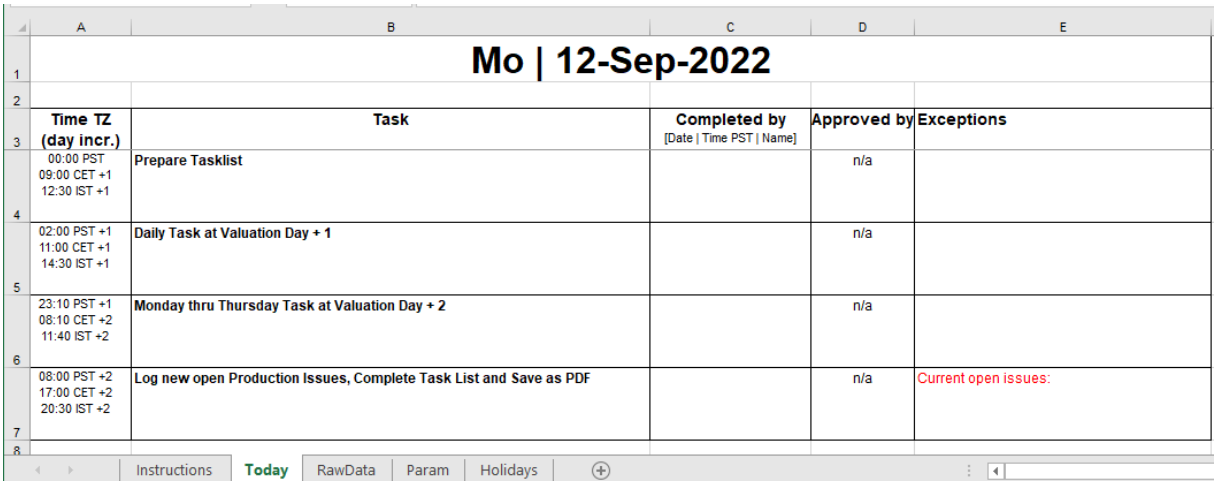
	A	B	C	D	E	F
1	**Valuation Date**	12-Sep-2022	**Please enter valuation date here.**			
2	**Weekday**	1				
3	**Working Day from Month Start**	8				
4	**Working Day from Month End**	-15				
5	**Footer Text**	www.Sulprobil.com				
6						

Im Arbeitsblatt RawData definieren Sie welche Aufgaben täglich, wöchentlich, oder monatlich zu welchen Zeiten durchgeführt werden müssen. Sie müssen diese Aufgaben nicht nach der Uhrzeit sortieren, aber es könnte hilfreich sein:



	A	B	C	D	E	F	G	H
1				Leave empty to run on each working day in a month. Set to any sequence of comma-separated integer numbers to run on this day (i.e. -1,1 for last and first working day).				
2				Leave empty to run on each working day in a week. Set to any sequence of comma-separated positive numbers to run on this weekday (i.e. 4,5 for Thursdays and Fridays).				
3	**Month Day**	**Week-day**	**Time [CET]**	**Task**	**Completed by** [Date | Time PST | Name]	**Approved by**	**Exceptions**	**Day Increment**
4			09:00	**Prepare Tasklist**		n/a		1
5			11:00	**Daily Task at Valuation Day + 1**		n/a		1
6		1,2,3,4	08:10	**Monday thru Thursday Task at Valuation Day + 2**		n/a		2
7		5	08:10	**Friday Task at Valuation Day + 2**				2
8	-1		14:00	**Monthly Task at Last Workday**		n/a		2
9	1,2,3		15:00	**Monthly Task at Last Workdays 1, 2, 3**		n/a		2
10			17:00	**Log new open Production Issues, Complete Task List and Save as PDF**		n/a	Current open issues:	2

Nun drücken Sie den Button im Arbeitsblatt Param und Sie erhalten in Arbeitsblatt Today:



	A	B	C	D	E
1	Mo | 12-Sep-2022				
2					
3	**Time TZ (day incr.)**	**Task**	**Completed by** [Date | Time PST | Name]	**Approved by**	**Exceptions**
4	00:00 PST 09:00 CET +1 12:30 IST +1	**Prepare Tasklist**		n/a	
5	02:00 PST +1 11:00 CET +1 14:30 IST +1	**Daily Task at Valuation Day + 1**		n/a	
6	23:10 PST +1 08:10 CET +2 11:40 IST +2	**Monday thru Thursday Task at Valuation Day + 2**		n/a	
7	08:00 PST +2 17:00 CET +2 20:30 IST +2	**Log new open Production Issues, Complete Task List and Save as PDF**		n/a	Current open issues:
8					

Instructions | Today | RawData | Param | Holidays

Drucken Sie das Arbeitsblatt Today aus. Lassen Sie Ihr Team alle Aufgaben abzeichnen (sobald sie erledigt wurden!) und lassen Sie sie alle aufgetretenen Ausnahmen (Probleme, Fehler, usw.) eintragen. Ich scannte normalerweise die signierte Taskliste am Ende jeden Tages, um einen papierlosen Revisionsnachweis zu haben.

Anmerkung: Sie können den gesamten Prozess papierlos durchführen, wenn Sie alle Eingaben elektronisch vornehmen lassen und die Ergebnisdatei als PDF abspeichern.

sbTaskList Programmcode

Hinweis: Dieses Programm verwendet die Klasse SystemState (siehe Seite 15).

```
Enum rawdata_columns
    rw_day = 1
    rw_weekday
    rw_time
    rw_task
    rw_completed_by
    rw_approved_by
    rw_exceptions
    rw_day_increment '+1 means time is given for day after valuation day, for example
    rw_comment
End Enum 'rawdata columns

Enum today_columns
    td_time = 1
    td_task
    td_completed_by
    td_approved_by
    td_exceptions
End Enum 'today columns

Sub Build_Tasklist()
'(C) (P) by Bernd Plumhoff 12-Sep-2022 PB V1.07

Dim bTBD                As Boolean 'To be done?
Dim dt                  As Date
Dim lrw                 As Long
Dim ltd                 As Long
Dim s                   As String
Dim v                   As Variant
Dim state               As SystemState

Set state = New SystemState
Application.Calculate
wsToday.Activate
wsToday.Rows("4:1048576").Delete

'Set destination column widths to source's
```

```
wsToday.Columns(Chr(64 + td_time) & ":" & Chr(64 + td_time)).ColumnWidth = _
    wsRawData.Columns(Chr(64 + rw_time) & ":" & Chr(64 + rw_time)).ColumnWidth
wsToday.Columns(Chr(64 + td_task) & ":" & Chr(64 + td_task)).ColumnWidth = _
    wsRawData.Columns(Chr(64 + rw_task) & ":" & Chr(64 + rw_task)).ColumnWidth
wsToday.Columns(Chr(64 + td_completed_by) & ":" & Chr(64 + td_completed_by)).ColumnWidth = _
    wsRawData.Columns(Chr(64 + rw_completed_by) & ":" & Chr(64 + rw_completed_by)).ColumnWidth
wsToday.Columns(Chr(64 + td_approved_by) & ":" & Chr(64 + td_approved_by)).ColumnWidth = _
    wsRawData.Columns(Chr(64 + rw_approved_by) & ":" & Chr(64 + rw_approved_by)).ColumnWidth
wsToday.Columns(Chr(64 + td_exceptions) & ":" & Chr(64 + td_exceptions)).ColumnWidth = _
    wsRawData.Columns(Chr(64 + rw_exceptions) & ":" & Chr(64 + rw_exceptions)).ColumnWidth

lrw = 4: ltd = 4
Do While Not (IsEmpty(wsRawData.Cells(lrw, rw_time))) 'As long as we have tasks timed ...

    Application.StatusBar = "Processing RawData row " & lrw & " ..."
    'Determine whether source row needs to be copied
    bTBD = False
    If IsEmpty(wsRawData.Cells(lrw, rw_day)) And IsEmpty(wsRawData.Cells(lrw, rw_weekday)) Then
        bTBD = True 'Empty rows will be copied
    Else
        'Check Month Day
        If Not (IsEmpty(wsRawData.Cells(lrw, rw_day))) Then
            For Each v In Split(wsRawData.Cells(lrw, rw_day).Text, ",")
                If CLng(v) = wsParam.Range("Evaldate_WDMS") Or CLng(v) = wsParam.Range("Evaldate_WDME")
Then
                    bTBD = True 'Right day from month start or month end: copy!
                    Exit For
                End If
            Next v
        End If
        'Check Weekday
        If Not (IsEmpty(wsRawData.Cells(lrw, rw_weekday))) Then
            For Each v In Split(wsRawData.Cells(lrw, rw_weekday).Text, ",")
                If CLng(v) = wsParam.Range("Evaldate_Weekday") Then
                    bTBD = True 'Right weekday: copy!
                    Exit For
                End If
            Next v
        End If
    End If

    If bTBD Then
        'Task needs to be done - copy into sheet Today
        wsRawData.Range(wsRawData.Cells(lrw, rw_time), wsRawData.Cells(lrw, rw_exceptions)).Copy
        wsToday.Range(Cells(ltd, td_time), Cells(ltd, td_exceptions)).PasteSpecial Paste:=xlPasteValues
        wsToday.Range(Cells(ltd, td_time), Cells(ltd, td_exceptions)).PasteSpecial Paste:=xlPasteFormats
        wsToday.Range(Cells(ltd, td_time), Cells(ltd, td_exceptions)).PasteSpecial
Paste:=xlPasteAllUsingSourceTheme

        dt = Range("Evaldate") + wsRawData.Cells(lrw, rw_day_increment) + wsRawData.Cells(lrw, rw_time)
        dt = ConvertTime(dt, "Central European Standard Time", "Pacific Standard Time")
        s = Format(dt, "hh:nn") & " PST" & IIf(dt - Range("Evaldate") > 1, _
            " +" & Format(Int(dt - Range("Evaldate")), "0"), "") & vbCrLf
        dt = Range("Evaldate") + wsRawData.Cells(lrw, rw_day_increment) + wsRawData.Cells(lrw, rw_time)
        s = s & Format(dt, "hh:nn") & " CET" & IIf(dt - Range("Evaldate") > 1, _
            " +" & Format(Int(dt - Range("Evaldate")), "0"), "") & vbCrLf
        dt = ConvertTime(dt, "Central European Standard Time", "India Standard Time")
        s = s & Format(dt, "hh:nn") & " IST" & IIf(dt - Range("Evaldate") > 1, _
            " +" & Format(Int(dt - Range("Evaldate")), "0"), "") & vbCrLf
        wsToday.Cells(ltd, td_time) = s

        wsToday.Rows(ltd & ":" & ltd).EntireRow.AutoFit
        If wsToday.Rows(ltd & ":" & ltd).RowHeight < wsRawData.Rows(lrw & ":" & lrw).RowHeight Then
            wsToday.Rows(ltd & ":" & ltd).RowHeight = wsRawData.Rows(lrw & ":" & lrw).RowHeight
        End If
        ltd = ltd + 1
    End If
    lrw = lrw + 1
Loop

With wsToday.PageSetup
    .PrintTitleRows = "$1:$3"
    .PrintArea = "$A$1:$" & Chr(64 + td_exceptions) & "$" & ltd - 1
    On Error Resume Next 'Quick and dirty because next command rows will fail in case no printer is defined
    .Orientation = xlPortrait
    .FitToPagesWide = 1
    .FitToPagesTall = 1 + Int(ltd / 5) 'Just to ensure that we have enough pages
    .LeftFooter = wsParam.Range("Footer_Text")
    .CenterFooter = ""
    .RightFooter = "Page &P/&N"
    On Error GoTo 0
End With

End Sub
```

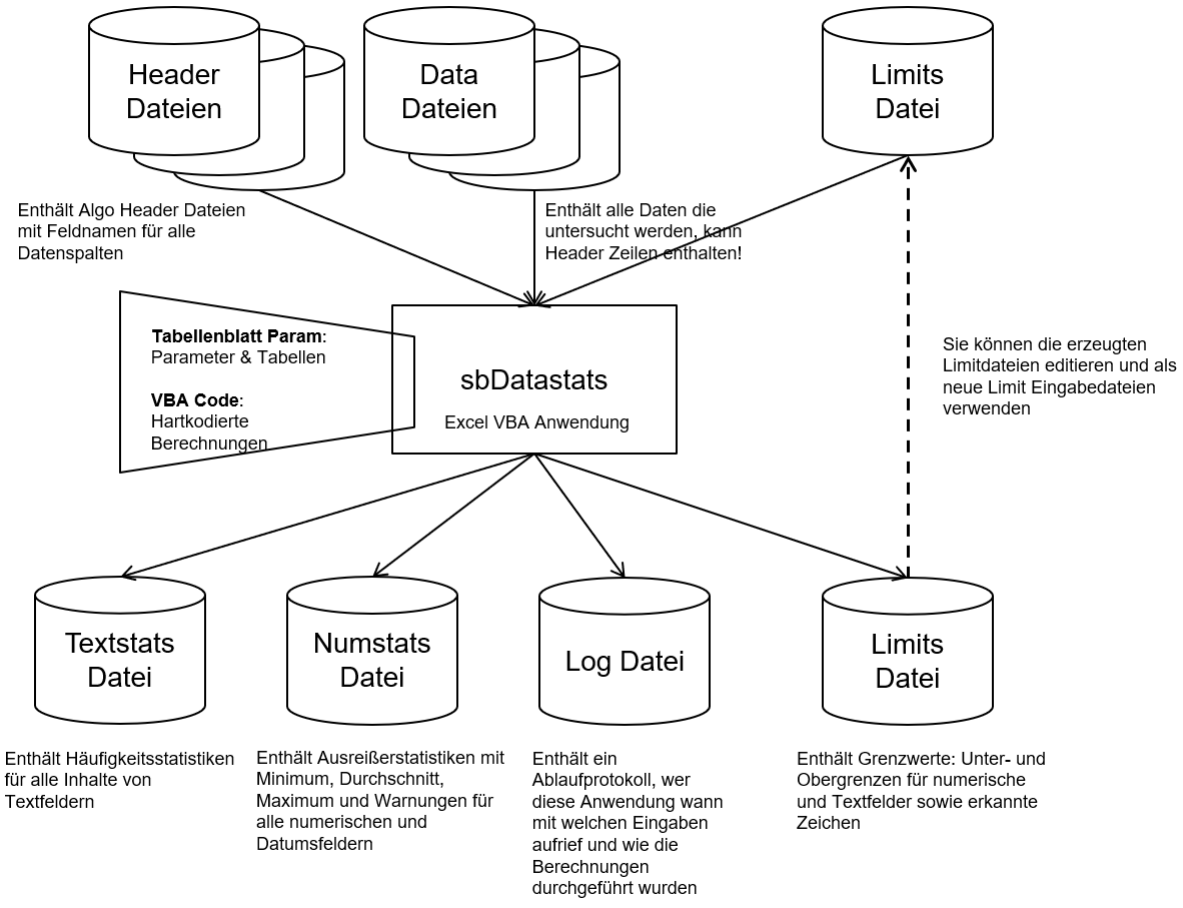
Data Analysis – *sbDatastats*

Wäre es nicht hilfreich, über ein Datenanalyse-Programm zu verfügen, welches numerische und Zeichenketten-Ausreißer ohne großen Aufwand identifizert?
Dies war zumindest die Designidee und der Umsetzungsansatz der Excel VBA sbDatastats Anwendung, die hier beschrieben wird.
Sie kopieren all Ihre Daten in ein Verzeichnis, alle Header Dateien (Dateien mit Spalteninformationen) in ein anderes und Sie legen einen Logverzeichnis fest. Dann drücken Sie lediglich eine Schaltfläche im Tabellenblatt Workflow und das Programm liefert eine Ausreißerstatistik für alle numerischen und Datumsfelder (mit Minimum, Durchschnitt, Maximum und weiteren Hinweisen) und für alle Zeichenkettenfelder (inklusive Häufigkeitsstatistik).
Dies kann sehr hilfreich sein, egal ob Sie im Datenservice, in IT oder Operations, oder in einer anderen Serviceeinheit oder sogar für die interne oder externe Revision arbeiten.

System Handbuch

Dieser Teil beschreibt wie die Anwendung installiert und gewartet wird. Es sind lediglich einfache Kenntnisse von Windows und MS Excel notwendig.

Übersicht



Die Installation erfolgt in wenigen einfachen Schritten. Zuerst erzeugen Sie ein neues Verzeichnis und kopieren die sbDatastats Anwendung hinein. Dann erzeugen Sie Unterverzeichnisse für die Daten (Data_Files und Data_Files_Prev), für AlgoOne Header (Header_Files), Logdateien (Logs), Konfigurationdateien (Config) und Ausgabedateien (Datastats). Schließlich initialisieren Sie die Anwendungsparameter im Tabellenblatt Param der Excel VBA Anwendung.
Das Anwendungsdesign ist das der Anwender nun lediglich die neuen Datendateien in die Datenunterverzeichnisse kopieren muss und dann die Limitdateien anpasst, umbenennt und in das Verzeichnis Config kopiert, um der Anwendung mitzuteilen, welche Grenzwerte beachtet werden müssen.

Parameter im Tabellenblatt Param

Log_Level – Gewöhnlich auf 1 gesetzt um alle Ausgaben inklusive Fehlern, Warnungen und einfachen Informationen zu erhalten. Wenn Sie dies auf 2 setzen, erhalten Sie nur noch Warnungen und Fehlermeldungen. Da keine verwendeten Dateinamen und Prüfinformationen mehr mitgeteilt werden, kann das Protokoll nicht mehr als Revisionsnachweis dienen.

Warning Threshold (Anzahl der Standardabweichungen) – Enthält die Anzahl der Standardabweichungen, ab der ein Extremwert in der Warnspalte der Numstats Ausgabe erscheint. Standardwert sollte 3 sein.

Keep stats in tabs – Die sbDatastats Anwendung enthält vier Tabellenblätter namens Numstats, Textstats, LimitsIn und LimitsOut, die normalerweise auch in die Ausgabedateien Numstats_yyyymmdd.csv, Textstats_yyyymmdd.csv, Input_Limits_File und Output_Limits_File im Unterverzeichnis Datastats geschrieben werden. Für häufige Programmläufe ist es sehr hilfreich, diese Informationen im sofortigen Zugriff zu haben, setzen Sie dann diesen Wert auf WAHR. Wenn Sie strikt die Daten vom Programm trennen wollen, dann setzen Sie den Wert auf FALSCH und die Tabellenblätter werden nach jedem Programmlauf geleert.

Max number of string attributes per field – Dieser Parameter begrenzt die Anzahl der Attributfelder in der Eingabedatei für Limite.

No single character check – Falls auf WAHR gesetzt werden einzelne Zeichen nicht auf Gültigkeit geprüft.

Input File Delimiter – Dieser Parameter definiert das Feldtrennzeichen in allen Eingabedateien. Für CSV Dateien ist dies normalerweise ein Komma, aber Sie können ein beliebiges Zeichen spezifizieren, zum Beispiel ‚|‘ für PSV (engl. pipe separated values). Konfigurationsdateien und Limitdateien sind immer CSV Dateien.

Anwenderhandbuch

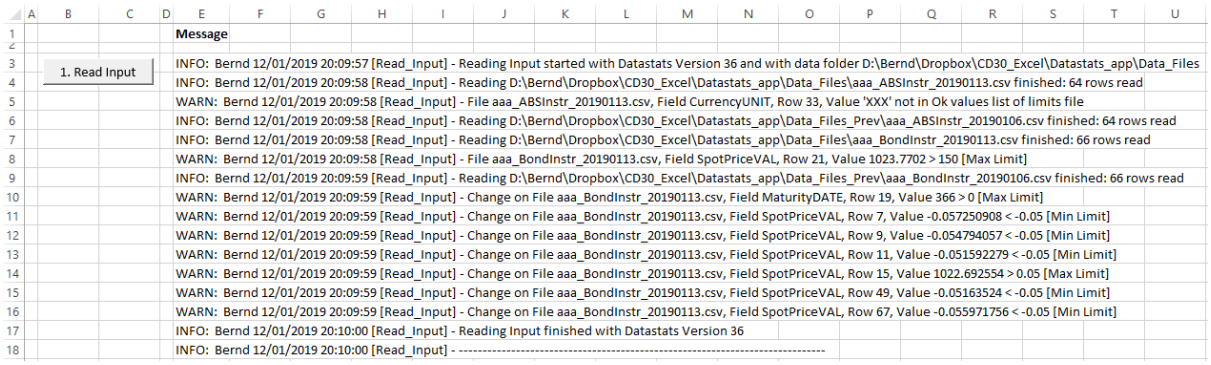
Dies beschreibt wie man die Anwendung ausführt.

Zusammenfassung

Kurz: Kopieren Sie alle Datendateien in die Unterverzeichnisse Data_Files und Daten der Vorperiode in das Unterverzeichnis Data_Files_Prev. Für AlgoOne Dateien müssen Sie die entsprechenden Header Dateien in das Unterverzeichnis Header_Files kopieren. Für alle anderen Dateien müssen Sie eine oder mehrere Sortierspalten in der FileSpecs.csv Datei im Unterverzeichnis Config bereitstellen. Danach drücken Sie die Schaltfläche ‚1. Read Input' im Tabellenblatt Workflow.

Nun warten Sie auf die finale Nachricht ‚INFO: … 16/08/2015 17:06:14 [Read_Input] – Reading Input finished'. Fehler, Warnungen und andere hilfreiche Informationen werden im Tabellenblatt Workflow ausgegeben.

Beispiel:



Diese Ausgaben werden auch in die Logdatei sbDatastats_Logfile_YYYYMMDD.txt im Unterverzeichnis Logs für Revisionszwecke ausgegeben.

Die eigentlichen Ausgabedateien der Anwendung sind schließlich im Unterverzeichnis Datastats.

Konfigurationsdatei FileSpecs.csv

Diese Datei befindet sich im Unterverzeichnis Config und enthält notwendige Informationen für die sbDatastats Anwendung, um Bewegungsprüfungen bei normalen (d. h. Nicht-AlgoOne-) Eingabedateien im Vergleich zur Vorperiode durchführen zu können:

	A	B	C	D
1	Filename	SortColumn1	SortColumn2	SortColumn3
2	Input	7		
3	More_Input	7		

Bis zu drei Sortierspalten geben der Anwendung die Möglichkeit, die Daten eindeutig zu sortieren, um gelöschte, neue, identische oder geänderte Datensätze zu identifizieren. Bitte beachten Sie, dass Spalte A lediglich die „Kern“-Dateinamen enthält. Das Programm erkennt automatisch, ob solch ein Kern-Dateiname DATEINAME ein Präfix (z. B. DATEINAME_YYYYMMDD.csv), ein Suffix (z. B. YYYYMMDD_DATEINAME.psv) oder mitten im tatsächlichen Dateinamen ist.

NumStats Ausgabe

Mit der Ausreißerstatistik für alle numerischen und Datumsfelder kann man leicht potentielle Fehler erkennen:

	A	B	C	D	E	F	G	H	I	J	K
1	Filename	RowType	Fieldname	Warning	Min 1	Min 2	Min 3	Average	Max 3	Max 2	Max 1
2	aaa_BondInstr_20190113.csv	BAS:DM Bond FutureSPEC	MaturityDATE		26-Nov-2030	11-Jan-2031	30-May-2031	27-Feb-2035	19-Jun-2039	19-Feb-2040	04-Sep-2040
3	aaa_BondInstr_20190113.csv	BAS:DM Bond FutureSPEC	SpotPriceVAL	Max is off by more than 3 stdev.	0.97	0.99	0.99	32.13	1.25	1.29	1,023.77

Die rote Umrandung zeigt einen Zinssatz, der um den Faktor 1000 zu hoch ist.

NumStatsMove Ausgabe

Mit der Ausreißerstatistik für Änderungen der numerischen und Datumsfelder kann man auch potentielle Fehler identifizieren:

	A	B	C	D	E	F	G	H	I	J	K
1	Filename	RowType	Fieldname	Warning	Min 1	Min 2	Min 3	Average	Max 3	Max 2	Max 1
2	aaa_BondInstr_20190113.csv	BAS:DM Bond FutureSPEC	MaturityDATE	Max is off by more than 3 stdev.	-	-	-	11.09	-	-	366.00
3	aaa_BondInstr_20190113.csv	BAS:DM Bond FutureSPEC	SpotPriceVAL	Max is off by more than 3 stdev.	(0.06)	(0.06)	(0.05)	30.98	0.04	0.04	1,022.69

Im obigen Fall wollen Sie sicherlich die Änderung des Ablaufdatums (engl. maturity date) und die Preisänderung hinterfragen.

TextStats Ausgabe

Mit der Häufigkeitsstatistik für jedes Textfeld können Sie verdächtige Zeichenketten leicht erkennen:

	A	B	C	D	E	F	G	H	I	J
1	Filename	RowType	Fieldname	Warning	Char	Char Frequency	Text Length	Frequency	Text	Count
2	aaa_ABSInstr_20190113.csv	BAS:DM Bond FutureSPEC	ContractSizeVAL		CHAR(067) = 'C'	32	15	32	'ContractSizeVAL'	32
3	aaa_ABSInstr_20190113.csv	BAS:DM Bond FutureSPEC	CurrencyUNIT		CHAR(088) = 'X'	3	3	32	'GBP'	14
4									'EUR'	11
5									'USD'	6
6									'XXX'	1

In diesem Fall handelt es sich um eine fehlerhafte Währung ‚XXX‘, die einmal auftrat. Im Allgemeinen werden Sie seltene Zeichenketten überprüfen wollen.

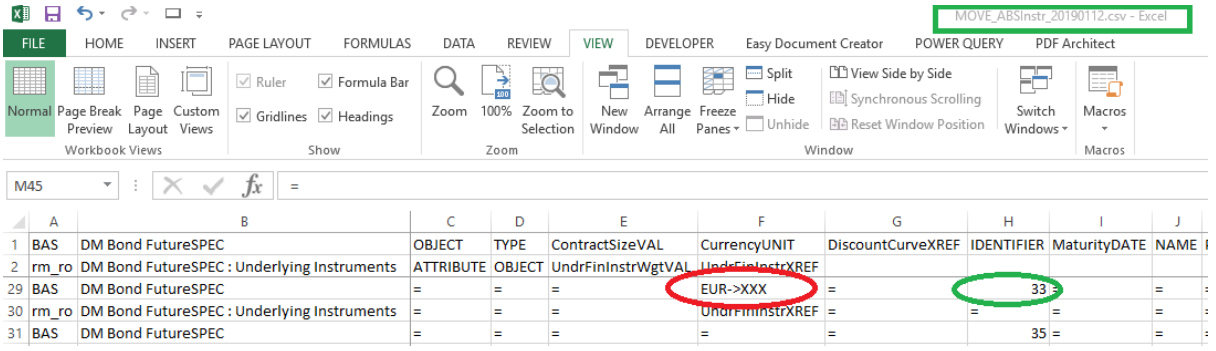
TextStatsMove Ausgabe

Mit der Häufigkeitsstatistik für Änderungen von Textfeldern erkennen Sie leicht verdächtige Änderungen:

	A	B	C	D	E	F	G	H	I	J
1	Filename	RowType	Fieldname	Warning	Char	Char Frequency	Text Length	Frequency	Text	Count
2	aaa_ABSInstr_20190113.csv	BAS:DM Bond FutureSPEC	ContractSizeVAL				1	32	'='	32
3	aaa_ABSInstr_20190113.csv	BAS:DM Bond FutureSPEC	CurrencyUNIT				1	31	'='	31
4							8		'EUR->XXX'	1
5	aaa_ABSInstr_20190113.csv	BAS:DM Bond FutureSPEC	DiscountCurveXREF				1	32	'='	32

Das Beispiel zeigt eine fehlerhafte Änderung der Währung von ‚EUR' zu ‚XXX'.

Wenn Sie den fehlerhaften Datensatz identifizieren wollen, müssen Sie in die entsprechende MOVE Datei für die ursprüngliche Datendatei in Spalte A sehen:



	A	B	C	D	E	F	G	H	I	J
1	BAS	DM Bond FutureSPEC	OBJECT	TYPE	ContractSizeVAL	CurrencyUNIT	DiscountCurveXREF	IDENTIFIER	MaturityDATE	NAME
2	rm_ro	DM Bond FutureSPEC : Underlying Instruments	ATTRIBUTE	OBJECT	UndrFinInstrWgtVAL	UndrFinInstrXREF				
29	BAS	DM Bond FutureSPEC	=	=	=	EUR->XXX	=	33	=	=
30	rm_ro	DM Bond FutureSPEC : Underlying Instruments	=	=	=	UndrFinInstrXREF	=	=	=	=
31	BAS	DM Bond FutureSPEC	=	=	=	=	=	35	=	=

Im obigen Fall wurde das Instrument mit dem Identifikator 33 fehlerhaft geändert.

Limits_Output Datei

Eine Beispielausgabe für eine Limits_Output Ausgabedatei nach einem Programmlauf:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Limit File Version 1.00												
2	Filename	RowType	Fieldname	Allow Empty	Min	Max	Min Date	Max Date	Length Min	Length Max	Formula	OkFormat	OkValues
3	ABSInstr	BAS:DM Bond FutureSPEC	ContractSizeVAL						15	15			ContractSizeVAL|
4	ABSInstr	BAS:DM Bond FutureSPEC	CurrencyUNIT						3	3			GBP|EUR|USD|XXX|
5	ABSInstr	BAS:DM Bond FutureSPEC	DiscountCurveXREF						17	17			DiscountCurveXREF|
6	ABSInstr	BAS:DM Bond FutureSPEC	IDENTIFIER		1	63							
41	BondInstr	BAS:DM Bond FutureSPEC	ProductTypeENUM						15	15			ProductTypeENUM|
42	BondInstr	BAS:DM Bond FutureSPEC	SettlementTYPE						14	14			SettlementTYPE|
43	BondInstr	BAS:DM Bond FutureSPEC	SpotPriceVAL		0.97169473	1023.7702							
44	BondInstr	BAS:DM Bond FutureSPEC	StlmntDayRuleBUSD						17	17			StlmntDayRuleBUSD|
45	BondInstr	BAS:DM Bond FutureSPEC	StlmntDayRuleCONV						17	17			StlmntDayRuleCONV|

Sie können sehen, dass die Datei die minimalen und maximalen numerischen und Datumswerte sowie die Textlängen und (wenn der Parameter NoSingleCharCheck auf FALSCH gesetzt wurde) die auftretenden Zeichen zeigt. Der Wert WAHR in der Spalte ‚Allow Empty' zeigt an, dass dieses Feld für mindestens einen Datensatz leer war.

Die Intention der Limits Ausgabedatei ist eine Hilfe zur Definition einer Limits Eingabedatei. Die Limitausgaben zeigen die Limite und Werte früherer Eingabedateien. Ein Beispiel für eine Limits Eingabedatei, die als Datei Limits_Input.csv in das Unterverzeichnis Config kopiert wird:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Limit File Version 1.00												
2	Filename	RowType	Fieldname	Allow Empty	Min	Max	Min Date	Max Date	Length Min	Length Max	Formula	OkFormat	OkValues
3	ABSInstr	BAS:DM Bond FutureSPEC	CurrencyUNIT						3	3			GBP|EUR|USD|
4	BondInstr	BAS:DM Bond FutureSPEC	SpotPriceVAL		0	150							

Die Spalten der Limits Datei sind:

Filename, RowType, und Fieldname – Diese Spalten identifizieren die Felder der Eingabedateien, auf die die Limite angewandt werden. RowType wird lediglich für AlgoOne Eingabedateien benötigt.

AllowEmpty – Muss WAHR oder FALSCH (leer) sein. WAHR bedeutet,dass dieses Feld leer sein darf. Wenn es leer oder FALSCH ist, warnt das Programm bei leeren Feldern.

Min, Max – Diese numerischen Werte definieren den kleinsten bzw. den größten erlaubten Wert für ein Feld. Lassen Sie diese Felder leer wenn keine Grenzwerte angewendet werden sollen.

Min Date, Max Date – Diese Datumswerte bestimmen die Grenzen der Datumswerte. Lassen Sie sie leer wenn sie nicht angewandt werden sollen.

Length Min, Length Max – Diese numerischen Werte definieren die Grenzen der erlaubten Eingabewerte. Leer lassen, wenn sie nicht gelten.

Formula – Sie können hier eine Tabellenblattformel eingeben, die WAHR oder FALSCH ergeben muss. Das Programm wird Datensätze bei WAHR ignorieren und bei FALSCH warnen. In der Formel referiert [Feld] auf das Datensatzfeld Feld und [ThisCell] auf die untersuchte Eingabezelle.

Hinweis: Excel / VBA's interne Datumsdarstellung verwendet das US Format mm/dd/yyyy. Wandeln Sie Zeichenketten in dieses Format um bevor Sie Funktionen wie DATWERT anwenden.

OKFormat – Hier können Sie einen regulären Ausdruck eingeben. Wenn die untersuchte Eingabezelle dem entspricht, warnt das Programm nicht.

OkValues – Geben Sie alle gültigen Werte getrennt durch ‚|' ein und das Programm wird nur diese akzeptieren.

CHAR(000) .. CHAR(255) . Definieren Sie diese Zeichen als gültig indem sie die entsprechende Spalte auf WAHR setzen.

Limits_Move_Output Datei

Ein Auszug aus einer Limitänderungsdatei, die Sie unter dem Namen Limits_Move_Output.csv im Unterordner Datastats finden, wenn Sie die Anwendung mit den Eingabedateien des vorherigen Zeitraums im Unterordner Data_Files_Prev ausgeführt haben, sodass das Programm die Änderungen bzw. Bewegungen analysieren konnte:

	A	B	C	D	E	F	G	H	I	J
1	Limit Moves File Version 1.00									
2	Filename	RowType	Fieldname	Allow Empty	Min	Max	Min Date	Max Date	Length Min	Length Max
33	BondInstr	BAS:DM Bond FutureSPEC	DiscountCurveXREF						1	1
34	BondInstr	BAS:DM Bond FutureSPEC	MaturityDATE		0	366				
35	BondInstr	BAS:DM Bond FutureSPEC	NAME						1	1
36	BondInstr	BAS:DM Bond FutureSPEC	PositionUnitsVAL						1	1
37	BondInstr	BAS:DM Bond FutureSPEC	PortfolioXREF						1	1
38	BondInstr	BAS:DM Bond FutureSPEC	ProductTypeENUM						1	1
39	BondInstr	BAS:DM Bond FutureSPEC	SettlementTYPE						1	1
40	BondInstr	BAS:DM Bond FutureSPEC	SpotPriceVAL		-0.057250908	1022.692554				

Wie Sie sehen können, listet diese Datei die minimalen und maximalen Änderungen von numerischen und Datumswerten sowie die Textlängen der Änderungen auf (eine ‚1' zeigt eindeutig keine

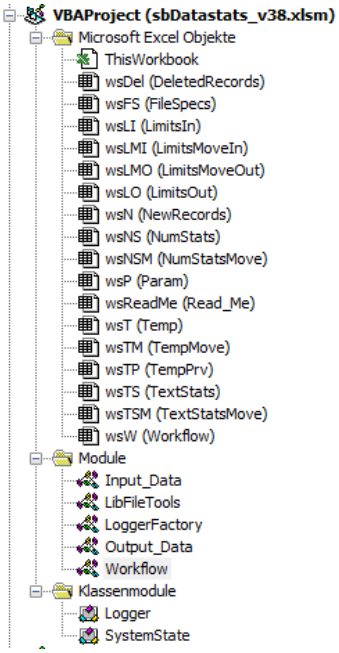
Änderung an, da es ‚=' bedeutet). Für das oben genannte Beispiel könnten Sie eine Limits_Move_Input.csv wie folgt definieren:

	A	B	C	D	E	F	G	H	I	J
1	Limit Moves File Version 1.00									
2	Filename	RowType	Fieldname	Allow Empty	Min	Max	Min Date	Max Date	Length Min	Length Max
3	BondInstr	BAS:DM Bond FutureSPEC	MaturityDATE		0	0				
4	BondInstr	BAS:DM Bond FutureSPEC	SpotPriceVAL		-0.05	0.05				

Programm Code sbDatastats

Diese Anwendung verwendet die Klassen Logger und SystemState sowie das Modul LibFileTools.

Das VBAProject:



Module Input_Data

```
Option Explicit

'Source: http://www.zerrtech.com/content/excel-vba-open-csv-file-and-import [14-Jun-2011]

Function doFileQuery(fileName As String, outSheet As String, Optional sDelimiter As String = ",") As
Boolean
'Reads in csv files.
' filename = CSV filename (complete pathname!)
' outSheet = name of the worksheet in the current workbook
'            where the data should go, will start in A1
'Change History:
'Version Date       Programmer Change
'1.00    14/06/2011 Bernd      Copied from the web
'1.01    15/06/2011 Bernd      Drop query before adding a new one, error handling, logging
'2       24/09/2017 Bernd      Also process psv files "|"
Dim connectionName           As String
Dim qt                       As QueryTable
On Error GoTo ErrHdl
doFileQuery = True
If Worksheets(outSheet).QueryTables.Count > 0 Then Worksheets(outSheet).QueryTables.Item(1).Delete
```

```
'GLogger.info "Create Querytable"
connectionName = "TEXT;" & fileName
With Worksheets(outSheet).QueryTables.Add(Connection:=connectionName, _
                                          Destination:=Worksheets(outSheet).Range("A1"))
  .Name = fileName
  .FieldNames = True
  .RowNumbers = False
  .FillAdjacentFormulas = False
  .PreserveFormatting = True
  .RefreshOnFileOpen = False
  .RefreshStyle = xlOverwriteCells
  .SavePassword = False
  .SaveData = True
  .AdjustColumnWidth = True
  .RefreshPeriod = 0
  .TextFileColumnDataTypes = Array(xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, xlTextFormat, _
                                   xlTextFormat, xlTextFormat, xlTextFormat)
  .TextFilePromptOnRefresh = False
  .TextFilePlatform = 437
  .TextFileStartRow = 1
  .TextFileParseType = xlDelimited
  .TextFileTextQualifier = xlTextQualifierDoubleQuote
  .TextFileConsecutiveDelimiter = False
  .TextFileTabDelimiter = False
  .TextFileSemicolonDelimiter = False
  .TextFileCommaDelimiter = False
  .TextFileSpaceDelimiter = False
  Select Case sDelimiter
  Case ","
    .TextFileCommaDelimiter = True
  Case ";"
    .TextFileSemicolonDelimiter = True
  Case " "
    .TextFileSpaceDelimiter = True
  Case Else
    .TextFileOtherDelimiter = sDelimiter
  End Select
  .Refresh BackgroundQuery:=False
End With
Exit Function
ErrHdl:
GLogger.SubName = "doFileQuery"
GLogger.fatal "Could not establish Querytable successfully"
doFileQuery = False
Resume errhdlexit
errhdlexit:
End Function

'If you have a file other than a CSV file, just change the
'Excel VBA options (.TextFileTabDelimiter, '.TextFileSemicolonDelimiter, etc..)
'to True and the (.TextFileCommaDelimiter) to False.
```

Module Output_Data

```
Option Explicit

Dim GsCurrentWorkbook As String
Dim GlCurrentFormat As Long

'Source: http://www.erlandsendata.no/english/index.php?d=envbatextexportcsv
Sub ExportRangeAsDelimitedText(SourceWS As String, SourceAddress As String, _
                               TargetFile As String, SepChar As String, SaveValues As Boolean, _
                               ExportLocalFormulas As Boolean, AppendToFile As Boolean)
' Exports the data in Workbooks(SourceWB).Worksheets(SourceWS).Range(SourceAddress) to
' the textfile TargetFile in CSV format, uses SepChar as column delimiter
' Example:     ExportRangeAsDelimitedText ThisWorkbook.Name, _
  "ExportSheet", "A3:E23", "C:\FolderName\DelimitedText.txt", _
  ";", True, True, False

  Dim SourceRange As Range, sC As String * 1
  Dim A As Integer, r As Long, c As Integer, totr As Long, pror As Long
  Dim fn As Integer, LineString As String, tLine As String
  ' validate the input data if necessary
  Worksheets(SourceWS).Activate
  If Application.WorksheetFunction.CountA(Range(SourceAddress)) = 0 Then Exit Sub
  If Not AppendToFile Then
    On Error Resume Next
    Kill TargetFile
    On Error GoTo 0
  End If
  If UCase(SepChar) = "TAB" Or UCase(SepChar) = "T" Then
    sC = Chr(9)
  Else
    sC = Left(SepChar, 1)
  End If

  ' perform export
  Set SourceRange = Range(SourceAddress)
  On Error GoTo NotAbleToExport
  fn = FreeFile
  Open TargetFile For Append As #fn  ' open textfile for new input
  On Error GoTo 0
  ' determine the total number of rows to process
  totr = 0
  For A = 1 To SourceRange.Areas.Count
    totr = totr + SourceRange.Areas(A).Rows.Count
  Next A
  ' start writing the character-separated textfile
  pror = 0
  For A = 1 To SourceRange.Areas.Count
    For r = 1 To SourceRange.Areas(A).Rows.Count
      LineString = ""
      For c = 1 To SourceRange.Areas(A).Columns.Count
        tLine = ""
        On Error Resume Next
        If SaveValues Then
          tLine = SourceRange.Areas(A).Cells(r, c).Value
        Else
          If ExportLocalFormulas Then
            tLine = SourceRange.Areas(A).Cells(r, c).FormulaLocal
          Else
            tLine = SourceRange.Areas(A).Cells(r, c).Formula
          End If
        End If
        On Error GoTo 0
        LineString = LineString & tLine & sC
      Next c
      pror = pror + 1
      If pror Mod 50 = 0 Then
        Application.StatusBar = "Writing delimited textfile " & _
                                Format(pror / totr, "0 %") & "..."
      End If
      If Len(LineString) > 1 Then LineString = Left(LineString, Len(LineString) - 1)
      If LineString = "" Then
        Print #fn,
      Else
        Print #fn, LineString
      End If
    Next r
  Next A
  Close #fn  ' close the textfile
NotAbleToExport:
  Set SourceRange = Nothing
  Application.StatusBar = False
End Sub
```

Module Workflow

```
Option Explicit

'Change History:
'Version Date       Programmer Change
'1       23/09/2015 Bernd      Initial Version
'37      19/01/2019 Bernd      lRowTypes + 2 to correctly skip header rows and use 1 instead of True
'                              for limAllowEmpty because of internationalization, and check for
'                              Datastats subfolder
'38      22/01/2025 Bernd      New modules LibFileTools, SystemState, and Logger / LoggerFactory

Public Const AppVersion As String = "Datastats_Version_38"
Public Const CLimFileVersion As String = "Limit File Version 1.00"
Public Const CLimMovFileVersion As String = "Limit Moves File Version 1.00"
Public Const CMinDouble As Double = -1.79769313486231E+308
Public Const CMaxDouble As Double = 1.79769313486231E+308
Public Const CMinDate As Double = -1.79769313486231E+308
Public Const CMaxDate As Double = 1.79769313486231E+308
Public Const CMinLength As Long = -1
Public Const CMaxLength As Long = 32767
Public Const CInitialUBounddStdDevInput As Long = 10000

Dim GdMax(1 To 3) As Double
Dim GlMaxIdx As Long
Dim GdMin(1 To 3) As Double
Dim GlMinIdx As Long

'We enumerate columns of some worksheets now which means we define
'names for the columns of some sheets we use.
'This is to stay flexible in case we need to delete/move/add a column.
'We can just delete, move or the/a name for that column and our code
'would get adjusted automatically.
'If you hardcode all columns (by referring to them with Range("H:H") for example)
'you would need to look for all locations and to change all these references
'AND other dependent columns.

Enum out_columns  'Structure of sheet NumStats and NumStatsMove
  outLbound = 0  'To be able to iterate from outLBound + 1
  outFile
  outHeader
  outField
  outWarn
  outMin1
  outMin2
  outMin3
  outAverage
  outMax3
  outMax2
  outMax1
  outUbound  'To be able to iterate until outUBound - 1
End Enum  'out_columns

Enum out2_columns  'Structure of sheet TextStats and TestStatsMove
  out2Lbound = 0  'To be able to iterate from out2LBound + 1
  out2File
  out2Header
  out2Field
  out2Warn
  out2Char
  out2CharCount
  out2Len
  out2LenCount
  out2Text
  out2TextCount
  out2Ubound  'To be able to iterate until out2UBound - 1
End Enum  'out2_columns

Enum lim_columns  'Structure of sheets LimitsIn, LimitsOut, LimitsMoveIn, and LimitsMoveOut
  limLbound = 0  'To be able to iterate from limLBound + 1
  limFilePrefix
  limHeader
  limField
  limAllowEmpty  'But we don't set this to True anymore - in Germany "WAHR"
  'won't be treated correctly. So let us use CBool(1)
  limNumMin
  limNumMax
  limDateMin
  limDateMax
  limLenMin
  limLenMax
  limFormula
  limOkFormat
  limOkValues
  limCharSet  '256 fields indicating whether this char may/does appear
  'limUbound 'To be able to iterate until limUBound - 1
End Enum  'lim_columns
```

```
Sub Read_Input()
'This code reads any csv file data and also Algo(rithmics) One input data files and performs
'numerical stats like min, avg, max on numerical fields and on date fields, and frequency
'stats on text fields.
'
'Change History:
'Version Date       Programmer Change
'1       09/10/2016 Bernd      Show Limit File version in tabs, right title row for limits file, and
'                              hint for Formula field
'8       19/01/2019 Bernd      lRowTypes + 2 to correctly skip header rows and use 1 instead of True
'                              for limAllowEmpty because of internationalization, and check for
'                              Datastats subfolders

  Const CsAttrDelim = "|"
  Dim b              As Boolean
  Dim bClearDataAfterRun As Boolean
  Dim bDeletedRecords As Boolean
  Dim bFatal         As Boolean
  Dim bFileSpecs     As Boolean
  Dim bHasEmpty      As Boolean
  Dim bIsIDField     As Boolean
  Dim bLimitsIn      As Boolean
  Dim bLimitsMoveIn  As Boolean
  Dim bIsDate        As Boolean
  Dim bHeaderFile    As Boolean
  Dim bMoveCheck     As Boolean
  Dim bNewRecords    As Boolean
  Dim bNoSingleCharCheck As Boolean
  Dim bProcessThisRow As Boolean
  Dim d              As Double
  Dim dMax           As Double
  Dim dMaxDate       As Double
  Dim dMin           As Double
  Dim dMinDate       As Double
  Dim dStdDevThreshold As Double
  Dim dSum           As Double
  Dim dVal           As Double
  Dim dtDate         As Date
  Dim i As Long, j As Long, k As Long, m As Long
  Dim lColumns       As Long
  Dim lColumnStart   As Long
  Dim lCount         As Long
  Dim lCountAll      As Long
  Dim lDel           As Long
  Dim lFileFormat    As Long
  Dim lLen           As Long
  Dim lLenMax        As Long
  Dim lLenMin        As Long
  Dim lLim           As Long
  Dim lLimRow        As Long
  Dim lLMO           As Long
  Dim lMainSortCol   As Long
  Dim lMax           As Long
  Dim lMaxColumns    As Long
  Dim lMaxOkValueCountPerField As Long
  Dim lMin           As Long
  Dim lNew           As Long
  Dim lNSM           As Long
  Dim lOut           As Long
  Dim lOut2          As Long
  Dim lRowTypes      As Long
  Dim lTM            As Long
  Dim lTSM           As Long
  Dim lUBounddStdDevInput As Long
  Dim oOkValues      As Object
  Dim oBASRow        As Object
  Dim oData          As Object
  Dim oFileSpecs     As Object
  Dim oHeader        As Object
  Dim oInputFiles    As Object
  Dim oLimitRow      As Object
  Dim oLimitMoveRow  As Object
  Dim oRegEx         As Object
  Dim oRowTypes      As Object
  Dim oSortCol       As Object
  Dim oStringLen     As Object
  Dim oChar          As Object
  Dim s              As String
  Dim sCell          As String
  Dim sCellPrev      As String
  Dim sDelimiter     As String
  Dim sDirData       As String
  Dim sDirDataPrev   As String
  Dim sField         As String
  Dim sFile          As String
  Dim sCoreFile      As String
  Dim sFilePrev      As String
  Dim sFileSpecs     As String
  Dim sFormula       As String
  Dim sFullName      As String
```

```
  Dim sHeader       As String
  Dim Sid           As String
  Dim sID2          As String
  Dim sID3          As String
  Dim sLimitsIn     As String
  Dim sLimitsMoveIn As String
  Dim sMsg          As String
  Dim sReverse      As String
  Dim state         As SystemState
  Dim v             As Variant
  Dim vFile         As Variant
  Dim vID           As Variant
  Dim vPresuffix    As Variant
  Dim vPresuffixPrev As Variant

  sFullName = ActiveWorkbook.FullName
  lFileFormat = ActiveWorkbook.FileFormat
  Set oRegEx = New regexp             'Add reference to Microsoft VBScript Regular Expressions 5.5 or more
  If GLogger Is Nothing Then Start_Log
  GLogger.LogLevel = wsP.Range("Log_Level")
  GLogger.SubName = "Read_Input"
  GLogger.info String(160, ">")
  GLogger.info "Reading Input started with data folder " & ThisWorkbook.Path & "\Data_Files"
  wsDel.Cells.NumberFormat = "@"                      'Treat whole sheet as text
  wsN.Cells.NumberFormat = "@"
  wsT.Cells.NumberFormat = "@"
  wsTP.Cells.NumberFormat = "@"
  sDelimiter = wsP.Range("Input_File_Delimiter")
  If sDelimiter = "" Then sDelimiter = ","

  'Now execute sub Class_Initialize() of class SystemState which stores those system variables
  Set state = New SystemState

  Set oHeader = CreateObject("Scripting.Dictionary")
  Set oInputFiles = CreateObject("Scripting.Dictionary")
  Set oSortCol = CreateObject("Scripting.Dictionary")
  bClearDataAfterRun = Not wsP.Range("Keep_Stats_In_Tabs")
  bNoSingleCharCheck = wsP.Range("NoSingleCharCheck")
  Call Delete_QueryTables

  wsN.Range("1:1048576").Delete
  wsDel.Range("1:1048576").Delete
  On Error GoTo ErrHdl

  'Check for Datastats folder
  If Dir(ThisWorkbook.Path & "\Datastats\", vbDirectory) = "" Then
    GLogger.fatal "Folder " & ThisWorkbook.Path & "\Datastats\" & " does not exist"
    wsW.Select
    Exit Sub
  End If

  'Read in Header info
  'Header files are supposed to be given in this format (header type in first column, subtype in second):
  'BAS DM  Fixed SwapSPEC                              OBJECT      TYPE            IDENTIFIER          NAME
BusDayRuleBUSD  BusDayRuleCONV  BusDayRuleRULE  CouponRatePERD  CouponGenENUM   CurrencyUNIT
DiscountCurveXREF   EffectiveDATE   AccrualDCBasisDAYC  LastRegCouponDATE   MaturityDATE
NotionalAtEndFLAG   OddFirstCouponENUM  PortfolioXREF   PositionUnitsVAL    ProductTypeENUM PayTermNB
PayTermUNIT NotionlAtStartFLAG  Company_SF_ClassificationNAME   Company_SF_Sub_ClassificationNAME
Company_SF_CounterpartyNAME Company_SF_RatingNAME   Company_SF_Attachment_PointNAME
Company_SF_Detachment_PointNAME Company_SF_TenureNAME   Company_Country_CodeNAME
Company_Counterparty_NameNAME   Company_Issuer_NameNAME Company_Level2_ClassificationNAME
Company_RatingNAME  Company_Industry_SectorNAME
  'rm_ro   DM Fixed SwapSPEC : Coupon List ATTRIBUTE   OBJECT      CouponListDATE  CouponListVAL
  'rm_ro   DM Fixed SwapSPEC : Variable Notional       ATTRIBUTE   OBJECT          VariabNotionalDATE
VariabNotionalVAL

  sFile = Dir(ThisWorkbook.Path & "\Header_Files\")
  Do While Len(sFile) > 0
    If FileLen(ThisWorkbook.Path & "\Header_Files\" & sFile) > 0 Then
      Application.StatusBar = "Read_Input: Reading header file " & sFile
      wsT.Range("1:1048576").Delete
      If Not doFileQuery(ThisWorkbook.Path & "\Header_Files\" & sFile, "Temp") Then
        GLogger.fatal "Reading " & sFile & " failed"
        wsW.Select
        Exit Sub
      End If
      wsT.Cells.EntireColumn.AutoFit
      i = 1
      Do While Not IsEmpty(wsT.Cells(i, 1))
        sHeader = wsT.Cells(i, 1).Text & ":" & wsT.Cells(i, 2).Text
        j = 1
        Do While Not IsEmpty(wsT.Cells(i, j))
          oHeader.Item(sHeader & ":" & j) = wsT.Cells(i, j).Text
          j = j + 1
        Loop
        oHeader.Item(sHeader & ":COLUMNS") = j - 1

        If wsT.Cells(i, 1).Text = "BAS" Then
          j = 5
          Do While Not IsEmpty(wsT.Cells(i, j))
```

```
            If wsT.Cells(i, j).Text = "IDENTIFIER" Then
              oSortCol.Item(wsT.Cells(i, 2).Text) = j
              Exit Do
            End If
            j = j + 1
          Loop
          If wsT.Cells(i, j).Text <> "IDENTIFIER" Then
            GLogger.fatal "BAS row " & i & " in header file " & sFile & " has no IDENTIFIER"
            wsW.Select
            Exit Sub
          End If
        ElseIf wsT.Cells(i, 1).Text = "rm_ro" Then
          j = 2
          Do While Not IsEmpty(wsT.Cells(i, j))
            If wsT.Cells(i, j).Text Like "*XREF" Or _
               wsT.Cells(i, j).Text Like "*Coupon List" Or _
               wsT.Cells(i, j).Text Like "*DATE" Or _
               wsT.Cells(i, j).Text Like "*0AXS" Or _
               wsT.Cells(i, j).Text Like "*NODE" Or _
               wsT.Cells(i, j).Text Like "*SEQ" Or _
               wsT.Cells(i, j).Text Like "*IDENTIFIER" Or _
               wsT.Cells(i, j).Text Like "*NB" Then
              oSortCol.Item(wsT.Cells(i, 2).Text) = j
              Exit Do
            End If
            j = j + 1
          Loop
          If CLng(oSortCol.Item(wsT.Cells(i, 2).Text)) = 0 Then
            If j = 6 And wsT.Cells(i, 5).Text Like "*VAL" Then
              oSortCol.Item(wsT.Cells(i, 2).Text) = 5
              GoTo Label_Next_Row_in_Headerfile
            Else
              oSortCol.Item(wsT.Cells(i, 2).Text) = 5
              GLogger.warn "rm_ro row " & i & " in header file " & sFile & _
                           " has no sort column id, column 5 taken"
            End If
          End If
        End If
Label_Next_Row_in_Headerfile:
        i = i + 1
      Loop
      wsT.QueryTables.Item(1).Delete
    Else
      GLogger.warn "Skipping zero file " & sFile
    End If
    sFile = Dir
  Loop

  'Check whether there is a FileSpecs file (a Sort Columns File for "non-Algo One" files!)
  'Expected file format (example, first row is constant title row):
  'Filename Suffix/Prefix   SortColumn_1 SortColumn_2 SortColumn_3
  'OB1_CASHFLOW             2            1            7
  'OB2_TRANS                1            4            7
  'OB7_LPIFWDS              2            5           16
  'SOL2_CREDIT_PROCESS_DATA 5

  sFileSpecs = ThisWorkbook.Path & "\Config\FileSpecs.csv"
  wsFS.Range("1:1048576").Delete
  If sFileSpecs = "" Then
    bFileSpecs = False
  ElseIf Len(Dir(sFileSpecs)) = 0 Then
    GLogger.info "File Specs file cannot be read. No individual file checks nor " & _
                 "previous period comparison"
    bFileSpecs = False
  ElseIf doFileQuery(sFileSpecs, "FileSpecs") Then
    bFileSpecs = True
    Set oFileSpecs = CreateObject("Scripting.Dictionary")
    i = 2
    Do While Not IsEmpty(wsFS.Cells(i, 1))
      j = 2
      Do While Not IsEmpty(wsFS.Cells(i, j))
        oFileSpecs.Item(wsFS.Cells(i, 1).Text & ":" & j) = CLng(wsFS.Cells(i, j))
        j = j + 1
      Loop
      i = i + 1
    Loop
  Else
    GLogger.fatal "Reading " & sFileSpecs & " failed"
    wsW.Select
    Exit Sub
  End If

  'Check whether there is a limits input file
  sLimitsIn = ThisWorkbook.Path & "\Config\Limits_Input.csv"
  lMaxOkValueCountPerField = wsP.Range("Max_Attr_Count_per_Field").Text
  wsLI.Range("1:1048576").Delete
  If sLimitsIn = "" Then
    bLimitsIn = False
  ElseIf Len(Dir(sLimitsIn)) = 0 Then
    GLogger.info "Input Limits file cannot be read. No input limits check"
```

```
    bLimitsIn = False
  ElseIf doFileQuery(sLimitsIn, "LimitsIn") Then
    If wsLI.Cells(1, 1).Text <> CLimFileVersion Then
      GLogger.fatal "Limits Input File version '" & CLimFileVersion & "' not compatible with " & _
                    "program version '" & wsLI.Cells(1, 1).Text & "'"
      wsW.Select
      Exit Sub
    End If
    bLimitsIn = True
    Set oLimitRow = CreateObject("Scripting.Dictionary")
    i = 3
    Do While Not IsEmpty(wsLI.Cells(i, limFilePrefix))
      oLimitRow.Item(wsLI.Cells(i, limFilePrefix).Text & ":" & _
                     wsLI.Cells(i, limField).Text) = i
      i = i + 1
    Loop
  Else
    GLogger.fatal "Reading " & sLimitsIn & " failed"
    wsW.Select
    Exit Sub
  End If

  'Check whether there is a limits move input file (containing movement limits over periods)
  sLimitsMoveIn = ThisWorkbook.Path & "\Config\Limits_Move_Input.csv"
  wsLMI.Range("1:1048576").Delete
  If sLimitsMoveIn = "" Then
    bLimitsMoveIn = False
  ElseIf Len(Dir(sLimitsMoveIn)) = 0 Then
    GLogger.info "Input Limits Move file cannot be read. No input limits move check"
    bLimitsMoveIn = False
  ElseIf doFileQuery(sLimitsMoveIn, "LimitsMoveIn") Then
    If wsLMI.Cells(1, 1).Text <> CLimMovFileVersion Then
      GLogger.fatal "Limits Move Input File version '" & _
                    CLimMovFileVersion & "' not compatible with " & _
                    "program version '" & wsLMI.Cells(1, 1).Text & "'"
      wsW.Select
      Exit Sub
    End If
    bLimitsMoveIn = True
    Set oLimitMoveRow = CreateObject("Scripting.Dictionary")
    i = 3
    Do While Not IsEmpty(wsLMI.Cells(i, limFilePrefix))
      oLimitMoveRow.Item(wsLMI.Cells(i, limFilePrefix).Text & ":" & _
                         wsLMI.Cells(i, limField).Text) = i
      i = i + 1
    Loop
  Else
    GLogger.fatal "Reading " & sLimitsMoveIn & " failed"
    wsW.Select
    Exit Sub
  End If

  'Read in data files
  sDirData = ThisWorkbook.Path & "\Data_Files\"
  sDirDataPrev = ThisWorkbook.Path & "\Data_Files_Prev\"

  dStdDevThreshold = wsP.Range("StdDevThreshold")

  wsNS.Range("1:1048576").Delete
  wsNS.Range(wsNS.Cells(1, outLbound + 1), _
             wsNS.Cells(1, outUbound - 1)).FormulaArray = _
                                                  Array("Filename", "RowType", _
                                                        "Fieldname", "Warning", "Min 1", _
                                                        "Min 2", "Min 3", "Average", _
                                                        "Max 3", "Max 2", "Max 1")
  lOut = 2

  wsNSM.Range("1:1048576").Delete
  wsNSM.Range(wsNSM.Cells(1, outLbound + 1), _
              wsNSM.Cells(1, outUbound - 1)).FormulaArray = _
                                                   Array("Filename", "RowType", _
                                                         "Fieldname", "Warning", "Min 1", _
                                                         "Min 2", "Min 3", "Average", _
                                                         "Max 3", "Max 2", "Max 1")
  lNSM = 2

  wsLO.Range("1:1048576").Delete
  wsLO.Cells(1, 1) = CLimFileVersion
  wsLO.Range(wsLO.Cells(2, limLbound + 1), _
             wsLO.Cells(2, limCharSet - 1)).FormulaArray = _
                                                  Array("Filename", "RowType", _
                                                        "Fieldname", "Allow Empty", _
                                                        "Min", "Max", "Min Date", "Max Date", _
                                                        "Length Min", "Length Max", _
                                                        "Formula", "OkFormat", "OkValues")
  For i = 0 To 255
    wsLO.Cells(2, limCharSet + i) = "'CHAR(" & Format(i, "000") & ") = '" & Chr(i) & "'"
  Next i
  wsLO.Cells(2, limCharSet + 10) = "'CHAR(010)"       'Otherwise you will face a surprise if you read in
the file
```

```
wsLO.Cells(2, limCharSet + 13) = "'CHAR(013)"
lLim = 3

wsLMO.Range("1:1048576").Delete
wsLMO.Cells(1, 1) = CLimMovFileVersion
wsLMO.Range(wsLMO.Cells(2, limLbound + 1), _
            wsLMO.Cells(2, limLenMax)).FormulaArray = _
                                                  Array("Filename", "RowType", "Fieldname", _
                                                        "Allow Empty", "Min", "Max", "Min Date", _
                                                        "Max Date", "Length Min", "Length Max")
lLMO = 3

wsTS.Range("1:1048576").Delete
wsTS.Range(wsTS.Cells(1, out2Lbound + 1), _
           wsTS.Cells(1, out2Ubound - 1)).FormulaArray = _
                                                    Array("Filename", "RowType", _
                                                          "Fieldname", "Warning", "Char", _
                                                          "Char Frequency", "Text Length", _
                                                          "Frequency", "Text", "Count")
lOut2 = 2

wsTSM.Range("1:1048576").Delete
wsTSM.Range(wsTSM.Cells(1, out2Lbound + 1), _
            wsTSM.Cells(1, out2Ubound - 1)).FormulaArray = _
                                                    Array("Filename", "RowType", _
                                                          "Fieldname", "Warning", "Char", _
                                                          "Char Frequency", "Text Length", _
                                                          "Frequency", "Text", "Count")
lTSM = 2

lNew = 1
lDel = 1

sFile = Dir(sDirData)
Do While Len(sFile) > 0
  oInputFiles.Item(sFile) = 1
  sFile = Dir()
Loop

'Find out common prefixes and suffixes of all (previous) data files
vPresuffix = Files_PrefixStart_SuffixEnd(sDirData)
vPresuffixPrev = Files_PrefixStart_SuffixEnd(sDirDataPrev)
'Kludge for "_YYYYMMDD.csv" resp. ".???" suffix
If Right(vPresuffix(3), 13) Like "_[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9].???" Then
  vPresuffix(3) = Right(vPresuffix(3), 13)
  vPresuffix(1) = -13
ElseIf Right(vPresuffix(3), 4) Like ".???" Then
  vPresuffix(3) = Right(vPresuffix(3), 4)
  vPresuffix(1) = -4
End If
If Right(vPresuffixPrev(3), 13) Like "_[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9].???" Then
  vPresuffixPrev(3) = Right(vPresuffixPrev(3), 13)
  vPresuffixPrev(1) = -13
ElseIf Right(vPresuffixPrev(3), 4) Like ".???" Then
  vPresuffixPrev(3) = Right(vPresuffixPrev(3), 4)
  vPresuffixPrev(1) = -4
End If
'Kludge for "YYYYMMDD_" prefix
If Right(vPresuffix(2), 9) Like "[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]_" Then
  vPresuffix(2) = Left(vPresuffix(2), 9)
  vPresuffix(0) = 9
End If
If Right(vPresuffixPrev(2), 9) Like "[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]_" Then
  vPresuffixPrev(2) = Left(vPresuffixPrev(2), 9)
  vPresuffixPrev(0) = 9
End If

For Each vFile In oInputFiles.keys()
  If FileLen(sDirData & vFile) > 0 Then
    Application.StatusBar = "Read_Input: Reading data file " & vFile
    'GLogger.info "Reading " & vFile
    wsT.Range("1:1048576").Delete
    If doFileQuery(sDirData & vFile, "Temp", sDelimiter) Then
      GLogger.info "Reading " & sDirData & vFile & " finished: " & _
                   wsT.Cells(1000000, 1).End(xlUp).Row & " rows read"
    Else
      GLogger.fatal "Reading " & sDirData & vFile & " failed"
      wsW.Select
      Exit Sub
    End If
    wsT.Cells.EntireColumn.AutoFit

    sCoreFile = Mid(vFile, vPresuffix(0) + 1, Len(vFile) + vPresuffix(1) - vPresuffix(0))

    'First collect all different kind of row types (BAS, RO, ...)
    i = 1
    sHeader = wsT.Cells(i, 1).Text & ":" & wsT.Cells(i, 2).Text & ":5"
    If oHeader.Item(sHeader) = "" Then
      'No header file - header info is in first row
      bHeaderFile = False
```

```
    j = 1
    Do While Not IsEmpty(wsT.Cells(1, j))
      oHeader.Item(CStr(j)) = wsT.Cells(1, j).Text
      j = j + 1
    Loop
    oHeader.Item(CStr(j)) = ""
    lColumns = j - 1
    lColumnStart = 1
    lRowTypes = 0
    bProcessThisRow = True
  Else
    'We do have a header file
    bHeaderFile = True
    lColumnStart = 5
    Set oRowTypes = CreateObject("Scripting.Dictionary")
    Do While Not IsEmpty(wsT.Cells(i, 1))
      oRowTypes.Item(wsT.Cells(i, 1).Text & ":" & wsT.Cells(i, 2).Text) = 1
      i = i + 1
    Loop
    lRowTypes = oRowTypes.Count - 1
  End If

  'Now iterate over a) row types, b) columns, and c) rows
  lMaxColumns = 0
  For k = 0 To lRowTypes
    If bHeaderFile Then
      lColumns = oHeader.Item(oRowTypes.keys()(k) & ":COLUMNS")
      If lColumns = 0 Then
        GLogger.fatal "File " & vFile & ", row type '" & _
                      oRowTypes.keys()(k) & "' not found in header"
        wsW.Select
        Exit Sub
      End If
    End If
    If lMaxColumns < lColumns Then lMaxColumns = lColumns
    For j = lColumnStart To lColumns
      Set oOkValues = CreateObject("Scripting.Dictionary")
      Set oData = CreateObject("Scripting.Dictionary")
      Set oStringLen = CreateObject("Scripting.Dictionary")
      Set oChar = CreateObject("Scripting.Dictionary")
      dSum = 0#
      sMsg = ""
      lCount = 0
      lUBounddStdDevInput = CInitialUBounddStdDevInput
      ReDim dStdDevInput(1 To lUBounddStdDevInput) As Double
      lCountAll = 0
      bIsDate = False
      For i = 1 To 3
        GdMax(i) = -1E+308
        GdMin(i) = 1E+308
      Next i
      GlMaxIdx = 0
      GlMinIdx = 0
      If bHeaderFile Then
        i = 1
      Else
        i = 2
      End If

      dMin = CMaxDouble
      dMax = CMinDouble
      dMinDate = CMaxDate
      dMaxDate = CMinDate
      lLenMin = CMaxLength
      lLenMax = CMinLength
      bHasEmpty = False

      If Not bNoSingleCharCheck Then
        ReDim bCharSet(0 To 255) As Boolean
      End If

      Do While Not (IsEmpty(wsT.Cells(i, 1)) And IsEmpty(wsT.Cells(i, 1).End(xlToRight)))
        bIsIDField = False
        If bHeaderFile Then
          sHeader = wsT.Cells(i, 1).Text & ":" & _
                    wsT.Cells(i, 2).Text
          bProcessThisRow = (sHeader = oRowTypes.keys()(k))
          sField = oHeader.Item(oRowTypes.keys()(k) & ":" & j)
        Else
          sField = oHeader.Item(CStr(j))
        End If
        If bProcessThisRow Then
          lCountAll = lCountAll + 1
          sCell = wsT.Cells(i, j).Text
          If sCell = "" Then
            bHasEmpty = True
            If bLimitsIn Then
              lLimRow = oLimitRow.Item(sCoreFile & ":" & sField)
              If lLimRow > 0 Then
                If Not CBool(wsLI.Cells(lLimRow, limAllowEmpty)) Then
```

```
          GLogger.warn "File " & vFile & ", Field " & _
                       sField & ", Row " & i & ", should not be empty"
        End If
      End If
    End If
  End If
  If Not bIsIDField Then
    If IsNumeric(sCell) Then
      dVal = CDbl(sCell)
      If dVal > dMax Then dMax = dVal
      If dVal < dMin Then dMin = dVal
      dSum = dSum + dVal
      lCount = lCount + 1
      dStdDevInput(lCount) = dVal
      Call Extreme(dVal)
      If bLimitsIn Then
        lLimRow = oLimitRow.Item(sCoreFile & ":" & sField)
        If lLimRow > 0 Then
          'Format check
          If Not IsEmpty(wsLI.Cells(lLimRow, limOkFormat)) Then
            oRegEx.Pattern = wsLI.Cells(lLimRow, limOkFormat).Text
            If Not oRegEx.Test(sCell) Then
              GLogger.warn "File " & vFile & ", Field " & sField & _
                           ", Row " & i & _
                           ", Value '" & sCell & "' not of format " & _
                           wsLI.Cells(lLimRow, limOkFormat).Text
            End If
          End If
          If Not IsEmpty(wsLI.Cells(lLimRow, limNumMin)) Then
            If dVal < wsLI.Cells(lLimRow, limNumMin) Then
              GLogger.warn "File " & vFile & ", Field " & _
                           sField & ", Row " & i & ", Value " & dVal & _
                           " < " & wsLI.Cells(lLimRow, limNumMin) & _
                           " [Min Limit]"
            End If
          End If
          If Not IsEmpty(wsLI.Cells(lLimRow, limNumMax)) Then
            If dVal > wsLI.Cells(lLimRow, limNumMax) Then
              GLogger.warn "File " & vFile & ", Field " & sField & _
                           ", Row " & i & ", Value " & dVal & " > " & _
                           wsLI.Cells(lLimRow, limNumMax) & " [Max Limit]"
            End If
          End If
        End If
      End If
    ElseIf IsDate(sCell) Then
      bIsDate = True
      dtDate = CDate(sCell)
      dVal = CDbl(dtDate)
      If dVal > dMaxDate Then dMaxDate = dVal
      If dVal < dMinDate Then dMinDate = dVal
      dSum = dSum + dVal
      lCount = lCount + 1
      dStdDevInput(lCount) = dVal
      Call Extreme(dVal)
      If bLimitsIn Then
        lLimRow = oLimitRow.Item(sCoreFile & ":" & sField)
        If lLimRow > 0 Then
          If Not IsEmpty(wsLI.Cells(lLimRow, limDateMin)) Then
            'Format check
            If Not IsEmpty(wsLI.Cells(lLimRow, limOkFormat)) Then
              oRegEx.Pattern = wsLI.Cells(lLimRow, limOkFormat).Text
              If Not oRegEx.Test(sCell) Then
                GLogger.warn "File " & vFile & ", Field " & sField & _
                             ", Row " & i & _
                             ", Value '" & sCell & "' not of format " & _
                             wsLI.Cells(lLimRow, limOkFormat).Text
              End If
            End If
            If dVal < CDbl(CDate(wsLI.Cells(lLimRow, limDateMin).Text)) Then
              GLogger.warn "File " & vFile & ", Field " & sField & _
                           ", Row " & i & ", Date " & Format(dVal, "dd-mmm-yyyy") & _
                           " < " & Format(wsLI.Cells(lLimRow, limDateMin), _
                                          "dd-mmm-yyyy") & " [Min Limit]"
            End If
          End If
          If Not IsEmpty(wsLI.Cells(lLimRow, limDateMax)) Then
            If dVal > CDbl(CDate(wsLI.Cells(lLimRow, limDateMax).Text)) Then
              GLogger.warn "File " & vFile & ", Field " & sField & _
                           ", Row " & i & ", Date " & Format(dVal, "dd-mmm-yyyy") & _
                           " > " & Format(wsLI.Cells(lLimRow, limDateMax), _
                                          "dd-mmm-yyyy") & " [Max Limit]"
            End If
          End If
        End If
      End If
    Else
      'String
      If sCell <> "" Then
        If oOkValues.Count <= lMaxOkValueCountPerField Then
```

```
            oOkValues.Item(sCell) = oOkValues.Item(sCell) + 1
          End If
        End If
        oData.Item("'" & sCell & "'") = _
                                        oData.Item("'" & sCell & "'") + 1
        lLen = Len(sCell)
        oStringLen.Item(CStr(lLen)) = oStringLen.Item(CStr(lLen)) + 1
        If lLen > 0 Then
          If lLen > lLenMax Then lLenMax = lLen
          If lLen < lLenMin Then lLenMin = lLen
          For m = 1 To lLen
            s = Mid(sCell, m, 1)
            If Not bNoSingleCharCheck Then bCharSet(Asc(s)) = True
            Sid = "CHAR(" & Format(Asc(s), "000") & ") = '" & s & "'"
            oChar.Item(Sid) = oChar.Item(Sid) + 1
          Next m
        End If
        If bLimitsIn Then
          lLimRow = oLimitRow.Item(sCoreFile & ":" & sField)
          If lLimRow > 0 Then
            If Not CBool(wsLI.Cells(lLimRow, limAllowEmpty)) And _
               Not IsEmpty(wsLI.Cells(lLimRow, limLenMin)) Then
              If lLen < wsLI.Cells(lLimRow, limLenMin).Value Then
                GLogger.warn "File " & vFile & ", Field " & sField & _
                             ", Row " & i & _
                             ", Length('" & sCell & "') = " & lLen & " < " & _
                             wsLI.Cells(lLimRow, limLenMin) & _
                             " [Min Length]"
              End If
            End If
            If Not IsEmpty(wsLI.Cells(lLimRow, limLenMax)) Then
              If lLen > wsLI.Cells(lLimRow, limLenMax).Value Then
                GLogger.warn "File " & vFile & ", Field " & sField & _
                             ", Row " & i & _
                             ", Length('" & sCell & "') = " & lLen & " > " & _
                             wsLI.Cells(lLimRow, limLenMax) & _
                             " [Max Length]"
              End If
            End If
            If Not bNoSingleCharCheck Then
              For m = 1 To lLen
                s = Mid(sCell, m, 1)
                If Not CBool(wsLI.Cells(lLimRow, limCharSet + Asc(s))) Then
                  GLogger.warn "File " & vFile & ", Field " & sField & _
                               ", Row " & i & _
                               ", Character " & m & " = '" & s & "' not legal"
                End If
              Next m
            End If
            If sCell <> "" Then
              'Format check
              If Not IsEmpty(wsLI.Cells(lLimRow, limOkFormat)) Then
                oRegEx.Pattern = wsLI.Cells(lLimRow, limOkFormat).Text
                If Not oRegEx.Test(sCell) Then
                  GLogger.warn "File " & vFile & ", Field " & sField & _
                               ", Row " & i & _
                               ", Value '" & sCell & "' not of format " & _
                               wsLI.Cells(lLimRow, limOkFormat).Text
                End If
              End If
              'Values check
              If Not IsEmpty(wsLI.Cells(lLimRow, limOkValues)) Then
                If InStr(CsAttrDelim & wsLI.Cells(lLimRow, limOkValues) & _
                         CsAttrDelim, sCell) = 0 Then
                  GLogger.warn "File " & vFile & ", Field " & sField & _
                               ", Row " & i & ", Value '" & sCell & _
                               "' not in Ok values list of limits file"
                End If
              End If
            End If                          'If sCell <> ""
          End If                            'If lLimRow > 0
        End If                              'If bLimitsIn
      End If                                'If IsNumeric(sCell) ... ElseIf ... Else ...
    End If                                  'If Not bIsIDField
  End If                                    'If bProcessThisRow
  i = i + 1
Loop

If bIsIDField Then
  wsTS.Cells(lOut2, out2File) = vFile
  If bHeaderFile Then
    wsTS.Cells(lOut2, out2Header) = oRowTypes.keys()(k)
    wsTS.Cells(lOut2, out2Field) = oHeader.Item(oRowTypes.keys()(k) & ":" & j)
  Else
    wsTS.Cells(lOut2, out2Field) = oHeader.Item(CStr(j))
  End If
  wsTS.Cells(lOut2, out2Warn) = "No stats - this field is an identifier field"
  lOut2 = lOut2 + 1
Else
  'Deal with numeric values
```

```
If lCount > 0 Then
  wsNS.Cells(lOut, outFile) = vFile
  If bHeaderFile Then
    wsNS.Cells(lOut, outHeader) = oRowTypes.keys()(k)
  End If
  wsNS.Cells(lOut, outField) = sField
  wsNS.Cells(lOut, outAverage) = dSum / lCount
  ReDim Preserve dStdDevInput(1 To lCount) As Double  'Reduce dStdDevInput size to used values
  If lCountAll > lCount Then
    sMsg = sMsg & lCountAll - lCount & " values of " & lCountAll & " are non-numeric. "
  End If
  d = 0#
  On Error Resume Next
  d = Application.WorksheetFunction.StDevP(dStdDevInput)
  On Error GoTo ErrHdl
  If d > 0# Then
    If Abs(dSum / lCount - GdMin(1)) > dStdDevThreshold * d Then
      sMsg = sMsg & "Min is off by more than " & dStdDevThreshold & " stdev. "
    End If
    If Abs(dSum / lCount - GdMax(1)) > dStdDevThreshold * d Then
      sMsg = sMsg & "Max is off by more than " & dStdDevThreshold & " stdev. "
    End If
  End If
  wsNS.Cells(lOut, outWarn) = sMsg
  sMsg = ""
  If GlMinIdx > 2 Then wsNS.Cells(lOut, outMin1) = GdMin(1)
  If GlMinIdx > 1 Then wsNS.Cells(lOut, outMin2) = GdMin(2 + (GlMinIdx < 3))
  If GlMinIdx > 0 Then wsNS.Cells(lOut, outMin3) = GdMin(3 + (GlMinIdx < 3) + (GlMinIdx < 2))
  If GlMaxIdx > 2 Then wsNS.Cells(lOut, outMax1) = GdMax(1)
  If GlMaxIdx > 1 Then wsNS.Cells(lOut, outMax2) = GdMax(2 + (GlMaxIdx < 3))
  If GlMaxIdx > 0 Then wsNS.Cells(lOut, outMax3) = GdMax(3 + (GlMaxIdx < 3) + (GlMaxIdx < 2))
  If bIsDate Then
    Range(wsNS.Cells(lOut, outMin1), wsNS.Cells(lOut, outMax1)).NumberFormat = "dd-mmm-yyyy"
  End If
  lOut = lOut + 1
End If

'Deal with string values
If lCount < lCountAll Then
  wsTS.Cells(lOut2, out2File) = vFile
  If bHeaderFile Then
    wsTS.Cells(lOut2, out2Header) = oRowTypes.keys()(k)
    wsTS.Cells(lOut2, out2Field) = oHeader.Item(oRowTypes.keys()(k) & ":" & j)
  Else
    wsTS.Cells(lOut2, out2Field) = oHeader.Item(CStr(j))
  End If
  If lCount > 0 Then
    sMsg = sMsg & lCount & " values of " & lCountAll & " are numeric. "
  End If
  If oData.Count > lCountAll - 5 Or (oData.Count / lCountAll > 0.5 And _
                                     lCountAll > 10) Then
    For m = oData.Count - 1 To 0 Step -1
      If oData.keys()(m) <> "'" & "'" And oData.keys()(m) <> "' " & "'" _
         And oData.Item(oData.keys()(m)) = 1 Then
        oData.Remove oData.keys()(m)
      End If
    Next m
    sMsg = sMsg & lCountAll - oData.Count & " strings occurring once not shown. "
  End If
  wsTS.Cells(lOut2, out2Warn) = sMsg
  sMsg = ""
  If oData.Count > 0 Then
    On Error Resume Next
    wsTS.Cells(lOut2, _
               out2Text).Resize(oData.Count, 2) = _
                                                  Application.WorksheetFunction.Transpose( _
                                                    Array(oData.keys, oData.items))
    On Error GoTo ErrHdl
  End If
  If oStringLen.Count > 0 Then
    On Error Resume Next
    wsTS.Cells(lOut2, _
               out2Len).Resize(oStringLen.Count, 2) = _
                                                  Application.WorksheetFunction.Transpose( _
                                                  Array(oStringLen.keys, oStringLen.items))
    On Error GoTo ErrHdl
  End If

  'Now print rarest occurring character
  lMin = 1000000
  Sid = ""
  For m = 0 To oChar.Count - 1
    If oChar.items()(m) < lMin Then
      lMin = oChar.items()(m)
      Sid = oChar.keys()(m)
    End If
  Next m
  wsTS.Cells(lOut2, out2Char) = Sid
  If lMin < 1000000 Then wsTS.Cells(lOut2, out2CharCount) = lMin
```

```
        'Increase row counter
        lMax = 1
        If lMax < oData.Count Then lMax = oData.Count
        If lMax < oStringLen.Count Then lMax = oStringLen.Count
        lOut2 = lOut2 + lMax
      End If

      'Update Limits Out sheet
      wsLO.Cells(lLim, limFilePrefix) = sCoreFile
      If bHeaderFile Then
        wsLO.Cells(lLim, limHeader) = oRowTypes.keys()(k)
        wsLO.Cells(lLim, limField) = oHeader.Item(oRowTypes.keys()(k) & ":" & j)
      Else
        wsLO.Cells(lLim, limField) = oHeader.Item(CStr(j))
      End If
      If bHasEmpty Then wsLO.Cells(lLim, limAllowEmpty) = 1  'True
      If dMin < CMaxDouble Then wsLO.Cells(lLim, limNumMin) = dMin
      If dMax > CMinDouble Then wsLO.Cells(lLim, limNumMax) = dMax
      If dMinDate < CMaxDate Then
        wsLO.Cells(lLim, limDateMin) = dMinDate
        wsLO.Cells(lLim, limDateMin).NumberFormat = "dd-mmm-yyyy"
      End If
      If dMaxDate > CMinDate Then
        wsLO.Cells(lLim, limDateMax) = dMaxDate
        wsLO.Cells(lLim, limDateMax).NumberFormat = "dd-mmm-yyyy"
      End If
      If lLenMin < CMaxLength Then wsLO.Cells(lLim, limLenMin) = lLenMin
      If lLenMax > CMinLength Then wsLO.Cells(lLim, limLenMax) = lLenMax
      If oOkValues.Count > 0 And oOkValues.Count <= lMaxOkValueCountPerField Then
        wsLO.Cells(lLim, limOkValues) = sbCat(Array(oOkValues.keys)(0), CsAttrDelim)
      End If
      'Now we protect "True" from becoming "TRUE" etc. if reloaded from csv later
      If Not IsEmpty(wsLO.Cells(lLim, limOkValues)) And _
         Right(wsLO.Cells(lLim, limOkValues), 1) <> CsAttrDelim Then
        wsLO.Cells(lLim, limOkValues) = wsLO.Cells(lLim, limOkValues) & CsAttrDelim
      End If
      If Not bNoSingleCharCheck Then
        For m = 0 To 255
          If bCharSet(m) Then wsLO.Cells(lLim, limCharSet + m) = True
        Next m
      End If
      lLim = lLim + 1

    End If                                          'If bIsIDField

    Set oChar = Nothing
    Set oData = Nothing
    Set oOkValues = Nothing
    Set oStringLen = Nothing

  Next j

Next k

'Now formula checks - we need to iterate through rows first, then columns
If bHeaderFile Then
  i = 1
  lColumnStart = 5
Else
  i = 2
  lColumnStart = 1
End If
Do While Not (IsEmpty(wsT.Cells(i, 1)) And IsEmpty(wsT.Cells(i, 1).End(xlToRight)))
  If bHeaderFile Then
    lColumns = oHeader.Item(wsT.Cells(i, 1).Text & ":" & wsT.Cells(i, 2).Text & ":COLUMNS")
    Debug.Assert lColumns > 0
    If wsT.Cells(i, 1).Text = "BAS" Then
      Set oBASRow = Nothing
      Set oBASRow = CreateObject("Scripting.Dictionary")
      For j = lColumnStart To lColumns
        oBASRow.Item(oHeader.Item(sHeader & ":" & j)) = wsT.Cells(i, j).Text
      Next j
    End If
  Else
    Set oBASRow = Nothing
    Set oBASRow = CreateObject("Scripting.Dictionary")
    For j = lColumnStart To lColumns
      oBASRow.Item(CStr(oHeader.Item(CStr(j)))) = wsT.Cells(i, j).Text
    Next j
  End If
  For j = lColumnStart To lColumns
    'Formula check
    If bHeaderFile Then
      sHeader = wsT.Cells(i, 1).Text & ":" & wsT.Cells(i, 2).Text
      sField = oHeader.Item(sHeader & ":" & j)
    Else
      sField = oHeader.Item(CStr(j))
    End If
    If bLimitsIn Then
      lLimRow = oLimitRow.Item(sCoreFile & ":" & sField)
```

```
      If lLimRow > 0 Then
        If Not IsEmpty(wsLI.Cells(lLimRow, limFormula)) Then
          oBASRow.Item("ThisCell") = wsT.Cells(i, j).Text
          'Parameter substitution in formula
          sFormula = wsLI.Cells(lLimRow, limFormula).Text
          m = InStr(sFormula, "[")
          Do While m > 0
            k = InStr(sFormula, "]")
            If k <= m Then
              GLogger.fatal "File " & vFile & ", Field " & sField & _
                            ", Row " & i & ", Formula '" & _
                            wsLI.Cells(lLimRow, limFormula).Text & _
                            "' has nonmatching parameter brackets []"
              wsW.Select
              Exit Sub
            End If
            s = Mid(sFormula, m + 1, k - m - 1)
            If oBASRow.exists(s) Then
              sFormula = Replace(sFormula, "[" & s & "]", """" & oBASRow.Item(s) & """")
            Else
              GLogger.fatal "File " & vFile & ", Field " & sField & _
                            ", Row " & i & ", Formula '" & _
                            wsLI.Cells(lLimRow, limFormula).Text & _
                            "', Param [" & s & "] is not a valid field name"
              wsW.Select
              Exit Sub
            End If
            m = InStr(sFormula, "[")
          Loop
          'Formula evaluation
          On Error Resume Next
          If Not Application.Evaluate(sFormula) Then
            If Err.Number <> 0 Then
              GLogger.fatal "File " & vFile & ", Field " & sField & _
                            ", Row " & i & _
                            ", syntax error " & Err.Number & " in Limits formula '" & _
                            wsLI.Cells(lLimRow, limFormula).Text & "' = '" & sFormula & "'"
              GLogger.fatal "Please note: Application.International(xlMDY) = " & _
                            Application.International(xlMDY)
              On Error GoTo 0
              wsW.Select
              Exit Sub
            Else
              On Error GoTo ErrHdl
              GLogger.warn "File " & vFile & ", Field " & sField & _
                           ", Row " & i & _
                           ", Value '" & wsT.Cells(i, j).Text & "' does not pass formula '" & _
                           wsLI.Cells(lLimRow, limFormula).Text & "' = '" & sFormula & "'"
            End If
          End If
          On Error GoTo ErrHdl
        End If
      End If
    End If
  Next j
  i = i + 1
Loop

wsTM.Range("1:1048576").Delete

'Movement Checks
bMoveCheck = True
If bMoveCheck Then
  sFilePrev = Dir(sDirDataPrev & vPresuffixPrev(2) & sCoreFile & vPresuffixPrev(3))
  If sFilePrev = "" Then
    bMoveCheck = False
    GLogger.warn "Filename " & sDirDataPrev & vPresuffixPrev(2) & sCoreFile & _
                 vPresuffixPrev(3) & " not found in Data_Files_Prev. No movement check."
  End If
End If
If bMoveCheck Then
  If FileLen(sDirDataPrev & sFilePrev) > 0 Then
    Application.StatusBar = "Read_Input: Reading previous data file " & sFilePrev
    wsTP.Range("1:1048576").Delete
    If doFileQuery(sDirDataPrev & sFilePrev, wsTP.Name, sDelimiter) Then
      GLogger.info "Reading " & sDirDataPrev & sFilePrev & " finished: " & _
                   wsTP.Cells(1000000, 1).End(xlUp).Row & " rows read"
    Else
      GLogger.fatal "Reading " & sDirDataPrev & sFilePrev & " failed"
      wsW.Select
      Exit Sub
    End If
    wsTP.Cells.EntireColumn.AutoFit
    bDeletedRecords = False
    bNewRecords = False
    'Prepare both temp sheets for comparison
    'Temp sheet first
    lMainSortCol = lMaxColumns + 1
    If bHeaderFile Then
      Sid = ""
```

```
    sCellPrev = ":"
    i = 1
    Do While Not (IsEmpty(wsT.Cells(i, 2)) And IsEmpty(wsT.Cells(i, 2).End(xlToRight)))
      sCell = wsT.Cells(i, 2)
      If InStr(sCell, ":") = 0 Then
        Sid = wsT.Cells(i, CLng(oSortCol.Item(wsT.Cells(i, 2).Text))).Text
        sID2 = "_BAS_"
        sID3 = "_BAS_"
      ElseIf sCell <> sCellPrev Then
        sID2 = wsT.Cells(i, 3)
        sID3 = wsT.Cells(i, CLng(oSortCol.Item(wsT.Cells(i, 2).Text))).Text
      Else
        sID3 = wsT.Cells(i, CLng(oSortCol.Item(wsT.Cells(i, 2).Text))).Text
      End If
      wsT.Cells(i, lMainSortCol) = Sid & "|" & sID2 & "|" & sID3
      sCellPrev = sCell
      i = i + 1
    Loop

    wsT.Sort.SortFields.Clear
    wsT.Sort.SortFields.Add _
        Key:=Range(wsT.Cells(1, lMainSortCol), _
                   wsT.Cells(i, lMainSortCol)), _
        SortOn:=xlSortOnValues, _
        Order:=xlAscending, DataOption:=xlSortNormal
    With wsT.Sort
      .SetRange Range(wsT.Cells(1, 1), wsT.Cells(i, lMainSortCol))
      .Header = xlNo
      .Apply
    End With

    'Now TempPrev sheet
    Sid = ""
    sCellPrev = ":"
    i = 1
    Do While Not (IsEmpty(wsTP.Cells(i, 2)) And IsEmpty(wsTP.Cells(i, 2).End(xlToRight)))
      sCell = wsTP.Cells(i, 2)
      If InStr(sCell, ":") = 0 Then
        Sid = wsTP.Cells(i, CLng(oSortCol.Item(wsTP.Cells(i, 2).Text))).Text
        sID2 = "_BAS_"
        sID3 = "_BAS_"
      ElseIf sCell <> sCellPrev Then
        sID2 = wsTP.Cells(i, 3)
        sID3 = wsTP.Cells(i, CLng(oSortCol.Item(wsTP.Cells(i, 2).Text))).Text
      Else
        sID2 = wsTP.Cells(i, 3)
        sID3 = wsTP.Cells(i, CLng(oSortCol.Item(wsTP.Cells(i, 2).Text))).Text
      End If
      wsTP.Cells(i, lMainSortCol) = Sid & "|" & sID2 & "|" & sID3
      sCellPrev = sCell
      i = i + 1
    Loop
    wsT.Cells.EntireColumn.AutoFit

    wsTP.Sort.SortFields.Clear
    wsTP.Sort.SortFields.Add Key:=Range(wsTP.Cells(1, lMainSortCol), _
                                        wsTP.Cells(i, lMainSortCol)), _
                             SortOn:=xlSortOnValues, Order:=xlAscending, _
                             DataOption:=xlSortNormal
    With wsTP.Sort
      .SetRange Range(wsTP.Cells(1, 1), wsTP.Cells(i, lMainSortCol))
      .Header = xlNo
      .Apply
    End With
    wsTP.Cells.EntireColumn.AutoFit
    For i = 0 To lRowTypes
      For j = 1 To oHeader.Item(oRowTypes.keys()(i) & ":COLUMNS")
        wsTM.Cells(i + 1, j) = oHeader.Item(oRowTypes.keys()(i) & ":" & j)
      Next j
    Next i
    lTM = i + 1
  Else
    'Check whether field names of "non-Algo One" files have changed - yes: error!
    j = 1
    bFatal = False
    Do While Not IsEmpty(wsTP.Cells(1, j))
      If oHeader.Item(CStr(j)) <> wsTP.Cells(1, j).Text Then
        bFatal = True
        GLogger.fatal "Name of field " & j & " has changed: " & vFile & _
                      "=""" & oHeader.Item(CStr(j)) & """, " & sFilePrev & "=""" & _
                      wsTP.Cells(1, j).Text & """"
      End If
      j = j + 1
    Loop
    If bFatal Then
      wsW.Select
      Exit Sub
    End If

    i = 2
```

```
            Do While Not (IsEmpty(wsT.Cells(i, 1)) And IsEmpty(wsT.Cells(i, 1).End(xlToRight)))
              If Not bFileSpecs Then
                GLogger.fatal "Please provide FileSpecs.csv file and specify main sort column " & _
                              "for file " & vFile
                wsW.Select
                Exit Sub
              End If
              If Not oFileSpecs.exists(sCoreFile & ":2") Then
                GLogger.fatal "Please specify main sort column for file " & vFile & _
                              " in Config file " & sFileSpecs
                wsW.Select
                End
              End If
              Sid = wsT.Cells(i, CLng(oFileSpecs.Item(sCoreFile & ":2")))
              sID2 = ""
              If CLng(oFileSpecs.Item(sCoreFile & ":3")) > 0 Then
                sID2 = wsT.Cells(i, CLng(oFileSpecs.Item(sCoreFile & ":3")))
              End If
              sID3 = ""
              If CLng(oFileSpecs.Item(sCoreFile & ":4")) > 0 Then
                sID3 = wsT.Cells(i, CLng(oFileSpecs.Item(sCoreFile & ":4")))
              End If
              wsT.Cells(i, lMainSortCol) = Sid & "|" & sID2 & "|" & sID3
              i = i + 1
            Loop

            wsT.Sort.SortFields.Clear
            wsT.Sort.SortFields.Add Key:=wsT.Range(wsT.Cells(2, lMainSortCol), _
                                                   wsT.Cells(i - 1, lMainSortCol)), _
                                    SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
            With wsT.Sort
              .SetRange Range(wsT.Cells(2, 1), wsT.Cells(i - 1, lMainSortCol))
              .Header = xlNo
              .Apply
            End With

            i = 2
            Do While Not (IsEmpty(wsTP.Cells(i, 1)) And IsEmpty(wsTP.Cells(i, 1).End(xlToRight)))
              Sid = wsTP.Cells(i, CLng(oFileSpecs.Item(sCoreFile & ":2")))
              sID2 = ""
              If CLng(oFileSpecs.Item(sCoreFile & ":3")) > 0 Then
                sID2 = wsTP.Cells(i, CLng(oFileSpecs.Item(sCoreFile & ":3")))
              End If
              sID3 = ""
              If CLng(oFileSpecs.Item(sCoreFile & ":4")) > 0 Then
                sID3 = wsTP.Cells(i, CLng(oFileSpecs.Item(sCoreFile & ":4")))
              End If
              wsTP.Cells(i, lMainSortCol) = Sid & "|" & sID2 & "|" & sID3
              i = i + 1
            Loop
            wsTP.Sort.SortFields.Clear
            wsTP.Sort.SortFields.Add Key:=Range(wsTP.Cells(2, lMainSortCol), _
                                                wsTP.Cells(i - 1, lMainSortCol)), _
                                     SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
            With wsTP.Sort
              .SetRange Range(wsTP.Cells(2, 1), wsTP.Cells(i - 1, lMainSortCol))
              .Header = xlNo
              .Apply
            End With
            wsTM.Range(wsTM.Cells(1, 1), _
                       wsTM.Cells(1, lColumns)).FormulaArray = _
                                                               Array(wsT.Range(wsT.Cells(1, 1), _
                                                                               wsT.Cells(1, lColumns)))
            lTM = 2
          End If

          'Now walk through sorted input files

          If bHeaderFile Then
            i = 1
            j = 1
          Else
            i = 2
            j = 2
          End If
Label_Start:
          If bHeaderFile Then
            'See that both Temp and TempPrev are on BAS

            'First Temp
            k = i
            Do While Not (IsEmpty(wsT.Cells(i, 1)) And _
                          IsEmpty(wsT.Cells(i, 1).End(xlToRight))) And wsT.Cells(i, 1) <> "BAS"
              i = i + 1
            Loop

            'Now TempPrev
            k = j
            Do While Not (IsEmpty(wsTP.Cells(j, 1)) And _
                          IsEmpty(wsTP.Cells(j, 1).End(xlToRight))) And wsTP.Cells(j, 1) <> "BAS"
```

```
    j = j + 1
  Loop
End If

Do While Not (IsEmpty(wsT.Cells(i, 1)) And IsEmpty(wsT.Cells(i, 1).End(xlToRight)) And _
              IsEmpty(wsTP.Cells(j, 1)) And IsEmpty(wsTP.Cells(j, 1).End(xlToRight)))
  k = lMainSortCol
  Debug.Assert k > 0
  If Not IsEmpty(wsTP.Cells(j, 1)) And _
     (IsEmpty(wsT.Cells(i, 1)) Or _
      StrComp(wsT.Cells(i, k).Text, wsTP.Cells(j, k).Text, vbTextCompare) = 1) Then
    'Copy old record from previous file into Deleted tab
    If bHeaderFile Then
      s = wsTP.Cells(j, 1)
      Do
        If wsTP.Cells(j, 1) = "BAS" Then
          lColumns = oHeader.Item(wsTP.Cells(j, 1).Text & ":" & _
                                  wsTP.Cells(j, 2).Text & ":COLUMNS")
          If Not bDeletedRecords Then
            wsDel.Cells(lDel, 1) = "FileName"
            For m = 1 To lColumns
              wsDel.Cells(lDel, m + 1) = oHeader.Item(wsTP.Cells(j, _
                                         1).Text & ":" & wsTP.Cells(j, 2).Text & ":" & m)
            Next m
            bDeletedRecords = True
            lDel = lDel + 1
          End If
          'Copy only BAS row
          wsDel.Cells(lDel, 1) = vFile
          On Error Resume Next
          wsDel.Range(wsDel.Cells(lDel, 2), _
                      wsDel.Cells(lDel, lColumns + 1)).FormulaArray = _
                                    Array(wsTP.Range(wsTP.Cells(j, 1), wsTP.Cells(j, lColumns)))
          On Error GoTo ErrHdl
          lDel = lDel + 1
        End If
        j = j + 1
      Loop Until wsTP.Cells(j, 1) = "" Or s = wsTP.Cells(j, 1)
    Else
      If Not bDeletedRecords Then
        wsDel.Cells(lDel, 1) = "FileName"
        wsDel.Range(wsDel.Cells(lDel, 2), _
                    wsDel.Cells(lDel, lColumns + 1)).FormulaArray = _
                                  Array(wsTP.Range(wsTP.Cells(1, 1), wsTP.Cells(1, lColumns)))
        bDeletedRecords = True
        lDel = lDel + 1
      End If
      wsDel.Cells(lDel, 1) = vFile
      wsDel.Range(wsDel.Cells(lDel, 2), _
                  wsDel.Cells(lDel, lColumns + 1)).FormulaArray = _
                                  Array(wsTP.Range(wsTP.Cells(j, 1), wsTP.Cells(j, lColumns)))
      lDel = lDel + 1
      j = j + 1
    End If
  ElseIf Not IsEmpty(wsT.Cells(i, 1)) And _
         (IsEmpty(wsTP.Cells(j, 1)) Or _
          StrComp(wsT.Cells(i, k).Text, wsTP.Cells(j, k).Text, vbTextCompare) = -1) Then
    'Copy new record from current file into New tab
    If bHeaderFile Then
      s = wsT.Cells(i, 1)
      Do
        If wsT.Cells(i, 1) = "BAS" Then
          lColumns = oHeader.Item(wsT.Cells(i, 1).Text & ":" & _
                                  wsT.Cells(i, 2).Text & ":COLUMNS")
          If Not bNewRecords Then
            wsN.Cells(lNew, 1) = "FileName"
            For m = 1 To lColumns
              wsN.Cells(lNew, m + 1) = oHeader.Item(wsT.Cells(i, 1).Text & _
                                                    ":" & wsT.Cells(i, 2).Text & ":" & m)
            Next m
            bNewRecords = True
            lNew = lNew + 1
          End If
          'Copy only BAS row
          wsN.Cells(lNew, 1) = vFile
          On Error Resume Next
          wsN.Range(wsN.Cells(lNew, 2), _
                    wsN.Cells(lNew, _
                              lColumns + 1)).FormulaArray = _
                    Array(wsT.Range(wsT.Cells(i, 1), wsT.Cells(i, lColumns)))
          On Error GoTo ErrHdl
          lNew = lNew + 1
        End If
        i = i + 1
      Loop Until wsT.Cells(i, 1) = "" Or s = wsT.Cells(i, 1)
    Else
      If Not bNewRecords Then
        wsN.Cells(lNew, 1) = "FileName"
        wsN.Range(wsN.Cells(lNew, 2), _
                  wsN.Cells(lNew, _
```

```
                                    lColumns + 1)).FormulaArray = _
                                    Array(wsT.Range(wsT.Cells(1, 1), wsT.Cells(1, lColumns)))
                  bNewRecords = True
                  lNew = lNew + 1
                End If
                wsN.Cells(lNew, 1) = vFile
                On Error Resume Next
                wsN.Range(wsN.Cells(lNew, 2), _
                          wsN.Cells(lNew, _
                                    lColumns + 1)).FormulaArray = _
                                    Array(wsT.Range(wsT.Cells(i, 1), wsT.Cells(i, lColumns)))
                On Error GoTo ErrHdl
                i = i + 1
                lNew = lNew + 1
              End If
            Else
              'Copy same or changed record into TempMove tab
              If bHeaderFile Then
                If wsT.Cells(i, 1) <> "BAS" Or wsTP.Cells(j, 1) <> "BAS" Then
                  GLogger.fatal "Internal error. BAS rows expected for both " & vFile & _
                                ", row " & i & " and " & sFilePrev & ", row " & j
                  Stop
                  Exit Sub
                End If
                'We implement a finite automaton = state engine, _
                 starting with two BAS rows with same Id
                Do
Label_State_BAS_BAS:
                  lColumns = oHeader.Item(wsT.Cells(i, 1).Text & ":" & _
                                          wsT.Cells(i, 2).Text & ":COLUMNS")
                  For k = 1 To lColumns
                    sCell = wsT.Cells(i, k).Text
                    If k = 1 Or k = 2 Or _
                       k = CLng(oSortCol.Item(wsT.Cells(i, 2).Text)) Then
                      wsTM.Cells(lTM, k) = sCell
                    Else
                      sCellPrev = wsTP.Cells(j, k).Text
                      If IsNumeric(sCell) And IsNumeric(sCellPrev) Then
                        wsTM.Cells(lTM, k) = CDbl(sCell) - CDbl(sCellPrev)
                      ElseIf IsDate(sCell) And IsDate(sCellPrev) Then
                        wsTM.Cells(lTM, k) = CDate(sCell) - CDate(sCellPrev)
                      Else
                        If sCell = "" Then sCell = "''"
                        If sCellPrev = "" Then sCellPrev = "''"
                        If sCell = " " Then sCell = "' '"
                        If sCellPrev = " " Then sCellPrev = "' '"
                        If sCell = sCellPrev Then
                          wsTM.Cells(lTM, k) = "="
                        Else
                          wsTM.Cells(lTM, k) = sCellPrev & "->" & sCell
                        End If
                      End If
                    End If
                  Next k
                  lTM = lTM + 1
                  i = i + 1
                  j = j + 1

                  'Now state transition: decide to which state to switch to
                  If wsT.Cells(i, 1) = "BAS" And wsTP.Cells(j, 1) = "BAS" Then
                    GoTo Label_Start
                  ElseIf wsT.Cells(i, 1) = "BAS" Then
                    k = j
                    Do While Not (IsEmpty(wsTP.Cells(j, 1)) And _
                                  IsEmpty(wsTP.Cells(j, 1).End(xlToRight))) And wsTP.Cells(j, 1) <> "BAS"
                      j = j + 1
                    Loop
                    GoTo Label_Start
                  ElseIf wsTP.Cells(j, 1) = "BAS" Then
                    k = i
                    Do While Not (IsEmpty(wsT.Cells(i, 1)) And _
                                  IsEmpty(wsT.Cells(i, 1).End(xlToRight))) And wsT.Cells(i, 1) <> "BAS"
                      i = i + 1
                    Loop
                    GoTo Label_Start
                  Else
                    sID2 = wsT.Cells(i, 3)
                    sID3 = wsTP.Cells(j, 3)
Label_Sublevel_Same_BAS:
                    If sID2 = sID3 Then
                      If sID2 = "" Then GoTo Label_Start
                      If wsT.Cells(i, _
                                   CLng(oSortCol.Item(wsT.Cells(i, _
                                   2).Text))).Text = wsTP.Cells(j, _
                                   CLng(oSortCol.Item(wsT.Cells(i, 2).Text))).Text Then
                        For k = 1 To lColumns
                          sCell = wsT.Cells(i, k).Text
                          If k = 1 Or k = 2 Or k = _
                             CLng(oSortCol.Item(wsT.Cells(i, 2).Text)) Then
                            wsTM.Cells(lTM, k) = sCell
```

```
      Else
        sCellPrev = wsTP.Cells(j, k).Text
        If IsNumeric(sCell) And IsNumeric(sCellPrev) Then
          wsTM.Cells(lTM, k) = CDbl(sCell) - CDbl(sCellPrev)
        ElseIf IsDate(sCell) And IsDate(sCellPrev) Then
          wsTM.Cells(lTM, k) = CDate(sCell) - CDate(sCellPrev)
        Else
          If sCell = "" Then sCell = "''"
          If sCellPrev = "" Then sCellPrev = "''"
          If sCell = " " Then sCell = "' '"
          If sCellPrev = " " Then sCellPrev = "' '"
          If sCell = sCellPrev Then
            wsTM.Cells(lTM, k) = "="
          Else
            wsTM.Cells(lTM, k) = sCellPrev & "->" & sCell
          End If
        End If
      End If
    Next k
  ElseIf wsT.Cells(i, _
                   CLng(oSortCol.Item(wsT.Cells(i, _
                   2).Text))).Text > wsTP.Cells(j, _
                   CLng(oSortCol.Item(wsT.Cells(i, 2).Text))).Text Then
    k = j
    'Only evaluate conditions until we face FALSE
    b = Not IsEmpty(wsTP.Cells(j, 1))
    If b Then
      b = b And (wsTP.Cells(j, 1) <> "BAS")
      If b Then
        b = b And (sID2 = sID3)
        If b Then
          b = b And wsT.Cells(i, _
                   CLng(oSortCol.Item(wsT.Cells(i, _
                   2).Text))).Text > wsTP.Cells(j, _
                   CLng(oSortCol.Item(wsT.Cells(i, _
                   2).Text))).Text
        End If
      End If
    End If
    Do While b
      j = j + 1
      sID3 = wsTP.Cells(j, 3)
      'Only evaluate conditions until we face FALSE
      b = Not IsEmpty(wsTP.Cells(j, 1))
      If b Then
        b = b And (wsTP.Cells(j, 1) <> "BAS")
        If b Then
          b = b And (sID2 = sID3)
          If b Then
            b = b And wsT.Cells(i, _
                              CLng(oSortCol.Item(wsT.Cells(i, _
                              2).Text))).Text > wsTP.Cells(j, _
                              CLng(oSortCol.Item(wsT.Cells(i, _
                              2).Text))).Text
          End If
        End If
      End If
    Loop
    If IsEmpty(wsTP.Cells(j, 1)) Or _
       wsTP.Cells(j, 1) = "BAS" Then
      GoTo Label_Start
    Else
      GoTo Label_Sublevel_Same_BAS
    End If
  Else
    k = i
    'Only evaluate conditions until we face FALSE
    b = Not IsEmpty(wsT.Cells(i, 1))
    If b Then
      b = b And (wsT.Cells(i, 1) <> "BAS")
      If b Then
        b = b And (sID2 = sID3)
        If b Then
          b = b And wsT.Cells(i, CLng(oSortCol.Item(wsT.Cells(i, _
                   2).Text))).Text < wsTP.Cells(j, CLng(oSortCol.Item(wsT.Cells(i, _
                   2).Text))).Text
        End If
      End If
    End If
    Do While b
      i = i + 1
      sID2 = wsT.Cells(i, 3)
      'Only evaluate conditions until we face FALSE
      b = Not IsEmpty(wsT.Cells(i, 1))
      If b Then
        b = b And (wsT.Cells(i, 1) <> "BAS")
        If b Then
          b = b And (sID2 = sID3)
          If b Then
            b = b And wsT.Cells(i, _
```

```
                                        CLng(oSortCol.Item(wsT.Cells(i, _
                                        2).Text))).Text < wsTP.Cells(j, _
                                        CLng(oSortCol.Item(wsT.Cells(i, _
                                        2).Text))).Text
                    End If
                  End If
                End If
              Loop
              If IsEmpty(wsT.Cells(i, 1)) Or wsT.Cells(i, 1) = "BAS" Then
                GoTo Label_Start
              Else
                GoTo Label_Sublevel_Same_BAS
              End If
            End If
          ElseIf sID2 > sID3 Then
            k = j
            Do While Not (IsEmpty(wsTP.Cells(j, 1)) And _
                          IsEmpty(wsTP.Cells(j, 1).End(xlToRight))) And _
               wsTP.Cells(j, 1) <> "BAS" And sID2 > sID3
              j = j + 1
            Loop
            If IsEmpty(wsTP.Cells(j, 1)) Or wsTP.Cells(j, 1) = "BAS" Then
              GoTo Label_Start
            Else
              GoTo Label_Sublevel_Same_BAS
            End If
          Else
            k = i
            Do While Not (IsEmpty(wsT.Cells(i, 1)) And _
                          IsEmpty(wsT.Cells(i, 1).End(xlToRight))) And _
               wsT.Cells(i, 1) <> "BAS" And sID2 < sID3
              i = i + 1
            Loop
            If IsEmpty(wsT.Cells(i, 1)) Or wsT.Cells(i, 1) = "BAS" Then
              GoTo Label_Start
            Else
              GoTo Label_Sublevel_Same_BAS
            End If
          End If
        End If
        lTM = lTM + 1
        i = i + 1
        j = j + 1

        'Now state transition: decide to which state to switch to
        If wsT.Cells(i, 1) = "BAS" And wsTP.Cells(j, 1) = "BAS" Then
          GoTo Label_Start
        ElseIf wsT.Cells(i, 1) = "BAS" Then
          k = j
          Do While Not (IsEmpty(wsTP.Cells(j, 1)) And IsEmpty(wsTP.Cells(j, 1).End(xlToRight))) _
             And wsTP.Cells(j, 1) <> "BAS"
            j = j + 1
          Loop
          GoTo Label_Start
        ElseIf wsTP.Cells(j, 1) = "BAS" Then
          k = i
          Do While Not (IsEmpty(wsT.Cells(i, 1)) And IsEmpty(wsT.Cells(i, 1).End(xlToRight))) _
             And wsT.Cells(i, 1) <> "BAS"
            i = i + 1
          Loop
          GoTo Label_Start
        End If
      Loop Until IsEmpty(wsT.Cells(i, 1)) Or IsEmpty(wsTP.Cells(j, 1)) Or _
           (wsT.Cells(i, 1) = "BAS" And wsTP.Cells(j, 1) = "BAS")
    Else
      For k = 1 To lColumns
        sCell = wsT.Cells(i, k).Text
        If k = CLng(oFileSpecs(sCoreFile & ":2")) Or _
           k = CLng(oFileSpecs(sCoreFile & ":3")) Or _
           k = CLng(oFileSpecs(sCoreFile & ":4")) Then
          wsTM.Cells(lTM, k) = sCell
        Else
          sCellPrev = wsTP.Cells(j, k).Text
          If IsNumeric(sCell) And IsNumeric(sCellPrev) Then
            wsTM.Cells(lTM, k) = CDbl(sCell) - CDbl(sCellPrev)
          ElseIf IsDate(sCell) And IsDate(sCellPrev) Then
            wsTM.Cells(lTM, k) = CDate(sCell) - CDate(sCellPrev)
          Else
            If sCell = "" Then sCell = "''"
            If sCellPrev = "" Then sCellPrev = "''"
            If sCell = " " Then sCell = "' '"
            If sCellPrev = " " Then sCellPrev = "' '"
            If sCell = sCellPrev Then
              wsTM.Cells(lTM, k) = "="
            Else
              wsTM.Cells(lTM, k) = sCellPrev & "->" & sCell
            End If
          End If
        End If
      Next k
```

```
        i = i + 1
        j = j + 1
        lTM = lTM + 1

        'Now state transition: decide to which state to switch to
        If wsT.Cells(i, 1) = "BAS" And wsTP.Cells(j, 1) = "BAS" Then
          GoTo Label_Start
        ElseIf wsT.Cells(i, 1) = "BAS" Then
          k = j
          Do While Not (IsEmpty(wsTP.Cells(j, 1)) And IsEmpty(wsTP.Cells(j, 1).End(xlToRight))) _
             And wsTP.Cells(j, 1) <> "BAS"
            j = j + 1
          Loop
          GoTo Label_Start
        ElseIf wsTP.Cells(j, 1) = "BAS" Then
          k = i
          Do While Not (IsEmpty(wsT.Cells(i, 1)) And IsEmpty(wsT.Cells(i, 1).End(xlToRight))) _
             And wsT.Cells(i, 1) <> "BAS"
            i = i + 1
          Loop
          GoTo Label_Start
        End If
      End If
    End If

    If IsEmpty(wsT.Cells(i, 1)) Or IsEmpty(wsT.Cells(j, 1)) Then GoTo Label_Start

  Loop  'While Not (IsEmpty(wsT.Cells(i, 1)) And IsEmpty(wsT.Cells(j, 1)))

  '############################## Now process TempMove tab ##############################
  wsTM.Cells.EntireColumn.AutoFit

  'Now iterate over a) row types, b) columns, and c) rows
  For k = 0 To lRowTypes
    'GLogger.info oRowTypes.Keys()(k) & ":COLUMNS = " & _
     oHeader.Item(oRowTypes.Keys()(k) & ":COLUMNS")
    If bHeaderFile Then
      lColumns = oHeader.Item(oRowTypes.keys()(k) & ":COLUMNS")
      If lColumns = 0 Then
        GLogger.warn "File " & vFile & ", row type '" & oRowTypes.keys()(k) & _
                     "' not found in header"
      End If
    End If
    For j = lColumnStart To lColumns
      Set oData = CreateObject("Scripting.Dictionary")
      Set oStringLen = CreateObject("Scripting.Dictionary")
      dSum = 0#
      sMsg = ""
      lCount = 0
      lUBounddStdDevInput = CInitialUBounddStdDevInput
      ReDim dStdDevInput(1 To lUBounddStdDevInput) As Double
      lCountAll = 0
      bIsDate = False
      For i = 1 To 3
        GdMax(i) = -1E+308
        GdMin(i) = 1E+308
      Next i
      GlMaxIdx = 0
      GlMinIdx = 0
      If bHeaderFile Then
        i = lRowTypes + 2
      Else
        i = 2
      End If

      dMin = CMaxDouble
      dMax = CMinDouble
      dMinDate = CMaxDate
      dMaxDate = CMinDate
      lLenMin = CMaxLength
      lLenMax = CMinLength
      bHasEmpty = False

      Do While Not (IsEmpty(wsTM.Cells(i, 1)) And IsEmpty(wsTM.Cells(i, 1).End(xlToRight)))
        If bHeaderFile Then
          sHeader = wsTM.Cells(i, 1).Text & ":" & _
                    wsTM.Cells(i, 2).Text
          bProcessThisRow = (sHeader = oRowTypes.keys()(k))
          sField = oHeader.Item(oRowTypes.keys()(k) & ":" & j)
        Else
          sField = oHeader.Item(CStr(j))
        End If
        If bProcessThisRow Then
          If bHeaderFile Then
            bIsIDField = (oSortCol.Item(wsTM.Cells(i, 2).Text) = j)
          Else
            bIsIDField = (oFileSpecs.Item(sCoreFile & ":2") = j Or _
                          oFileSpecs.Item(sCoreFile & ":3") = j Or _
                          oFileSpecs.Item(sCoreFile & ":4") = j)
          End If
```

```
lCountAll = lCountAll + 1
sCell = wsTM.Cells(i, j).Text
If sCell = "" Then
  bHasEmpty = True
  If bLimitsMoveIn Then
    lLimRow = oLimitMoveRow.Item(sCoreFile & ":" & sField)
    If lLimRow > 0 Then
      If Not CBool(wsLMI.Cells(lLimRow, limAllowEmpty)) Then
        GLogger.warn "Change on File " & vFile & _
                     ", Field " & sField & ", Row " & i & _
                     ", should not be empty"
      End If
    End If
  End If
End If
If IsNumeric(sCell) Then
  dVal = CDbl(sCell)
  If dVal > dMax Then dMax = dVal
  If dVal < dMin Then dMin = dVal
  dSum = dSum + dVal
  lCount = lCount + 1
  dStdDevInput(lCount) = dVal
  Call Extreme(dVal)
  If bLimitsMoveIn Then
    lLimRow = oLimitMoveRow.Item(sCoreFile & ":" & sField)
    If lLimRow > 0 Then
      If Not IsEmpty(wsLMI.Cells(lLimRow, limNumMin)) Then
        If dVal < wsLMI.Cells(lLimRow, limNumMin) Then
          GLogger.warn "Change on File " & vFile & _
                       ", Field " & sField & ", Row " & i & _
                       ", Value " & dVal & " < " & _
                       wsLMI.Cells(lLimRow, limNumMin) & _
                       " [Min Limit]"
        End If
      End If
      If Not IsEmpty(wsLMI.Cells(lLimRow, limNumMax)) Then
        If dVal > wsLMI.Cells(lLimRow, limNumMax) Then
          GLogger.warn "Change on File " & vFile & _
                       ", Field " & sField & ", Row " & i & _
                       ", Value " & dVal & " > " & _
                       wsLMI.Cells(lLimRow, limNumMax) & _
                       " [Max Limit]"
        End If
      End If
    End If
  End If
ElseIf IsDate(sCell) Then
  bIsDate = True
  dtDate = CDate(sCell)
  dVal = CDbl(dtDate)
  If dVal > dMaxDate Then dMaxDate = dVal
  If dVal < dMinDate Then dMinDate = dVal
  dSum = dSum + dVal
  lCount = lCount + 1
  dStdDevInput(lCount) = dVal
  Call Extreme(dVal)
  If bLimitsMoveIn Then
    lLimRow = oLimitMoveRow.Item(sCoreFile & ":" & sField)
    If lLimRow > 0 Then
      If Not IsEmpty(wsLMI.Cells(lLimRow, limDateMin)) Then
        If dVal < CDbl(CDate(wsLMI.Cells(lLimRow, _
                                         limDateMin).Text)) Then
          GLogger.warn "Change on File " & vFile & _
                       ", Field " & sField & ", Row " & i & _
                       ", Date " & Format(dVal, "dd-mmm-yyyy") & _
                       " < " & Format(wsLMI.Cells(lLimRow, _
                                                  limDateMin), "dd-mmm-yyyy") & _
                       " [Min Limit]"
        End If
      End If
      If Not IsEmpty(wsLMI.Cells(lLimRow, limDateMax)) Then
        If dVal > CDbl(CDate(wsLMI.Cells(lLimRow, _
                                         limDateMax).Text)) Then
          GLogger.warn "Change on File " & vFile & _
                       ", Field " & sField & ", Row " & i & _
                       ", Date " & Format(dVal, "dd-mmm-yyyy") & _
                       " > " & Format(wsLMI.Cells(lLimRow, _
                                                  limDateMax), "dd-mmm-yyyy") & _
                       " [Max Limit]"
        End If
      End If
    End If
  End If
Else
  'String
  oData.Item("'" & sCell & "'") = _
                                  oData.Item("'" & sCell & "'") + 1
  lLen = Len(sCell)
  oStringLen.Item(CStr(lLen)) = oStringLen.Item(CStr(lLen)) + 1
  If lLen > 0 Then
```

```
        If lLen > lLenMax Then lLenMax = lLen
        If lLen < lLenMin Then lLenMin = lLen
      End If
      If bLimitsMoveIn Then
        lLimRow = oLimitMoveRow.Item(sCoreFile & ":" & sField)
        If lLimRow > 0 Then
          If Not CBool(wsLI.Cells(lLimRow, limAllowEmpty)) And _
             Not IsEmpty(wsLMI.Cells(lLimRow, limLenMin)) Then
            If lLen < wsLMI.Cells(lLimRow, limLenMin).Value Then
              GLogger.warn "Change on File " & vFile & _
                           ", Field " & sField & ", Row " & i & _
                           ", Length('" & sCell & "') = " & lLen & " < " & _
                           wsLMI.Cells(lLimRow, limLenMin) & _
                           " [Min Length]"
            End If
          End If
          If Not IsEmpty(wsLMI.Cells(lLimRow, limLenMax)) Then
            If lLen > wsLMI.Cells(lLimRow, limLenMax).Value Then
              GLogger.warn "Change on File " & vFile & _
                           ", Field " & sField & ", Row " & i & _
                           ", Length('" & sCell & "') = " & lLen & " > " & _
                           wsLMI.Cells(lLimRow, limLenMax) & _
                           " [Max Length]"
            End If
          End If
        End If
      End If
    End If
  End If  'If bProcessThisRow
  i = i + 1
Loop

If bIsIDField Then
  wsTSM.Cells(lTSM, out2File) = vFile
  If bHeaderFile Then
    wsTSM.Cells(lTSM, out2Header) = oRowTypes.keys()(k)
    wsTSM.Cells(lTSM, out2Field) = oHeader.Item(oRowTypes.keys()(k) & ":" & j)
  Else
    wsTSM.Cells(lTSM, out2Field) = oHeader.Item(CStr(j))
  End If
  wsTSM.Cells(lTSM, out2Warn) = "No stats - this field is an identifier field"
  lTSM = lTSM + 1
Else
  'Deal with numeric values
  If lCount > 0 Then
    wsNSM.Cells(lNSM, outFile) = vFile
    If bHeaderFile Then
      wsNSM.Cells(lNSM, outHeader) = oRowTypes.keys()(k)
    End If
    wsNSM.Cells(lNSM, outField) = sField
    wsNSM.Cells(lNSM, outAverage) = dSum / lCount
    ReDim Preserve dStdDevInput(1 To lCount) As Double
    If lCountAll > lCount Then
      sMsg = sMsg & lCountAll - lCount & " values of " & lCountAll & _
             " are non-numeric. "
    End If
    d = 0#
    On Error Resume Next
    d = Application.WorksheetFunction.StDevP(dStdDevInput)
    On Error GoTo ErrHdl
    If d > 0# Then
      If Abs(dSum / lCount - GdMin(1)) > dStdDevThreshold * d Then
        sMsg = sMsg & "Min is off by more than " & dStdDevThreshold & _
               " stdev. "
      End If
      If Abs(dSum / lCount - GdMax(1)) > dStdDevThreshold * d Then
        sMsg = sMsg & "Max is off by more than " & dStdDevThreshold & _
               " stdev. "
      End If
      '                                Else
      'GLogger.warn "No standard deviation possible for file " & _
       vFile & ", field " & wsNSM.Cells(lNSM, outField)
    End If
    wsNSM.Cells(lNSM, outWarn) = sMsg
    sMsg = ""
    If GlMinIdx > 2 Then wsNSM.Cells(lNSM, outMin1) = GdMin(1)
    If GlMinIdx > 1 Then wsNSM.Cells(lNSM, outMin2) = GdMin(2 + _
                                                      (GlMinIdx < 3))
    If GlMinIdx > 0 Then wsNSM.Cells(lNSM, outMin3) = GdMin(3 + _
                                                      (GlMinIdx < 3) + (GlMinIdx < 2))
    If GlMaxIdx > 2 Then wsNSM.Cells(lNSM, outMax1) = GdMax(1)
    If GlMaxIdx > 1 Then wsNSM.Cells(lNSM, outMax2) = GdMax(2 + _
                                                      (GlMaxIdx < 3))
    If GlMaxIdx > 0 Then wsNSM.Cells(lNSM, outMax3) = GdMax(3 + _
                                                      (GlMaxIdx < 3) + (GlMaxIdx < 2))
    If bIsDate Then
      Range(wsNSM.Cells(lNSM, outMin1), wsNSM.Cells(lNSM, _
                                                    outMax1)).NumberFormat = "dd-mmm-yyyy"
    End If
    lNSM = lNSM + 1
```

```
      End If

      'Deal with string values
      If lCount < lCountAll Then
        wsTSM.Cells(lTSM, out2File) = vFile
        If bHeaderFile Then
          wsTSM.Cells(lTSM, out2Header) = oRowTypes.keys()(k)
          wsTSM.Cells(lTSM, out2Field) = oHeader.Item(oRowTypes.keys()(k) & ":" & j)
        Else
          wsTSM.Cells(lTSM, out2Field) = oHeader.Item(CStr(j))
        End If
        If lCount > 0 Then
          sMsg = sMsg & lCount & " values of " & lCountAll & " are numeric. "
        End If
        If oData.Count > lCountAll - 5 Or (oData.Count / lCountAll > 0.5 And _
                                           lCountAll > 10) Then
          For i = oData.Count - 1 To 0 Step -1
            If oData.keys()(i) <> "''" And oData.keys()(i) <> "' '" _
               And oData.Item(oData.keys()(i)) = 1 _
               And InStr(oData.keys()(i), "->") = 0 Then
              oData.Remove oData.keys()(i)
            End If
          Next i
          If lCountAll - oData.Count > 0 Then
            sMsg = sMsg & lCountAll - oData.Count & _
                   " strings occurring once not shown. "
          End If
        End If
        wsTSM.Cells(lTSM, out2Warn) = sMsg
        sMsg = ""
        If oData.Count > 0 Then
          On Error Resume Next
          wsTSM.Cells(lTSM, out2Text).Resize(oData.Count, 2) = _
               Application.WorksheetFunction.Transpose(Array(oData.keys, oData.items))
          On Error GoTo ErrHdl
        End If
        If oStringLen.Count > 0 Then
          On Error Resume Next
          wsTSM.Cells(lTSM, out2Len).Resize(oStringLen.Count, 2) = _
               Application.WorksheetFunction.Transpose(Array(oStringLen.keys, oStringLen.items))
          On Error GoTo ErrHdl
        End If

        'Increase row counter
        lMax = 1
        If lMax < oData.Count Then lMax = oData.Count
        If lMax < oStringLen.Count Then lMax = oStringLen.Count
        lTSM = lTSM + lMax
      End If

      'Update Limits Move Out sheet
      wsLMO.Cells(lLMO, limFilePrefix) = sCoreFile
      If bHeaderFile Then
        wsLMO.Cells(lLMO, limHeader) = oRowTypes.keys()(k)
        wsLMO.Cells(lLMO, limField) = oHeader.Item(oRowTypes.keys()(k) & ":" & j)
      Else
        wsLMO.Cells(lLMO, limField) = oHeader.Item(CStr(j))
      End If
      If bHasEmpty Then wsLMO.Cells(lLMO, limAllowEmpty) = 1  'True
      If dMin < CMaxDouble Then wsLMO.Cells(lLMO, limNumMin) = dMin
      If dMax > CMinDouble Then wsLMO.Cells(lLMO, limNumMax) = dMax
      If dMinDate < CMaxDate Then
        wsLMO.Cells(lLMO, limDateMin) = dMinDate
        wsLMO.Cells(lLMO, limDateMin).NumberFormat = "dd-mmm-yyyy"
      End If
      If dMaxDate > CMinDate Then
        wsLMO.Cells(lLMO, limDateMax) = dMaxDate
        wsLMO.Cells(lLMO, limDateMax).NumberFormat = "dd-mmm-yyyy"
      End If
      If lLenMin < CMaxLength Then wsLMO.Cells(lLMO, limLenMin) = lLenMin
      If lLenMax > CMinLength Then wsLMO.Cells(lLMO, limLenMax) = lLenMax
      lLMO = lLMO + 1
    End If  'If bIsIDField

    Set oData = Nothing
    Set oStringLen = Nothing

  Next j
Next k

If lTM > 1 Then
  Call ExportRangeAsDelimitedText("TempMove", "A1:" & _
       wsTM.Cells(1, 1).SpecialCells(xlCellTypeLastCell).Address, _
       ThisWorkbook.Path & "\Datastats\" & "MOVE_" & sCoreFile & _
       "_" & Format(Now, "yyyymmdd") & ".csv", ",", True, _
       True, False)
End If

'####################### TempMove tab processing finished ########################
```

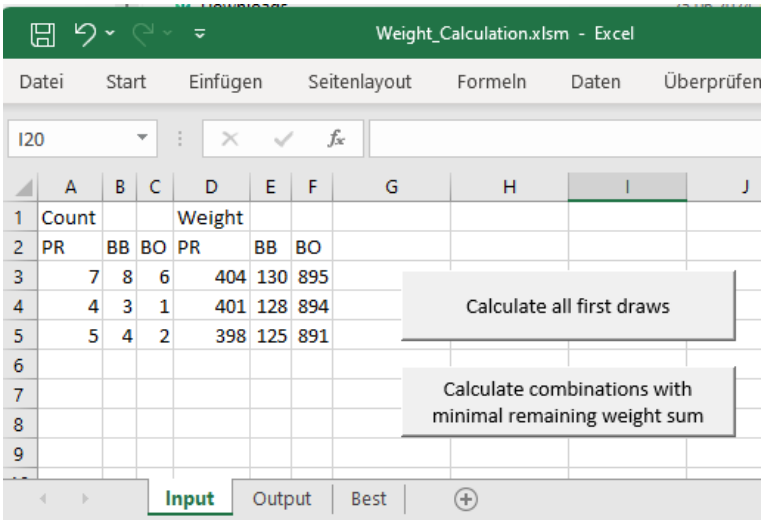
```
          wsTP.QueryTables.Item(1).Delete
        Else
          GLogger.warn "Previous period's file " & sFilePrev & " does not exist"
        End If
      End If  'bMoveCheck

      wsT.QueryTables.Item(1).Delete
      Set oRowTypes = Nothing
    Else
      GLogger.warn "Skipping zero file " & vFile
    End If
    If bClearDataAfterRun Then
      wsT.Range("1:1048576").Delete
      wsTP.Range("1:1048576").Delete
    End If
  Next vFile
  Call ExportRangeAsDelimitedText("NumStats", "A1:" & wsNS.Cells(lOut - 1, outUbound - 1).Address, _
                                  ThisWorkbook.Path & "\Datastats\" & "NumStats_" & _
                                  Format(Now, "yyyymmdd") & ".csv", ",", True, _
                                  True, False)
  Call ExportRangeAsDelimitedText("NumStatsMove", "A1:" & wsNSM.Cells(lNSM - 1, outUbound - 1).Address, _
                                  ThisWorkbook.Path & "\Datastats\" & "NumStatsMove_" & _
                                  Format(Now, "yyyymmdd") & ".csv", ",", True, _
                                  True, False)
  Call ExportRangeAsDelimitedText("TextStats", "A1:" & wsTS.Cells(lOut2 - 1, out2Ubound - 1).Address, _
                                  ThisWorkbook.Path & "\Datastats\" & "TextStats_" & _
                                  Format(Now, "yyyymmdd") & ".csv", ",", True, _
                                  True, False)
  Call ExportRangeAsDelimitedText("TextStatsMove", "A1:" & wsTSM.Cells(lTSM - 1, out2Ubound - 1).Address, _
                                  ThisWorkbook.Path & "\Datastats\" & "TextStatsMove_" & _
                                  Format(Now, "yyyymmdd") & ".csv", ",", True, _
                                  True, False)
  Call ExportRangeAsDelimitedText("LimitsOut", "A1:" & wsLO.Cells(lLim - 1, limCharSet + 255).Address, _
                                  ThisWorkbook.Path & "\Datastats\Limits_Output_" & _
                                  Format(Now, "yyyymmdd") & ".csv", ",", True, _
                                  True, False)
  Call ExportRangeAsDelimitedText("LimitsMoveOut", _
                                  "A1:" & wsLMO.Cells(lLMO - 1, limCharSet + 255).Address, _
                                  ThisWorkbook.Path & "\Datastats\Limits_Move_Output_" & _
                                  Format(Now, "yyyymmdd") & ".csv", ",", True, _
                                  True, False)
  Call ExportRangeAsDelimitedText("DeletedRecords", _
                                  "A1:" & wsDel.Range("A1").SpecialCells(xlCellTypeLastCell).Address, _
                                  ThisWorkbook.Path & "\Datastats\" & "DeletedRecords_" & _
                                  Format(Now, "yyyymmdd") & ".csv", ",", True, _
                                  True, False)
  Call ExportRangeAsDelimitedText("NewRecords", _
                                  "A1:" & wsN.Range("A1").SpecialCells(xlCellTypeLastCell).Address, _
                                  ThisWorkbook.Path & "\Datastats\" & "NewRecords_" & _
                                  Format(Now, "yyyymmdd") & ".csv", ",", True, _
                                  True, False)
  wsW.Select
  If bClearDataAfterRun Then
    wsNS.Range("1:1048576").Delete
    wsDel.Range("1:1048576").Delete
    wsFS.Range("1:1048576").Delete
    wsLI.Range("1:1048576").Delete
    wsLO.Range("1:1048576").Delete
    wsLMI.Range("1:1048576").Delete
    wsLMO.Range("1:1048576").Delete
    wsN.Range("1:1048576").Delete
    wsNSM.Range("1:1048576").Delete
    wsTS.Range("1:1048576").Delete
    wsTM.Range("1:1048576").Delete
    wsTSM.Range("1:1048576").Delete
  End If
  ThisWorkbook.SaveAs sFullName, lFileFormat
  GLogger.info "Reading Input finished"
  GLogger.info String(160, "<")
  Call Delete_QueryTables

  'Function doFileQuery provided unwanted names which now show #REF! errors. Delete those.
  For Each v In ActiveWorkbook.Names
    If InStr(v.Value, "#REF!") Then v.Delete
  Next

  'At the end of this sub the variable state gets destroyed, i.e. previous
  'system state variable values will be written back

  Exit Sub

ErrHdl:
  If Err.Number = 9 Then
    If lCount > lUBounddStdDevInput Then
      'Here we normally get if we breach dStdDevInput(lCount)
      'So we need to increase array size of dStdDevInput
      lUBounddStdDevInput = 2 * lUBounddStdDevInput
      ReDim Preserve dStdDevInput(1 To lUBounddStdDevInput) As Double
      Err.Number = 0
      Resume  'Back to statement which caused error
```

```
    End If
  End If
  'Other error - terminate
  On Error GoTo 0
  Resume

End Sub

Sub Extreme(d)
'Keep track of extreme values
'
'Change History:
'Version Date       Programmer Change
'1.00    21/12/2015 Bernd      Create

'Initialise with, for example:
'Dim GdMax(1 To 3) As Double
'Dim GlMaxIdx As Long
'Dim GdMin(1 To 3) As Double
'Dim GlMinIdx As Long
'For i = 1 To 3
'    GdMax(i) = -1E+308
'    GdMin(i) = 1E+308
'Next i
'GlMaxIdx = 0
'GlMinIdx = 0

  Dim i As Long, j As Long
  i = 1
  Do While i <= GlMaxIdx
    If d > GdMax(i) Then Exit Do
    i = i + 1
  Loop
  If i <= UBound(GdMax) Then
    If UBound(GdMax) > GlMaxIdx Then
      GlMaxIdx = GlMaxIdx + 1
    End If
    For j = GlMaxIdx To i + 1 Step -1
      GdMax(j) = GdMax(j - 1)
    Next j
    GdMax(i) = d
  End If
  i = 1
  Do While i <= GlMinIdx
    If d < GdMin(i) Then Exit Do
    i = i + 1
  Loop
  If i <= UBound(GdMin) Then
    If UBound(GdMin) > GlMinIdx Then
      GlMinIdx = GlMinIdx + 1
    End If
    For j = GlMinIdx To i + 1 Step -1
      GdMin(j) = GdMin(j - 1)
    Next j
    GdMin(i) = d
  End If

End Sub

Sub Delete_QueryTables()
'Delete query tables - hard code ignoring errors
'
'Change History:
'Version Date       Programmer Change
'1.00    21/12/2015 Bernd      Create

  On Error Resume Next  'If the QueryTable does not exist the next row will encounter an error but not
terminate
  wsNS.QueryTables.Item(1).Delete
  wsFS.QueryTables.Item(1).Delete
  wsLI.QueryTables.Item(1).Delete
  wsLMI.QueryTables.Item(1).Delete
  wsT.QueryTables.Item(1).Delete
  wsTP.QueryTables.Item(1).Delete
  wsTS.QueryTables.Item(1).Delete
  On Error GoTo 0  'Any errors will let the program terminate

End Sub

Function Files_PrefixStart_SuffixEnd(ParamArray vPath() As Variant) As Variant
'Determines common prefix and suffix strings of all files in one or more folders
'and returns prefix end (positive number) and suffix start (negative number) as
'well as prefix string and suffix string in a variant array with four values.
'Example: Files abc_file1_20181231.csv and abc_file2_20181231.csv would
'         result in return variant array [8,-13,"abc_file","_20181231.csv"}
'
'Change History:
'Version Date       Programmer Change
'0.1     05/01/2019 Bernd      Create
Dim i As Long, k As Long, lPrefix As Long, lSuffix As Long
```

```
Dim sFile1 As String, sFile As String, sPrefix As String, sSuffix As String

If Right(vPath(0), 1) <> "\" And Right(vPath(0), 1) <> "/" Then
  Files_PrefixStart_SuffixEnd = CVErr(xlErrValue)
  Exit Function
End If
sFile = Dir(vPath(0)): sPrefix = sFile: sSuffix = sFile: sFile1 = sFile
lPrefix = 0: lSuffix = 0    'Just one file so far
For k = LBound(vPath) To UBound(vPath)
  sFile = Dir(vPath(k))
  Do While Len(sFile) > 0
    If Len(sPrefix) > 0 Then
      If Left(sFile, Len(sPrefix)) <> sPrefix Then
        i = 0
        Do While Mid(sFile, i + 1, 1) = Mid(sPrefix, i + 1, 1)
          If i + 1 > Len(sFile) Or i + 1 > Len(sPrefix) Then Exit Do
          i = i + 1
        Loop
        If i < Len(sPrefix) Then sPrefix = Left(sPrefix, i): lPrefix = i
      End If
    End If
    If Len(sSuffix) > 0 Then
      If Right(sFile, Len(sSuffix)) <> sSuffix Then
        i = 0
        Do While Left(Right(sFile, i + 1), 1) = Left(Right(sSuffix, i + 1), 1)
          If i + 1 > Len(sFile) Or i + 1 > Len(sSuffix) Then Exit Do
          i = i + 1
        Loop
        If i < Len(sSuffix) Then sSuffix = Right(sSuffix, i): lSuffix = -i
      End If
    End If
    sFile = Dir()
  Loop
Next k
If lPrefix = 0 And lSuffix = 0 And Left(Right(sFile1, 4), 1) = "." Then sPrefix = "": sSuffix =
Right(sFile1, 4)
lPrefix = Len(sPrefix): lSuffix = Len(sSuffix)
Files_PrefixStart_SuffixEnd = Array(lPrefix, lSuffix, sPrefix, sSuffix)
End Function

Function sbCat(vP As Variant, _
               Optional sDel As String = ",", _
               Optional bNonEmpty As Boolean = True) As String
'Concatenate all cells in a range or array, delimited
'by sDel. If bNonEmpty is TRUE then only non-empty
'cells will be concatenated.
Dim v, s                    As String

For Each v In vP
  If Not (bNonEmpty And v = "") Then
    sbCat = sbCat & s & v
    s = sDel
  End If
Next v

End Function
```

Gewichtberechnung

Angenommen, Sie haben drei verschiedene Produkte in jeweils 3 Ausprägungen mit unterschiedlichen Gewichten, die auch in unterschiedlicher Anzahl vorliegen. Das Produkt PR existiert mit den Gewichten 404 g (davon gibt es 7 Stück), 401 g (vier Mal) und 398 g (fünf Mal), und es gibt entsprechend weitere Produkte BB und BO:



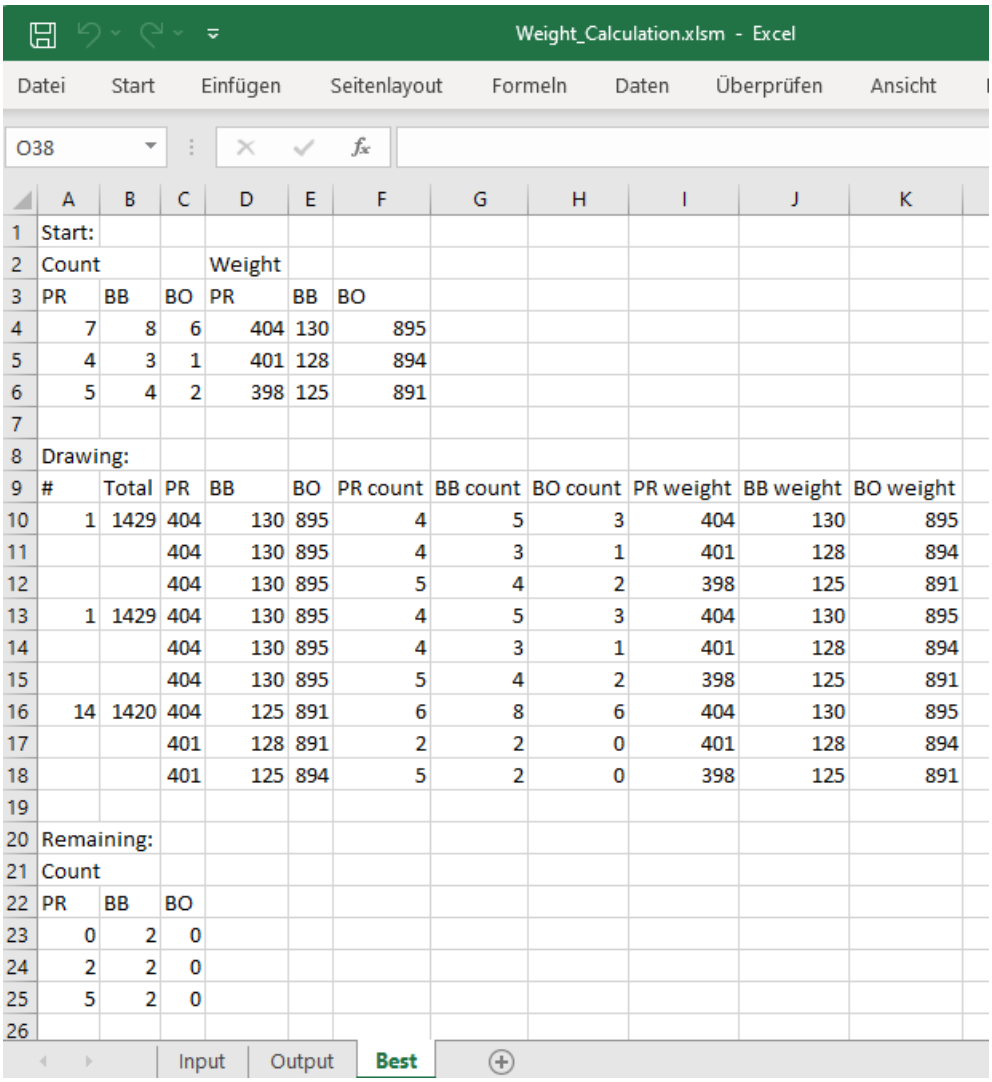
Nun sollen jeweils drei mal drei Produkte (immer von allen drei Produkten genau eins) mit identischer Gewichtssumme ermittelt und die Restmenge an Produkten ausgegeben werden:

	A	B	C	D	E	F	G	H	I	J	K
1	#	Total	PR	BB	BO	PR count	BB count	BO count	PR weight	BB weight	BO weight
2	1	1429	404	130	895	4	5	3	404	130	895
3			404	130	895	4	3	1	401	128	894
4			404	130	895	5	4	2	398	125	891
5	2	1427	404	128	895	4	8	3	404	130	895
6			404	128	895	4	0	1	401	128	894
7			404	128	895	5	4	2	398	125	891
8	3	1423	404	128	891	4	8	6	404	130	895
9			404	128	891	4	1	0	401	128	894
10			404	125	894	5	3	0	398	125	891
11	4	1424	404	125	895	4	8	3	404	130	895
12			404	125	895	4	3	1	401	128	894
13			404	125	895	5	1	2	398	125	891
14	5	1425	404	130	891	5	5	6	404	130	895
15			404	130	891	3	3	0	401	128	894
16			401	130	894	5	4	0	398	125	891
17	6	1423	404	128	891	5	8	6	404	130	895
18			404	128	891	3	0	0	401	128	894
19			401	128	894	5	4	0	398	125	891
20	7	1424	404	125	895	5	8	3	404	130	895
21			404	125	895	3	2	1	401	128	894
22			401	128	895	5	2	2	398	125	891
23	8	1420	404	125	891	5	8	6	404	130	895
24			404	125	891	3	3	0	401	128	894
25			401	125	894	5	1	0	398	125	891
26	9	1423	404	128	891	5	7	5	404	130	895
27			404	128	891	4	1	1	401	128	894
28			398	130	895	4	4	0	398	125	891
29	10	1423	404	128	891	5	7	5	404	130	895
30			404	125	894	4	2	0	401	128	894
31			398	130	895	4	3	1	398	125	891
32	11	1420	404	125	891	5	8	6	404	130	895
33			404	125	891	4	2	0	401	128	894
34			398	128	894	4	2	0	398	125	891
35	12	1426	404	128	894	6	6	4	404	130	895
36			401	130	895	2	2	0	401	128	894
37			401	130	895	5	4	2	398	125	891
38	13	1424	404	125	895	6	8	3	404	130	895
39			401	128	895	2	1	1	401	128	894
40			401	128	895	5	3	2	398	125	891
41	14	1420	404	125	891	6	8	6	404	130	895
42			401	128	891	2	2	0	401	128	894
43			401	125	894	5	2	0	398	125	891
44	15	1423	404	128	891	6	7	5	404	130	895
45			401	128	894	3	1	0	401	128	894
46			398	130	895	4	4	1	398	125	891
47	16	1420	404	125	891	6	8	6	404	130	895
48			401	128	891	3	1	0	401	128	894
49			398	128	894	4	3	0	398	125	891
50	17	1423	404	128	891	6	6	4	404	130	895
51			398	130	895	4	2	1	401	128	894
52			398	130	895	3	4	1	398	125	891
53	18	1423	404	125	894	6	6	4	404	130	895
54			398	130	895	4	3	0	401	128	894
55			398	130	895	3	3	2	398	125	891

56	19	1426	401	130	895	7	5	3	404	130	895
57			401	130	895	1	3	1	401	128	894
58			401	130	895	5	4	2	398	125	891
59	20	1424	401	128	895	7	8	3	404	130	895
60			401	128	895	1	0	1	401	128	894
61			401	128	895	5	4	2	398	125	891
62	21	1420	401	128	891	7	8	6	404	130	895
63			401	128	891	1	1	0	401	128	894
64			401	125	894	5	3	0	398	125	891
65	22	1421	401	125	895	7	8	3	404	130	895
66			401	125	895	1	3	1	401	128	894
67			401	125	895	5	1	2	398	125	891
68	23	1422	401	130	891	7	5	6	404	130	895
69			401	130	891	2	3	0	401	128	894
70			398	130	894	4	4	0	398	125	891
71	24	1420	401	128	891	7	8	6	404	130	895
72			401	128	891	2	0	0	401	128	894
73			398	128	894	4	4	0	398	125	891
74	25	1421	401	125	895	7	8	3	404	130	895
75			401	125	895	2	2	1	401	128	894
76			398	128	895	4	2	2	398	125	891
77	26	1417	401	125	891	7	8	6	404	130	895
78			401	125	891	2	3	0	401	128	894
79			398	125	894	4	1	0	398	125	891
80	27	1423	401	128	894	7	6	4	404	130	895
81			398	130	895	3	2	0	401	128	894
82			398	130	895	3	4	2	398	125	891
83	28	1421	401	125	895	7	8	3	404	130	895
84			398	128	895	3	1	1	401	128	894
85			398	128	895	3	3	2	398	125	891
86	29	1417	401	125	891	7	8	6	404	130	895
87			398	128	891	3	2	0	401	128	894
88			398	125	894	3	2	0	398	125	891
89	30	1423	398	130	895	7	5	3	404	130	895
90			398	130	895	4	3	1	401	128	894
91			398	130	895	2	4	2	398	125	891
92	31	1421	398	128	895	7	8	3	404	130	895
93			398	128	895	4	0	1	401	128	894
94			398	128	895	2	4	2	398	125	891
95	32	1417	398	128	891	7	8	6	404	130	895
96			398	128	891	4	1	0	401	128	894
97			398	125	894	2	3	0	398	125	891
98	33	1418	398	125	895	7	8	3	404	130	895
99			398	125	895	4	3	1	401	128	894
100			398	125	895	2	1	2	398	125	891

Es gibt viele Alternativen, alle möglichen Entnahmen zu ermitteln. Im Anhang ist eine sehr einfache, aber recht aufwendige Option genannt (siehe ersten Programmcode), die naiv alle möglichen Kombinationen durchgeht. Der zweite Programmcode zeigt eine Monte Carlo Simulation, die die Funktion UniqRandInt verwendet und mit 500.000 Iterationen sehr wahrscheinlich (aber nicht sicher) alle Möglichkeiten ermittelt. Eine dritte Möglichkeit bestünde in der Nutzung einer Permutationsfunktion, die nur alle möglichen 84 * 84 * 20 = 141,120 Permutationen untersucht.

Eine (von 12 verschiedenen) kombinierte Entnahme mit der geringsten Restgewichtssumme ist:



	A	B	C	D	E	F	G	H	I	J	K
1	Start:										
2	Count			Weight							
3	PR	BB	BO	PR	BB	BO					
4	7	8	6	404	130	895					
5	4	3	1	401	128	894					
6	5	4	2	398	125	891					
7											
8	Drawing:										
9	#	Total	PR	BB	BO	PR count	BB count	BO count	PR weight	BB weight	BO weight
10	1	1429	404	130	895	4	5	3	404	130	895
11			404	130	895	4	3	1	401	128	894
12			404	130	895	5	4	2	398	125	891
13	1	1429	404	130	895	4	5	3	404	130	895
14			404	130	895	4	3	1	401	128	894
15			404	130	895	5	4	2	398	125	891
16	14	1420	404	125	891	6	8	6	404	130	895
17			401	128	891	2	2	0	401	128	894
18			401	125	894	5	2	0	398	125	891
19											
20	Remaining:										
21	Count										
22	PR	BB	BO								
23	0	2	0								
24	2	2	0								
25	5	2	0								
26											

Alle 12 verschiedenen Entnahmekombinationen - die Nummern beziehen sich auf die o. g. Ausgabevariante:

Erste Ziehung	Zweite Ziehung	Dritte Ziehung
1	1	14
1	1	16
1	1	21
1	1	24
1	2	23
1	3	19
1	5	7
1	5	13
1	5	20
1	6	19
1	9	12
2	5	19

Sinnvolle Erweiterungen und Verallgemeinerungen

Die rasch entwickelte erste obige Lösung könnte gemäß folgender Ansätze erweitert und verallgemeinert werden:

https://stackoverflow.com/questions/54669041/vba-write-all-permutations-of-numbers-to-an-array (auch hier: https://www.vitoshacademy.com/vba-nested-loops-with-recursion/)

https://www.physicsforums.com/threads/loop-with-variable-nesting-depth-and-variable-count-at-each-level.1046986/

https://www.codeproject.com/Tips/759707/Generating-dynamically-nested-loops

https://stackoverflow.com/questions/1737289/dynamic-nested-loops-level

AllFirstDraws Programmcode

```
'Calculates 3 * 3 - tuples of same total weights.
'(C) (P) by Bernd Plumhoff 26-Jun-2024 PB V0.4

Sub AllFirstDraws()
Dim i As Long, j As Long, k As Long
Dim i2 As Long, j2 As Long, k2 As Long
Dim i3 As Long, j3 As Long, k3 As Long
Dim m As Long, n As Long, t As Long, v As Long

Dim oGetRidofDupes           As Object

Dim vCount                   As Variant
Dim vWeight                  As Variant

Dim state                    As SystemState

With Application.WorksheetFunction
  Set state = New SystemState
  wsI.Cells.EntireColumn.AutoFit
  wsO.Cells.ClearContents
  Set oGetRidofDupes = CreateObject("Scripting.Dictionary")
  i = 1
  Do While wsI.Cells(2, i) <> ""
    i = i + 1
  Loop
  n = (i - 1) \ 2
  vCount = .Transpose(Range(wsI.Cells(3, 1), wsI.Cells(3, n).End(xlDown)))
  vWeight = .Transpose(Range(wsI.Cells(3, n + 1), wsI.Cells(3, 2 * n).End(xlDown)))
  For i = 1 To n
    k = 0
    For j = 1 To UBound(vCount, 2)
      k = k + vCount(j, i)
    Next j
    If k < n Then
      Call MsgBox("Not enough items in column " & i, vbOKOnly, "Error")
      Exit Sub
    End If
  Next i
  m = j - 1
  'Debug.Print "n = " & n, "m = " & m
  'Now we know the dimensions
  ReDim sItem(1 To n) As String
  wsO.Cells(1, 1) = "#"
  wsO.Cells(1, 2) = "Total"
  For i = 1 To n
    sItem(i) = wsI.Cells(2, i)
    wsO.Cells(1, i + 2) = sItem(i)
    wsO.Cells(1, n + 2 + i) = sItem(i) & " count"
    wsO.Cells(1, 2 * n + 2 + i) = sItem(i) & " weight"
  Next i

  ReDim lPermutWeight(1 To n, 1 To n * m) As Long
  ReDim lPermutIdx(1 To n) As Long
  ReDim lPermutSubGroupIdx(1 To n, 1 To n * m) As Long

  For i = 1 To n
```

```
  t = 0
  For j = 1 To m
    For k = 1 To .Min(n, vCount(i, j))
      t = t + 1
      lPermutWeight(i, t) = vWeight(i, j)
      lPermutSubGroupIdx(i, t) = j
    Next k
  Next j
  lPermutIdx(i) = t
Next i

v = 2
For i = 1 To lPermutIdx(1)
  For j = 1 To lPermutIdx(1)
    If j <> i Then
      For k = 1 To lPermutIdx(1)
        If k <> j And k <> i Then
          For i2 = 1 To lPermutIdx(2)
            For j2 = 1 To lPermutIdx(2)
              If j2 <> i2 Then
                For k2 = 1 To lPermutIdx(2)
                  If k2 <> j2 And k2 <> i2 Then
                    For i3 = 1 To lPermutIdx(3)
                      For j3 = 1 To lPermutIdx(3)
                        If j3 <> i3 Then
                          For k3 = 1 To lPermutIdx(3)
                            If k3 <> j3 And k3 <> i3 Then
                              'Debug.Print lPermutWeight(1, i) & " + " & lPermutWeight(2, i2) & _
                               '" + " & lPermutWeight(3, i3) & " ?= " & lPermutWeight(1, j) & _
                               '" + " & lPermutWeight(2, j2) & " + " & lPermutWeight(3, j3) & _
                               '" And " & lPermutWeight(1, i) & " + " & lPermutWeight(2, i2) & _
                               '" + " & lPermutWeight(3, i3) & " ?= " & lPermutWeight(1, k) & _
                               '" + " & lPermutWeight(2, k2) & " + " & lPermutWeight(3, k3)
                              If lPermutWeight(1, i) + lPermutWeight(2, i2) + lPermutWeight(3, i3) = _
                                 lPermutWeight(1, j) + lPermutWeight(2, j2) + lPermutWeight(3, j3) And _
                                 lPermutWeight(1, i) + lPermutWeight(2, i2) + lPermutWeight(3, i3) = _
                                 lPermutWeight(1, k) + lPermutWeight(2, k2) + lPermutWeight(3, k3) Then
                                If Not oGetRidofDupes.exists(lPermutWeight(1, i) & "|" & _
                                                             lPermutWeight(2, i2) & "|" & _
                                                             lPermutWeight(3, i3) & "|" & _
                                                             lPermutWeight(1, j) & "|" & _
                                                             lPermutWeight(2, j2) & "|" & _
                                                             lPermutWeight(3, j3) & "|" & _
                                                             lPermutWeight(1, k) & "|" & _
                                                             lPermutWeight(2, k2) & "|" & _
                                                             lPermutWeight(3, k3)) Then
                                  oGetRidofDupes(lPermutWeight(1, i) & "|" & lPermutWeight(2, i2) & _
                                                 "|" & lPermutWeight(3, i3) & "|" & _
                                                 lPermutWeight(1, j) & "|" & _
                                                 lPermutWeight(2, j2) & "|" & _
                                                 lPermutWeight(3, j3) & "|" & _
                                                 lPermutWeight(1, k) & "|" & _
                                                 lPermutWeight(2, k2) & "|" & _
                                                 lPermutWeight(3, k3)) = 1
                                  oGetRidofDupes(lPermutWeight(1, i) & "|" & lPermutWeight(2, i2) & _
                                                 "|" & lPermutWeight(3, i3) & "|" & _
                                                 lPermutWeight(1, k) & "|" & _
                                                 lPermutWeight(2, k2) & "|" & _
                                                 lPermutWeight(3, k3) & "|" & _
                                                 lPermutWeight(1, j) & "|" & _
                                                 lPermutWeight(2, j2) & "|" & _
                                                 lPermutWeight(3, j3)) = 1
                                  oGetRidofDupes(lPermutWeight(1, j) & "|" & _
                                                 lPermutWeight(2, j2) & "|" & _
                                                 lPermutWeight(3, j3) & "|" & _
                                                 lPermutWeight(1, i) & "|" & _
                                                 lPermutWeight(2, i2) & "|" & _
                                                 lPermutWeight(3, i3) & "|" & _
                                                 lPermutWeight(1, k) & "|" & _
                                                 lPermutWeight(2, k2) & "|" & _
                                                 lPermutWeight(3, k3)) = 1
                                  oGetRidofDupes(lPermutWeight(1, j) & "|" & _
                                                 lPermutWeight(2, j2) & "|" & _
                                                 lPermutWeight(3, j3) & "|" & _
                                                 lPermutWeight(1, k) & "|" & _
                                                 lPermutWeight(2, k2) & "|" & _
                                                 lPermutWeight(3, k3) & "|" & _
                                                 lPermutWeight(1, i) & "|" & _
                                                 lPermutWeight(2, i2) & "|" & _
                                                 lPermutWeight(3, i3)) = 1
                                  oGetRidofDupes(lPermutWeight(1, k) & "|" & _
                                                 lPermutWeight(2, k2) & "|" & _
                                                 lPermutWeight(3, k3) & "|" & _
                                                 lPermutWeight(1, i) & "|" & _
                                                 lPermutWeight(2, i2) & "|" & _
                                                 lPermutWeight(3, i3) & "|" & _
                                                 lPermutWeight(1, j) & "|" & _
                                                 lPermutWeight(2, j2) & "|" & _
                                                 lPermutWeight(3, j3)) = 1
                                  oGetRidofDupes(lPermutWeight(1, k) & "|" & _
```

```
                                    lPermutWeight(2, k2) & "|" & _
                                    lPermutWeight(3, k3) & "|" & _
                                    lPermutWeight(1, j) & "|" & _
                                    lPermutWeight(2, j2) & "|" & _
                                    lPermutWeight(3, j3) & "|" & _
                                    lPermutWeight(1, i) & "|" & _
                                    lPermutWeight(2, i2) & "|" & _
                                    lPermutWeight(3, i3)) = 1
                                wsO.Cells(v, 1) = (v + 1) \ n
                                wsO.Cells(v, 2) = lPermutWeight(1, i) + lPermutWeight(2, i2) + _
                                    lPermutWeight(3, i3)
                                wsO.Cells(v, 3) = lPermutWeight(1, i)
                                wsO.Cells(v, 4) = lPermutWeight(2, i2)
                                wsO.Cells(v, 5) = lPermutWeight(3, i3)
                                wsO.Cells(v + 1, 3) = lPermutWeight(1, j)
                                wsO.Cells(v + 1, 4) = lPermutWeight(2, j2)
                                wsO.Cells(v + 1, 5) = lPermutWeight(3, j3)
                                wsO.Cells(v + 2, 3) = lPermutWeight(1, k)
                                wsO.Cells(v + 2, 4) = lPermutWeight(2, k2)
                                wsO.Cells(v + 2, 5) = lPermutWeight(3, k3)
                                wsO.Cells(v, 6) = vCount(1, 1) - _
                                    IIf(lPermutSubGroupIdx(1, i) = 1, 1, 0) - _
                                    IIf(lPermutSubGroupIdx(1, j) = 1, 1, 0) - _
                                    IIf(lPermutSubGroupIdx(1, k) = 1, 1, 0)
                                wsO.Cells(v, 7) = vCount(2, 1) - _
                                    IIf(lPermutSubGroupIdx(2, i2) = 1, 1, 0) - _
                                    IIf(lPermutSubGroupIdx(2, j2) = 1, 1, 0) - _
                                    IIf(lPermutSubGroupIdx(2, k2) = 1, 1, 0)
                                wsO.Cells(v, 8) = vCount(3, 1) - _
                                    IIf(lPermutSubGroupIdx(3, i3) = 1, 1, 0) - _
                                    IIf(lPermutSubGroupIdx(3, j3) = 1, 1, 0) - _
                                    IIf(lPermutSubGroupIdx(3, k3) = 1, 1, 0)
                                wsO.Cells(v + 1, 6) = vCount(1, 2) - _
                                      IIf(lPermutSubGroupIdx(1, i) = 2, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(1, j) = 2, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(1, k) = 2, 1, 0)
                                wsO.Cells(v + 1, 7) = vCount(2, 2) - _
                                      IIf(lPermutSubGroupIdx(2, i2) = 2, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(2, j2) = 2, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(2, k2) = 2, 1, 0)
                                wsO.Cells(v + 1, 8) = vCount(3, 2) - _
                                      IIf(lPermutSubGroupIdx(3, i3) = 2, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(3, j3) = 2, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(3, k3) = 2, 1, 0)
                                wsO.Cells(v + 2, 6) = vCount(1, 3) - _
                                      IIf(lPermutSubGroupIdx(1, i) = 3, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(1, j) = 3, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(1, k) = 3, 1, 0)
                                wsO.Cells(v + 2, 7) = vCount(2, 3) - _
                                      IIf(lPermutSubGroupIdx(2, i2) = 3, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(2, j2) = 3, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(2, k2) = 3, 1, 0)
                                wsO.Cells(v + 2, 8) = vCount(3, 3) - _
                                      IIf(lPermutSubGroupIdx(3, i3) = 3, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(3, j3) = 3, 1, 0) - _
                                      IIf(lPermutSubGroupIdx(3, k3) = 3, 1, 0)
                                wsO.Cells(v, 9) = vWeight(1, 1)
                                wsO.Cells(v, 10) = vWeight(2, 1)
                                wsO.Cells(v, 11) = vWeight(3, 1)
                                wsO.Cells(v + 1, 9) = vWeight(1, 2)
                                wsO.Cells(v + 1, 10) = vWeight(2, 2)
                                wsO.Cells(v + 1, 11) = vWeight(3, 2)
                                wsO.Cells(v + 2, 9) = vWeight(1, 3)
                                wsO.Cells(v + 2, 10) = vWeight(2, 3)
                                wsO.Cells(v + 2, 11) = vWeight(3, 3)
                                v = v + 3
                            End If
                        End If
                    End If
                Next k3
            End If
        Next j3
    Next i3
End If
Next k2
End If
Next j2
Next i2
End If
Next k
End If
Next j
Next i
wsO.Cells.EntireColumn.AutoFit
End With
End Sub
```

AllFirstDraws Monte Carlo Programmcode

```
'Calculates 3 * 3 - tuples of same total weights.
'(C) (P) by Bernd Plumhoff 07-Jul-2024 PB V0.41

Sub AllFirstDraws()
Dim i                           As Long
Dim j                           As Long
Dim k                           As Long
Dim i2                          As Long
Dim j2                          As Long
Dim k2                          As Long
Dim i3                          As Long
Dim j3                          As Long
Dim k3                          As Long
Dim m                           As Long
Dim n                           As Long
Dim t                           As Long
Dim u                           As Long
Dim v                           As Long
Dim x                           As Long

Dim v1                          As Variant
Dim v2                          As Variant
Dim v3                          As Variant

Dim oGetRidofDupes              As Object

Dim vCount                      As Variant
Dim vWeight                     As Variant

Dim state                       As SystemState

With Application.WorksheetFunction
  Set state = New SystemState
  wsI.Cells.EntireColumn.AutoFit
  wsO.Cells.ClearContents
  Set oGetRidofDupes = CreateObject("Scripting.Dictionary")
  i = 1
  Do While wsI.Cells(2, i) <> ""
    i = i + 1
  Loop
  n = (i - 1) \ 2
  vCount = .Transpose(Range(wsI.Cells(3, 1), wsI.Cells(3, n).End(xlDown)))
  vWeight = .Transpose(Range(wsI.Cells(3, n + 1), wsI.Cells(3, 2 * n).End(xlDown)))
  For i = 1 To n
    k = 0
    For j = 1 To UBound(vCount, 2)
      k = k + vCount(j, i)
    Next j
    If k < n Then
      Call MsgBox("Not enough items in column " & i, vbOKOnly, "Error")
      Exit Sub
    End If
  Next i
  m = j - 1
  'Debug.Print "n = " & n, "m = " & m
  'Now we know the dimensions
  ReDim sItem(1 To n) As String
  wsO.Cells(1, 1) = "#"
  wsO.Cells(1, 2) = "Total"
  For i = 1 To n
    sItem(i) = wsI.Cells(2, i)
    wsO.Cells(1, i + 2) = sItem(i)
    wsO.Cells(1, n + 2 + i) = sItem(i) & " count"
    wsO.Cells(1, 2 * n + 2 + i) = sItem(i) & " weight"
  Next i

  ReDim lPermutWeight(1 To n, 1 To n * m) As Long
  ReDim lPermutIdx(1 To n) As Long
  ReDim lPermutSubGroupIdx(1 To n, 1 To n * m) As Long

  For i = 1 To n
    t = 0
    For j = 1 To m
      For k = 1 To .Min(n, vCount(i, j))
        t = t + 1
        lPermutWeight(i, t) = vWeight(i, j)
        lPermutSubGroupIdx(i, t) = j
      Next k
    Next j
    lPermutIdx(i) = t
  Next i

  v1 = Rnd(-1)    'With this initialization we only need 195026 iterations for 33 solutions
  Randomize (1)

  v = 2
```

```
For u = 1 To 500000
  v1 = VBUniqRandInt(3, lPermutIdx(1))
  v2 = VBUniqRandInt(3, lPermutIdx(2))
  v3 = VBUniqRandInt(3, lPermutIdx(3))
  i = v1(1)
  j = v1(2)
  k = v1(3)
  i2 = v2(1)
  j2 = v2(2)
  k2 = v2(3)
  i3 = v3(1)
  j3 = v3(2)
  k3 = v3(3)
  'Debug.Print lPermutWeight(1, i) & " + " & lPermutWeight(2, i2) & " + " & _
   'lPermutWeight(3, i3) & " ?= " & lPermutWeight(1, j) & " + " & lPermutWeight(2, j2) & _
   '" + " & lPermutWeight(3, j3) & " And " & lPermutWeight(1, i) & " + " & _
   'lPermutWeight(2, i2) & " + " & lPermutWeight(3, i3) & " ?= " & lPermutWeight(1, k) & _
   '" + " & lPermutWeight(2, k2) & " + " & lPermutWeight(3, k3)
  If lPermutWeight(1, i) + lPermutWeight(2, i2) + lPermutWeight(3, i3) = _
     lPermutWeight(1, j) + lPermutWeight(2, j2) + lPermutWeight(3, j3) And _
     lPermutWeight(1, i) + lPermutWeight(2, i2) + lPermutWeight(3, i3) = _
     lPermutWeight(1, k) + lPermutWeight(2, k2) + lPermutWeight(3, k3) Then
    If Not oGetRidofDupes.exists(lPermutWeight(1, i) & "|" & lPermutWeight(2, i2) & "|" & _
                                 lPermutWeight(3, i3) & "|" & lPermutWeight(1, j) & "|" & _
                                 lPermutWeight(2, j2) & "|" & lPermutWeight(3, j3) & "|" & _
                                 lPermutWeight(1, k) & "|" & lPermutWeight(2, k2) & "|" & _
                                 lPermutWeight(3, k3)) Then
      oGetRidofDupes(lPermutWeight(1, i) & "|" & lPermutWeight(2, i2) & "|" & _
                     lPermutWeight(3, i3) & "|" & lPermutWeight(1, j) & "|" & _
                     lPermutWeight(2, j2) & "|" & lPermutWeight(3, j3) & "|" & _
                     lPermutWeight(1, k) & "|" & lPermutWeight(2, k2) & "|" & _
                     lPermutWeight(3, k3)) = 1
      oGetRidofDupes(lPermutWeight(1, i) & "|" & lPermutWeight(2, i2) & "|" & _
                     lPermutWeight(3, i3) & "|" & lPermutWeight(1, k) & "|" & _
                     lPermutWeight(2, k2) & "|" & lPermutWeight(3, k3) & "|" & _
                     lPermutWeight(1, j) & "|" & lPermutWeight(2, j2) & "|" & _
                     lPermutWeight(3, j3)) = 1
      oGetRidofDupes(lPermutWeight(1, j) & "|" & lPermutWeight(2, j2) & "|" & _
                     lPermutWeight(3, j3) & "|" & lPermutWeight(1, i) & "|" & _
                     lPermutWeight(2, i2) & "|" & lPermutWeight(3, i3) & "|" & _
                     lPermutWeight(1, k) & "|" & lPermutWeight(2, k2) & "|" & _
                     lPermutWeight(3, k3)) = 1
      oGetRidofDupes(lPermutWeight(1, j) & "|" & lPermutWeight(2, j2) & "|" & _
                     lPermutWeight(3, j3) & "|" & lPermutWeight(1, k) & "|" & _
                     lPermutWeight(2, k2) & "|" & lPermutWeight(3, k3) & "|" & _
                     lPermutWeight(1, i) & "|" & lPermutWeight(2, i2) & "|" & _
                     lPermutWeight(3, i3)) = 1
      oGetRidofDupes(lPermutWeight(1, k) & "|" & lPermutWeight(2, k2) & "|" & _
                     lPermutWeight(3, k3) & "|" & lPermutWeight(1, i) & "|" & _
                     lPermutWeight(2, i2) & "|" & lPermutWeight(3, i3) & "|" & _
                     lPermutWeight(1, j) & "|" & lPermutWeight(2, j2) & "|" & _
                     lPermutWeight(3, j3)) = 1
      oGetRidofDupes(lPermutWeight(1, k) & "|" & lPermutWeight(2, k2) & "|" & _
                     lPermutWeight(3, k3) & "|" & lPermutWeight(1, j) & "|" & _
                     lPermutWeight(2, j2) & "|" & lPermutWeight(3, j3) & "|" & _
                     lPermutWeight(1, i) & "|" & lPermutWeight(2, i2) & "|" & _
                     lPermutWeight(3, i3)) = 1
      wsO.Cells(v, 1) = (v + 1) \ n
      wsO.Cells(v, 2) = lPermutWeight(1, i) + lPermutWeight(2, i2) + lPermutWeight(3, i3)
      wsO.Cells(v, 3) = lPermutWeight(1, i)
      wsO.Cells(v, 4) = lPermutWeight(2, i2)
      wsO.Cells(v, 5) = lPermutWeight(3, i3)
      wsO.Cells(v + 1, 3) = lPermutWeight(1, j)
      wsO.Cells(v + 1, 4) = lPermutWeight(2, j2)
      wsO.Cells(v + 1, 5) = lPermutWeight(3, j3)
      wsO.Cells(v + 2, 3) = lPermutWeight(1, k)
      wsO.Cells(v + 2, 4) = lPermutWeight(2, k2)
      wsO.Cells(v + 2, 5) = lPermutWeight(3, k3)
      wsO.Cells(v, 6) = vCount(1, 1) - IIf(lPermutSubGroupIdx(1, i) = 1, 1, 0) - _
                        IIf(lPermutSubGroupIdx(1, j) = 1, 1, 0) - _
                        IIf(lPermutSubGroupIdx(1, k) = 1, 1, 0)
      wsO.Cells(v, 7) = vCount(2, 1) - IIf(lPermutSubGroupIdx(2, i2) = 1, 1, 0) - _
                        IIf(lPermutSubGroupIdx(2, j2) = 1, 1, 0) - _
                        IIf(lPermutSubGroupIdx(2, k2) = 1, 1, 0)
      wsO.Cells(v, 8) = vCount(3, 1) - IIf(lPermutSubGroupIdx(3, i3) = 1, 1, 0) - _
                        IIf(lPermutSubGroupIdx(3, j3) = 1, 1, 0) - _
                        IIf(lPermutSubGroupIdx(3, k3) = 1, 1, 0)
      wsO.Cells(v + 1, 6) = vCount(1, 2) - IIf(lPermutSubGroupIdx(1, i) = 2, 1, 0) - _
                            IIf(lPermutSubGroupIdx(1, j) = 2, 1, 0) - _
                            IIf(lPermutSubGroupIdx(1, k) = 2, 1, 0)
      wsO.Cells(v + 1, 7) = vCount(2, 2) - IIf(lPermutSubGroupIdx(2, i2) = 2, 1, 0) - _
                            IIf(lPermutSubGroupIdx(2, j2) = 2, 1, 0) - _
                            IIf(lPermutSubGroupIdx(2, k2) = 2, 1, 0)
      wsO.Cells(v + 1, 8) = vCount(3, 2) - IIf(lPermutSubGroupIdx(3, i3) = 2, 1, 0) - _
                            IIf(lPermutSubGroupIdx(3, j3) = 2, 1, 0) - _
                            IIf(lPermutSubGroupIdx(3, k3) = 2, 1, 0)
      wsO.Cells(v + 2, 6) = vCount(1, 3) - IIf(lPermutSubGroupIdx(1, i) = 3, 1, 0) - _
                            IIf(lPermutSubGroupIdx(1, j) = 3, 1, 0) - _
                            IIf(lPermutSubGroupIdx(1, k) = 3, 1, 0)
      wsO.Cells(v + 2, 7) = vCount(2, 3) - IIf(lPermutSubGroupIdx(2, i2) = 3, 1, 0) - _
```

```
                            IIf(lPermutSubGroupIdx(2, j2) = 3, 1, 0) - _
                            IIf(lPermutSubGroupIdx(2, k2) = 3, 1, 0)
        wsO.Cells(v + 2, 8) = vCount(3, 3) - IIf(lPermutSubGroupIdx(3, i3) = 3, 1, 0) - _
                            IIf(lPermutSubGroupIdx(3, j3) = 3, 1, 0) - _
                            IIf(lPermutSubGroupIdx(3, k3) = 3, 1, 0)
        wsO.Cells(v, 9) = vWeight(1, 1)
        wsO.Cells(v, 10) = vWeight(2, 1)
        wsO.Cells(v, 11) = vWeight(3, 1)
        wsO.Cells(v + 1, 9) = vWeight(1, 2)
        wsO.Cells(v + 1, 10) = vWeight(2, 2)
        wsO.Cells(v + 1, 11) = vWeight(3, 2)
        wsO.Cells(v + 2, 9) = vWeight(1, 3)
        wsO.Cells(v + 2, 10) = vWeight(2, 3)
        wsO.Cells(v + 2, 11) = vWeight(3, 3)
        v = v + 3
        If v > 98 Then Exit For
      End If
  End If
Next u
Debug.Print u & " iterations, " & v \ n & " solutions"
wsO.Cells.EntireColumn.AutoFit
End With
End Sub
```

CombinationsWithMinRemainingWeight Programmcode

```
Sub CombinationsWithMinRemainingWeight()

Dim i                           As Long
Dim j                           As Long
Dim k                           As Long
Dim m                           As Long
Dim maxsum                      As Long
Dim n                           As Long
Dim sum(1 To 33)                As Long
Dim t                           As Long
Dim u                           As Long
Dim v                           As Long
Dim w                           As Long

Dim vCount                      As Variant
Dim vC(1 To 33)                 As Variant
Dim vCi(1 To 3)                 As Variant

Dim state                       As SystemState

With Application.WorksheetFunction
Set state = New SystemState
i = 1
Do While wsI.Cells(2, i) <> ""
  i = i + 1
Loop
n = (i - 1) \ 2
vCount = .Transpose(.Transpose(Range(wsI.Cells(3, 1), wsI.Cells(3, n).End(xlDown))))
For i = 1 To n
  k = 0
  For j = 1 To UBound(vCount, 2)
    k = k + vCount(j, i)
  Next j
  If k < n Then
    Call MsgBox("Not enough items in column " & i, vbOKOnly, "Error")
    Exit Sub
  End If
Next i
m = j - 1

i = 2
t = wsO.Cells(i, 1)
Do While t <> 0
  sum(t) = wsO.Cells(i, 2)
  vC(t) = .Transpose(.Transpose(Range(wsO.Cells(i, 6), wsO.Cells(i + 2, 8))))
  i = i + 3
  t = wsO.Cells(i, 1)
Loop

t = 0
maxsum = 0
For i = 1 To 33
  vCi(1) = vC(i)
  For j = 1 To 33
    vCi(2) = vCi(1)
    For m = 1 To 3
      For n = 1 To 3
        If vCi(1)(m, n) < vCount(m, n) - vC(j)(m, n) Then GoTo Label_Next_j
        vCi(2)(m, n) = vCi(1)(m, n) - vCount(m, n) + vC(j)(m, n)
      Next n
```

```
    Next m
    For k = 1 To 33
      vCi(3) = vCi(2)
      For m = 1 To 3
        For n = 1 To 3
          If vCi(2)(m, n) < vCount(m, n) - vC(k)(m, n) Then GoTo Label_Next_k
          vCi(3)(m, n) = vCi(2)(m, n) - vCount(m, n) + vC(k)(m, n)
        Next n
      Next m

      If maxsum <= 3 * (sum(i) + sum(j) + sum(k)) Then
        maxsum = 3 * (sum(i) + sum(j) + sum(k))
        t = t + 1
        Debug.Print t, maxsum, i, j, k
      End If

Label_Next_k:
    Next k
Label_Next_j:
  Next j
Next i

End With

End Sub
```

Minirechner

Dies ist ein kleiner Minirechner mit zwei Registern: ein Programmzähler pc und ein Akkumulator acc. Zwei VBA Programme interpretieren die unterschiedlichen Modi: ein direkter Modus - auch Kommandozeilen Modus genannt, und ein Programmmodus:

Minirechner			**Direkte Eingabe = Kommandozeile:**					**Meldung / Fehlermeldung:**
==========			srt					Programm wird gestartet bei pc = start
Direkter Modus:			**Zeile #**	**Label**	**OpCode**	**Argument**	**Kommentar**	**Ausgabebereich:**
bgn <adr>	pc := <adr>		1	start	bsa	ggt		Der größte gemeinsame Teiler ist:
dbg on|off	Debug Modus ein oder aus		2		out	out_ggt		250
srt	Programmstart		3		out	result_ggt		Das kleinste gemeinsame Vielfache ist:
			4		bsa	kgv		3750
Programmmodus:			5		out	out_kgv		Programmende erreicht in Zeile 7.
add <adr>	acc := acc + <adr>		6		out	result_kgv		
beq <adr>	Springe nach <adr> wenn acc = 0		7		hlt			
bgr <adr>	Springe nach <adr> wenn acc > 0		8	ggt	lda	arg1		
ble <adr>	Springe nach <adr> wenn acc < 0		9		sta	temp1		
bsa <adr>	Sprung zur Subroutine ab <adr>		10		lda	arg2		
bun <adr>	Sprung zu <adr>		11		sta	temp2		
cla	acc := 0		12	ggt_intern	lda	temp1		
dac	acc := acc - 1		13		sub	temp2		
div <adr>	acc := acc / <adr> (ganzzahlige Division!)		14		beq	end_ggt		
hlt	Programmende		15		bgr	store		
iac	acc := acc + 1		16		lda	temp2		
lda <adr>	acc := <adr>		17		sub	temp1		
mul <adr>	acc := acc * <adr>		18		sta	temp2		
out <adr>	Print <adr>		19		bun	ggt_intern		
ret	Return von Subroutine		20	store	sta	temp1		
sta <adr>	<adr> := (acc)		21		bun	ggt_intern		
sub <adr>	acc := acc - <adr>		22	end_ggt	lda	temp1		
			23		sta	result_ggt		
			24		ret			
			25	kgv	lda	arg1		
			26		div	result_ggt		
			27		mul	arg2		
			28		sta	result_kgv		
			29		ret			
			30	temp1		250		
			31	temp2		250		
			32	out_ggt		Der größte gemeinsame Teiler ist:		
			33	out_kgv		Das kleinste gemeinsame Vielfache ist:		
			34	arg1		750		
			35	arg2		1250		
			36	result_ggt		250		
			37	result_kgv		3750		

Das Beispielprogramm berechnet für zwei ganze Zahlen den größten gemeinsamen Teiler und das kleinste gemeinsame Vielfache.

Um das Beispielprogramm auszuführen, starten Sie den Minirechner wie folgt:

- Setzen Sie in der Kommandozeile mit *bgn 1* oder *bgn start* (wenn Sie die Sprungmarke *start* gesetzt haben) den Startpunkt Ihres Programms.
- Gegebenenfalls können Sie mit *dbg on* oder *dbg off* in der Kommandozeile den Debug Modus ein- oder ausschalten.
- Mit srt in der Kommandozeile starten Sie Ihr Programm.

Wenn Sie *dbg on* eingeben, bevor Sie das Beispielprogramm starten, erhalten Sie diese Ausgabe:

```
Ausgabebereich:
Label 'start' := 1
Label 'ggt' := 8
Label 'ggt_intern' := 12
Label 'store' := 20
Label 'end_ggt' := 22
Label 'kgv' := 25
```

```
Label 'temp1' := 30
Label 'temp2' := 31
Label 'out_ggt' := 32
Label 'out_kgv' := 33
Label 'arg1' := 34
Label 'arg2' := 35
Label 'result_ggt' := 36
Label 'result_kgv' := 37
Unterprogrammaufruf 'ggt'. Rücksprung wurde auf Zeile 2 gesetzt. Stackindex 1.
Programmzähler wurde auf Zeile 8 gesetzt.
acc := 750
Argument in Zeile 30 wurde auf acc = 750 gesetzt.
acc := 1250
Argument in Zeile 31 wurde auf acc = 1250 gesetzt.
acc := 750
acc := acc - 1250
acc != 0 -> keine Verzweigung.
acc <= 0 -> keine Verzweigung.
acc := 1250
acc := acc - 750
Argument in Zeile 31 wurde auf acc = 500 gesetzt.
Verzweige nach ggt_intern.
Programmzähler wurde auf Zeile 12 gesetzt.
acc := 750
acc := acc - 500
acc != 0 -> keine Verzweigung.
acc > 0 -> verzweige nach store.
Programmzähler wurde auf Zeile 20 gesetzt.
Argument in Zeile 30 wurde auf acc = 250 gesetzt.
Verzweige nach ggt_intern.
Programmzähler wurde auf Zeile 12 gesetzt.
acc := 250
acc := acc - 500
acc != 0 -> keine Verzweigung.
acc <= 0 -> keine Verzweigung.
acc := 500
acc := acc - 250
Argument in Zeile 31 wurde auf acc = 250 gesetzt.
Verzweige nach ggt_intern.
Programmzähler wurde auf Zeile 12 gesetzt.
acc := 250
acc := acc - 250
acc = 0 -> verzweige nach end_ggt.
Programmzähler wurde auf Zeile 22 gesetzt.
acc := 250
Argument in Zeile 36 wurde auf acc = 250 gesetzt.
Unterprogrammrücksprung nach '2'. Stackindex 0.
Der größte gemeinsame Teiler ist:
250
Unterprogrammaufruf 'kgv'. Rücksprung wurde auf Zeile 5 gesetzt. Stackindex 1.
Programmzähler wurde auf Zeile 25 gesetzt.
acc := 750
acc := acc / 250
acc := acc * 1250
Argument in Zeile 37 wurde auf acc = 3750 gesetzt.
Unterprogrammrücksprung nach '5'. Stackindex 0.
Das kleinste gemeinsame Vielfache ist:
3750
Programmende erreicht in Zeile 7.
```

Der Kommandozeilen Interpreter - Worksheet_Change Programmcode

Dieser Code befindet sich im Tabellenblatt wsMain:

```
Private Sub Worksheet_Change(ByVal Target As Range)
'This implements the command line interpreter of the mini-calculator.
'(C) (P) by Bernd Plumhoff  26-Dec-2023 PB V0.1
Dim s          As String
'Application.EnableEvents = False
If Target.Address = Range("Kommandozeile").Address Then
  s = Range("Kommandozeile")
  Select Case Left(s, 3)
  Case "srt"
    If pc = 0 Or pc = "" Then pc = 1
    Range("Meldung") = "Programm wird gestartet bei pc = " & pc
    Range("Meldung").Font.ColorIndex = xlCIGreen
    Call interpreter
  Case "bgn"
    s = Right(s, Len(s) - 4)
```

```
    pc = s
    Range("Meldung") = "pc := " & s
    Range("Meldung").Font.ColorIndex = xlCIGreen
  Case "dbg"
    s = Right(s, Len(s) - 4)
    Select Case s
    Case "on"
      dbg = True
      Range("Meldung") = "dbg := on"
      Range("Meldung").Font.ColorIndex = xlCIGreen
    Case "off"
      dbg = False
      Range("Meldung") = "dbg := off"
      Range("Meldung").Font.ColorIndex = xlCIGreen
    Case Else
      Range("Meldung") = "Ungültiger Debug Modus '" & s & "'"
      Range("Meldung").Font.ColorIndex = xlCIRed
    End Select
  Case Else
    Range("Meldung") = "Unbekannter Befehl '" & s & "'"
    Range("Meldung").Font.ColorIndex = xlCIRed
  End Select
End If
'Application.EnableEvents = True
End Sub
```

Der Programm Interpreter - Interpreter Programmcode

Dieser Code befindet sich im Modul General:

```
'This implements the main program interpreter of the mini-calculator.
'(C) (P) by Bernd Plumhoff  26-Dec-2023 PB V0.1

Public Enum xlCI 'Excel Color Index
: xlCIBlack = 1: xlCIWhite: xlCIRed: xlCIBrightGreen: xlCIBlue '1 - 5
: xlCIYellow: xlCIPink: xlCITurquoise: xlCIDarkRed: xlCIGreen '6 - 10
: xlCIDarkBlue: xlCIDarkYellow: xlCIViolet: xlCITeal: xlCIGray25 '11 - 15
: xlCIGray50: xlCIPeriwinkle: xlCIPlum: xlCIIvory: xlCILightTurquoise '16 - 20
: xlCIDarkPurple: xlCICoral: xlCIOceanBlue: xlCIIceBlue: xlCILightBrown '21 - 25
: xlCIMagenta2: xlCIYellow2: xlCICyan2: xlCIDarkPink: xlCIDarkBrown '26 - 30
: xlCIDarkTurquoise: xlCISeaBlue: xlCISkyBlue: xlCILightTurquoise2: xlCILightGreen '31 - 35
: xlCILightYellow: xlCIPaleBlue: xlCIRose: xlCILavender: xlCITan '36 - 40
: xlCILightBlue: xlCIAqua: xlCILime: xlCIGold: xlCILightOrange '41 - 45
: xlCIOrange: xlCIBlueGray: xlCIGray40: xlCIDarkTeal: xlCISeaGreen '46 - 50
: xlCIDarkGreen: xlCIGreenBrown: xlCIBrown: xlCIDarkPink2: xlCIIndigo '51 - 55
: xlCIGray80 '56
End Enum

Enum pcol 'Spalten in jeder Programmzeile
  pcol_zeile = 0                    'Zeilennummer
  pcol_label
  pcol_opcode
  pcol_argument
  pcol_kommentar
End Enum

Public dbg              As Boolean      'Debug Modus
Public i                As Integer      'Ausgabebereich Index
Public pc               As Variant      'Programmzähler

Sub interpreter()
Dim b_end               As Boolean      'Programmzeile leer?
Dim p                   As Integer      'Programm Index
Dim r                   As Integer      'Unterprogramm Stack Index
Dim ustack(1 To 100)    As Integer      'Unterprogramm Stack
Dim acc                 As Long         'Akkumulator
Dim st                  As Object       'Symboltabelle (Labels)
Dim op                  As String       'OpCode
Dim s                   As String
Dim v                   As Variant

'Initialisierungen

Range("Ausgabebereich").Resize(65536).ClearContents
i = 0

If pc = "" Then
  pc = 1
  debug_ausgabe ("Programmzähler wurde auf 1 initialisiert.")
End If

'Lade Symboltabelle
Set st = CreateObject("Scripting.Dictionary")
```

```
p = 1
b_end = (Range("Programmcode").Offset(p, pcol_label) = "" And _
         Range("Programmcode").Offset(p, pcol_opcode) = "" And _
         Range("Programmcode").Offset(p, pcol_argument) = "")
Do Until b_end
  s = Range("Programmcode").Offset(p, pcol_label)
  If s <> "" Then
    If st.exists(s) Then
      Call debug_ausgabe("Identische Label '" & s & "' in Zeilen " & st(s) & " und " & p & ". Abbruch!",
True)
      Exit Sub
    End If
    st(s) = p
    debug_ausgabe ("Label '" & s & "' := " & p)
  End If
  p = p + 1
  b_end = (Range("Programmcode").Offset(p, pcol_label) = "" And _
           Range("Programmcode").Offset(p, pcol_opcode) = "" And _
           Range("Programmcode").Offset(p, pcol_argument) = "")
Loop

'Interpretiere das Programm

Do

continue_do:

  If Not IsNumeric(pc) Then
    If st.exists(pc) Then
      pc = st(pc)
      debug_ausgabe ("Programmzähler wurde auf Zeile " & pc & " gesetzt.")

    Else
      Call debug_ausgabe("Programmzähler hat ungültiges Label '" & pc & "'. Abbruch!", True)
      Exit Sub
    End If
  End If

  op = Range("Programmcode").Offset(pc, pcol_opcode)
  Select Case op
  Case "add"
    v = Range("Programmcode").Offset(pc, pcol_argument)
    If Not IsNumeric(v) Then
      If st.exists(v) Then
        v = st(v)
      Else
        Call debug_ausgabe("Unbekanntes Argument '" & v & "' in Zeile " & pc & ". Abbruch!", True)
        Exit Sub
      End If
    End If
    acc = acc + Range("Programmcode").Offset(v, pcol_argument)
    debug_ausgabe ("acc := acc + " & Range("Programmcode").Offset(v, pcol_argument))
  Case "beq"
    If acc = 0 Then
      pc = Range("Programmcode").Offset(pc, pcol_argument)
      debug_ausgabe ("acc = 0 -> verzweige nach " & pc & ".")
      GoTo continue_do
    Else
      debug_ausgabe ("acc != 0 -> keine Verzweigung.")
    End If
  Case "bgr"
    If acc > 0 Then
      pc = Range("Programmcode").Offset(pc, pcol_argument)
      debug_ausgabe ("acc > 0 -> verzweige nach " & pc & ".")
      GoTo continue_do
    Else
      debug_ausgabe ("acc <= 0 -> keine Verzweigung.")
    End If
  Case "ble"
    If acc < 0 Then
      pc = Range("Programmcode").Offset(pc, pcol_argument)
      debug_ausgabe ("acc < 0 -> verzweige nach " & pc & ".")
      GoTo continue_do
    Else
      debug_ausgabe ("acc >= 0 -> keine Verzweigung.")
      End If
  Case "bsa"
    r = r + 1
    ustack(r) = pc + 1
    pc = Range("Programmcode").Offset(pc, pcol_argument)
    debug_ausgabe ("Unterprogrammaufruf '" & pc & _
        "'. Rücksprung wurde auf Zeile " & ustack(r) & _
        " gesetzt. Stackindex " & r & ".")
    GoTo continue_do
  Case "bun"
    pc = Range("Programmcode").Offset(pc, pcol_argument)
    debug_ausgabe ("Verzweige nach " & pc & ".")
    GoTo continue_do
  Case "cla"
    acc = 0
```

```
    debug_ausgabe ("acc := 0")
  Case "dac"
    acc = acc - 1
    debug_ausgabe ("acc := acc - 1")
  Case "div"
    v = Range("Programmcode").Offset(pc, pcol_argument)
    If Not IsNumeric(v) Then
      If st.exists(v) Then
        v = st(v)
      Else
        Call debug_ausgabe("Unbekanntes Argument '" & v & "' in Zeile " & pc & ". Abbruch!", True)
        Exit Sub
      End If
    End If
    acc = acc / Range("Programmcode").Offset(v, pcol_argument)
    debug_ausgabe ("acc := acc / " & Range("Programmcode").Offset(v, pcol_argument))
  Case "hlt"
    Call debug_ausgabe("Programmende erreicht in Zeile " & pc & ".", True)
    Exit Sub
  Case "iac"
    acc = acc + 1
    debug_ausgabe ("acc := acc + 1")
  Case "lda"
    v = Range("Programmcode").Offset(pc, pcol_argument)
    If Not IsNumeric(v) Then
      If st.exists(v) Then
        v = st(v)
      Else
        debug_ausgabe ("Unbekanntes Argument '" & v & "' in Zeile " & pc & ". Abbruch!")
        End
      End If
    End If
    acc = Range("Programmcode").Offset(v, pcol_argument)
    debug_ausgabe ("acc := " & acc)
  Case "mul"
    v = Range("Programmcode").Offset(pc, pcol_argument)
    If Not IsNumeric(v) Then
      If st.exists(v) Then
        v = st(v)
      Else
        debug_ausgabe ("Unbekanntes Argument '" & v & "' in Zeile " & pc & ". Abbruch!")
        End
      End If
    End If
    acc = acc * Range("Programmcode").Offset(v, pcol_argument)
    debug_ausgabe ("acc := acc * " & Range("Programmcode").Offset(v, pcol_argument))
  Case "out"
    v = Range("Programmcode").Offset(pc, pcol_argument)
    If Not IsNumeric(v) Then
      If st.exists(v) Then
        v = st(v)
      Else
        debug_ausgabe ("Unbekanntes Argument '" & v & "' in Zeile " & pc & ". Abbruch!")
        End
      End If
    End If
    Range("Ausgabebereich").Offset(i) = Range("Programmcode").Offset(v, pcol_argument)
    i = i + 1
  Case "ret"
    pc = ustack(r)
    r = r - 1
    debug_ausgabe ("Unterprogrammrücksprung nach '" & pc & _
        "'. Stackindex " & r & ".")
    GoTo continue_do
  Case "sta"
    v = Range("Programmcode").Offset(pc, pcol_argument)
    If Not IsNumeric(v) Then
      If st.exists(v) Then
        v = st(v)
      Else
        Call debug_ausgabe("Unbekanntes Argument '" & v & "' in Zeile " & pc & ". Abbruch!", True)
        Exit Sub
      End If
    End If
    Range("Programmcode").Offset(v, pcol_argument) = acc
    debug_ausgabe ("Argument in Zeile " & v & " wurde auf acc = " & acc & " gesetzt.")
  Case "sub"
    v = Range("Programmcode").Offset(pc, pcol_argument)
    If Not IsNumeric(v) Then
      If st.exists(v) Then
        v = st(v)
      Else
        Call debug_ausgabe("Unbekanntes Argument '" & v & "' in Zeile " & pc & ". Abbruch!", True)
        Exit Sub
      End If
    End If
    acc = acc - Range("Programmcode").Offset(v, pcol_argument)
    debug_ausgabe ("acc := acc - " & Range("Programmcode").Offset(v, pcol_argument))
  Case Else
    Call debug_ausgabe("Ungültiger OpCode '" & op & "' in Zeile " & pc & ". Abbruch!", True)
```

```
    Exit Sub
  End Select

  pc = pc + 1

  b_end = (Range("Programmcode").Offset(pc, pcol_label) = "" And _
           Range("Programmcode").Offset(pc, pcol_opcode) = "" And _
           Range("Programmcode").Offset(pc, pcol_argument) = "")

Loop Until b_end

End Sub

Sub debug_ausgabe(s As String, Optional force As Boolean = False)
If dbg Or force Then
  Range("Ausgabebereich").Offset(i) = s
  i = i + 1
End If
End Sub
```

Finanzmathematik – Optionen

Ein klassisches Buch zur Bewertung von Optionen ist:
Les Clewlow and Chris Strickland: Implementing Derivatives Models (ISBN 0471966517)

Leider enthält dieses Buch eine Vielzahl von Fehlern, und die 8 Seiten lange Fehlerliste vom 23. November 2000 ist leider nicht mehr öffentlich verfügbar. Einige Fehlerkorrekturen für die erste Auflage:

Page	Row	Correction
xvi	9, column 3	0v should be C
12	15	Si,j = Su^jd^(**n**-j)
24	2 from top in figure 2.12	set coeffi**c**ients
70	4 from bottom in figure 3.13	for j =Nj-2 downto -N**j+1** do
70	Insert before last row of figure 3.13	C[1,-Nj] = C[1,-Nj + 1] - lambda_L
75	4 from bottom in figure 3.16	for j =Nj-2 downto -N**j+1** do
75	Insert before last row of figure 3.16	C[1,-Nj] = C[1,-Nj + 1] - lambda_L
85	2 from bottom in figure 4.2	SD = sqrt((-sum_CT2 + sum_CT * sum_CT / M) **)** * exp(-2*r*T) / (M - 1)
89	2 from bottom in figure 4.5	SD = sqrt((-sum_CT2 + sum_CT * sum_CT / M) **)** * exp(-2*r*T) / (M - 1)
99	Delete row 9 from top	lnS = ln(S) = 4.6052 **<- delete this row**
113	13 from top in figure 4.19	**Insert before for j loop** V1 = sig1*sig1; V2 = sig2*sig2
113	18 from top in figure 4.19	**Insert before for i loop** Vt1 = V1; Vt2 = V2
118	Last row	a = **m/N *** ln(Gt) ...
120	11 from bottom in figure 4.24	G = productSt **^(1**/N)

Ich implementierte einige der im Buch vorgestellten Algorithmen mit Excel VBA.

Das Binomialbaummodell

Chapter 2 - The Binomial Method

Multiplicative Binomial Tree Valuation of a European Call

K	T	S	r	N	u	d	Function Result	Correct Result
100	1	100	0,06	3	1,1	0,9091	10,14546215	10,14546215

Multiplicative Binomial Tree Valuation of an American Put

K	T	S	r	N	u	d	Function Result	Correct Result
100	1	100	0,06	3	1,1	0,9091	4,654346769	4,654346769

General Additive Binomial Valuation of a European Call

K	T	S	sig	r	N	Function Result	Correct Result
100	1	100	0,2	0,06	3	11,59199121	11,59199121

General Additive Binomial Valuation of an American Put

K	T	S	sig	r	N	Function Result	Correct Result
100	1	100	0,2	0,06	3	6,162109199	6,162109199

Equal Jump General Additive Binomial Valuation of an American Put

K	T	S	sig	r	N	Function Result	Correct Result
100	1	100	0,2	0,06	3	6,162109199	6,162109199

General Additive Binomial Valuation of an American Down-and-Out Call

K	T	S	sig	r	H	N	Function Result	Correct Result
100	1	100	0,2	0,06	95	3	9,995775103	9,995775103

American Spread Call Option by Two-variable Binomial

K	T	S1	S2	sig1	sig2	div1	div2	rho	r	N	Function Result	Correct Result
1	1	100	100	0,2	0,3	0,03	0,04	0,5	0,06	3	10,04478653	10,04478653

Chapter 2 Programmcode

```
Function European_Call_MBTV( _
           K As Double, _
           T As Double, _
           S As Double, _
           r As Double, _
           N As Long, _
           u As Double, _
           d As Double) As Double

'Multiplicative Binomial Tree Valuation of a European Call
'Clewlow/Strickland: Implementing Derivatives Models, page 13
'ISBN 0-471-96651-7

Dim dt                      As Double
Dim p                       As Double
Dim disc                    As Double
ReDim St(0 To N) As Double
ReDim C(0 To N) As Double
Dim i As Long, j            As Long

'precompute constants

dt = T / N
p = (Exp(r * dt) - d) / (u - d)
disc = Exp(-r * dt)

'Debug.Print dt, p, disc
'initialise asset prices at maturity time step N

St(0) = S * d ^ N
For j = 1 To N
  St(j) = St(j - 1) * u / d
Next j

'Debug.Print St(0), St(1), St(2), St(3)
'initialise option values at maturity

For j = 0 To N
  C(j) = St(j) - K: If C(j) < 0# Then C(j) = 0#
Next j

'Debug.Print C(0), C(1), C(2), C(3)
'step back through the tree

For i = N - 1 To 0 Step -1
  For j = 0 To i
    C(j) = disc * (p * C(j + 1) + (1 - p) * C(j))
    'Debug.Print C(j);
  Next j
  'Debug.Print
Next i

European_Call_MBTV = C(0)

End Function

Function American_Put_MBTV( _
           K As Double, _
           T As Double, _
           S As Double, _
           r As Double, _
           N As Long, _
           u As Double, _
           d As Double) As Double

'Multiplicative Binomial Tree Valuation of an American Put
'Clewlow/Strickland: Implementing Derivatives Models, page 15

Dim dt                      As Double
Dim p                       As Double
Dim disc                    As Double
ReDim St(0 To N) As Double
ReDim C(0 To N) As Double
Dim i As Long, j            As Long

'precompute constants

dt = T / N
p = (Exp(r * dt) - d) / (u - d)
disc = Exp(-r * dt)

'Debug.Print dt, p, disc

'initialise asset prices at maturity time step N

St(0) = S * d ^ N
```

```
For j = 1 To N
  St(j) = St(j - 1) * u / d
Next j

'Debug.Print St(0), St(1), St(2), St(3)

'initialise option values at maturity

For j = 0 To N
  C(j) = K - St(j): If C(j) < 0# Then C(j) = 0#
Next j

'Debug.Print C(0), C(1), C(2), C(3)

'step back through the tree applying the early exercise condition

For i = N - 1 To 0 Step -1
  For j = 0 To i
    C(j) = disc * (p * C(j + 1) + (1 - p) * C(j))
    St(j) = St(j) / d
    If C(j) < K - St(j) Then C(j) = K - St(j)
    'Debug.Print C(j);
  Next j
  'Debug.Print
Next i

American_Put_MBTV = C(0)

End Function

Function European_Call_GABV( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           N As Long) As Double

'General Additive Binomial Valuation of a European Call
'Clewlow/Strickland: Implementing Derivatives Models, page 22

Dim dt                      As Double
Dim pu As Double, pd        As Double
Dim dxu As Double, dxd As Double, nu As Double
Dim disc                    As Double
ReDim St(0 To N) As Double
ReDim C(0 To N) As Double
Dim i As Long, j            As Long

'set coefficients - Trigeorgis

dt = T / N
nu = r - 0.5 * sig ^ 2#
dxu = Sqr(sig ^ 2# * dt + (nu * dt) ^ 2#)
dxd = -dxu
pu = 0.5 + 0.5 * (nu * dt / dxu)
pd = 1# - pu

'precompute constants

disc = Exp(-r * dt)

'initialise asset prices at maturity N

St(0) = S * Exp(N * dxd)
For j = 1 To N
  St(j) = St(j - 1) * Exp(dxu - dxd)
Next j
'Debug.Print St(0), St(1), St(2), St(3)

'initialise option values at maturity

For j = 0 To N
  C(j) = St(j) - K: If C(j) < 0# Then C(j) = 0#
Next j
'Debug.Print C(0), C(1), C(2), C(3)

'step back through the tree

For i = N - 1 To 0 Step -1
  For j = 0 To i
    C(j) = disc * (pu * C(j + 1) + pd * C(j))
    'Debug.Print C(j);
  Next j
  'Debug.Print
Next i

European_Call_GABV = C(0)

End Function
```

```
Function American_Put_GABV( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           N As Long) As Double

'General Additive Binomial Valuation of an American Put
'Clewlow/Strickland: Implementing Derivatives Models, page 24

Dim dt                      As Double
Dim pu As Double, pd        As Double
Dim dxu As Double, dxd As Double, nu As Double
Dim disc                    As Double
ReDim St(0 To N) As Double
ReDim C(0 To N) As Double
Dim i As Long, j            As Long

'set coefficients - Trigeorgis

dt = T / N
nu = r - 0.5 * sig ^ 2#
dxu = Sqr(sig ^ 2 * dt + (nu * dt) ^ 2#)
dxd = -dxu
pu = 0.5 + 0.5 * (nu * dt / dxu)
pd = 1# - pu

'precompute constants

disc = Exp(-r * dt)

'initialise asset prices at maturity N

St(0) = S * Exp(N * dxd)
For j = 1 To N
  St(j) = St(j - 1) * Exp(dxu - dxd)
Next j
'Debug.Print St(0), St(1), St(2), St(3)

'initialise option values at maturity

For j = 0 To N
  C(j) = K - St(j): If C(j) < 0# Then C(j) = 0#
Next j
'Debug.Print C(0), C(1), C(2), C(3)

'step back through the tree applying the early exercise condition

For i = N - 1 To 0 Step -1
  For j = 0 To i
    C(j) = disc * (pu * C(j + 1) + pd * C(j))
    'adjust asset price to current time step
    St(j) = St(j) / Exp(dxd)
    'apply the early exercise condition
    If K - St(j) > C(j) Then C(j) = K - St(j)
    'Debug.Print C(j);
  Next j
  'Debug.Print
Next i

American_Put_GABV = C(0)

End Function

Function American_Put_EJGABV( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           N As Long) As Double

'Equal Jump General Additive Binomial Valuation of an American Put
'Clewlow/Strickland: Implementing Derivatives Models, page 28

Dim dt                      As Double
Dim pu As Double, pd        As Double
Dim dx As Double, nu        As Double
Dim disc As Double, edx     As Double
Dim dpu As Double, dpd      As Double
ReDim St(-N To N) As Double
ReDim C(-N To N) As Double
Dim i As Long, j            As Long

'set coefficients - Trigeorgis

dt = T / N
nu = r - 0.5 * sig ^ 2#
```

```
dx = Sqr(sig ^ 2# * dt + (nu * dt) ^ 2#)
pu = 0.5 + 0.5 * (nu * dt / dx)
pd = 1# - pu

'precompute constants

disc = Exp(-r * dt)
dpu = disc * pu
dpd = disc * pd
edx = Exp(dx)

'initialise asset prices at maturity N

St(-N) = S * Exp(-N * dx)
For j = -N + 1 To N
  St(j) = St(j - 1) * edx
Next j

'initialise option values at maturity

For j = -N To N Step 2
  C(j) = K - St(j): If C(j) < 0# Then C(j) = 0#
Next j

'step back through the tree applying the early exercise condition

For i = N - 1 To 0 Step -1
  For j = -i To i Step 2
    C(j) = dpd * C(j - 1) + dpu * C(j + 1)
    'apply the early exercise condition
    If K - St(j) > C(j) Then C(j) = K - St(j)
  Next j
Next i

American_Put_EJGABV = C(0)

End Function

Function American_Down_and_Out_Call_GABV( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           H As Double, _
           N As Long) As Double

'General Additive Binomial Valuation of an American Down-and-Out Call
'Clewlow/Strickland: Implementing Derivatives Models, page 41

Dim dt                          As Double
Dim pu As Double, pd            As Double
Dim dpu As Double, dpd          As Double
Dim dxd As Double, dxu As Double, nu As Double
Dim edxud                       As Double
Dim edxd                        As Double
Dim disc                        As Double
ReDim St(0 To N) As Double
ReDim C(0 To N) As Double
Dim i As Long, j                As Long

'set coefficients - Trigeorgis

dt = T / N
nu = r - 0.5 * sig ^ 2#
dxu = Sqr(sig ^ 2# * dt + (nu * dt) ^ 2#)
dxd = -dxu
pu = 0.5 + 0.5 * (nu * dt / dxu)
pd = 1# - pu

'precompute constants

disc = Exp(-r * dt)
dpu = disc * pu
dpd = disc * pd
edxud = Exp(dxu - dxd)
edxd = Exp(dxd)

'initialise asset prices at maturity N

St(0) = S * Exp(N * dxd)
For j = 1 To N
  St(j) = St(j - 1) * edxud
Next j

'initialise option values at maturity

For j = 0 To N
  If St(j) > H Then
    C(j) = St(j) - K: If C(j) < 0# Then C(j) = 0#
```

```
  Else
    C(j) = 0#
  End If
Next j

'step back through the tree applying the barrier
'and early exercise condition

For i = N - 1 To 0 Step -1
  For j = 0 To i
    'adjust asset price to current time step
    St(j) = St(j) / edxd

    If St(j) > H Then
      C(j) = dpd * C(j) + dpu * C(j + 1)

      'apply the early exercise condition
      If C(j) < St(j) - K Then C(j) = St(j) - K
    Else
      C(j) = 0#
    End If
  Next j
Next i

American_Down_and_Out_Call_GABV = C(0)

End Function

Function American_Spread_Call_Option_bTvB( _
           K As Double, _
           T As Double, _
           S1 As Double, _
           S2 As Double, _
           sig1 As Double, _
           sig2 As Double, _
           div1 As Double, _
           div2 As Double, _
           rho As Double, _
           r As Double, _
           N As Long) As Double

'American Spread Call Option by Two-variable Binomial
'Clewlow/Strickland: Implementing Derivatives Models, page 46

Dim dt                        As Double
Dim puu As Double, pdu        As Double
Dim pud As Double, pdd        As Double
Dim dx1 As Double, dx2        As Double
Dim nu1 As Double, nu2        As Double
Dim edx1 As Double, edx2      As Double
Dim disc                      As Double
ReDim s1t(-N To N) As Double
ReDim s2t(-N To N) As Double
ReDim C(-N To N, -N To N) As Double
Dim i As Long, j As Long, k1 As Long

'precompute constants

dt = T / N
nu1 = r - div1 - 0.5 * sig1 ^ 2#
nu2 = r - div2 - 0.5 * sig2 ^ 2#
dx1 = sig1 * Sqr(dt)
dx2 = sig2 * Sqr(dt)
disc = Exp(-r * dt)
puu = (dx1 * dx2 + (dx2 * nu1 + dx1 * nu2 + rho * sig1 * sig2) * dt) / _
      (4 * dx1 * dx2) * disc
pud = (dx1 * dx2 + (dx2 * nu1 - dx1 * nu2 - rho * sig1 * sig2) * dt) / _
      (4 * dx1 * dx2) * disc
pdu = (dx1 * dx2 - (dx2 * nu1 - dx1 * nu2 + rho * sig1 * sig2) * dt) / _
      (4 * dx1 * dx2) * disc
pdd = (dx1 * dx2 - (dx2 * nu1 + dx1 * nu2 - rho * sig1 * sig2) * dt) / _
      (4 * dx1 * dx2) * disc
edx1 = Exp(dx1)
edx2 = Exp(dx2)

'initialise asset prices at time step N

s1t(-N) = S1 * Exp(-N * dx1)
s2t(-N) = S2 * Exp(-N * dx2)
For j = -N + 1 To N
  s1t(j) = s1t(j - 1) * edx1
  s2t(j) = s2t(j - 1) * edx2
Next j

'initialise option values at maturity

For j = -N To N Step 2
  For k1 = -N To N Step 2
    C(j, k1) = s1t(j) - s2t(k1) - K
    If C(j, k1) < 0# Then C(j, k1) = 0#
```

```
  Next k1
Next j

'step back through the tree applying the early exercise condition

For i = N - 1 To 0 Step -1
  For j = -i To i Step 2
    For k1 = -i To i Step 2
      C(j, k1) = pdd * C(j - 1, k1 - 1) + pud * C(j + 1, k1 - 1) + _
                 pdu * C(j - 1, k1 + 1) + puu * C(j + 1, k1 + 1)
      'apply the early exercise condition
      If s1t(j) - s2t(k1) - K > C(j, k1) Then C(j, k1) = s1t(j) - s2t(k1) - K
    Next k1
  Next j
Next i

American_Spread_Call_Option_bTvB = C(0, 0)

End Function
```

Das Trinomial-Optionspreismodell und die Methode der Finiten Differenzen

Chapter 3 - Trinomial Trees and Finite Difference Methods

European Call Option by Trinomial Tree

K	T	S	sig	r	div	N	dx	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	3	0,2	8,425336177	8,425336177

European Call Option by Explicit Finite Difference Method

K	T	S	sig	r	div	N	Nj	dx	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	3	3	0,2	8,545479558	8,545479558

American Put Option by Explicit Finite Difference Method

K	T	S	sig	r	div	N	Nj	dx	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	3	3	0,2	6,005828214	6,005828214

American Put Option by Implicit Finite Difference Method

K	T	S	sig	r	div	N	Nj	dx	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	3	3	0,2	4,922117823	4,922117823

American Put Option by Crank-Ncolson Finite Difference Method

K	T	S	sig	r	div	N	Nj	dx	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	3	3	0,2	5,418377337	5,418377337

Chapter 3 Programmcode

```
Function European_Call_Option_bTT( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           div As Double, _
           N As Long, _
           dx As Double) As Double

'European Call Option by Trinomial Tree
'Clewlow/Strickland: Implementing Derivatives Models, page 54

Dim dt                      As Double
Dim pu As Double, pm As Double, pd As Double
Dim nu As Double, edx       As Double
Dim disc                    As Double
ReDim St(-N To N) As Double
ReDim C(0 To N, -N To N) As Double
Dim i As Long, j            As Long

'set coefficients - Trigeorgis

dt = T / N
nu = r - div - 0.5 * sig ^ 2#
edx = Exp(dx)
pu = 0.5 * ((sig ^ 2# * dt + nu ^ 2# * dt ^ 2#) / dx ^ 2# + nu * dt / dx)
pm = 1# - (sig ^ 2# * dt + nu ^ 2# * dt ^ 2#) / dx ^ 2#
pd = 0.5 * ((sig ^ 2# * dt + nu ^ 2# * dt ^ 2#) / dx ^ 2# - nu * dt / dx)
disc = Exp(-r * dt)

'initialise asset prices at maturity N
```

```
St(-N) = S * Exp(-N * dx)
For j = -N + 1 To N
  St(j) = St(j - 1) * edx
Next j

'initialise option values at maturity

For j = -N To N
  If St(j) - K > 0# Then
    C(N, j) = St(j) - K
  Else
    C(N, j) = 0#
  End If
Next j

'step back through lattice

For i = N - 1 To 0 Step -1
  For j = -i To i
    C(i, j) = disc * (pu * C(i + 1, j + 1) + pm * C(i + 1, j) + pd * C(i + 1, j - 1))
  Next j
Next i

European_Call_Option_bTT = C(0, 0)

End Function


Function European_Call_Option_bEFDM( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           div As Double, _
           N As Long, _
           Nj As Long, _
           dx As Double) As Double

'European Call Option by Explicit Finite Difference Method
'Clewlow/Strickland: Implementing Derivatives Models, page 60

Dim dt                      As Double
Dim pu As Double, pm As Double, pd As Double
Dim nu As Double, edx       As Double
ReDim St(-Nj To Nj) As Double
ReDim C(0 To N, -Nj To Nj) As Double
Dim i As Long, j            As Long

'set coefficients - Trigeorgis

dt = T / N
nu = r - div - 0.5 * sig ^ 2#
edx = Exp(dx)
pu = 0.5 * dt * ((sig / dx) ^ 2# + nu / dx)
pm = 1# - dt * ((sig / dx) ^ 2# + r)
pd = 0.5 * dt * ((sig / dx) ^ 2# - nu / dx)

'initialise asset prices at maturity N

St(-Nj) = S * Exp(-Nj * dx)
For j = -Nj + 1 To Nj
  St(j) = St(j - 1) * edx
Next j

'initialise option values at maturity

For j = -Nj To Nj
  If St(j) - K > 0# Then
    C(N, j) = St(j) - K
  Else
    C(N, j) = 0#
  End If
Next j

'step back through lattice

For i = N - 1 To 0 Step -1
  For j = -Nj + 1 To Nj - 1
    C(i, j) = pu * C(i + 1, j + 1) + pm * C(i + 1, j) + pd * C(i + 1, j - 1)
  Next j
  'boundary conditions
  C(i, -Nj) = C(i, -Nj + 1)
  C(i, Nj) = C(i, Nj - 1) + St(Nj) - St(Nj - 1)
Next i

European_Call_Option_bEFDM = C(0, 0)

End Function
```

```
Function American_Put_Option_bEFDM( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           div As Double, _
           N As Long, _
           Nj As Long, _
           dx As Double) As Double

'American Put Option by Explicit Finite Difference Method
'Clewlow/Strickland: Implementing Derivatives Models, page 62

Dim dt                      As Double
Dim pu As Double, pm As Double, pd As Double
Dim nu As Double, edx       As Double
ReDim St(-Nj To Nj) As Double
ReDim C(0 To 1, -Nj To Nj) As Double
Dim i As Long, j            As Long

'Precompute constants

dt = T / N
nu = r - div - 0.5 * sig ^ 2#
edx = Exp(dx)
pu = 0.5 * dt * ((sig / dx) ^ 2# + nu / dx)
pm = 1# - dt * (sig / dx) ^ 2# - r * dt
pd = 0.5 * dt * ((sig / dx) ^ 2# - nu / dx)

'Initialise asset prices at maturity N

St(-Nj) = S * Exp(-Nj * dx)
For j = -Nj + 1 To Nj
  St(j) = St(j - 1) * edx
Next j

'Initialise option values at maturity

For j = -Nj To Nj
  If K - St(j) > 0# Then
    C(0, j) = K - St(j)
  Else
    C(0, j) = 0#
  End If
Next j

'Step back through lattice

For i = N - 1 To 0 Step -1

  For j = -Nj + 1 To Nj - 1
    C(1, j) = pu * C(0, j + 1) + pm * C(0, j) + pd * C(0, j - 1)
  Next j

  'Boundary conditions

  C(1, -Nj) = C(1, -Nj + 1) + St(-Nj + 1) - St(-Nj)
  C(1, Nj) = C(1, Nj - 1)

  'Apply early exercise condition

  For j = -Nj To Nj
    C(0, j) = K - St(j)
    If C(0, j) < C(1, j) Then C(0, j) = C(1, j)
  Next j

Next i

American_Put_Option_bEFDM = C(0, 0)

End Function


Function American_Put_Option_bIFDM( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           div As Double, _
           N As Long, _
           Nj As Long, _
           dx As Double) As Double

'American Put Option by Implicit Finite Difference Method
'Clewlow/Strickland: Implementing Derivatives Models, page 69

Dim dt                      As Double
```

```
Dim pu As Double, pm As Double, pd As Double
Dim nu As Double, edx        As Double
Dim lambda_L As Double, lambda_U As Double
ReDim St(-Nj To Nj) As Double
ReDim C(0 To 1, -Nj To Nj) As Double
Dim i As Long, j             As Long
ReDim pmp(-Nj To Nj) As Double
ReDim pp(-Nj To Nj) As Double

'Precompute constants

dt = T / N
nu = r - div - 0.5 * sig ^ 2#
edx = Exp(dx)
pu = -0.5 * dt * ((sig / dx) ^ 2# + nu / dx)
pm = 1# + dt * (sig / dx) ^ 2# + r * dt
pd = -0.5 * dt * ((sig / dx) ^ 2# - nu / dx)

'Initialise asset prices at maturity N

St(-Nj) = S * Exp(-Nj * dx)
For j = -Nj + 1 To Nj
  St(j) = St(j - 1) * edx
Next j

'Initialise option values at maturity

For j = -Nj To Nj
  If K - St(j) > 0# Then
    C(0, j) = K - St(j)
  Else
    C(0, j) = 0#
  End If
Next j

'Compute derivative boundary condition

lambda_L = St(-Nj) - St(-Nj + 1)
lambda_U = 0#

'Step back through lattice

For i = N - 1 To 0 Step -1

  'Solve implicit tridiagonal system

  'Substitute boundary condition at j = -Nj into j = -Nj + 1

  pmp(-Nj + 1) = pm + pd
  pp(-Nj + 1) = C(0, -Nj + 1) + pd * lambda_L

  'Eliminate upper diagonal

  For j = -Nj + 2 To Nj - 1

    pmp(j) = pm - pu * pd / pmp(j - 1)
    pp(j) = C(0, j) - pp(j - 1) * pd / pmp(j - 1)

  Next j

  'Use boundary condition at j = Nj and equation at j = Nj - 1

  C(1, Nj) = (pp(Nj - 1) + pmp(Nj - 1) * lambda_U) / (pu + pmp(Nj - 1))
  C(1, Nj - 1) = C(1, Nj) - lambda_U

  'Back-substitution

  For j = Nj - 2 To -Nj + 1 Step -1
    C(1, j) = (pp(j) - pu * C(1, j + 1)) / pmp(j)
  Next j
  C(1, -Nj) = C(1, -Nj + 1) - lambda_L

  'Apply early exercise condition

  For j = -Nj To Nj
    C(0, j) = K - St(j)
    If C(0, j) < C(1, j) Then C(0, j) = C(1, j)
  Next j

Next i

American_Put_Option_bIFDM = C(0, 0)

End Function


Function American_Put_Option_bCNFDM( _
           K As Double, _
           T As Double, _
```

```
            S As Double, _
            sig As Double, _
            r As Double, _
            div As Double, _
            N As Long, _
            Nj As Long, _
            dx As Double) As Double

'American Put Option by Crank-Nicolson Finite Difference Method
'Clewlow/Strickland: Implementing Derivatives Models, page 74

Dim dt                      As Double
Dim pu As Double, pm As Double, pd As Double
Dim nu As Double, edx       As Double
Dim lambda_L As Double, lambda_U As Double
ReDim St(-Nj To Nj) As Double
ReDim C(0 To 1, -Nj To Nj) As Double
Dim i As Long, j            As Long
ReDim pmp(-Nj To Nj) As Double
ReDim pp(-Nj To Nj) As Double

'Precompute constants

dt = T / N
nu = r - div - 0.5 * sig ^ 2#
edx = Exp(dx)
pu = -0.25 * dt * ((sig / dx) ^ 2# + nu / dx)
pm = 1# + 0.5 * dt * (sig / dx) ^ 2# + 0.5 * r * dt
pd = -0.25 * dt * ((sig / dx) ^ 2# - nu / dx)

'Initialise asset prices at maturity N

St(-Nj) = S * Exp(-Nj * dx)
For j = -Nj + 1 To Nj
  St(j) = St(j - 1) * edx
Next j

'Initialise option values at maturity

For j = -Nj To Nj
  If K - St(j) > 0# Then
    C(0, j) = K - St(j)
  Else
    C(0, j) = 0#
  End If
Next j

'Compute derivative boundary condition
lambda_L = St(-Nj) - St(-Nj + 1)
lambda_U = 0#

'Step back through lattice

For i = N - 1 To 0 Step -1

  'Solve Crank-Nicolson tridiagonal system

  'Substitute boundary condition at j = -Nj into j = -Nj + 1
  pmp(-Nj + 1) = pm + pd
  pp(-Nj + 1) = -pu * C(0, -Nj + 2) - (pm - 2) * C(0, -Nj + 1) - pd * C(0, -Nj) + pd * lambda_L

  'Eliminate upper diagonal
  For j = -Nj + 2 To Nj - 1

    pmp(j) = pm - pu * pd / pmp(j - 1)
    pp(j) = -pu * C(0, j + 1) - (pm - 2) * C(0, j) - pd * C(0, j - 1) - pp(j - 1) * pd / pmp(j - 1)

  Next j

  'Use boundary condition at j = Nj and equation at j = Nj - 1
  C(1, Nj) = (pp(Nj - 1) + pmp(Nj - 1) * lambda_U) / (pu + pmp(Nj - 1))
  C(1, Nj - 1) = C(1, Nj) - lambda_U

  'Back-substitution
  For j = Nj - 2 To -Nj + 1 Step -1
    C(1, j) = (pp(j) - pu * C(1, j + 1)) / pmp(j)
  Next j
  C(1, -Nj) = C(1, -Nj + 1) - lambda_L

  'Apply early exercise condition
  For j = -Nj To Nj
    C(0, j) = K - St(j)
    If C(0, j) < C(1, j) Then C(0, j) = C(1, j)
  Next j

Next i

American_Put_Option_bCNFDM = C(0, 0)

End Function
```

Chapter 4 - Monte Carlo Simulation

European Call Option by Monte Carlo with Black-Scholes

K	T	S	sig	r	div	N	M	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	10	100000	9,151602315	~9,1

European Call Option by Monte Carlo with Black-Scholes and Antithetic Variance Reduction

K	T	S	sig	r	div	N	M	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	10	100000	9,11952827	~9,1

European Call Option by Monte Carlo with Black-Scholes and Delta-based Control Variate

K	T	S	sig	r	div	N	M	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	10	100000	9,143949534	~9,1

European Call Option by Monte Carlo with Black-Scholes and Antithetic and Delta-based Control Variates

K	T	S	sig	r	div	N	M	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	10	100000	9,143077108	~9,1

European Call Option by Monte Carlo with Black-Scholes and Antithetic, Delta- and Gamma- based Control Variates

K	T	S	sig	r	div	N	M	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	10	100000	9,138332376	~9,1

European Spread Call Option by Monte Carlo with Black-Scholes

K	T	S1	S2	sig1	sig2	div1	div2	rho	r	N	M	Function Result	Correct Result
1	1	100	110	0,2	0,3	0,03	0,04	0,5	0,06	1	100000	6,479866322	~6,5

European Spread Call Option by Monte Carlo with Black-Scholes and Stochastic Volatilities

K	T	S1	S2	sig1	sig2	div1	div2	rhoz				Function Result	Correct Result
1	1	100	110	0,2	0,3	0,03	0,04	1	0,5	0,2	0,01	7,54370877	~7,0389
alpha1	alpha2	Vbar1	Vbar2	xi1	xi2	r		0,5	1	0,01	0,3		
1	2	0,04	0,09	0,05	0,06	0,06		0,2	0,01	1	0,3		
N	M							0,01	0,3	0,3	1		
10	10000												

European Down and Out Call Option by Monte Carlo with Black-Scholes

K	T	S	sig	r	div	H	N	M	Function Result	Correct Result
100	1	100	0,2	0,06	0,03	99	10	100000	5,013266976	~5

Monte Carlo Simulation Programmcode

```
#Const MODULE_TEST = False    'Used for optional compiling of
'debugging/test parts - set to FALSE for production

Function European_Call_bMCBS( _
            K As Double, _
            T As Double, _
            S As Double, _
            sig As Double, _
            r As Double, _
            div As Double, _
            N As Long, _
            M As Long) As Double

'European Call Option by Monte Carlo with Black-Scholes
'Clewlow/Strickland: Implementing Derivatives Models, page 85
'12.04.2021 corrected: sum_CT2 = sum_CT2 + CT * CT

Dim dt                      As Double
Dim nudt As Double, sigsdt  As Double
Dim lnS As Double, lnSt     As Double
Dim sum_CT As Double, sum_CT2 As Double
Dim e                       As Double
Dim St As Double, CT        As Double
Dim SD As Double, SE        As Double
Dim i As Long, j            As Long

'Precompute constants

dt = T / N
nudt = (r - div - 0.5 * sig ^ 2#) * dt
sigsdt = sig * Sqr(dt)
lnS = Log(S)

sum_CT = 0#
sum_CT2 = 0#

For j = 1 To M

  'For each simulation
```

```
  lnSt = lnS

  For i = 1 To N

    'For each time step

    '        e = Application.WorksheetFunction.NormSInv(Rnd()) 'Standard normal sample
    e = Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + _
        Rnd() + Rnd() + Rnd() + Rnd() - 6#    'Standard normal sample
    lnSt = lnSt + nudt + sigsdt * e    'Evolve the stock price

  Next i

  St = Exp(lnSt)
  CT = St - K: If CT < 0# Then CT = 0#
  sum_CT = sum_CT + CT
  sum_CT2 = sum_CT2 + CT * CT

Next j

SD = Sqr((sum_CT2 - sum_CT * sum_CT / M) * Exp(-2# * r * T) / (M - 1))    'Standard deviation
SE = SD / Sqr(M)    'Standard error
European_Call_bMCBS = sum_CT / M * Exp(-r * T)

End Function

Function European_Call_bMCBS_AVR( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           div As Double, _
           N As Long, _
           M As Long) As Double

'European Call Option by Monte Carlo with Black-Scholes and Antithetic Variance Reduction
'Clewlow/Strickland: Implementing Derivatives Models, page 89

Dim dt                      As Double
Dim nudt As Double, sigsdt  As Double
Dim lnS As Double, LnSt1 As Double, LnSt2 As Double
Dim sum_CT As Double, sum_CT2 As Double
Dim e                       As Double
Dim St1 As Double, St2      As Double
Dim CT As Double, CT1 As Double, CT2 As Double
Dim SD As Double, SE        As Double
Dim i As Long, j            As Long

'Precompute constants

dt = T / N
nudt = (r - div - 0.5 * sig ^ 2#) * dt
sigsdt = sig * Sqr(dt)
lnS = Log(S)

sum_CT = 0#
sum_CT2 = 0#

For j = 1 To M

  'For each simulation

  LnSt1 = lnS
  LnSt2 = lnS

  For i = 1 To N

    'For each time step

    '        e = Application.WorksheetFunction.NormSInv(Rnd()) 'Standard normal sample
    e = Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + _
        Rnd() + Rnd() + Rnd() + Rnd() - 6#    'Standard normal sample
    LnSt1 = LnSt1 + nudt + sigsdt * e    'Evolve the stock price
    LnSt2 = LnSt2 + nudt + sigsdt * -e    'Evolve the stock price

  Next i

  St1 = Exp(LnSt1)
  St2 = Exp(LnSt2)
  CT1 = St1 - K: If CT1 < 0# Then CT1 = 0#
  CT2 = St2 - K: If CT2 < 0# Then CT2 = 0#
  CT = (CT1 + CT2) / 2#
  sum_CT = sum_CT + CT
  sum_CT2 = sum_CT + CT * CT

Next j

SD = Sqr(sum_CT * sum_CT / M - sum_CT2) * Exp(-2# * r * T) / (M - 1)    'Standard deviation
```

```
SE = SD / Sqr(M)     'Standard error
European_Call_bMCBS_AVR = sum_CT / M * Exp(-r * T)

End Function

Function European_Call_bMCBS_DCV( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
           r As Double, _
           div As Double, _
           N As Long, _
           M As Long) As Double

'European Call Option by Monte Carlo with Black-Scholes and Delta-based Control Variate
'Clewlow/Strickland: Implementing Derivatives Models, page 97

Dim dt                      As Double
Dim nudt As Double, sigsdt As Double, erddt As Double
Dim sum_CT As Double, sum_CT2 As Double
Dim e                       As Double
Dim St As Double, Stn       As Double
Dim CT                      As Double
Dim SD As Double, SE        As Double
Dim beta1 As Double, t1     As Double
Dim cv                      As Double
Dim delta1                  As Double
Dim i As Long, j            As Long

'Dim eps As Variant
'eps = Array(0.5391, 0.1692, 0.4583, 0.4164, 0.9223, 0.1597, 0.4011, -0.2382, 1.2152, -0.5506)

'Precompute constants

dt = T / N
nudt = (r - div - 0.5 * sig ^ 2#) * dt
sigsdt = sig * Sqr(dt)
erddt = Exp((r - div) * dt)
beta1 = -1#
'Debug.Print "dt = " & dt & ", nudt = " & nudt & ", sigsdt = " & sigsdt & _
  '  ", erddt = " & erddt & ", beta1 = " & beta1

sum_CT = 0#
sum_CT2 = 0#

For j = 1 To M

  'For each simulation

  St = S
  cv = 0#

  For i = 1 To N

    'For each time step

    t1 = CDbl(i - 1) * dt
    delta1 = Black_Scholes_Delta(St, t1, K, T, sig, r, div)
    '        Debug.Print "e = " & e & ", St = " & St & ", delta1 = " & delta1 & ", cv = " & cv
    '        e = Application.WorksheetFunction.NormSInv(Rnd()) 'Standard normal sample
    e = Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + _
        Rnd() + Rnd() + Rnd() + Rnd() - 6#    'Standard normal sample
    '        e = eps(i - 1)
    Stn = St * Exp(nudt + sigsdt * e)
    cv = cv + delta1 * (Stn - St * erddt)    '* exp(T - (t1 + dt)) 'inflation factor *exp can be omitted
    St = Stn

  Next i

  '    Debug.Print "e = " & e & ", St = " & St & ", delta1 = " & delta1 & ", cv = " & cv
  CT = St - K: If CT < 0# Then CT = 0#
  CT = CT + beta1 * cv
  sum_CT = sum_CT + CT
  sum_CT2 = sum_CT + CT * CT
  '    Debug.Print "CT = " & CT & ", CT2 = " & CT * CT

Next j

SD = Sqr(sum_CT * sum_CT / M - sum_CT2) * Exp(-2# * r * T) / (M - 1)    'Standard deviation
SE = SD / Sqr(M)     'Standard error
European_Call_bMCBS_DCV = sum_CT / M * Exp(-r * T)

End Function

Function European_Call_bMCBS_ADCV( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
```

```
           r As Double, _
           div As Double, _
           N As Long, _
           M As Long) As Double

'European Call Option by Monte Carlo with Black-Scholes and Antithetic and Delta-based Control Variates
'Clewlow/Strickland: Implementing Derivatives Models, page 100

Dim dt                      As Double
Dim nudt As Double, sigsdt As Double, erddt As Double
Dim sum_CT As Double, sum_CT2 As Double
Dim e                       As Double
Dim St1 As Double, Stn1     As Double
Dim St2 As Double, Stn2     As Double
Dim CT As Double, CT1 As Double, CT2 As Double
Dim SD As Double, SE        As Double
Dim beta1 As Double, t1     As Double
Dim cv1 As Double, cv2      As Double
Dim delta1 As Double, delta2 As Double
Dim i As Long, j            As Long

'Dim eps As Variant
'eps = Array(0.5391, 0.1692, 0.4583, 0.4164, 0.9223, 0.1597, 0.4011, -0.2382, 1.2152, -0.5506)

'Precompute constants

dt = T / N
nudt = (r - div - 0.5 * sig ^ 2#) * dt
sigsdt = sig * Sqr(dt)
erddt = Exp((r - div) * dt)
beta1 = -1#
'Debug.Print "dt = " & dt & ", nudt = " & nudt & ", sigsdt = " & sigsdt & _
  '", erddt = " & erddt & ", beta1 = " & beta1

sum_CT = 0#
sum_CT2 = 0#

For j = 1 To M

  'For each simulation

  St1 = S
  St2 = S
  cv1 = 0#
  cv2 = 0#

  For i = 1 To N

    'For each time step

    t1 = CDbl(i - 1) * dt
    delta1 = Black_Scholes_Delta(St1, t1, K, T, sig, r, div)
    delta2 = Black_Scholes_Delta(St2, t1, K, T, sig, r, div)
    '        Debug.Print "e = " & e & ", St = " & St & ", delta1 = " & delta1 & ", cv = " & cv
    '        e = Application.WorksheetFunction.NormSInv(Rnd()) 'Standard normal sample
    e = Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + _
        Rnd() + Rnd() + Rnd() + Rnd() - 6#    'Standard normal sample
    '        e = eps(i - 1)
    Stn1 = St1 * Exp(nudt + sigsdt * e)
    Stn2 = St2 * Exp(nudt + sigsdt * -e)
    cv1 = cv1 + delta1 * (Stn1 - St1 * erddt)    '* exp(T - (t1 + dt)) 'inflation factor *exp can be
omitted
    cv2 = cv2 + delta2 * (Stn2 - St2 * erddt)    '* exp(T - (t1 + dt)) 'inflation factor *exp can be
omitted
    St1 = Stn1
    St2 = Stn2

  Next i

  '    Debug.Print "e = " & e & ", St = " & St & ", delta1 = " & delta1 & ", cv = " & cv
  CT1 = St1 - K: If CT1 < 0# Then CT1 = 0#
  CT2 = St2 - K: If CT2 < 0# Then CT2 = 0#
  CT = (CT1 + beta1 * cv1 + CT2 + beta1 * cv2) / 2#
  sum_CT = sum_CT + CT
  sum_CT2 = sum_CT + CT * CT
  '    Debug.Print "CT = " & CT & ", CT2 = " & CT * CT

Next j

SD = Sqr(sum_CT * sum_CT / M - sum_CT2) * Exp(-2# * r * T) / (M - 1)    'Standard deviation
SE = SD / Sqr(M)    'Standard error
European_Call_bMCBS_ADCV = sum_CT / M * Exp(-r * T)

End Function

Function European_Call_bMCBS_ADGCV( _
           K As Double, _
           T As Double, _
           S As Double, _
           sig As Double, _
```

```
            r As Double, _
            div As Double, _
            N As Long, _
            M As Long) As Double

'European Call Option by Monte Carlo with Black-Scholes and Antithetic, Delta-
'and Gamma-based Control Variates
'Clewlow/Strickland: Implementing Derivatives Models, page 102

Dim dt                      As Double
Dim nudt As Double, sigsdt  As Double
Dim erddt As Double, egamma As Double
Dim sum_CT As Double, sum_CT2 As Double
Dim e                       As Double
Dim St1 As Double, Stn1     As Double
Dim St2 As Double, Stn2     As Double
Dim CT As Double, CT1 As Double, CT2 As Double
Dim SD As Double, SE        As Double
Dim beta1 As Double, beta2 As Double, t1 As Double
Dim cv1 As Double, cv2      As Double
Dim delta1 As Double, delta2 As Double
Dim gamma1 As Double, gamma2 As Double
Dim i As Long, j            As Long

'Dim eps As Variant
'eps = Array(0.5391, 0.1692, 0.4583, 0.4164, 0.9223, 0.1597, 0.4011, -0.2382, 1.2152, -0.5506)

'Precompute constants

dt = T / N
nudt = (r - div - 0.5 * sig ^ 2#) * dt
sigsdt = sig * Sqr(dt)
erddt = Exp((r - div) * dt)
egamma = Exp((2# * (r - div) + sig * sig) * dt) - 2# * erddt + 1#
beta1 = -1#
beta2 = -0.5
'Debug.Print "dt = " & dt & ", nudt = " & nudt & ", sigsdt = " & sigsdt & _
  '", erddt = " & erddt & ", beta1 = " & beta1

sum_CT = 0#
sum_CT2 = 0#

For j = 1 To M

  'For each simulation

  St1 = S
  St2 = S
  cv1 = 0#
  cv2 = 0#

  For i = 1 To N

    'For each time step

    'Compute hedge sensitivities
    t1 = CDbl(i - 1) * dt
    delta1 = Black_Scholes_Delta(St1, t1, K, T, sig, r, div)
    delta2 = Black_Scholes_Delta(St2, t1, K, T, sig, r, div)
    gamma1 = Black_Scholes_Gamma(St1, t1, K, T, sig, r, div)
    gamma2 = Black_Scholes_Gamma(St2, t1, K, T, sig, r, div)

    'Evolve asset prices
    '        Debug.Print "e = " & e & ", St = " & St & ", delta1 = " & delta1 & ", cv = " & cv
    '        e = Application.WorksheetFunction.NormSInv(Rnd()) 'Standard normal sample
    e = Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + _
        Rnd() + Rnd() + Rnd() + Rnd() - 6#    'Standard normal sample
    '        e = eps(i - 1)
    Stn1 = St1 * Exp(nudt + sigsdt * e)
    Stn2 = St2 * Exp(nudt + sigsdt * -e)

    'Accumulate control variates
    cv1 = cv1 + delta1 * (Stn1 - St1 * erddt) + _
          delta2 * (Stn2 - St2 * erddt)
    cv2 = cv2 + gamma1 * ((Stn1 - St1) ^ 2 - St1 ^ 2 * egamma) + _
          gamma2 * ((Stn2 - St2) ^ 2 - St2 ^ 2 * egamma)
    St1 = Stn1
    St2 = Stn2

  Next i

  '    Debug.Print "e = " & e & ", St = " & St & ", delta1 = " & delta1 & ", cv = " & cv
  CT1 = St1 - K: If CT1 < 0# Then CT1 = 0#
  CT2 = St2 - K: If CT2 < 0# Then CT2 = 0#
  CT = (CT1 + CT2 + beta1 * cv1 + beta2 * cv2) / 2#
  sum_CT = sum_CT + CT
  sum_CT2 = sum_CT + CT * CT
  '    Debug.Print "CT = " & CT & ", CT2 = " & CT * CT

Next j
```

```
SD = Sqr(sum_CT * sum_CT / M - sum_CT2) * Exp(-2# * r * T) / (M - 1)    'Standard deviation
SE = SD / Sqr(M)    'Standard error
European_Call_bMCBS_ADGCV = sum_CT / M * Exp(-r * T)

End Function

Function European_Spread_bMCBS( _
           K As Double, _
           T As Double, _
           S1 As Double, _
           S2 As Double, _
           sig1 As Double, _
           sig2 As Double, _
           div1 As Double, _
           div2 As Double, _
           rho As Double, _
           r As Double, _
           N As Long, _
           M As Long) As Double

'European Spread Option by Monte Carlo with Black-Scholes
'Clewlow/Strickland: Implementing Derivatives Models, page 110

Dim dt                      As Double
Dim nu1dt As Double, nu2dt  As Double
Dim sig1sdt As Double, sig2sdt As Double
Dim sum_CT As Double, sum_CT2 As Double
Dim e1 As Double, e2        As Double
Dim z1 As Double, z2        As Double
Dim St1 As Double, St2      As Double
Dim CT                      As Double
Dim SD As Double, SE        As Double
Dim srho                    As Double
Dim i As Long, j            As Long

'Precompute constants

dt = T / N
nu1dt = (r - div1 - 0.5 * sig1 ^ 2#) * dt
nu2dt = (r - div2 - 0.5 * sig2 ^ 2#) * dt
sig1sdt = sig1 * Sqr(dt)
sig2sdt = sig2 * Sqr(dt)
srho = Sqr(1# - rho ^ 2#)

sum_CT = 0#
sum_CT2 = 0#

For j = 1 To M

  'For each simulation

  St1 = S1
  St2 = S2

  For i = 1 To N

    'For each time step

    e1 = Application.WorksheetFunction.NormSInv(Rnd())    'Standard normal sample
    e2 = Application.WorksheetFunction.NormSInv(Rnd())    'Standard normal sample
    '       e1 = Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + _
            '            Rnd() + Rnd() + Rnd() + Rnd() - 6# 'Standard normal sample
    '       e2 = Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + _
            '            Rnd() + Rnd() + Rnd() + Rnd() - 6# 'Standard normal sample
    z1 = e1
    z2 = rho * e1 + srho * e2
    St1 = St1 * Exp(nu1dt + sig1sdt * z1)
    St2 = St2 * Exp(nu2dt + sig2sdt * z2)

  Next i

  CT = St1 - St2 - K: If CT < 0# Then CT = 0#
  sum_CT = sum_CT + CT
  sum_CT2 = sum_CT + CT * CT

Next j

SD = Sqr(sum_CT * sum_CT / M - sum_CT2) * Exp(-2# * r * T) / (M - 1)    'Standard deviation
SE = SD / Sqr(M)    'Standard error
European_Spread_bMCBS = sum_CT / M * Exp(-r * T)

End Function

Function European_Spread_bMCBS_SV( _
           K As Double, _
           T As Double, _
           S1 As Double, _
           S2 As Double, _
           sig1 As Double, _
```

```
            sig2 As Double, _
            div1 As Double, _
            div2 As Double, _
            alpha1 As Double, _
            alpha2 As Double, _
            Vbar1 As Double, _
            Vbar2 As Double, _
            xi1 As Double, _
            xi2 As Double, _
            rhoz As Variant, _
            r As Double, _
            N As Long, _
            M As Long) As Double

'European Spread Option by Monte Carlo with Black-Scholes and Stochastic Volatilities
'Clewlow/Strickland: Implementing Derivatives Models, page 113

Dim dt                      As Double
Dim xi1sdt As Double, xi2sdt As Double
Dim alpha1dt As Double, alpha2dt As Double
Dim sum_CT As Double, sum_CT2 As Double
Dim LnS1 As Double, LnS2    As Double
Dim LnSt1 As Double, LnSt2  As Double
Dim v1 As Double, v2        As Double
Dim vt1 As Double, vt2      As Double
Dim z1 As Double, z2        As Double
Dim St1 As Double, St2      As Double
Dim CT                      As Double
Dim SD As Double, SE        As Double
Dim sdt                     As Double
Dim i As Long, j            As Long
Dim z                       As Variant

'Precompute constants

dt = T / N
alpha1dt = alpha1 * dt
alpha2dt = alpha2 * dt
xi1sdt = xi1 * Sqr(dt)
xi2sdt = xi2 * Sqr(dt)
sdt = Sqr(dt)
LnS1 = Log(S1)
LnS2 = Log(S2)
v1 = sig1 * sig1
v2 = sig2 * sig2

sum_CT = 0#
sum_CT2 = 0#

For j = 1 To M

  'For each simulation

  LnSt1 = LnS1
  LnSt2 = LnS2
  vt1 = v1
  vt2 = v2

  'Generate correlated normals
  z = RandCorr(N, rhoz)

  For i = 1 To N    'For each time step

    'Simulate variances first
    vt1 = vt1 + alpha1dt * (Vbar1 - vt1) + xi1sdt * Sqr(vt1) * z(i, 3)
    vt2 = vt2 + alpha2dt * (Vbar2 - vt2) + xi2sdt * Sqr(vt2) * z(i, 4)

    'Simulate asset prices
    LnSt1 = LnSt1 + (r - div1 - vt1 / 2#) * dt + Sqr(vt1) * sdt * z(i, 1)
    LnSt2 = LnSt2 + (r - div2 - vt2 / 2#) * dt + Sqr(vt2) * sdt * z(i, 2)

  Next i

  St1 = Exp(LnSt1)
  St2 = Exp(LnSt2)

  CT = St1 - St2 - K: If CT < 0# Then CT = 0#
  sum_CT = sum_CT + CT
  sum_CT2 = sum_CT + CT * CT

Next j

SD = Sqr(sum_CT * sum_CT / M - sum_CT2) * Exp(-2# * r * T) / (M - 1)    'Standard deviation
SE = SD / Sqr(M)    'Standard error
European_Spread_bMCBS_SV = sum_CT / M * Exp(-r * T)

End Function

Function European_Down_and_Out_Call_bMCBS( _
          K As Double, _
```

```
            T As Double, _
            S As Double, _
            sig As Double, _
            r As Double, _
            div As Double, _
            H As Long, _
            N As Long, _
            M As Long) As Double

'European Down and Out Call Option by Monte Carlo with Black-Scholes
'Clewlow/Strickland: Implementing Derivatives Models, page 116

Dim dt                          As Double
Dim nudt As Double, sigsdt      As Double
Dim lnS As Double, lnSt         As Double
Dim sum_CT As Double, sum_CT2 As Double
Dim e                           As Double
Dim St As Double, CT            As Double
Dim SD As Double, SE            As Double
Dim i As Long, j                As Long
Dim bBarrierCrossed             As Boolean

#If MODULE_TEST Then
  e = Rnd(-1)
  Randomize 1    'maybe even additional parameter to function
#End If

'Precompute constants
dt = T / N
nudt = (r - div - sig * sig / 2#) * dt
sigsdt = sig * Sqr(dt)

sum_CT = 0#
sum_CT2 = 0#

For j = 1 To M

  'For each simulation
  St = S
  bBarrierCrossed = False

  For i = 1 To N

    'For each time step
    '   e = Application.WorksheetFunction.NormSInv(Rnd()) 'Standard normal sample
    e = Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + _
        Rnd() + Rnd() + Rnd() + Rnd() - 6#    'Standard normal sample
    St = St * Exp(nudt + sigsdt * e)    'Evolve the stock price
    If St <= H Then
      bBarrierCrossed = True
      Exit For
    End If

  Next i

  If bBarrierCrossed Then
    CT = 0#
  Else
    CT = St - K
    If CT < 0# Then CT = 0#
  End If

  sum_CT = sum_CT + CT
  sum_CT2 = sum_CT + CT * CT

Next j

SD = Sqr(Abs(sum_CT * sum_CT / M - sum_CT2)) * Exp(-2# * r * T) / (M - 1)    'Standard deviation
SE = SD / Sqr(M)    'Standard error
European_Down_and_Out_Call_bMCBS = sum_CT / M * Exp(-r * T)

End Function

Function European_Arithmetic_Asian_Call_bMCwGAOCV( _
            K As Double, _
            T As Variant, _
            S As Double, _
            sig As Double, _
            r As Double, _
            div As Double, _
            N As Long, _
            M As Long) As Double

'European Arithmetic Asian Call Option by Monte Carlo
'with a Geometric Asian Call Option Control Variate
'Clewlow/Strickland: Implementing Derivatives Models, page 120

Dim sumSt As Double, productSt As Double
Dim sum_CT As Double, sum_CT2 As Double
Dim e                           As Double
```

```
Dim SD As Double, SE          As Double
Dim St As Double, CT          As Double
Dim A As Double, G            As Double
Dim i As Long, j              As Long

ReDim nudt(0 To N) As Double
ReDim sigsdt(0 To N) As Double

'Precompute constants

For i = 1 To N
  'For each fixing
  nudt(i) = (r - div - sig * sig / 2#) * (T(i) - T(i - 1))
  sigsdt(i) = sig * Sqr(T(i) - T(i - 1))
Next i

sum_CT = 0#
sum_CT2 = 0#

For j = 1 To M

  'For each simulation
  St = S
  sumSt = 0#
  productSt = 1#

  For i = 1 To N

    'For each fixing
    '       e = Application.WorksheetFunction.NormSInv(Rnd()) 'Standard normal sample
    e = Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + Rnd() + _
        Rnd() + Rnd() + Rnd() + Rnd() - 6#    'Standard normal sample
    St = St * Exp(nudt(i) + sigsdt(i) * e)    'Evolve the stock price
    sumSt = sumSt + St
    productSt = productSt * St

  Next i

  A = sumSt / N
  G = productSt ^ (1# / N)
  CT = A - K: If CT < 0# Then CT = 0#
  If G > K Then CT = CT - G + K
  sum_CT = sum_CT + CT
  sum_CT2 = sum_CT + CT * CT

Next j

SD = Sqr(sum_CT * sum_CT / M - sum_CT2) * Exp(-2# * r * T) / (M - 1)    'Standard deviation
SE = SD / Sqr(M)    'Standard error
European_Arithmetic_Asian_Call_bMCwGAOCV = sum_CT / M * Exp(-r * T)
'+ Geometric_Asian_Call(K, T, S, sig, r, div, N)

End Function

Function Black_Scholes_Delta( _
           St As Double, _
           t1 As Double, _
           K As Double, _
           T As Double, _
           sig As Double, _
           r As Double, _
           div As Double) As Double

With Application.WorksheetFunction
  Black_Scholes_Delta = Exp(-div * (T - t1)) * .NormSDist((Log(St / K) + _
                                                    (r - div + sig * sig / 2#) * (T - t1)) / (sig *
Sqr(T - t1)))
End With

End Function

Function Black_Scholes_Gamma( _
           St As Double, _
           t1 As Double, _
           K As Double, _
           T As Double, _
           sig As Double, _
           r As Double, _
           div As Double) As Double

With Application.WorksheetFunction
  Black_Scholes_Gamma = Exp(-div * (T - t1)) * _
                        .NormDist((Log(St / K) + (r - div + sig * sig / 2#) * _
                                  (T - t1)) / (sig * Sqr(T - t1)), _
                                 0, 1, False) / (St * sig * Sqr(T - t1))
End With

End Function

Function RandCorr(N As Long, vVarCovar As Variant) As Variant
```

```
'Returns Ubound(vVarCovar,1) correlated random number vectors of length n.
'vVarCovar is a square matrix containing the variance/covariance matrix.
'Please notice that you will only get a "proxy" correlation, not an exact one.
'Reverse("moc.LiborPlus.www") V0.2 PB 06-Nov-2009
Dim vA As Variant, d         As Double
Dim i As Long, j As Long, K As Long, M As Long

With Application.WorksheetFunction
  vA = .Transpose(.Transpose(vVarCovar))
  M = UBound(vA, 1)
  If M <> UBound(vA, 2) Then
    RandCorr = CVErr(xlErrRef)
    Exit Function
  End If

  ReDim Db(1 To M, 1 To M) As Double
  For j = 1 To M
    d = 0#
    For K = 1 To j - 1
      d = d + Db(j, K) * Db(j, K)
    Next K
    Db(j, j) = vA(j, j) - d
    If Db(j, j) <= 0# Then
      RandCorr = CVErr(xlErrNum)
      Exit Function
    End If
    Db(j, j) = Sqr(Db(j, j))

    For i = j + 1 To M
      d = 0#
      For K = 1 To j - 1
        d = d + Db(i, K) * Db(j, K)
      Next K
      Db(i, j) = (vA(i, j) - d) / Db(j, j)
    Next i
  Next j

  ReDim vR(1 To N, 1 To M) As Variant
  For i = 1 To N
    For j = 1 To M
      vR(i, j) = .NormSInv(Rnd())
    Next j
  Next i
  vR = .MMult(vR, Db)
  RandCorr = vR
End With
End Function
```

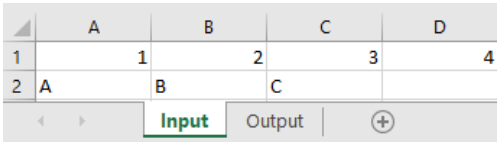
Erzeuge alle Permutationen eines Arrays – *Quickperm*

Wenn Sie alle Permutationen eines Arrays erzeugen wollen, empfehle ich den Algorithmus Quickperm. Es ist einer der effizientesten Permutationsalgorithmen. Er basiert auf dem dem Austausch (engl. Swap) einzelner Elemente und wurde durch Heap Sort inspiriert.

Die hier vorgestellte Variante ist die CountDown Variante. Für die Implementierung in Excel / VBA wurden alle Indizes um 1 erhöht, damit Arrays mit dem Index 1 beginnen.

Beispiel

Eingabe:



	A	B	C	D
1	1	2	3	4
2	A	B	C	

Input | Output

Ausgabe:

	A	B	C	D
1	1	2	3	4
2	2	1	3	4
3	3	1	2	4
4	1	3	2	4
5	2	3	1	4
6	3	2	1	4
7	3	2	4	1
8	2	3	4	1
9	4	3	2	1
10	3	4	2	1
11	2	4	3	1
12	4	2	3	1
13	4	1	3	2
14	1	4	3	2
15	3	4	1	2
16	4	3	1	2
17	1	3	4	2
18	3	1	4	2
19	2	1	4	3
20	1	2	4	3
21	4	2	1	3
22	2	4	1	3
23	1	4	2	3
24	4	1	2	3
25	A	B	C	
26	B	A	C	
27	C	A	B	
28	A	C	B	
29	B	C	A	
30	C	B	A	

Input | Output

Links

Quickperm:

Quickperm – Die originale Website: https://www.quickperm.org/

Baeldung – Generate All Permutations of an Array. Ein hilfreiches Tutorial (engl.) welches in Quickperm mündet: https://www.baeldung.com/cs/array-generate-all-permutations

Permutationen mit Nebenbedingungen:

Rosetta Code – Permutations with some identical elements: https://rosettacode.org/wiki/Permutations_with_some_identical_elements

Computer Science – Enumerating all partial permutations of given length in lexicographic order: https://cs.stackexchange.com/questions/133662/enumerating-all-partial-permutations-of-given-length-in-lexicographic-order

Quickperm Programmcode

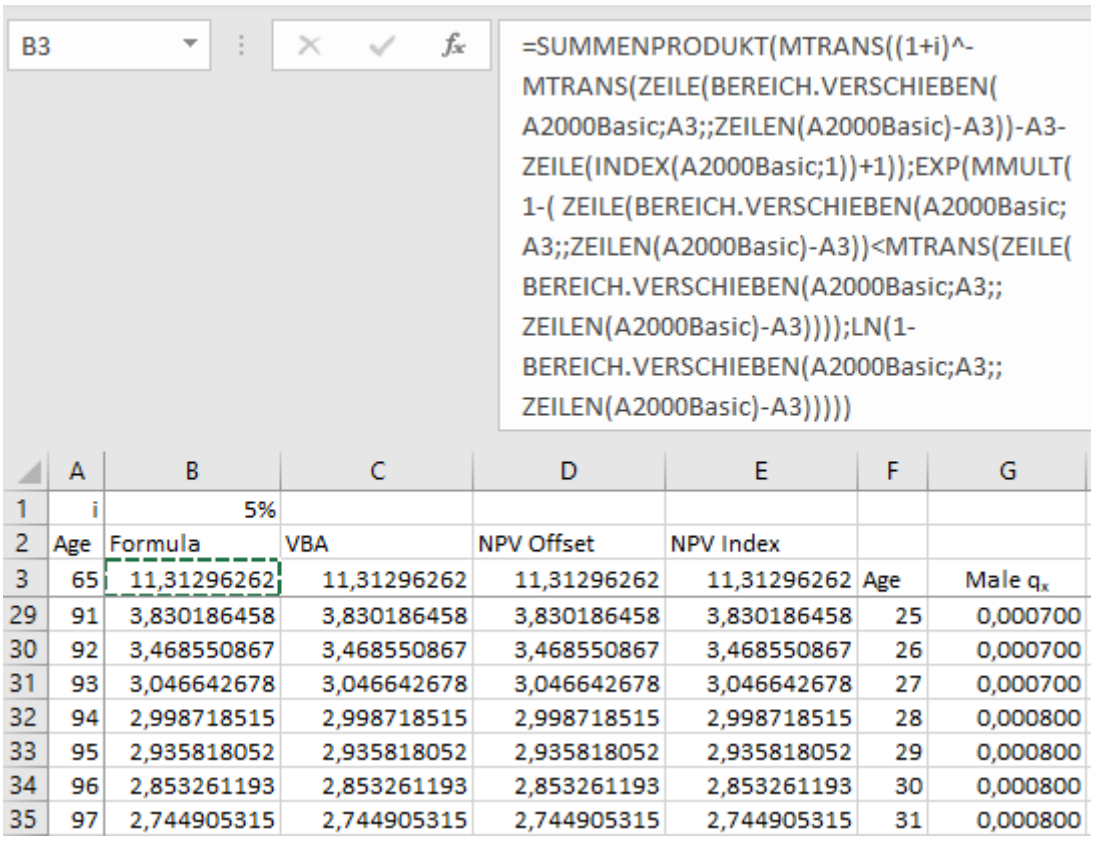
```
Public r As Long 'Output row

Sub QuickPerm(a As Variant)
'Generates all permutations of array a.
'Quickperm is one of the most efficient permutation algorithms.
'It is based on swapping and inspired by Heap sort.
'Countdown variant, indexes increased by 1.
'Algorithm originally from https://www.quickperm.org, migrated to VBA.
'See also: https://www.baeldung.com/cs/array-generate-all-permutations
'Version 0.1 28-Jun-2024
Dim i As Long, idx As Long, j As Long, n As Long, v As Variant
With Application.WorksheetFunction
a = .Transpose(.Transpose(a)) 'Assuming horizontal range
Call VisitPerm(a)
n = UBound(a)
ReDim p(1 To n + 1) As Long
For i = 1 To n + 1: p(i) = i: Next i 'Initialize p()
idx = 2
Do While idx < n + 1
  p(idx) = p(idx) - 1
  If idx Mod 2 = 0 Then
    j = p(idx)
  Else
    j = 1
  End If
  v = a(j): a(j) = a(idx): a(idx) = v 'Swap a(j) and a(idx)
  Call VisitPerm(a)
  idx = 2
  Do While p(idx) = 1
    p(idx) = idx
    idx = idx + 1
  Loop
Loop
End With
End Sub

Sub VisitPerm(a As Variant)
'Print current permutation in immediate window and on sheet Output.
'You can analyze the permutation or do other things as well.
Dim i As Long
For i = 1 To UBound(a)
  Debug.Print a(i);
  wsO.Cells(r, i) = a(i)
Next i
Debug.Print
r = r + 1
End Sub
```

Sterblichkeitsrente

Sie haben viele verschiedene Möglichkeiten, eine Sterblichkeitsrente zu berechnen:

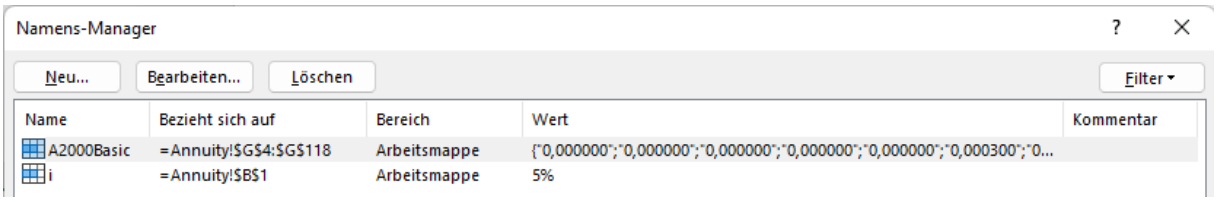
Komplexe Matrixformel (Schlechteste Wahl)

Wenden Sie eine recht komplexe Matrixformel an:



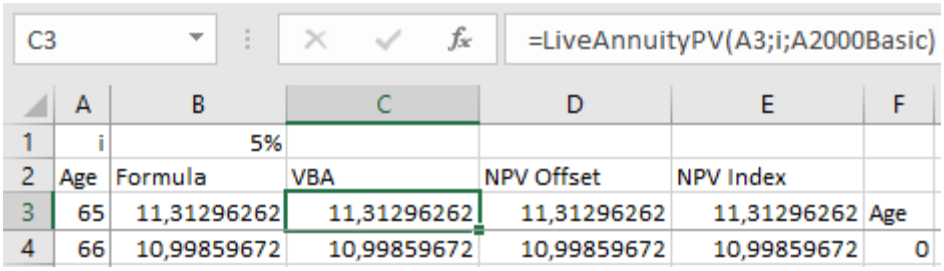
	A	B	C	D	E	F	G
1	i	5%					
2	Age	Formula	VBA	NPV Offset	NPV Index		
3	65	11,31296262	11,31296262	11,31296262	11,31296262	Age	Male q_x
29	91	3,830186458	3,830186458	3,830186458	3,830186458	25	0,000700
30	92	3,468550867	3,468550867	3,468550867	3,468550867	26	0,000700
31	93	3,046642678	3,046642678	3,046642678	3,046642678	27	0,000700
32	94	2,998718515	2,998718515	2,998718515	2,998718515	28	0,000800
33	95	2,935818052	2,935818052	2,935818052	2,935818052	29	0,000800
34	96	2,853261193	2,853261193	2,853261193	2,853261193	30	0,000800
35	97	2,744905315	2,744905315	2,744905315	2,744905315	31	0,000800

Die Namen sind wie folgt definiert:



Einfache Benutzerdefinierte Funktion mit VBA (Bessere Wahl)

Ein weitaus besser Weg besteht in einer einfachen benutzerdefinierten Funktion:

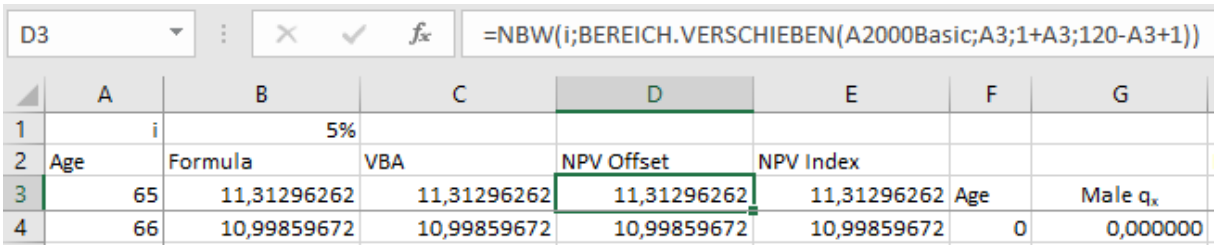


	A	B	C	D	E	F
1	i	5%				
2	Age	Formula	VBA	NPV Offset	NPV Index	
3	65	11,31296262	11,31296262	11,31296262	11,31296262	Age
4	66	10,99859672	10,99859672	10,99859672	10,99859672	0

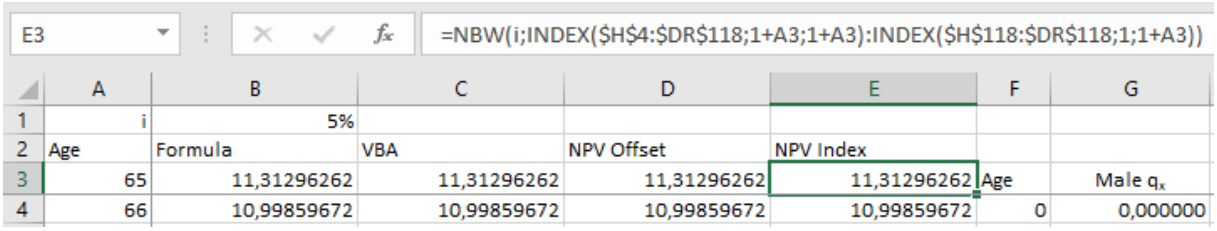
```
Function LiveAnnuityPV(lYears As Long, dInterestRate As Double, _
    rMortalityTable As Range) As Double
'(C) (P) by Bernd Plumhoff 22-Mar-2014 PB V0.1
Dim j As Long
Dim dSum As Double, dProd As Double, dPV As Double

dProd = 1#
dPV = 1#
For j = 1 To rMortalityTable.Count - lYears
    dPV = dPV / (1# + dInterestRate)
    dProd = dProd * (1# - rMortalityTable(j + lYears))
    dSum = dSum + dPV * dProd
Next j

LiveAnnuityPV = dSum

End Function
```

Vorkalkulierte Tabelle und eine NBW Formel (Wahrscheinlich am Besten)

Der schnellste und wahrscheinlich beste Ansatz besteht aus einer vorkalkulierten Tabelle und einer NBW Formel:



	A	B	C	D	E	F	G
1	i	5%					
2	Age	Formula	VBA	NPV Offset	NPV Index		
3	65	11,31296262	11,31296262	11,31296262	11,31296262	Age	Male q_x
4	66	10,99859672	10,99859672	10,99859672	10,99859672	0	0,000000



	A	B	C	D	E	F	G
1	i	5%					
2	Age	Formula	VBA	NPV Offset	NPV Index		
3	65	11,31296262	11,31296262	11,31296262	11,31296262	Age	Male q_x
4	66	10,99859672	10,99859672	10,99859672	10,99859672	0	0,000000

Bitte beachten Sie: Die BEREICH.VERSCHIEBEN Formel ist etwas schneller als die INDEX Formel. Aber sie ist volatil, d.h. sie wird bei jedem Drücken von F9 erneut berechnet und nicht erst nach Änderung ihrer Eingabewerte.

H9 =(1-$G9)*H8

	A	H	I	J	K	L	M
1	i						
2	Age	Precalculated values					
3	65	0	1	2	3	4	5
4	66	1,000000					
5	67	1	1,000000				
6	68	1	1	1,000000			
7	69	1	1	1	1,000000		
8	70	1	1	1	1	1,000000	
9	71	0,9997	0,9997	0,9997	0,9997	0,9997	0,999700
10	72	0,99940009	0,99940009	0,99940009	0,99940009	0,99940009	0,99940009
11	73	0,99910027	0,99910027	0,99910027	0,99910027	0,99910027	0,99910027
12	74	0,99880054	0,99880054	0,99880054	0,99880054	0,99880054	0,99880054
13	75	0,99840102	0,99840102	0,99840102	0,99840102	0,99840102	0,99840102
14	76	0,998001659	0,998001659	0,998001659	0,998001659	0,998001659	0,998001659
15	77	0,997602459	0,997602459	0,997602459	0,997602459	0,997602459	0,997602459
16	78	0,997203418	0,997203418	0,997203418	0,997203418	0,997203418	0,997203418
17	79	0,996804536	0,996804536	0,996804536	0,996804536	0,996804536	0,996804536
18	80	0,996306134	0,996306134	0,996306134	0,996306134	0,996306134	0,996306134

Mit FastExcel © können Sie die Rechengeschwindigjeit Ihrer Excel Berechnungen messen. Im vorliegenden Fall ist der NBW Ansatz etwa 45-mal schneller als die suboptimale Matrixformel und etwa 27-mal schneller als die VBA Lösung:

WorkSheet Areas Profile **Annuity**

Address Area	Count of Formula	Count of ArrayCells	Count of CondFmt	Calc Millisecs Area/Col	Calc Millisecs Sheet	MicroSecs /Formula	Area/Col % of Sheet
Column							
B	50	50	0	12.01	26.43	240.29	45.5%
C	50	0	0	7.21	26.43	144.27	27.3%
E	50	0	0	0.31	26.43	6.27	1.2%
D	50	0	0	0.27	26.43	5.42	1.0%

Summenerhaltendes Runden mit *RoundToSum*

Abstract

Gerundete Werte ergeben zusammen nicht immer ihre gerundete Summe. Wie stelle ich sicher, dass meine Aufstellung von gerundeten Prozentzahlen genau 100% ergibt? Kann ich für meine Buchhaltung sicherstellen, dass meine Gemeinkostenverrechnung genau die originale Kostensumme verteilt? Diese Fragen sind seit langem bekannt und wurden oft analysiert.

In diesem Dokument wird eine Lösung mit Excel / VBA vorgestellt. Sie kann relative Werte (Prozentzahlen) auf 100% runden oder absolute Werte (z. B. Kostenrechnungsergebnisse) runden, ohne deren gerundete Summe zu verändern. Dabei kann je nach Parameter im Vergleich zur üblichen kaufmännischen Rundung der absolute Fehler oder der relative Fehler minimal gehalten werden.

Summenerhaltendes Runden

Beim summenerhaltenden Runden werden die Summanden so gerundet, dass deren Summe gleich der gerundeten Summe der Summanden ist. Dabei kann es notwendig sein, einen oder mehrere Summanden zum entfernteren gerundeten Wert zu runden.

Beispiel für Prozentzahlen

Die Werte 11, 45 und 555 mit der Summe 611 zeigen als Prozentsumme auf 2 Nachkommastellen gerundet nicht 100,00, sondern 99,99. Die **fett** markierten Werte innerhalb der Tabelle zeigen die von der Funktion RoundToSum veränderten gerundeten Werte:

	Werte	Prozent auf 2 Stellen gerundet	Minimiere absoluten Fehler	Minimiere relativen Fehler
	11	1,80	1,80	1,80
	45	7,36	**7,37**	7,36
	555	90,83	90,83	**90,84**
Summe	611	99,99	100,00	100,00

Die hier vorgestellte Excel / VBA Funktion würde jedoch mit dem Aufruf *RoundToSum({11;45;555};2;FALSCH)* die Ergebniswerte *{1,80;7,37;90,83}* liefern. Der Prozentwert *7,364975* wurde hier zur "falschen" Seite hin gerundet, um die Prozentsumme 100,00 zu erhalten und dabei den absoluten Fehler gegenüber der kaufmännischen Rundung zu minimieren. Mit dem Aufruf *RoundToSum({11;45;555};2;FALSCH;2)* hätten wir die Ergebniswerte *{1,80;7,36;90,84}* erhalten, weil hier der Fehlertypparameter 2 den relativen Fehler minimal gehalten hätte.

Beispiel für absolute Zahlen

Die Summe der kaufmännisch gerundeten Werte in Spalte 2 weicht um +2.000 von der gerundeten Summe ab. Die **fett** markierten Werte innerhalb der Tabelle zeigen die von der Funktion *RoundToSum* veränderten gerundeten Werte:

	Werte	Absolut gerundet auf 1,000	Minimiere absoluten Fehler	Minimiere relativen Fehler
	4,523	5,000	5,000	5,000
	456	0	0	0
	-78,845	-79,000	-79,000	-79,000
	-14,491	-14,000	**-15,000**	-14,000
	65,789	66,000	66,000	66,000
	129,512	130,000	**129,000**	**129,000**
	15,562	16,000	16,000	16,000
	548,555	549,000	549,000	**548,000**
	1,590	2,000	2,000	2,000
	-897	-1,000	-1,000	-1,000
	6,968	7,000	7,000	7,000
	2,987	3,000	3,000	3,000
Summe	681,709	684,000	682,000	682,000

Die benutzerdefinierte VBA Funktion *RoundToSum*

Name

RoundToSum – Summanden runden ohne Veränderung der gerundeten Summe

Synopsis

RoundToSum (vInput; [lDigits]; [bAbsSum]; [lErrorType]; [bDontAmend])

Beschreibung

RoundToSum rundet die Summanden, ohne deren gerundete Summe zu verändern. Es verwendet die Largest Remainder Methode (auch Hare-Niemeyer Verfahren genannt), um den Fehler gegenüber der üblichen kaufmännischen Rundung zu minimieren. Falls dieser Fehler für mehrere Summanden identisch ist, wird der erste oder die ersten (von oben oder von links gesehen) Summanden angepasst.

Anmerkung: Die hier vorgestellte Lösung ist auf eindimensionale Tabellen ohne Teilsummen beschränkt. Für zweidimensionale Tabellen oder Tabellen mit Teilsummen existiert keine allgemeingültige Lösung.

Parameter

vInput Bereich oder Array, das die nicht gerundeten Eingabewerte enthält.

lDigits Optional, der Standardwert ist 2. Anzahl der Stellen, auf die gerundet werden soll. Zum Beispiel: 0 rundet auf ganze Zahlen, 2 rundet auf den Cent, -3 rundet auf Tausender.

bAbsSum Optional, der Standardwert ist WAHR. WAHR nimmt die Summanden (Eingabewerte) als unveränderte absolute Werte, was für buchhalterische Rechnungen oft notwendig ist. FALSCH verwendet die Prozentzahlen der Summanden, um genau auf die Summe 100% zu kommen. Dies wird meist für Präsentationen von prozentualen Verteilungen verwendet.

lErrorType Optional, der Standardwert ist 1. Fehlertyp, der minimal gehalten werden soll: 1 – absoluter Fehler, 2 – relativer Fehler. Den absoluten Fehler minimieren Sie üblicherweise bei zu buchenden Beträgen. Der relative Fehler wird normalerweise bei Verteilungen minimiert, um Anpassungen in flachen Außenbereichen zu vermeiden.

RoundToSum Programmcode

```
Enum mc_Macro_Categories
  mcFinancial = 1
  mcDate_and_Time
  mcMath_and_Trig
  mcStatistical
  mcLookup_and_Reference
  mcDatabase
  mcText
  mcLogical
  mcInformation
  mcCommands
  mcCustomizing
  mcMacro_Control
  mcDDE_External
  mcUser_Defined
  mcFirst_custom_category
  mcSecond_custom_category 'and so on
End Enum 'mc_Macro_Categories

Function RoundToSum (vInput As Variant, Optional lDigits As Long = 2, Optional bAbsSum As Boolean = True, _
    Optional lErrorType As Long = 1) As Variant
'Calculate rounded summands which exactly add up to the rounded sum of unrounded summands.
'It uses the largest remainder method which minimizes the error to the original unrounded summands.
'V2.3 PB 27-Oct-2024 (C) (P) by Bernd Plumhoff
Dim b As Boolean, i As Long, j As Long, k As Long, n As Long, lCount As Long, lSgn As Long
Dim d As Double, dDiff As Double, dRoundedSum As Double, dSumAbs As Double: Dim vA As Variant
With Application.WorksheetFunction
vA = .Transpose(.Transpose(vInput)): On Error GoTo Errhdl: i = vA(1) 'Force error in case of vertical arrays
On Error GoTo 0: n = UBound(vA): ReDim vC(1 To n) As Variant, vD(1 To n) As Variant: dSumAbs = .Sum(vA)
For i = 1 To n
  d = IIf(bAbsSum, vA(i), vA(i) / dSumAbs * 100#): vC(i) = .Round(d, lDigits)
  If lErrorType = 1 Then 'Absolute error
    vD(i) = vC(i) - d
  ElseIf lErrorType = 2 Then 'Relative error
    vD(i) = (vC(i) - d) * d
  Else
    RoundToSum = CVErr(xlErrValue): Exit Function
  End If
Next i
dRoundedSum = .Round(IIf(bAbsSum, dSumAbs, 100#), lDigits)
dDiff = .Round(dRoundedSum - .Sum(vC), lDigits)
If dDiff <> 0# Then
  lSgn = Sgn(dDiff): lCount = .Round(Abs(dDiff) * 10 ^ lDigits, 0)
  'Now find highest (lowest) lCount indices in vD
  ReDim m(1 To lCount) As Long
  For i = 1 To lCount: m(i) = i: Next i
  For i = 1 To lCount - 1
    For j = i + 1 To lCount
      If lSgn * vD(m(i)) > lSgn * vD(m(j)) Then k = m(i): m(i) = m(j): m(j) = k
    Next j
  Next i
  For i = lCount + 1 To n
    If lSgn * vD(i) < lSgn * vD(m(lCount)) Then
      j = lCount - 1
      Do While j > 0
        If lSgn * vD(i) >= lSgn * vD(m(j)) Then Exit Do
        j = j - 1
      Loop
      For k = lCount To j + 2 Step -1: m(k) = m(k - 1): Next k: m(j + 1) = i
    End If
  Next i
  For i = 1 To lCount: vC(m(i)) = .Round(vC(m(i)) + dDiff / lCount, lDigits): Next i
End If
```

```
If b Then vC = .Transpose(vC)
RoundToSum = vC
Exit Function
Errhdl:
'Transpose variants to be able to address them with vA(i), not vA(i,1)
b = True: vA = .Transpose(vA): Resume Next
End With
End Function

Sub DescribeFunction_RoundToSum ()
'Run this only once, then you will see this description in the function menu
Dim FuncName As String, FuncDesc As String, Category As String, ArgDesc(1 To 4) As String
FuncName = "RoundToSum"
FuncDesc = "Rounding values preserving their rounded sum"
Category = mcMath_and_Trig
ArgDesc(1) = "Range or array which contains unrounded values"
ArgDesc(2) = "[Optional = 2] Number of digits to round to. For example: 0 rounds to integers, 2 rounds to the cent, -3 will
use thousands"
ArgDesc(3) = "[Optional = True] True takes the summands as they are; False works on the summands' percentages to make all
percentages add up to 100% exactly"
ArgDesc(4) = "[Optional = 1] Error type: 1= absolute error, 2 = relative error"
Application.MacroOptions _
  Macro:=FuncName, _
  Description:=FuncDesc, _
  Category:=Category, _
  ArgumentDescriptions:=ArgDesc
End Sub
```

Round2Sum Lambda-Ausdruck

Mithilfe dreier Lambda-Ausdrücke können wir die VBA Funktion *RandToSum* durch diesen *Round2Sum* Lambda-Ausdruck ersetzen:

```
=LAMBDA(vI;lD;bA;lE;
    LET(
        i;WENN(bA;vI;vI/SUMME(vI)%);
        r;RUNDEN(i;lD);
        _C;RUNDEN(SUMME(i);lD)-SUMME(r);
        _E;WAHL(lE;r-i;(r-i)*i);
        _R;UniqRank(_E;WENN(_C>0;1;0));
        _D;WENN(_R<=RUNDEN(ABS(_C*10^lD);0);VORZEICHEN(_C)*10^-lD;0);
        r+WENN(ZEILEN(r)=1;MTRANS(_D);_D)
    )
)
```

UniqRank wird definiert als:

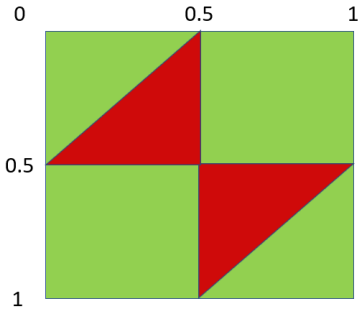
```
=LAMBDA(Ref;[Order];
    LET(
        _ord;WENN(WURDEAUSGELASSEN(Order);-1;WENN(Order=0;-1;1));
        _r;INDEX(WENN(ZEILEN(Ref)=1;MTRANS(Ref);Ref);;1);
        _c;ZEILEN(_r);
        _i;SEQUENZ(ZEILEN(_r));
        INDEX(SORTIEREN(HSTAPELN2(_i;INDEX(SORTIEREN(HSTAPELN2(_r;_i);;_ord);;2));2;1);;1)
    )
 )
```

Und – da Excel's Tabellenblattfunktion HSTAPELN nur Bereiche, aber keine Arrays annimt – *HSTAPELN2* als:

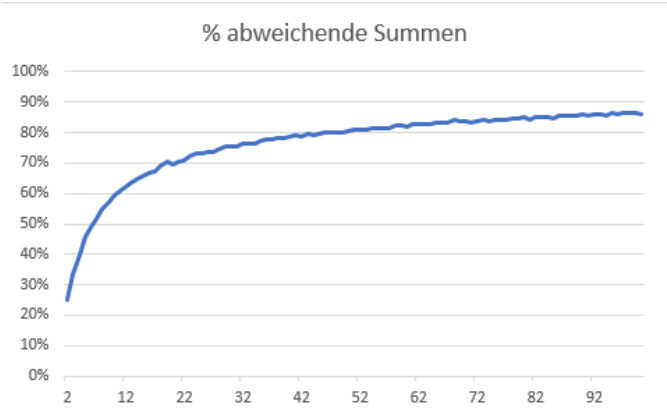
```
=LAMBDA(a;b;
    MATRIXERSTELLEN(
        ZEILEN(a);
        2;
        LAMBDA(r;c;
            WENN(c=1;INDEX(a;r);INDEX(b;r))
        )
    )
)
```

Werte runden ändert ihre Summe

Wie wahrscheinlich ist es, dass die Summe von gerundeten Werten nicht gleich ihrer gerundeten Summe ist? Für zwei zufällige Gleitkommazahlen ist dies klar: die Wahrscheinlichkeit liegt bei etwa 25% - das ist der **Rot**-Anteil an der Gesamtfläche der gezeigten Grafik:



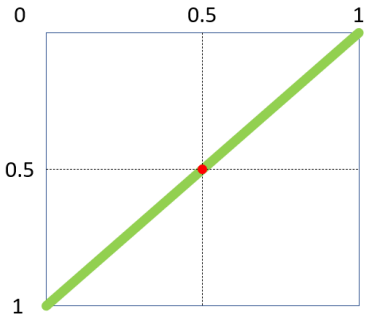
Aber etwas überraschend mag sein, dass die Wahrscheinlichkeit sich 90% nähert, je mehr Zahlen gerundet und addiert werden:



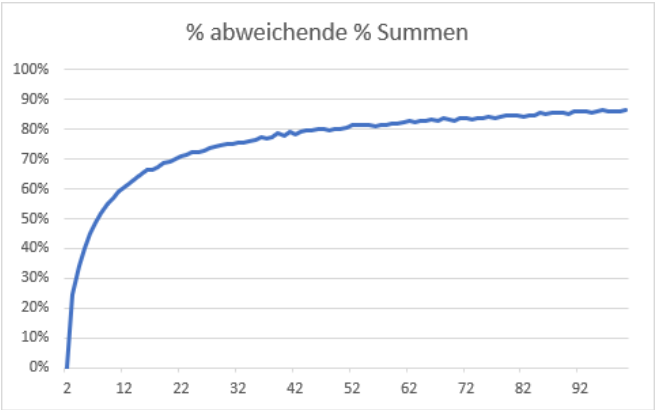
Bei sieben zufälligen Gleitkommazahlen ist die Wahrscheinlichkeit bereits größer als 50%, dass die Summe ihrer gerundeten Werten nicht gleich ihrer gerundeten Summe ist.

Gerundete Prozentanteile

Gerundete Prozentanteile ergeben addiert auch häufig nicht 100%. Bei zwei zufälligen Zahlen tritt das Problem nur auf, wenn beide Zahlen genau gleich 0,5 sind:



Aber bei mehr Zahlen stellt sich dasselbe Problem wie eingangs erwähnt, nur mit etwa einer Zahl mehr. Gerundete Prozentanteile von drei zufälligen Zahlen ergeben in etwa 25% der Fälle keine Summe von 1:



Programmcode Monte Carlo

```
Const n = 100
Const runs = 20000
Const bOnlyPositive = True 'Without loss of generality

Sub monte_carlo_add_rounded_values()
'Calculates for 2 to n how likely it is
'that rounding would not alter their sum.
'Example: for 2 numbers there is a 25% chance
'that the sum of their rounded values is not
'equal to their rounded sum.
'(C) (P) by Bernd Plumhoff 16-Dec-2023 PB V0.3
Dim i                As Long
Dim j                As Long
Dim k                As Long
Dim m                As Long
Dim d                As Double
Dim s1               As Double
Dim s2               As Double

With Application.WorksheetFunction
Randomize
For i = 2 To n
  m = 0
  For j = 1 To runs
    s1 = 0#
    s2 = 0#
    For k = 1 To i
      If bOnlyPositive Then
        d = Rnd()
      Else
        d = 2# * Rnd() - 1#
      End If
      s1 = s1 + d
      s2 = s2 + .Round(d, 0)
    Next k
    s1 = .Round(s1, 0)
    If s1 <> s2 Then
      m = m + 1
    End If
  Next j
  Cells(i, 1) = i
  Cells(i, 2) = m / runs
Next i
End With
End Sub

Sub monte_carlo_percentage_sum_of_rounded_values()
'Calculates for 2 to n how likely it is that
'rounding would not alter their percentage sum.
'Example: for 2 numbers there is a 25% chance
'that the sum of their rounded values is not
'equal to their rounded sum.
'(C) (P) by Bernd Plumhoff 16-Dec-2023 PB V0.2
Dim i                As Long
Dim j                As Long
Dim k                As Long
Dim m                As Long
Dim s1               As Double
Dim s2               As Double

With Application.WorksheetFunction
Randomize
For i = 2 To n
  m = 0
  ReDim e(1 To i) As Double
  For j = 1 To runs
    s1 = 0#
    For k = 1 To i
      If bOnlyPositive Then
        e(k) = Rnd()
      Else
        e(k) = 2# * Rnd() - 1#
      End If
      s1 = s1 + e(k)
    Next k
    s2 = 0#
    For k = 1 To i
      e(k) = .Round(1000# * e(k) / s1, 0)
      s2 = s2 + e(k)
    Next k
    If s2 <> 1000# Then
      m = m + 1
    End If
  Next j
  Cells(i, 1) = i
  Cells(i, 2) = m / runs
Next i
End With
End Sub
```

Anwendungsbeispiele für *RoundToSum*

Gemeinkostenumlage

Wenn Sie Gemeinkosten auf Kostenträger verrechnen müssen, ergibt sich häufig das Problem, dass die Summe der umgelegten Kosten nicht cent-genau mit der Ursprungssumme übereinstimmt. Hier hilft die benutzerdefinierte Excel Funktion *RoundToSum*.

Ein praktisches Beispiel

Vorgestellt wird eine Gemeinkostenumlage, bei der sich die einzelnen Summanden cent-genau zu den jeweiligen Gesamtsummen addieren.

Zunächst legen Sie die Gemeinkosten auf alle anderen Kostenstellen um (Tabellenblatt ‚Schlüssel'):

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**Umlage 1 - Kostenstellen-Gemeinkosten auf alle anderen Kostenstellen**															
2			allg. Kostenstellen								Hilfskostenstellen			Hauptkostenstellen		
3	Schlüsselbezeichnung	Gesamt	GF	Sekretariat	RW	Controlling	Personal	ÖA/Werbung	Auszub.	Betriebsrat	Werkstatt 1	Werkstatt 2	Fuhrpark	Fertigung1	Fertigung2	Fertigung3
4	pro Kopf	102	1	1	3	1	2	3	3	1	12	10		20	20	25
5	m²	2685	50	40	100	30	50	50		15	250	350	100	500	550	600
6	gleich	14	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	Gewichtung	16	2	1	1	1	1	2	1	1	1	1	1	1	1	1

Read_me | **Schlüssel** | 1_Umlage | 2_Umlage

Diese erste Gemeinkostenumlage erfolgt mit dem Aufruf einer Rundungskorrektur, damit die einzelnen Summanden cent-genau die Spaltensummen pro Kostenstelle ergeben (Tabellenblatt ‚1_Umlage':

D13 | =RoundToSum($C13/Schlüssel!$B$8*Schlüssel!C$8:P$8)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	**Umlage der Kostenstellengemeinkosten auf alle Kostenstellen**																	
2																		
3				Verwaltungskostenstellen								Hilfskostenstellen			Hauptkostenstellen			Gesamt
4	Bezeichnung	Schlüssel	KSt.- Gemeinkosten	GF	Sekretariat	RW	Controlling	Personal	ÖA/Werbung	Auszub.	Betriebsrat	Werkstatt 1	Werkstatt 2	Fuhrpark	Fertigung1	Fertigung2	Fertigung3	
5	Reise- u. Übernachtung	Gewichtung	10.000,00	1.250,00	625,00	625,00	625,00	625,00	1.250,00	625,00	625,00	625,00	625,00	625,00	625,00	625,00	625,00	10.000,00
6	Aufw. Aufsichtsrat	gleich	3.000,00	375,00	187,50	187,50	187,50	187,50	375,00	187,50	187,50	187,50	187,50	187,50	187,50	187,50	187,50	3.000,00
7	Bewirtung	Gewichtung	2.000,00	250,00	125,00	125,00	125,00	125,00	250,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	2.000,00
8	Geschenke	Gewichtung	1.000,00	125,00	62,50	62,50	62,50	62,50	125,00	62,50	62,50	62,50	62,50	62,50	62,50	62,50	62,50	1.000,00
9	Gebühren, Beiträge	gleich	500,00	62,50	31,25	31,25	31,25	31,25	62,50	31,25	31,25	31,25	31,25	31,25	31,25	31,25	31,25	500,00
10	Kfz-Kosten	Gewichtung	4.500,00	562,50	281,25	281,25	281,25	281,25	562,50	281,25	281,25	281,25	281,25	281,25	281,25	281,25	281,25	4.500,00
11	Geräte Ausstattungsgegens	Gewichtung	200,00	25,00	12,50	12,50	12,50	12,50	25,00	12,50	12,50	12,50	12,50	12,50	12,50	12,50	12,50	200,00
12	Prüfungen Jahresabschl.	gleich	10.000,00	1.250,00	625,00	625,00	625,00	625,00	1.250,00	625,00	625,00	625,00	625,00	625,00	625,00	625,00	625,00	10.000,00
13	sonst. Bewirtschaftungsk.	m²	5.500,00	687,50	343,75	343,75	343,75	343,75	687,50	343,75	343,75	343,75	343,75	343,75	343,75	343,75	343,75	5.500,00
14	Energiekosten	m²	6.000,00	750,00	375,00	375,00	375,00	375,00	750,00	375,00	375,00	375,00	375,00	375,00	375,00	375,00	375,00	6.000,00
15	Versicherungen	pro Kopf	5.000,00	625,00	312,50	312,50	312,50	312,50	625,00	312,50	312,50	312,50	312,50	312,50	312,50	312,50	312,50	5.000,00
16	Rechts-, Beratungskosten	Gewichtung	5.000,00	625,00	312,50	312,50	312,50	312,50	625,00	312,50	312,50	312,50	312,50	312,50	312,50	312,50	312,50	5.000,00
17	Buchführungskosten	Gewichtung	1.000,00	125,00	62,50	62,50	62,50	62,50	125,00	62,50	62,50	62,50	62,50	62,50	62,50	62,50	62,50	1.000,00
18	Bürobedarf	Gewichtung	2.000,00	250,00	125,00	125,00	125,00	125,00	250,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	2.000,00
19	Telekommunikation	Gewichtung	2.500,00	312,50	156,25	156,25	156,25	156,25	312,50	156,25	156,25	156,25	156,25	156,25	156,25	156,25	156,25	2.500,00
20	Aufw. f. Versand	Gewichtung	2.000,00	250,00	125,00	125,00	125,00	125,00	250,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	2.000,00
21	Bücher, Zeitschriften	Gewichtung	1.000,00	125,00	62,50	62,50	62,50	62,50	125,00	62,50	62,50	62,50	62,50	62,50	62,50	62,50	62,50	1.000,00
22	Nebenk. d. Geldverkehrs	Gewichtung	500,00	62,50	31,25	31,25	31,25	31,25	62,50	31,25	31,25	31,25	31,25	31,25	31,25	31,25	31,25	500,00
23	Schadensersatz	gleich	250,00	31,24	15,62	15,62	15,62	15,62	31,25	15,62	15,63	15,63	15,63	15,63	15,63	15,63	15,63	250,00
24	Berufsbekleidung	Gewichtung	1.500,00	187,50	93,75	93,75	93,75	93,75	187,50	93,75	93,75	93,75	93,75	93,75	93,75	93,75	93,75	1.500,00
25	Ausgleichsabgabe Schwerbe	gleich	2.000,00	250,00	125,00	125,00	125,00	125,00	250,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	2.000,00
26	Fort- u. Weiterbildung	Gewichtung	2.000,00	250,00	125,00	125,00	125,00	125,00	250,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	125,00	2.000,00
27	Sonstige Betriebl. Aufw.	Gewichtung	1.500,00	187,50	93,75	93,75	93,75	93,75	187,50	93,75	93,75	93,75	93,75	93,75	93,75	93,75	93,75	1.500,00
28	Gesamt (gerundete Summe)		68.950,00	8.618,74	4.309,37	4.309,37	4.309,37	4.309,37	8.618,75	4.309,37	4.309,38	4.309,38	4.309,38	4.309,38	4.309,38	4.309,38	4.309,38	68.950,00

Tabellenblattformeln		
Bereich	Formel	
D5:D27	**D5**	=RoundToSum($C5/Schlüssel!$B$6*Schlüssel!C$6:P$6)
C28:Q28	**C28**	=SUMME(C5:C27)
R5:R28	**R5**	=SUMME(D5:Q5)
R30	**R30**	=R28-C28

Die zweite Gemeinkostenumlage (Tabellenblatt ‚Schlüssel') verteilt die allgemeinen Kostenstellen und die Hilfskostenstellen mit dem Aufruf einer Rundungskorrektur centgenau auf die Hauptkostenstellen:

	A	B	C	D	E	F
10	**Umlage 2 - allgemeine Kostenstellen und Hilfskostenstellen auf Hauptkostenstellen**					
11						
12						
13	umzulegende KSt.	Schlüssel	Fertigung1	Fertigung2	Fertigung3	Gesamt
14						
15	GF	Gewichtung	30%	40%	30%	100%
16	Sekretariat	Gewichtung	40%	50%	10%	100%
17	RW	Gewichtung	30%	13%	57%	100%
18	Controlling	gleich	1	1	1	3
19	Personal	pro Kopf	20	20	25	65
20	ÖA/Werbung	Gewichtung	30%	42%	28%	100%
21	Auszub.	gleich	1	1	1	3
22	Betriebsrat	pro Kopf	20	20	25	65
23	Werkstatt 1	Gewichtung	25%	20%	55%	100%
24	Werkstatt 2	Gewichtung	20%	20%	60%	100%
25	Fuhrpark	Gewichtung	40%	30%	30%	100%

Read_me | **Schlüssel** | 1_Umlage | 2_Umlage

Das Endergebnis (Tabellenblatt ‚2_Umlage'):

	A	B	C	D	E	F	G	H
1	**Umlage der allgemeinen Kostenstellen und Hilfskostenstellen auf die Hauptkostenstellen**							
2								
3	**umzulegende KSt.**	**KST- Einzel-kosten**	**Umlage 1**	**Gesamt**	**Fertigung1**	**Fertigung2**	**Fertigung3**	**Gesamt**
4								
5	GF	111.666,00	4.924,95	116.590,95	34.977,28	46.636,38	34.977,29	116.590,95
6	Sekretariat	34.627,00	4.924,95	39.551,95	15.820,78	19.775,97	3.955,20	39.551,95
7	RW	96.834,00	4.924,97	101.758,97	30.527,69	13.228,67	58.002,61	101.758,97
8	Controlling	83.875,00	4.924,97	88.799,97	29.599,99	29.599,99	29.599,99	88.799,97
9	Personal	53.765,00	4.925,00	58.690,00	18.058,46	18.058,46	22.573,08	58.690,00
10	ÖA/Werbung	239.170,00	4.925,00	244.095,00	73.228,50	102.519,90	68.346,60	244.095,00
11	Auszub.	147.397,00	4.925,02	152.322,02	50.774,00	50.774,01	50.774,01	152.322,02
12	Betriebsrat	471,00	4.925,02	5.396,02	1.660,32	1.660,31	2.075,39	5.396,02
13	Werkstatt 1	125.225,00	4.925,02	130.150,02	32.537,51	26.030,00	71.582,51	130.150,02
14	Werkstatt 2	2.398.512,00	4.925,02	2.403.437,02	480.687,41	480.687,40	1.442.062,21	2.403.437,02
15	Fuhrpark	26.992,00	4.925,02	31.917,02	12.766,81	9.575,10	9.575,11	31.917,02
16	1. Umlage				4.925,02	4.925,02	4.925,02	14.775,06
17	2. Umlage	3.318.534,00	54.174,94	3.372.708,94	780.638,75	798.546,19	1.793.524,00	3.372.708,94
18	KSt. - Einzelkosten				738.060,00	854.000,00	650.360,00	2.242.420,00
19	Gesamt Haupkostenstellen				1.523.623,77	1.657.471,21	2.448.809,02	5.629.904,00
20								
21	Gemeinkostenzuschlagssatz				106,4%	94,1%	276,5%	151,1%
22								
23							Kontrolle	0,00

Tabellenblattformeln		
Bereich	Formel	
C5:C15	**C5**	=MTRANS('1_Umlage'!D28:N28)
D5:D15	**D5**	=SUMME(B5:C5)
E5:E15	**E5**	=RoundToSum($D5/Schlüssel!$F15*Schlüssel!C15:E15)
H5:H16	**H5**	=SUMME(E5:G5)
E16	**E16**	='1_Umlage'!O28:Q28
B17:H17	**B17**	=SUMME(B5:B15)
E19:H19	**E19**	=SUMME(E16:E18)
E21:H21	**E21**	=(E17+E16)/E18
H23	**H23**	=H19-SUMME(E18:G18)-SUMME(B5:B15)-SUMME('1_Umlage'!C5:C27)

Diese korrekte Gemeinkostenumlage können Sie in einem Buchungssystem ohne Cent-Differenzen buchen.

Beispiel für ein exaktes Verhältnis von Zufallszahlen

Es ist recht leicht, einen “unfairen” Würfel zu simulieren. Wenn wir z. B. im Mittel die 6 doppelt so häufig wie alle anderen Zahlen 1 bis 5 erhalten wollen, geben Sie in Zelle A1 ein:
*=MIN(GANZZAHL(ZUFALLSZAHL()*7+1);6)*

Aber wenn Sie einen Würfel 7 mal würfeln wollen und alle Zahlen von 1 bis 5 genau einmal und die 6 genau zweimal erscheinen soll?

Eine allgemeine Lösung:

	A	B	C	D	E	F	G	H	I	J	K	L
1				Nur statistische Wahrscheinlichkeit						Total		
2	**Farbe**	**Häufigkeit**	**Pos / Iteration**	**Eins**	**Zwei**	**Drei**	**Vier**	**Fünf**	**Sechs**	**Grün**	**Gelb**	**Rot**
3	**Grün**	50,00%	**1**	Grün	Grün	Grün	Grün	Grün	Rot	5	0	1
4	**Gelb**	33,33%	**2**	Gelb	Gelb	Grün	Grün	Gelb	Gelb	2	4	0
5	**Rot**	16,67%	**3**	Rot	Rot	Rot	Grün	Grün	Gelb	2	1	3
6			**4**	Gelb	Gelb	Grün	Grün	Gelb	Grün	3	3	0
7			**5**	Grün	Grün	Gelb	Grün	Gelb	Grün	4	2	0
8			**6**	Rot	Rot	Grün	Grün	Gelb	Grün	3	1	2
9			**7**	Gelb	Grün	Grün	Gelb	Grün	Rot	3	2	1
10			**8**	Grün	Gelb	Grün	Gelb	Grün	Gelb	3	3	0
11			**9**	Rot	Gelb	Gelb	Gelb	Gelb	Gelb	0	5	1
12			**10**	Grün	Grün	Grün	Gelb	Gelb	Rot	3	2	1
13									**Total:**	28	23	9
14									**Sollte stochastisch ergeben:**	30	20	10
16				Genaue Häufigkeit						Total		
17			**Pos / Iteration**	**Eins**	**Zwei**	**Drei**	**Vier**	**Fünf**	**Sechs**	**Grün**	**Gelb**	**Rot**
18			**1**	Grün	Grün	Gelb	Grün	Rot	Gelb	3	2	1
19			**2**	Gelb	Grün	Gelb	Grün	Grün	Rot	3	2	1
20			**3**	Grün	Grün	Rot	Gelb	Grün	Gelb	3	2	1
21			**4**	Rot	Grün	Grün	Grün	Gelb	Gelb	3	2	1
22			**5**	Gelb	Grün	Rot	Grün	Grün	Gelb	3	2	1
23			**6**	Rot	Grün	Grün	Grün	Gelb	Gelb	3	2	1
24			**7**	Grün	Grün	Gelb	Gelb	Grün	Rot	3	2	1
25			**8**	Grün	Grün	Rot	Gelb	Gelb	Grün	3	2	1
26			**9**	Gelb	Rot	Gelb	Grün	Grün	Grün	3	2	1
27			**10**	Grün	Rot	Grün	Gelb	Gelb	Grün	3	2	1
28									**Total:**	30	20	10
29									**Sollte stochastisch ergeben:**	30	20	10

Tabellenblattformeln		
Bereich	Formel	
D3:I12	**D3**	=INDEX($A$3:$A$5;GANZZAHL(sbRandHistogrm(1;4;$B$3:$B$5)))
J3:L12;J18:L27	**J3**	=ZÄHLENWENN($D3:$I3;J$2)
J13:L13;J28:L28	**J13**	=SUMME(J3:J12)
J14;J29	**J14**	=ANZAHL2($D$3:$I$12)*MTRANS($B$3:$B$5)
D18:D27	**D18**	=INDEX($A$3:$A$5;GANZZAHL(sbExactRandHistogrm(6;1;4;$B$3:$B$5)))

Die benutzerdefinierte VBA Funktion sbExactRandHistogrm

Name

sbExactRandHistogrm – Erzeuge eine exakte Histogramm-Verteilung des Datentyps Double.

Synopsis

sbExactRandHistogrm(ldraw; dmin; dmax; vWeight)

Beschreibung

sbExactRandHistogrm erzeugt eine exakte Histogramm-Verteilung für ldraw Ziehungen von Gleitkommazahlen mit doppelter Genauigkeit im Intervall dmin:dmax. Dieses Intervall ist in vWeight.count Klassen unterteilt. Jede Klasse besitzt das Gewicht vWeight(i), welches die Wahrscheinlichkeit des Erscheinens eines Wertes innerhalb dieser Klasse darstellt. Wenn die Gewichte nicht genau für ldraw Ziehungen erreicht werden können, dann wird das Hare-Niemeyer-Verfahren (“Quotenverfahren mit Restausgleich nach größten Bruchteilen”) angewandt, um den absoluten Fehler zu minimieren. Diese Funktion ruft *RoundToSum* auf.

Parameter

ldraw Anzahl der Ziehungen

dmin Minimum = Untergrenze für die zu ziehenden Zahlen

dmax Maximum = Obergrenze für die zu ziehenden Zahlen

vWeight Array von Gewichten. Die Array Größe bestimmt die Anzahl der Klassen, in die das Intervall dmin : dmax aufgeteilt wird. Die Array-Werte bestimmen die Wahrscheinlichkeit, mit der Werte innerhalb dieser Klasse gezogen werden.

sbRandHistogrm Programmcode

```
Function sbExactRandHistogrm(ldraw As Long, _
            dmin As Double, _
            dmax As Double, _
            vWeight As Variant) As Variant
'Creates an exact histogram distribution for ldraw draws within range dmin:dmax.
'This range is divided into vWeight.count classes. Each class has weight vWeight(i)
'reflecting the probability of occurrence of a value within the class.
'If weights can't be achieved exactly for ldraw draws the largest remainder method will
'be applied to minimize the absolute error. This function calls (needs) RoundToSum.
'(C) (P) by Bernd Plumhoff 01-May-2021 PB V0.9

Dim i As Long, j As Long, n As Long
Dim vW As Variant
Dim dSumWeight As Double, dR As Double

Randomize
With Application.WorksheetFunction
vW = .Transpose(vWeight)
On Error GoTo Errhdl
i = vW(1) 'Throw error in case of horizontal array
On Error GoTo 0

n = UBound(vW)
ReDim dWeight(1 To n) As Double
ReDim dSumWeightI(0 To n) As Double
ReDim vR(1 To ldraw) As Variant

For i = 1 To n
    If vW(i) < 0# Then 'A negative weight is an error
        sbExactRandHistogrm = CVErr(xlErrValue)
        Exit Function
    End If
    'Calculate sum of all weights
    dSumWeight = dSumWeight + vW(i)
Next i

If dSumWeight = 0# Then
    'Sum of weights has to be greater zero
    sbExactRandHistogrm = CVErr(xlErrValue)
    Exit Function
End If

For i = 1 To n
    'Align weights to number of draws
    dWeight(i) = CDbl(ldraw) * vW(i) / dSumWeight
Next i

vW = RoundToSum(dWeight, 0)
On Error GoTo Errhdl
i = vW(1) 'Throw error in case of horizontal array
On Error GoTo 0

For j = 1 To ldraw

    dSumWeight = 0#
    dSumWeightI(0) = 0#
    For i = 1 To n
        'Calculate sum of all weights
        dSumWeight = dSumWeight + vW(i)
        'Calculate sum of weights till i
        dSumWeightI(i) = dSumWeight
    Next i

    dR = dSumWeight * Rnd

    i = n
    Do While dR < dSumWeightI(i)
        i = i - 1
    Loop

    vR(j) = dmin + (dmax - dmin) * (CDbl(i) + _
            (dR - dSumWeightI(i)) / vW(i + 1)) / CDbl(n)
    vW(i + 1) = vW(i + 1) - 1#

Next j

sbExactRandHistogrm = vR

Exit Function

Errhdl:
'Transpose variants to be able to address
'them with vW(i), not vW(i,1)
vW = .Transpose(vW)
Resume Next
End With

End Function
```

Faire Mitarbeiterauswahl nach Teamgröße – sbFairStaffSelection

Nehmen Sie an, in Ihrem Unternehmen müssen Sonderaufgaben durchgeführt werden, die von jedem Mitarbeiter getätigt werden können. Sie wollen alle Teams fairerweise auf Basis ihrer Mitarbeiteranzahl belasten. Für diese Auswahl bietet sich die benutzerdefinierte Excel Funktion *RoundToSum* an. Da nicht davon ausgegangen werden kann, dass für jede Sonderaufgabe jedes Team prozentual zu seiner Teamgröße belastet werden kann, sollte der Aufruf von *RoundToSum* eine Rückschau über vergangene Mitarbeiterabstellungen enthalten.

RoundToSum verwendet das Hare-Niemeyer Verfahren, welches das Mandatszuwachsparadoxon aufweist. Wenn ein Mitarbeiter mehr ausgewählt wird, kann es vorkommen, dass ein Team weniger Mitarbeiter als zuvor abstellen muss. Da dies nicht rückwirkend geschehen kann, muss dieser Effekt ausgeglichen werden, sobald er auftritt.

Beispiel

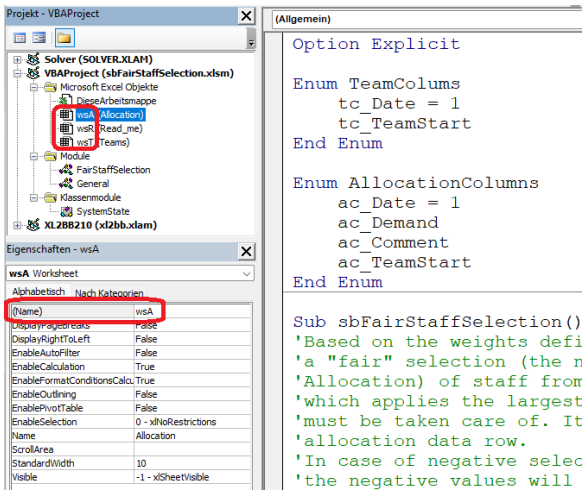
Am 1. Januar 2023 existieren die folgenden Teams (Tabellenblatt ‚Teams', VBA Name ‚wsT'):

	A	B	C	D	E
1	**Date**	**Team A**	**Team B**	**Team C**	**Team D**
2	01.01.2023	5670	3850	420	60

Über drei Monate hinweg werden die folgenden Mitarbeiter für Sonderaufgaben benötigt und ausgewählt (Tabellenblatt ‚Allocation', VBA Name ‚wsA'):

	A	B	C	D	E	F	G
		Calculate Allocation					
1	**Date**	**Demand**	**Comment**	**Team A**	**Team B**	**Team C**	**Team D**
2	01.01.2023	323		183	124	14	2
3	01.02.2023	1	Recalc 11.03.2023 10:52:24. Allocation for 1 amended to 0. Allocation for 3 set to 0.	0	1	0	0
4	01.03.2023	9676	Recalc 11.03.2023 10:52:24.	5487	3725	406	58

Am 1. Februar 2023 hätte das Hare-Niemeyer Verfahren insgesamt 184, 125, 13 und 2 Mitarbeiter für die Teams A, B, C und D ausgewählt. Da aber am 1. Januar Team C bereits 14 Mitarbeiter bereitgestellt hatte und keine rückwirkenden Anpassungen möglich sind, muss Team A oder B einen Mitarbeiter weniger geben. Der implementierte Algorithmus schaut von links nach rechts, ob angepasst werden kann, also trifft es hier Team A. Am 1. März 2023 werden genau alle restlichen Mitarbeiter aller Teams benötigt. Der Algorithmus wählt insgesamt für jedes Team genau die Gesamtzahl der Mitarbeiter aus, weil die Rückschau alle Anforderungs-Datensätze umfasst.



Anmerkung: Der VBA Name eines Tabellenblattes kann direkt mit VBA angesprochen werden. Er kann von dem Tabellenblattnamen abweichen, den der Anwender sieht. Leider kann man ihn nur manuell ändern, nicht via VBA.

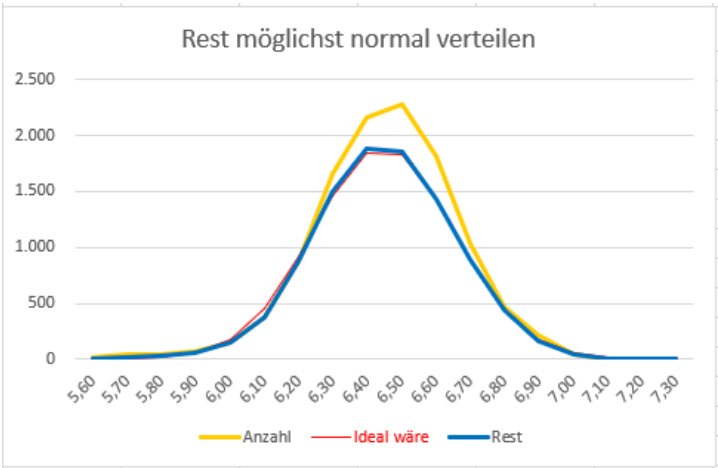
sbFairStaffSelection Programmcode

```
Enum TeamColums
    tc_Date = 1
    tc_TeamStart
End Enum

Enum AllocationColumns
    ac_Date = 1
    ac_Demand
    ac_Comment
    ac_TeamStart
End Enum

Sub sbFairStaffSelection()
'Based on the weights defined in tab Teams this program allocates
'a "fair" selection (the number given in column Demand of tab
'Allocation) of staff from these teams. This program uses (calls) RoundToSum
'which applies the largest remainder method, so the Alabama paradoxon
'must be taken care of. It also applies a lookback up to the topmost
'allocation data row.
'In case of negative selection counts (i. e. the Alabama paradoxon)
'the negative values will be set to zero and the necessary amendments
'(reductions) will be applied from left to right. Please order your
'teams with ascending sizes or descending sizes to account for this.
'(C) (P) by Bernd Plumhoff 09-Mar-2023 PB V0.1

Dim bLookBack                As Boolean
Dim bReCalc                  As Boolean

Dim i                        As Long
Dim j                        As Long
Dim k                        As Long
Dim m                        As Long
Dim lAmend                   As Long
Dim lCellResult              As Long
Dim lDemand                  As Long
Dim lRowSum                  As Long
Dim lSum                     As Long
Dim lTotal                   As Long 'Most recent total number of staff in all teams

Dim sComment                 As String

Dim vAlloc                   As Variant
Dim vTeams                   As Variant

Dim state                    As SystemState

Set state = New SystemState

With Application.WorksheetFunction

vTeams = .Transpose(.Transpose(Range(wsT.Cells(1, 1).End(xlDown).Offset(0, tc_TeamStart - 1), _
          wsT.Cells(1, 1).End(xlDown).End(xlToRight))))
j = UBound(vTeams)
ReDim dAlloc(1 To j) As Double
lTotal = .Sum(vTeams)

bReCalc = False
i = 2
lDemand = wsA.Cells(i, ac_Demand)
Do While lDemand > 0

    lRowSum = .Sum(Range(wsA.Cells(i, ac_TeamStart), wsA.Cells(i, ac_TeamStart + j)))

    If lDemand <> lRowSum Then bReCalc = True

    If bReCalc Or wsA.Cells(i + 1, ac_Demand) = 0 Then

        sComment = "Recalc " & Format(Now(), "DD.MM.YYYY HH:nn:ss") & ". "
        bLookBack = False
        k = i - 1
        If k > 1 Then
            bLookBack = True
            lDemand = 0
            lSum = 0
            ReDim lTeamSum(1 To j) As Long
            Do While k > 1
                lSum = lSum + wsA.Cells(k, ac_Demand)
                lDemand = wsA.Cells(i, ac_Demand) + lSum
                For m = 1 To j
                    lTeamSum(m) = lTeamSum(m) + wsA.Cells(k, m + ac_TeamStart - 1)
                Next m
                'If lSum >= lTotal Then Exit Do 'Uncomment if lookback should be restricted
                                               'to total staff number
                k = k - 1
            Loop
        End If

        For m = 1 To j
            dAlloc(m) = lDemand * vTeams(m) / lTotal
        Next m

        vAlloc = RoundToSum(vInput:=dAlloc, lDigits:=0)

        If bLookBack Then
            For m = 1 To j
                lCellResult = vAlloc(m) - lTeamSum(m)
                If lCellResult < 0 Then
                    'The Alabama Paradoxon: we have to reduce other parties'
                    'allocations because we cannot have negative allocations
                    lAmend = lAmend - lCellResult
```

```
                End If
                vAlloc(m) = lCellResult
            Next m
            If lAmend > 0 Then
                For m = 1 To j
                    lCellResult = vAlloc(m)
                    If lCellResult < 0 Then
                        vAlloc(m) = 0
                        sComment = sComment & "Allocation for " & m & " set to 0. "
                    ElseIf lCellResult > 0 And lAmend > 0 Then
                        If lCellResult > lAmend Then
                            vAlloc(m) = lCellResult - lAmend
                            lAmend = 0
                        Else
                            vAlloc(m) = 0
                            lAmend = lAmend - lCellResult
                        End If
                        sComment = sComment & "Allocation for " & m & " amended to " & _
                                   vAlloc(m) & ". "
                    End If
                Next m
            End If
        End If
        wsA.Cells(i, ac_Comment) = sComment
        For m = 1 To j
            wsA.Cells(i, ac_TeamStart + m - 1) = vAlloc(m)
        Next m

    End If

    i = i + 1
    lDemand = wsA.Cells(i, ac_Demand)

Loop

Range(wsT.Cells(1, tc_TeamStart), wsT.Cells(1, 250)).Copy Destination:=wsA.Cells(1, ac_TeamStart)

End With

End Sub
```

Stichprobe normalverteilen

Sie haben 11.256 Weihnachtsbäume. Ein Kunde möchte Ihnen 1.500 davon abkaufen. Die einzige Bedingung: die durchschnittliche Länge soll 6,50 Meter betragen. Sie würden nun gern die Restmenge Ihrer Weihnachtsbäume möglichst normalverteilt haben:



Wie können Sie dies erreichen?

Eine Beispielrechnung

	A	B	C	D	E	F	G	H
1		Vorgabe Gesamtzahl Entnahme:				1500		
2		Vorgabe Mittelwert Länge:				6,5		
3	**Länge**	**Anzahl**	Ideal wäre	Auto-Entnahme	Zwischen-ergebnis	**Entnahme**	**Rest**	Ideal wäre
4	5,60	20	0	20	0	10	10	0
5	5,70	40	3	37	3	15	25	3
6	5,80	40	14	26	14	8	32	14
7	5,90	72	59	16	56	8	64	56
8	6,00	148	192	0	148	0	148	179
9	6,10	372	497	0	372	0	372	456
10	6,20	876	1.016	0	876	0	876	918
11	6,30	1.660	1.644	200	1.460	165	1.495	1.460
12	6,40	2.160	2.102	323	1.837	281	1.879	1.837
13	6,50	2.276	2.125	449	1.827	416	1.860	1.827
14	6,60	1.820	1.698	384	1.436	383	1.437	1.436
15	6,70	1.036	1.073	143	893	143	893	893
16	6,80	464	536	25	439	28	436	439
17	6,90	212	212	41	171	43	169	171
18	7,00	48	66	0	48	0	48	52
19	7,10	12	16	0	12	0	12	13
20	7,20	0	3	0	0	0	0	2
21	7,30	0	0	0	0	0	0	0
22	**Total**	**11.256**	11.256	1664	9.592	1.500	**9.756**	9.756
23								
24	AVERAGE	6,45	6,45	6,47	6,45	6,50	6,45	6,45
25	STDEV.P	0,21	0,21		0,20		0,21	0,21
26	SKEW.P	-0,35	-0,00		-0,01		-0,20	-0,00
27	KURT	0,95	-0,02		0,03		0,53	-0,02

Tabellenblattformeln		
Bereich	Formel	
C4	**C4**	=RoundToSUm(NORM.VERT(A4:A21;B24;B25;FALSCH)*B22/10;0;;2)
D4	**D4**	=WENN(B4:B21-H4:H21<0;0;B4:B21-H4:H21)
E4	**E4**	=B4:B21-D4:D21
F4:F21	**F8**	=D8
G4	**G4**	=B4:B21-F4:F21
H4	**H4**	=RoundToSUm(NORM.VERT(A4:A21;(B24*B22-F2*F1)/(B22-F1);B25;FALSCH)*(B22-F1)/10;0;;2)
B22:H22	**B22**	=SUMME(B4:B21)
B24:H24	**B24**	=sbSWV($A24;$A$4:$A$21;B$4:B$21)
B25:C27;E25:E27;G25:H27	**B25**	=sbSWV($A25;$A$4:$A$21;B$4:B$21)

Angenommen, Sie haben die oben gezeigte Menge an Bäumen mit den angegebenen Längen. Eine erste interessante Rechnung ist sicherlich, inwieweit Ihre Ausgangsmenge bereits normalverteilt ist. Die Schiefe ermitteln wir mittels der Funktion sbSWV: sie beträgt gerundet *=sbSWV("SKEW.P";$A$4:$A$21;B$4:B$21) = -0,35*. Der Exzess (Maß für die Wölbung) beträgt gerundet *=sbSWV("KURT";$A$4:$A$21;B$4:B$21) = 0,95*. Wie wir am oben gezeigten Diagramm am gelb-orangen Graphen sehen können, ist diese Ausgangsmenge bereits "einigermaßen" normalverteilt.

Idealerweise wäre diese Stichprobe allerdings verteilt wie in Spalte C gezeigt mit der Formel *=MTRANS(RoundToSum(NORM.VERT(A4:A21;B24;B25;FALSCH)*B22/10;0))*: die Schiefe und der Exzess wären gleich Null (die vorgenommenen Rundungen führen hier zu leichten Abweichungen). Spalte H zeigt die ideale Restmengenverteilung nach Entnahme.

Mit der in Spalte D gezeigten automatischen Entnahme versuchen wir, diese ideale Restmenge möglichst zu erreichen. Dies kann natürlich nur gelingen, wenn wir ausreichend viele Bäume in den jeweiligen Längen zur Verfügung haben. Wo dies nicht der Fall ist können wir leider keine Bäume hinzufügen und müssen als Entnahmemenge 0 eintragen - im Diagramm sehen Sie z. B., dass die ideale Verteilung bei Länge 6,10 m höher ist als die tatsächliche Restverteilung.
Die ursprünglichen Formeln in Spalte F sollten =D4 bis =D21 lauten.

Diese überschreiben wir nun mit manuellen Werten, um

- auf eine Gesamtentnahme von genau 1.500 Bäumen,
- auf einen Mittelwert von 6,5 Länge der Bäume,
- auf etwa die gleiche Standardabweichung (Streuung) der Restmenge wie der Originalmenge,
- auf eine betragsmäßig kleinere Schiefe als bei der Originalmenge
- und auf einen betragsmäßig kleineren Exzess als bei der Originalmenge

zu kommen.

In der unten angebotenen Beispieldatei werden erhöhte Abweichungen durch bedingte Formatierungen angezeigt.

Anmerkung: Es ist nicht immer möglich, eine “hinreichend” normal verteilte Restmenge zu erhalten. Wie man leicht sehen kann, ist manchmal nicht einmal eine gewünschte Durchschnittslänge zu erreichen - fragen Sie bei der oben gezeigten Menge z. B. nach 21 Bäumen mit der Durchschnittslänge 5,60 m.

Hilfsfunktionen

Excel verfügt zwar über viele statistische Grundfunktionen. Diese sind jedoch nicht in der Lage, gewichtete Werte zu verarbeiten. Die hier verwendete benutzerdefinierte Funktion *sbSWV* (engl. statistics for weighted values) ermöglicht eine einfache und schnelle Anzeige, wie gut die Stichproben normalverteilt sind.

Damit die Summen der ganzzahligen Idealverteilungen der Stichproben genau mit den Summen der originalen Stichproben übereinstimmen, wurde die benutzerdefinierte Funktion *RoundToSum* verwendet. Man beachte, dass hierbei der Parameter 2 für den Fehlertyp zur Minimierung des relativen Fehlers gewählt wurde. Dies verhindert künstliche Rundungen zur “falschen” Seite in den Außenbereichen der Verteilungen.

sbSWV Programmcode

```
#Const SORTED = False

Function sbSWV(sStat As String, _
        ParamArray vInput() As Variant) As Variant
'Calculate some statistical measures of weighted values
'(C) (P) by Bernd Plumhoff 20-Aug-2024 PB V0.81
Dim d As Double, d2 As Double, dSum As Double
Dim i As Long, j As Long, k As Long, m As Long, n As Long
Dim vV, vV2, vV3, vW 'Variants

With Application.WorksheetFunction
vV = .Transpose(vInput(0))
Select Case sStat
Case "COVAR", "CORREL"
    vV2 = .Transpose(vInput(1))
    vW = .Transpose(vInput(2))
Case Else
    vW = .Transpose(vInput(1))
End Select
On Error GoTo errhdl
i = vV(1) 'Force error in case of vertical arrays
On Error GoTo 0
If UBound(vV) <> UBound(vW) Then
    'Arrays of values and of weights must have same dimension
    sbSWV = CVErr(xlErrNum)
    Exit Function
End If
Select Case UCase(sStat)
Case "AVERAGE"
    sbSWV = .SumProduct(vV, vW) / .Sum(vW)
Case "CORREL"
    vV3 = vV
    dSum = .Sum(vW)
    d = .SumProduct(vV, vW) / dSum
    d2 = .SumProduct(vV2, vW) / dSum
    For i = LBound(vV) To UBound(vV)
        vV3(i) = vW(i) * (vV(i) - d) * (vV2(i) - d2)
        vV(i) = vW(i) * (vV(i) - d) ^ 2#
        vV2(i) = vW(i) * (vV2(i) - d2) ^ 2#
    Next i
    sbSWV = .Sum(vV3) / Sqr(.Sum(vV) * .Sum(vV2))
Case "COVAR"
    dSum = .Sum(vW)
    d = .SumProduct(vV, vW) / dSum
    d2 = .SumProduct(vV2, vW) / dSum
    For i = LBound(vV) To UBound(vV)
        vV(i) = vW(i) * (vV(i) - d) * (vV2(i) - d2)
    Next i
    sbSWV = .Sum(vV) / dSum
Case "KURT"
    n = .Sum(vW)
    ReDim dV(1 To n) As Double
    k = 1
    For i = 1 To UBound(vW)
      For j = 1 To vW(i)
        dV(k) = vV(i)
        k = k + 1
      Next j
    Next i
    sbSWV = .Kurt(dV)
Case "MODE"
    k = .Max(vW)
    If k < 2 Then
        sbSWV = CVErr(xlErrNA)
        Exit Function
    End If
    sbSWV = vV(.Match(.Max(vW), vW, False))
Case "MEDIAN"
    If .Min(vW) < 1 Then
        sbSWV = CVErr(xlErrNA)
        Exit Function
    End If
    k = 0
    j = .Sum(vW)
    m = j Mod 2
    For i = LBound(vW) To UBound(vW)
        If vW(i) Mod 1 <> 0 Then
            sbSWV = CVErr(xlErrNum)
            Exit Function
        End If
        #If Not SORTED Then
            'Ensure ascending values in case input is unsorted.
            'This simple bubble sort leads to a quadratic runtime
            'but it's still quicker on 50 input values or more than
            'Lorimer Miller's nifty worksheet function approach
            '=LOOKUP(2,1/FREQUENCY(SUM(B1:B50)/2,SUMIF(A1:A50,"<="&A1:A50,B1:B50)),A1:A50)
            'BTW: Lorimer's approach is different from Excel's MEDIAN
            '(see below); and his other elegant array formula
            '=MEDIAN(IF(TRANSPOSE(ROW(A1:A1000))<=B1:B50,A1:A50))
            'calculates like Excel's MEDIAN but IMHO it's way too slow
            For n = i + 1 To UBound(vW)
                If vV(n) < vV(i) Then
                    d = vV(i)
                    vV(i) = vV(n)
                    vV(n) = d
                    d = vW(i)
                    vW(i) = vW(n)
                    vW(n) = d
                End If
            Next n
```

```
        #End If
        k = k + vW(i)
        Select Case 2 * k
        Case j + m
            If m = 0 Then
                #If Not SORTED Then
                    'Ensure vV(i + 1) is next greater value
                    For n = i + 2 To UBound(vW)
                        If vV(n) < vV(i + 1) Then
                            vV(i + 1) = vV(n)
                        End If
                    Next n
                #End If
                'Here Lorimer's function mentioned above would
                'return vV(i), the lower value
                sbSWV = (vV(i) + vV(i + 1)) / 2#
            Else
                sbSWV = vV(i)
            End If
            Exit Function
        Case Is > j + m
            sbSWV = vV(i)
            Exit Function
        End Select
    Next i
Case "SKEW.P"
    n = .Sum(vW)
    ReDim dV(1 To n) As Double
    k = 1
    For i = 1 To UBound(vW)
      For j = 1 To vW(i)
        dV(k) = vV(i)
        k = k + 1
      Next j
    Next i
    sbSWV = .Skew_p(dV)
Case "STDEV"
    dSum = .Sum(vW)
    d = .SumProduct(vV, vW) / dSum
    For i = LBound(vV) To UBound(vV)
        vV(i) = Abs(vV(i) - d) ^ 2#
    Next i
    sbSWV = Sqr(.SumProduct(vV, vW) / (dSum - 1#))
Case "STDEV.P"
    dSum = .Sum(vW)
    d = .SumProduct(vV, vW) / dSum
    For i = LBound(vV) To UBound(vV)
        vV(i) = Abs(vV(i) - d) ^ 2#
    Next i
    sbSWV = Sqr(.SumProduct(vV, vW) / dSum)
Case "VAR"
    dSum = .Sum(vW)
    d = .SumProduct(vV, vW) / dSum
    For i = LBound(vV) To UBound(vV)
        vV(i) = vW(i) * (vV(i) - d) ^ 2#
    Next i
    sbSWV = .Sum(vV) / (dSum - 1#)
Case Else
    sbSWV = CVErr(xlErrValue)
End Select
Exit Function
errhdl:
'Transpose variants to be able to address them
'with vV(i), not vV(i,1)
vV = .Transpose(vV)
vW = .Transpose(vW)
Select Case sStat
Case "COVAR", "CORREL"
    vV2 = .Transpose(vV2)
End Select
Resume Next
End With
End Function
```

Verteilung nach Restmenge

Die Budgets ausscheidender Mitarbeiter werden auf die verbliebenen gemäß deren bisherigen Budgets verteilt. Wie können Sie dies korrekt durchführen?

Ein simpler Ansatz

Eine simple Formel ist *=RUNDEN(C3*$B$2/$C$2;2)*, die Sie von D3 nach D12 hinunterkopieren können.

Sie können die Budgets der ausscheidenden Mitarbeiter in Spalte C löschen. Die Reihenfolge der Löschungen ist egal. Der offensichtliche Nachteil besteht in einem möglichen Rundungsfehler, weil die Summe gerundeter Summanden nicht notwendig der gerundeten Summe der nicht gerundeten Summanden entspricht. Dieses Beispiel zeigt eine Differenz von 0,02.

	A	B	C	D
1	**Name**	**Betrag**	**Löschung**	**Neuer Betrag**
2	**Summe**	**94.020,00**	**40.000,00**	**94.020,02**
3	Lehmann	49.000,00		-
4	Schulze	6.000,00	6.000,00	14.103,00
5	Schultze	5.750,00	5.750,00	13.515,38
6	Schmidt	5.500,00	5.500,00	12.927,75
7	Schmitt	5.270,00	5.250,00	12.340,13
8	Müller	5.000,00		-
9	Maier	4.750,00	4.750,00	11.164,88
10	Mayer	4.500,00	4.500,00	10.577,25
11	Meier	4.250,00	4.250,00	9.989,63
12	Meyer	4.000,00	4.000,00	9.402,00

Tabellenblattformeln		
Bereich	Formel	
B2:D2	**B2**	=SUMME(B3:B12)
D3	**D3**	=RUNDEN(C3:C12*$B$2/$C$2;2)

Eine korrekte Rechnung

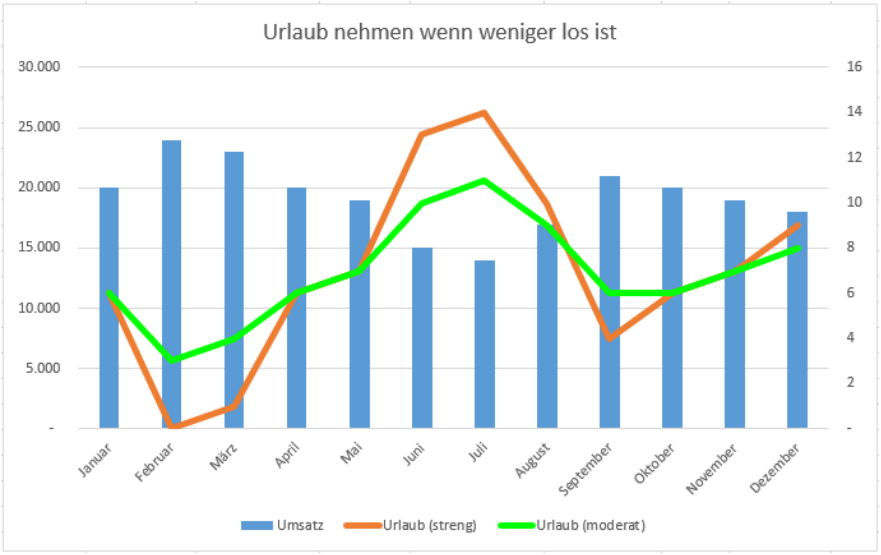
Mit der benutzerdefinierten Funktion *RoundToSum* können Sie die Spillformel *=RoundToSum(C4:C13*$B$3/$C$3;D1)* verwenden.

RoundToSum muss manchmal in die ‘falsche’ Richtung runden, aber dann wird dies wenigstens mit dem kleinstmöglichen Fehler getan.

	A	B	C	D
1	Runden auf Nachkommastellen:			-1
2	**Name**	**Betrag**	**Löschung**	**Neuer Betrag**
3	**Summe**	**94.020,00**	**40.000,00**	**94.020,00**
4	Lehmann	49.000,00		-
5	Schulze	6.000,00	6.000,00	14.100,00
6	Schultze	5.750,00	5.750,00	13.520,00
7	Schmidt	5.500,00	5.500,00	12.930,00
8	Schmitt	5.270,00	5.250,00	12.340,00
9	Müller	5.000,00		-
10	Maier	4.750,00	4.750,00	11.160,00
11	Mayer	4.500,00	4.500,00	10.580,00
12	Meier	4.250,00	4.250,00	9.990,00
13	Meyer	4.000,00	4.000,00	9.400,00

Tabellenblattformeln		
Bereich	Formel	
B3:D3	**D3**	=SUMME(D4:D13)
D4	**D4**	=RoundToSum(C4:C13*$B$3/$C$3;D1)

Urlaub nehmen wenn weniger los ist



Wenn Ihr Geschäft saisonal stark schwankt, können Sie den Urlaub Ihrer Belegschaft entsprechend planen und ggf. Saisonarbeitskräfte anstellen:

Hinweis: Selbstverständlich kann man keinem Mitarbeiter vorschreiben, wieviel Urlaub wann genommen werden muss. Diese Rechnungen sind lediglich Vorschläge, die als vernünftige Indikatoren dienen sollen.

Simples Beispiel

Wenn sie den maximalen Umsatzmonat (hier: 24.000) als Basis nehmen wollen, in dem kein Urlaub genommen werden sollte, und die restlichen Urlaubstage linear gemäß der Umsätze verteilen wollen:

	A	B	C	D	E	F	G	H
1			**Umsatzgrenze (kein Urlaub):**		**Höhere Umsatzgrenze (etwas Urlaub):**	Erhöht um auch Urlaub bei Umsatzmaximum zuzulassen.		
2			24.000		28.000			
3		**Umsatz**	**Urlaub (streng)**	**Ganz-zahlig**	**Urlaub (moderat)**	**Ganz-zahlig**		
4	**Total**	**230.000**	83		83			
5	**Januar**	20.000	5,7	6	6,3	6		
6	**Februar**	24.000	-	-	3,1	3		
7	**März**	23.000	1,4	1	3,9	4		
8	**April**	20.000	5,7	6	6,3	6		
9	**Mai**	19.000	7,2	7	7,0	7		
10	**Juni**	15.000	12,9	13	10,2	10		
11	**Juli**	14.000	14,3	14	11,0	11		
12	**August**	17.000	10,0	10	8,6	9		
13	**September**	21.000	4,3	4	5,5	6		
14	**Oktober**	20.000	5,7	6	6,3	6		
15	**November**	19.000	7,2	7	7,0	7		
16	**Dezember**	18.000	8,6	9	7,8	8		
17		**Prüfsumme**	**83,0**	**83**	**83,0**	**83**		

Tabellenblattformeln		
Bereich	Formel	
C5	**C5**	=(C$2-$B5:$B16)/(C$2*12-$B$4)*C$4
D5	**D5**	=RoundToSum(C5:C16;0)
E5	**E5**	=(E$2-$B5:$B16)/(E$2*12-$B$4)*E$4
F5	**F5**	=RoundToSum(E5:E16;0)
C17:F17	**C17**	=SUMME(C5:C16)

Komplexeres Beispiel

Hier werden alle Mitarbeiter einzeln berücksichtigt, auch hinsichtlich ihrer Anwesenheitsmonate.
In der letzten Tabelle kommt *RoundToSum* zum Einsatz, um summenerhaltend auf ganze Urlaubstage zu runden:

	A	B	C	D	E	F	G	H
1			Umsatzgrenze (kein Urlaub):		Höhere Werte erlauben auch Urlaub bei Maximalumsatz			
2			24.000					
3		Umsatz	Urlaub	Urlaub (ganze Tage)	Anwesenheit [x] und Urlaubsanspruch			
4	Total	230.000			Andrew	Benjamin	Charlie	David
5	Januar	20.000	5,7	6	x	x		x
6	Februar	24.000	-	-	x	x		x
7	März	23.000	1,4	1	x	x	x	x
8	April	20.000	5,7	6	x	x	x	x
9	Mai	19.000	7,2	7	x	x	x	
10	Juni	15.000	12,9	13	x	x	x	
11	Juli	14.000	14,3	14	x	x	x	
12	August	17.000	10,0	10	x	x	x	
13	September	21.000	4,3	4	x	x	x	x
14	Oktober	20.000	5,7	6	x	x	x	x
15	November	19.000	7,2	7	x		x	x
16	Dezember	18.000	8,6	9	x		x	x
17		Total	83,0	83	25,0	21,0	21,0	16,0
18								
19				Urlaub				
20				Total	Andrew	Benjamin	Charlie	David
21			Januar	6,1	1,8	1,9	-	2,5
22			Februar	-	-	-	-	-
23			März	1,3	0,3	0,3	0,3	0,4
24			April	7,8	1,8	1,9	1,6	2,5
25			Mai	6,2	2,1	2,2	1,9	-
26			Juni	11,5	3,9	4,1	3,5	-
27			Juli	12,4	4,2	4,4	3,8	-
28			August	8,9	3,0	3,1	2,7	-
29			September	5,2	1,2	1,3	1,1	1,6
30			Oktober	7,8	1,8	1,9	1,6	2,5
31			November	6,9	2,1	-	1,9	2,9
32			Dezember	8,9	2,7	-	2,5	3,7
33			Total	83,0	25,0	21,0	21,0	16,0
34								
35				Urlaub (ganze Tage)				
36				Total	Andrew	Benjamin	Charlie	David
37			Januar	7	2	2	-	3
38			Februar	-	-	-	-	-
39			März	-	-	-	-	-
40			April	8	2	2	2	2
41			Mai	6	2	2	2	-
42			Juni	11	4	4	3	-
43			Juli	13	4	5	4	-
44			August	9	3	3	3	-
45			September	5	1	1	1	2
46			Oktober	8	2	2	2	2
47			November	7	2	-	2	3
48			Dezember	9	3	-	2	4
49			Total	83	25	21	21	16

Tabellenblattformeln		
Bereich	Formel	
C5	**C5**	=($C$2-B5:B16)/($C$2*12-$B$4)*$C$17
D5	**D5**	=RoundToSum(C5:C16;0)
D21:D32;D37:D48	**D21**	=SUMME(E21:H21)
E21:H21	**E21**	=WENNFEHLER((E$5:E$16="x")*E$17*$D$5:$D$16/SUMME((E$5:E$16="x")*$D$5:$D$16);0)
D33:H33;D49:H49	**D33**	=SUMME(D21:D32)
E37:H37	**E37**	=WENNFEHLER(RoundToSum(E21:E32;0);0)

Zuweisen von Arbeitseinheiten vermindert um geleistete

Wie können Sie Ihren Mitarbeitern Arbeitseinheiten fair zuweisen, wenn Sie bereits geleistete Arbeit berücksichtigen wollen?

Gelbe Zellen sind Eingabezellen, grüne zeigen Zwischenergebnisse, und blaue kennzeichnen endgültige Ergebnisse. Hinweis: Sie müssen ‚Units done' in absteigender Reihenfolge eingeben.

In diesem Beispiel wurden bereits 90,6 Einheiten geliefert, aber 86 weitere Einheiten müssen noch 28 Lehrern zugewiesen werden. Ein fairer Anteil wäre für jeden Lehrer *(90.6 + 86) / 28 = 6,3*. Aber 7 Lehrer haben bereits mehr als das geliefert.

Spalte C zeigt die Ergebnisse mit Nachkommastellen. In Spalte D wurde mit Tabellenblattfunktionen auf ganze Zahlen gerundet, ohne dass sich die ursprüngliche Summe ändert.

Wie Sie leicht sehen können, zeigt Spalte E bessere Ergebnisse. Sie wurde mit der benutzerdefinierten Funktion *RoundToSum* erstellt.

	A	B	C	D	E
1	**Noch offene Arbeitseinheiten**	86			
2	**Total**	90,6	86	86	86
3	**Lehrer**	**Erledigt**	**Hilfsspalte**	**Noch offen (Formel)**	**Noch offen (RoundToSum)**
4	**Fairer Anteil**	6,307143			
5	Lecturer 1	12	0	0	0
6	Lecturer 2	11	0	0	0
7	Lecturer 3	9	0	0	0
8	Lecturer 4	8	0	0	0
9	Lecturer 5	8	0	0	0
10	Lecturer 6	7	0	0	0
11	Lecturer 7	7	0	0	0
12	Lecturer 8	6	0	0	0
13	Lecturer 9	5	0,43	0	1
14	Lecturer 10	3	2,43	3	3
15	Lecturer 11	3	2,43	2	3
16	Lecturer 12	2	3,43	4	4
17	Lecturer 13	2	3,43	3	4
18	Lecturer 14	2	3,43	4	4
19	Lecturer 15	2	3,43	3	4
20	Lecturer 16	2	3,43	3	4
21	Lecturer 17	1	4,43	5	4
22	Lecturer 18	0,6	4,83	5	5
23	Lecturer 19	0	5,43	5	5
24	Lecturer 20	0	5,43	6	5
25	Lecturer 21	0	5,43	5	5
26	Lecturer 22	0	5,43	5	5
27	Lecturer 23	0	5,43	6	5
28	Lecturer 24	0	5,43	5	5
29	Lecturer 25	0	5,43	6	5
30	Lecturer 26	0	5,43	5	5
31	Lecturer 27	0	5,43	6	5
32	Lecturer 28	0	5,43	5	5

Tabellenblattformeln		
Bereich	Formel	
B2:E2	**B2**	=SUMME(B5:B32)
B4	**B4**	=(B2+B1)/ZEILEN(B5:B32)
C5:C32	**C5**	=WENN(B5>=B6;MAX(0;B$4-B5-SUMMENPRODUKT(--(C$4:C4=0);B$4:B4-B$4)/(ZEILEN(B$5:B$32)-SUMMENPRODUKT(--(C$4:C4=0))+1));"Werte in Spalte B sind nicht absteigend!")
D5:D32	**D5**	=RUNDEN(SUMME(C$4:C5);0)-SUMME(D$4:D4)
E5	**E5**	=WENNFEHLER(RoundToSum(C5:C32;0);0)

RoundToSum im Vergleich

RoundToSum im Vergleich mit anderen "einfachen" Methoden

Es zirkulieren mehrere verschiedene naive Ansätze für das summenerhaltende Runden:

- (der schlechteste) Runde alle Werte mit Ausnahme des Letzten und ersetze dann den letzten Wert durch die Differenz der gerundeten Originalsumme minus der Summe der vorher gerundeten Werte (d.h. aggregiere alle Rundungsfehler im letzten Summanden):

	A	B	C	
1		**Originaldaten**	**Aggregiere Rundungsfehler**	**Formel in C**
2	**Total**	**2,594**	**2,59**	=SUMME(C4:C8)
3				
4		0,875	0,88	=RUNDEN(B4;2)
5		0,865	0,87	=RUNDEN(B5;2)
6		0,344	0,34	=RUNDEN(B6;2)
7		0,455	0,46	=RUNDEN(B7;2)
8		0,055	**0,04**	=RUNDEN(B$2;2)-SUMME(C$4:C7)

- (besser, aber immer noch schlecht) Wende das hintereinandergeschaltete (gleitende) Runden an:

	A	B	C	
1		**Originaldaten**	**Hintereinander-geschaltetes Runden**	**Formel in C**
2	**Total**	**2,593**	**2,59**	=SUMME(C4:C8)
3				
4		0,875	0,88	=RUNDEN(SUMME($B$3:$B4);2)-SUMME($C$3:$C3)
5		0,865	**0,86**	=RUNDEN(SUMME($B$3:$B5);2)-SUMME($C$3:$C4)
6		0,344	0,34	=RUNDEN(SUMME($B$3:$B6);2)-SUMME($C$3:$C5)
7		0,454	**0,46**	=RUNDEN(SUMME($B$3:$B7);2)-SUMME($C$3:$C6)
8		0,055	**0,05**	=RUNDEN(SUMME($B$3:$B8);2)-SUMME($C$3:$C7)

Vergleichen wir beide Ansätze mit *RoundToSum*.

Rechenbeispiel

Wir erzeugen 40 Zufallszahlen *ZUFALLSZAHL() * 1000* und vergleichen wie folgt:

	A	B	C	D	E	F	G	H	I	J
1			I	II	III	IV	V	VI	VII	VIII
2			Original	RoundToSum	Gleitendes Runden	Letzten Summanden anoassen	Einfaches Runden	Differenz II - V	Differenz III - V	Differenz IV - V
3		Summanen	948,5426666	948,54	948,54	948,54	948,54			
4			640,6107903	640,61	640,61	640,61	640,61			
5			604,8177225	604,82	604,82	604,82	604,82			
6			759,719267	759,72	759,72	759,72	759,72			
7			716,9320656	716,93	716,93	716,93	716,93			
8			263,431133	263,43	263,43	263,43	263,43			
9			726,0940269	726,09	726,10	726,09	726,09		0,01	
10			70,69027141	70,69	70,69	70,69	70,69			
11			468,6681995	468,67	468,67	468,67	468,67			
12			695,6816155	695,68	695,68	695,68	695,68			
13			68,51388814	68,51	68,51	68,51	68,51			
14			179,9413044	179,94	179,94	179,94	179,94			
15			994,1708842	994,17	994,17	994,17	994,17			
16			450,2225474	450,22	450,23	450,22	450,22		0,01	
17			875,4975592	875,50	875,49	875,5	875,5		-0,01	
18			217,4084507	217,41	217,41	217,41	217,41			
19			186,4643542	186,47	186,47	186,46	186,46	0,01	0,01	
20			428,5237989	428,52	428,52	428,52	428,52			
21			692,9424797	692,94	692,94	692,94	692,94			
22			460,6134853	460,61	460,62	460,61	460,61		0,01	
23			699,4999856	699,50	699,50	699,5	699,5			
24			512,7661261	512,77	512,76	512,77	512,77		-0,01	
25			173,039623	173,04	173,04	173,04	173,04			
26			385,9625179	385,96	385,96	385,96	385,96			
27			221,3543041	221,36	221,36	221,35	221,35	0,01	0,01	
28			945,2643498	945,27	945,26	945,26	945,26	0,01		
29			401,3771987	401,38	401,38	401,38	401,38			
30			666,2311689	666,23	666,23	666,23	666,23			
31			378,0140135	378,01	378,02	378,01	378,01		0,01	
32			446,3934267	446,39	446,39	446,39	446,39			
33			903,7448716	903,75	903,74	903,74	903,74	0,01		
34			987,4524282	987,45	987,46	987,45	987,45		0,01	
35			553,6299239	553,63	553,63	553,63	553,63			
36			349,8348857	349,84	349,83	349,83	349,83	0,01		
37			14,55826737	14,56	14,56	14,56	14,56			
38			152,9945856	153,00	152,99	152,99	152,99	0,01		
39			783,5934795	783,59	783,60	783,59	783,59		0,01	
40			178,9163192	178,92	178,91	178,92	178,92		-0,01	
41			922,6008936	922,60	922,60	922,6	922,6			
42			776,412911	776,41	776,42	776,47	776,41		0,01	0,06
43		Total	20903,12779	20.903,13	20.903,13	20.903,13	20.903,07	0,06	0,06	0,06
44	ABS Differenz zum Original			0,11	0,14	0,15	0,10			

Tabellenblattformeln		
Bereich	Formel	
D3	**D3**	=RoundToSum(C3:C42)
E3	**E3**	=RUNDEN(SUMME($C$3:$C3);2)-SUMME(E$2:E2)
F3	**F3**	=RUNDEN(C3:C41;2)
F42	**F42**	=RUNDEN(C43;2)-SUMME(F3:F41)
G3	**G3**	=RUNDEN(C3:C42;2)
H3:J3	**H3**	=WENN(ABS(D3:D42-$G3:$G42)<0,000001;"";D3:D42-$G3:$G42)
C43:J43	**C43**	=SUMME(C3:C42)
D44:G44	**D44**	=SUMMENPRODUKT(ABS(D3:D42-$C$3:$C$42))

Wie man sieht, erhalten wir einen aggregierten Rundungsfehler in Höhe von *0,06*, wenn wir alle Werte einzeln runden. Spalte J (VIII) zeigt die Differenz des aggregierten Rundungsfehlers von *0,06* im letzten Summanden. Spalte F (IV) zeigt die korrespondierenden gerundeten Werte. Im schlimmsten Fall würden Sie hier einen aggregierten Rundungsfehler von *n * 0,005* erhalten, wobei n die Anzahl der Zahlen ist. Beispiel: Nehmen Sie an Stelle der 40 Zufallszahlen 40-mal die Zahl *0,005*.

Gute Praxisbeispiele, warum man die Rundungsfehler nicht im letzten Summanden sammeln sollte, bieten normalverteilte Stichproben ganzer Zahlen. Siehe Kapital *Stichprobe normalverteilen*.

Der Ansatz des hintereinandergeschalteten (gleitenden) Rundens in Spalte I (VII) zeigt 12 Rundungen zur falschen Seite. Spalte E (III) zeigt die korrespondierenden gerundeten Werte. Beim hintereinandergeschalteten Runden können Sie im schlimmsten Fall die Hälfte der Zahlen zur

falschen Seite runden, obwohl alle Zahlen korrekt gerundet werden könnten. Beispiel: Nehmen Sie an Stelle der 40 Zufallszahlen 20-mal die Zahl *-0,0049999* und dann 20-mal die Zahl *0,0049999*.

Im Gegensatz dazu rundet das optimale *RoundToSum* lediglich 6 Werte zur falschen Seite und wendet die geringste Anzahl von Änderungen an, um die korrekte gerundete Gesamtsumme mit dem kleinsten absoluten Fehler zu erhalten. Im schlimmsten Fall würden Sie nun n/2 Male zur falschen Seite runden müssen, wobei n die Anzahl der Zahlen ist. Beispiel: Nehmen Sie an Stelle der 40 Zufallszahlen erneut 40-mal die Zahl *0,005*. Es ist jedoch die beste Lösung mit dem kleinsten absoluten Rundungsfehler für jede Zahl und danach mit der geringsten Anzahl von Rundungen zur falschen Seite.

Zusammenfassung

Verwenden Sie *RoundToSum*. Es wendet die geringste Anzahl von Änderungen an, um die korrekte gerundete Gesamtsumme mit dem kleinsten absoluten (oder relativen) Fehler zu erhalten.

Ein hintereinandergeschaltetes Runden benötigt kein VBA und auch keine Matrixformel, aber es benötigt mindestens so viele Rundungsabweichungen wie *RoundToSum* und es kann leicht wesentlich mehr unnatürliche Rundungen aufweisen, die man nur schwerlich erklären kann.

Am schlimmsten ist jedoch der Ansatz, alle Rundungsdifferenzen in einer (hier: der letzten) Zelle zu aggregieren. Man stelle sich *1.000* Menschen mit je *49* Cent in der Tasche (also insgesamt *490* EUR) vor. Wenn Sie diese Beträge fair auf ganze Euro gerundet verteilen wollen, würde bei diesem Ansatz der Letzte die gesamten *490* Euro erhalten. RoundToSum gäbe den ersten *490* Personen je einen Euro und allen anderen Null Euro.

RoundToSum im Vergleich zum D'Hondt Verfahren

RoundToSum implementiert das Hare-Niemeyer Verfahren. Dies ist in mancher Hinsicht hinsichtlich der fairen Mandatsverteilung dem D'Hondt Verfahren überlegen. So ist z. B. beim Hare-Niemeyer Verfahren die relative Prozentdifferenz zur idealen Verteilung absolut geringer:

	A	B	C	D	E	F
1		69	Sitze		Rel. % Differenz zur Idealverteilung	
2	Partei	Stimmen	D'Hondt	Hare-Niemeyer	D'Hondt	Hare-Niemeyer
3	A	576.100	30	29	3,175%	-0,265%
4	B	554.844	29	28	3,425%	-0,014%
5	C	94.920	4	5	-2,720%	0,719%
6	D	89.330	4	4	-1,749%	-1,749%
7	E	51.901	2	3	-2,131%	1,308%
8	Total	1.367.095	69	69		

Tabellenblattformeln		
Bereich	Formel	
C3	C3	=sbdHondt(B1;B3:B7)
D3	D3	=RoundToSum(B1*B3:B7/B8;0)
E3:F3	E3	=(C3:C7/C$8-$B3:$B7/$B$8)/($B3/$B$8)
B8:D8	B8	=SUMME(B3:B7)

Programmcode sbDHondt

```
Function sbdHondt(lSeats As Long, vVotes As Variant) As Variant
'Implements the d'Hondt method for allocating seats in
'party-list proportional representation political election
'systems.
'(C) (P) by Bernd Plumhoff 01-Dec-2009 PB V0.10
Dim i As Long, k As Long, n As Long
Dim vA As Variant, vB As Variant, vR As Variant
Dim dMax As Double

With Application.WorksheetFunction
vA = .Transpose(.Transpose(vVotes))
vB = vA
n = UBound(vA, 1)
ReDim vR(1 To n, 1 To 1) As Variant
ReDim lDenom(1 To n) As Long

Do While i < lSeats
  'identify max
  dMax = .Max(vB)
  k = .Match(dMax, vB, 0)
  lDenom(k) = lDenom(k) + 1
  vB(k, 1) = vA(k, 1) / (lDenom(k) + 1#)
  vR(k, 1) = vR(k, 1) + 1
  i = i + 1
Loop
sbdHondt = vR
End With
End Function
```

Literatur

Diaconis, P., & Freedman, D. (13. Juli 2007), On Rounding Percentages.

Sande, G. (2005, August 7), Guaranteed Controlled Rounding for Many Totals in Multi-way and Hierarchical Tables.

N. Herrmann, Mathematik ist überall, Oldenbourg Verlag München Wien, ISBN 3-486-57583-X (Kapitel 12, Das Wahl-Problem).

Zufallszahlen erzeugen

Abstract

Zufallszahlen benötigt man oft für Simulationen, für Was-wäre-wenn Rechnungen oder zur Anonymisierung von Daten. Hier stelle ich meine Sammlung von Programmen vor, die Zufallszahlen erzeugen: natürliche Zahlen, ganze Zahlen, oder Gleitkommazahlen. Da ich Excel's eingebaute Zufallsfunktionen verwende, erzeugen meine Programme Pseudozufallszahlen.

Ganze Zufallszahlen

Natürliche Zufallszahlen – *UniqRandInt*

Hinweis: Diese Funktion wird lediglich aus historischen Gründen gezeigt, weil sie effizient ausgeführt wird und weil mit ihr VBA Compilerkonstanten gelernt werden können. Die Funktion *sbRandInt* (siehe Ganze Zufallszahlen – *sbRandInt*) erlaubt auch negative Untergrenzen und ermittelt die beste Ausführungsart zur Ausführungszeit, nicht während der Compilierung.

Manchmal benötigt man ganze Zufallszahlen, die sich nicht oder lediglich begrenzt häufig wiederholen. Wenn Sie 20 positive ganze Zufallszahlen zwischen 1 und 100 benötigen, geben Sie ein: *=UniqRandInt(20;100)* ein. Falls der Bereich zwischen 100 und 199 liegen soll, dann *=UniqRandInt(20;100)+99*. Allgemein:

=UniqRandInt(Anzahl;Endwert - Startwert + 1) + Startwert - 1

Wenn Sie 10 natürliche Zufallszahlen im Bereich 1..2 benötigen, die bis zu 10-mal erscheinen dürfen, verwenden Sie *=UniqRandInt(10;2;10)*. In diesem Fall muss die Compilerkostante *ALLOW_REPETITION* auf *True* gesetzt sein. Und falls Sie nur 3 Zahlen zwischen 1 und 100 Millionen brauchen, die aber nicht mehrfach vorkommen dürfen, dann sollte die Compilerkonstante *LATE_INITIALISATION* auf *True* gesetzt sein:

	A	B	C	D	E	F
1	**n**	6	12	10	6	3
2	**lRange**	6	6	2	6	100.000.000
3	**lMaxOccurrence**	1	2	10	1	1
4						
5	**Ergebnis**	5	1	2	1	84.876.019
6		6	5	2	2	39.223.814
7		4	1	2	3	51.271.980
8		3	2	1	6	
9		1	4	1	5	
10		2	6	1	4	
11			5	1		
12			2	1		
13			3	2		
14			6	2		
15			4			
16			3			

Tabellenblattformeln		
Bereich	Formel	
B5:F5	**B5**	=MTRANS(UniqRandInt(B1;B2;B3))

UniqRandInt Programmcode

```
'If lRange >> n then set LATE_INITIALISATION to true. For example,
'if lRange=1,000,000 and if 1,000 cells are selected (n=1000).
#Const LATE_INITIALISATION = True
'If random integers may occur more than once, allow repetitions
#Const ALLOW_REPETITION = True

#If ALLOW_REPETITION Then
Function UniqRandInt(n As Long, ByVal lRange As Long, _
  Optional lMaxOccurence As Long = 1) As Variant
#Else
Function UniqRandInt(n As Long, ByVal lRange As Long) As Variant
#End If
'Returns n unique (=non-repeating) random integers within 1..lRange,
'lRange >= n. Set ALLOW_REPETITION = True and call with
'lMaxOccurences > 1 if random integers may occur more than once.
'(C) (P) by Bernd Plumhoff 30-Oct-2024 PB V1.04

Static bRandomized As Boolean
Dim vA             As Variant
Dim vR             As Variant
Dim i              As Long
Dim j              As Long
Dim lr             As Long

If Not bRandomized Then Randomize: bRandomized = True

#If ALLOW_REPETITION Then
  If lMaxOccurence < 1 Then
    UniqRandInt = CVErr(xlErrNum)
    Exit Function
  End If
  lRange = lRange * lMaxOccurence
#End If

If n > lRange Then UniqRandInt = CVErr(xlErrValue): Exit Function

ReDim vR(1 To n) As Variant

ReDim vA(1 To lRange)
#If Not LATE_INITIALISATION Then
  For i = 1 To lRange
    #If ALLOW_REPETITION Then
      vA(i) = Int((i - 1) / lMaxOccurence) + 1
    #Else
      vA(i) = i
    #End If
  Next i
#End If

i = 1
For j = 1 To UBound(vR, 1)
  lr = Int(((lRange - i + 1) * Rnd) + 1)
  #If LATE_INITIALISATION Then
    If vA(lr) = 0 Then
      #If ALLOW_REPETITION Then
        vR(j) = Int((lr - 1) / lMaxOccurence) + 1
      #Else
        vR(j) = lr
      #End If
    Else
  #End If
      vR(j) = vA(lr)
  #If LATE_INITIALISATION Then
    End If
    If vA(lRange - i + 1) = 0 Then
      #If ALLOW_REPETITION Then
        vA(lr) = Int((lRange - i + 1 - 1) / lMaxOccurence) + 1
      #Else
        vA(lr) = lRange - i + 1
      #End If
    Else
  #End If
      vA(lr) = vA(lRange - i + 1)
  #If LATE_INITIALISATION Then
    End If
  #End If
  i = i + 1
Next j

UniqRandInt = vR

End Function
```

Ganze Zufallszahlen – *sbRandInt*

Falls Sie ganzzahlige Zufallszahlen zwischen zwei gegebenen Werten benötigen, die sich nicht oder lediglich begrenzt häufig wiederholen, dann empfehle ich, meine benutzerdefinierte Funktion *sbRandInt* zu verwenden:

	A	B	C	D	E	F
1	**ICount**	7	14	10	12	3
2	**IMin**	-3	-3	1	1	-100.000.000
3	**IMax**	3	3	2	3	100.000.000
4	**IRept**	1	2	10	4	1
5						
6	**Ergebnis**	-1	-2	2	2	18.993.127
7		-2	0	1	3	-72.571.540
8		2	-1	2	1	-9.510.840
9		-3	2	1	3	
10		3	-3	2	1	
11		0	-3	1	1	
12		1	2	1	2	
13			1	2	2	
14			3	1	3	
15			3	2	1	
16			-1		2	
17			-2		3	
18			0			
19			1			

Tabellenblattformeln		
Bereich	Formel	
B6:F6	**B6**	=MTRANS(sbRandInt(B1;B2;B3;B4))

Anmerkung: Wenn der mögliche Zufallszahlenbereich wesentlich größer ist als die Anzahl zu erzeugenden Zufallszahlen, dann initialisiert *sbRandInt* seine Arrays zur Ausführungszeit verzögert, während man bei *UniqRandInt* (Natürliche Zufallszahlen – *UniqRandInt*) eine verzögerte Initialisierung mit einer Compilerkonstanten vor Programmausführung setzen muss.

sbRandInt – Programmcode

```
Function sbRandInt(ByVal lCount As Long, _
  lMin As Long, _
  lMax As Long, _
  Optional lRept As Long = 1) As Variant
'Returns lCount random integers between lMin and lMax, each one
'occurring zero to lRept times. lMax - lMin + 1 must be greater
'or equal to lCount.
'Error values:
'#NUM!   - lRept is less than 1
'#REF!   - lCount is greater than (lMax - lMin + 1) * lRept
'#VALUE! - lCount is less than 1
'(C) (P) by Bernd Plumhoff  30-Dec-2024 PB V1.02
Static bRandomized As Boolean
Dim i As Long, j As Long, k As Long
Dim lRnd As Long, lRange As Long
Const CLateInitFactor = 50

If lCount < 1 Then sbRandInt = CVErr(xlErrValue): Exit Function
If lRept < 1 Then sbRandInt = CVErr(xlErrNum): Exit Function
If lCount > (lMax - lMin + 1) * lRept Then sbRandInt = CVErr(xlErrRef): Exit Function

lRange = (lMax - lMin + 1) * lRept

ReDim lr(1 To lCount) As Long

If Not bRandomized Then Randomize: bRandomized = True

ReDim lT(1 To lRange) As Long
'If we have a huge range of possible random integers and a comparably
'small number of draws, i.e. if (lMax - lMin) * lRept >> lCount
'then we can save some runtime with late initialization.
If lRange / lCount < CLateInitFactor Then
  For i = 1 To lRange
    lT(i) = Int((i - 1) / lRept) + lMin
  Next i
End If

i = 1
If lRange / lCount < CLateInitFactor Then
  For k = 1 To UBound(lr)
    lRnd = Int(((lRange - i + 1) * Rnd) + 1)
    lr(k) = lT(lRnd)
    lT(lRnd) = lT(lRange - i + 1)
    i = i + 1
  Next k
Else
  j = lMin: If lMin <= 0 And lMax >= 0 Then j = 1
  For k = 1 To UBound(lr)
    lRnd = Int(((lRange - i + 1) * Rnd) + 1)
    If lT(lRnd) = 0 Then
      lr(k) = Int((lRnd - 1) / lRept) + j
    Else
      lr(k) = lT(lRnd)
    End If
    If lT(lRange - i + 1) = 0 Then
      lT(lRnd) = Int((lRange - i) / lRept) + j
    Else
      lT(lRnd) = lT(lRange - i + 1)
    End If
    i = i + 1
  Next k
  'If lRange includes zero we need to shift result array
  If lMin <= 0 And lMax >= 0 Then
    For k = 1 To UBound(lr)
      lr(k) = lr(k) + lMin - 1
    Next k
  End If
End If

sbRandInt = lr

End Function
```

Zufallszahlen mit einer festgelegten Summe

Minimum für die Zufallszahlen vorgegeben - sbLongRandSumN

Sie benötigen 20 natürliche Zufallszahlen mit der Summe 100? Dann schlage ich meine hier gezeigte benutzerdefinierte Funktion *sbLongRandSumN* vor. Sie können beliebig viele ganze Zahlen mit einer vorgegebenen Summe erzeugen, wobei die erzeugten Zahlen ein spezifiertes Minimum nicht unterschreiten dürfen:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	lSum	lCount	lMin	Total	sbLongRandSumN									
2	92	7	6	92	8	6	9	7	17	13	32			
3	87	6	5	87	6	5	5	5	11	55				
4	58	4	7	58	12	8	8	30						
5	21	3	2	21	9	9	3							
6	65	10	4	65	5	4	5	5	4	4	5	4	4	25
7	64	3	12	64	35	16	13							
8	46	3	15	46	16	15	15							
9	83	3	16	83	23	16	44							
10	59	10	5	59	6	5	5	5	5	5	5	5	8	10

Tabellenblattformeln		
Bereich	Formel	
D2:D10	**D2**	=SUMME(E2:K2)
E2:E10	**E2**	=sbLongRandSumN(A2;B2;C2)

sbLongRandSumN Programmcode

```
Function sbLongRandSumN(lSum As Long, _
  ByVal lCount As Long, _
  Optional ByVal lMin As Long = 0) As Variant
'Generates lCount random integers greater equal lMin
'which sum up to lSum.
'(C) (P) by Bernd Plumhoff 26-Apr-2013 PB V0.1
Dim i As Long
Dim lSumRest As Long

If lCount * lMin > lSum Then
  sbLongRandSumN = CVErr(xlErrNum)
  Exit Function
End If
If lCount < 1 Then
  sbLongRandSumN = CVErr(xlErrValue)
  Exit Function
End If
Randomize
ReDim vR(1 To lCount) As Variant
lSumRest = lSum
For i = lCount To 2 Step -1
  vR(i) = lMin + Int(Rnd * (lSumRest - lMin * i))
  lSumRest = lSumRest - vR(i)
Next i
vR(1) = lSumRest
sbLongRandSumN = vR
End Function
```

Minimum und Maximum für die Zufallszahlen vorgegeben - sbRandIntFixSum

Sie können *lCount* ganze Zufallszahlen zwischen *lMin* und *lMax* mit der Summe *lSum* mit Tabellenblattfunktionen erzeugen:

	A	B	C	D	E	F
1	**Summe**	1000			**Prüfung**	1000
2	**Untergrenze**	30				
3	**Obergrenze**	110				
4	**Anzahl**	10				
5				**Unter-**	**Ober-**	
6			**Anzahl**	**grenze**	**grenze**	**Zufalls-**
7	**Runde**	**Rest**	**Rest**	**Rest**	**Rest**	**zahl**
8	0	1000	10	30	110	
9	1	1000	10	30	110	110
10	2	890	9	30	110	59
11	3	831	8	61	110	101
12	4	730	7	70	110	101
13	5	629	6	79	110	102
14	6	527	5	87	110	99
15	7	428	4	98	110	105
16	8	323	3	103	110	104
17	9	219	2	109	110	110
18	10	109	1	109	109	109

Tabellenblattformeln		
Bereich	Formel	
F1	**F1**	=SUMME(WENNFEHLER(F9:F29;0))
A8	**A8**	=SEQUENZ(B4+1;;0)
B8	**B8**	=$B$1
B9:B29	**B9**	=B8-F8
C8	**C8**	=$B$4
C9:C29	**C9**	=C8-1*(A9<>1)
D8	**D8**	=$B$2
D9:D29	**D9**	=AUFRUNDEN(MAX(D8;MIN(B9/C9;B9/C9-(C9-1)*(E8-B9/C9)));0)
E8	**E8**	=$B$3
E9:E29	**E9**	=ABRUNDEN(MIN(E8;MAX(B9/C9;B9/C9+(C9-1)*(B9/C9-D8)));0)
F9:F29	**F9**	=GANZZAHL(ZUFALLSZAHL()*(E9-D9+1)+D9)

Mit der benutzerdefinierten Funktion *sbRandFixIntSum* könnten Sie stattdessen aber auch in Zelle F9 eingeben: *=sbRandIntFixSum(B1;B2;B3;B4)*

sbRandIntFixSum Programmcode

```
Function sbRandIntFixSum(lSum As Long, lMin As Long, _
  lMax As Long, Optional lCount As Long = 0, _
  Optional bUseRandTriang As Boolean = True, _
  Optional bVolatile As Boolean = False) As Variant
'Returns lCount (or selected cell count in case a range is select when
'called as a matrix formula) random integers between lMin and lMax
'which sum up to lSum. If bUseRandTriang the sbRandTriang distribution
'is used to "bias" the randomness to be "less extreme".

'Error values:
'#NUM!   - No solution exists
'#VALUE! - lCount is less than 1
'(C) (P) by Bernd Plumhoff 05-Aug-2020 PB V0.3

Dim i As Long
Dim lRnd As Long, lMinPrev As Long
Dim lRow As Long, lCol As Long

With Application

If TypeName(.Caller) = "Range" And lCount = 0 Then
  lCount = .Caller.Count
  ReDim lR(1 To .Caller.Rows.Count, 1 To .Caller.Columns.Count) As Long
ElseIf lCount < 1 Then
  sbRandIntFixSum = CVErr(xlErrValue)
  Exit Function
Else
  ReDim lR(1 To lCount, 1 To 1) As Long
End If

Randomize
If bVolatile Then .Volatile

For lRow = 1 To UBound(lR, 1)
  For lCol = 1 To UBound(lR, 2)
    lMinPrev = lMin
    lMin = .RoundUp(.Max(lMin, .Min(lSum / lCount, lSum / lCount _
           - (lCount - 1) * (lMax - lSum / lCount))), 0)
    lMax = .RoundDown(.Min(lMax, .Max(lSum / lCount, lSum / lCount _
           + (lCount - 1) * (lSum / lCount - lMinPrev))), 0)
    If lMin > lMax Or lSum / lCount <> .Median(lMin, lMax, lSum / lCount) Then
      'No solution exists
      sbRandIntFixSum = CVErr(xlErrNum)
      Exit Function
    End If
    If bUseRandTriang Then
      If lMin = lMax Then
        lRnd = lMin
      Else
        lRnd = Int(sbRandTriang(CDbl(lMin), lSum / lCount, CDbl(lMax)) + 0.5)
      End If
    Else
      lRnd = Int(Rnd() * (lMax - lMin + 1) + lMin)
    End If
    lR(lRow, lCol) = lRnd
    lSum = lSum - lRnd
    lCount = lCount - 1
  Next lCol
Next lRow

sbRandIntFixSum = lR
End With

End Function
```

Praktische Anwendungen ganzer Zufallszahlen

Monte Carlo Simulation für eine faire Teamverteilung – *sbGenerateTeams*

Sie und Ihre 15 Freunde wollen in 4 Teams spielen mit je 4 Spielern und Sie fragen sich, wie Sie die Teams zufällig aber möglichst gleichstark aufstellen können?

Dies kann man mit dem Programm *sbGenerateTeams* erreichen:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Spieler**	**Name**	**Stärke**		**Anzahl Teams**		**Anzahl Monte Carlo Läufe**		**Team**	**Spieler**	**Stärke**		**Team**	**Summe Stärke**
2	1	Andrew	27		4		20000		1	Isaac	39		1	141
3	2	Benjamin	38						1	George	41		2	140
4	3	Charlie	31		**Spieler pro Team**		Erzeuge Teams		1	Mary	26		3	140
5	4	David	47		4				1	Edward	35		4	140
6	5	Edward	35						2	Lucy	45		StDev	0,5
7	6	Frederick	26						2	David	47			
8	7	George	41						2	Frederick	26			
9	8	Harry	43						2	Oliver	22			
10	9	Isaac	39						3	Nellie	50			
11	10	Jack	44						3	King	25			
12	11	King	25						3	Benjamin	38			
13	12	Lucy	45						3	Andrew	27			
14	13	Mary	26						4	Charlie	31			
15	14	Nellie	50						4	Peter	22			
16	15	Oliver	22						4	Harry	43			
17	16	Peter	22						4	Jack	44			

Dieses Programm vereint mehrere Funktionalitäten, die ich gern nutze:

- Die Klasse *SystemState* reduziert die Laufzeit.
- Mit Enumerierungen organisiere ich den Zugriff auf Spalten flexibel - für zusätzliche oder entfallende Spalten ändere ich lediglich die Enumerierung, und das Programm passt die Spaltennummern automatisch an.
- Neues Mischen einer Menge von Elementen mit *UniqRandInt* (Natürliche Zufallszahlen – *UniqRandInt*).
- Testdaten (Namen) erzeugte ich mit *sbGenerateTestData* (Testdaten erzeugen – *sbGenerateTestData*).

Ein komplexeres Beispiel

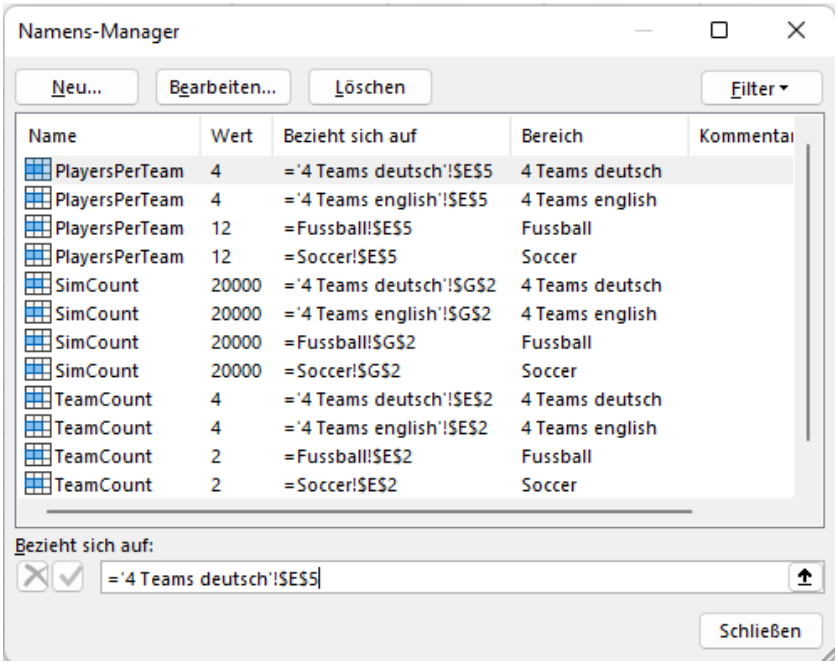
Falls Sie zufällige Teams gleicher Stärke generieren möchten, die Untergruppen verschiedener Spielerarten haben, können Sie Spielstärkewerte mit unterschiedlichen Zehnerpotenzen (oder andere Potenzen) je Untergruppe vergeben. Sie müssen lediglich darauf achten, dass alle Untergruppen in allen Teams dieselbe Spieleranzahl haben - Ausnahme: die Untergruppe mit den kleinsten Spielstärkewerten kann unterschiedlich viele Spieler in den Teams haben:

Spieler	Name	Stärke		Anzahl Teams		Anzahl Monte Carlo Läufe		Team	Spieler	Stärke		Team	Summe Stärke
1	Torwart 1	50000		2		20000		1	Mittelfeld 6	500		1	68550
2	Torwart 2	50000						1	Verteidiger 6	5000		2	70300
3	Verteidiger 1	5000		Spieler pro Team		Erzeuge Teams		1	Mittelfeld 5	500		StDev	1237,436867
4	Verteidiger 2	5000		12				1	Torwart 1	50000			
5	Verteidiger 3	5000						1	Stürmer 6	50			
6	Verteidiger 4	5000						1	Verteidiger 4	5000			
7	Verteidiger 5	5000						1	Mittelfeld 2	500			
8	Verteidiger 6	5000						1	Verteidiger 8	500			
9	Verteidiger 7	5000						1	Mittelfeld 1	500			
10	Verteidiger 8	500						1	Mittelfeld 4	500			
11	Mittelfeld 1	500						1	Mittelfeld 3	500			
12	Mittelfeld 2	500						1	Verteidiger 3	5000			
13	Mittelfeld 3	500						2	Stürmer 3	50			
14	Mittelfeld 4	500						2	Verteidiger 5	5000			
15	Mittelfeld 5	500						2	Verteidiger 7	5000			
16	Mittelfeld 6	500						2	Verteidiger 2	5000			
17	Stürmer 1	50						2	Verteidiger 1	5000			
18	Stürmer 2	50						2	Stürmer 5	50			
19	Stürmer 3	50						2	Stürmer 4	50			
20	Stürmer 4	50						2	Stürmer 7	50			
21	Stürmer 5	50						2	Stürmer 2	50			
22	Stürmer 6	50						2	[Empty]	0			
23	Stürmer 7	50						2	Stürmer 1	50			
								2	Torwart 2	50000			

Sie können nach einem Spiel die Spielstärkewerte anpassen. Zum Beispiel könnten Sie die Werte der Gewinner um 1 erhöhen, bis ein Maximalwert je Untergruppe erreicht ist. Oder Sie verringern die Werte für die Verlierer um 1, bis ein Minimalwert je Untergruppe erreicht ist. So stellen Sie sicher, dass auch Spielstärkeänderungen fair und nachvollziehbar abgebildet werden.

sbGenerateTeams Programmcode

Bitte beachten: Dieses Programm benötigt (verwendet) die Klasse SystemState und die benutzerdefinierte Funktion *UniqRandInt* und ist auf die vergebenen Bereichsnamen abgestimmt:



```
#Const I_Want_Colors = True

#If I_Want_Colors Then
Private Enum xlCI 'Excel Color Index
: xlCIBlack = 1: xlCIWhite: xlCIRed: xlCIBrightGreen: xlCIBlue '1 - 5
: xlCIYellow: xlCIPink: xlCITurquoise: xlCIDarkRed: xlCIGreen '6 - 10
: xlCIDarkBlue: xlCIDarkYellow: xlCIViolet: xlCITeal: xlCIGray25 '11 - 15
: xlCIGray50: xlCIPeriwinkle: xlCIPlum: xlCIIvory: xlCILightTurquoise '16 - 20
: xlCIDarkPurple: xlCICoral: xlCIOceanBlue: xlCIIceBlue: xlCILightBrown '21 - 25
: xlCIMagenta2: xlCIYellow2: xlCICyan2: xlCIDarkPink: xlCIDarkBrown '26 - 30
: xlCIDarkTurquoise: xlCISeaBlue: xlCISkyBlue: xlCILightTurquoise2: xlCILightGreen '31 - 35
: xlCILightYellow: xlCIPaleBlue: xlCIRose: xlCILavender: xlCITan '36 - 40
: xlCILightBlue: xlCIAqua: xlCILime: xlCIGold: xlCILightOrange '41 - 45
: xlCIOrange: xlCIBlueGray: xlCIGray40: xlCIDarkTeal: xlCISeaGreen '46 - 50
: xlCIDarkGreen: xlCIGreenBrown: xlCIBrown: xlCIDarkPink2: xlCIIndigo '51 - 55
: xlCIGray80 '56
End Enum
#End If

Enum col_worksheet
    col_LBound = 0 'To be able to iterate from here + 1
    col_in_player_no
    col_in_player_name
    col_in_player_skill
    col_blank_1
    col_in_team_stats
    col_blank_2
    col_in_sim_stats
    col_blank_3
    col_out_team_no
    col_out_player_name
    col_out_player_skill
    col_blank_4
    col_stat_team_no
    col_stat_sum_skills
    col_Ubound 'To be able to iterate until here - 1
End Enum 'col_worksheet

Sub sbGenerateTeams()
'Implements a simple Monte Carlo simulation to randomly generate
'teams fairly, keeping track of the teams with the lowest standard
'deviation of skill sums.
'This sub needs UniqRandInt - google for sulprobil and uniqrandint.
'and the SystemState class - google for sulprobil and systemstate.
'(C) (P) by Bernd Plumhoff 07-Nov-2024 PB V0.5

Dim i                   As Long
Dim j                   As Long
Dim k                   As Long
Dim n                   As Long
Dim teamcount           As Long
Dim playersperteam      As Long
Dim stdev_hc_sum        As Double
Dim min_stdev           As Double
Dim s                   As Double
Dim v                   As Variant
Dim wsI                 As Worksheet
Dim state               As SystemState

'Initialize
Set state = New SystemState
Set wsI = ThisWorkbook.ActiveSheet
teamcount = wsI.Range("TeamCount")
wsI.Range("PlayersPerTeam").Calculate
playersperteam = wsI.Range("PlayersPerTeam")
n = teamcount * playersperteam
ReDim hc(1 To n) As Double
ReDim mina(1 To n) As Double
ReDim hc_sum(1 To teamcount) As Double
wsI.Cells.Interior.ColorIndex = False
#If I_Want_Colors Then
wsI.Range("A1:C1").Interior.ColorIndex = xlCIYellow
wsI.Range("E1").Interior.ColorIndex = xlCIYellow
wsI.Range("G1").Interior.ColorIndex = xlCIYellow
wsI.Range("E4").Interior.ColorIndex = xlCIYellow
wsI.Range("E2").Interior.ColorIndex = xlCILightYellow
wsI.Range("G2").Interior.ColorIndex = xlCILightYellow
wsI.Range("E5").Interior.ColorIndex = xlCILightYellow
wsI.Range("I1:K1").Interior.ColorIndex = xlCIBrightGreen
wsI.Range("M1:N1").Interior.ColorIndex = xlCIBrightGreen
wsI.Range("M" & teamcount + 2 & ":N" & teamcount + 2).Interior.ColorIndex = xlCILightGreen
#End If
For j = 1 To n
  hc(j) = wsI.Cells(j + 1, col_in_player_skill)
  #If I_Want_Colors Then
  wsI.Range("A" & j + 1 & ":C" & j + 1).Interior.ColorIndex = xlCILightYellow
  #End If
Next j
min_stdev = 1E+308
```

```
k = 1
Do
  v = UniqRandInt(n, n)
  For i = 1 To teamcount
    hc_sum(i) = 0
    For j = 1 To playersperteam
      hc_sum(i) = hc_sum(i) + hc(v((i - 1) * playersperteam + j))
    Next j
  Next i
  stdev_hc_sum = WorksheetFunction.StDev(hc_sum)
  If stdev_hc_sum < min_stdev Then
    For i = 1 To n
      mina(i) = v(i)
    Next i
    min_stdev = stdev_hc_sum
    Application.StatusBar = "Iteration " & k & ", new min stdev = " & min_stdev
  End If
  k = k + 1
Loop Until k > wsI.Range("SimCount")

wsI.Range(wsI.Cells(2, col_out_team_no), _
  wsI.Cells(1000, col_stat_sum_skills)).ClearContents

For i = 1 To teamcount
  s = 0#
  For j = 1 To playersperteam
    wsI.Cells(1 + (i - 1) * playersperteam + j, col_out_team_no) = i
    wsI.Cells(1 + (i - 1) * playersperteam + j, col_out_player_name) = _
      IIf("" = wsI.Cells(1 + mina((i - 1) * playersperteam + j), col_in_player_name), _
        "[Empty]", wsI.Cells(1 + mina((i - 1) * playersperteam + j), col_in_player_name))
    wsI.Cells(1 + (i - 1) * playersperteam + j, col_out_player_skill) = _
      CDbl(wsI.Cells(1 + mina((i - 1) * playersperteam + j), col_in_player_skill))
    s = s + wsI.Cells(1 + mina((i - 1) * playersperteam + j), col_in_player_skill)
    #If I_Want_Colors Then
    wsI.Range("I" & 1 + (i - 1) * playersperteam + j & ":K" & 1 + (i - 1) * _
      playersperteam + j).Interior.ColorIndex = xlCILightGreen
    #End If
  Next j
  wsI.Cells(1 + i, col_stat_team_no) = i
  wsI.Cells(1 + i, col_stat_sum_skills) = s
  #If I_Want_Colors Then
  wsI.Range("M" & i + 1 & ":N" & i + 1).Interior.ColorIndex = xlCILightGreen
  #End If
Next i
wsI.Cells(2 + teamcount, col_stat_team_no) = "StDev"
wsI.Cells(2 + teamcount, col_stat_sum_skills) = min_stdev
End Sub
```

Monte Carlo Simulation für einen Regatta Flight Plan – *sbRegattaFlightPlan*

Falls Sie einen Regatta Flight Plan für 12 Flights (Segelrunden) für 12 Einzelsegler bei 4 vorhandenen Booten erzeugen wollen, wobei

- kein Segler gegen einen anderen zu häufig antritt
- kein Segler ein Boot zu häufig zugewiesen bekommt
- möglichst kein Segler in aufeinanderfolgenden Flights segeln muss

dann hilft Ihnen hoffentlich dieses Programm, welches UniqRandInt verwendet.

Eingabe:

	A	B
1		
2	Starte Simulation	
3		
4		
5	**Anzahl der Simulationen**	20.000
6	**Anzahl Flights**	12
7	**Anzahl Boote**	4
8	**Die Segler:**	
9	Abe	
10	Ben	
11	Carl	
12	Dan	
13	Eve	
14	Fred	
15	Giles	
16	Hanna	
17	Ida	
18	Joe	
19	Kim	
20	Luke	

Ausgabe:

	D	E	F	G	H	I	J	K	L	M	N	O	P
1		Flight 1	Flight 2	Flight 3	Flight 4	Flight 5	Flight 6	Flight 7	Flight 8	Flight 9	Flight 10	Flight 11	Flight 12
2	Boot 1	Dan	Ida	Carl	Joe	Dan	Eve	Hanna	Carl	Fred	Hanna	Eve	Luke
3	Boot 2	Joe	Abe	Hanna	Luke	Hanna	Carl	Kim	Abe	Giles	Dan	Giles	Fred
4	Boot 3	Eve	Kim	Fred	Giles	Ben	Abe	Ben	Dan	Ida	Abe	Joe	Carl
5	Boot 4	Giles	Luke	Ben	Kim	Ida	Fred	Luke	Joe	Eve	Kim	Ida	Ben
6	Maximale Anzahl von sich erneut treffenden Seglern	3											
7	Maximale Anzahl von Booten die ein Segler erneut nutzt	2											
8	Anzahl von Seglern mit angrenzenden Flights	0											

sbRegattaFlightPlan Programmcode

Dieses Programm benötigt *UniqRandInt* und verwendet die Klasse *SystemState*.

```
#Const I_Want_Colors = True
#If I_Want_Colors Then
Private Enum xlCI 'Excel Color Index
: xlCIBlack = 1: xlCIWhite: xlCIRed: xlCIBrightGreen: xlCIBlue '1 - 5
: xlCIYellow: xlCIPink: xlCITurquoise: xlCIDarkRed: xlCIGreen '6 - 10
: xlCIDarkBlue: xlCIDarkYellow: xlCIViolet: xlCITeal: xlCIGray25 '11 - 15
: xlCIGray50: xlCIPeriwinkle: xlCIPlum: xlCIIvory: xlCILightTurquoise '16 - 20
: xlCIDarkPurple: xlCICoral: xlCIOceanBlue: xlCIIceBlue: xlCILightBrown '21 - 25
: xlCIMagenta2: xlCIYellow2: xlCICyan2: xlCIDarkPink: xlCIDarkBrown '26 - 30
: xlCIDarkTurquoise: xlCISeaBlue: xlCISkyBlue: xlCILightTurquoise2: xlCILightGreen '31 - 35
: xlCILightYellow: xlCIPaleBlue: xlCIRose: xlCILavender: xlCITan '36 - 40
: xlCILightBlue: xlCIAqua: xlCILime: xlCIGold: xlCILightOrange '41 - 45
: xlCIOrange: xlCIBlueGray: xlCIGray40: xlCIDarkTeal: xlCISeaGreen '46 - 50
: xlCIDarkGreen: xlCIGreenBrown: xlCIBrown: xlCIDarkPink2: xlCIIndigo '51 - 55
: xlCIGray80 '56
End Enum
Private xlFC(1 To 56) As Boolean 'Font color: True is black, False is white
#End If

Sub sbRegattaFlightPlan()
'Performs a simple Monte Carlo simulation to create a regatta flight plan.
'(C) (P) by Bernd Plumhoff 07-Jan-2023 PB V0.3
Dim i As Long, j As Long, k As Long, m As Long
Dim lAdjacentFlights        As Long
Dim lBestSailorInBoat       As Long
Dim lBestSailorMeetSailor   As Long
Dim lBoatCount              As Long
Dim lFlightCount            As Long
Dim lLowestAdjacentFlights  As Long
Dim lMaxSailorInBoat        As Long
Dim lMaxSailorMeetSailor    As Long
Dim lSailorIndex            As Long
Dim lSailorCount            As Long
Dim lSimulationCount        As Long
Dim ws                      As Worksheet
Dim state                   As SystemState

With Application.WorksheetFunction

'Initialize
Set ws = ThisWorkbook.ActiveSheet
Set state = New SystemState
ws.Cells.Interior.Pattern = xlNone
ws.Cells.Interior.TintAndShade = 0
ws.Cells.Interior.PatternTintAndShade = 0
ws.Cells.Font.ColorIndex = xlAutomatic
ws.Cells.Font.TintAndShade = 0

#If I_Want_Colors Then
For i = 1 To 56: xlFC(i) = True: Next i
xlFC(xlCIBlack) = False: xlFC(xlCIRed) = False: xlFC(xlCIBlue) = False
xlFC(xlCIDarkRed) = False: xlFC(xlCIGreen) = False: xlFC(xlCIDarkBlue) = False
xlFC(xlCIDarkYellow) = False: xlFC(xlCIViolet) = False: xlFC(xlCIDarkPurple) = False
xlFC(xlCILightBrown) = False: xlFC(xlCIDarkPink) = False: xlFC(xlCIDarkBrown) = False
xlFC(xlCISeaBlue) = False: xlFC(xlCIBlueGray) = False: xlFC(xlCIDarkTeal) = False
xlFC(xlCIDarkGreen) = False: xlFC(xlCIGreenBrown) = False: xlFC(xlCIIndigo) = False
xlFC(xlCIGray80) = False
#End If

Randomize
i = Range("Sailors").Row + 1
Do While Not IsEmpty(ws.Cells(i + lSailorCount, 1))
  lSailorCount = lSailorCount + 1
Loop
ReDim sSailor(1 To lSailorCount) As String
i = Range("Sailors").Row
j = 1
Do While Not IsEmpty(ws.Cells(i + j, 1))
  sSailor(j) = ws.Cells(i + j, 1)
  #If I_Want_Colors Then
  k = (j Mod 56) + 1
  ws.Cells(i + j, 1).Interior.ColorIndex = k
  If xlFC(k) Then
    ws.Cells(i + j, 1).Font.ColorIndex = xlCIBlack
  Else
    ws.Cells(i + j, 1).Font.ColorIndex = xlCIWhite
  End If
  #End If
  j = j + 1
Loop

lBoatCount = Range("Boats")
```

```
lFlightCount = Range("Flights")
lSimulationCount = Range("Simulations")
lBestSailorMeetSailor = lSailorCount
lBestSailorInBoat = lBoatCount
lLowestAdjacentFlights = lFlightCount * lSailorCount

If lFlightCount * lBoatCount Mod lSailorCount <> 0 Then
  Call MsgBox("Number of flights" & vbCrLf & "times number of boats" & vbCrLf & _
    "needs to be divisible" & vbCrLf & "by number of sailors!", vbOKOnly, "Error")
  Exit Sub
End If
If lBoatCount > lSailorCount Then
  Call MsgBox("Number of boats" & vbCrLf & "needs to be less or equal" & _
    vbCrLf & "to number of sailors!", vbOKOnly, "Error")
  Exit Sub
End If

Range("D:XFD").EntireColumn.Delete

ReDim lBestBoatInFlight(1 To lBoatCount, 1 To lFlightCount) As Long
For i = 1 To lSimulationCount
  ReDim lSailorInBoat(1 To lSailorCount, 1 To lBoatCount) As Long
  ReDim lSailorMeetSailor(1 To lSailorCount, 1 To lSailorCount) As Long
  ReDim lBoatInFlight(1 To lBoatCount, 1 To lFlightCount) As Long
  lAdjacentFlights = 0
  For j = 1 To lFlightCount
    ReDim lBoat(1 To lBoatCount) As Long
    For k = 1 To lBoatCount
      If lSailorIndex = 0 Then
        ReDim vSailor(1 To lSailorCount) As Variant
        vSailor = UniqRandInt(lSailorCount, lSailorCount)
        lSailorIndex = 1
      End If
      lBoat(k) = vSailor(lSailorIndex)
      lBoatInFlight(k, j) = vSailor(lSailorIndex)
      For m = 1 To k - 1
        lSailorMeetSailor(lBoat(k), lBoat(m)) = _
          lSailorMeetSailor(lBoat(k), lBoat(m)) + 1
        lSailorMeetSailor(lBoat(m), lBoat(k)) = _
          lSailorMeetSailor(lBoat(m), lBoat(k)) + 1
      Next m
      If j > 1 Then
        For m = 1 To lBoatCount
          If lBoatInFlight(k, j) = lBoatInFlight(m, j - 1) Then
            lAdjacentFlights = lAdjacentFlights + 1
          End If
        Next m
      End If
      lSailorInBoat(vSailor(lSailorIndex), k) = _
        lSailorInBoat(vSailor(lSailorIndex), k) + 1
      lSailorIndex = (lSailorIndex + 1) Mod (lSailorCount + 1)
    Next k
  Next j
  lMaxSailorMeetSailor = 0
  For j = 1 To lSailorCount - 1
    For m = j + 1 To lSailorCount
      If lMaxSailorMeetSailor < lSailorMeetSailor(j, m) Then
        lMaxSailorMeetSailor = lSailorMeetSailor(j, m)
      End If
    Next m
  Next j
  lMaxSailorInBoat = 0
  For j = 1 To lSailorCount
    For m = 1 To lBoatCount
      If lMaxSailorInBoat < lSailorInBoat(j, m) Then
        lMaxSailorInBoat = lSailorInBoat(j, m)
      End If
    Next m
  Next j
  If lBestSailorMeetSailor + lBestSailorInBoat + lLowestAdjacentFlights > _
    lMaxSailorMeetSailor + lMaxSailorInBoat + lAdjacentFlights Then
    For j = 1 To lBoatCount
      For m = 1 To lFlightCount
        lBestBoatInFlight(j, m) = lBoatInFlight(j, m)
      Next m
    Next j
    lBestSailorMeetSailor = lMaxSailorMeetSailor
    lBestSailorInBoat = lMaxSailorInBoat
    lLowestAdjacentFlights = lAdjacentFlights
  End If
Next i

For m = 1 To lFlightCount: ws.Cells(1, 4 + m) = "Flight " & m: Next m
For j = 1 To lBoatCount
  ws.Cells(1 + j, 4) = "Boat " & j
  For m = 1 To lFlightCount
    ws.Cells(1 + j, 4 + m) = sSailor(lBestBoatInFlight(j, m))
    #If I_Want_Colors Then
    k = (lBestBoatInFlight(j, m) Mod 56) + 1
    ws.Cells(1 + j, 4 + m).Interior.ColorIndex = k
```

```
    If xlFC(k) Then
      ws.Cells(1 + j, 4 + m).Font.ColorIndex = xlCIBlack
    Else
      ws.Cells(1 + j, 4 + m).Font.ColorIndex = xlCIWhite
    End If
    #End If
  Next m
Next j

ws.Cells(j + 1, 4) = "Maximal meet of sailor pairs"
ws.Cells(j + 1, 5) = lBestSailorMeetSailor
ws.Cells(j + 2, 4) = "Maximal repetition of boat per sailor"
ws.Cells(j + 2, 5) = lBestSailorInBoat
ws.Cells(j + 3, 4) = "Number of sailors with adjacent flights"
ws.Cells(j + 3, 5) = lLowestAdjacentFlights
Range("D:XFD").EntireColumn.AutoFit

End With
End Sub
```

Chancen beim Brettspiel Risiko

Kennen Sie Ihre Chance, einen Angriff mit 15 Armeen auf einem Ihrer Länder gegen einen benachbarten Verteidiger mit 11 Armeen zu gewinnen? Nach den alten, originalen Regeln können Sie mit 14 Ihrer 15 Armeen angreifen und hätten eine etwa 32%-ige Chance, zu gewinnen, aber nach den neuen Regeln läge die Chance bei 79%: (siehe blaue Kreise)

Alte Version: Sowohl Angreifer als auch Verteidiger verwenden bis zu 3 Würfel.

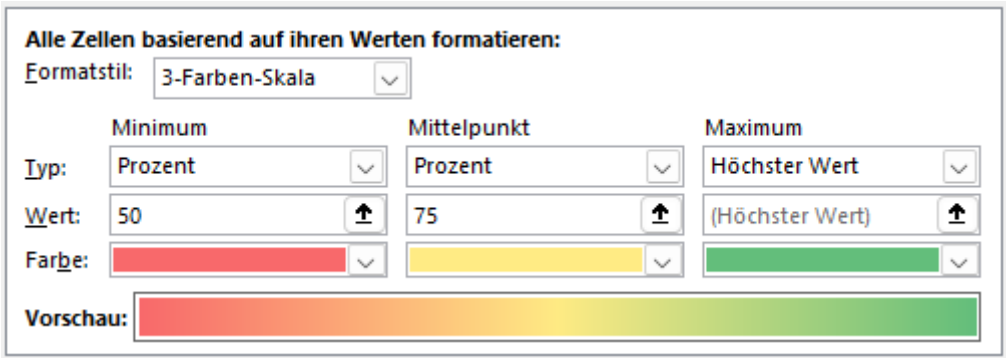
Angreifer Armeen \ Verteidiger Armeen	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	0,41	0,11	0,02	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
3	0,76	0,36	0,12	0,05	0,02	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
4	0,92	0,66	0,32	0,22	0,12	0,06	0,03	0,02	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	0,97	0,79	0,45	0,31	0,20	0,11	0,07	0,04	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
6	0,99	0,89	0,57	0,41	0,28	0,17	0,11	0,07	0,04	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00
7	1,00	0,94	0,69	0,52	0,37	0,26	0,17	0,12	0,07	0,05	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00
8	1,00	0,97	0,76	0,61	0,46	0,33	0,24	0,16	0,11	0,07	0,05	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00
9	1,00	0,98	0,82	0,69	0,54	0,41	0,31	0,22	0,15	0,11	0,07	0,05	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00
10	1,00	0,99	0,87	0,75	0,62	0,49	0,36	0,28	0,20	0,15	0,11	0,07	0,05	0,04	0,02	0,02	0,01	0,01	0,00	0,00
11	1,00	0,99	0,90	0,80	0,68	0,55	0,45	0,34	0,25	0,19	0,14	0,10	0,07	0,05	0,03	0,02	0,01	0,01	0,01	0,00
12	1,00	1,00	0,93	0,84	0,73	0,62	0,51	0,40	0,32	0,24	0,17	0,13	0,09	0,07	0,05	0,03	0,02	0,02	0,01	0,01
13	1,00	1,00	0,95	0,88	0,78	0,68	0,57	0,47	0,36	0,28	0,22	0,16	0,12	0,09	0,06	0,04	0,03	0,02	0,01	0,01
14	1,00	1,00	0,96	0,90	0,83	0,73	0,62	0,52	0,43	0,34	0,26	0,20	0,16	0,11	0,08	0,06	0,04	0,03	0,02	0,01
15	1,00	1,00	0,97	0,93	0,86	0,77	0,68	0,58	0,48	0,39	0,32	0,25	0,19	0,15	0,11	0,08	0,05	0,04	0,03	0,02
16	1,00	1,00	0,98	0,94	0,89	0,81	0,72	0,63	0,54	0,44	0,37	0,29	0,24	0,18	0,13	0,10	0,07	0,06	0,04	0,03
17	1,00	1,00	0,98	0,96	0,90	0,85	0,78	0,68	0,58	0,50	0,42	0,34	0,27	0,22	0,17	0,13	0,10	0,07	0,05	0,04
18	1,00	1,00	0,99	0,97	0,93	0,88	0,80	0,73	0,64	0,55	0,47	0,39	0,31	0,26	0,21	0,15	0,12	0,09	0,07	0,05
19	1,00	1,00	0,99	0,98	0,94	0,89	0,83	0,76	0,68	0,60	0,52	0,44	0,35	0,30	0,23	0,19	0,15	0,12	0,08	0,06
20	1,00	1,00	1,00	0,98	0,96	0,91	0,86	0,80	0,71	0,64	0,56	0,50	0,41	0,34	0,28	0,22	0,17	0,14	0,11	0,08

Neue Version: Angreifer verwendet bis zu 3 Würfel, Verteidiger nur bis zu 2.

Angreifer Armeen \ Verteidiger Armeen	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	0,41	0,10	0,03	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
3	0,75	0,36	0,21	0,09	0,05	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
4	0,92	0,67	0,47	0,32	0,21	0,13	0,08	0,05	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	0,97	0,78	0,65	0,46	0,36	0,26	0,18	0,13	0,09	0,06	0,04	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00
6	0,99	0,88	0,78	0,64	0,51	0,39	0,29	0,23	0,17	0,11	0,08	0,06	0,04	0,03	0,02	0,01	0,01	0,01	0,00	0,00
7	1,00	0,94	0,85	0,74	0,64	0,52	0,43	0,32	0,26	0,18	0,15	0,11	0,08	0,06	0,04	0,03	0,02	0,02	0,01	0,01
8	1,00	0,97	0,91	0,83	0,74	0,64	0,54	0,45	0,35	0,28	0,22	0,17	0,13	0,10	0,07	0,05	0,04	0,03	0,02	0,02
9	1,00	0,98	0,95	0,89	0,82	0,73	0,65	0,55	0,45	0,38	0,31	0,24	0,19	0,15	0,12	0,09	0,07	0,06	0,04	0,03
10	1,00	0,99	0,97	0,93	0,87	0,81	0,73	0,65	0,55	0,48	0,40	0,33	0,27	0,22	0,17	0,14	0,10	0,08	0,06	0,05
11	1,00	0,99	0,98	0,95	0,92	0,86	0,80	0,73	0,64	0,56	0,50	0,42	0,35	0,29	0,24	0,19	0,15	0,11	0,10	0,08
12	1,00	1,00	0,99	0,97	0,94	0,91	0,85	0,80	0,73	0,64	0,58	0,50	0,43	0,38	0,31	0,25	0,21	0,17	0,14	0,11
13	1,00	1,00	0,99	0,98	0,96	0,93	0,90	0,85	0,79	0,72	0,65	0,59	0,51	0,45	0,38	0,33	0,28	0,23	0,19	0,15
14	1,00	1,00	1,00	0,99	0,98	0,95	0,92	0,89	0,84	0,79	0,72	0,66	0,59	0,53	0,47	0,40	0,34	0,29	0,25	0,21
15	1,00	1,00	1,00	0,99	0,99	0,97	0,95	0,91	0,88	0,83	0,79	0,72	0,66	0,60	0,54	0,47	0,41	0,37	0,31	0,25
16	1,00	1,00	1,00	1,00	0,99	0,98	0,97	0,94	0,91	0,88	0,83	0,78	0,72	0,67	0,60	0,55	0,48	0,43	0,37	0,32
17	1,00	1,00	1,00	1,00	0,99	0,99	0,98	0,96	0,93	0,90	0,87	0,83	0,78	0,73	0,67	0,62	0,56	0,49	0,44	0,39
18	1,00	1,00	1,00	1,00	1,00	0,99	0,98	0,97	0,95	0,93	0,90	0,87	0,82	0,78	0,73	0,67	0,61	0,57	0,50	0,45
19	1,00	1,00	1,00	1,00	1,00	0,99	0,99	0,98	0,97	0,95	0,92	0,90	0,87	0,82	0,78	0,73	0,68	0,62	0,57	0,51
20	1,00	1,00	1,00	1,00	1,00	1,00	0,99	0,99	0,98	0,97	0,94	0,92	0,89	0,86	0,82	0,78	0,73	0,68	0,62	0,57

Starte Simulationen

Ein bedingtes Format färbt die Zellen rot für Chancen in Höhe von 50% oder weniger, ein gelber Hintergrund zeigt Chancen zwischen 50% und 75% an, und grüne Zellfarben zeigen Chancen von 75% oder höher an:



Theoretisch müssten beide Tabellen für die ersten 2 Spalten identisch sein. Kleine Differenzen werden durch die “unvollständige” Zufälligkeit der endlichen Monte Carlo Simulation mit 10.000 Läufen verursacht.

Game of Risk Programmcode

```
Const GCMonteCarloRuns = 10000

Sub Schedule()
'Calculate chances for an attacker at the game of risk for both the original
'version (both attacker and defender roll up to 3 dice) and the new version
'(attacker rolls up to 3 dice, defender only up to 2).
'Calls parametrized sub Calculate_Chances twice.
'(C) (P) by Bernd Plumhoff 30-Sep-2012 PB V0.1
Dim ws As Worksheet
Dim cPerf As clsPerf 'See: https://jkp-ads.com/Articles/performanceclass.asp

'Include SystemState class from https://sulprobil.com/html/systemstate.html
Dim state As SystemState
If gbDebug Then
  Set cPerf = New clsPerf
  cPerf.SetRoutine "Schedule"
End If
Application.StatusBar = False
Set state = New SystemState

'Preparation
Set ws = Sheets("Chances")
ws.Cells.ClearContents

Call Calculate_Chances("Old Version: Both Attacker and defender roll up to" & _
  " 3 dice.", 1, 3)
Call Calculate_Chances("New Version: Attacker rolls up to 3 dice, defender" & _
  " only up to 2.", 23, 2)

Call ReportPerformance

End Sub

Sub Calculate_Chances(sTitle As String, _
    lOutputRow As Long, _
    lMaxDefenderArmies As Long)
'Calculate chances for an attacker at the game of risk.
'This sub calculates the chances for a matrix of 2 to 20 attacking armies
'against 1 to 20 defending armies.
'Reverse(moc.liborplus.www) V0.1 30-Sep-2012
Dim i As Long
Dim j As Long
Dim k As Long
Dim m As Long
Dim lAttackerDice As Long
Dim lAttackerThrow As Long
Dim lAttackerResult(1 To 3) As Long
Dim lAttackerWins As Long
Dim lDefenderDice As Long
Dim lDefenderThrow As Long
Dim lDefenderResult(1 To 3) As Long
Dim ws As Worksheet
Dim cPerf As clsPerf 'See: https://jkp-ads.com/Articles/performanceclass.asp

'Include SystemState class from https://sulprobil.com/html/systemstate.html
Dim state As SystemState
If gbDebug Then
  Set cPerf = New clsPerf
```

```
  cPerf.SetRoutine "Calculate_Chances"
End If

Application.StatusBar = False
Set state = New SystemState

With Application.WorksheetFunction
'Preparation
Set ws = Sheets("Chances")
ws.Cells(lOutputRow, 1) = sTitle
ws.Cells(lOutputRow + 1, 1) = "Attacker armies \ Defender armies"
For i = 2 To 20
  Application.StatusBar = "Calculating " & i & " attackers for " & sTitle
  For j = 1 To 20
    ws.Cells(i + lOutputRow, 1) = i
    ws.Cells(1 + lOutputRow, j + 1) = j
    lAttackerWins = 0
    For k = 1 To GCMonteCarloRuns
      lAttackerDice = i - 1 'One army needs to occupy the land and
                            'cannot be used to attack
      lDefenderDice = j
      Do While lAttackerDice > 0 And lDefenderDice > 0
        lAttackerThrow = lAttackerDice
        If lAttackerThrow > 3 Then lAttackerThrow = 3
        lDefenderThrow = lDefenderDice
        If lDefenderThrow > lMaxDefenderArmies Then
          lDefenderThrow = lMaxDefenderArmies
        End If
        'Roll the dice
        For m = 2 To 3
          lAttackerResult(m) = 0
          lDefenderResult(m) = 0
        Next m
        For m = 1 To lAttackerThrow
          lAttackerResult(m) = Int(1 + Rnd * 6)
        Next m
        For m = 1 To lDefenderThrow
          lDefenderResult(m) = Int(1 + Rnd * 6)
        Next m
        'Sort results
        If lAttackerResult(1) < lAttackerResult(2) Then
          If lAttackerResult(1) < lAttackerResult(3) Then
              If lAttackerResult(2) < lAttackerResult(3) Then
                  '3-2-1
                  m = lAttackerResult(1)
                  lAttackerResult(1) = lAttackerResult(3)
                  lAttackerResult(3) = m
              Else
                  '2-3-1
                   m = lAttackerResult(1)
                  lAttackerResult(1) = lAttackerResult(2)
                  lAttackerResult(2) = lAttackerResult(3)
                  lAttackerResult(3) = m
              End If
          Else
              '2-1-3
              m = lAttackerResult(1)
              lAttackerResult(1) = lAttackerResult(2)
              lAttackerResult(2) = m
          End If
        Else
          If lAttackerResult(1) < lAttackerResult(3) Then
              If lAttackerResult(2) < lAttackerResult(3) Then
                  '3-1-2
                  m = lAttackerResult(1)
                  lAttackerResult(1) = lAttackerResult(3)
                  lAttackerResult(3) = lAttackerResult(2)
                  lAttackerResult(2) = m
              End If
          Else
              If lAttackerResult(2) < lAttackerResult(3) Then
                  '1-3-2
                  m = lAttackerResult(2)
                  lAttackerResult(2) = lAttackerResult(3)
                  lAttackerResult(3) = m
              End If
          End If
        End If
        If lDefenderResult(1) < lDefenderResult(2) Then
          If lDefenderResult(1) < lDefenderResult(3) Then
            If lDefenderResult(2) < lDefenderResult(3) Then
              '3-2-1
              m = lDefenderResult(1)
              lDefenderResult(1) = lDefenderResult(3)
              lDefenderResult(3) = m
            Else
              '2-3-1
               m = lDefenderResult(1)
              lDefenderResult(1) = lDefenderResult(2)
              lDefenderResult(2) = lDefenderResult(3)
```

```
            lDefenderResult(3) = m
          End If
        Else
          '2-1-3
          m = lDefenderResult(1)
          lDefenderResult(1) = lDefenderResult(2)
          lDefenderResult(2) = m
        End If
      Else
        If lDefenderResult(1) < lDefenderResult(3) Then
          If lDefenderResult(2) < lDefenderResult(3) Then
            '3-1-2
            m = lDefenderResult(1)
            lDefenderResult(1) = lDefenderResult(3)
            lDefenderResult(3) = lDefenderResult(2)
            lDefenderResult(2) = m
          End If
        Else
          If lDefenderResult(2) < lDefenderResult(3) Then
            '1-3-2
            m = lDefenderResult(2)
            lDefenderResult(2) = lDefenderResult(3)
            lDefenderResult(3) = m
          End If
        End If
      End If
      'Analyze result and reduce armies
      For m = 1 To 3
        If lAttackerResult(m) > 0 And lDefenderResult(m) > 0 Then
          If lAttackerResult(m) > lDefenderResult(m) Then
            lDefenderDice = lDefenderDice - 1
          Else
            lAttackerDice = lAttackerDice - 1
          End If
        Else
          Exit For
        End If
      Next m
    Loop
    If lAttackerDice > 0 Then
      lAttackerWins = lAttackerWins + 1
    End If
  Next k
  ws.Cells(i + lOutputRow, j + 1) = lAttackerWins / GCMonteCarloRuns
 Next j
Next i
End With
End Sub
```

Krabat – Wie alt können die Lehrlinge werden?

Krabat ist ein Jugendbuch von Otfried Preußler. Ich fand die Geschichte faszinierend, aber etwas unlogisch: 12 Lehrlinge arbeiten in der Mühle. Jedes Jahr stirbt einer, und jedes Jahr wird ein neuer Lehrling im Alter von 14 Jahren aufgenommen. Alle altern innerhalb eines Jahres um drei Jahre.

Nach 30 Jahren kann es einen Lehrling geben, der 101 Jahre alt ist:

E9 =WENN(SPALTE()-3=$C9;$E$1;E8+3)

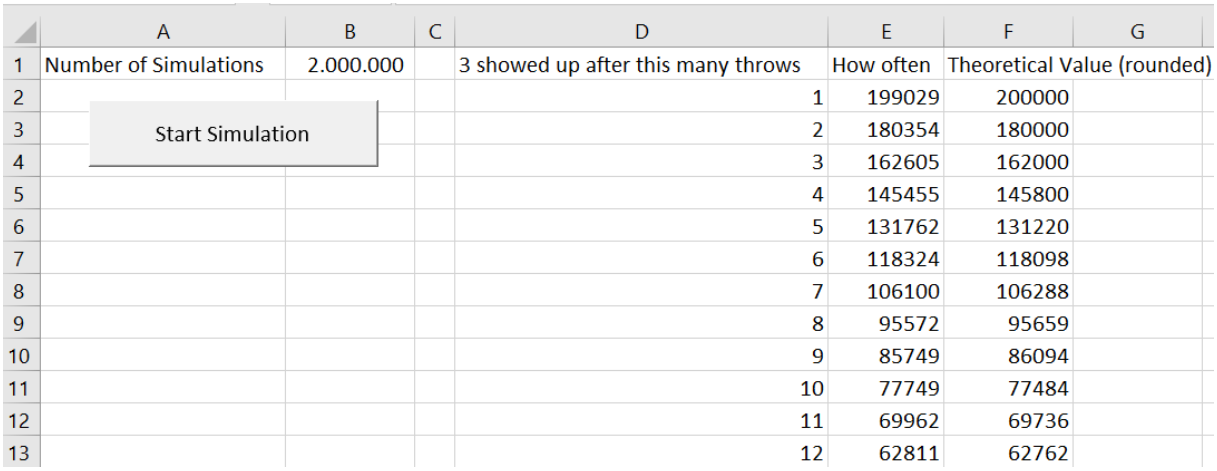
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	**Anfangsalter**				14										
2	**Maximales Alter**				101										
3	**Maximales Alter in 20 Jahren**				71										
4															
5	**Mühlen-**	**Echtes**	**Wer**	**Lehrlinge und ihr Alter**											
6	**jahr**	**Jahr**	**stirbt?**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
7	1	1		14	14	14	14	14	14	14	14	14	14	14	14
8	2	4	11	17	17	17	17	17	17	17	17	17	17	14	17
9	3	7	5	20	20	20	20	14	20	20	20	20	20	17	20
10	4	10	8	23	23	23	23	17	23	23	14	23	23	20	23
11	5	13	5	26	26	26	26	14	26	26	17	26	26	23	26
12	6	16	2	29	14	29	29	17	29	29	20	29	29	26	29
13	7	19	2	32	14	32	32	20	32	32	23	32	32	29	32
14	8	22	5	35	17	35	35	14	35	35	26	35	35	32	35
15	9	25	3	38	20	14	38	17	38	38	29	38	38	35	38
16	10	28	9	41	23	17	41	20	41	41	32	14	41	38	41
17	11	31	8	44	26	20	44	23	44	44	14	17	44	41	44
18	12	34	4	47	29	23	14	26	47	47	17	20	47	44	47
19	13	37	12	50	32	26	17	29	50	50	20	23	50	47	14
20	14	40	8	53	35	29	20	32	53	53	14	26	53	50	17
21	15	43	9	56	38	32	23	35	56	56	17	14	56	53	20
22	16	46	9	59	41	35	26	38	59	59	20	14	59	56	23
23	17	49	3	62	44	14	29	41	62	62	23	17	62	59	26
24	18	52	5	65	47	17	32	14	65	65	26	20	65	62	29
25	19	55	3	68	50	14	35	17	68	68	29	23	68	65	32
26	20	58	1	14	53	17	38	20	71	71	32	26	71	68	35
27	21	61	5	17	56	20	41	14	74	74	35	29	74	71	38
28	22	64	7	20	59	23	44	17	77	14	38	32	77	74	41
29	23	67	5	23	62	26	47	14	80	17	41	35	80	77	44
30	24	70	4	26	65	29	14	17	83	20	44	38	83	80	47
31	25	73	2	29	14	32	17	20	86	23	47	41	86	83	50
32	26	76	1	14	17	35	20	23	89	26	50	44	89	86	53
33	27	79	1	14	20	38	23	26	92	29	53	47	92	89	56
34	28	82	8	17	23	41	26	29	95	32	14	50	95	92	59
35	29	85	1	14	26	44	29	32	98	35	17	53	98	95	62
36	30	88	4	17	29	47	14	35	101	38	20	56	101	98	65

Tabellenblattformeln		
Bereich	Formel	
E2	**E2**	=MAX(D8:O36)
E3	**E3**	=MAX(D8:O26)
B7:B36	**B7**	=A7*3-2
C8:C36	**C8**	=ABRUNDEN(ZUFALLSZAHL()*12;0)+1
D8:O36	**D8**	=WENN(SPALTE()-3=$C8;$E$1;D7+3)

Eine simple Monte Carlo Simulation

Mit Excel/VBA kann man rasch eine Monte Carlo Simulation erzeugen, aber man sollte einige mögliche Fallstricke umgehen:

- Verwende die Klasse SystemState, um das Programm durch Ausschalten von ScreenUpdating und durch Setzen der Calculation auf xlCalculationManual zu beschleunigen.
- Halte den Anwender über den Prozess der Simulation auf dem Laufenden.
- Behandle unbekanntes Dimensionswachstum während des Programms effizient.
- Sei Dir bewusst, dass Excel im Allgemeinen nicht das beste (nicht das schnellste) Simulationstool ist.



	A	B	C	D	E	F	G
1	Number of Simulations	2.000.000		3 showed up after this many throws	How often	Theoretical Value (rounded)	
2				1	199029	200000	
3				2	180354	180000	
4				3	162605	162000	
5				4	145455	145800	
6				5	131762	131220	
7				6	118324	118098	
8				7	106100	106288	
9				8	95572	95659	
10				9	85749	86094	
11				10	77749	77484	
12				11	69962	69736	
13				12	62811	62762	

Literatur

Ab Excel 2010 scheint es ok zu sein, Excel für Monte Carlo Simulationen zu verwenden, wenn es nicht zu langsam ist:

Alexei Botchkarev, Assessing Excel VBA Suitability for Monte Carlo Simulation, https://arxiv.org/ftp/arxiv/papers/1503/1503.08376.pdf

sbMontaCarloSimulation Programmcode

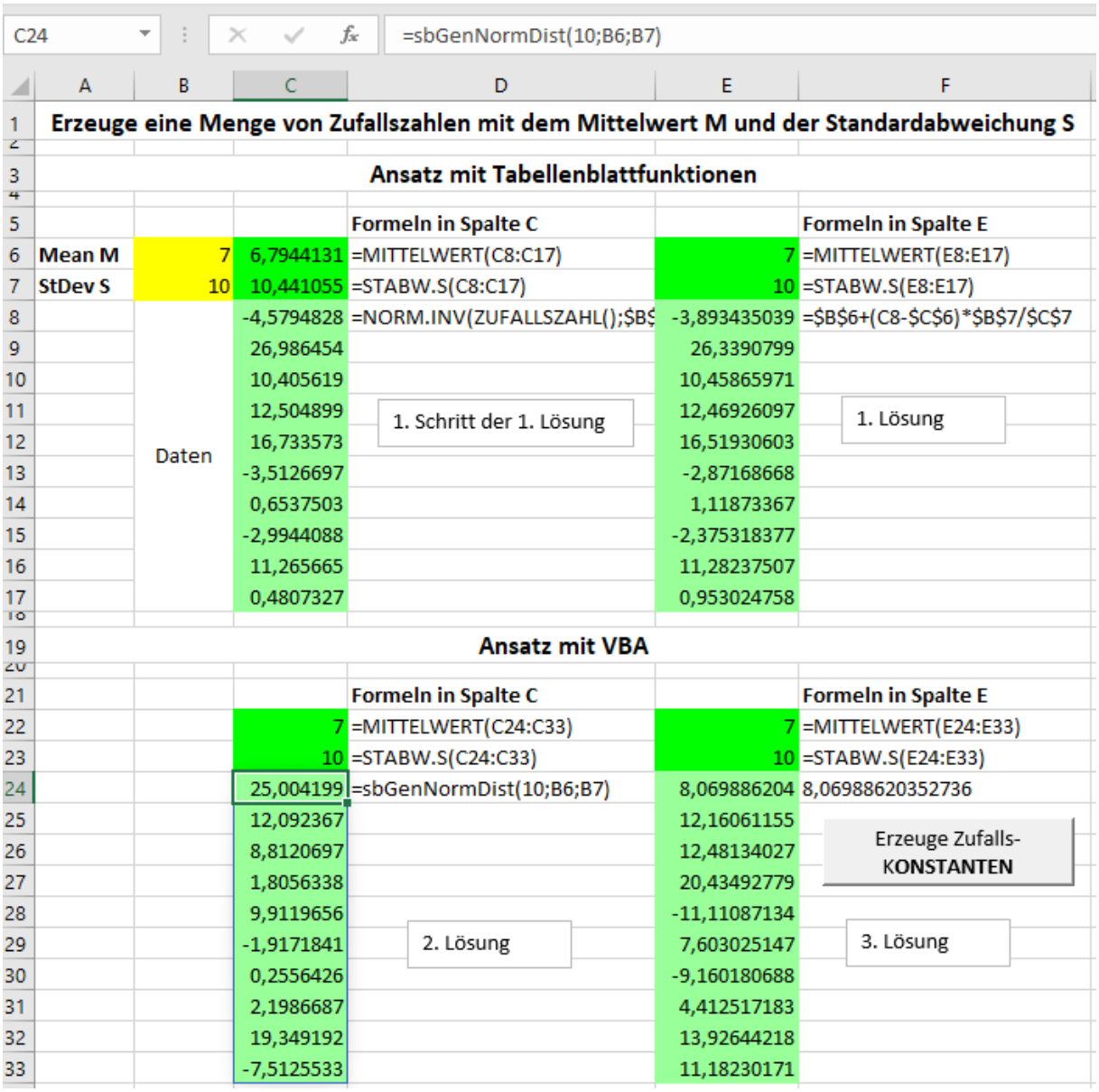
```
Sub Simulate()
'Creates a simple Monte Carlo simulation by counting how long
'it takes to throw a 3 with a die with 10 surfaces (likelihood
'for each to show is 1/10).
'(C) (P) by Bernd Plumhoff  23-Nov-2022 PB V0.2
Dim i                   As Long
Dim lSimulations        As Long
Dim lTries              As Long

Dim state               As SystemState

With Application.WorksheetFunction
Set state = New SystemState
Randomize
lSimulations = Range("Simulations")
ReDim lResult(1 To 1) As Long 'Error Handler will increase as needed
On Error GoTo ErrHdl
For i = 1 To lSimulations
  If i Mod 10000 = 1 Then Application.StatusBar = "Simulation " & _
    Format(i, "#,##0") 'Inform the user that program is still alive
  lTries = 0
  Do
    lTries = lTries + 1
  Loop Until .RandBetween(1, 10) = 3 'This is the simulation
  lResult(lTries) = lResult(lTries) + 1
Next i
On Error GoTo 0
Range("D:F").ClearContents
Range("D1:F1").FormulaArray = Array("3 showed up after this many throws", _
  "How often", "Theoretical Value (rounded)")
For i = 1 To UBound(lResult)
  Cells(i + 1, 4) = i
  Cells(i + 1, 5) = lResult(i)
  lTries = .Round(lSimulations * 0.1, 0)
  Cells(i + 1, 6) = lTries
  lSimulations = lSimulations - lTries
Next i
End With
Exit Sub

ErrHdl:
If Err.Number = 9 Then
  'Here we normally get if we breach Ubound(lResult)
  If lTries > UBound(lResult) Then
    'So we need to increase dimension
    ReDim Preserve lResult(1 To lTries)
    Resume 'Back to statement which caused error
  End If
End If
'Other error - terminate
On Error GoTo 0
Resume
End Sub
```

Gleitkomma-Zufallszahlen

Eine ideale Normalverteilung – *sbGenNormDist*

Um eine Standardnormalverteilung zu generieren, verwenden Sie normalerweise *=NORM.S.INV(ZUFALLSZAHL())*. Wenn Sie also eine Standardnormalverteilung mit dem Mittelwert 7 und der Standardabweichung 10 benötigen, dann rufen Sie *=NORM.INV(ZUFALLSZAHL();7;10)* auf.

Aber: Ihre Normalverteilung wird nie einen Mittelwert von genau 7 (und nie eine Standardabweichung von genau 10) zeigen, wenn die Anzahl der Zufallszahlen nicht gegen unendlich geht. Falls Sie einen Mittelwert von genau 7 und eine Standardabweichung von genau 10 erreichen wollen, dann müssen Sie die erzeugte Menge von Zufallszahlen zum Mittelwert verschieben und sie dann zur gewünschten Standardabweichung stauchen oder dehnen. Sie können dies auf mindestens drei verschiedene Art und Weisen erreichen:



sbGenNormDist Programmcode

```
Function sbGenNormDist(lCount As Long, _
    dMean As Double, _
    dStDev As Double) As Variant
'Generates lCount normally distributed random values
'with mean dMean and standard deviation dStDev.
'(C) (P) by Bernd Plumhoff 30-May-2024 V0.3
Dim i As Long
Dim dSampleMean As Double, dSampleStDev As Double

If lCount < 2 Then
  sbGenNormDist = CVErr(xlErrValue)
  Exit Function
End If
ReDim vR(1 To lCount) As Variant
With Application
For i = 1 To lCount
  vR(i) = .Norm_Inv(Rnd(), dMean, dStDev)
Next i
dSampleMean = .Average(vR)
dSampleStDev = .StDev_S(vR)
For i = 1 To lCount
  vR(i) = dMean + (vR(i) - dSampleMean) * dStDev / dSampleStDev
Next i
sbGenNormDist = .Transpose(vR)
End With
End Function
```

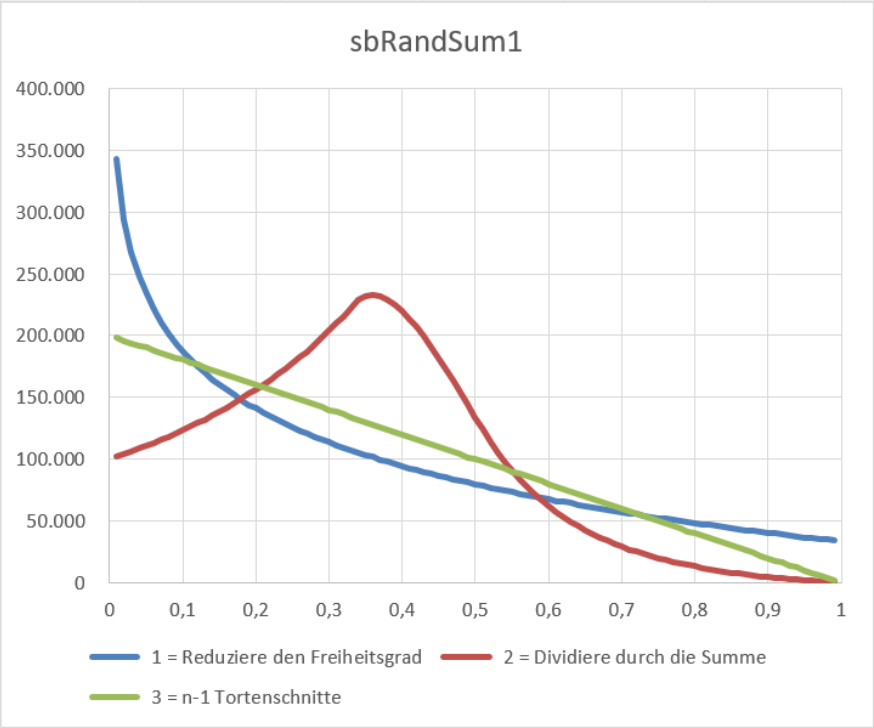
Zufallszahlen mit der Summe 1 – *sbRandSum1*

Wir erzeugen *n* Zufallszahlen mit einer Bedingung: Die Summe aller erzeugten Zahlen soll 1 sein. Dies kann man mit vielen verschiedenen Ansätzen erreichen.

Drei mögliche Ansätze sind:

- Reduziere den Freiheitsgrad sukzessive: Erzeuge die erste Zufallszahl, danach die zweite im Bereich [0,1-Erste_Zahl), dann die dritte [0,1-Erste_Zahl-Zweite_Zahl), …, die letzte Zahl muss schließlich gleich 1-Summe_aller_anderen_Zahlen sein.
- Erzeuge n Zufallszahlen und dividiere sie durch ihre Summe.
- Simuliere die Teilung einer Torte: wo immer man schneidet, man kann nicht mehr und nicht weniger als die ganze Torte verteilen.

Die resultierenden Verteilungen sehen so aus:



Sie können leicht erkennen, dass der weitverbreitete Ansatz mit n Zufallszahlen dividiert durch ihre Summe eine schlechte Wahl ist: Sie erhalten meist Zahlen zwischen 0,2 und 0,5 (siehe die rote Kurve).

Bemerkung: Ein genereller Ansatz wäre die Dirichlet Verteilung. Für eine Impementierung mit Python siehe numpy - für unsere obige Ausgabe müsste der Parameter size auf 1 gesetzt werden: https://numpy.org/doc/stable/reference/random/generated/numpy.random.dirichlet.html?highlight=dirichlet#numpy.random.dirichlet

sbRandSum1 Programmcode

```
Function sbRandSum1(ByVal lDist As Long, Optional ByVal lCount As Long, _
    Optional bVolatile As Boolean = False) As Variant
'sbRandSum1 produces lCount (or the number of selected cells if
'called from a worksheet range) random numbers which sum up to 1.
'Possible values of lDist to specify desired distribution:
'        1 = reduce degree of freedom linearly
'        2 = divide lCount random numbers by their sum
'        3 = lCount-1 random cuts of (0,1)-interval
'If TypeName(Application.Caller) <> "Range" Then lCount has to be set.
'It specifies the count of summands which have to have the sum of 1.
'(C) (P) by Bernd Plumhoff 02-Aug-2020 PB V0.4
Static bRandomized As Boolean
Dim bRowWise As Boolean, vA As Variant, vT  As Variant
Dim i As Long, j As Long, dSum As Double

If bVolatile Then Application.Volatile
If Not bRandomized Then Randomize: bRandomized = True
If TypeName(Application.Caller) <> "Range" Then
  If lCount < 1 Then
    sbRandSum1 = CVErr(xlErrRef)
    Exit Function
  End If
  bRowWise = False
Else
  With Application.Caller
    lCount = .Rows.Count
    bRowWise = True
    If lCount < .Columns.Count Then
      lCount = .Columns.Count
      bRowWise = False
    End If
    If lCount = 1 Then
      sbRandSum1 = 1
      Exit Function
    End If
  End With
End If
ReDim vA(1 To lCount) As Variant
Select Case lDist
  Case 1
    ReDim nRand(1 To lCount) As Long
    For i = 1 To lCount
      nRand(i) = i
    Next i
    For i = 1 To lCount - 1
      j = Int(Rnd * (lCount - i + 1)) + i
      vA(nRand(j)) = Rnd * (1# - dSum)
      dSum = dSum + vA(nRand(j))
      nRand(j) = nRand(i)
    Next i
    vA(nRand(lCount)) = 1# - dSum
  Case 2
    For i = 1 To lCount
      vA(i) = Rnd
      dSum = dSum + vA(i)
    Next i
    For i = 1 To lCount
      vA(i) = vA(i) / dSum
    Next i
  Case 3
    For i = 1 To lCount - 1
      vA(i) = Rnd
      j = i - 1
      Do While j > 0
        If vA(j) > vA(j + 1) Then
          vT = vA(j + 1)
          vA(j + 1) = vA(j)
          vA(j) = vT
        End If
        j = j - 1
      Loop
    Next i
    vA(lCount) = 1# - vA(lCount - 1)
    i = lCount - 1
    Do While i > 1
      vA(i) = vA(i) - vA(i - 1)
      i = i - 1
    Loop
  Case Else
    sbRandSum1 = CVErr(xlErrValue)
    Exit Function
End Select
If bRowWise Then vA = Application.WorksheetFunction.Transpose(vA)
sbRandSum1 = vA
End Function
```

Zufallsportfolio mit Gesamtsumme und Asset-Schranken – *sbAllocate*

Falls Sie ein Zufallsportfolio mit einer gegebenen Summe und Ober- und Untergrenzen für jedes Asset erzeugen müssen, können Sie die Funktion PF_Allocate() verwenden:

	A	B	C	D
1			**Budget:**	
2			**11.000,00**	
3		Unter-grenzen	Ergebnis Vektor	Ober-grenzen
4	Asset 1	700,00	874,62	1.300,00
5	Asset 2	1.300,00	1.367,48	1.400,00
6	Asset 3	-1.000,00	958,90	1.000,00
7	Asset 4	-2.000,00	-1.231,11	-1.000,00
8	Asset 5	0,00	1.214,01	5.000,00
9	Asset 6	-300,00	-147,25	0,00
10	Asset 7	1.700,00	7.963,35	9.000,00
11	**Summe**	**400,00**	**11.000,00**	**16.700,00**

Tabellenblattformeln		
Bereich	Formel	
C4:C10	**C4**	=MTRANS(sbAllocate(C2;B4:B10;D4:D10))
B11:D11	**B11**	=SUMME(B4:B10)

sbAllocate Programmcode

Hinweis: Dieses Programm verwendet UniqRandInt (siehe Kapitel „Natürliche Zufallszahlen – *UniqRandInt*“, Seite 239).

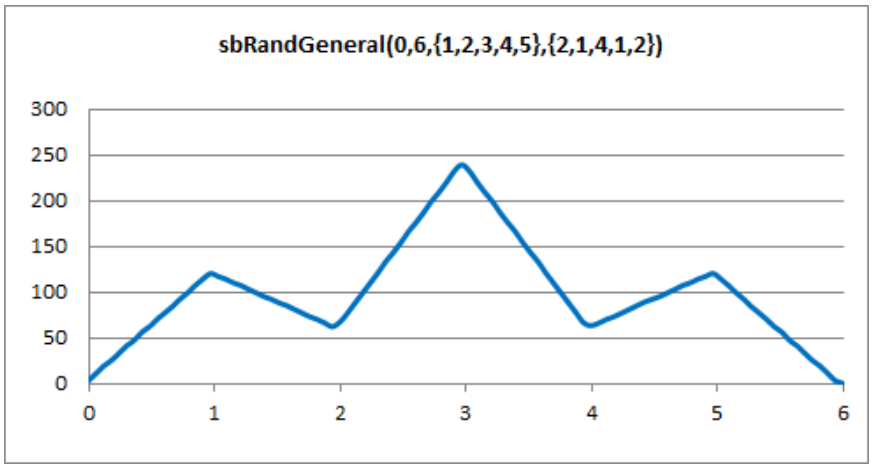
```
Function sbAllocate(db As Double, _
    vlb As Variant, _
    vub As Variant) As Double()
'Generate a portfolio of assets x1..xN
'x1..xN being random numbers (double) with:
'x1+x2+..xN = db 'budget
'xi >= vlb(i)    'lower bound vector
'xi <= vub(i)    'upper bound vector
'V0.2 PB 12-Jan-2025 (C) (P) by Bernd Plumhoff
Dim i As Variant, vR As Variant
Dim n As Long
Dim dcumx As Double, dcumlb As Double, dcumub As Double
Dim dxlb As Double, dxub As Double

dcumlb = Application.Sum(vlb)
dcumub = Application.Sum(vub)
If dcumlb > db Or dcumub < db Then
  sbAllocate = CVErr(xlErrValue)
  Exit Function
End If
n = vlb.Count
ReDim dR(1 To n) As Double
Set vR = vlb
dcumx = 0#
For Each i In UniqRandInt(n, n)
  If vlb(i) > vub(i) Then
    sbAllocate = CVErr(xlErrValue)
    Exit Function
  End If
  dcumlb = dcumlb - vlb(i)
  dcumub = dcumub - vub(i)
  dxlb = db - dcumx - dcumub
  If dxlb < vlb(i) Then dxlb = vlb(i) 'dxlb = Min(..)
  dxub = db - dcumx - dcumlb
  If dxub > vub(i) Then dxub = vub(i) 'dxub = Max(..)
  dR(i) = dxlb + Rnd() * (dxub - dxlb)
  dcumx = dcumx + dR(i)
Next i
sbAllocate = dR
End Function
```

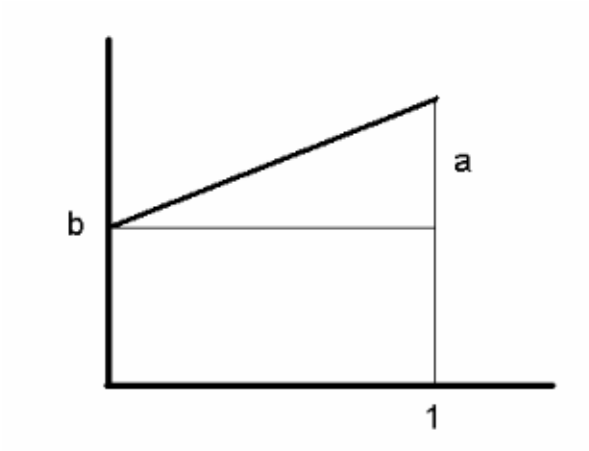
Verteilungen von Gleitkomma-Zufallszahlen

sbRandGeneral

Wenn Sie eine schrittweise lineare Verteilung von Zufallszahlen benötigen, dann empfehle ich meine benutzerdefinierte Funktion *sbRandGeneral*.

Bemerkung: Man kann jede beliebige Verteilung mit einer festgelegten Mindestgenauigkeit durch eine schrittweise lineare Verteilung wie hier angeboten approximieren.





Given: Values a and b.

Needed: Inverse function of area below $f(x) = ax + b$.

Function F for area below f(x): $F(x) = \frac{a}{2}x^2 + bx$

$\Leftrightarrow$ [a≠0]

$$\frac{2}{a}F(x) = x^2 + \frac{2b}{a}x + \frac{b^2}{a^2} - \frac{b^2}{a^2} = \left(x + \frac{b}{a}\right)^2 - \frac{b^2}{a^2}$$

$\Leftrightarrow$

$$x = -\frac{b}{a} \pm \sqrt{\frac{2}{a}F(x) + \frac{b^2}{a^2}}$$

$\Leftrightarrow$

$$G(x) = -\frac{b}{a} \pm \sqrt{\frac{2}{a}x + \frac{b^2}{a^2}}$$

With $b = w_i$ and $a = \frac{w_{i+1} - w_i}{x_{i+1} - x_i}$ we get

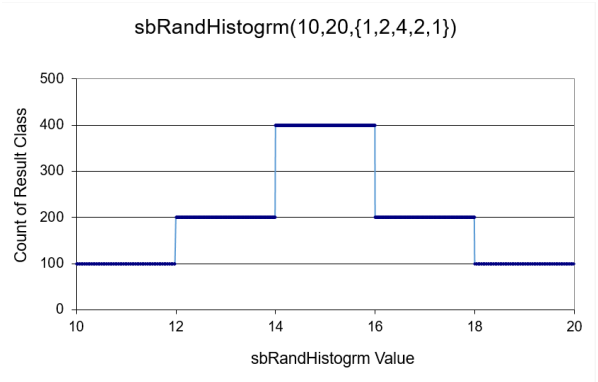
$$G(x) = -\frac{w_i(x_{i+1} - x_i)}{w_{i+1} - w_i} \pm \sqrt{\frac{2(x_{i+1} - x_i)}{w_{i+1} - w_i}x + \left(\frac{w_i(x_{i+1} - x_i)}{w_{i+1} - w_i}\right)^2}$$

sbRandGeneral Programmcode

```
Function sbRandGeneral(dMin As Double, dMax As Double, vXi As Variant, _
    vWi As Variant, Optional dRandom As Double = 1#) As Double
'Generates a random number, General distributed.
'[see Vose: Risk Analysis, 2nd ed., p. 116]
'(C) (P) by Bernd Plumhoff  26-Jul-2020 PB V1.01
'Similar to @RISK's (C) RiskGeneral function.
Static bRandomized As Boolean
Dim i As Long, lWiCount As Long, lXiCount As Long
Dim dA As Double, dRand As Double, dSgn As Double

On Error GoTo ErrorLabelIsVariant
lXiCount = vXi.Count
lWiCount = vWi.Count
ErrorLabelWasVariant:
On Error GoTo 0
If lWiCount <> lXiCount Then
    sbRandGeneral = CVErr(xlErrValue)
    Exit Function
End If
If Not bRandomized Then Randomize: bRandomized = True
ReDim dX(0 To lXiCount + 1) As Double
ReDim dW(0 To lWiCount + 1) As Double

dX(0) = dMin
dX(UBound(dX)) = dMax
dW(0) = 0#
dW(UBound(dW)) = 0#
For i = 1 To lXiCount
    dX(i) = vXi(i)
    dW(i) = vWi(i)
Next i

'Calculate area
dA = 0#
For i = 0 To UBound(dX) - 1
    If dX(i) >= dX(i + 1) Or dW(i) < 0# Then
        sbRandGeneral = CVErr(xlErrValue)
        Exit Function
    End If
    dA = dA + (dX(i + 1) - dX(i)) * (dW(i + 1) + dW(i)) / 2#
Next i

'Normalise weights to set area to 1
For i = 1 To UBound(dW) - 1
    dW(i) = dW(i) / dA
Next i

ReDim dF(0 To UBound(dX)) As Double
'Calculate border points of value ranges for
'cumulative inverse function
dF(0) = 0#
dA = 0#
For i = 0 To UBound(dX) - 1
    dA = dA + (dX(i + 1) - dX(i)) * (dW(i + 1) + dW(i)) / 2#
    dF(i + 1) = dA
Next i
If dRandom = 1# Then
    dRand = Rnd()
Else
    dRand = dRandom
End If
i = 1
Do While dF(i) <= dRand
    i = i + 1
Loop
dSgn = Sgn(dW(i) - dW(i - 1))
If dSgn = 0# Then
    sbRandGeneral = dX(i - 1) + (dRand - dF(i - 1)) / _
                    (dF(i) - dF(i - 1)) * (dX(i) - dX(i - 1))
Else
    sbRandGeneral = dX(i - 1) + _
                    dSgn * Sqr((dRand - dF(i - 1)) * _
                    2# * (dX(i) - dX(i - 1)) / (dW(i) - dW(i - 1)) + _
                    (dW(i - 1) * (dX(i) - dX(i - 1)) / _
                    (dW(i) - dW(i - 1))) ^ 2#) - _
                    dW(i - 1) * (dX(i) - dX(i - 1)) / (dW(i) - dW(i - 1))
End If
Exit Function

ErrorLabelIsVariant:
lXiCount = UBound(vXi) - 1
lWiCount = UBound(vWi) - 1
Resume ErrorLabelWasVariant

End Function
```

sbRandHistogrm

Eine stufenweise konstante Verteilung ist die Histogram Verteilung:



sbRandHistogram Programmcode

```
Function sbRandHistogrm(dmin As Double, dMax As Double, _
            vWeight As Variant, Optional dRandom = 1#) As Double
'Specifies a histogram distribution with range dmin:dmax.
'This range is divided into vWeight.count classes. Each
'class has weight vWeight(i) reflecting the probability
'of occurrence of a value within the class.
'Similar to @Risk's function RiskHistogrm.
'(C) (P) by Bernd Plumhoff 18-Oct-2020 PB V1.01

Dim i As Long, n As Long, vW As Variant
Dim dRand As Double, dR As Double, dSumWeight As Double

With Application.WorksheetFunction
vW = .Transpose(.Transpose(vWeight))
End With

n = UBound(vW)
ReDim dSumWeightI(0 To n) As Double

dSumWeight = 0#
dSumWeightI(0) = 0#
For i = 1 To n
    If vW(i) < 0# Then 'A negative weight is an error
        sbRandHistogrm = CVErr(xlErrValue)
        Exit Function
    End If
    dSumWeight = dSumWeight + vW(i) 'Calculate sum of all weights
    dSumWeightI(i) = dSumWeight     'Calculate sum of weights till i
Next i

If dSumWeight = 0# Then  'Sum of weights has to be greater than zero
    sbRandHistogrm = CVErr(xlErrValue)
    Exit Function
End If

If dRandom = 1# Then
    dRand = Rnd()
Else
    dRand = dRandom
End If
dR = dSumWeight * dRand

i = n
Do While dR < dSumWeightI(i)
    i = i - 1
Loop

sbRandHistogrm = dmin + (dMax - dmin) * _
     (CDbl(i) + (dR - dSumWeightI(i)) / vW(i + 1)) / CDbl(n)

End Function
```

Ähnliche Verteilungen erzeugen die Programme rww und redw:

rww Programmcode

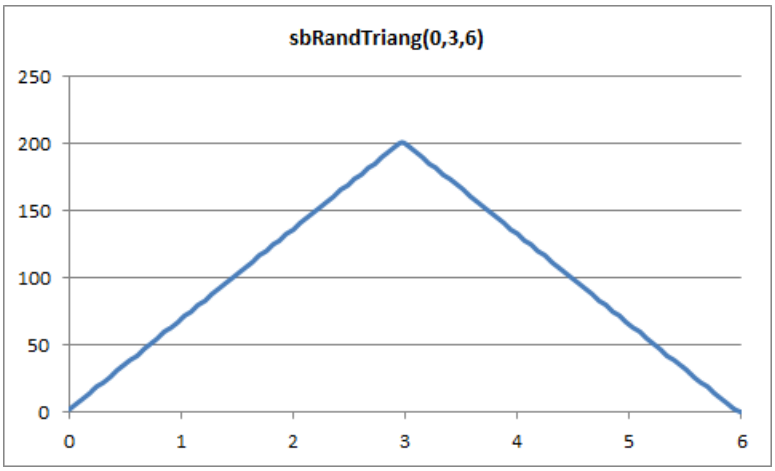
```
Function rww(ParamArray w() As Variant) As Double
'Produces random numbers with defined widths & weights
'Rww expects a vector of n random widths and weightings
'of type double and returns a random number of type double.
'This random number will lie in the given n-width-range of the
'(0,1)-interval with the given likelihood of the n weightings.
' Call Randomize before calling rww!
'(C) (P) by Bernd Plumhoff 06-Aug-2004 PB V0.50
'Examples:
'a) rww(1,0,1,1,8,0) will return a random number between 0.1 and 0.2
'b) rww(5,2,5,1) will return a random number between 0 and 0.5 twice as
'   often as a random number between 0.5 and 1.
'c) rww(1/3,0,1/3,1,1/3,0) will return a random number between
'   0.33333333333333 and 0.66666666666666.
'd) rww(5,15.4,3,7.7,2,0) would return a random value between
'   0 and 0.8, first 5 deciles with double likelihood than decile 6-8.

Dim i As Long
Dim swidths As Double
Dim sw As Double

If (UBound(w) + 1) Mod 2 <> 0 Then
  rww = -2     'No even number of arguments: Error
  Exit Function
End If

ReDim swidthsi(0 To (UBound(w) + 1) / 2 + 1) As Double
ReDim swi(0 To (UBound(w) + 1) / 2 + 1) As Double
ReDim weights(0 To (UBound(w) + 1) / 2) As Double
ReDim widths(0 To (UBound(w) + 1) / 2) As Double

swidths = 0#
sw = 0#
swi(0) = 0#
swidthsi(0) = 0#
For i = 0 To (UBound(w) - 1) / 2
  If w(2 * i) < 0# Then     'A negative width is an error
    rww = -3#
    Exit Function
  End If
  widths(i) = w(2 * i)
  swidths = swidths + widths(i)
  swidthsi(i + 1) = swidths
  If w(2 * i + 1) < 0# Then 'A negative weight is an error
    rww = -1#
    Exit Function
  End If
  weights(i) = w(2 * i + 1)
  If widths(i) > 0# Then
    sw = sw + weights(i)
  End If
  swi(i + 1) = sw
Next i
rww = sw * Rnd
'i = (UBound(w) - 1) / 2 + 1 'i already equals (UBound(w) - 1)/2 + 1, you may omit this statement.
Do While rww < swi(i)
  i = i - 1
Loop

rww = (swidthsi(i) + (rww - swi(i)) / weights(i) * widths(i)) / swidths

End Function
```

redw Programmcode

```
Function redw(ParamArray vWeights() As Variant) As Double
'Produces random numbers with equidistant weights. Redw expects a vector of n random
'weights of type double and returns a random number of type double. This random
'number will lie in the given equidistant n-split-range of the [0,1)-interval with
'the given likelihood of weightings. Call Randomize before calling redw! '(C) (P) by Bernd Plumhoff 09-Dec-
2009 PB V0.50
Examples:
'a) redw(0,1,0,0,0,0,0,0,0,0) will return a random number d, 0.1 <= d < 0.2
'b) redw(2,1) will return a random number between 0 and 0.5 twice as
'   often as a random number between 0.5 and 1.
'c) redw(0,1,0) will return a random number d, 0.333333333333333 <= d < 0.666666666666666.
'd) redw(15.4,15.4,15.4,15.4,15.4,7.7,7.7,7.7,0,0) would return a random value between
'   0 and 0.8, first 5 deciles with double likelihood than decile 6-8.

Dim i As Long
Dim dw As Double
ReDim dwi(0 To UBound(vWeights) + 2) As Double

dw = 0#
dwi(0) = 0#
For i = 0 To UBound(vWeights)
  If vWeights(i) < 0# Then  'A negative weight is an error
    redw = CVErr(xlErrValue)
    Exit Function
  End If
  dw = dw + vWeights(i)     'Calculate sum of all weights
  dwi(i + 1) = dw           'Calculate sum of weights till i
Next i

redw = dw * Rnd

'i = UBound(vWeights) + 1 'i already equals UBound(vWeights) + 1, you may omit this statement.
Do While redw < dwi(i)
  i = i - 1
Loop

redw = (CDbl(i) + (redw - dwi(i)) / vWeights(i)) / (CDbl(UBound(vWeights) + 1))

End Function
```

sbRandTriang

Die Dreiecksverteilung ist eine stetige Wahrscheinlichkeitsverteilung, deren Dichtefunktion wie ein Dreieck aussieht. Es ist eine einfache Verteilung, weil Sie lediglich ihr Minimum, ihren Median und ihr Maximum kennen müssen:



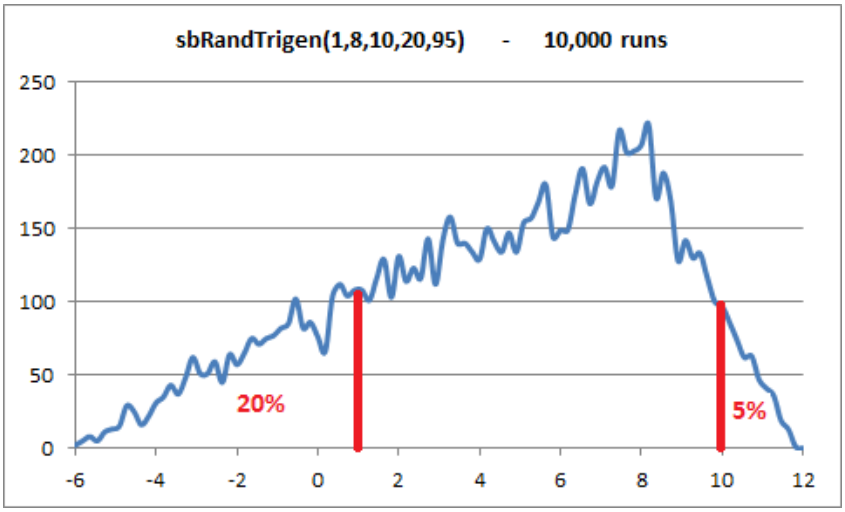
Im englischen Sprachraum wird diese Verteilung auch *Distribution of Missing Data* genannt, weil für ihre Bestimmung nur ein Minimum an Information benötigt wird. Diese Verteilung wird oft zur Simulation von Expertenwissen eingesetzt oder wenn eine genauere Datenermittlung zu schwierig oder zu teuer erscheint.

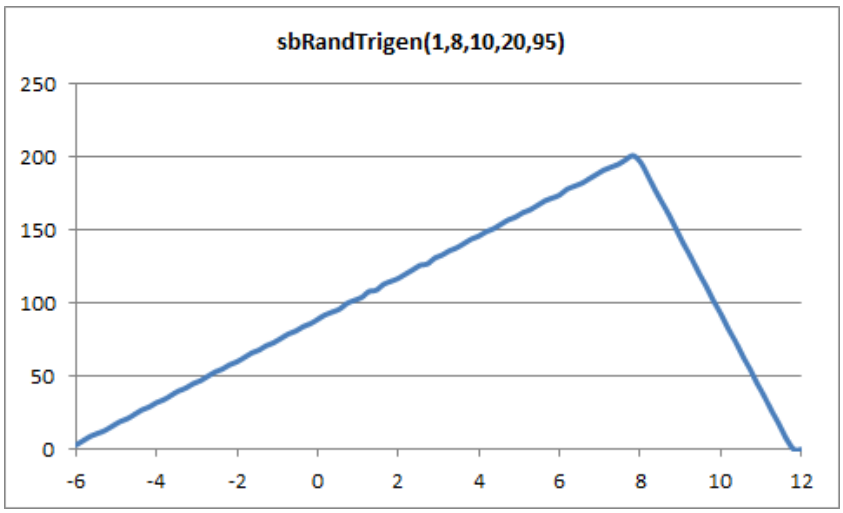
sbRandTriang Programmcode

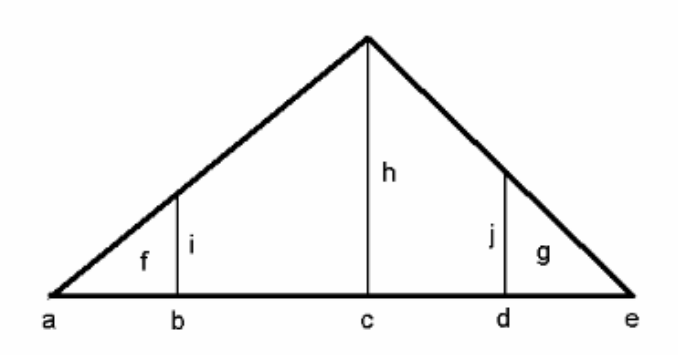
```
Function sbRandTriang(dMin As Double, dMode As Double, _
    dMax As Double, Optional dRandom = 1#) As Double
'Generates a random number, Triang distributed
'[see Vose: Risk Analysis, 2nd ed., p. 128]
'(C) (P) by Bernd Plumhoff 30-Aug-2024 PB V0.32
'Similar to @RISK's (C) RiskTriang function.
'sbRandTriang(minimum,mode,maximum) specifies a triangular
'distribution with three points – a minimum, a mode and
'a maximum. The skew of the triangular distribution is
'driven by the distance of the mode from the minimum and
'from the maximum. Reducing the distance from mode to
'minimum will increase the skew.
'Please ensure that you execute Randomize before you call
'this function for the first time.
Dim dRand As Double, dc_a As Double, db_a As Double
Dim dc_b As Double

If dMode < dMin Or dMax < dMode Then
  sbRandTriang = CVErr(xlErrValue)
  Exit Function
End If
If dMin = dMax Then
  sbRandTriang = dMin 'Triangle is just one point
  Exit Function
End If
dc_a = dMax - dMin
db_a = dMode - dMin
dc_b = dMax - dMode
If dRandom = 1# Then
  dRand = Rnd()
Else
  dRand = dRandom
End If
If dRand < db_a / dc_a Then
  sbRandTriang = dMin + Sqr(dRand * db_a * dc_a)
Else
  sbRandTriang = dMax - Sqr((1# - dRand) * dc_a * dc_b)
End If
End Function
```

sbRandTrigen

sbRandTrigen bestimmt eine Dreiecksverteilung mit drei Punkten: einen mit der höchsten Wahrscheinlichkeit und zwei weitere an den spezifizierten unteren und oberen Perzentilen. Diese Perzentilparameter haben Werte zwischen 0 und 100 und repräsentieren die kumulierten Wahrscheinlichkeiten für diese unteren und oberen Werte. Dies ist manchmal nützlich, wenn man die absolut kleinsten und größten Werte der Verteilung nicht kennt oder nicht leicht bestimmen kann.







Given: Points b, c and d; Areas f and g. Area of the triangle is 1.

To be computed: Points a and e.

Triangle area is 1: [i] $h(e-a)=2 \Leftrightarrow h=\frac{2}{e-a}$

[ii] $\frac{h}{c-a}=\frac{i}{b-a} \Leftrightarrow i=h\frac{b-a}{c-a}=\frac{2(b-a)}{(e-a)(c-a)}$

[iii] $\frac{h}{e-c}=\frac{j}{e-d} \Leftrightarrow j=h\frac{e-d}{e-c}=\frac{2(e-d)}{(e-a)(e-c)}$

[iv] $2g=j(e-d)=\frac{2(e-d)^2}{(e-a)(e-c)} \Rightarrow a=e-\frac{(e-d)^2}{g(e-c)}$

[v] $2f=i(b-a)=\frac{2(b-a)^2}{(e-a)(c-a)}$

Substituting a in [v] and putting everything on one side gives:

$$\left\{b-e+\frac{(e-d)^2}{g(e-c)}\right\}^2-f\left\{e-e+\frac{(e-d)^2}{g(e-c)}\right\}\left\{c-e+\frac{(e-d)^2}{g(e-c)}\right\}=0$$

$\Leftrightarrow$

$$\frac{\left((b-e)g(e-c)+(e-d)^2\right)^2}{\left(g(e-c)\right)^2}-f\frac{(e-d)^2}{g(e-c)}\frac{(c-e)(g(e-c)+(e-d)^2}{g(e-c)}=0$$

$\Leftrightarrow$ [g≠0 and e≠c]

$$\left(g(b-c)(e-c)+(e-d)^2\right)^2 - f(e-d)^2\left((e-d)^2 - g(e-c)^2\right)=0$$

$$\Leftrightarrow$$

$$g^2(b-e)^2(e-c)^2 + 2g(b-e)(e-c)(e-d)^2 + (e-d)^4 - f(e-d)^4 - fg(e-d)^2(e-c)^2 = 0$$

$$\Leftrightarrow$$

$$\begin{aligned}
&e^4(fg - f + 1 - 2g + g^2)\\
&+e^3(-2fgd - 2fgc - 4d + 4fd + 2gc + 4gd + 2gb - 2g^2c - 2g^2b)\\
&+e^2(fgd^2 + 4f\,\mathrm{gcd} + fgc^2 - 6fd^2 + 6d^2 - 4\,\mathrm{gcd} - 2gd^2 - 2gbc - 4gbd + g^2c^2 + 4g^2bc + g^2b^2)\\
&+e(-2f\,\mathrm{gcd}^2 - 2fgc^2d + 4d^3f - 4d^3 + 2\,\mathrm{gcd}^2 + 4gbcd + 2gbd^2 - 2g^2bc^2 - 2g^2b^2c)\\
&+(fgc^2d^2 - fd^4 + d^4 - 2gbcd^2 + g^2b^2c^2 = 0
\end{aligned}$$

The correct solution for e can now be calculated with a Newton iteration with a starting value > e, for example with the start value

$$d + \frac{d-c}{(1-g)^2}$$

If g=0 and f>0 then switch f and g, b and d and later the solution a and e.

sbRandTrigen Programmcode

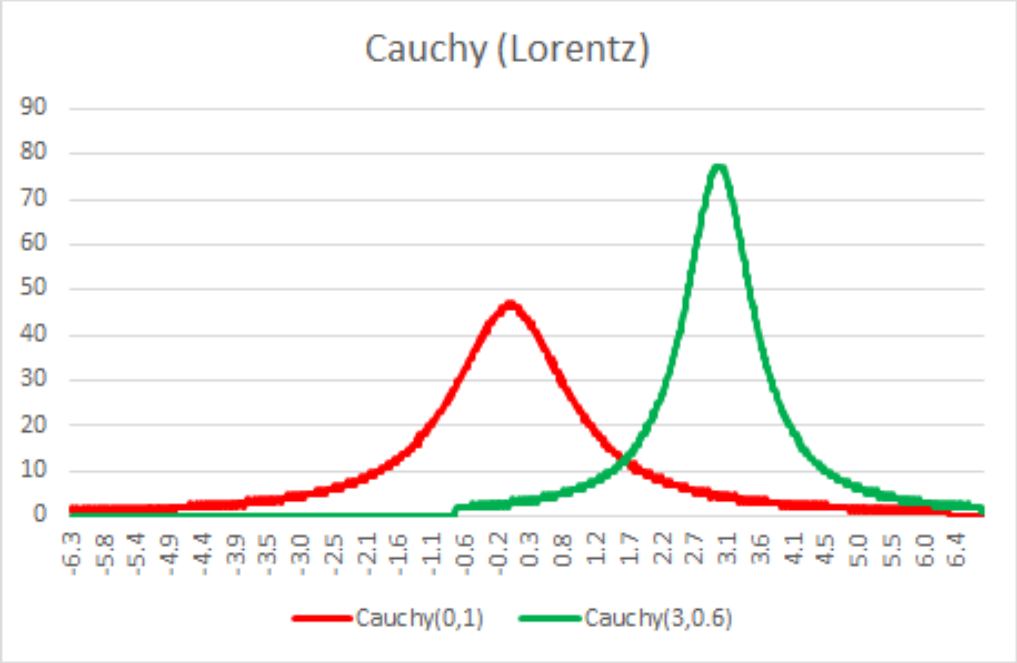
Bitte beachten Sie, daß *sbRandTrigen sbRandTriang* benötigt und aufruft.

```
Function sbRandTrigen(dBottom As Double, dMode As Double, _
    dTop As Double, dBottomPerc As Double, _
    dTopPerc As Double, Optional dRandom = 1#) As Double
'Generates dMin random number, Triang distributed
'with given first and last decile
'[see Vose: Risk Analysis, 2nd ed., p. 129]
'(C) (P) by Bernd Plumhoff 19-Nov-2011 PB V0.32
'Similar to @RISK's (C) RiskTrigen function.
'sbRandTrigen(bottom, mode, top, bottom percentile, top percentile)
'specifies a triangular distribution with three points – one
'at the mode and two at the specified bottom and top percentiles.
'The bottom percentile and top percentile are values between
'0 and 100. Each percentile value gives the percentile of the
'total area under the triangle that is on the left side of the
'given point.
'Example:
'sbRandTrigen(1,8,10,20,95) will call
'sbRandTriang(-6.13212712795534, 8, 11.8648937411641).
'Please ensure that you execute Randomize before you call
'this function for the first time.

Static dBottomLast As Double
Static dModeLast As Double
Static dTopLast As Double
Static dBottomPercLast As Double
Static dTopPercLast As Double
Static dMin As Double
Static dMax As Double
Dim dMaxNew As Double
Dim da0 As Double, da1 As Double, da2 As Double
Dim da3 As Double, da4 As Double
```

```
Dim dfe As Double, df1e As Double
Dim dBottomPerc2 As Double, dTopPerc2 As Double
Dim i As Long

If dBottom = dBottomLast And dMode = dModeLast And dTop = dTopLast _
   And dBottomPerc = dBottomPercLast And dTopPerc = dTopPercLast _
   And Not IsError(dMin) Then
    sbRandTrigen = sbRandTriang(dMin, dMode, dMax, dRandom)
    Exit Function
End If

dBottomLast = dBottom
dModeLast = dMode
dTopLast = dTop
dBottomPercLast = dBottomPerc
dTopPercLast = dTopPerc

dBottomPerc2 = dBottomPerc / 100#
dTopPerc2 = 1# - dTopPerc / 100#
If dMode <= dBottom Or dTop <= dMode Then
    dMin = CVErr(xlErrValue) 'Trigger rerun next time
    sbRandTrigen = CVErr(xlErrValue)
    Exit Function
End If
If dBottomPerc2 < 0# Or dTopPerc2 < 0# Then
    dMin = CVErr(xlErrDiv0) 'Trigger rerun next time
    sbRandTrigen = CVErr(xlErrValue)
    Exit Function
End If

If dTopPerc2 = 0# Then
    If dBottomPerc2 = 0# Then
        sbRandTrigen = sbRandTriang(dBottom, dMode, dTop, dRandom)
        Exit Function
    End If
    sbRandTrigen = sbRandTrigen(dBottom, dMode, dTop, dBottomPerc2, dTopPerc2)
    Exit Function
End If

da4 = dBottomPerc2 * dTopPerc2 - dBottomPerc2 + 1# - 2# * dTopPerc2 + dTopPerc2 ^ 2#
da3 = -2# * dBottomPerc2 * dTopPerc2 * dTop - 2# * dBottomPerc2 * dTopPerc2 * dMode - _
      4# * dTop + 4# * dBottomPerc2 * dTop + 2# * dTopPerc2 * dMode + 4# * dTopPerc2 * _
      dTop + 2# * dTopPerc2 * dBottom - 2# * dTopPerc2 ^ 2# * dMode - _
      2# * dTopPerc2 ^ 2# * dBottom
da2 = dBottomPerc2 * dTopPerc2 * dTop ^ 2# + 4# * dBottomPerc2 * dTopPerc2 * dMode * _
      dTop + dBottomPerc2 * dTopPerc2 * dMode ^ 2# - 6# * dBottomPerc2 * dTop ^ 2# + _
      6# * dTop ^ 2# - 4# * dTopPerc2 * dMode * dTop - 2# * dTopPerc2 * dTop ^ 2# - 2# * _
      dTopPerc2 * dBottom * dMode - 4# * dTopPerc2 * dBottom * dTop + dTopPerc2 ^ 2# * _
      dMode ^ 2# + 4# * dTopPerc2 ^ 2# * dBottom * dMode + dTopPerc2 ^ 2# * dBottom ^ 2#
da1 = -2# * dBottomPerc2 * dTopPerc2 * dMode * dTop ^ 2# - 2# * dBottomPerc2 * dTopPerc2 * _
      dMode ^ 2# * dTop + 4# * dTop ^ 3# * dBottomPerc2 - 4# * dTop ^ 3# + 2# * dTopPerc2 * _
      dMode * dTop ^ 2# + 4# * dTopPerc2 * dBottom * dMode * dTop + 2# * dTopPerc2 * _
      dBottom * dTop ^ 2# - 2# * dTopPerc2 ^ 2# * dBottom * dMode ^ 2# - 2# * _
      dTopPerc2 ^ 2# * dBottom ^ 2# * dMode
da0 = dBottomPerc2 * dTopPerc2 * dMode ^ 2# * dTop ^ 2# - dBottomPerc2 * dTop ^ 4# + dTop ^ 4# - _
      2# * dTopPerc2 * dBottom * dMode * dTop ^ 2# + dTopPerc2 ^ 2# * dBottom ^ 2# * dMode ^ 2#

dMax = dTop + (dTop - dMode) / (1# - dTopPerc2) ^ 2#

'Newton iteration
Do While Abs(dMaxNew - dMax) > 0.000000000001

    i = i + 1
    If i > 30 Then
        If Abs(dfe) > 0.000000000001 Then
            dMin = CVErr(xlErrDiv0) 'Trigger rerun next time
            sbRandTrigen = CVErr(xlErrValue)
            Exit Function
        Else
            Exit Do
        End If
    End If
    dMaxNew = dMax
    dfe = da4 * dMaxNew ^ 4# + da3 * dMaxNew ^ 3# + da2 * dMaxNew ^ 2# + da1 * dMaxNew + da0
    df1e = 4# * da4 * dMaxNew ^ 3# + 3# * da3# * dMaxNew ^ 2# + 2# * da2 * dMaxNew + da1
    dMax = dMax - dfe / df1e

Loop

dMin = dMax - (dMax - dTop) ^ 2# / dTopPerc2 / (dMax - dMode)
sbRandTrigen = sbRandTriang(dMin, dMode, dMax, dRandom)
End Function
```

sbRandCauchy

Wenn Sie simulieren wollen, wie Partikel ausgehend von einem festen Punkt auf eine Gerade treffen, können Sie eine Cauchy Verteilung verwenden. Diese Verteilung wird manchmal auch Lorentz Verteilung genannt. Auch der Quotient zweier (0,1) Normalverteilungen führt zu einer Cauchy Verteilung.



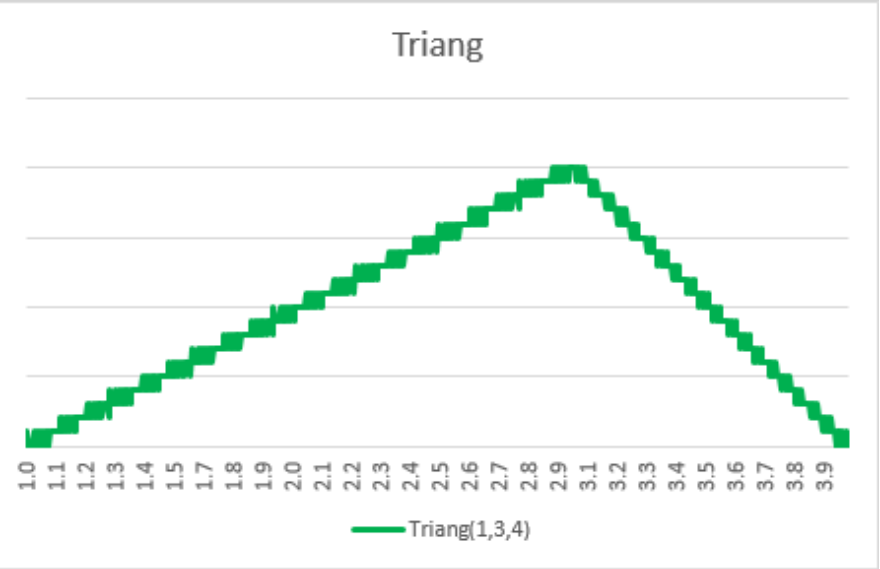
sbRandCauchy Programmcode

```
Const GCPi = 3.14159265358979

Function sbRandCauchy(dLocation As Double, dScale As Double, _
  Optional dRandom = 1#) As Double
'(C) (P) by Bernd Plumhoff 03-Nov-2020 PB V0.2
Static bRandomized As Boolean
Dim dRand As Double
If dRandom < 0# Or dRandom > 1# Or dScale <= 0# Then
  sbRandCauchy = CVErr(xlErrValue)
  Exit Function
End If
If Not bRandomized Then
  Randomize
  bRandomized = True
End If
If dRandom = 1# Then
  dRand = Rnd()
Else
  dRand = dRandom
End If
sbRandCauchy = dLocation + dScale * Tan((dRand - 0.5) * GCPi)
End Function
```

sbRandCDFInv

Sie können Zufallszahlen mit einer gewünschten Verteilung leicht erzeugen, wenn die inverse Verteilungsfunktion explizit vorliegt.

Ein Beispiel einer geschichteten Stichprobe:



Ohne die explizit vorliegende inverse Verteilungsfunktion können Sie eine lineare Approximation mit der Function *sbRandPDF* anwenden.

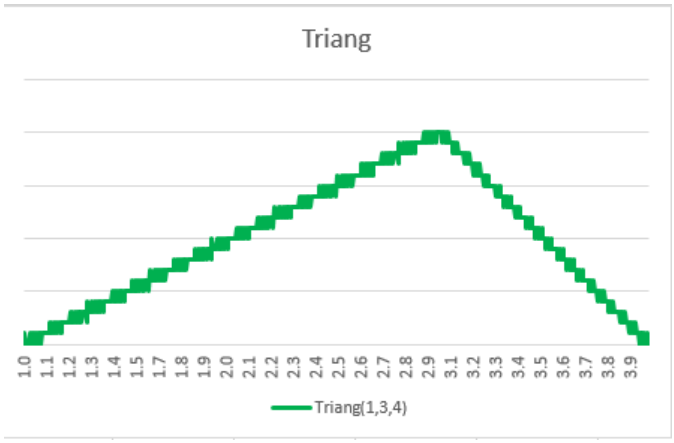
sbRandCFDInv Programmcode

```
Function sbRandCDFInv(dParam1 As Double, dParam2 As Double, _
    dParam3 As Double, Optional dRandom = 1#) As Double
'(C) (P) by Bernd Plumhoff  03-Nov-2020 PB V0.2
Static bRandomized As Boolean
Dim dRand As Double
If dRandom < 0# Or dRandom > 1# Then
  sbRandCDFInv = CVErr(xlErrValue)
  Exit Function
End If
If Not bRandomized Then
  Randomize
  bRandomized = True
End If
If dRandom = 1# Then
    dRand = Rnd()
Else
    dRand = dRandom
End If
'Here you need to define the inverse of the cumulative distribution function
sbRandCDFInv = sbRandTriang(dParam1, dParam2, dParam3, dRand)
End Function
```

sbRandPDF

Mit einer expliziten Form der inversen Verteilungsfunktion sollten Sie *sbRandCDFInv* verwenden. Aber wenn eine solche nicht vorliegt, können Sie mit einer linearen Approximation die Funktion *sbRandPDF* nutzen. Unglücklicherweise ist dieser Ansatz sehr rechenintensiv, selbst wenn Sie die Anzahl der Punkte bei identischen oder nahezu identischen Steigungen reduzieren können.

Ein Beispiel einer geschichteten Stichprobe:

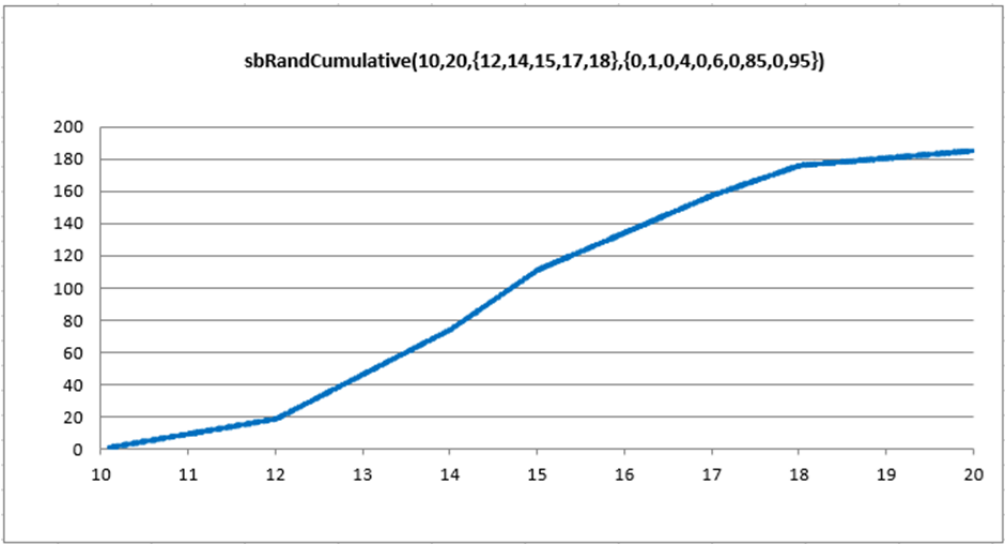


sbRandPDF Programmcode

```
Function sbRandPDF(Optional dParam1, Optional dParam2, _
    Optional dParam3, Optional dRandom = 1#) As Double
'(C) (P) by Bernd Plumhoff  12-Sep-2014 PB V0.15
Dim dRand As Double
Dim i As Long
Static dPar1 As Double
Static dPar2 As Double
Static dPar3 As Double
Static vX(0 To 1000) As Variant
Static vY(0 To 1000) As Variant
If dRandom < 0# Or dRandom > 1# Then
  sbRandPDF = CVErr(xlErrValue)
  Exit Function
End If
If dRandom = 1# Then
  dRand = Rnd()
Else
  dRand = dRandom
End If
If dParam1 <> dPar1 Or dParam2 <> dPar2 Or dParam3 <> dPar3 Then
  dPar1 = dParam1
  dPar2 = dParam2
  dPar3 = dParam3
  'Initialize RandGeneral call parameters
  For i = 0 To 1000
    vX(i) = dPar1 + i * (dPar3 - dPar1) / 1000#
    'Now we can insert an arbitrary PDF function
    If vX(i) < dPar2 Then
      vY(i) = (vX(i) - dPar1) / ((dPar3 - dPar1) * (dPar2 - dPar1))
      If vY(i) < 0# Then vY(i) = 0#
    Else
      vY(i) = (dPar3 - vX(i)) / ((dPar3 - dPar1) * (dPar3 - dPar2))
      If vY(i) < 0# Then vY(i) = 0#
    End If
  Next i
End If
'Depending on the PDF input range you need to feed start
'and end values to sbRandGeneral
sbRandPDF = sbRandGeneral(dPar1, dPar3, vX, vY, dRand)
End Function
```

sbRandCumulative

Wenn Sie eine schrittweise kumulierte Verteilungsfunktion von Zufallszahlen erzeugen müssen, können Sie diese benutzerdefinierte Funktion *sbRandCumulative* verwenden. Die Herleitung des Algorithmus ist analog zu *sbRandGeneral* (Herleitung des Algorithmus).



sbRandCumulative Programmcode

```
Function sbRandCumulative(dMin As Double, dMax As Double, _
    vXi As Variant, vWi As Variant, Optional dRandom = 1#) As Double
'Generates a random number, Cumulative distributed. [see Vose: Risk Analysis, 2nd ed., p. 109]
'(C) (P) by Bernd Plumhoff 23-Dec-2020 PB V0.50
'Similar to @RISK's (C) RiskCumulative function.
Static bRandomized As Boolean, i As Long
Dim dA As Double, dRand As Double, dSgn As Double

If vWi.Count <> vXi.Count Then sbRandCumulative = CVErr(xlErrValue): Exit Function
ReDim dX(0 To vXi.Count + 1) As Double
ReDim dW(0 To vWi.Count + 1) As Double

dX(0) = dMin
dX(UBound(dX)) = dMax
dW(0) = 0#
dW(UBound(dW)) = 1#
For i = 1 To vXi.Count
  dX(i) = vXi(i)
  dW(i) = vWi(i)
  If dW(i) < dW(i - 1) Then
    'Weights need to be monotonously increasing
    sbRandCumulative = CVErr(xlErrValue)
    Exit Function
  End If
Next i
If dW(UBound(dW)) < dW(UBound(dW) - 1) Then
  'Weights need to be monotonously increasing
  sbRandCumulative = CVErr(xlErrValue)
  Exit Function
End If

'Calculate area
dA = 0#
For i = 0 To UBound(dX) - 1
  If dX(i) >= dX(i + 1) Or dW(i) < 0# Then
    sbRandCumulative = CVErr(xlErrValue)
    Exit Function
  End If
  dA = dA + (dX(i + 1) - dX(i)) * (dW(i + 1) + dW(i)) / 2#
Next i

'Normalise weights to set area to 1
For i = 1 To UBound(dW)
  dW(i) = dW(i) / dA
Next i

ReDim dF(0 To UBound(dX)) As Double
'Calculate border points of value ranges for
'cumulative inverse function
dF(0) = 0#
dA = 0#
For i = 0 To UBound(dX) - 1
  dA = dA + (dX(i + 1) - dX(i)) * (dW(i + 1) + dW(i)) / 2#
  dF(i + 1) = dA
Next i

If dRandom = 1# Then
  If Not bRandomized Then
    Randomize
    bRandomized = True
  End If
  dRand = Rnd()
Else
  dRand = dRandom
End If

i = 1
Do While dF(i) <= dRand
  i = i + 1
Loop
dSgn = Sgn(dW(i) - dW(i - 1))
If dSgn = 0# Then
  sbRandCumulative = dX(i - 1) + (dRand - dF(i - 1)) / _
                (dF(i) - dF(i - 1)) * (dX(i) - dX(i - 1))
Else
  sbRandCumulative = dX(i - 1) + _
                dSgn * Sqr((dRand - dF(i - 1)) * _
                2# * (dX(i) - dX(i - 1)) / (dW(i) - dW(i - 1)) + _
                (dW(i - 1) * (dX(i) - dX(i - 1)) / _
                (dW(i) - dW(i - 1))) ^ 2#) - _
                dW(i - 1) * (dX(i) - dX(i - 1)) / (dW(i) - dW(i - 1))
End If

End Function
```

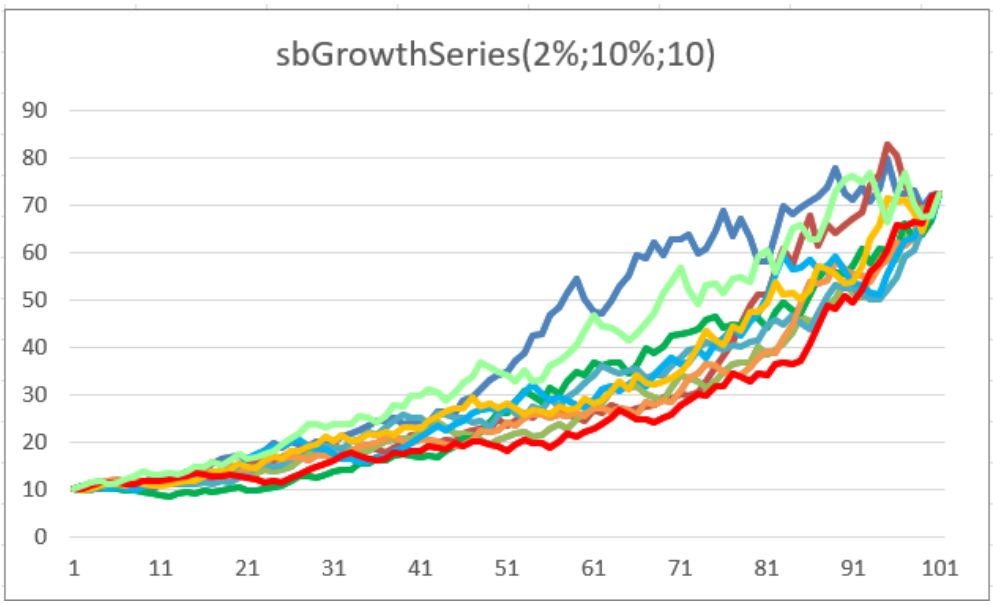
Brownsche Brücken

Eine brownsche Brücke ist eine brownsche (zufällige) Bewegung mit bestimmtem Start- und Endwert. Sie wird zur Modellierung von zufälligen Entwicklungen von Zeitreihen verwendet, deren Wert zu zwei Zeitpunkten bekannt ist.

Neben den hier vorgestellten Beispielen kann auch *sbRandIntFixSum* als brownsche Brücke angesehen werden.

sbGrowthSeries

Sie können Zufallszahlen mit einer kumulierten Wachstumsrate *dblRate*, mit einer maximalen relativen Änderungsrate pro Zeitschritt *dblMaxRatePerStep* und mit einem optionalen Startwert *dblStartVal* erzeugen. Die Anzahl der Zeitschritte (Perioden) wird implizit durch die Anzahl der ausgewählten Zellen gewählt, in die der Funktionsaufruf als Matrixformel mit STRG + SHIFT + ENTER eingegeben wird. Dies ist eine spezielle Art von Brownscher Brücke.



sbGrowthSeries Programmcode

```
Function sbGrowthSeries(dblRate As Double, _
       dblMaxRatePerStep As Double, _
       Optional dblStartVal As Double = 1#) As Variant
'Returns random data with a compound growth rate dblRate, with
'a maximal relative change rate per step of dblMaxRatePerStep
'and with a start value dblStartVal. The number of periods
'is implicitly chosen by the number of selected cells which
'call this function as an array formula (entered with
'CTRL + SHIFT + ENTER). This is sort of a brownian bridge.
'(C) (P) by Bernd Plumhoff 20-Mar-2011 PB V0.91

Dim vR As Variant
Dim lP As Long 'Periods
Dim lrow As Long
Dim lcol As Long
Dim dblCurrVal As Double
Dim dblCurrRate As Double
Dim dblCurrMin As Double
Dim dblCurrMax As Double
Dim dblRelMin As Double
Dim dblRelMax As Double
Dim dblEndVal As Double

If TypeName(Application.Caller) <> "Range" Then
  sbGrowthSeries = CVErr(xlErrRef)
  Exit Function
End If

If Application.Caller.Rows.Count <> 1 And _
  Application.Caller.Columns.Count <> 1 Then
  sbGrowthSeries = CVErr(xlErrValue)
  Exit Function
End If

If Abs(dblRate) > dblMaxRatePerStep Then
  sbGrowthSeries = CVErr(xlErrNum)
  Exit Function
End If

lP = Application.Caller.Count

ReDim vR(1 To Application.Caller.Rows.Count, 1 To Application.Caller.Columns.Count)

dblCurrVal = dblStartVal
dblEndVal = dblStartVal * (1# + dblRate) ^ CDbl(lP)
dblCurrMin = dblEndVal / (1# + dblMaxRatePerStep) ^ CDbl(lP)
dblCurrMax = dblEndVal / (1# - dblMaxRatePerStep) ^ CDbl(lP)
For lrow = 1 To UBound(vR, 1)
  For lcol = 1 To UBound(vR, 2)
    dblCurrRate = (dblEndVal / dblCurrVal) ^ (1# / CDbl(lP - lcol * lrow + 1)) - 1#
    dblCurrMin = dblCurrMin * (1# + dblMaxRatePerStep)
    dblCurrMax = dblCurrMax * (1# - dblMaxRatePerStep)
    dblRelMin = (dblCurrMin - dblCurrVal) / dblCurrVal
    If dblRelMin < -dblMaxRatePerStep Then
      dblRelMin = -dblMaxRatePerStep
    End If
    dblRelMax = (dblCurrMax - dblCurrVal) / dblCurrVal
    If dblRelMax > dblMaxRatePerStep Then
      dblRelMax = dblMaxRatePerStep
    End If
    If dblCurrRate - dblRelMin < dblRelMax - dblCurrRate Then
      dblRelMax = 2# * dblCurrRate - dblRelMin
    Else
      dblRelMin = 2# * dblCurrRate - dblRelMax
    End If
    dblCurrVal = dblCurrVal * (1# + (dblRelMin + dblRelMax) / 2# + (Rnd() - 0.5) * (dblRelMax - dblRelMin))
    vR(lrow, lcol) = dblCurrVal
  Next lcol
Next lrow

sbGrowthSeries = vR

End Function
```

Fixe Summe aus verschiedenen Zufallsbereichen

Sie benötigen sieben Zufallszahlen aus verschiedenen Zahlenbereichen, die zusammen genau 100 ergeben?

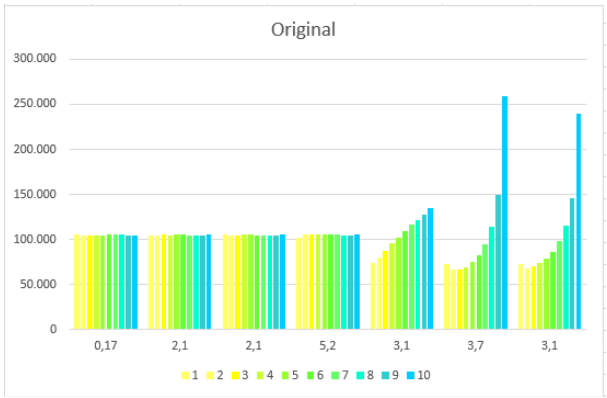
	A	B	C	D	E	F	G	H	I	J	K
1	**Zielwert**	100								**Summe**	**Check**
2	**Untergrenze**	0,27	2,8	15,3	43,3	15,1	9,8	2,7		89,27	
3	**Obergrenze**	0,44	4,9	17,4	48,5	18,2	13,5	5,8		108,74	
4	**Formellösungen:**										
5		0,40206022	3,51356284	16,5085186	44,8899007	17,3533623	11,5549935	5,77760189		100	
6		0,39120156	3,44220507	17,1827317	47,9229809	16,7717078	10,9405165	3,34865645		100	
7		0,40951144	3,79485159	15,8092308	47,7845392	15,2303171	11,6556437	5,31590611		100	
8		0,4239384	4,15342804	16,0116953	45,3050432	15,6557401	12,8155512	5,63460382		100	
9		0,3845167	3,64892537	15,6595322	43,5092385	18,00071	13,0754472	5,72162988		100	
10		0,30599978	3,54247014	17,0732974	47,3858742	17,8938342	10,0380864	3,76043783		100	
11		0,41335292	3,60723858	15,8304168	46,851858	18,1192622	11,5286128	3,64925862		100	
12		0,28130997	3,51554573	17,3374028	45,6987532	16,5240956	13,4954458	3,14744687		100	
13		0,28909348	3,73813934	16,4378585	47,4888238	17,6815508	10,2536789	4,11085527		100	
14		0,30455272	3,96973418	16,3162646	46,7935995	15,2630869	11,7514806	5,60128144		100	

Tabellenblattformeln		
Bereich	Formel	
J2:J3;J5:J14	**J2**	=SUMME(B2:H2)
K2	**K2**	=WENN(J2>$B$1;"Keine Lösung, weil Summe der Untergrenzen größer als " & $B$1 & " ist.";"")
K3	**K3**	=WENN(J3<$B$1;"Keine Lösung, weil Summe der Obergrenzen kleiner als " & $B$1 & " ist.";"")
B5:H14	**B5**	=MAX(B$2;$B$1-SUMME($A5:A5)-SUMME(C$3:$I$3))+ZUFALLSZAHL()*(MIN(B$3;$B$1-SUMME($A5:A5)-SUMME(C$2:$I$2))-MAX(B$2;$B$1-SUMME($A5:A5)-SUMME(C$3:$I$3)))

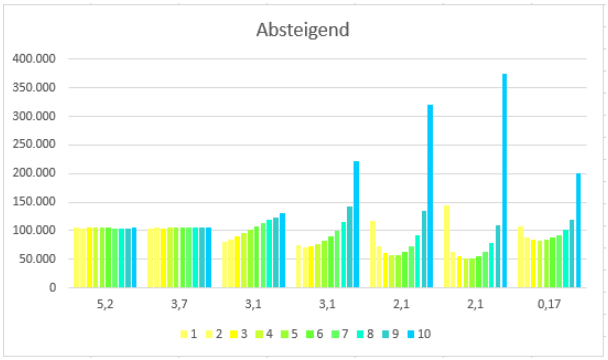
Wichtiger Hinweis: Es existiert keine Lösung, wenn die Summe der Untergrenzen größer als 100 ist oder wenn die Summe der Obergrenzen kleiner ist als 100! Dies wird in den Zellen K2:K3 geprüft.

Die Verteilung der Zufallszahlen

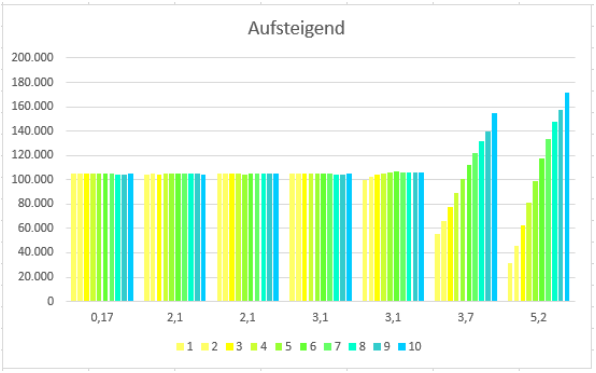
Die erzeugten Zufallszahlen im obigen Beispiel sind relativ gleichverteilt. Bei 1.048.572 erzeugten Reihen von jeweils 7 Zahlen erhält man für die originale Sortierung der Grenzwert-Korridore (siehe rechts):



Bei absteigender Sortierung der Grenzwerte nach Korridorbreite erhält man (siehe links):



Bei aufsteigender Sortierung nach Korridorbreite:

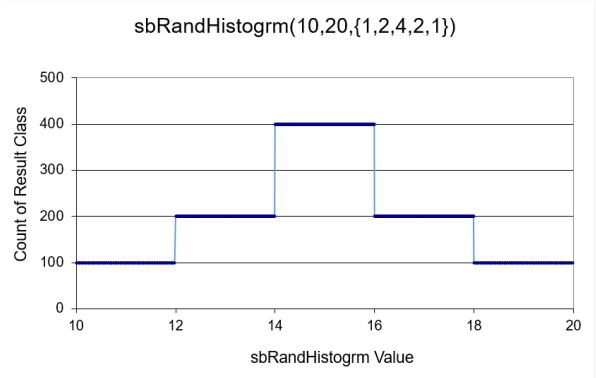


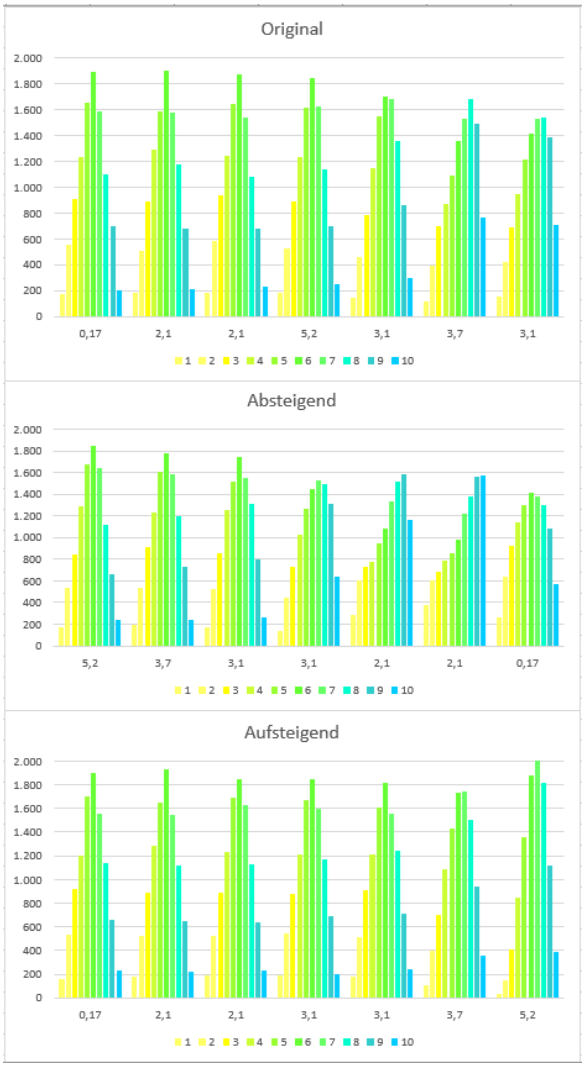
Um möglichst gleichverteilte Zufallszahlen zu erhalten, sollte man deshalb die Spalten nach aufsteigenden Korridorbreiten sortieren, weil die erzeugenden Formeln die Freiheitsgrade von links nach rechts reduzieren. Falls Sie - aus welchen Gründen auch immer - absteigend sortierte Grenzwertbereiche vorliegen haben, müssen Sie mit deutlich extremeren Verteilungen rechnen.

Mit einer Dreiecksverteilung

Mit der DreiecksverteilungsbRandHistogrm

Eine stufenweise konstante Verteilung ist die Histogram Verteilung:





sbRandHistogram Programmcode

```
Function sbRandHistogrm(dmin As Double, dMax As
Double, _
            vWeight As Variant, Optional dRandom = 1#)
As Double
'Specifies a histogram distribution with range
dmin:dmax.
'This range is divided into vWeight.count classes.
Each
'class has weight vWeight(i) reflecting the
probability
'of occurrence of a value within the class.
'Similar to @Risk's function RiskHistogrm.
'(C) (P) by Bernd Plumhoff 18-Oct-2020 PB V1.01

Dim i As Long, n As Long, vW As Variant
Dim dRand As Double, dR As Double, dSumWeight As
Double

With Application.WorksheetFunction
vW = .Transpose(.Transpose(vWeight))
End With

n = UBound(vW)
ReDim dSumWeightI(0 To n) As Double

dSumWeight = 0#
dSumWeightI(0) = 0#
For i = 1 To n
```

```
    If vW(i) < 0# Then 'A negative weight is an error
        sbRandHistogrm = CVErr(xlErrValue)
        Exit Function
    End If
    dSumWeight = dSumWeight + vW(i) 'Calculate sum of all weights
    dSumWeightI(i) = dSumWeight     'Calculate sum of weights till i
Next i

If dSumWeight = 0# Then  'Sum of weights has to be greater than zero
    sbRandHistogrm = CVErr(xlErrValue)
    Exit Function
End If

If dRandom = 1# Then
    dRand = Rnd()
Else
    dRand = dRandom
End If
dR = dSumWeight * dRand

i = n
Do While dR < dSumWeightI(i)
    i = i - 1
Loop

sbRandHistogrm = dmin + (dMax - dmin) * _
     (CDbl(i) + (dR - dSumWeightI(i)) / vW(i + 1)) / CDbl(n)

End Function
```

Ähnliche Verteilungen erzeugen die Programme rww und redw:

rww Programmcode

```
Function rww(ParamArray w() As Variant) As Double
'Produces random numbers with defined widths & weights
'Rww expects a vector of n random widths and weightings
'of type double and returns a random number of type double.
'This random number will lie in the given n-width-range of the
'(0,1)-interval with the given likelihood of the n weightings.
' Call Randomize before calling rww!
'(C) (P) by Bernd Plumhoff 06-Aug-2004 PB V0.50
'Examples:
'a) rww(1,0,1,1,8,0) will return a random number between 0.1 and 0.2
'b) rww(5,2,5,1) will return a random number between 0 and 0.5 twice as
'   often as a random number between 0.5 and 1.
'c) rww(1/3,0,1/3,1,1/3,0) will return a random number between
'   0.33333333333333 and 0.66666666666666.
'd) rww(5,15.4,3,7.7,2,0) would return a random value between
'   0 and 0.8, first 5 deciles with double likelihood than decile 6-8.

Dim i As Long
Dim swidths As Double
Dim sw As Double

If (UBound(w) + 1) Mod 2 <> 0 Then
  rww = -2     'No even number of arguments: Error
  Exit Function
End If

ReDim swidthsi(0 To (UBound(w) + 1) / 2 + 1) As Double
ReDim swi(0 To (UBound(w) + 1) / 2 + 1) As Double
ReDim weights(0 To (UBound(w) + 1) / 2) As Double
ReDim widths(0 To (UBound(w) + 1) / 2) As Double

swidths = 0#
sw = 0#
swi(0) = 0#
swidthsi(0) = 0#
For i = 0 To (UBound(w) - 1) / 2
  If w(2 * i) < 0# Then     'A negative width is an error
    rww = -3#
    Exit Function
  End If
  widths(i) = w(2 * i)
  swidths = swidths + widths(i)
  swidthsi(i + 1) = swidths
  If w(2 * i + 1) < 0# Then 'A negative weight is an error
    rww = -1#
    Exit Function
  End If
  weights(i) = w(2 * i + 1)
  If widths(i) > 0# Then
    sw = sw + weights(i)
  End If
  swi(i + 1) = sw
Next i
rww = sw * Rnd
'i = (UBound(w) - 1) / 2 + 1 'i already equals (UBound(w) - 1)/2 + 1, you may omit this statement.
Do While rww < swi(i)
  i = i - 1
Loop

rww = (swidthsi(i) + (rww - swi(i)) / weights(i) * widths(i)) / swidths

End Function
```

redw Programmcode

```
Function redw(ParamArray vWeights() As Variant) As Double
'Produces random numbers with equidistant weights. Redw expects a vector of n random
'weights of type double and returns a random number of type double. This random
'number will lie in the given equidistant n-split-range of the [0,1)-interval with
'the given likelihood of weightings. Call Randomize before calling redw! '(C) (P) by Bernd Plumhoff 09-Dec-
2009 PB V0.50
Examples:
'a) redw(0,1,0,0,0,0,0,0,0,0) will return a random number d, 0.1 <= d < 0.2
'b) redw(2,1) will return a random number between 0 and 0.5 twice as
'   often as a random number between 0.5 and 1.
'c) redw(0,1,0) will return a random number d, 0.333333333333333 <= d < 0.666666666666666.
'd) redw(15.4,15.4,15.4,15.4,15.4,7.7,7.7,7.7,0,0) would return a random value between
'   0 and 0.8, first 5 deciles with double likelihood than decile 6-8.

Dim i As Long
Dim dw As Double
ReDim dwi(0 To UBound(vWeights) + 2) As Double

dw = 0#
dwi(0) = 0#
For i = 0 To UBound(vWeights)
  If vWeights(i) < 0# Then  'A negative weight is an error
    redw = CVErr(xlErrValue)
    Exit Function
  End If
  dw = dw + vWeights(i)     'Calculate sum of all weights
  dwi(i + 1) = dw           'Calculate sum of weights till i
Next i

redw = dw * Rnd

'i = UBound(vWeights) + 1 'i already equals UBound(vWeights) + 1, you may omit this statement.
Do While redw < dwi(i)
  i = i - 1
Loop

redw = (CDbl(i) + (redw - dwi(i)) / vWeights(i)) / (CDbl(UBound(vWeights) + 1))

End Function
```

sbRandTriang erhält man bei 10.000 erzeugten Reihen:

Die entsprechende Formel in Zelle B5 lautet: *=sbRandTriang(MAX(B$2;$B$1-SUMME($A5:A5) - SUMME(C$3:$I$3));MIN(MAX(MAX(B$2;$B$1 - SUMME($A5:A5) - SUMME(C$3:$I$3));B$2+($B$1 - (SUMME($A5:A5) + SUMME(B$2:$I$2))) / (SUMME(B$3:$I$3) - SUMME(B$2:$I$2)) * (B$3-B$2));MIN(B$3;$B$1 - SUMME($A5:A5) - SUMME(C$2:$I$2)));MIN(B$3;$B$1 - SUMME($A5:A5) - SUMME(C$2:$I$2)))*.

Gerundete Ergebnisse

Müssen die Ergebnisse auf eine bestimmte Anzahl von Nachkommastellen gerundet werden, dann kann die oben genannte allgemeine Formel z. B. für 2 Nachkommastellen in =RUNDEN(…; 2) eingebettet werden.

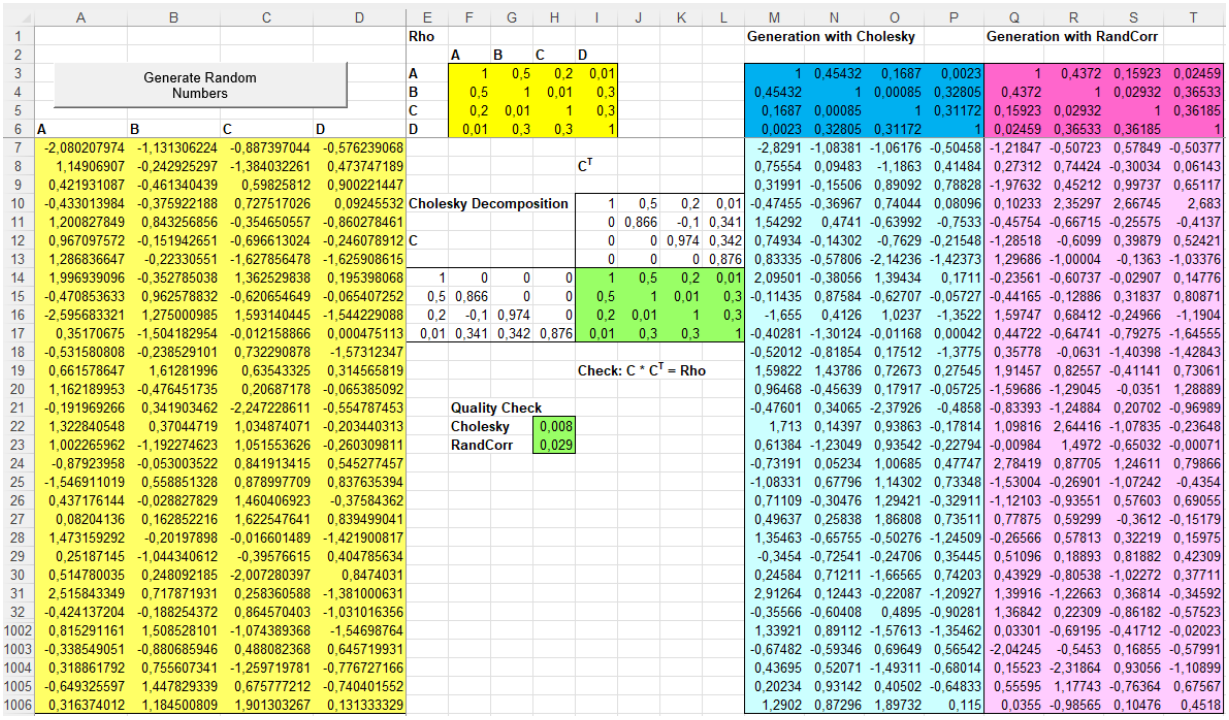
Wichtig ist nur, dass mindestens auf die maximale Anzahl der in den Grenzwerten verwendeten Nachkommastellen gerundet wird, damit

- die Ergebnisse nach Rundung noch im Korridorbereich sind
- nicht Teile des Korridorbereiches durch die Rundung unerreichbar werden
- das Zielergebnis immer erreicht wird.

Korrelierte Zufallszahlen

Cholesky Zerlegung

Mit der Cholesky (sprich: “Koleski”) Zerlegung können Sie korrelierte Zufallszahlen erzeugen:



Generate Random Numbers

	A	B	C	D
A	-2,080207974	-1,131306224	-0,887397044	-0,576239068
B	1,14906907	-0,242925297	-1,384032261	0,473747189
C	0,421931087	-0,461340439	0,59825812	0,900221447
D	-0,433013984	-0,375922188	0,727517026	0,09245532
	-0,380827849	0,843256856	-0,354650557	-0,860278461
	0,967097572	-0,151942651	-0,696613024	-0,246078912
	1,286836647	-0,22330551	-1,627856478	-1,625908615
	1,996939096	-0,352785038	1,362529838	0,195398068
	-0,470853633	0,962578832	-0,620654649	-0,065407252
	-2,595683321	1,275000985	1,593140445	-1,544229088
	0,35170675	-1,504182954	-0,012158866	0,000475113
	-0,531580808	-0,238529101	0,732290878	-1,57312347
	0,661578647	1,61281996	0,63543325	0,314565819
	1,162189953	-0,476451735	0,20687178	-0,065385092
	-0,191969266	0,341903462	-2,247228611	-0,554787453
	1,322840548	0,37044719	1,034874071	-0,203440313
	1,002265962	-1,192274623	1,051553626	-0,260309811
	-0,87923958	-0,053003522	0,841913415	0,545277457
	-1,546911019	0,558851328	0,878997709	0,837635394
	0,437176144	-0,028827829	1,460406923	-0,37584362
	0,08204136	0,162852216	1,622547641	0,839499041
	1,473159292	-0,20197898	-0,016601489	-1,421900817
	0,25187145	-1,044340612	-0,39576615	0,404785634
	0,514780035	0,248092185	-2,007280397	0,8474031
	2,515843349	0,717871931	0,258360588	-1,381000631
	-0,424137204	-0,188254372	0,864570403	-1,031016356
	0,815291161	1,508528101	-1,074389368	-1,54698764
	-0,338549051	-0,880685946	0,488082368	0,645719931
	0,318861792	0,755607341	-1,259719781	-0,776727166
	-0,649325597	1,447829339	0,675777212	-0,740401552
	0,316374012	1,184500809	1,901303267	0,131333329

Rho

	A	B	C	D
A	1	0,5	0,2	0,01
B	0,5	1	0,01	0,3
C	0,2	0,01	1	0,3
D	0,01	0,3	0,3	1

Cholesky Decomposition

1	0	0	0
0,5	0,866	0	0
0,2	-0,1	0,974	0
0,01	0,341	0,342	0,876

C^T

1	0,5	0,2	0,01
0	0,866	-0,1	0,341
0	0	0,974	0,342
0	0	0	0,876

Check: $C * C^T = Rho$

1	0,5	0,2	0,01
0,5	1	0,01	0,3
0,2	0,01	1	0,3
0,01	0,3	0,3	1

Quality Check

Cholesky	0,008
RandCorr	0,029

Generation with Cholesky

1	0,45432	0,1687	0,0023
0,45432	1	0,00085	0,32805
0,1687	0,00085	1	0,31172
0,0023	0,32805	0,31172	1
-2,8291	-1,08381	-1,06176	-0,50458
0,75554	0,09483	-1,1863	0,41484
0,31991	-0,15506	0,89092	0,78828
-0,47455	-0,36967	0,74044	0,08096
1,54292	0,4741	-0,63992	-0,7533
0,74934	-0,14302	-0,7629	-0,21548
0,83335	-0,57806	-2,14236	-1,42373
2,09501	-0,38056	1,39434	0,1711
-0,11435	0,87584	-0,62707	-0,05727
-1,655	0,4126	1,0237	-1,3522
-0,40281	-1,30124	-0,01168	0,00042
-0,52012	-0,81854	0,17512	-1,3775
1,59822	1,43786	0,72673	0,27545
0,96468	-0,45639	0,17917	-0,05725
-0,47601	0,34065	-2,37926	-0,4858
1,713	0,14397	0,93863	-0,17814
0,61384	-1,23049	0,93542	-0,22794
-0,73191	0,05234	1,00685	0,47747
-1,08331	0,67796	1,14302	0,73348
0,71109	-0,30476	1,29421	-0,32911
0,49637	0,25838	1,86808	0,73511
1,35463	-0,65755	-0,50276	-1,24509
-0,3454	-0,72541	-0,24706	0,35445
0,24584	0,71211	-1,66565	0,74203
2,91264	0,12443	-0,22087	-1,20927
-0,35566	-0,60408	0,4895	-0,90281
1,33921	0,89112	-1,57613	-1,35462
-0,67482	-0,59346	0,69649	0,56542
0,43695	0,52071	-1,49311	-0,68014
0,20234	0,93142	0,40502	-0,64833
1,2902	0,87296	1,89732	0,115

Generation with RandCorr

1	0,4372	0,15923	0,02459
0,4372	1	0,02932	0,36533
0,15923	0,02932	1	0,36185
0,02459	0,36533	0,36185	1
-1,21847	-0,50723	0,57849	-0,50377
0,27312	0,74424	-0,30034	0,06143
-1,97632	0,45212	0,99737	0,65117
0,10233	2,35297	2,66745	2,683
-0,45754	-0,66715	-0,25575	-0,4137
-1,28518	-0,6099	0,39879	0,52421
1,29686	-1,00004	-0,1363	-1,03376
-0,23561	-0,60737	-0,02907	0,14776
-0,44165	-0,12886	0,31837	0,80871
1,59747	0,68412	-0,24966	-1,1904
0,44722	-0,64741	-0,79275	-1,64555
0,35778	-0,0631	-1,40398	-1,42843
1,91457	0,82557	-0,41141	0,73061
-1,59686	-1,29045	-0,0351	1,28889
-0,83393	-1,24884	0,20702	-0,96989
1,09816	2,64416	-1,07835	-0,23648
-0,00984	1,4972	-0,65032	-0,00071
2,78419	0,87705	1,24611	0,79866
-1,53004	-0,26901	-1,07242	-0,4354
-1,12103	-0,93551	0,57603	0,69055
0,77875	0,59299	-0,3612	-0,15179
-0,26566	0,57813	0,32219	0,15975
0,51096	0,18893	0,81882	0,42309
0,43929	-0,80538	-1,02272	0,37711
1,39916	-1,22663	0,36814	-0,34592
1,36842	0,22309	-0,86182	-0,57523
0,03301	-0,69195	-0,41712	-0,02023
-2,04245	-0,5453	0,16855	-0,57991
0,15523	-2,31864	0,93056	-1,10899
0,55595	1,17743	-0,76364	0,67567
0,0355	-0,98565	0,10476	0,4518

Tabellenblattformeln		
Bereich	Formel	
E14	**E14**	=Cholesky(F3:I6)
I10	**I10**	=MTRANS(E14:H17)
I14	**I14**	=MMULT(E14:H17;I10:L13)
Cholesky_Check	**H22**	=SUMME((F3:I6-M3:P6)^2)
RandCorr_Check	**H23**	=SUMME((F3:I6-Q3:T6)^2)
M3:P6	**M3**	=KORREL(INDEX($M$7:$P$1005;;ZEILE()-2);INDEX($M$7:$P$1005;;SPALTE()-12))
M7	**M7**	=MMULT(A7:D1006;E14:H17)
Q3:T6	**Q3**	=KORREL(INDEX($Q$7:$T$1005;;ZEILE()-2);INDEX($Q$7:$T$1005;;SPALTE()-16))
Q7	**Q7**	=RandCorr(1000;Rho)

Cholesky und RandCorr Programmcode

```
Function Cholesky(vA As Variant) As Variant
'I suggest to use the Cholesky decomposition just for purposes of demonstration.
'Better options are (in this order): tred2, tqli, eigsrt from Numerical Recipes.
'SVD also works but is computationally more expensive by far since it does not
'make use of symmetry.
'(Thanks to my former colleague Glen R.)
'Bernd Plumhoff 02-Nov-2024 PB V1.1
Dim d As Double
Dim i As Long, j As Long, k As Long, n As Long
With Application.WorksheetFunction
On Error Resume Next
vA = .Transpose(.Transpose(vA))
On Error GoTo 0
n = UBound(vA, 1)
If n <> UBound(vA, 2) Then
  Cholesky = CVErr(xlErrRef)
  Exit Function
End If
```

```
ReDim dR(1 To n, 1 To n) As Double 'Zeroing all elements
For j = 1 To n
  d = 0#
  For k = 1 To j - 1
    d = d + dR(j, k) * dR(j, k)
  Next k
  dR(j, j) = vA(j, j) - d
  If dR(j, j) > 0# Then
    dR(j, j) = Sqr(dR(j, j))
    For i = j + 1 To n
      d = 0#
      For k = 1 To j - 1
        d = d + dR(i, k) * dR(j, k)
      Next k
      dR(i, j) = (vA(i, j) - d) / dR(j, j)
    Next i
  Else
    'Cannot continue with usual Cholesky
    'Fill this column with zeros. Idea: Glen R.
    For i = j To n
      dR(i, j) = 0#
    Next i
  End If
Next j
Cholesky = dR
End With
End Function

Function RandCorr(n As Long, vVarCovar As Variant) As Variant
'Returns Ubound(vVarCovar,1) correlated random number vectors of length n.
'vVarCovar is a square matrix containing the variance/covariance matrix.
'Please notice that you will only get a "proxy" correlation, not an exact one.
'Bernd Plumhoff 06-Nov-2009 PB V0.2
Dim vA As Variant
Dim d As Double
Dim i As Long, j As Long, k As Long, m As Long

With Application.WorksheetFunction
vA = .Transpose(.Transpose(vVarCovar))
m = UBound(vA, 1)
If m <> UBound(vA, 2) Then
  RandCorr = CVErr(xlErrRef)
  Exit Function
End If

ReDim Db(1 To m, 1 To m) As Double
For j = 1 To m
  d = 0#
  For k = 1 To j - 1
    d = d + Db(j, k) * Db(j, k)
  Next k
  Db(j, j) = vA(j, j) - d
  If Db(j, j) <= 0 Then
    RandCorr = CVErr(xlErrNum)
    Exit Function
  End If
  Db(j, j) = Sqr(Db(j, j))

  For i = j + 1 To m
    d = 0#
    For k = 1 To j - 1
      d = d + Db(i, k) * Db(j, k)
    Next k
    Db(i, j) = (vA(i, j) - d) / Db(j, j)
  Next i
Next j

ReDim vR(1 To n, 1 To m) As Variant
For i = 1 To n
  For j = 1 To m
    vR(i, j) = .Norm_S_Inv(Rnd())
  Next j
Next i
vR = .MMult(vR, Db)
RandCorr = vR
End With
End Function
```

Iman-Conover Methode

Falls Sie korrelierte Zufallszahlen erzeugen müssen, ist die *Iman-Conover* Methode besser als die Cholesky Zerlegung.

1982 veröffentlichten Iman und Conover ihren ursprünglichen Artikel „A distribution-free approach to inducing rank correlation among input variables”:
https://www.uio.no/studier/emner/matnat/math/STK4400/v05/undervisningsmateriale/A%20distribution-free%20approach%20to%20rank%20correlation.pdf

Rick Wicklin schrieb dazu im Jahr 2021 „Simulate correlated variables by using the Iman-Conover transformation”. Sein Artikel enthält eine SAS Implementierung der Iman Conover Methode:
https://blogs.sas.com/content/iml/2021/06/14/simulate-iman-conover-transformation.html

2005 veröffentlichte Stephen J. Mildenhall „Correlation and Aggregate Loss Distributions with an Emphasis on the Iman-Conover Method”:
https://www.mynl.com/old/wp/ic.pdf

Ich implementierte das Beispiel aus Mildenhall's Artikel sowohl mit Excel Tabellenblattfunktionen als auch mit Excel / VBA:
Die Eingabematrix X (rechts):
Die Zielkorrelation S:

	A	B	C	D
1	1	0,8	0,4	0
2	0,8	1	0,3	-0,2
3	0,4	0,3	1	0,1
4	0	-0,2	0,1	1

Target_Correlation_S | Cholesky_C | Intermediate_M | Covariance

Tabellenblattformeln		
Bereich	Formel	
A2;A3:B3;A4:C4	**A2**	=INDEX($A$1:$D$4;SPALTE();ZEILE())

	A	B	C	D
1	123.567	44.770	15.934	13.273
2	126.109	45.191	16.839	15.406
3	138.713	47.453	17.233	16.706
4	139.016	47.941	17.265	16.891
5	152.213	49.345	17.620	18.821
6	153.224	49.420	17.859	19.569
7	153.407	50.686	20.804	20.166
8	155.716	52.931	21.110	20.796
9	155.780	54.010	22.728	20.968
10	161.678	57.346	24.072	21.178
11	161.805	57.685	25.198	23.236
12	167.447	57.698	25.393	23.375
13	170.737	58.380	30.357	24.019
14	171.592	60.948	30.779	24.785
15	178.881	66.972	32.634	25.000
16	181.678	68.053	33.117	26.754
17	184.381	70.592	35.248	27.079
18	206.940	72.243	36.656	30.136
19	217.092	86.685	38.483	30.757
20	240.935	87.138	39.483	35.108

Input_Matrix_X | Target_Correlation_S | Cholesky_C | Inter

Die Cholesky Zerlegung C von S:

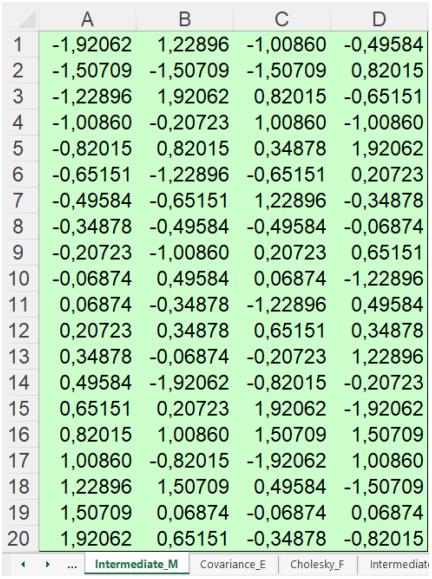
	A	B	C	D
1	1,0000	0,8000	0,4000	0,0000
2	0,0000	0,6000	-0,0333	-0,3333
3	0,0000	0,0000	0,9159	0,0970
4	0,0000	0,0000	0,0000	0,9378

Cholesky_C | Intermediate_M | Covariance_E | Cholesky_F

Tabellenblattformeln		
Bereich	Formel	
A1	**A1**	=MTRANS(Cholesky(Target_Correlation_S!A1:D4))

Die Zwischenmatrix M (konstante Werte, identisch zu Mildenhall's Daten):

Tabellenblattformeln		
Bereich	Formel	
I1	**I1**	=NORM.S.INV(SEQUENZ(20;1;1;1)/21)/STABWNA(NORM.S.INV(SEQUENZ(20;1;1;1)/21))
J1:L1	**J1**	=MTRANS(randomshuffle($A$1:$A$20))

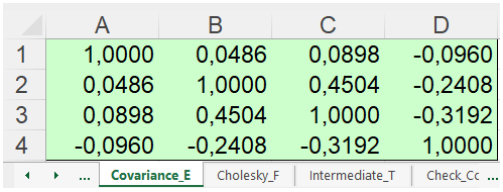


	A	B	C	D
1	-1,92062	1,22896	-1,00860	-0,49584
2	-1,50709	-1,50709	-1,50709	0,82015
3	-1,22896	1,92062	0,82015	-0,65151
4	-1,00860	-0,20723	1,00860	-1,00860
5	-0,82015	0,82015	0,34878	1,92062
6	-0,65151	-1,22896	-0,65151	0,20723
7	-0,49584	-0,65151	1,22896	-0,34878
8	-0,34878	-0,49584	-0,49584	-0,06874
9	-0,20723	-1,00860	0,20723	0,65151
10	-0,06874	0,49584	0,06874	-1,22896
11	0,06874	-0,34878	-1,22896	0,49584
12	0,20723	0,34878	0,65151	0,34878
13	0,34878	-0,06874	-0,20723	1,22896
14	0,49584	-1,92062	-0,82015	-0,20723
15	0,65151	0,20723	1,92062	-1,92062
16	0,82015	1,00860	1,50709	1,50709
17	1,00860	-0,82015	-1,92062	1,00860
18	1,22896	1,50709	0,49584	-1,50709
19	1,50709	0,06874	-0,06874	0,06874
20	1,92062	0,65151	-0,34878	-0,82015

Intermediate_M | Covariance_E | Cholesky_F | Intermediat

Sie können analoge Daten automatisch mit dieser Formel in A1:A20 erzeugen:
=NORM.S.INV(SEQUENZ(20;1;1;1)/21) / STABWNA(NORM.S.INV(SEQUENZ(20;1;1;1)/21))
und mit dieser Formel
=MTRANS(RandomShuffle($A$1:$A$20))
in den Zellen B1:B20 (kopieren Sie diese entsprechend in die Spalten C und D).

Nun erhalten Sie die Kovarianzmatrix E:

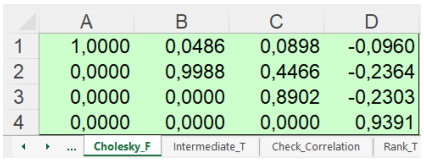


	A	B	C	D
1	1,0000	0,0486	0,0898	-0,0960
2	0,0486	1,0000	0,4504	-0,2408
3	0,0898	0,4504	1,0000	-0,3192
4	-0,0960	-0,2408	-0,3192	1,0000

Covariance_E | Cholesky_F | Intermediate_T | Check_Cc

Tabellenblattformeln		
Bereich	Formel	
A1:D4	**A1**	=KOVAR(INDEX(Intermediate_M!$A$1:$D$20;;ZEILE());INDEX(Intermediate_M!$A$1:$D$20;;SPALTE()))

Und ihre Cholesky Zerlegung F:



	A	B	C	D
1	1,0000	0,0486	0,0898	-0,0960
2	0,0000	0,9988	0,4466	-0,2364
3	0,0000	0,0000	0,8902	-0,2303
4	0,0000	0,0000	0,0000	0,9391

Cholesky_F | Intermediate_T | Check_Correlation | Rank_T

Tabellenblattformeln		
Bereich	Formel	
A1	**A1**	=MTRANS(Cholesky(Covariance_E!A1:D4))

Die Zwischenmatrix T:

Tabellenblattformeln		
Bereich	Formel	
A1	**A1**	=MMULT(MMULT(Intermediate_M!A1:D20;MINV(Cholesky_F!A1:D4));Cholesky_C!A1:D4)

	A	B	C	D
1	-1,92062	-0,74214	-2,28105	-1,33232
2	-1,50709	-2,06697	-1,30678	0,54577
3	-1,22896	0,20647	-0,51141	-0,94465
4	-1,0086	-0,9019	0,80546	-0,65873
5	-0,82015	-0,13949	-0,31782	1,7696
6	-0,65151	-1,24042	-0,28	0,23988
7	-0,49584	-0,77356	1,42145	0,23612
8	-0,34878	-0,56669	-0,38117	-0,14744
9	-0,20723	-0,76561	0,64214	0,97494
10	-0,06874	0,24487	-0,19673	-1,33695
11	0,06874	-0,15653	-1,06954	0,14014
12	0,20723	0,36925	0,56695	0,51206
13	0,34878	0,22754	-0,06362	1,1955
14	0,49584	-0,77155	0,26828	0,03168
15	0,65151	0,62666	2,08987	-1,21744
16	0,82015	1,23804	1,32493	1,8568
17	1,0086	0,28474	-1,23688	0,59246
18	1,22896	1,85259	0,17411	-1,62428
19	1,50709	1,20294	0,39517	0,13931
20	1,92062	1,87176	-0,04335	-0,97245

Cholesky_F | Intermediate_T | Check_Correlation | Rank_T

Sie können die erzeugten Korrelationen prüfen:

	A	B	C	D
1	1,0000	0,8000	0,4000	0,0000
2	0,8000	1,0000	0,3000	-0,2000
3	0,4000	0,3000	1,0000	0,1000
4	0,0000	-0,2000	0,1000	1,0000
5				
6	**Difference to Target Correlation:**			
7				
8	0	0	0	-4,44089E-17
9	0	0	0	0
10	0	0	0	0
11	-4,44089E-17	0	0	0

... | Cholesky_F | Intermediate_T | Check_Correlation | Rank_T | Result_Y | Check_Correlatic

Tabellenblattformeln		
Bereich	Formel	
A1:D4	**A1**	=KORREL(INDEX(Intermediate_T!$A$1:$D$20;;ZEILE());INDEX(Intermediate_T!$A$1:$D$20;;SPALTE()))
A8	**A8**	=A1:D4-Target_Correlation_S!A1:D4

Berechnen Sie den Rang der Zahlen in den Spalten von T in Tabellenblatt Rank_T:

Tabellenblattformeln		
Bereich	Formel	
A1:D1	**A1**	=RANG(Intermediate_T!A1:A20;Intermediate_T!A1:A20;1)

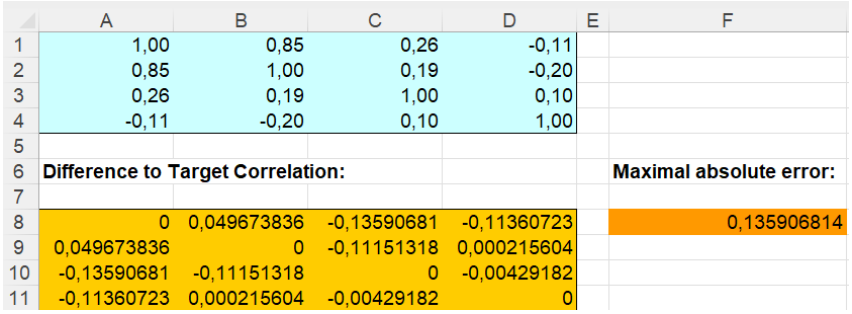
	A	B	C	D
1	1	7	1	3
2	2	1	2	15
3	3	11	5	6
4	4	3	17	7
5	5	10	7	19
6	6	2	8	13
7	7	4	19	12
8	8	8	6	8
9	9	6	16	17
10	10	13	9	2
11	11	9	4	11
12	12	15	15	14
13	13	12	10	18
14	14	5	13	9
15	15	16	20	4
16	16	18	18	20
17	17	14	3	16
18	18	19	12	1
19	19	17	14	10
20	20	20	11	5

Schließlich erhalten Sie im Tabellenblatt Result_Y:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	123.567	50.686	15.934	16.706		VBA	123.567	44.770	17.265	24.785		Worksheet formulae	123.567	50.686	15.934	16.706
2	126.109	44.770	16.839	25.000			126.109	45.191	33.117	24.019			126.109	44.770	16.839	25.000
3	138.713	57.685	17.620	19.569			138.713	50.686	17.233	13.273			138.713	57.685	17.620	19.569
4	139.016	47.453	35.248	20.166			139.016	57.685	20.804	30.136			139.016	47.453	35.248	20.166
5	152.213	57.346	20.804	30.757			152.213	57.346	30.357	21.178			152.213	57.346	20.804	30.757
6	153.224	45.191	21.110	24.019			153.224	49.420	16.839	20.968			153.224	45.191	21.110	24.019
7	153.407	47.941	38.483	23.375			153.407	47.941	15.934	26.754			153.407	47.941	38.483	23.375
8	155.716	52.931	17.859	20.796			155.716	47.453	22.728	19.569			155.716	52.931	17.859	20.796
9	155.780	49.420	33.117	27.079			155.780	52.931	25.393	35.108			155.780	49.420	33.117	27.079
10	161.678	58.380	22.728	15.406			161.678	49.345	25.198	23.236			161.678	58.380	22.728	15.406
11	161.805	54.010	17.265	23.236			161.805	86.685	36.656	15.406			161.805	54.010	17.265	23.236
12	167.447	66.972	32.634	24.785			167.447	66.972	30.779	16.706			167.447	66.972	32.634	24.785
13	170.737	57.698	24.072	30.136			170.737	54.010	35.248	20.796			170.737	57.698	24.072	30.136
14	171.592	49.345	30.357	20.968			171.592	68.053	17.859	18.821			171.592	49.345	30.357	20.968
15	178.881	68.053	39.483	16.891			178.881	57.698	38.483	27.079			178.881	68.053	39.483	16.891
16	181.678	72.243	36.656	35.108			181.678	70.592	17.620	16.891			181.678	72.243	36.656	35.108
17	184.381	60.948	17.233	26.754			184.381	58.380	24.072	20.166			184.381	60.948	17.233	26.754
18	206.940	86.685	25.393	13.273			206.940	72.243	32.634	30.757			206.940	86.685	25.393	13.273
19	217.092	70.592	30.779	21.178			217.092	60.948	39.483	23.375			217.092	70.592	30.779	21.178
20	240.935	87.138	25.198	18.821			240.935	87.138	21.110	25.000			240.935	87.138	25.198	18.821

Tabellenblattformeln		
Bereich	Formel	
A1:D1;M1:P1	**A1**	=INDEX(Input_Matrix_X!$A$1:$A$20;Rank_T!$A$1:$A$20)
G1	**G1**	=ImanConover(Input_Matrix_X!$A$1:$D$20;Target_Correlation_S!$A$1:$D$4)

Sie können die Differenz zur Zielkorrelation im Tabellenblatt Check_Correlation_Result prüfen:



	A	B	C	D	E	F
1	1,00	0,85	0,26	-0,11		
2	0,85	1,00	0,19	-0,20		
3	0,26	0,19	1,00	0,10		
4	-0,11	-0,20	0,10	1,00		
5						
6	**Difference to Target Correlation:**					**Maximal absolute error:**
7						
8	0	0,049673836	-0,13590681	-0,11360723		0,135906814
9	0,049673836	0	-0,11151318	0,000215604		
10	-0,13590681	-0,11151318	0	-0,00429182		
11	-0,11360723	0,000215604	-0,00429182	0		

Tabellenblattformeln		
Bereich	Formel	
A1:D4	**A1**	=KORREL(INDEX(Result_Y!$A$1:$D$20;;ZEILE());INDEX(Result_Y!$A$1:$D$20;;SPALTE()))
A8	**A8**	=A1:D4-Target_Correlation_S!A1:D4
F8	**F8**	=MAX(ABS(A8:D11))

RandomShuffle Programmcode

```
Function RandomShuffle(vtemp As Variant) As Variant
Dim j As Long, k As Long, n As Long
Dim temp As Double, u As Double
'Application.Volatile 'Uncomment if you think you need this.
With Application.WorksheetFunction
On Error Resume Next 'Ignore error: VBA calls already with 1-dim array.
vtemp = .Transpose(vtemp)
On Error GoTo 0
n = UBound(vtemp)
j = n
Do While j > 0
  u = Rnd()
  k = Int(j * u + 1)
  temp = vtemp(j)
  vtemp(j) = vtemp(k)
  vtemp(k) = temp
  j = j - 1
Loop
RandomShuffle = vtemp
End With
End Function
```

IndexX Programmcode

```
Function IndexX(n As Long, arr As Variant, colNo As Long) As Variant
'Indexes an array arr[1..n], i.e., outputs the array indx[1..n] such
'that arr[indx[j]] is in ascending order for j = 1, 2, . . . ,n. The
'input quantities n and arr are not changed. Translated from [31].
Const m As Long = 7
Const NSTACK As Long = 50
Dim i As Long, indxt As Long, ir As Long, itemp As Long, j As Long, k As Long, l As Long
Dim jstack As Long, istack(1 To NSTACK) As Long, a As Double

ir = n: l = 1
ReDim indx(1 To n) As Long
For j = 1 To n: indx(j) = j: Next j

Do While 1
  If (ir - l < m) Then
    For j = l + 1 To ir
      indxt = indx(j)
      a = arr(indxt, colNo)
      For i = j - 1 To l Step -1
        If (arr(indx(i), colNo) <= a) Then Exit For
        indx(i + 1) = indx(i)
      Next i
      indx(i + 1) = indxt
    Next j
    If (jstack = 0) Then Exit Do
    ir = istack(jstack)
    jstack = jstack - 1
    l = istack(jstack)
    jstack = jstack - 1
  Else
    k = (l + ir) / 2
    itemp = indx(k)
    indx(k) = indx(l + 1)
    indx(l + 1) = itemp
    If (arr(indx(l), colNo) > arr(indx(ir), colNo)) Then
      itemp = indx(l)
      indx(l) = indx(ir)
      indx(ir) = itemp
    End If
    If (arr(indx(l + 1), colNo) > arr(indx(ir), colNo)) Then
      itemp = indx(l + 1)
      indx(l + 1) = indx(ir)
      indx(ir) = itemp
    End If
    If (arr(indx(l), colNo) > arr(indx(l + 1), colNo)) Then
      itemp = indx(l)
      indx(l) = indx(l + 1)
      indx(l + 1) = itemp
    End If
    i = l + 1
    j = ir
    indxt = indx(l + 1)
    a = arr(indxt, colNo)
    Do While 1
      Do
        i = i + 1
      Loop While (arr(indx(i), colNo) < a)
      Do
        j = j - 1
      Loop While (arr(indx(j), colNo) > a)
      If (j < i) Then Exit Do
      itemp = indx(i)
      indx(i) = indx(j)
      indx(j) = itemp
    Loop
    indx(l + 1) = indx(j)
    indx(j) = indxt
    jstack = jstack + 2
    If (jstack > NSTACK) Then
      'STACK too small in indexx
      IndexX = CVErr(xlErrNum)
      Exit Function
    End If
    If (ir - i + 1 >= j - l) Then
      istack(jstack) = ir
      istack(jstack - 1) = i
      ir = j - 1
    Else
      istack(jstack) = j - 1
      istack(jstack - 1) = l
      l = i
    End If
  End If
Loop
IndexX = indx
End Function
```

ImanConover Programmcode

Dies ist die VBA Implementierung der Iman Conover Methode. Ich habe diesen Code auch in der Anwendung *sbGenerateTestData* (Testdaten erzeugen – *sbGenerateTestData*) verwendet.

Bitte beachten Sie, dass die Funktion ImanConover neben den oben genannten Funktionen IndexX und RandomShuffle auch die Funktion *Cholesky* (*Cholesky* Zerlegung) benötigt (aufruft).

```
Function ImanConover(rInputMatrix As Range, _
        rTargetCorrelation As Range) As Variant
'Implements the Iman-Conover method to generate random
'number vectors with a given correlation.
'Algorithm as described in:
'Mildenham, November 27, 2005
'Correlation and Aggregate Loss Distributions With An
'Emphasis On The Iman-Conover Method
'V0.3 PB 02-Nov-2024 by Bernd Plumhoff
Dim vX As Variant   'Input matrix
Dim vS As Variant   'Target correlation matrix
Dim vC As Variant   'Cholesky decomposition of vS
Dim vM As Variant   'Intermediate matrix M
Dim vE As Variant   'Covariance matrix E
Dim vF As Variant   'Cholesky decomposition of vE
Dim vT As Variant   'Intermediate matrix T
Dim d As Double, dS As Double
Dim i As Long, j As Long, k As Long
Dim lRow As Long, lCol As Long
Dim state As SystemState

With Application
Set state = New SystemState
vX = .Transpose(.Transpose(rInputMatrix))
lRow = rInputMatrix.Rows.Count
lCol = rInputMatrix.Columns.Count

'################################################################################
'#                                Check inputs                                  #
'################################################################################

If lCol <> rTargetCorrelation.Columns.Count _
  And rTargetCorrelation.Rows.Count <> rTargetCorrelation.Columns.Count Then
  'Structure of target correlation matrix needs to fit input matrix
  ImanConover = CVErr(xlErrNum)
  Exit Function
End If
vS = .Transpose(.Transpose(rTargetCorrelation))
For i = 1 To lCol
  If vS(i, i) <> 1# Then
    'Target correlation matrix not 1 on diagonal
    ImanConover = CVErr(xlErrValue)
    Exit Function
  End If
  For j = 1 To i - 1
    If vS(i, j) <> vS(j, i) Then
      'Target correlation matrix not symmetric
      ImanConover = CVErr(xlErrValue)
      Exit Function
    End If
  Next j
Next i

vC = .Transpose(Cholesky(vS))

'################################################################################
'#                          Create intermediate matrix M                        #
'################################################################################

ReDim vMV(1 To lRow) As Double
d = 0#
dS = 0#
For i = 1 To Int(lRow / 2)
  vMV(i) = .NormSInv(i / (lRow + 1))
  vMV(lRow - i + 1) = -vMV(i)
  d = d + 2# * vMV(i) * vMV(i)
Next i
If lRow Mod 2 = 1 Then vMV((lRow + 1) / 2) = 0 'Just for clarity, it's already 0
d = Sqr(d / lRow)
For i = 1 To lRow
  vMV(i) = vMV(i) / d
Next i

vM = vX
For i = 1 To lRow
```

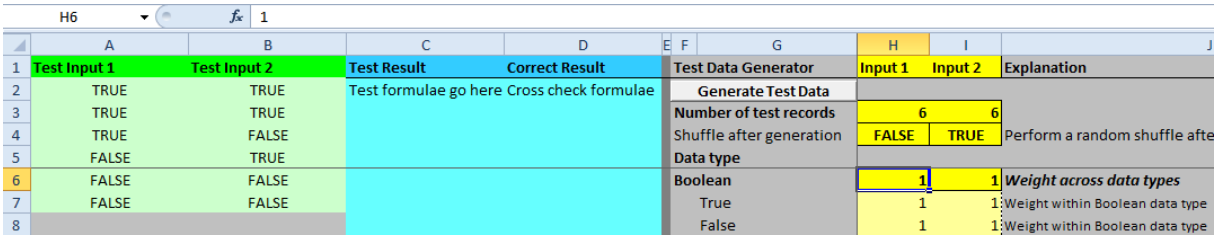
```
  vM(i, 1) = vMV(i)
Next i

Dim vMW As Variant
For i = 2 To lCol
  vMW = RandomShuffle(vMV)
  For j = 1 To lRow
    vM(j, i) = vMW(j)
  Next j
Next i

'###############################################################################
'#                          Calculate covariance matrix E                      #
'###############################################################################

vE = vC
For i = 1 To lCol
  vE(i, i) = .Covar(.Index(.Transpose(vM), i), .Index(.Transpose(vM), i))
  For j = i + 1 To lCol
    vE(i, j) = .Covar(.Index(.Transpose(vM), i), .Index(.Transpose(vM), j))
    vE(j, i) = vE(i, j)
  Next j
Next i

vF = .Transpose(Cholesky(vE))

vT = .MMult(.MMult(vM, .MInverse(vF)), vC)

'###############################################################################
'#                          Compute ranks of matrix T                          #
'###############################################################################

Dim vRT As Variant, vR As Variant
vRT = vX
For j = 1 To lCol
  vR = IndexX(lRow, vT, j)
  For i = 1 To lRow
    vRT(i, j) = vR(i)
  Next i
  vR = IndexX(lRow, vX, j)
  For i = 1 To lRow
    vX(i, j) = vX(vR(i), j)
  Next i
Next j

'###############################################################################
'#                          Calculate result matrix Y                          #
'###############################################################################

Dim vY As Variant
vY = vX
For i = 1 To lRow
  For j = 1 To lCol
    vY(i, j) = vX(vRT(i, j), j)
  Next j
Next i

ImanConover = vY
End With
End Function
```

Praktische Anwendungen allgemeiner Zufallszahlen

Testdaten erzeugen – *sbGenerateTestData*

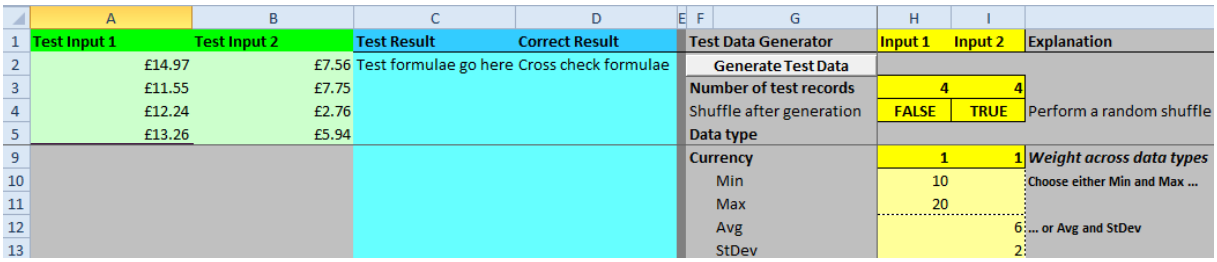
Wenn Sie eine Anwendung oder ein Programm auf Herz und Nieren testen wollen, benötigen Sie häufig Testdaten. Die Anwendung *sbGenerateTestData* soll Sie dabei unterstützen, zufällige Testdaten in numerischer Form oder als Text zu erzeugen.

Wollen Sie beispielsweise sechs Wahrheitswerte, davon 50% WAHR und 50% FALSCH, einmal in der erzeugten Reihenfolge und einmal zufällig gemischt:



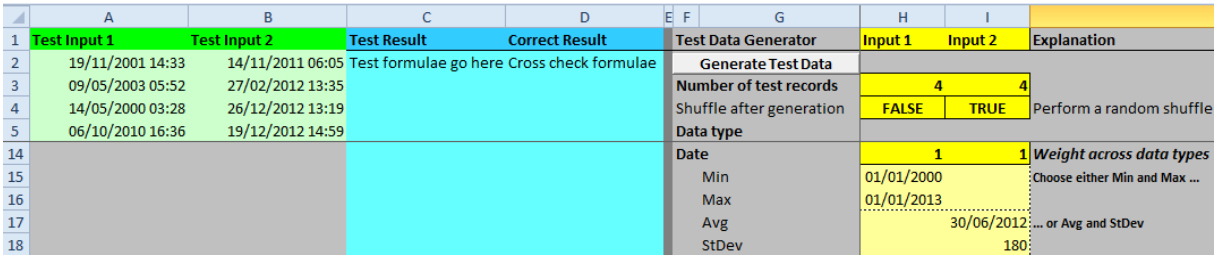
	Test Input 1	Test Input 2	Test Result	Correct Result	Test Data Generator	Input 1	Input 2	Explanation
2	TRUE	TRUE	Test formulae go here	Cross check formulae	Generate Test Data			
3	TRUE	TRUE			Number of test records	6	6	
4	TRUE	FALSE			Shuffle after generation	FALSE	TRUE	Perform a random shuffle afte
5	FALSE	TRUE			Data type			
6	FALSE	FALSE			Boolean	1	1	*Weight across data types*
7	FALSE	FALSE			True	1	1	Weight within Boolean data type
8					False	1	1	Weight within Boolean data type

Oder Sie benötigen 4 Geldbeträge in britischen Pfund Sterling (GBP), die erste Serie zwischen 10 GBP und 20 GBP und die zweite mit einem Durchschnitsswert von 6 GBP und einer Standardabweichung von 2 GBP:



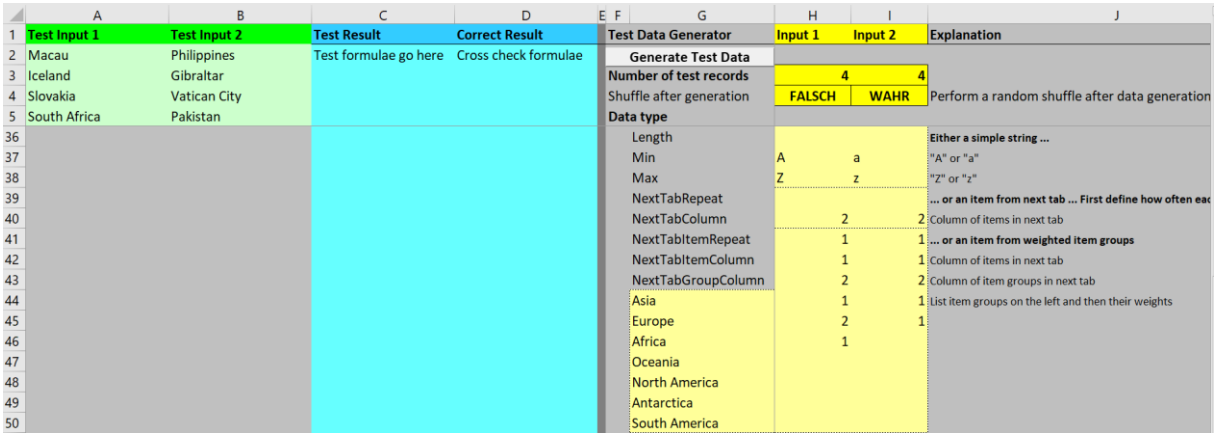
	Test Input 1	Test Input 2	Test Result	Correct Result	Test Data Generator	Input 1	Input 2	Explanation
2	£14.97	£7.56	Test formulae go here	Cross check formulae	Generate Test Data			
3	£11.55	£7.75			Number of test records	4	4	
4	£12.24	£2.76			Shuffle after generation	FALSE	TRUE	Perform a random shuffle
5	£13.26	£5.94			Data type			
9					Currency	1	1	*Weight across data types*
10					Min	10		Choose either Min and Max ...
11					Max	20		
12					Avg		6	... or Avg and StDev
13					StDev		2	

Falls Sie vier Daten zwischen 1-Jan-2000 und 1-Jan-2013 oder vier Daten mit dem Durschschnittswert 30-Jun-2012 und einer Standardabweichung von 180 Tagen benötigen:



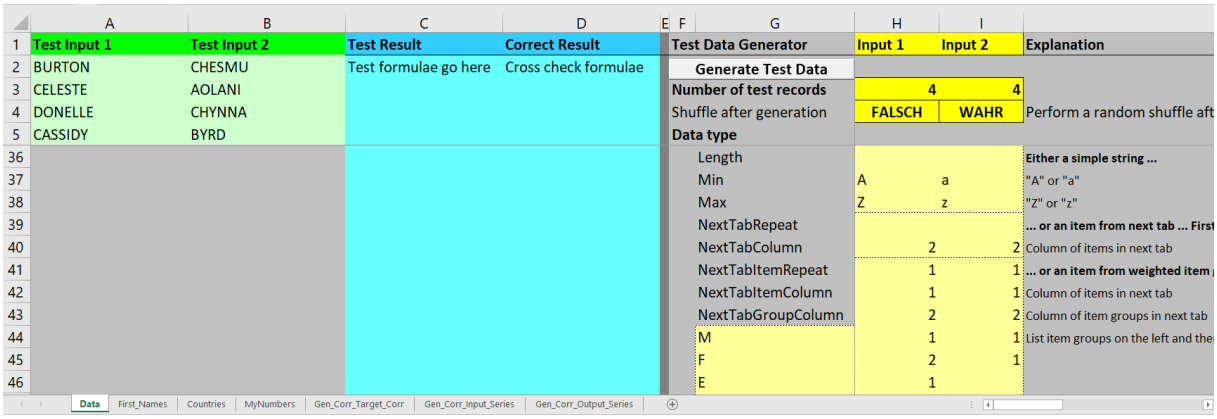
	Test Input 1	Test Input 2	Test Result	Correct Result	Test Data Generator	Input 1	Input 2	Explanation
2	19/11/2001 14:33	14/11/2011 06:05	Test formulae go here	Cross check formulae	Generate Test Data			
3	09/05/2003 05:52	27/02/2012 13:35			Number of test records	4	4	
4	14/05/2000 03:28	26/12/2012 13:19			Shuffle after generation	FALSE	TRUE	Perform a random shuffle
5	06/10/2010 16:36	19/12/2012 14:59			Data type			
14					Date	1	1	*Weight across data types*
15					Min	01/01/2000		Choose either Min and Max ...
16					Max	01/01/2013		
17					Avg		30/06/2012	... or Avg and StDev
18					StDev		180	

Wenn Sie 4 Ländernamen erzeugen wollen, davon einen aus Afrika, einen aus Asien, und zwei aus Europa; oder Sie brauchen 2 asiatische und 2 europäische Ländernamen (ziehen Sie das Tabellenblatt „Countries" gleich rechts neben das Tabellenblatt „Data" so dass es Blatt 2 ist:



	A	B	C	D	E	F	G	H	I	J
1	Test Input 1	Test Input 2	Test Result	Correct Result			Test Data Generator	Input 1	Input 2	Explanation
2	Macau	Philippines	Test formulae go here	Cross check formulae			Generate Test Data			
3	Iceland	Gibraltar					Number of test records	4	4	
4	Slovakia	Vatican City					Shuffle after generation	FALSCH	WAHR	Perform a random shuffle after data generation
5	South Africa	Pakistan					Data type			
36							Length			Either a simple string ...
37							Min	A	a	"A" or "a"
38							Max	Z	z	"Z" or "z"
39							NextTabRepeat			... or an item from next tab ... First define how often ea
40							NextTabColumn	2	2	Column of items in next tab
41							NextTabItemRepeat	1	1	... or an item from weighted item groups
42							NextTabItemColumn	1	1	Column of items in next tab
43							NextTabGroupColumn	2	2	Column of item groups in next tab
44							Asia	1	1	List item groups on the left and then their weights
45							Europe	2	1	
46							Africa	1		
47							Oceania			
48							North America			
49							Antarctica			
50							South America			

Falls Sie Vornamen zufällig aus einer gegebenen Liste ziehen wollen, ziehen SIe das Tabellenblatt „First_Names“ rechts neben das „Data“ Blatt. Sie erhalten nach erneutem Drücken des Knopfes „Generate Test Data“ eine Warnung. Drücken Sie dann „Ok“:



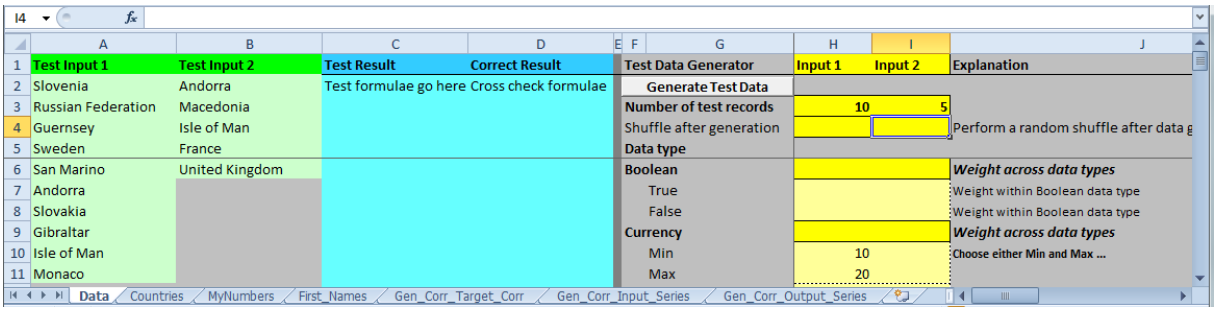
	A	B	C	D	E	F	G	H	I	J
1	Test Input 1	Test Input 2	Test Result	Correct Result			Test Data Generator	Input 1	Input 2	Explanation
2	BURTON	CHESMU	Test formulae go here	Cross check formulae			Generate Test Data			
3	CELESTE	AOLANI					Number of test records	4	4	
4	DONELLE	CHYNNA					Shuffle after generation	FALSCH	WAHR	Perform a random shuffle aft
5	CASSIDY	BYRD					Data type			
36							Length			Either a simple string ...
37							Min	A	a	"A" or "a"
38							Max	Z	z	"Z" or "z"
39							NextTabRepeat			... or an item from next tab ... First
40							NextTabColumn	2	2	Column of items in next tab
41							NextTabItemRepeat	1	1	... or an item from weighted item
42							NextTabItemColumn	1	1	Column of items in next tab
43							NextTabGroupColumn	2	2	Column of item groups in next tab
44							M	1	1	List item groups on the left and the
45							F	2	1	
46							E	1		

Data | First_Names | Countries | MyNumbers | Gen_Corr_Target_Corr | Gen_Corr_Input_Series | Gen_Corr_Output_Series

Bemerkung: Die Spalten für die Listenelemente und deren Gruppen sind hier nicht zufällig identisch. Dies wurde so absichtlich gestaltet, damit man durch Ziehen des entsprechenden Tabellenblatts neben das „Data“ Blatt die gewünschten Werte ändern kann, entweder zu Vornamen oder zu Ländernamen.

Sie können mit dieser Anwendung auch korrelierte Pseudozufallszahlen erzeugen. Ich implementierte die *Iman-Conover* Methode mit VBA.

Die Tabellenblätter:



	A	B	C	D	E	F	G	H	I	J
1	Test Input 1	Test Input 2	Test Result	Correct Result			Test Data Generator	Input 1	Input 2	Explanation
2	Slovenia	Andorra	Test formulae go here	Cross check formulae			Generate Test Data			
3	Russian Federation	Macedonia					Number of test records	10	5	
4	Guernsey	Isle of Man					Shuffle after generation			Perform a random shuffle after data g
5	Sweden	France					Data type			
6	San Marino	United Kingdom					Boolean			*Weight across data types*
7	Andorra						True			Weight within Boolean data type
8	Slovakia						False			Weight within Boolean data type
9	Gibraltar						Currency			*Weight across data types*
10	Isle of Man						Min	10		Choose either Min and Max ...
11	Monaco						Max	20		

Data | Countries | MyNumbers | First_Names | Gen_Corr_Target_Corr | Gen_Corr_Input_Series | Gen_Corr_Output_Series

Tabellenblatt ‚Countries':
Quelle: https://raw.github.com/wikimedia/limn-data/master/geo/country-codes.csv

	A	B
1	**Country Name**	**Continent Name**
2	Afghanistan	Asia
3	Åland	Europe
4	Albania	Europe
5	Algeria	Africa
6	American Samoa	Oceania
7	Andorra	Europe
8	Angola	Africa
9	Anguilla	North America
10	Antarctica	Antarctica
11	Antigua and Barbuda	North America
12	Argentina	South America
13	Armenia	Asia
14	Aruba	North America
15	Australia	Oceania
16	Austria	Europe
17	Azerbaijan	Asia

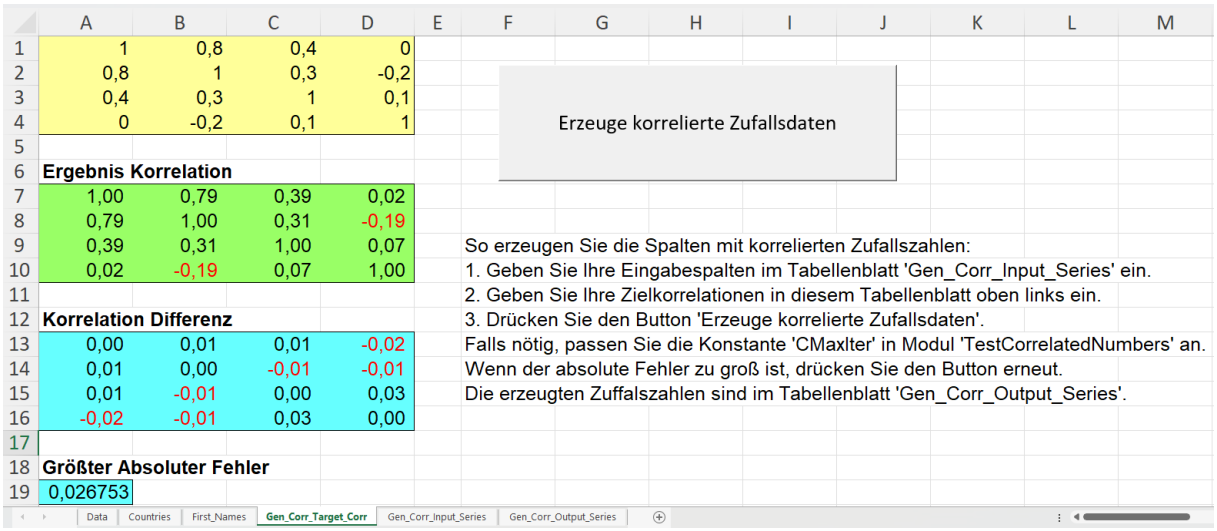
Data | **Countries** | First_Names | Gen_Corr_T

Tabellenblatt ‚First_Names':
Quellen: https://www2.gov.bc.ca/assets/gov/birth-adoption-death-marriage-and-divorce/statistics-reports/baby-names-trends-m-2023.csv
https://www2.gov.bc.ca/assets/gov/birth-adoption-death-marriage-and-divorce/statistics-reports/baby-names-trends-m-2023.csv

	A	B
1	**Name**	**Gender**
2	AADHYA	F
3	AADYA	F
4	AAHANA	F
5	AALIYAH	F
6	AANYA	F
7	AARNA	F
8	AARYA	F
9	AARZA	F
10	AASHVI	F
11	ABBEY	F
12	ABBIE	F
13	ABBIGAIL	F
14	ABBY	F
15	ABBYGAIL	F
16	ABIGAIL	F
17	ABIGALE	F

First_Names

Korrelierte Zufallszahlen erzeugen Sie mit dieser Anwendung wie folgt:



	A	B	C	D
1	1	0,8	0,4	0
2	0,8	1	0,3	-0,2
3	0,4	0,3	1	0,1
4	0	-0,2	0,1	1
5				
6	**Ergebnis Korrelation**			
7	1,00	0,79	0,39	0,02
8	0,79	1,00	0,31	-0,19
9	0,39	0,31	1,00	0,07
10	0,02	-0,19	0,07	1,00
11				
12	**Korrelation Differenz**			
13	0,00	0,01	0,01	-0,02
14	0,01	0,00	-0,01	-0,01
15	0,01	-0,01	0,00	0,03
16	-0,02	-0,01	0,03	0,00
17				
18	**Größter Absoluter Fehler**			
19	0,026753			

Tabellenblattformeln		
Bereich	Formel	
A7:D10	**A7**	=KORREL(INDEX(Gen_Corr_Output_Series!$A$1:$D$20;;ZEILE()-6);INDEX(Gen_Corr_Output_Series!$A$1:$D$20;;SPALTE()))
A13	**A13**	=A1:D4-A7:D10
A19	**A19**	=MAX(ABS(A13:D16))

	A	B	C	D
1	123.567	44.770	15.934	13.273
2	126.109	45.191	16.839	15.406
3	138.713	47.453	17.233	16.706
4	139.016	47.941	17.265	16.891
5	152.213	49.345	17.620	18.821
6	153.224	49.420	17.859	19.569
7	153.407	50.686	20.804	20.166
8	155.716	52.931	21.110	20.796
9	155.780	54.010	22.728	20.968
10	161.678	57.346	24.072	21.178
11	161.805	57.685	25.198	23.236
12	167.447	57.698	25.393	23.375
13	170.737	58.380	30.357	24.019
14	171.592	60.948	30.779	24.785
15	178.881	66.972	32.634	25.000
16	181.678	68.053	33.117	26.754
17	184.381	70.592	35.248	27.079
18	206.940	72.243	36.656	30.136
19	217.092	86.685	38.483	30.757
20	240.935	87.138	39.483	35.108

Gen_Corr_Input_Series | Gen_Corr_Output_Series

	A	B	C	D
1	123.567	44.770	17.859	20.796
2	126.109	45.191	15.934	23.375
3	138.713	50.686	17.620	20.968
4	139.016	47.453	17.233	15.406
5	152.213	49.345	35.248	30.757
6	153.224	66.972	25.198	24.019
7	153.407	49.420	17.265	19.569
8	155.716	52.931	38.483	23.236
9	155.780	57.685	25.393	21.178
10	161.678	60.948	21.110	35.108
11	161.805	57.698	36.656	16.706
12	167.447	57.346	16.839	18.821
13	170.737	47.941	39.483	30.136
14	171.592	58.380	20.804	26.754
15	178.881	87.138	32.634	16.891
16	181.678	54.010	24.072	27.079
17	184.381	68.053	33.117	13.273
18	206.940	72.243	22.728	25.000
19	217.092	70.592	30.357	24.785
20	240.935	86.685	30.779	20.166

Gen_Corr_Input_Series | Gen_Corr_Output_Series

sbGenerateTestData Programmcode

```
Enum types
  ty_start = 0 'So that we can iterate from ty_start + 1 to ty_end - 1
  ty_boolean
  ty_currency
  ty_date
  ty_decimal
  ty_double
  ty_long
  ty_string
  ty_end  'So that we can iterate from ty_start + 1 to ty_end - 1
End Enum 'types

Enum param_rows
  pr_records = 3
  pr_shuffle
  pr_Boolean = 6
    pr_bTrue
    pr_bFalse
  pr_Currency
    pr_ccyMin
    pr_ccyMax
    pr_ccyAvg
    pr_ccyStDev
  pr_Date
    pr_dtMin
    pr_dtMax
    pr_dtAvg
    pr_dtStDev
  pr_Decimal
    pr_decMin
    pr_decMax
    pr_decAvg
    pr_decStDev
  pr_Double
    pr_dMin
    pr_dMax
    pr_dAvg
    pr_dStDev
  pr_Long
    pr_lSum
    pr_lMin1
    pr_lMin2
    pr_lMax
    pr_lMaxRepeat
  pr_String
    pr_sLength
    pr_sMin
    pr_sMax
    pr_sNextTabRepeat
    pr_sNextTabColumn
    pr_sNextTabItemRepeat
    pr_sNextTabItemColumn
    pr_sNextTabGroupColumn
    pr_sNextTabGroupWeights 'Item group weights start from here and can go down any number
End Enum 'param_rows

Enum param_columns
  pc_Output1 = 1
  pc_Output2
  pc_ItemGroups = 7
  pc_Input1 = 8
  pc_Input2
End Enum 'param_columns

Private Enum xlCI 'Excel Color Index
: xlCIBlack = 1: xlCIWhite: xlCIRed: xlCIBrightGreen: xlCIBlue '1 - 5
: xlCIYellow: xlCIPink: xlCITurquoise: xlCIDarkRed: xlCIGreen '6 - 10
: xlCIDarkBlue: xlCIDarkYellow: xlCIViolet: xlCITeal: xlCIGray25 '11 - 15
: xlCIGray50: xlCIPeriwinkle: xlCIPlum: xlCIIvory: xlCILightTurquoise '16 - 20
: xlCIDarkPurple: xlCICoral: xlCIOceanBlue: xlCIIceBlue: xlCILightBrown '21 - 25
: xlCIMagenta2: xlCIYellow2: xlCICyan2: xlCIDarkPink: xlCIDarkBrown '26 - 30
: xlCIDarkTurquoise: xlCISeaBlue: xlCISkyBlue: xlCILightTurquoise2: xlCILightGreen '31 - 35
: xlCILightYellow: xlCIPaleBlue: xlCIRose: xlCILavender: xlCITan '36 - 40
: xlCILightBlue: xlCIAqua: xlCILime: xlCIGold: xlCILightOrange '41 - 45
: xlCIOrange: xlCIBlueGray: xlCIGray40: xlCIDarkTeal: xlCISeaGreen '46 - 50
: xlCIDarkGreen: xlCIGreenBrown: xlCIBrown: xlCIDarkPink2: xlCIIndigo '51 - 55
: xlCIGray80 '56
End Enum

Sub sbGenerateTestData()
'Randomly generate test data as specified in input area.
'Bernd Plumhoff 06-Apr-2021 PB V0.2

Dim bGroupsUpToDate As Boolean
Dim dAvg As Double
Dim dmax As Double
```

```
Dim dmin As Double
Dim dStDev As Double
Dim dSumWeights As Double
ReDim dTypeWeight(ty_start + 1 To ty_end - 1) As Double
ReDim sTypeName(ty_start + 1 To ty_end - 1) As String
Dim i As Long
Dim j As Long
Dim k As Long
Dim lCol As Long
Dim lLength As Long
Dim lRecord As Long
Dim lRow As Long
Dim lIdx As Long
Dim lTypeSum As Long
Dim objItem As Object
Dim objGroup As Object
Dim s As String
Dim sErrMsg As String
Dim v As Variant
Dim vThisType As Variant
Dim vType As Variant
Dim vGroup As Variant
Dim wsItem As Worksheet
Dim state As SystemState

Set state = New SystemState
Randomize

With Application.WorksheetFunction

'Clear input
wsD.Range("A:A").Offset(, pc_Output1 - 1).ClearContents
wsD.Range("A:A").Offset(, pc_Output2 - 1).ClearContents
wsD.Range("A:A").Offset(, pc_Output1 - 1).ClearFormats
wsD.Range("A:A").Offset(, pc_Output2 - 1).ClearFormats
wsD.Range("A:A").Offset(, pc_Output1 - 1).Interior.ColorIndex = xlCIGray25
wsD.Range("A:A").Offset(, pc_Output2 - 1).Interior.ColorIndex = xlCIGray25
With wsD.Range("A1").Offset(, pc_Output1 - 1)
  .Formula = "Test Input 1"
  .Font.Bold = True
  .Interior.ColorIndex = xlCIBrightGreen
End With
With wsD.Range("A1").Offset(, pc_Output2 - 1)
  .Formula = "Test Input 2"
  .Font.Bold = True
  .Interior.ColorIndex = xlCIBrightGreen
End With

sTypeName(ty_boolean) = "Boolean"
sTypeName(ty_currency) = "Currency"
sTypeName(ty_date) = "Date"
sTypeName(ty_decimal) = "Decimal"
sTypeName(ty_double) = "Double"
sTypeName(ty_long) = "Long"
sTypeName(ty_string) = "String"

For lCol = pc_Input1 To pc_Input2
  sErrMsg = ""
  lRecord = wsD.Cells(pr_records, lCol)
  If lRecord <= 0 Then
    Call MsgBox("Number of test records must be greater zero!" & vbCrLf, vbOKOnly, "Error")
    Exit Sub
  End If
  wsD.Cells(2, lCol - pc_Input1 + pc_Output1).Resize(lRecord).Interior.ColorIndex = xlCILightGreen
  ReDim vInput(1 To lRecord) As Variant
  lIdx = 1
  dTypeWeight(ty_boolean) = wsD.Cells(pr_Boolean, lCol)
  dTypeWeight(ty_currency) = wsD.Cells(pr_Currency, lCol)
  dTypeWeight(ty_date) = wsD.Cells(pr_Date, lCol)
  dTypeWeight(ty_decimal) = wsD.Cells(pr_Decimal, lCol)
  dTypeWeight(ty_double) = wsD.Cells(pr_Double, lCol)
  dTypeWeight(ty_long) = wsD.Cells(pr_Long, lCol)
  dTypeWeight(ty_string) = wsD.Cells(pr_String, lCol)
  dSumWeights = 0#
  For i = LBound(dTypeWeight) To UBound(dTypeWeight)
    If dTypeWeight(i) < 0 Then sErrMsg = sErrMsg & _
      "Weight for data type " & sTypeName(i) & " must be greater equal zero!" & vbCrLf
    dSumWeights = dSumWeights + dTypeWeight(i)
  Next i
  If dSumWeights <= 0 Then sErrMsg = sErrMsg & _
    "Sum of weights for data types (Boolean, ..., String) must be greater zero!" & vbCrLf

  If Len(sErrMsg) > 0 Then
    Call MsgBox(sErrMsg & vbCrLf, vbOKOnly, "Error")
    Exit Sub
  End If
  For i = LBound(dTypeWeight) To UBound(dTypeWeight)
    dTypeWeight(i) = dTypeWeight(i) / dSumWeights * lRecord
  Next i
  'Decide how many records to generate for each data type
```

```
vType = RoundToSum(dTypeWeight, 0)

For i = LBound(vType, 1) To UBound(vType, 1)
  If vType(i) > 0 Then
    Select Case i
    Case ty_boolean
      ReDim dThisTypeWeight(1 To 2) As Double
      If Abs(wsD.Cells(pr_bTrue, lCol) + wsD.Cells(pr_bFalse, lCol)) < 0.0000000000001 Then
        'No weights means equal weights
        dThisTypeWeight(1) = vType(i) / 2
        dThisTypeWeight(2) = dThisTypeWeight(1)
      Else
        dThisTypeWeight(1) = wsD.Cells(pr_bTrue, lCol) / _
                             (wsD.Cells(pr_bTrue, lCol) + _
                             wsD.Cells(pr_bFalse, lCol)) * _
                             vType(i)
        dThisTypeWeight(2) = wsD.Cells(pr_bFalse, lCol) / _
                             (wsD.Cells(pr_bFalse, lCol) + _
                             wsD.Cells(pr_bTrue, lCol)) * _
                             vType(i)
      End If
      vThisType = RoundToSum(dThisTypeWeight, 0)
      For j = 1 To vThisType(1)
        vInput(lIdx) = True
        lIdx = lIdx + 1
      Next j
      For j = 1 To vThisType(2)
        vInput(lIdx) = False
        lIdx = lIdx + 1
      Next j
    Case ty_currency
      If IsEmpty(wsD.Cells(pr_ccyAvg, lCol)) Or IsEmpty(wsD.Cells(pr_ccyStDev, lCol)) Then
        'Work with Min and Max
        dmin = wsD.Cells(pr_ccyMin, lCol)
        dmax = wsD.Cells(pr_ccyMax, lCol)
        For j = 1 To vType(i)
          vInput(lIdx) = CCur(dmin + Rnd() * (dmax - dmin))
          lIdx = lIdx + 1
        Next j
      Else
        'Work with Avg and StDev
        ReDim dThisDouble(1 To vType(i)) As Double
        For j = 1 To vType(i)
          dThisDouble(j) = Rnd()
        Next j
        dAvg = .Average(dThisDouble)
        dStDev = .StDevP(dThisDouble)
        If dStDev < 0.0000000000001 Then
          If vType(i) = 1 Then
            vInput(lIdx) = CCur(dAvg)
            lIdx = lIdx + 1
          Else
            Call MsgBox("StDev of data type " & sTypeName(ty_currency) & _
                " must not be zero!", vbOKOnly, "Error!")
            Exit Sub
          End If
        End If
        For j = 1 To vType(i)
          vInput(lIdx) = CCur(wsD.Cells(pr_ccyAvg, lCol) + _
                         (dThisDouble(j) - dAvg) * _
                         wsD.Cells(pr_ccyStDev, lCol) / dStDev)
          lIdx = lIdx + 1
        Next j
      End If
    Case ty_date
      If IsEmpty(wsD.Cells(pr_dtAvg, lCol)) Or IsEmpty(wsD.Cells(pr_dtStDev, lCol)) Then
        'Work with Min and Max
        dmin = wsD.Cells(pr_dtMin, lCol)
        dmax = wsD.Cells(pr_dtMax, lCol)
        For j = 1 To vType(i)
          vInput(lIdx) = CDate(dmin + Rnd() * (dmax - dmin))
          lIdx = lIdx + 1
        Next j
      Else
        'Work with Avg and StDev
        ReDim dThisDouble(1 To vType(i)) As Double
        For j = 1 To vType(i)
          dThisDouble(j) = Rnd()
        Next j
        dAvg = .Average(dThisDouble)
        dStDev = .StDevP(dThisDouble)
        If dStDev < 0.0000000000001 Then
          If vType(i) = 1 Then
            vInput(lIdx) = CDate(dAvg)
            lIdx = lIdx + 1
          Else
            Call MsgBox("StDev of data type " & sTypeName(ty_date) & _
                " must not be zero!", vbOKOnly, "Error!")
            Exit Sub
          End If
```

```
      End If
      For j = 1 To vType(i)
        vInput(lIdx) = CDate(wsD.Cells(pr_dtAvg, lCol) + _
                       (dThisDouble(j) - dAvg) * _
                       wsD.Cells(pr_dtStDev, lCol) / dStDev)
        lIdx = lIdx + 1
      Next j
    End If
  Case ty_decimal
    If IsEmpty(wsD.Cells(pr_decAvg, lCol)) Or IsEmpty(wsD.Cells(pr_decStDev, lCol)) Then
      'Work with Min and Max
      dmin = wsD.Cells(pr_decMin, lCol)
      dmax = wsD.Cells(pr_decMax, lCol)
      For j = 1 To vType(i)
        vInput(lIdx) = CDec(dmin + Rnd() * (dmax - dmin))
        lIdx = lIdx + 1
      Next j
    Else
      'Work with Avg and StDev
      ReDim dThisDouble(1 To vType(i)) As Double
      For j = 1 To vType(i)
        dThisDouble(j) = Rnd()
      Next j
      dAvg = .Average(dThisDouble)
      dStDev = .StDevP(dThisDouble)
      If dStDev < 0.0000000000001 Then
        If vType(i) = 1 Then
          vInput(lIdx) = CDec(dAvg)
          lIdx = lIdx + 1
        Else
          Call MsgBox("StDev of data type " & sTypeName(ty_decimal) & _
               " must not be zero!", vbOKOnly, "Error!")
          Exit Sub
        End If
      End If
      For j = 1 To vType(i)
        vInput(lIdx) = CDec(wsD.Cells(pr_decAvg, lCol) + _
                       (dThisDouble(j) - dAvg) * _
                       wsD.Cells(pr_decStDev, lCol) / dStDev)
        lIdx = lIdx + 1
      Next j
    End If
  Case ty_double
    If IsEmpty(wsD.Cells(pr_dAvg, lCol)) Or IsEmpty(wsD.Cells(pr_dStDev, lCol)) Then
      'Work with Min and Max
      dmin = wsD.Cells(pr_dMin, lCol)
      dmax = wsD.Cells(pr_dMax, lCol)
      For j = 1 To vType(i)
        vInput(lIdx) = CDbl(dmin + Rnd() * (dmax - dmin))
        lIdx = lIdx + 1
      Next j
    Else
      'Work with Avg and StDev
      ReDim dThisDouble(1 To vType(i)) As Double
      For j = 1 To vType(i)
        dThisDouble(j) = Rnd()
      Next j
      dAvg = .Average(dThisDouble)
      dStDev = .StDevP(dThisDouble)
      If dStDev < 0.0000000000001 Then
        If vType(i) = 1 Then
          vInput(lIdx) = CDbl(dAvg)
          lIdx = lIdx + 1
        Else
          Call MsgBox("StDev of data type " & sTypeName(ty_double) & _
               " must not be zero!", vbOKOnly, "Error!")
          Exit Sub
        End If
      End If
      For j = 1 To vType(i)
        vInput(lIdx) = CDbl(wsD.Cells(pr_dAvg, lCol) + _
                       (dThisDouble(j) - dAvg) * _
                       wsD.Cells(pr_dStDev, lCol) / dStDev)
        lIdx = lIdx + 1
      Next j
    End If
  Case ty_long
    If IsEmpty(wsD.Cells(pr_lSum, lCol)) Then
      If IsEmpty(wsD.Cells(pr_lMaxRepeat, lCol)) Then
        'Work with arbitrary repetitions
        dmin = wsD.Cells(pr_lMin2, lCol)
        dmax = wsD.Cells(pr_lMax, lCol)
        For j = 1 To vType(i)
          vInput(lIdx) = Int(dmin + Rnd() * (dmax - dmin + 1))
          lIdx = lIdx + 1
        Next j
      Else
        If (wsD.Cells(pr_lMax, lCol) - wsD.Cells(pr_lMin2, lCol) + 1) * _
          wsD.Cells(pr_lMaxRepeat, lCol) < vType(i) Then
          Call MsgBox("Not enough random numbers for data type " & sTypeName(ty_long) & _
```

```
            "!", vbOKOnly, "Error!")
          Exit Sub
        End If
        v = sbRandInt(CLng(vType(i)), wsD.Cells(pr_lMin2, lCol), wsD.Cells(pr_lMax, lCol), _
            wsD.Cells(pr_lMaxRepeat, lCol))
        For j = 1 To vType(i)
          vInput(lIdx) = v(j)
          lIdx = lIdx + 1
        Next j
      End If
    Else
      v = sbLongRandSumN(wsD.Cells(pr_lSum, lCol), vType(i), _
          wsD.Cells(pr_lMin1, lCol))
      For j = 1 To vType(i)
        vInput(lIdx) = v(j)
        lIdx = lIdx + 1
      Next j
    End If
  Case ty_string
    If Not IsEmpty(wsD.Cells(pr_sLength, lCol)) Then
      'Simple string
      lLength = wsD.Cells(pr_sLength, lCol)
      If lLength <= 0 Then lLength = 1
      dmin = Asc(wsD.Cells(pr_sMin, lCol))
      dmax = Asc(wsD.Cells(pr_sMax, lCol))
      For j = 1 To vType(i)
        s = ""
        For k = 1 To lLength
          s = s & Chr(dmin + Rnd() * (dmax - dmin))
        Next k
        vInput(lIdx) = s
        lIdx = lIdx + 1
      Next j
    ElseIf Not IsEmpty(wsD.Cells(pr_sNextTabRepeat, lCol)) Then
      'Simple items from next tab
      Set wsItem = Sheets(2)
      If (wsItem.Cells(1, wsD.Cells(pr_sNextTabColumn, lCol)).End(xlDown).Row - 1) * _
        wsD.Cells(pr_sNextTabRepeat, lCol) < vType(i) Then
        Call MsgBox("Not enough random numbers for data type " & sTypeName(ty_string) & _
             "!", vbOKOnly, "Error!")
        Exit Sub
      End If
      v = sbRandInt(CLng(vType(i)), 2, _
          wsItem.Cells(1, wsD.Cells(pr_sNextTabColumn, lCol)).End(xlDown).Row, _
          wsD.Cells(pr_sNextTabRepeat, lCol))
      For j = 1 To vType(i)
        vInput(lIdx) = wsItem.Cells(1, wsD.Cells(pr_sNextTabColumn, lCol))(v(j))
        lIdx = lIdx + 1
      Next j
    Else
      'Items from weighted groups from next tab
      Set wsItem = Sheets(2)
      Set objGroup = CreateObject("Scripting.Dictionary")
      j = 2
      Do While Not IsEmpty(wsItem.Cells(j, wsD.Cells(pr_sNextTabGroupColumn, lCol)))
        objGroup.Item(wsItem.Cells(j, wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value) = _
            objGroup.Item(wsItem.Cells(j, wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value) + 1
        j = j + 1
      Loop
      'Are the item groups still identical to the ones in the param list?
      bGroupsUpToDate = True
      j = 0
      Do While Not IsEmpty(wsD.Cells(pr_sNextTabGroupWeights + j, pc_ItemGroups))
        If objGroup.Item(wsD.Cells(pr_sNextTabGroupWeights + j, pc_ItemGroups).Value) > 0 Then
          objGroup.Item(wsD.Cells(pr_sNextTabGroupWeights + j, pc_ItemGroups).Value) = 0
        Else
          Set objGroup = Nothing
          Set objGroup = CreateObject("Scripting.Dictionary")
          j = 2
          Do While Not IsEmpty(wsItem.Cells(j, wsD.Cells(pr_sNextTabGroupColumn, lCol)))
            objGroup.Item(wsItem.Cells(j, wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value) = _
                objGroup.Item(wsItem.Cells(j, wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value) + 1
            j = j + 1
          Loop
          bGroupsUpToDate = False
          Exit Do
        End If
        j = j + 1
      Loop
      If j <> objGroup.Count Then bGroupsUpToDate = False
      If Not bGroupsUpToDate Then
        Range(wsD.Cells(pr_sNextTabGroupWeights, pc_ItemGroups), _
          wsD.Cells(pr_sNextTabGroupWeights, pc_ItemGroups).End(xlDown)).ClearContents
        wsD.Cells(pr_sNextTabGroupWeights, pc_ItemGroups).Resize(objGroup.Count).FormulaArray = _
          .Transpose(objGroup.keys)
        If vbCancel = MsgBox("Item groups from next tab are not up to date!" & vbCrLf & _
          vbCrLf & "OK to continue anyway" & _
          vbCrLf & "Cancel to stop", vbOKCancel, "Warning") Then
          Exit Sub
        End If
```

```
          End If
          dSumWeights = 0#
          j = 0
          Do While Not IsEmpty(wsD.Cells(pr_sNextTabGroupWeights + j, pc_ItemGroups))
            dSumWeights = dSumWeights + wsD.Cells(pr_sNextTabGroupWeights + j, lCol)
            j = j + 1
          Loop
          ReDim dGroupWeights(1 To j) As Double
          For j = LBound(dGroupWeights) To UBound(dGroupWeights)
            dGroupWeights(j) = wsD.Cells(pr_sNextTabGroupWeights + j - 1, lCol) / dSumWeights * vType(i)
          Next j
          'Decide how many records to generate for each item group
          vGroup = RoundToSum(dGroupWeights, 0)
          For j = LBound(vGroup, 1) To UBound(vGroup, 1)
            If vGroup(j) > 0 Then
              Set wsItem = Sheets(2)
              Set objItem = CreateObject("Scripting.Dictionary")
              lRow = 2
              Do While Not IsEmpty(wsItem.Cells(lRow, wsD.Cells(pr_sNextTabGroupColumn, lCol)))
                If wsItem.Cells(lRow, wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value = _
                  objGroup.keys()(j - 1) Then
                  objItem.Item(wsItem.Cells(lRow, wsD.Cells(pr_sNextTabItemColumn, lCol)).Value) = _
                      objItem.Item(wsItem.Cells(lRow, wsD.Cells(pr_sNextTabItemColumn, lCol)).Value) + 1
                End If
                lRow = lRow + 1
              Loop
              If objItem.Count * wsD.Cells(pr_sNextTabItemRepeat, lCol) < vGroup(j) Then
                Call MsgBox("Not enough random numbers for data type string, item group " & _
                     wsD.Cells(pr_sNextTabGroupWeights + j, pc_ItemGroups).Value & _
                     "!", vbOKOnly, "Error!")
                Exit Sub
              End If
              v = sbRandInt(CLng(vGroup(j)), 1, objItem.Count, wsD.Cells(pr_sNextTabItemRepeat, lCol))
              For k = 1 To vGroup(j)
                vInput(lIdx) = objItem.keys()(v(k) - 1)
                lIdx = lIdx + 1
              Next k
              Set objItem = Nothing
            End If
          Next j
          Set objGroup = Nothing
        End If
      End Select
    End If
  Next i
  'Now shuffle the result vector into random order if specified
  If wsD.Cells(pr_shuffle, lCol) Then
    lRow = 2
    For Each v In UniqRandInt(lRecord, lRecord)
      wsD.Cells(lRow, lCol - pc_Input1 + pc_Output1) = vInput(v)
      lRow = lRow + 1
    Next v
  Else
    For lRow = 2 To lRecord + 1
      wsD.Cells(lRow, lCol - pc_Input1 + pc_Output1) = vInput(lRow - 1)
    Next lRow
  End If
Next lCol
wsD.Calculate
End With

End Sub
```

Exkurs

Wahrscheinlichkeiten berechnen – Ziehen von Karten mit und ohne Zurücklegen

Wenn Sie aus einem vollständigen Kartensatz von 52 Spielkarten 7 Karten ohne Zurücklegen ziehen, wie hoch ist die Wahrscheinlichkeit, dass Sie dann 3 Asse in der Hand halten?

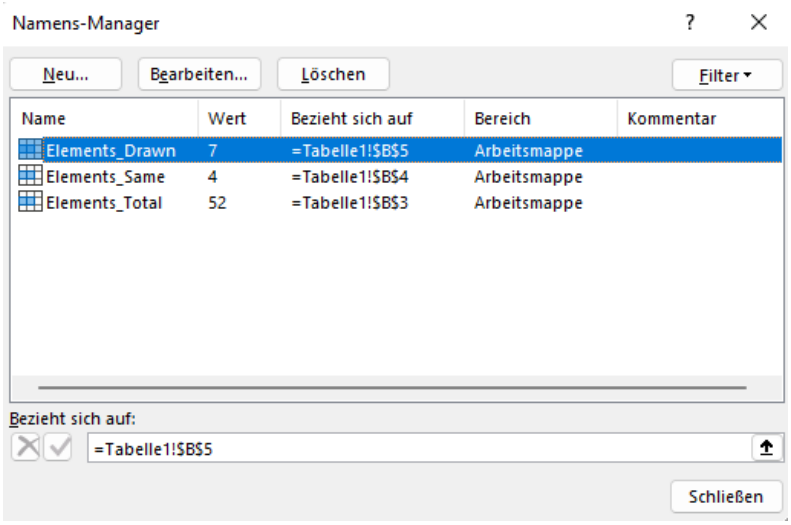
Die Antwort lautet: etwa 0,58%.

7 von 52 Karten ziehen (mit und ohne Zurücklegen)							
Anzahl Karten gesamt	**52**						
Anzahl Asse	**4**						
Anzahl gezogene Karten	**7**						
Wahrscheinlichkeiten	**Mit Zurücklegen**	**Läufe**	**Kein Ass**	**1 Ass**	**2 Asse**	**3 Asse**	**4 Asse**
Formel	WAHR		57,10%	33,31%	8,33%	1,16%	0,10%
Monte Carlo	WAHR	10000	57,13%	33,30%	8,43%	1,05%	0,09%
Formel	FALSCH		55,04%	36,69%	7,68%	0,58%	0,01%
Monte Carlo	FALSCH	10000	55,79%	36,23%	7,37%	0,60%	0,01%

Die genaue Formel für die Wahrscheinlichkeit ohne Zurücklegen ist für Excel 365 oder Excel 2021:
*=WENNFEHLER(KOMBINATIONEN(Elements_Drawn;SEQUENZ(1;Elements_Same + 1;0;1)) * (Elements_Same/Elements_Total)^SEQUENZ(1;Elements_Same + 1;0;1) * WENNFEHLER((1 - Elements_Same/Elements_Total)^(Elements_Drawn-SEQUENZ(1;Elements_Same + 1;0;1));1);0)*

Mit Zurücklegen lautet sie:
*=WENNFEHLER(KOMBINATIONEN(Elements_Same;SEQUENZ(1;Elements_Same + 1;0;1)) * KOMBINATIONEN(Elements_Total - Elements_Same;Elements_Drawn - SEQUENZ(1;Elements_Same + 1;0;1)) / KOMBINATIONEN(Elements_Total;Elements_Drawn); 0)*

Dabei wurden folgende Namen definiert:



Näherungsweise können Sie diese Wahrscheinlichkeiten auch mit einer Monte Carlo Simulation ermitteln:

monte Programmcode

```
Function monte(bWithReplacement As Boolean, _
    Optional runs As Long = 100000) As Variant
'(C) (P) by Bernd Plumhoff  27-Oct-2022 PB V0.2
Dim i As Long, j As Long, n As Long
Dim lAces As Long, lCards As Long
Dim lCardsDrawn As Long, lCardsSame As Long, lCardsTotal As Long
Dim r(1 To 5) As Variant
With Application.WorksheetFunction
lCardsTotal = Range("Elements_Total")
lCardsSame = Range("Elements_Same")
lCardsDrawn = Range("Elements_Drawn")
Randomize
For i = 1 To runs
  n = 0
  For j = 1 To lCardsDrawn
    If bWithReplacement Then
      lCards = lCardsTotal
      lAces = lCardsSame
    Else
      lCards = lCardsTotal + 1 - j
      lAces = lCardsSame - n
    End If
    If .RandBetween(1, lCards) < 1 + lAces Then
      n = n + 1
      If n = lCardsSame Then Exit For
    End If
  Next j
  r(1 + n) = r(1 + n) + 1
Next i
For i = 1 To lCardsSame + 1: r(i) = r(i) / runs: Next i
monte = r
End With
End Function
```

Spaß ohne Praxisrelevanz für Fortgeschrittene

Kleine VBA Pivot-Lösung – *sbMiniPivot*

“It’s not what you look at that matters, it’s what you see.” [Henry David Thoreau]

Bitte beachten Sie dass Excel 365 und Excel 2024 jetzt zwei ähnliche, aber mächtigere Tabellenblattfunktionen als sbMiniPivot anbietet – GROUPBY und PIVOTMIT. Verwenden Sie also sbMiniPivot lediglich für VBA Trainingszwecke. Wenn Sie eine Funktion wie CAT, COUNT, MAX, MIN oder SUM auf eine Liste von Zahlen- oder Zeichenketten-Kombinationen anwenden wollen und dabei auch Nebenbedingungen erfüllt werden sollen, dann können Sie sbMiniPivot verwenden:

	A	B	C	D	E	F
1	**Input**					
2	a	1	a			
3	a	2	b			
4	a	3	c			
5	b	4	d			
6	b	5	e			
7	c	6	f			
8	c	7	g			
9	c	8	h			
10	c	9	i			
11						
12	**Output numerical minimum**			**Output string minimum**		
13	a	1	=sbMiniPivot("min", A2:A10<>"c", A2:A10,B2:B10)	a	a	=sbMiniPivot("min",A2:A10<>"c", A2:A10,C2:C10)
14	b	4		b	d	
15						
16	**Output numerical maximum**			**Output string maximum**		
17	a	3	=sbMiniPivot("max",A2:A10<>"b", A2:A10,B2:B10)	a	c	=sbMiniPivot("max",A2:A10<>"b", A2:A10,C2:C10)
18	c	9		c	i	
19						
20	**Output numerical sum**					
21	a	6	=sbMiniPivot("sum", TRUE,A2:A10,B2:B10)			
22	b	9				
23	c	30				

Name

sbMiniPivot - Verbinde, zähle, summiere oder gib das Minimum oder Maximum der letzten gegebenen Eingabespalte zurück für alle Kombinationen der vorherigen Spalteneinträge bei der die entsprechende Zeile der Bedingungsspalte WAHR ist.

Synopsis

sbMiniPivot(sFunction, vCondition, ParamArray vInput)

Beschreibung

sbMiniPivot führt die Funktion sFunction für die letzte Spalte von vInput aus für alle Kombinationen der vorhergehenden Spalten bei der der entsprechende Wert in der Bedingungsspalte vCondition WAHR ist. Es wird ein Variant Array zurückgegeben.

Optionen

sFunction - Spezifiziert die Funktion, die auf alle Kombinationen angewendet wird. Dies kann concatenate (cat), count, max(imum), min(imum), oder sum sein.

vCondition - Die Bedingungskonstante WAHR oder FALSCH oder eine Spalte mit Boolschen WAHR/FALSCH Werten

vInput - Zwei oder mehr Spalten. sFunction wird auf die letzte Spalte für alle Kombinationen der vorigen Spalten angewandt bei der der entsprechende Wert der Bedingungsspalte WAHR ist.

sbMiniPivot Programmcode

```
Enum mc_Macro_Categories
  mcFinancial = 1
  mcDate_and_Time
  mcMath_and_Trig
  mcStatistical
  mcLookup_and_Reference
  mcDatabase
  mcText
  mcLogical
  mcInformation
  mcCommands
  mcCustomizing
  mcMacro_Control
  mcDDE_External
  mcUser_Defined
  mcFirst_custom_category
  mcSecond_custom_category 'and so on
End Enum 'mc_Macro_Categories

Function sbMiniPivot(sFunction As String, vCondition As Variant, _
  ParamArray vInput() As Variant) As Variant
'sbMiniPivot performs the function sFunction on last given column of
'vInput() for all combinations of the previous ones where corresponding
'elements of vCondition are TRUE.
'Example:
'    A     B     C
' 1 Smith Adam   1
' 2 Myer  Ben    3
' 3 Smith Ben    2
' 4 Smith Adam   7
' 5 Myer  Ben    4
'Now array-enter into D1
'=sbMiniPivot("sum", B1:B5="Ben", A1:A5,B1:B5,C1:C5) and you will get
'    D      E    F
' 1 Myer   Ben   7
' 2 Smith  Ben   2
'(C) (P) by Bernd Plumhoff 31-Jul-2022 PB V1.1
Dim b As Boolean, bCondition As Boolean
Dim i As Long, j As Long, k As Long, liscount As Long
Dim lvdim As Long, lcdim As Long
Dim obj As Object
Dim s As String, sC As String
Dim vR As Variant
```

```
With Application.WorksheetFunction
sC = "|"
k = 0
vInput(0) = .Transpose(.Transpose(vInput(0)))
If LCase(sFunction) = "count" Then liscount = 1
If UBound(vInput) < 1 - liscount Then
  sbMiniPivot = CVErr(xlErrValue)
  Exit Function
End If
lvdim = UBound(vInput(0))
Select Case VarType(vCondition)
Case vbBoolean
  bCondition = True
Case vbArray + vbVariant
  bCondition = False
  vCondition = .Transpose(.Transpose(vCondition))
  lcdim = UBound(vCondition, 1)
  If lcdim <> lvdim Then
    sbMiniPivot = CVErr(xlErrRef)
    Exit Function
  End If
Case Else
  sbMiniPivot = CVErr(xlErrNA)
  Exit Function
End Select
If lvdim > 100 Then lvdim = 100 'Let us start with small dimension
On Error GoTo ErrHdl
ReDim vR(0 To UBound(vInput) + liscount, 1 To lvdim)
For j = 1 To UBound(vInput)
  vInput(j) = .Transpose(.Transpose(vInput(j)))
  If UBound(vInput(0)) <> UBound(vInput(j)) Then
    sbMiniPivot = CVErr(xlErrRef)
    Exit Function
  End If
Next j
Set obj = CreateObject("Scripting.Dictionary")
For i = 1 To UBound(vInput(0))
  b = bCondition
  If Not b Then b = vCondition(i, 1)
  If b Then
    s = vInput(0)(i, 1)
    For j = 1 To UBound(vInput) - 1 + liscount
      s = s & sC & vInput(j)(i, 1)
    Next j
    If obj.Item(s) > 0 Then
      Select Case LCase(sFunction)
      Case "cat", "concatenate"
        vR(UBound(vInput), obj.Item(s)) = vR(UBound(vInput), _
          obj.Item(s)) & "," & vInput(UBound(vInput))(i, 1)
      Case "count"
        vR(UBound(vInput) + 1, obj.Item(s)) = vR(UBound(vInput) + 1, _
          obj.Item(s)) + 1
      Case "max", "maximum"
        If vR(UBound(vInput), obj.Item(s)) < vInput(UBound(vInput))(i, 1)
Then
          vR(UBound(vInput), obj.Item(s)) = vInput(UBound(vInput))(i, 1)
        End If
      Case "min", "minimum"
        If vR(UBound(vInput), obj.Item(s)) > vInput(UBound(vInput))(i, 1)
Then
          vR(UBound(vInput), obj.Item(s)) = vInput(UBound(vInput))(i, 1)
        End If
      Case "sum"
```

```
          vR(UBound(vInput), obj.Item(s)) = vR(UBound(vInput), _
            obj.Item(s)) + vInput(UBound(vInput))(i, 1)
        Case Else
          sbMiniPivot = CVErr(xlErrRef)
        End Select
      Else
        k = k + 1
        obj.Item(s) = k
        For j = 0 To UBound(vInput)
          vR(j, k) = vInput(j)(i, 1)
        Next j
        If liscount = 1 Then vR(UBound(vInput) + 1, k) = 1
      End If
    End If
Next i
'Reduce result array to used area
If k > 0 Then ReDim Preserve vR(0 To UBound(vInput) + liscount, 1 To k)
sbMiniPivot = .Transpose(vR)
Set obj = Nothing
End With
Exit Function

ErrHdl:
If Err.Number = 9 Then
  If i > lvdim Then
    'Here we normally get if we breach Ubound(vR,2)
    lvdim = 10 * lvdim 'So we need to increase last dimension
    If lvdim > UBound(vInput(0)) Then lvdim = UBound(vInput(0))
    ReDim Preserve vR(0 To UBound(vInput) + liscount, 1 To lvdim)
    Resume 'Back to statement which caused error
  End If
End If
On Error GoTo 0 'Other error - terminate
Resume
End Function

Sub DescribeFunction_sbMiniPivot()
'Run this only once, then you will see this description in the function
menu
Dim FuncName As String, FuncDesc As String, Category As String
Dim ArgDesc(1 To 3) As String
FuncName = "sbMiniPivot"
FuncDesc = "Concatenate, count, sum or return min or max of last given
input " & _
  "column for all combinations of the previous ones where same row " & _
  "of condition column is True"
Category = mcStatistical
ArgDesc(1) = "Function to apply - cat, count, max, min, or sum"
ArgDesc(2) = "Condition constant True or False or column which contains
True/False values"
ArgDesc(3) = "Two or more columns"
Application.MacroOptions _
  Macro:=FuncName, _
  Description:=FuncDesc, _
  Category:=Category, _
  ArgumentDescriptions:=ArgDesc
End Sub
```

Rundenturnier-Paarungen mit Excel Tabellenblattfunktionen

Eine Lösungsansatz mit Tabellenblattfunktionen für Rundenturniere Jeder gegen Jeden:

Normalerweise verwenden Sie Bereichsnamen in Excel, um eine hartkodierte Zuweisung zu Eingabe- oder Ausgabezellen zu vermeiden. Wenn Sie wichtigen Zellen Namen zuweisen, werden deren Bezüge bei Zeilen- oder Spalteneinfügungen oder -löschungen automatisch angepasst.

Hier wollen wir Bereichsnamen in Excel für die effiziente Implementierung einer Lösung mit verschachtelten Tabellenblattfunktionen verwenden, die nicht mehr verändert werden muss:

Für eine (fast) beliebige Anzahl von Spielern wollen wir ein Rundenturnier organisieren, bei dem jeder Spieler gegen jeden anderen genau einmal spielt. Der Einfachheit halber und ohne Beschränkung der Allgemeinheit legen wir fest, dass Spieler 1 im ersten Spiel Heimrecht haben soll (oder anders ausgedrückt: mit Weiß spielt).

Wir können nun verschachtelte Formeln mit Bereichsnamen realisieren. Dabei sind die notwendigen Parameter die jeweiligen Zeilen- und Spaltenpositionen relativ zum Ankerpunkt der Paarungstabelle oben links.

Bitte beachten Sie: Ein Bereichsname wird genau dann ausgewertet, wenn eine Formel berechnet wird, die sich auf ihn bezieht. Dies bedeutet, dass er mehrfach ausgewertet wird, wenn er in mehreren neuberechneten Formeln genutzt wird. Unrefenzierte Bereichsnamen werden nicht ausgewertet, auch nicht bei einer vollständigen Neuberechnung (STRG + ALT + F9) .

Entwickeln einer Formellösung für ein Rundenturnier Jeder gegen Jeden

Für unser Rundenturnier haben wir nach Festlegung der Turnierart Jeder gegen Jeden lediglich einen Parameter, die Anzahl der Spieler.

Wir wollen die zu spielenden Paarungen in einer Tabelle ausgeben. Beispiel:

	Tisch 1	Tisch 2	Tisch 3
Runde 1	1 - 6	5 - 2	3 - 4
Runde 2	5 - 1	4 - 6	2 - 3
Runde 3	1 - 4	3 - 5	6 - 2
Runde 4	3 - 1	2 - 4	5 - 6
Runde 5	1 - 2	6 - 3	4 - 5

Die Paarungen je Runde zeigen wir in den Zeilen, an welchem Tisch (oder Platz) gespielt werden soll, steht in den Spalten, und der jeweils erstgenannte Spieler hat Heimrecht (spielt mit Weiß).

Der Name des Tabellenblattes, in dem unsere Eingabe (Spieleranzahl) und Ausgaben (die Paarungstabelle) stehen, soll Eingabe_und_Ausgabe lauten.

Die Eingabezelle für die Spieleranzahl nennen wir Player_Count. Sie hat den Wert =Eingabe_und_Ausgabe!$B$1.

Damit die Paarungstabelle schnell und unaufwändig verschoben werden kann, legen wir als Anker den Namen Table_Top_Left (hier mit dem Wert =Eingabe_und_Ausgabe!$A$10) fest. Die weiteren Bereichsnamen legen wir relativ zu diesem Anker fest, um absolute Bezüge zu vermeiden.

Der aktuelle Tisch (Spalte) soll This_Table heißen mit dem Wert =SPALTE() - SPALTE(Table_Top_Left). Die aktuelle Runde (Zeile) soll This_Round heißen mit dem Wert =ZEILE() - ZEILE(Table_Top_Left).

Die Spaltenköpfe (Tisch 1, Tisch 2, …) können wir dann Table_Title mit dem Wert =WENN(ISTGERADE(Player_Count); WENN(This_Table <= Player_Count / 2; “Tisch " & This_Table; “"); WENN(This_Table = 1; “Frei”; WENN(This_Table <= (Player_Count + 1) / 2; “Tisch " & This_Table - 1; “"))) nennen. Die Zeilenköpfe (Runde 1, Runde 2, …) heißen Round_Title mit dem Wert =WENN(This_Round <= (Player_Count - WENN(ISTUNGERADE(Player_Count); 0; 1)); “Runde " & This_Round; “").

Schließlich legen wir für die jeweiligen Paarungen den allgemeinen Namen This_Game mit dem Wert =WENN(UND(This_Round <= (Player_Count - WENN(ISTUNGERADE(Player_Count); 0; 1)); This_Table <= (Player_Count + 1) / 2); WENN(ISTGERADE(Player_Count); Even_This_Game; Odd_This_Game); “") fest.

Anmerkung: Die weiter notwendigen Namen Even_This_Game und Odd_This_Game erklären wir in den unten gezeigten Kapiteln für eine gerade bzw. ungerade Spieleranzahl.

Nun die Festlegung der auszugebenden Paarungstabelle: In die erste Zeile geben wir in die zweite Spalte Table_Title ein. In die erste Spalte geben wir in die zweite Zeile Round_Title ein. Die erste Paarung in der zweiten Zeile und zweiten Spalte lautet: This_Game. Wir kopieren die letzte Spalte nach rechts und die unterste Zeile nach unten, bis wir leere Zellen als Ausgabe erhalten.

Beispiel für eine gerade Spieleranzahl

Das Bildungsgesetz für die Paarungen bei gerader Spieleranzahl: Wir sehen an der obigen Beispieltabelle für 6 Spieler, dass Spieler 1 immer an Tisch 1 sitzen bleibt und gegen die Spieler 6, 5, 4, … mit wechselnden Farben beginnend mit Weiß spielt. In der ersten Runde spielen der erste Spieler gegen den letzten, der zweite gegen den vorletzten, usw., aber immer mit wechselnden Farben. In den folgenden Runden wird an den Tischen größer 1 jeder Spieler durch den nächstniedrigeren (in der Formel zu erkennen an dem Summanden - This_Round) ersetzt. Ausnahme: Auf Spieler 2 folgt der letzte Spieler (hier 6).

Damit wird die Festlegung der folgenden Namen klar:

Name	Bezieht sich auf
Even_Equal_One_PlayerA	=1
Even_Equal_One_PlayerB	=(1 + Player_Count - This_Round)
Even_Greater_One_PlayerA	=(2 + REST(Player_Count - This_Table - This_Round; Player_Count - 1))
Even_Greater_One_PlayerB	=(2 + REST(This_Table - This_Round - 1; Player_Count - 1))
Even_This_Game	=WENN(This_Table = 1; WENN(ISTUNGERADE(This_Round); Even_Equal_One_PlayerA & " - " & Even_Equal_One_PlayerB; Even_Equal_One_PlayerB & " - " & Even_Equal_One_PlayerA); WENN(ISTGERADE(This_Table); Even_Greater_One_PlayerA & " - " & Even_Greater_One_PlayerB; Even_Greater_One_PlayerB & " - " & Even_Greater_One_PlayerA))

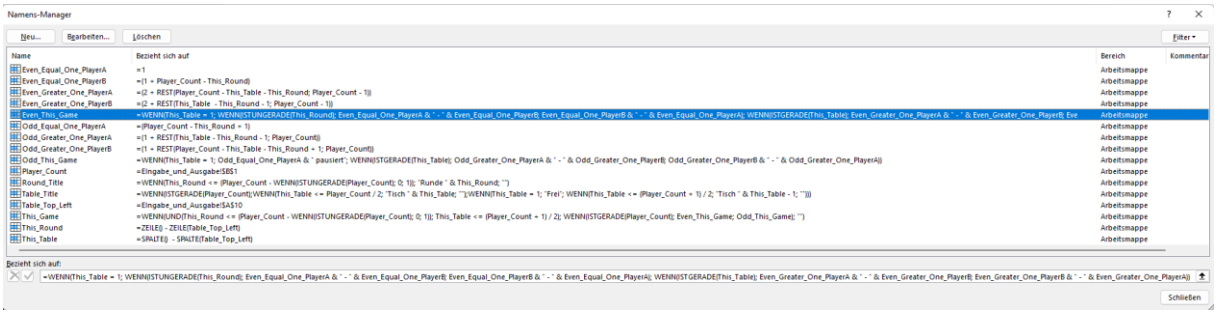
Beispiel für eine ungerade Spieleranzahl

	Frei	Tisch 1	Tisch 2	Tisch 3
Runde 1	7 pausiert	1 - 6	5 - 2	3 - 4
Runde 2	6 pausiert	7 - 5	4 - 1	2 - 3
Runde 3	5 pausiert	6 - 4	3 - 7	1 - 2
Runde 4	4 pausiert	5 - 3	2 - 6	7 - 1
Runde 5	3 pausiert	4 - 2	1 - 5	6 - 7
Runde 6	2 pausiert	3 - 1	7 - 4	5 - 6
Runde 7	1 pausiert	2 - 7	6 - 3	4 - 5

Das Bildungsgesetz für die Paarungen bei ungerader Spieleranzahl: In der ersten Runde pausiert der letzte Spieler, in der zweiten der vorletzte, usw. In der ersten Runde spielen der erste Spieler gegen den vorletzten, der zweite gegen den vorvorletzten, usw., aber immer mit wechselnden Farben. In den folgenden Runden wird an den Tischen größer 1 jeder Spieler durch den nächstniedrigeren (in der Formel zu erkennen an dem Summanden - This_Round) und Spieler 1 durch den Letzten ersetzt.

Damit wird auch die Festlegung der folgenden Namen klar:

Name	Bezieht sich auf
Odd_Equal_One_PlayerA	=(Player_Count - This_Round + 1)
Odd_Greater_One_PlayerA	=(1 + REST(This_Table - This_Round - 1; Player_Count))
Odd_Greater_One_PlayerB	=(1 + REST(Player_Count - This_Table - This_Round + 1; Player_Count))
Odd_This_Game	=WENN(This_Table = 1; Odd_Equal_One_PlayerA & " pausiert"; WENN(ISTGERADE(This_Table); Odd_Greater_One_PlayerA & " - " & Odd_Greater_One_PlayerB; Odd_Greater_One_PlayerB & " - " & Odd_Greater_One_PlayerA))



Das Interessante ist: Sie können auch diesen Ansatz für (fast) beliebig viele Spieler verwenden. Kopieren Sie lediglich die Zeilen so weit wie nötig nach unten und die Spalten nach rechts, bis Sie Leerzellen erhalten.

Die Formeln funktionieren sogar für die pathologischen Fälle mit keinem, einem oder zwei Spielern.

Eine kleine Quizfrage: Warum existiert der Name *Odd_Equal_One_PlayerB* nicht?

Antwort: Weil die Formeln als Tisch 1 den pausierenden Spieler ansehen und die Tischnummern bei ungerader Spieleranzahl künstlich um 1 reduziert werden - siehe die Formel für *Table_Title*.

Noch eine kleine Quizfrage: Wie kann man diese Lösung erweitern, um die Paarungen for mehrere verschiedene Turniere in einer Excel-Datei zu berechnen?

Antwort: Erstellen Sie alle genannten Bereichsnamen als lokale Namen im aktuellen Tabellenblatt und kopieren Sie dann das Tabellenblatt so oft wie nötig. Leider können Sie nicht lediglich die Namen Player_Count und Table_Top_Left lokal definieren. Alle Bereichsnamen müssen so oft existieren wie sie Turniere berechnen wollen.

Anmerkung: Eine allgemeinere Lösung für Rundenturniere, bei der Sie auch die Farbe für Spieler 1 in der ersten Runde frei wählen können und die eine zusätzliche Ausgabematrix erzeugt, finden Sie im Kapitel „**Fehler! Verweisquelle konnte nicht gefunden werden.**“ auf Seite **Fehler! Textmarke nicht definiert.**.

TEXTVERKETTEN

Die Excel Funktion TEXTVERKETTEN kann ab Excel Version 2019 die Inhalte von Zellbereichen und Arrays als Zeichenketten verbinden und mit vorgegebenem Trennzeichen ausgeben. Für Excel Versionen älter als Excel 2019 können Sie die hier vorgestellte benutzerdefinierte Funktion TEXTVERKETTEN verwenden.

Das Gute ist: Wenn Sie dann auf Excel 2019 oder neuer umstellen, müssen Sie nichts machen. Excel wird dann automatisch die eingebaute Funktion TEXTVERKETTEN verwenden.

Bitte beachten: Ich habe bewusst darauf verzichtet, mich um pathologische Fälle zu kümmern wie Trennzeichen ist ein Array oder Leer_ignorieren enthält ganze Zahlen wie 0 oder 1. Ich kopierte aber auch nicht die Fehler von Excel's TEXTVERKETTEN. So behandelt Excel's TEXTVERKETTEN Funktion Überschneidungen von nicht-zusammenhängenden Bereichen wie z. B. (A1:C3,D4:F6,G7:I9) (A1:B2,C3:D4,E5:F6,G7:H8,I9:J10) fehlerhaft. Wie bei fast jeder Analogie musste und wollte ich eine Grenze ziehen.

TEXTVERKETTEN Programmcode

```
Function TEXTVERKETTEN(Trennzeichen As String, _
    Leer_ignorieren As Boolean, _
    ParamArray Text() As Variant) As String
'(C) (P) by Bernd Plumhoff 07-Jan-2022 PB V1.1
Dim v, i As Long, s As String, t As String
For i = LBound(Text) To UBound(Text)
  If IsArray(Text(i)) Then
    For Each v In Text(i)
      t = IIf(IsMissing(v), "", v)
      If Not (Leer_ignorieren And t = "") Then
        TEXTVERKETTEN = TEXTVERKETTEN & s & t
        s = Trennzeichen
      End If
    Next v
  Else
    t = IIf(IsMissing(Text(i)), "", Text(i))
    If Not (Leer_ignorieren And t = "") Then
      TEXTVERKETTEN = TEXTVERKETTEN & s & t
      s = Trennzeichen
    End If
  End If
Next i
End Function
```

Index