Summenerhaltendes Runden mit *RoundToSum* (Excel / VBA)

(C) (P) 2024 Bernd Plumhoff Stand: 19. Oktober 2024

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 einfach nutzbare 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.

Inhaltsverzeichnis

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 Eingabewerte als unveränderte absolute Werte. FALSCH verwendet die Prozentzahlen der Eingabewerte, um genau auf die Summe 100% zu kommen.
lErrorType	Optional, der Standardwert ist 1. Fehlertyp, der minimal gehalten werden soll: 1 – absoluter Fehler, 2 – relativer Fehler.
bDontAmend	Optional, der Standardwert ist FALSCH. WAHR lässt die Eingabewerte unverändert. FALSCH führt die Funktion wie beschrieben aus. Dieser Parameter dient zur einfachen Veranschaulichung der Funktion.

RoundToSum Programmcode

```
Option Explicit

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, Optional bDontAmend As Boolean = False) 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.
'V2.1 PB 12-Oct-2024 (C) (P) by Bernd Plumhoff
Dim 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
If Not bDontAmend Then
  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
End If
RoundToSum = vC
If TypeName(Application.Caller) = "Range" Then
  If Application.Caller.Rows.Count > Application.Caller.Columns.Count Then
    RoundToSum = .Transpose(vC) 'It's two-dimensional with 2nd dim const = 1
  End If
End If
Exit Function
Errhdl:
'Transpose variants to be able to address them with vA(i), not vA(i,1)
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 5) 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"
ArgDesc(5) = "[Optional = False] True does not amend the rounded summands to match the rounded sum; False performs the
calculation as described"
Application.MacroOptions _
  Macro:=FuncName, _
  Description:=FuncDesc, _
  Category:=Category, _
  ArgumentDescriptions:=ArgDesc
End Sub
```

Round2Sum Lambda-Ausdruck

Wenn wir eine verlässliche Rank.Dense Funktion als Lambda-Ausdruck zur Verfügung hätten, könnten wir die VBA Funktion *RandToSum* durch diesen *Round2Sum* Lambda-Ausdruck ersetzen:

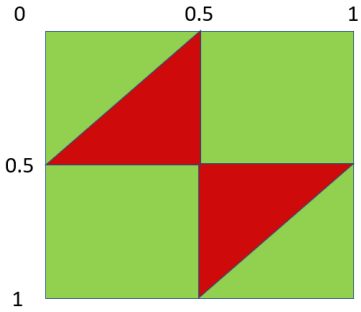
```
=LAMBDA(vI;lD;bA;lE;bD;
  LET(
    i;WENN(bA;
        vI;
        vI/SUMME(vI)%
      );
    r;RUNDEN(i;lD);
    _C;RUNDEN(SUMME(i);lD)-SUMME(r);
    _E;WAHL(lE;r-i;(r-i)*i);
    _R;Rank.Dense(_E;_E;WENN(_C>0;1;0));
    _D;WENN(_R <= RUNDEN(ABS(_C*10^lD);0);
         VORZEICHEN(_C)*10^-lD;
         0
       );
    WENN(bD;vI;r+_D)
  )
)
```

Leider gibt dieser Rank.Dense Lambda-Ausdruck, der von der Webseite
https://www.flexyourdata.com/blog/excel-lambda-rank-dense-an-excel-lambda-function-for-dense_rank-from-sql/
kopiert und ins Deutsche übertragen wurde, zum Beispiel für die beiden Gleitpunktzahlen
A1: =Pi()
A2: =Pi()
zweimal den Rang 1 aus:

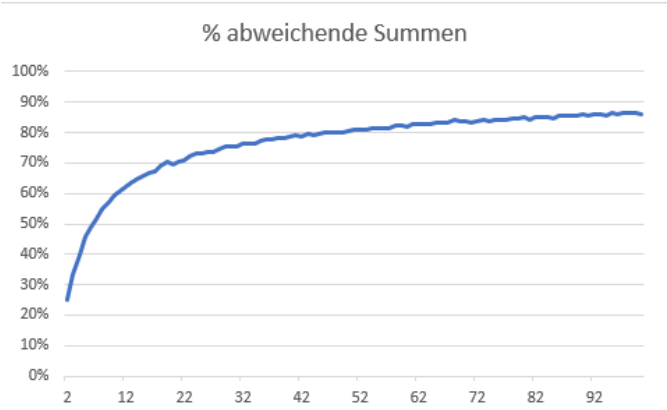
```
=LAMBDA(Zahl;Ref;[Order];
    LET(
      _ord; WENN(WURDEAUSGELASSEN(Order); -1; WENN(Order=0; -1; 1));
      _z; Zahl;
      _r; INDEX(WENN(ZEILEN(Ref)=1; MTRANS(Ref); Ref);; 1);
      _d; SORTIEREN(_r; 1; _ord);
      _i; SEQUENZ(ZEILEN(_d));
      _ranks; SCAN(0; _i;
            LAMBDA(a;b;
              WENNS(
                b=1; 1;
                INDEX(_d; b-1; 1)=INDEX(_d; b; 1); a;
                WAHR; a+1
              )
            )
           );
      _out; MAP(_z; LAMBDA(x; XVERWEIS(x; _d; _ranks; "Kein Rang")));
      _out
    )
  )
```

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:



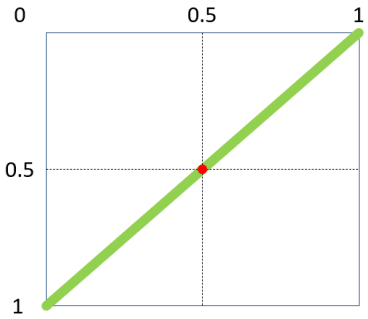
Aber etwas überraschend mag sein, dass die Wahrscheinlichkeit sich 90% nähert, je mehr Zahlen gerundet und addiert werden:



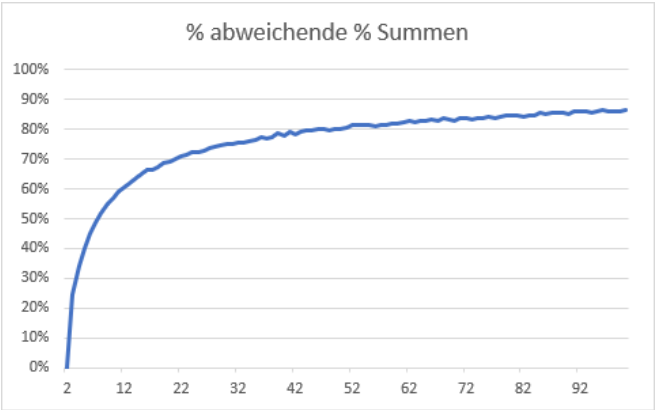
Bei sieben zufälligen Gleitkommazahlen ist die Wahrscheinlichkeit bereits größer als 50%, dass die Summe ihrer gerundeten Werten nicht gleich ihrer gerundeten Summe ist.

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.
'Source (EN): https://www.sulprobil.com/rounding_values_alters_their_sum_en/
'Source (DE): https://www.bplumhoff.de/werte_runden_aendert_ihre_summe_de/
'(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.
'Source (EN): https://www.sulprobil.com/rounding_values_alters_their_sum_en/
'Source (DE): https://www.bplumhoff.de/werte_runden_aendert_ihre_summe_de/
'(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:

Umlage 1 - Kostenstellen - Gemeinkosten auf alle anderen Kostenstellen

		Allgemeine Kostenstellen								Hilfskostenstellen			Hauptkostenstellen		
Schlüssel-bezeichnung	Gesamt	Geschäfts-führung	Sekretariat	Rechnungs-wesen	Controlling	Personal	Marketing	Auszubil-dende	Betriebs-rat	Werkstatt 1	Werkstatt 2	Fuhrpark	Fertigung1	Fertigung2	Fertigung3
pro Kopf	102	1	1	3	1	2	3	3	1	12	10		20	20	25
m²	2685	50	40	100	30	50	50		15	250	350	100	500	550	600
gleich	14	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gewichtung	16	2	1	1	1	1	2	1	1	1	1	1	1	1	1

Diese erste Gemeinkostenumlage erfolgt mit einer Matrixformel und dem Aufruf einer Rundungs-korrektur, damit die einzelnen Summanden cent-genau die Spaltensummen pro Kostenstelle ergeben:

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

Die zweite Gemeinkostenumlage verteilt die allgemeinen Kostenstellen und die Hilfskostenstellen mit einer Matrixformel und dem Aufruf einer Rundungskorrektur cent-genau auf die Hauptkostenstellen:

Umlage 2 - Allgemeine Kostenstellen und Hilfskostenstellen auf Hauptkostenstellen

Umzulegende Kostenstellen	Schlüssel	Fertigung1	Fertigung2	Fertigung3	Gesamt
Geschäftsführung	Gewichtung	30%	40%	30%	100%
Sekretariat	Gewichtung	40%	50%	10%	100%
Rechnungswesen	Gewichtung	30%	13%	57%	100%
Controlling	gleich	1	1	1	3
Personal	pro Kopf	20	20	25	65
Marketing	Gewichtung	30%	42%	28%	100%
Auszubildende	gleich	1	1	1	3
Betriebsrat	pro Kopf	20	20	25	65
Werkstatt 1	Gewichtung	25%	20%	55%	100%
Werkstatt 2	Gewichtung	20%	20%	60%	100%
Fuhrpark	Gewichtung	40%	30%	30%	100%

Das Endergebnis:

E9 =RoundToSum($D9/Schlüssel!$F20*Schlüssel!C20:E20)

	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	8.618,74	120.284,74	36.085,42	48.113,90	36.085,42	120.284,74
6	Sekretariat	34.627,00	4.309,37	38.936,37	15.574,55	19.468,18	3.893,64	38.936,37
7	RW	96.834,00	4.309,37	101.143,37	30.343,01	13.148,64	57.651,72	101.143,37
8	Controlling	83.875,00	4.309,37	88.184,37	29.394,79	29.394,79	29.394,79	88.184,37
9	Personal	53.765,00	4.309,37	58.074,37	17.869,04	17.869,04	22.336,29	58.074,37
10	ÖA/Werbung	239.170,00	8.618,75	247.788,75	74.336,62	104.071,28	69.380,85	247.788,75
11	Auszub.	147.397,00	4.309,37	151.706,37	50.568,79	50.568,79	50.568,79	151.706,37
12	Betriebsrat	471,00	4.309,38	4.780,38	1.470,88	1.470,89	1.838,61	4.780,38
13	Werkstatt 1	125.225,00	4.309,38	129.534,38	32.383,59	25.906,88	71.243,91	129.534,38
14	Werkstatt 2	2.398.512,00	4.309,38	2.402.821,38	480.564,27	480.564,28	1.441.692,83	2.402.821,38
15	Fuhrpark	26.992,00	4.309,38	31.301,38	12.520,55	9.390,42	9.390,41	31.301,38
16	1. Umlage				4.309,38	4.309,38	4.309,38	12.928,14
17	2. Umlage	3.318.534,00	56.021,86	3.374.555,86	781.111,51	799.967,09	1.793.477,26	3.374.555,86
18	KSt. - Einzelkosten				738.060,00	854.000,00	650.360,00	2.242.420,00
19	Gesamt Haupkostenstellen				1.523.480,89	1.658.276,47	2.448.146,64	5.629.904,00
20								
21	Gemeinkostenzuschlagssatz				106,4%	94,2%	276,4%	151,1%

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:

D18 f_x {=INDEX($A$3:$A$5,INT(sbExactRandHistogrm(6,1,4,$B$3:$B$5)))}

	A	B	C	D	E	F	G	H	I	J	K	L
1				**Just statistical likelihood**						**Total**		
2	**Color**	**Likelihood**	**Pos / Iteration**	**One**	**Two**	**Three**	**Four**	**Five**	**Six**	**Green**	**Yellow**	**Red**
3	**Green**	50.00%	**1**	Green	Green	Green	Green	Green	Red	5	0	1
4	**Yellow**	33.33%	**2**	Green	Green	Yellow	Green	Yellow	Red	3	2	1
5	**Red**	16.67%	**3**	Green	Yellow	Yellow	Green	Red	Green	3	2	1
6			**4**	Yellow	Green	Red	Green	Yellow	Yellow	2	3	1
7			**5**	Green	Yellow	Yellow	Green	Green	Yellow	3	3	0
8			**6**	Green	Yellow	Red	Green	Green	Green	4	1	1
9			**7**	Green	Yellow	Red	Green	Yellow	Red	2	2	2
10			**8**	Yellow	Green	Green	Yellow	Red	Yellow	2	3	1
11			**9**	Yellow	Green	Red	Red	Red	Green	2	1	3
12			**10**	Green	Yellow	Green	Yellow	Red	Red	2	2	2
13									**Total:**	28	19	13
14									**Should stochastically be:**	30	20	10
16				**Exact likelihood**						**Total**		
17			**Pos / Iteration**	**One**	**Two**	**Three**	**Four**	**Five**	**Six**	**Green**	**Yellow**	**Red**
18			**1**	Green	Red	Green	Yellow	Green	Yellow	3	2	1
19			**2**	Yellow	Green	Green	Green	Red	Yellow	3	2	1
20			**3**	Green	Green	Green	Red	Yellow	Yellow	3	2	1
21			**4**	Yellow	Green	Green	Green	Yellow	Red	3	2	1
22			**5**	Red	Green	Green	Yellow	Yellow	Green	3	2	1
23			**6**	Green	Yellow	Green	Yellow	Red	Green	3	2	1
24			**7**	Green	Green	Yellow	Yellow	Green	Red	3	2	1
25			**8**	Green	Yellow	Green	Yellow	Green	Red	3	2	1
26			**9**	Yellow	Red	Yellow	Green	Green	Green	3	2	1
27			**10**	Green	Green	Yellow	Yellow	Red	Green	3	2	1
28									**Total:**	30	20	10
29									**Should stochastically be:**	30	20	10

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.
'Source (EN): http://www.sulprobil.de/sbexactrandhistogrm_en/
'Source (DE): http://www.berndplumhoff.de/sbexactrandhistogrm_de/
'(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

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:

	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:

	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.

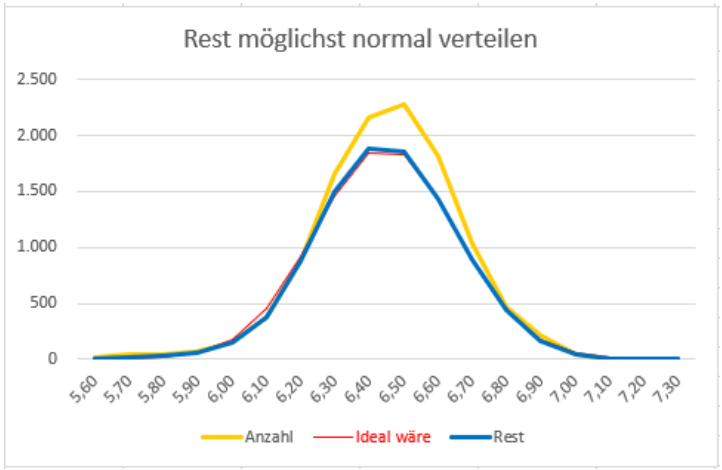
sbFairStaffSelection Programmcode

```
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.
'Source (EN): https://www.sulprobil.de/sbfairstaffselection_en
'Source (DE): https://www.bplumhoff.com/sbfairstaffselection_de
'(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

C4 =MTRANS(RoundToSUm(NORM.VERT(A4:A21;B24;B25;FALSCH)*B22/10;0;;2))

	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**	**1.664**	**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

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 einfach 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
'Source (EN): http://www.sulprobil.de/sbswv_en/
'Source (DE): http://www.berndplumhoff.de/sbswv_de/
'(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
```

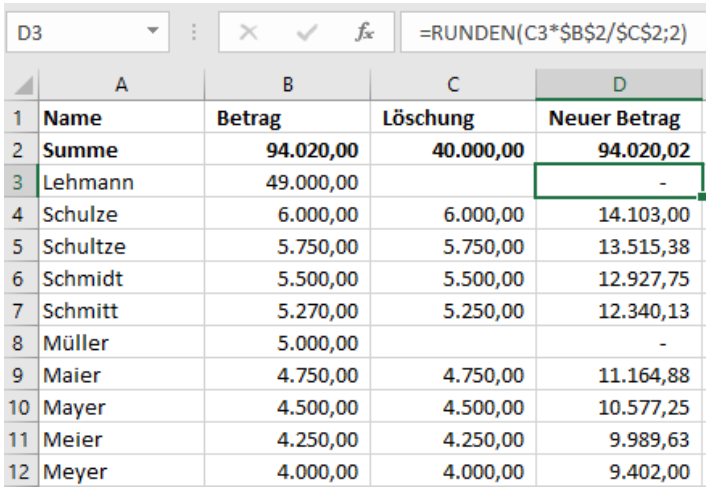
```
                End If
            Next n
        #End If
        k = k + vW(i)
        Select Case 2 * k
        Case j + m
            If m = 0 Then
                #If Not SORTED Then
                    'Ensure vV(i + 1) is next greater value
                    For n = i + 2 To UBound(vW)
                        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 einfacher Ansatz

Eine einfache Formel ist *=RUNDEN(C3*$B$2/$C$2;2)*, die Sie von D3 nach D12 hinunterkopieren können:



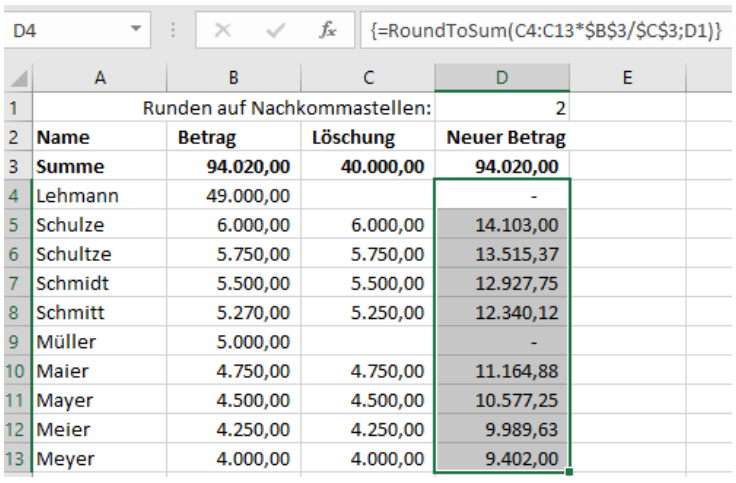
	A	B	C	D
1	**Name**	**Betrag**	**Löschung**	**Neuer Betrag**
2	**Summe**	**94.020,00**	**40.000,00**	**94.020,02**
3	Lehmann	49.000,00		-
4	Schulze	6.000,00	6.000,00	14.103,00
5	Schultze	5.750,00	5.750,00	13.515,38
6	Schmidt	5.500,00	5.500,00	12.927,75
7	Schmitt	5.270,00	5.250,00	12.340,13
8	Müller	5.000,00		-
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

Sie können die Budgets der ausscheidenden Mitarbeiter einfach 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. Das obige Beispiel ergibt eine Differenz von 0,02.

Eine korrekte Rechnung

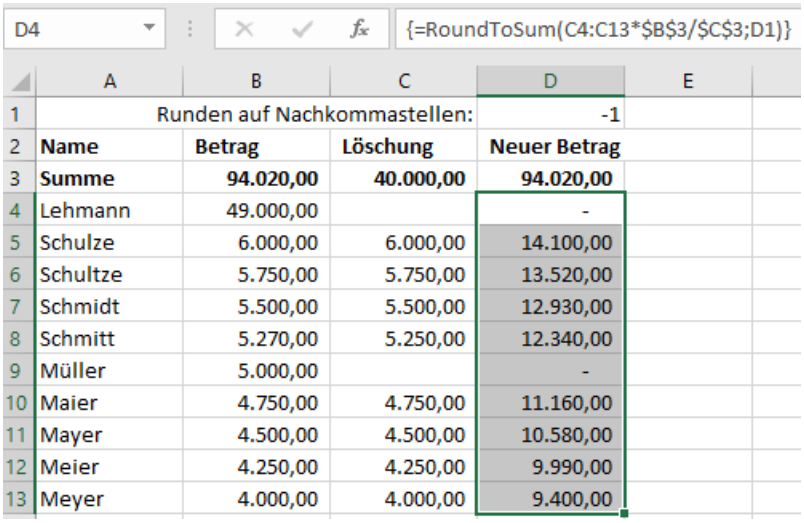
Mit der benutzerdefinierten Funktion *RoundToSum* können Sie die Matrixformel *{=RoundToSum(C4:C13*$B$3/$C$3;D1)}* verwenden:



	A	B	C	D	E
1		Runden auf Nachkommastellen:		2	
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	14.103,00	
6	Schultze	5.750,00	5.750,00	13.515,37	
7	Schmidt	5.500,00	5.500,00	12.927,75	
8	Schmitt	5.270,00	5.250,00	12.340,12	
9	Müller	5.000,00		-	
10	Maier	4.750,00	4.750,00	11.164,88	
11	Mayer	4.500,00	4.500,00	10.577,25	
12	Meier	4.250,00	4.250,00	9.989,63	
13	Meyer	4.000,00	4.000,00	9.402,00	

RoundToSum muss manchmal in die ‘falsche’ Richtung runden, aber dann wird dies wenigstens mit dem kleinstmöglichen Fehler getan.

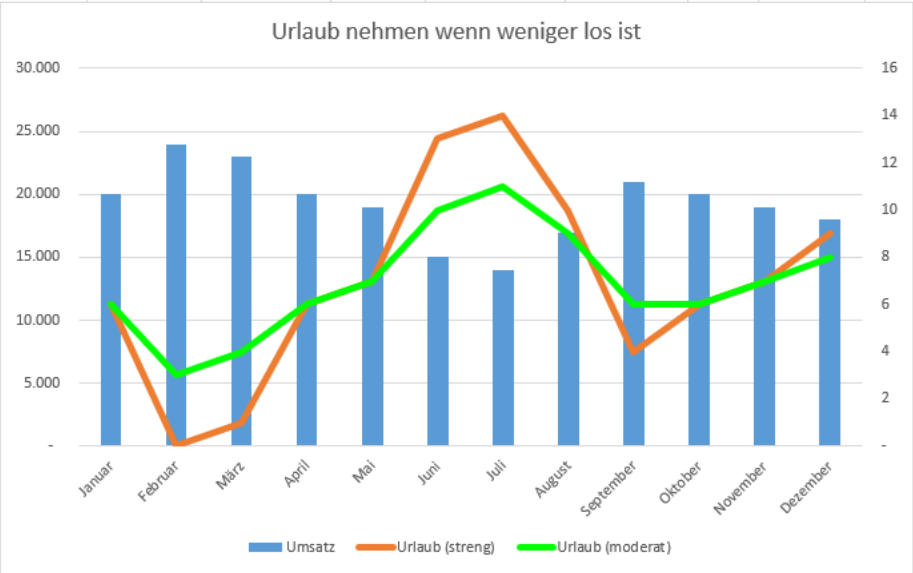
Darüberhinaus können Sie mit *RoundToSum* auch auf andere Dezimalstellen runden, z. B. auf eine Vorkommastelle:



	A	B	C	D	E
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	

Urlaub nehmen wenn weniger los ist

Wenn Ihr Geschäft saisonal stark schwankt, können Sie den Urlaub Ihrer Belegschaft entsprechend planen und ggf. Saisonarbeitskräfte anstellen:



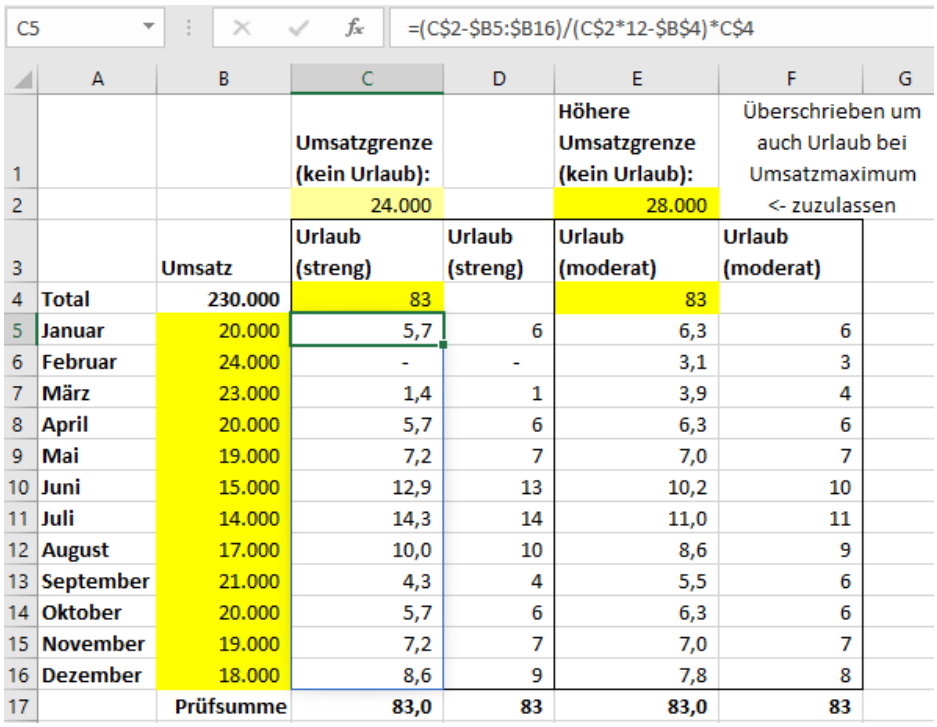
Hinweis: Selbstverständlich kann man keinem Mitarbeiter vorschreiben, wieviel Urlaub wann genommen werden muss. Diese Rechnungen sind lediglich Vorschläge, die als vernünftige Indikatoren dienen sollen.

Einfaches 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:

	Umsatz	Ganze Urlaubstage
Total	**230.000**	**83**
Januar	20.000	6
Februar	24.000	-
Mä rz	23.000	1
April	20.000	6
Mai	19.000	7
Juni	15.000	13
Juli	14.000	14
August	17.000	10
September	21.000	4
Oktober	20.000	6
November	19.000	7
Dezember	18.000	9

Die Formel in Zelle C5, die in Excel 2021 / 365 bis hinunter zu Zelle B16 läuft:
=($C$2-B5:B16)/($C$2*12-$B$4)*$C$4:



C5 | =(C$2-$B5:$B16)/(C$2*12-$B$4)*C$4

	A	B	C	D	E	F	G
1			**Umsatzgrenze (kein Urlaub):**		**Höhere Umsatzgrenze (kein Urlaub):**	Überschrieben um auch Urlaub bei Umsatzmaximum	
2			24.000		28.000	<- zuzulassen	
3		**Umsatz**	**Urlaub (streng)**	**Urlaub (streng)**	**Urlaub (moderat)**	**Urlaub (moderat)**	
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**	

Komplexeres Beispiel

Falls Sie alle Mitarbeiter einzeln berücksichtigen müssen, auch hinsichtlich ihrer Anwesenheitsmonate:

Die Formel in Zelle E21 lautet:
*=WENNFEHLER((E$5:E$16="x")*E$17*$D$5:$D$16/SUMME((E$5:E$16="x")*$D$5:$D$16);0)*

In der letzten Tabelle kommt *RoundToSum* zum Einsatz, um summenerhaltend auf ganze Urlaubstage zu runden. Die Formel in Zelle E37 lautet:
=MTRANS(WENNFEHLER(RoundToSum(E21:E32;0);0))

E21 =WENNFEHLER((E$5:E$16="x")*E$17*$D$5:$D$16/SUMME((E$5:E$16="x")*$D$5:$D$16);0)

	A	B	C	D	E	F	G	H	I	J	K
1			Umsatzgrenze (kein Urlaub):								
2			24.000	<- Höhere Werte erlauben auch Urlaub bei Maximalumsatz							
3		Umsatz	Urlaub	Urlaub (ganze Tage)	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			

Zuweisen von Arbeitseinheiten vermindert um geleistete

Wie können Sie Ihren Mitarbeitern Arbeitseinheiten fair zuweisen, wenn Sie bereits geleistete Arbeit berücksichtigen wollen?

	A	B	C	D	E
1	**Total units still to do**	86			
2	**Total**	90,6	86	86	86
3	**Lecturer**	**Units done**	**Helper**	**Units to do (Formulas)**	**Units to do (RoundToSum)**
4	**Fair share**	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

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 sollen 28 Lehrern noch zugewiesen werden. Ein fairer Anteil wäre für jeden Lehrer *(90.6 + 86) / 28 = 6,3*. Aber 7 Lehrer haben bereits mehr als das geliefert.

Die Kernformel befindet sich in Zelle C5:
=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))

Bitte beachten Sie, dass der faire Anteil absichtlich in Zelle B4 steht und dass C4 leer ist, damit diese Formel unverändert nach unten kopiert werden kann. Relative Referenzen passen sich dabei automatisch an.

Spalte C zeigt die Ergebnisse mit Nachkommastellen. In Spalte D wurde mit einfachen Tabellenblatt-funktionen 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 benutzer-definierten Funktion *RoundToSum* erstellt.

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 unrounded	RoundToSum	Cascading Round	Simple Round & Amend Last	Simple Round	Difference II - V	Difference III - V	Difference IV - V
3		Summands	678,6474579	678,65	678,65	678,65	678,65			
4			146,7029479	146,70	146,70	146,7	146,7			
5			808,4307786	808,43	808,43	808,43	808,43			
6			878,0595004	878,06	878,06	878,06	878,06			
7			801,0013684	801,00	801,00	801	801			
8			895,2150029	895,21	895,22	895,22	895,22	-0,01		
9			55,04805448	55,05	55,05	55,05	55,05			
10			57,68633069	57,69	57,68	57,69	57,69		-0,01	
11			740,3151284	740,31	740,32	740,32	740,32	-0,01		
12			437,4782795	437,48	437,47	437,48	437,48		-0,01	
13			185,281457	185,28	185,29	185,28	185,28		0,01	
14			950,9552226	950,96	950,95	950,96	950,96		-0,01	
15			692,0965454	692,10	692,10	692,1	692,1			
16			147,1681062	147,17	147,17	147,17	147,17			
17			237,137552	237,14	237,13	237,14	237,14		-0,01	
18			487,2154213	487,22	487,22	487,22	487,22			
19			364,4641508	364,46	364,46	364,46	364,46			
20			525,2537907	525,25	525,26	525,25	525,25		0,01	
21			186,8746365	186,87	186,87	186,87	186,87			
22			731,7332769	731,73	731,74	731,73	731,73		0,01	
23			629,6751693	629,67	629,67	629,68	629,68	-0,01	-0,01	
24			76,5434454	76,54	76,54	76,54	76,54			
25			796,2709821	796,27	796,27	796,27	796,27			
26			718,8760902	718,88	718,88	718,88	718,88			
27			822,8369312	822,84	822,84	822,84	822,84			
28			816,4265379	816,43	816,42	816,43	816,43		-0,01	
29			815,4299402	815,43	815,43	815,43	815,43			
30			925,4513501	925,45	925,46	925,45	925,45		0,01	
31			130,5991436	130,60	130,59	130,6	130,6		-0,01	
32			743,1380489	743,14	743,14	743,14	743,14			
33			397,6661651	397,67	397,67	397,67	397,67			
34			759,5541378	759,55	759,55	759,55	759,55			
35			517,7971853	517,80	517,80	517,8	517,8			
36			668,9198847	668,92	668,92	668,92	668,92			
37			927,6280481	927,63	927,63	927,63	927,63			
38			690,0299826	690,03	690,03	690,03	690,03			
39			10,44383544	10,44	10,44	10,44	10,44			
40			994,0458854	994,05	994,05	994,05	994,05			
41			719,3612933	719,36	719,36	719,36	719,36			
42			871,3192169	871,32	871,32	871,29	871,32			-0,03
43		Total	23038,77828	23.038,78	23.038,78	23.038,78	23.038,81	-0,03	-0,03	-0,03
44	ABS Difference to Original			0,11	0,15	0,14				

Bitte beachten Sie, dass die Formel in C3 beim hintereinandergeschalteten Runden die Titelzeile beinhaltet, damit sie einfach herunterkopiert werden konnte.

Wie man sieht, erhalten wir einen aggregierten Rundungsfehler in Höhe von *-0,03*, wenn wir alle Werte einfach einzeln runden. Spalte J (VIII) zeigt die Differenz des aggregierten Rundungsfehlers von *-0,03* 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 11 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 3 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

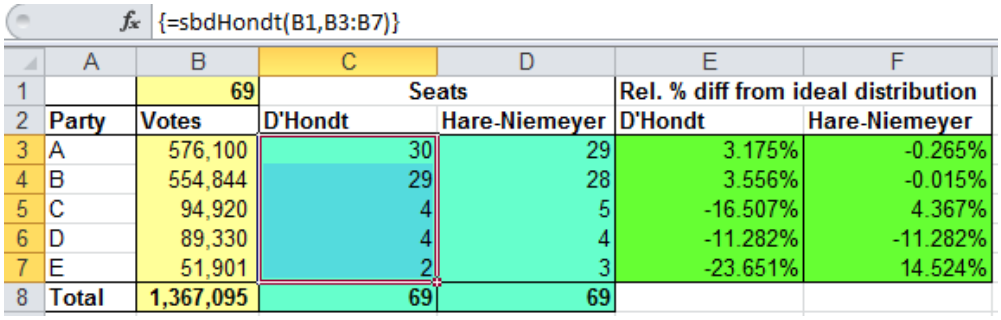
Verwenden Sie *RoundToSum*. Es wendet die geringste Anzahl von Änderungen an, um die korrekte gerundete Gesamtsumme mit dem kleinsten absoluten (oder relativen) Fehler zu erhalten. Eine Matrixformel ist unabdingbar, weil sich alle n > 1 Eingabewerte auf alle n > 1 Ausgabewerte auswirken.

Ein hintereinandergeschaltetes Runden - wie oben in Zelle E6 mit der Formel *=RUNDEN(SUMME($C$5:$C5);2)-SUMME($E$4:$E4)* gezeigt - benötigt kein VBA und auch keine Matrixformel, aber 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 zu sbDHondt

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 geringer:



	A	B	C	D	E	F
1		69	Seats		Rel. % diff from ideal distribution	
2	Party	Votes	D'Hondt	Hare-Niemeyer	D'Hondt	Hare-Niemeyer
3	A	576,100	30	29	3.175%	-0.265%
4	B	554,844	29	28	3.556%	-0.015%
5	C	94,920	4	5	-16.507%	4.367%
6	D	89,330	4	4	-11.282%	-11.282%
7	E	51,901	2	3	-23.651%	14.524%
8	Total	1,367,095	69	69		

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.
'Source (EN): http://www.sulprobil.de/sbdhondt_en/
'Source (DE): http://www.berndplumhoff.de/sbdhondt_de/
'(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).