Rounding Values Preserving Their Sum

Bernd Plumhoff

16-Aug-2025

Abstract

Rounded values do not always sum up to their original total, as demonstrated in this article. How can you ensure that the sum of rounded percentages equals exactly 100%? Is it possible to guarantee that, for accounting purposes, the distribution of overhead costs precisely matches the original total? These challenges are well-known and have been studied extensively.

This article introduces a simple solution using Excel/VBA. The function presented here can round relative values (e.g., percentages) to ensure they sum to exactly 100%. It can also round absolute values (such as cost distributions) while preserving their original sum after rounding. A key parameter allows users to choose which type of error to minimize – absolute error or relative error – compared to the common half-up rounding method.

Contents

List of Figures

1 Rounding Values Preserving Their Sum

If you need to round values without changing their rounded sum, you might need to round one or more summands to the more distant rounded value.

1.1 Percentage Example

For example, the values 11, 45, and 555, which sum to 611, do not yield a percentage total of 100.00 but rather 99.99 if rounded to two decimal places. The **bold** values in non-sum cells have been adjusted using the `RoundToSum` function:

	Values	Percentage rounded to 2 decimals	Minimize absolute Error	Minimize relative Error
	11	1.80	1.80	1.80
	45	7.36	**7.37**	7.36
	555	90.83	90.83	**90.84**
Sum	611	99.99	100.00	100.00

The Excel / VBA function call `RoundToSum({11,45,555},2,FALSE,1)` would result in {1.80, 7.37, 90.83}, though. Here, the percentage value 7.364975 is rounded differently to achieve a percentage sum of 100.00 and to minimize the absolute error compared to half-up rounding. By using `RoundToSum({11,45,555},2,FALSE,2)` we would have received {1.80, 7.36, 90.84}, as this would minimize the relative error.

1.2 Example with Absolute Values

The sum of the second column differs by +2,000 from the rounded sum. The **bold** values in non-sum cells have been adjusted using the `RoundToSum` function:

	Values	Rounded to absolute 1,000	Minimize absolute Error	Minimize relative Error
	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
Sum	681.709	684.000	682.000	682.000

1.3 The User-Defined VBA Function `RoundToSum`

Name

`RoundToSum` – Rounding values preserving their rounded sum

Synopsis

```
RoundToSum(vInput, [lDigits], [bAbsSum], [lErrorType])
```

Description

`RoundToSum` rounds values without altering their rounded sum. It uses the largest remainder method to minimize the error compared to the commonly used half-up rounding method. If the error is identical for one or more values, the first value(s) encountered will be adjusted.

Note: This solution is limited to one-dimensional tables without subtotals. There is no general solution for higher-dimensional tables or tables with subtotals.

Parameters

`vInput` — Range or array containing the unrounded input values.

`lDigits` — Optional, default value is 2. The number of digits to round to. For example: 0 rounds to integers, 2 rounds to the nearest cent, -3 rounds to the nearest thousand.

`bAbsSum` — Optional, default value is TRUE. TRUE rounds the values directly which you often need for accounting calculations. FALSE adjusts the percentages so they sum to exactly 100%. This is frequently used in presentations of percentage distributions.

`lErrorType` — Optional, default value is 1. The type of error to minimize: 1 for absolute error, 2 for relative error. The absolute error you normally minimize for values you need to book in general ledgers. For statistical distributions you often minimize the relative error to avoid amendments in the tails of the distributions.

1.4 RoundToSum Program Code

```
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
```

1.5 Round2Sum Lambda Expression

With three Lambda expressions, we can replace the VBA function `RandToSum` by this `Round2Sum` Lambda expression:

```
=LAMBDA(vI,lD,bA,lE,
    LET(
        i,IF(bA,vI,vI/SUM(vI)%),
        r,ROUND(i,lD),
        _C,ROUND(SUM(i),lD)-SUM(r),
        _E,CHOOSE(lE,r-i,(r-i)*i),
        _R, UniqRank(_E,IF(_C>0,1,0)),
        _D,IF(_R<=ROUND(ABS(_C*10^lD),0),SGN(_C)*10^-lD,0),
        r+IF(ROWS(r)=1,TRANSPOSE(_D),_D)
    )
)
```

`UniqRank` is defined as:

```
=LAMBDA(Ref,[Order],
    LET(
        _ord,IF(ISOMITTED(Order),-1,IF(Order=0,-1,1)),
        _r,INDEX(IF(ROWS(Ref)=1,TRANSPOSE(Ref),Ref),,1),
        _c,ROWS(_r),
        _i,SEQUENCE(ROWS(_r)),
        INDEX(SORT(HSTACK2(_i,INDEX(SORT(HSTACK2(_r,_i),,_ord),,2)),2,1),,1)
    )
)
```

And – since Excel's worksheet function `HSTACK` only accepts ranges, not arrays – `HSTACK2` as:

```
=LAMBDA(a,b,
    MAKEARRAY(
        ROWS(a),
        2,
        LAMBDA(r,c,
            IF(c=1,INDEX(a,r),INDEX(b,r))
        )
    )
)
```

2 Rounding Values Alters Their Sum

How likely is it that a sum of rounded values is not identical to their rounded sum?

For two random floating point numbers this is obvious: The likelihood is around 25% - that is the percentage of red in this picture:

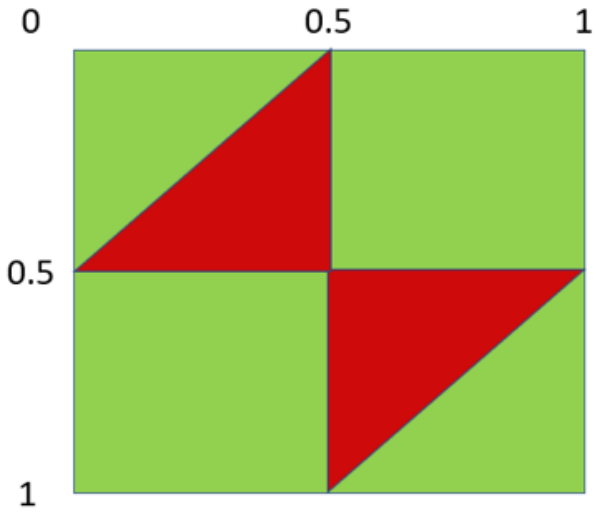


Figure 1: Rounding two random numbers

But it might be somewhat surprising that the likelihood approaches 90% if you round and add more and more numbers:

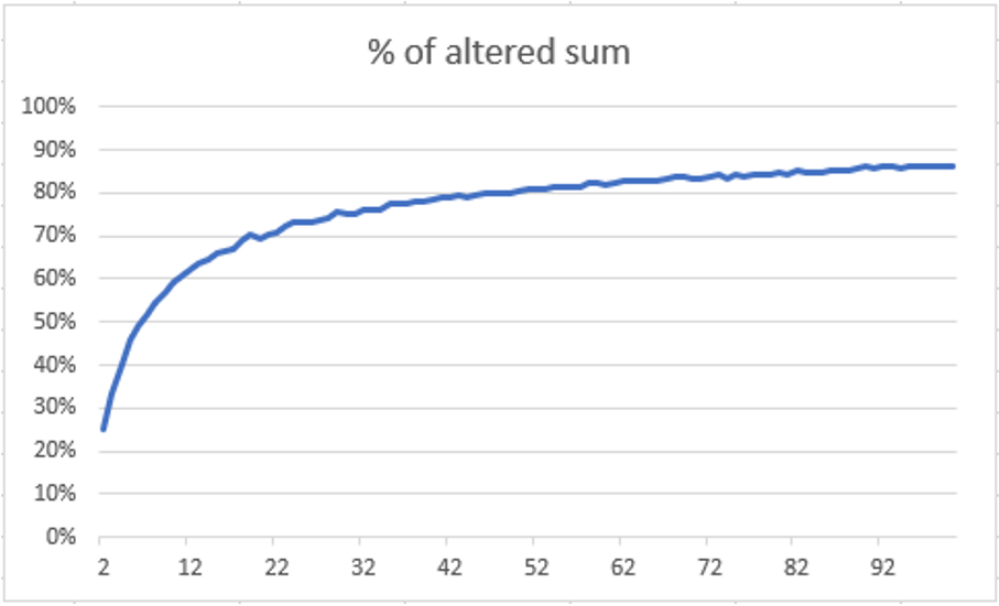


Figure 2: Percentage altered Sums of random Numbers

With seven floating point numbers the likelihood is already larger than 50% that the sum of rounded values is not equal to their rounded sum.

2.1 Rounded Percentages

Rounded percentages also often fail to add up to 100%. With two random numbers the issue arises only if both numbers equal 0.5:

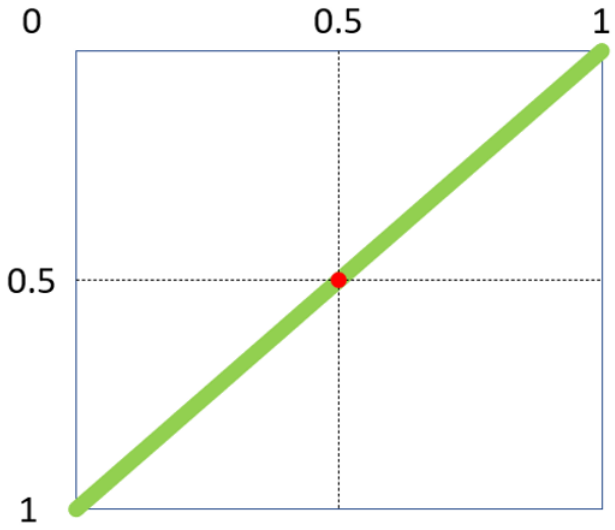


Figure 3: Rounding two random Percentages

But with more random numbers it is similar to the problem stated initially, just with around one number more. Rounded percentages of three arbitrary numbers fail to add up to 1 with a chance of around 25%:

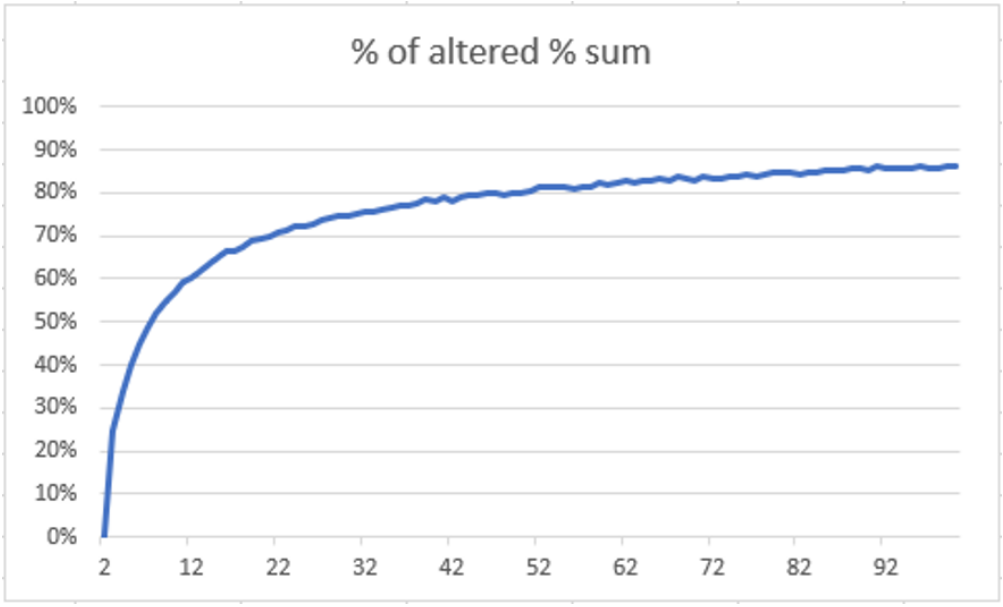


Figure 4: Percentage altered Percentage Sums of random Numbers

2.2 Monte Carlo Code

```
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 + eFehler! Textmarke nicht definiert.(k)
    Next k
    s2 = 0#
    For k = 1 To i
      e(k) = .Round(1000# * e(k) / s1, 0)
      s2 = s2 + (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
```

3 Usage Examples of `RoundToSum`

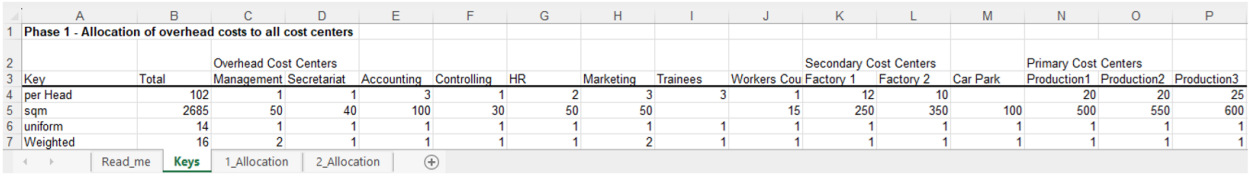
3.1 Allocation of Overheads

When allocating overhead costs to products you often encounter the fact that the resulting sum of allocated overheads does not equal the original cost sum. Due to rounding differences you frequently face a little cent difference. In this case the user defined function `RoundToSum` can help.

A Real-Life Example

We present an allocation of overheads where all individual cent values accurately add up to their intermediate or final sums.

First you define how the overheads have to be allocated to support cost centres (sheet 'Keys'):

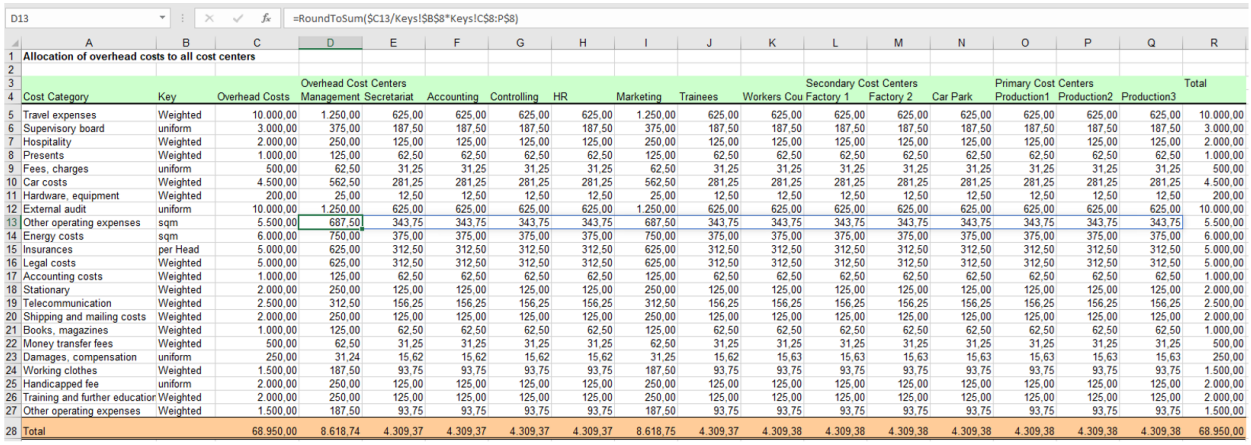


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**Phase 1 - Allocation of overhead costs to all cost centers**															
2			Overhead Cost Centers								Secondary Cost Centers			Primary Cost Centers		
3	Key	Total	Management	Secretariat	Accounting	Controlling	HR	Marketing	Trainees	Workers Cou	Factory 1	Factory 2	Car Park	Production1	Production2	Production3
4	per Head	102	1	1	3	1	2	3	3	1	12	10		20	20	25
5	sqm	2685	50	40	100	30	50	50		15	250	350	100	500	550	600
6	uniform	14	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	Weighted	16	2	1	1	1	1	2	1	1	1	1	1	1	1	1

Read_me | Keys | 1_Allocation | 2_Allocation

Figure 5: Allocation of Overheads - Keys Overheads

The first allocation of overheads uses a rounding correction so that all summands accurately sum up on support cost centre level (sheet '1_Allocation'):



D13 =RoundToSum($C13/Keys!$B$8*Keys!C$8:P$8)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	**Allocation of overhead costs to all cost centers**																	
2																		
3				Overhead Cost Centers								Secondary Cost Centers			Primary Cost Centers			Total
4	Cost Category	Key	Overhead Costs	Management	Secretariat	Accounting	Controlling	HR	Marketing	Trainees	Workers Cou	Factory 1	Factory 2	Car Park	Production1	Production2	Production3	
5	Travel expenses	Weighted	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	Supervisory board	uniform	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	Hospitality	Weighted	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	Presents	Weighted	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	Fees, charges	uniform	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	Car costs	Weighted	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	Hardware, equipment	Weighted	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	External audit	uniform	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	Other operating expenses	sqm	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	Energy costs	sqm	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	Insurances	per Head	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	Legal costs	Weighted	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	Accounting costs	Weighted	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	Stationary	Weighted	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	Telecommunication	Weighted	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	Shipping and mailing costs	Weighted	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	Books, magazines	Weighted	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	Money transfer fees	Weighted	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	Damages, compensation	uniform	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	Working clothes	Weighted	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	Handicapped fee	uniform	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	Training and further education	Weighted	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	Other operating expenses	Weighted	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	Total		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

Worksheet Formulas		
Range	Formula	
D5:D27	**D5**	=RoundToSum($C5/Keys!$B$7*Keys!C$7:P$7)
C28:Q28	**C28**	=SUM(C5:C27)
R5:R28	**R5**	=SUM(D5:Q5)
R30	**R30**	=R28-C28

Figure 6: Allocation of Overheads - Overhead Costs

The second allocation of overheads (sheet 'Keys') also uses a rounding correction so that all support cost centres get accurately distributed to products:

	A	B	C	D	E	F	G	H
10	**Phase 2 - Allocation of Overhead Cost Centers and Secondary Cost Centers to Primary Cost Centers**							
11								
12								
13	Secondary Cost Center	Key	Production1	Production2	Production3	Total		
14								
15	Management	Weighted	30%	40%	30%	100%		
16	Secretariat	Weighted	40%	50%	10%	100%		
17	Accounting	Weighted	30%	13%	57%	100%		
18	Controlling	uniform	1	1	1	3		
19	HR	per Head	20	20	25	65		
20	Marketing	Weighted	30%	42%	28%	100%		
21	Trainees	uniform	1	1	1	3		
22	Workers Council	per Head	20	20	25	65		
23	Factory 1	Weighted	25%	20%	55%	100%		
24	Factory 2	Weighted	20%	20%	60%	100%		
25	Car Park	Weighted	40%	30%	30%	100%		

Read_me | **Keys** | 1_Allocation | 2_Allocation

Figure 7: Allocation of Overheads - Keys Support Cost Centres

The final result (sheet '2_Allocation'):

	A	B	C	D	E	F	G	H
1	**Allocation of Overhead Cost Centers and Secondary Cost Centers to Primary Cost Centers**							
2								
3	**Allocated Cost Centers**	**Direct Costs**	**Allocation Phase 1**	**Total**	**Production1**	**Production2**	**Production3**	**Total**
4								
5	Management	111.666,00	8.618,75	120.284,75	36.085,42	48.113,90	36.085,43	120.284,75
6	Secretariat	34.627,00	4.309,37	38.936,37	15.574,55	19.468,18	3.893,64	38.936,37
7	Accounting	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	HR	53.765,00	4.309,37	58.074,37	17.869,04	17.869,04	22.336,29	58.074,37
10	Marketing	239.170,00	8.618,75	247.788,75	74.336,62	104.071,28	69.380,85	247.788,75
11	Trainees	147.397,00	4.309,37	151.706,37	50.568,79	50.568,79	50.568,79	151.706,37
12	Workers Council	471,00	4.309,37	4.780,37	1.470,88	1.470,88	1.838,61	4.780,37
13	Factory 1	125.225,00	4.309,38	129.534,38	32.383,59	25.906,88	71.243,91	129.534,38
14	Factory 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	Car Park	26.992,00	4.309,38	31.301,38	12.520,55	9.390,42	9.390,41	31.301,38
16	Phase 1 Allocation				4.309,38	4.309,38	4.309,38	12.928,14
17	Phase 2 Allocation	3.318.534,00	56.021,86	3.374.555,86	781.111,51	799.967,08	1.793.477,27	3.374.555,86
18	Directs Costs				738.060,00	854.000,00	650.360,00	2.242.420,00
19	Total Primary Cost Centers				1.523.480,89	1.658.276,46	2.448.146,65	5.629.904,00
20								
21	Overhead rate				106,4%	94,2%	276,4%	151,1%
22								
23							Check	0,00

Worksheet Formulas		
Range	Formula	
C5:C15	**C5**	=TRANSPOSE('1_Allocation'!D28:N28)
D5:D15	**D5**	=SUM(B5:C5)
E5:E15	**E5**	=RoundToSum($D5/Keys!$F15*Keys!C15:E15)
H5:H16	**H5**	=SUM(E5:G5)
E16	**E16**	='1_Allocation'!O28
B17:H17	**B17**	=SUM(B5:B15)
E19:H19	**E19**	=SUM(E16:E18)
E21:H21	**E21**	=(E17+E16)/E18
H23	**H23**	=H19-SUM(E18:G18)-SUM(B5:B15)-SUM('1_Allocation'!C5:C27)

Figure 8: Allocation of Overheads - Support Cost Centres

This correct allocation of overheads you will be able to enter into a general ledger without any cent / penny difference.

3.2 Exact Relation of Random Numbers - `sbExactRandHistogrm`

It is fairly easy to create a loaded die, let us say on average the 6 should appear twice as often as all the other numbers 1 thru 5: Enter into A1: `=MIN(INT(RAND()*7+1),6)`

But what if you want to create 7 rolls of this die and all numbers between 1 and 5 should appear exactly once and 6 exactly twice?

Here is a general solution:

	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	Yellow	Green	Red	Green	Yellow	3	2	1
4	Yellow	33,33%	2	Yellow	Yellow	Yellow	Green	Green	Red	2	3	1
5	Red	16,67%	3	Yellow	Green	Red	Red	Green	Yellow	2	2	2
6			4	Green	Green	Yellow	Green	Red	Green	4	1	1
7			5	Green	Green	Red	Yellow	Yellow	Yellow	2	3	1
8			6	Yellow	Yellow	Yellow	Yellow	Green	Yellow	1	5	0
9			7	Green	Red	Green	Green	Yellow	Yellow	3	2	1
10			8	Green	Green	Green	Yellow	Green	Red	4	1	1
11			9	Yellow	Green	Green	Yellow	Green	Green	4	2	0
12			10	Green	Red	Yellow	Green	Red	Green	3	1	2
13									Total:	28	22	10
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	Green	Red	Green	Yellow	Yellow	3	2	1
19			2	Green	Yellow	Red	Yellow	Green	Green	3	2	1
20			3	Yellow	Green	Green	Red	Green	Yellow	3	2	1
21			4	Green	Yellow	Green	Green	Red	Yellow	3	2	1
22			5	Green	Yellow	Green	Red	Green	Yellow	3	2	1
23			6	Green	Green	Green	Yellow	Red	Yellow	3	2	1
24			7	Yellow	Green	Red	Yellow	Green	Green	3	2	1
25			8	Green	Yellow	Green	Yellow	Red	Green	3	2	1
26			9	Yellow	Yellow	Green	Green	Green	Red	3	2	1
27			10	Green	Yellow	Green	Yellow	Red	Green	3	2	1
28									Total:	30	20	10
29									Should stochastically be:	30	20	10

Figure 9: sbExactRandHistogrm

Worksheet Formulas		
Range	Formula	
D3:I12	**D3**	=INDEX($A$3:$A$5,INT(sbRandHistogrm(1,4,$B$3:$B$5)))
J3:L12;J18:L27	**J3**	=COUNTIF($D3:$I3,J$2)
J13:L13;J28:L28	**J13**	=SUM(J3:J12)
J14;J29	**J14**	=COUNTA($D$3:$I$12)*TRANSPOSE($B$3:$B$5)
D18:D27	**D18**	=INDEX($A$3:$A$5,INT(sbExactRandHistogrm(6,1,4,$B$3:$B$5)))

Figure 10: sbExactRandHistogrm Formulas

The User-Defined VBA Function `sbExactRandHistogrm`

Name

`sbExactRandHistogrm` – Create an exact double histogram distribution.

Synopsis

sbExactRandHistogrm(ldraw, dmin, dmax, vWeight)

Description

`sbExactRandHistogrm` creates an exact histogram distribution for ldraw draws of floating point numbers with double precision 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 `RoundToSum` - see Appendix A.

Parameters

ldraw - Number of draws

dmin - Minimum = lower boundary of range of numbers to draw

dmax - Maximum = upper boundary of range of numbers to draw

vWeight - Array of weights. Array size determines the number of different classes the range dmin : dmax is divided into. Values in this array specify likelihood of this class' numbers to appear (be drawn).

Program Code `sbExactRandHistogrm`

```
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
```

3.3 Take Vacation When Less is Going on

If your business fluctuates strongly seasonally, you can plan the vacation of your staff accordingly and consider hiring seasonal staff:

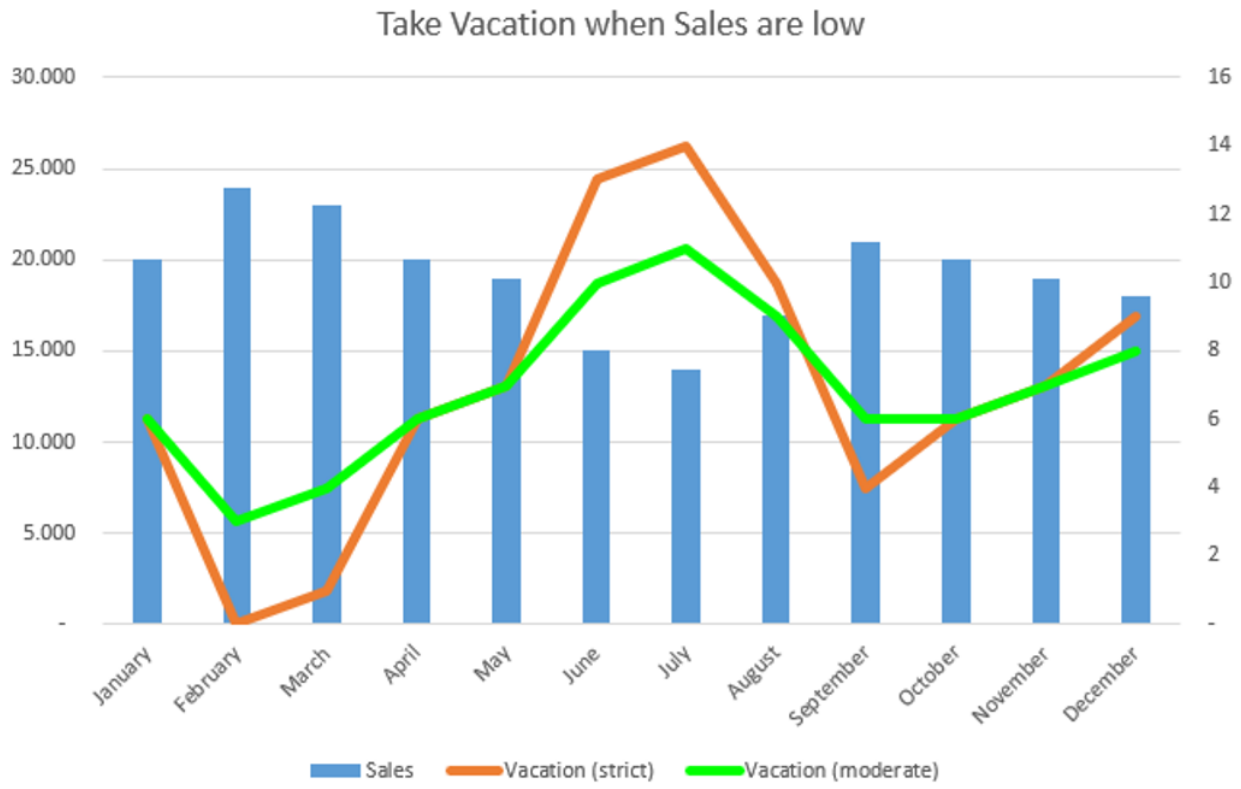


Figure 11: Vacation Chart

Note: Of course you cannot force anybody when to take a vacation and how many days are to be taken. These calculations are just meant to be suggestions of reasonable indicators.

3.3.1 Simple Example

If you like to take the maximum sales values (here: 24,000) as a basis, applying zero vacations to it, and scale the vacation days linearly to the other sales values:

	A	B	C	D	E	F	G
1			Sales Limit for no vacation:		Higher Sales Limit for no vacation:	Increased to allow for some vacation at sales maimum.	
2			24.000		28.000		
3		Sales	Vacation (strict)	Integers	Vacation (moderate)	Integers	
4	Total	230.000	83		83		
5	January	20.000	5,7	6	6,3	6	
6	February	24.000	-	-	3,1	3	
7	March	23.000	1,4	1	3,9	4	
8	April	20.000	5,7	6	6,3	6	
9	May	19.000	7,2	7	7,0	7	
10	June	15.000	12,9	13	10,2	10	
11	July	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	October	20.000	5,7	6	6,3	6	
15	November	19.000	7,2	7	7,0	7	
16	December	18.000	8,6	9	7,8	8	
17	Checksum Vacation		83,0	83	83,0	83	

Worksheet Formulas		
Range	Formula	
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	=SUM(C5:C16)

Figure 12: Vacation - Simple Example

3.3.2 More Complex Example

If you got employees who are not present at specified months – `RoundToSum` rounds to whole vacation days in the last table:

	A	B	C	D	E	F	G	H
1			Sales Limit for no vacation:		Overwrite with higher value to allow for vacation in month with max sales			
2			24.000					
3		Sales	Vacation days (fractional)	Vacation days (integer)	Vacation Claim			
4	Total	230.000			Andrew	Benjamin	Charlie	David
5	January	20.000	5,7	6	x	x		x
6	February	24.000	-	-	x	x		x
7	March	23.000	1,4	1	x	x	x	x
8	April	20.000	5,7	6	x	x	x	x
9	May	19.000	7,2	7	x	x	x	
10	June	15.000	12,9	13	x	x	x	
11	July	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	October	20.000	5,7	6	x	x	x	x
15	November	19.000	7,2	7	x		x	x
16	December	18.000	8,6	9	x		x	x
17		Total	83,0	83	25,0	21,0	21,0	16,0
18								
19				Vacation days (fractional)				
20				Total	Andrew	Benjamin	Charlie	David
21			January	6,1	1,8	1,9	-	2,5
22			February	-	-	-	-	-
23			March	1,3	0,3	0,3	0,3	0,4
24			April	7,8	1,8	1,9	1,6	2,5
25			May	6,2	2,1	2,2	1,9	-
26			June	11,5	3,9	4,1	3,5	-
27			July	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			October	7,8	1,8	1,9	1,6	2,5
31			November	6,9	2,1	-	1,9	2,9
32			December	8,9	2,7	-	2,5	3,7
33			Total	83,0	25,0	21,0	21,0	16,0
34								
35				Vacation days (integer)				
36				Total	Andrew	Benjamin	Charlie	David
37			January	7	2	2	-	3
38			February	-	-	-	-	-
39			March	-	-	-	-	-
40			April	8	2	2	2	2
41			May	6	2	2	2	-
42			June	11	4	4	3	-
43			July	13	4	5	4	-
44			August	9	3	3	3	-
45			September	5	1	1	1	2
46			October	8	2	2	2	2
47			November	7	2	-	2	3
48			December	9	3	-	2	4
49			Total	83	25	21	21	16

Worksheet Formulas		
Range	Formula	
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)

Figure 13: Vacation - More complex Example

4 `RoundToSum` Versus Other 'Simple' Methods

There are several different naïve approaches circulating around which try to round values preserving their rounded sum:

- (worst) Round all values but the last one and replace the last one by the rounded original sum minus the sum of the previously rounded values (i.e. aggregate all rounding errors in the last summand):

	A	B	C	
1		**Original Data**	**Aggregate Rounding Error**	**Formula in C**
2	**Total**	**2,594**	**2,59**	=SUM(C4:C8)
3				
4		0,875	0,88	=ROUND(B4,2)
5		0,865	0,87	=ROUND(B5,2)
6		0,344	0,34	=ROUND(B6,2)
7		0,455	0,46	=ROUND(B7,2)
8		0,055	**0,04**	=ROUND(B$2,2)-SUM(C$4:C7)

Figure 14: RoundToSum Comparison - Amend Last

- (better, but still bad) Apply a cascading (sliding) round:

	A	B	C	
1		**Original Data**	**Cascading Round**	**Formula in C**
2	**Total**	2,593	2,59	=SUM(C4:C8)
3				
4		0,875	0,88	=ROUND(SUM($B$3:$B4),2)-SUM($C$3:$C3)
5		0,865	**0,86**	=ROUND(SUM($B$3:$B5),2)-SUM($C$3:$C4)
6		0,344	0,34	=ROUND(SUM($B$3:$B6),2)-SUM($C$3:$C5)
7		0,454	**0,46**	=ROUND(SUM($B$3:$B7),2)-SUM($C$3:$C6)
8		0,055	**0,05**	=ROUND(SUM($B$3:$B8),2)-SUM($C$3:$C7)

Figure 15: RoundToSum Comparison - Cascading Round

Let us compare these approaches to `RoundToSum`.

4.1 Calculation Example

We create 40 random numbers $RAND() * 1000$ and compare as follows:

	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	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 Difference to Original			0,11	0,14	0,15	0,10			

Figure 16: RoundToSum Comparison - Example

Worksheet Formulas		
Range	Formula	
D3	**D3**	=RoundToSum(C3:C42)
E3	**E3**	=ROUND(SUM($C$3:$C3),2)-SUM(E$2:E2)
F3	**F3**	=ROUND(C3:C41,2)
F42	**F42**	=ROUND(C43,2)-SUM(F3:F41)
G3	**G3**	=ROUND(C3:C42,2)
H3:J3	**H3**	=IF(ABS(D3:D42-$G3:$G42)<0.000001,"",D3:D42-$G3:$G42)
C43:J43	**C43**	=SUM(C3:C42)
D44:G44	**D44**	=SUMPRODUCT(ABS(D3:D42-$C$3:$C$42))

Figure 17: RoundToSum Comparison - Example Formulas

As you can see, if we simply round each single number, the resulting sum would differ from the original rounded sum by 0.06. Column J (VIII) shows the difference of the aggregated rounding error -0.06 in the last summand. Column F (IV) shows the corresponding rounded numbers. Worst case would be here to come up with an aggregated rounding error of n * 0,005 with n being the count of your numbers. Example: Take 40 times the number 0.005 instead of the 40 random numbers.

Good practical examples, why you should not aggregate rounding errors in the last summand, are normally distributed samples of integers.

The cascading (sliding) round in column I (VII) shows 12 roundings to the wrong side. Column E (III) shows the corresponding rounded numbers. Worst case would be for the cascading round to round half of your numbers to the wrong side when all numbers could have been rounded correctly. Example: Take 20 times the number -0.0049999 and then 20 times the number 0.0049999 instead of the 40 random numbers.

On the other hand, the optimal `RoundToSum` just rounds 6 values to the wrong side which result in the least number of changes which achieve the correct rounded sum. The worst case would now involve n/2 roundings to the wrong side with n being the count of your numbers. Example: Take 40 times the number 0.005 again instead of the 40 random numbers. This is the best solution with the smallest absolute rounding error for each number and then with the smallest number of roundings to the wrong side.

4.2 Conclusion

Use `RoundToSum`. It will apply the least number of changes and it will result in the correct sum with the smallest absolute (or relative) error.

A cascading round as shown above does not need any VBA nor does it apply any array formula, but it requires at least as many rounding differences as `RoundToSum` but can leave you with much more unnatural roundings which you can hardly explain to any senior manager.

But worst of all is the approach of aggregating all rounding differences in the last summand. Just imagine 1,000 people, each having 49 Cents, adding up to $490, which you should distribute fairly, but rounded to a whole Dollar. In this case you would end up with $490 at the last person, while `RoundToSum` would give the first 490 persons one Dollar each and all the others zero.

5 References

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.

Excel VBA A Collection - Contains this text extract and many other VBA Programs and Excel functions

Excel VBA Eine Sammlung - Contains this text extract and many other VBA Programs and Excel functions IN GERMAN