Zufallszahlen erzeugen mit (Excel / VBA)

Bernd Plumhoff

1. September 2025

Abstract

Zufallszahlen benötigt man oft für Simulationen, für Was-wäre-wenn Rechnungen oder zur Anonymisierung von Daten. Hier stelle ich meine Sammlung von Programmen vor, die Zufallszahlen erzeugen: natürliche Zahlen, ganze Zahlen, oder Gleitkommazahlen. Da ich Excel's eingebaute Zufalls¬funktionen verwende, erzeugen meine Programme Pseudozufallszahlen.

Contents

List of Figures

1 Ganze Zufallszahlen

1.1 Natürliche Zufallszahlen– `UniqRandInt`

Hinweis: Diese Funktion wird lediglich aus historischen Gründen gezeigt, weil sie effizient ausgeführt wird und weil mit ihr VBA Compilerkonstanten gelernt werden können. Die Funktion `sbRandInt` (see below) erlaubt auch negative Untergrenzen und ermittelt die beste Ausführungsart zur Ausführungszeit, nicht während der Compilierung.

Manchmal benötigt man ganze Zufallszahlen, die sich nicht oder lediglich begrenzt häufig wiederholen. Wenn Sie 20 positive ganze Zufallszahlen zwischen 1 und 100 benötigen, geben Sie ein:

```
=UniqRandInt(20;100)
```

Falls der Bereich zwischen 100 und 199 liegen soll, dann

```
=UniqRandInt(20;100)+99
```

Allgemeinl:

```
=UniqRandInt(Anzahl;Endwert - Startwert + 1) + Startwert - 1
```

Wenn Sie 10 natürliche Zufallszahlen im Bereich 1..2 benötigen, die bis zu 10-mal erscheinen dürfen, verwenden Sie

```
=UniqRandInt(10;2;10))
```

In diesem Fall muss die Compilerkostante `ALLOW_REPETITION` auf `True` gesetzt sein. Und falls Sie nur 3 Zahlen zwischen 1 und 100 Millionen brauchen, die aber nicht mehrfach vorkommen dürfen, dann sollte die Compilerkonstante `LATE_INITIALISATION` auf `True` gesetzt sein:

	A	B	C	D	E	F
1	**n**	6	12	10	6	3
2	**IRange**	6	6	2	6	100.000.000
3	**IMaxOccurrence**	1	2	10	1	1
4						
5	**Ergebnis**	5	1	2	1	84.876.019
6		6	5	2	2	39.223.814
7		4	1	2	3	51.271.980
8		3	2	1	6	
9		1	4	1	5	
10		2	6	1	4	
11			5	1		
12			2	1		
13			3	2		
14			6	2		
15			4			
16			3			

Tabellenblattformeln		
Bereich	Formel	
B5:F5	**B5**	=MTRANS(UniqRandInt(B1;B2;B3))

Figure 1: UniqRandInt

Programmcode `UniqRandInt`

```
'If lRange  n then set LATE_INITIALISATION to true. For example,
'if lRange=1,000,000 and if 1,000 cells are selected (n=1000).
#Const LATE_INITIALISATION = True
'If random integers may occur more than once, allow repetitions
#Const ALLOW_REPETITION = True

#If ALLOW_REPETITION Then
Function UniqRandInt(n As Long, ByVal lRange As Long, _
  Optional lMaxOccurence As Long = 1) As Variant
#Else
Function UniqRandInt(n As Long, ByVal lRange As Long) As Variant
#End If
'Returns n unique (=non-repeating) random integers within 1..lRange,
'lRange >= n. Set ALLOW_REPETITION = True and call with
'lMaxOccurences > 1 if random integers may occur more than once.
'(C) (P) by Bernd Plumhoff 30-Oct-2024 PB V1.04

Static bRandomized As Boolean
Dim vA             As Variant
Dim vR             As Variant
Dim i              As Long
Dim j              As Long
Dim lr             As Long

If Not bRandomized Then Randomize: bRandomized = True

#If ALLOW_REPETITION Then
  If lMaxOccurence < 1 Then
    UniqRandInt = CVErr(xlErrNum)
    Exit Function
  End If
  lRange = lRange * lMaxOccurence
#End If

If n > lRange Then UniqRandInt = CVErr(xlErrValue): Exit Function

ReDim vR(1 To n) As Variant

ReDim vA(1 To lRange)
#If Not LATE_INITIALISATION Then
  For i = 1 To lRange
    #If ALLOW_REPETITION Then
      vA(i) = Int((i - 1) / lMaxOccurence) + 1
    #Else
      vA(i) = i
    #End If
  Next i
#End If
```

```
i = 1
For j = 1 To UBound(vR, 1)
  lr = Int(((lRange - i + 1) * Rnd) + 1)
  #If LATE_INITIALISATION Then
    If vA(lr) = 0 Then
      #If ALLOW_REPETITION Then
        vR(j) = Int((lr - 1) / lMaxOccurence) + 1
      #Else
        vR(j) = lr
      #End If
    Else
  #End If
      vR(j) = vA(lr)
  #If LATE_INITIALISATION Then
    End If
    If vA(lRange - i + 1) = 0 Then
      #If ALLOW_REPETITION Then
        vA(lr) = Int((lRange - i + 1 - 1) / lMaxOccurence) + 1
      #Else
        vA(lr) = lRange - i + 1
      #End If
    Else
  #End If
      vA(lr) = vA(lRange - i + 1)
  #If LATE_INITIALISATION Then
    End If
  #End If
  i = i + 1
Next j

UniqRandInt = vR
End Function
```

1.2 Ganze Zufallszahlen – `sbRandInt`

Falls Sie ganzzahlige Zufallszahlen zwischen zwei gegebenen Werten benötigen, die sich nicht oder lediglich begrenzt häufig wiederholen, dann empfehle ich, meine benutzerdefinierte Funktion `sbRandInt` zu verwenden:

Anmerkung: Wenn der mögliche Zufallszahlenbereich wesentlich größer ist als die Anzahl zu erzeugenden Zufallszahlen, dann initialisiert `sbRandInt` seine Arrays zur Ausführungszeit verzögert, während man bei `UniqRandInt` eine verzögerte Initialisierung mit einer Compilerkonstanten vor Programmausführung setzen muss.

	A	B	C	D	E	F
1	lCount	7	14	10	12	3
2	lMin	-3	-3	1	1	-100.000.000
3	lMax	3	3	2	3	100.000.000
4	lRept	1	2	10	4	1
5						
6	Ergebnis	-1	-2	2	2	18.993.127
7		-2	0	1	3	-72.571.540
8		2	-1	2	1	-9.510.840
9		-3	2	1	3	
10		3	-3	2	1	
11		0	-3	1	1	
12		1	2	1	2	
13			1	2	2	
14			3	1	3	
15			3	2	1	
16			-1		2	
17			-2		3	
18			0			
19			1			

Tabellenblattformeln		
Bereich	Formel	
B6:F6	**B6**	=MTRANS(sbRandInt(B1;B2;B3;B4))

Figure 2: sbRandInt

Programmcode `sbRandInt`

```
Function sbRandInt(ByVal lCount As Long, _
  lMin As Long, _
  lMax As Long, _
  Optional lRept As Long = 1) As Variant
'Returns lCount random integers between lMin and lMax, each one
'occurring zero to lRept times. lMax - lMin + 1 must be greater
'or equal to lCount.
'Error values:
'#NUM!   - lRept is less than 1
'#REF!   - lCount is greater than (lMax - lMin + 1) * lRept
'#VALUE! - lCount is less than 1
'(C) (P) by Bernd Plumhoff  30-Dec-2024 PB V1.02
Static bRandomized As Boolean
Dim i As Long, j As Long, k As Long
Dim lRnd As Long, lRange As Long
Const CLateInitFactor = 50

If lCount < 1 Then sbRandInt = CVErr(xlErrValue): Exit Function
If lRept < 1 Then sbRandInt = CVErr(xlErrNum): Exit Function
If lCount > (lMax - lMin + 1) * lRept Then
  sbRandInt = CVErr(xlErrRef)
  Exit Function
End If
```

```
lRange = (lMax - lMin + 1) * lRept

ReDim lr(1 To lCount) As Long

If Not bRandomized Then Randomize: bRandomized = True

ReDim lT(1 To lRange) As Long
'If we have a huge range of possible random integers and a comparably
'small number of draws, i.e. if (lMax - lMin) * lRept >> lCount
'then we can save some runtime with late initialization.
If lRange / lCount < CLateInitFactor Then
  For i = 1 To lRange
    lT(i) = Int((i - 1) / lRept) + lMin
  Next i
End If

i = 1
If lRange / lCount < CLateInitFactor Then
  For k = 1 To UBound(lr)
    lRnd = Int(((lRange - i + 1) * Rnd) + 1)
    lr(k) = lT(lRnd)
    lT(lRnd) = lT(lRange - i + 1)
    i = i + 1
  Next k
Else
  j = lMin: If lMin <= 0 And lMax >= 0 Then j = 1
  For k = 1 To UBound(lr)
    lRnd = Int(((lRange - i + 1) * Rnd) + 1)
    If lT(lRnd) = 0 Then
      lr(k) = Int((lRnd - 1) / lRept) + j
    Else
      lr(k) = lT(lRnd)
    End If
    If lT(lRange - i + 1) = 0 Then
      lT(lRnd) = Int((lRange - i) / lRept) + j
    Else
      lT(lRnd) = lT(lRange - i + 1)
    End If
    i = i + 1
  Next k
'If lRange includes zero we need to shift result array
  If lMin <= 0 And lMax >= 0 Then
    For k = 1 To UBound(lr)
      lr(k) = lr(k) + lMin - 1
    Next k
  End If
End If

sbRandInt = lr

End Function
```

2 Zufallszahlen mit einer festgelegten Summe

2.1 Minimum für die Zufallszahlen vorgegeben – `sbLongRandSumN`

Sie benötigen 20 natürliche Zufallszahlen mit der Summe 100? Dann schlage ich meine hier gezeigte benutzerdefinierte Funktion `sbLongRandSumN` vor. Sie können beliebig viele ganze Zahlen mit einer vorgegebenen Summe erzeugen, wobei die erzeugten Zahlen ein spezifiertes Minimum nicht unterschreiten dürfen:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	lSum	lCount	lMin	Total	sbLongRandSumN									
2	92	7	6	92	8	6	9	7	17	13	32			
3	87	6	5	87	6	5	5	5	11	55				
4	58	4	7	58	12	8	8	30						
5	21	3	2	21	9	9	3							
6	65	10	4	65	5	4	5	5	4	4	5	4	4	25
7	64	3	12	64	35	16	13							
8	46	3	15	46	16	15	15							
9	83	3	16	83	23	16	44							
10	59	10	5	59	6	5	5	5	5	5	5	5	8	10

Tabellenblattformeln		
Bereich	Formel	
D2:D10	**D2**	=SUMME(E2:K2)
E2:E10	**E2**	=sbLongRandSumN(A2;B2;C2)

Figure 3: sbLongRandSumN

Programmcode `sbLongRandSumN`

```
Function sbLongRandSumN(lSum As Long, _
    ByVal lCount As Long, _
    Optional ByVal lMin As Long = 0) As Variant
'Generates lCount random integers greater equal lMin
'which sum up to lSum.
'(C) (P) by Bernd Plumhoff 26-Apr-2013 PB V0.1
Dim i As Long
Dim lSumRest As Long

If lCount * lMin > lSum Then sbLongRandSumN = CVErr(xlErrNum): Exit Function
If lCount < 1 Then sbLongRandSumN = CVErr(xlErrValue): Exit Function
Randomize
ReDim vR(1 To lCount) As Variant
lSumRest = lSum
For i = lCount To 2 Step -1
    vR(i) = lMin + Int(Rnd * (lSumRest - lMin * i))
    lSumRest = lSumRest - vR(i)
Next i
vR(1) = lSumRest: sbLongRandSumN = vR
End Function
```

2.2 Minimum und Maximum für die Zufallszahlen vorgegeben – `sbRandIntFixSum`

Sie können `lCount` ganze Zufallszahlen zwischen `lMin` und lMax mit der Summe `lSum` mit Tabellenblattfunktionen erzeugen:

	A	B	C	D	E	F
1	**Summe**	1000			**Prüfung**	1000
2	**Untergrenze**	30				
3	**Obergrenze**	110				
4	**Anzahl**	10				
5				**Unter-**	**Ober-**	
6			**Anzahl**	**grenze**	**grenze**	**Zufalls-**
7	**Runde**	**Rest**	**Rest**	**Rest**	**Rest**	**zahl**
8	0	1000	10	30	110	
9	1	1000	10	30	110	110
10	2	890	9	30	110	59
11	3	831	8	61	110	101
12	4	730	7	70	110	101
13	5	629	6	79	110	102
14	6	527	5	87	110	99
15	7	428	4	98	110	105
16	8	323	3	103	110	104
17	9	219	2	109	110	110
18	10	109	1	109	109	109

Tabellenblattformeln		
Bereich	Formel	
F1	**F1**	=SUMME(WENNFEHLER(F9:F29;0))
A8	**A8**	=SEQUENZ(B4+1;;0)
B8	**B8**	=$B$1
B9:B29	**B9**	=B8-F8
C8	**C8**	=$B$4
C9:C29	**C9**	=C8-1*(A9<>1)
D8	**D8**	=$B$2
D9:D29	**D9**	=AUFRUNDEN(MAX(D8;MIN(B9/C9;B9/C9-(C9-1)*(E8-B9/C9)));0)
E8	**E8**	=$B$3
E9:E29	**E9**	=ABRUNDEN(MIN(E8;MAX(B9/C9;B9/C9+(C9-1)*(B9/C9-D8)));0)
F9:F29	**F9**	=GANZZAHL(ZUFALLSZAHL()*(E9-D9+1)+D9)

Figure 4: sbRandIntFixSum

Programmcode `sbRandIntFixSum`

```
Function sbRandIntFixSum(lSum As Long, lMin As Long, _
    lMax As Long, Optional lCount As Long = 0, _
    Optional bUseRandTriang As Boolean = True, _
    Optional bVolatile As Boolean = False) As Variant
'Returns lCount (or selected cell count in case a range is select when
'called as a matrix formula) random integers between lMin and lMax
'which sum up to lSum. If bUseRandTriang the sbRandTriang distribution
'is used to "bias" the randomness to be "less extreme".

'Error values:
'#NUM!   - No solution exists
'#VALUE! - lCount is less than 1
'(C) (P) by Bernd Plumhoff 05-Aug-2020 PB V0.3

Dim i As Long
Dim lRnd As Long, lMinPrev As Long
Dim lRow As Long, lCol As Long

With Application

If TypeName(.Caller) = "Range" And lCount = 0 Then
    lCount = .Caller.Count
    ReDim lR(1 To .Caller.Rows.Count, 1 To .Caller.Columns.Count) As Long
ElseIf lCount < 1 Then
    sbRandIntFixSum = CVErr(xlErrValue)
    Exit Function
Else
    ReDim lR(1 To lCount, 1 To 1) As Long
End If

Randomize
If bVolatile Then .Volatile

For lRow = 1 To UBound(lR, 1)
    For lCol = 1 To UBound(lR, 2)
        lMinPrev = lMin
        lMin = .RoundUp(.Max(lMin, .Min(lSum / lCount, lSum / lCount _
            - (lCount - 1) * (lMax - lSum / lCount))), 0)
        lMax = .RoundDown(.Min(lMax, .Max(lSum / lCount, lSum / lCount _
            + (lCount - 1) * (lSum / lCount - lMinPrev))), 0)
        If lMin > lMax Or lSum / lCount <> .Median(lMin, lMax, lSum / _
            lCount) Then
            'No solution exists
            sbRandIntFixSum = CVErr(xlErrNum)
            Exit Function
        End If
        If bUseRandTriang Then
            If lMin = lMax Then
                lRnd = lMin
```

```
            Else
                lRnd = Int(sbRandTriang(CDbl(lMin), _
                        lSum / lCount, CDbl(lMax)) + 0.5)
            End If
        Else
            lRnd = Int(Rnd() * (lMax - lMin + 1) + lMin)
        End If
        lR(lRow, lCol) = lRnd
        lSum = lSum - lRnd
        lCount = lCount - 1
    Next lCol
Next lRow

sbRandIntFixSum = lR

End With

End Function
```

3 Praktische Anwendungen ganzer Zufallszahlen

3.1 Krabat – Wie alt können die Lehrlinge werden?

Krabat ist ein Jugendbuch von Otfried Preußler. Ich fand die Geschichte faszinierend, aber etwas unlogisch: 12 Lehrlinge arbeiten in der Mühle. Jedes Jahr stirbt einer, und jedes Jahr wird ein neuer Lehrling im Alter von 14 Jahren aufgenommen. Alle altern innerhalb eines Jahres um drei Jahre. Nach 30 Jahren kann es einen Lehrling geben, der 101 Jahre alt ist:

E9 =WENN(SPALTE()-3=$C9;$E$1;E8+3)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	**Anfangsalter**				14										
2	**Maximales Alter**				101										
3	**Maximales Alter in 20 Jahren**				71										
4															
5	**Mühlen-**	**Echtes**	**Wer**	**Lehrlinge und ihr Alter**											
6	**jahr**	**Jahr**	**stirbt?**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
7	1	1		14	14	14	14	14	14	14	14	14	14	14	14
8	2	4	11	17	17	17	17	17	17	17	17	17	17	14	17
9	3	7	5	20	20	20	20	14	20	20	20	20	20	17	20
10	4	10	8	23	23	23	23	17	23	23	14	23	23	20	23
11	5	13	5	26	26	26	26	14	26	26	17	26	26	23	26
12	6	16	2	29	14	29	29	17	29	29	20	29	29	26	29
13	7	19	2	32	14	32	32	20	32	32	23	32	32	29	32
14	8	22	5	35	17	35	35	14	35	35	26	35	35	32	35
15	9	25	3	38	20	14	38	17	38	38	29	38	38	35	38
16	10	28	9	41	23	17	41	20	41	41	32	14	41	38	41
17	11	31	8	44	26	20	44	23	44	44	14	17	44	41	44
18	12	34	4	47	29	23	14	26	47	47	17	20	47	44	47
19	13	37	12	50	32	26	17	29	50	50	20	23	50	47	14
20	14	40	8	53	35	29	20	32	53	53	14	26	53	50	17
21	15	43	9	56	38	32	23	35	56	56	17	14	56	53	20
22	16	46	9	59	41	35	26	38	59	59	20	14	59	56	23
23	17	49	3	62	44	14	29	41	62	62	23	17	62	59	26
24	18	52	5	65	47	17	32	14	65	65	26	20	65	62	29
25	19	55	3	68	50	14	35	17	68	68	29	23	68	65	32
26	20	58	1	14	53	17	38	20	71	71	32	26	71	68	35
27	21	61	5	17	56	20	41	14	74	74	35	29	74	71	38
28	22	64	7	20	59	23	44	17	77	14	38	32	77	74	41
29	23	67	5	23	62	26	47	14	80	17	41	35	80	77	44
30	24	70	4	26	65	29	14	17	83	20	44	38	83	80	47
31	25	73	2	29	14	32	17	20	86	23	47	41	86	83	50
32	26	76	1	14	17	35	20	23	89	26	50	44	89	86	53
33	27	79	1	14	20	38	23	26	92	29	53	47	92	89	56
34	28	82	8	17	23	41	26	29	95	32	14	50	95	92	59
35	29	85	1	14	26	44	29	32	98	35	17	53	98	95	62
36	30	88	4	17	29	47	14	35	101	38	20	56	101	98	65

Tabellenblattformeln		
Bereich	Formel	
E2	**E2**	=MAX(D8:O36)
E3	**E3**	=MAX(D8:O26)
B7:B36	**B7**	=A7*3-2
C8:C36	**C8**	=ABRUNDEN(ZUFALLSZAHL()*12;0)+1
D8:O36	**D8**	=WENN(SPALTE()-3=$C8;$E$1;D7+3)

Figure 5: Krabat

3.2 Ein Mathematiktest mit ganzen Zufallszahlen - `Generate_Math_Test`

Sie wollen einen ein¬fachen Mathematiktest für Ihr Kind oder Ihre Klasse erzeugen?
Die Eingaben im Tabellenblatt Main:

	A	B	C	D	E	F	G	H	I	J
1	**Min**	**Max**		**Sample Size**	15					
2	(									
3	1	5		Generate Random Sample						
4	+									
5	5	9		15 Expressions and Results, generated Thursday 09-Jun-2022 16:15:50						
6	)									
7	*									
8	2	4								

Figure 6: Math Test Input

The solution output in sheet Sample_Q_and_A:

	A	B	C	D	E	F
1	15 Expressions and Results, generated Thursday 09-Jun-2022 16:15					
2	Expressio	equals	Result			
3	(4+7)*2	=	22			
4	(2+5)*4	=	28			
5	(2+9)*3	=	33			
6	(5+8)*2	=	26			
7	(5+9)*2	=	28			
8	(5+9)*2	=	28			
9	(3+7)*2	=	20			
10	(3+7)*3	=	30			
11	(2+6)*4	=	32			
12	(3+9)*2	=	24			
13	(2+7)*3	=	27			
14	(4+7)*2	=	22			
15	(4+8)*2	=	24			
16	(3+6)*4	=	36			
17	(2+6)*2	=	16			

Figure 7: Math Test Solution

The exam output in sheet Sample_Q:

	A	B	C	D	E
1	15 Questions, generated Thursday 09-Jun-2022 16:15				
2	Expressio	equals	?		
3	(4+7)*2	=	?		
4	(2+5)*4	=	?		
5	(2+9)*3	=	?		
6	(5+8)*2	=	?		
7	(5+9)*2	=	?		
8	(5+9)*2	=	?		
9	(3+7)*2	=	?		
10	(3+7)*3	=	?		
11	(2+6)*4	=	?		
12	(3+9)*2	=	?		
13	(2+7)*3	=	?		
14	(4+7)*2	=	?		
15	(4+8)*2	=	?		
16	(3+6)*4	=	?		
17	(2+6)*2	=	?		

Figure 8: Math Test Exam

Programmcode `Generate_Math_Test`

Note: This program requires (uses) the class `SystemState`.

```
Sub Generate_Math_Test()
'(C) (P) by Bernd Plumhoff  09-Jun-2022 PB V1.0
Dim vi, vTime
Dim lInputRow As Long
Dim lOutputRow As Long
Dim lNum As Long
Dim sExpr As String

Dim state As SystemState
Application.StatusBar = False
Set state = New SystemState

vTime = Now
Randomize
wsQA.Range("1:1048576").Delete
wsQ.Range("1:1048576").Delete
wsQA.[a1] = Range("Sample_Size") & " Expressions and Results, generated " _
            & Format(vTime, "dddd dd-mmm-yyyy hh:mm")
wsQA.[a2] = "Expression"
wsQA.[b2] = "equals"
wsQA.[c2] = "Result"
wsQ.[a1] = Range("Sample_Size") & " Questions, generated " & _
           Format(vTime, "dddd dd-mmm-yyyy hh:mm")
wsQ.[a2] = "Expression"
wsQ.[b2] = "equals"
wsQ.[c2] = "?"

lInputRow = 2
sExpr = ""
Do While Not IsEmpty(wsMain.Cells(lInputRow, 1))
  If IsNumeric(wsMain.Cells(lInputRow, 1)) _
    And IsNumeric(wsMain.Cells(lInputRow, 2)) Then
    lNum = wsMain.Cells(lInputRow, 1) _
           + Int(Rnd() * (1 + wsMain.Cells(lInputRow, 2) - _
           wsMain.Cells(lInputRow, 1)))
    sExpr = sExpr & lNum
  Else
    sExpr = sExpr & wsMain.Cells(lInputRow, 1).Text
  End If
  lInputRow = lInputRow + 1
Loop
If IsError(Evaluate(sExpr)) Then
  wsMain.[d5] = "Expression """ & sExpr & """ evaluates to error!"
  Exit Sub
End If

For lOutputRow = 2 To 1 + Range("Sample_Size")
  lInputRow = 2
```

```
    sExpr = ""
    Do While Not IsEmpty(wsMain.Cells(lInputRow, 1))
      If IsNumeric(wsMain.Cells(lInputRow, 1)) _
        And IsNumeric(wsMain.Cells(lInputRow, 2)) Then
        lNum = wsMain.Cells(lInputRow, 1) _
          + Int(Rnd() * (1 + wsMain.Cells(lInputRow, 2) - _
          wsMain.Cells(lInputRow, 1)))
        sExpr = sExpr & lNum
      Else
        sExpr = sExpr & wsMain.Cells(lInputRow, 1).Text
      End If
      lInputRow = lInputRow + 1
    Loop
    wsQA.Cells(lOutputRow + 1, 1) = sExpr
    wsQA.Cells(lOutputRow + 1, 2) = "="
    wsQ.Cells(lOutputRow + 1, 1) = sExpr
    wsQ.Cells(lOutputRow + 1, 2) = "="
    If IsError(Evaluate(sExpr)) Then
      wsQA.Cells(lOutputRow + 1, 3) = "Expression """ & _
        sExpr & """ evaluates to error!"
      wsQ.Cells(lOutputRow + 1, 3) = "Expression """ & _
        sExpr & """ evaluates to error!"
    Else
      wsQA.Cells(lOutputRow + 1, 3) = Evaluate(sExpr)
      wsQ.Cells(lOutputRow + 1, 3) = "?"
    End If
  Next lOutputRow

  wsMain.[d5] = Range("Sample_Size") & _
                " Expressions and Results, generated " & _
                Format(vTime, "dddd dd-mmm-yyyy hh:mm:ss")

End Sub
```

3.3 Monte Carlo Simulation für eine faire Teamverteilung – `sbGenerateTeams`

Sie und Ihre 15 Freunde wollen in 4 Teams spielen mit je 4 Spielern und Sie fragen sich, wie Sie die Teams zufällig aber möglichst gleichstark aufstellen können?

Dies kann man mit dem Programm `sbGenerateTeams` erreichen:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Spieler**	**Name**	**Stärke**		**Anzahl Teams**		**Anzahl Monte Carlo Läufe**		**Team**	**Spieler**	**Stärke**		**Team**	**Summe Stärke**
2	1	Andrew	27		4		20000		1	Isaac	39		1	141
3	2	Benjamin	38						1	George	41		2	140
4	3	Charlie	31		**Spieler pro Team**		Erzeuge Teams		1	Mary	26		3	140
5	4	David	47		4				1	Edward	35		4	140
6	5	Edward	35						2	Lucy	45		StDev	0,5
7	6	Frederick	26						2	David	47			
8	7	George	41						2	Frederick	26			
9	8	Harry	43						2	Oliver	22			
10	9	Isaac	39						3	Nellie	50			
11	10	Jack	44						3	King	25			
12	11	King	25						3	Benjamin	38			
13	12	Lucy	45						3	Andrew	27			
14	13	Mary	26						4	Charlie	31			
15	14	Nellie	50						4	Peter	22			
16	15	Oliver	22						4	Harry	43			
17	16	Peter	22						4	Jack	44			

Figure 9: sbGenerateTeams

Dieses Programm vereint mehrere Funktionalitäten, die ich gern nutze:

- Die Klasse `SystemState` reduziert die Laufzeit.
- Mit Enumerierungen organisiere ich den Zugriff auf Spalten flexibel - für zusätzliche oder entfallende Spalten ändere ich lediglich die Enumerierung, und das Programm passt die Spaltennummern automatisch an.
- Neues Mischen einer Menge von Elementen mit `UniqRandInt`.
- Testdaten (Namen) erzeugte ich mit `sbGenerateTestData`.

3.3.1 Ein komplexeres Beispiel

Falls Sie zufällige Teams gleicher Stärke generieren möchten, die Untergruppen verschiedener Spielerarten haben, können Sie Spielstärkewerte mit unterschiedlichen Zehnerpotenzen (oder andere Potenzen) je Untergruppe vergeben. Sie müssen lediglich darauf achten, dass alle Untergruppen in allen Teams dieselbe Spieleranzahl haben - Ausnahme: die Untergruppe mit den kleinsten Spielstärkewerten kann unterschiedlich viele Spieler in den Teams haben.

Sie können nach einem Spiel die Spielstärkewerte anpassen. Zum Beispiel könnten Sie die Werte der Gewinner um 1 erhöhen, bis ein Maximalwert je Untergruppe erreicht ist. Oder Sie verringern die Werte für die Verlierer um 1, bis ein Minimalwert je Untergruppe erreicht ist. So stellen Sie sicher, dass auch Spielstärkeänderungen fair und nachvollziehbar abgebildet werden.

Spieler	Name	Stärke
1	Torwart 1	50000
2	Torwart 2	50000
3	Verteidiger 1	5000
4	Verteidiger 2	5000
5	Verteidiger 3	5000
6	Verteidiger 4	5000
7	Verteidiger 5	5000
8	Verteidiger 6	5000
9	Verteidiger 7	5000
10	Verteidiger 8	500
11	Mittelfeld 1	500
12	Mittelfeld 2	500
13	Mittelfeld 3	500
14	Mittelfeld 4	500
15	Mittelfeld 5	500
16	Mittelfeld 6	500
17	Stürmer 1	50
18	Stürmer 2	50
19	Stürmer 3	50
20	Stürmer 4	50
21	Stürmer 5	50
22	Stürmer 6	50
23	Stürmer 7	50

Anzahl Teams
2

Spieler pro Team
12

Anzahl Monte Carlo Läufe
20000

Erzeuge Teams

Team	Spieler	Stärke
1	Mittelfeld 6	500
1	Verteidiger 6	5000
1	Mittelfeld 5	500
1	Torwart 1	50000
1	Stürmer 6	50
1	Verteidiger 4	5000
1	Mittelfeld 2	500
1	Verteidiger 8	500
1	Mittelfeld 1	500
1	Mittelfeld 4	500
1	Mittelfeld 3	500
1	Verteidiger 3	5000
2	Stürmer 3	50
2	Verteidiger 5	5000
2	Verteidiger 7	5000
2	Verteidiger 2	5000
2	Verteidiger 1	5000
2	Stürmer 5	50
2	Stürmer 4	50
2	Stürmer 7	50
2	Stürmer 2	50
2	[Empty]	0
2	Stürmer 1	50
2	Torwart 2	50000

Team	Summe Stärke
1	68550
2	70300
StDev	1237,436867

Figure 10: sbGenerateTeams Komplexeres Beispiel

Programmcode `sbGenerateTeams`

Bitte beachten: Dieses Programm benötigt (verwendet) die Klasse `SystemState` und die benutzerdefinierte Funktion `UniqRandInt` und ist auf die vergebenen Bereichsnamen abgestimmt:

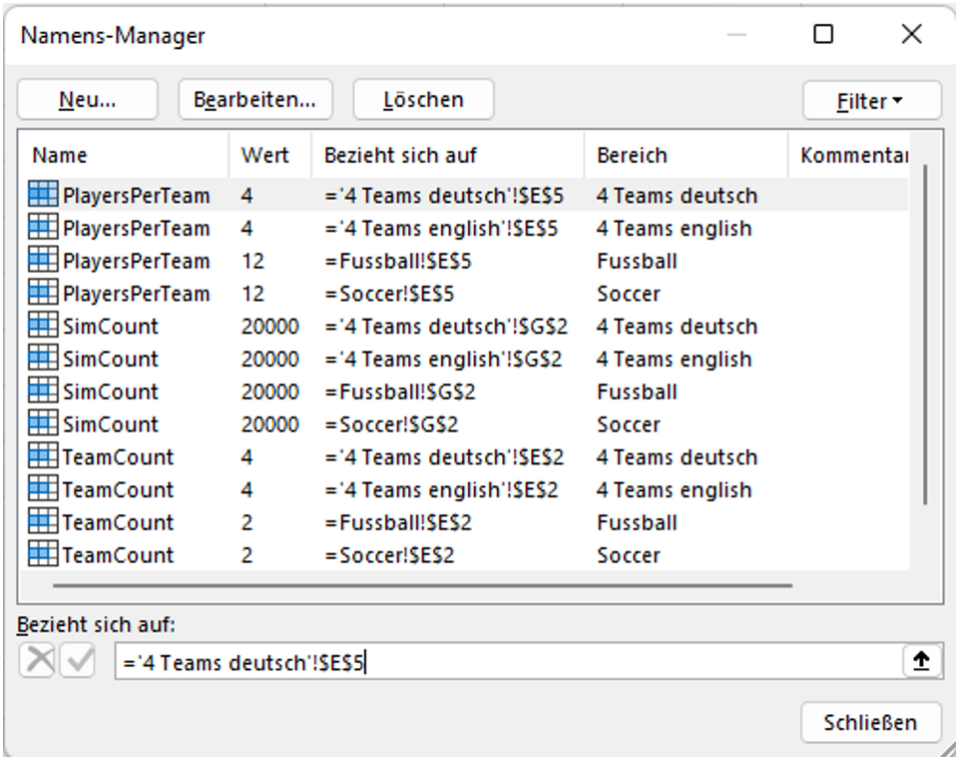


Figure 11: sbGenerateTeams Named Ranges

```
Option Explicit

#Const I_Want_Colors = True

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

Enum col_worksheet
    col_LBound = 0 'To be able to iterate from here + 1
    col_in_player_no
    col_in_player_name
    col_in_player_skill
    col_blank_1
    col_in_team_stats
    col_blank_2
    col_in_sim_stats
    col_blank_3
    col_out_team_no
    col_out_player_name
    col_out_player_skill
    col_blank_4
    col_stat_team_no
    col_stat_sum_skills
    col_Ubound 'To be able to iterate until here - 1
End Enum 'col_worksheet

Sub sbGenerateTeams()
'Implements a simple Monte Carlo simulation to randomly generate
'teams fairly, keeping track of the teams with the lowest standard
'deviation of skill sums.
```

```
'This sub needs UniqRandInt - google for sulprobil and uniqrandint.
'and the SystemState class - google for sulprobil and systemstate.
'(C) (P) by Bernd Plumhoff 07-Nov-2024 PB V0.5

Dim i As Long, j As Long, k As Long, n As Long
Dim teamcount           As Long
Dim playersperteam      As Long
Dim stdev_hc_sum        As Double
Dim min_stdev           As Double
Dim s                   As Double
Dim v                   As Variant
Dim wsI                 As Worksheet
Dim state               As SystemState

'Initialize
Set state = New SystemState
Set wsI = ThisWorkbook.ActiveSheet
teamcount = wsI.Range("TeamCount")
wsI.Range("PlayersPerTeam").Calculate
playersperteam = wsI.Range("PlayersPerTeam")
n = teamcount * playersperteam
ReDim hc(1 To n) As Double
ReDim mina(1 To n) As Double
ReDim hc_sum(1 To teamcount) As Double
wsI.Cells.Interior.ColorIndex = False
#If I_Want_Colors Then
wsI.Range("A1:C1").Interior.ColorIndex = xlCIYellow
wsI.Range("E1").Interior.ColorIndex = xlCIYellow
wsI.Range("G1").Interior.ColorIndex = xlCIYellow
wsI.Range("E4").Interior.ColorIndex = xlCIYellow
wsI.Range("E2").Interior.ColorIndex = xlCILightYellow
wsI.Range("G2").Interior.ColorIndex = xlCILightYellow
wsI.Range("E5").Interior.ColorIndex = xlCILightYellow
wsI.Range("I1:K1").Interior.ColorIndex = xlCIBrightGreen
wsI.Range("M1:N1").Interior.ColorIndex = xlCIBrightGreen
wsI.Range("M" & teamcount + 2 & ":N" & _
  teamcount + 2).Interior.ColorIndex = xlCILightGreen
#End If
For j = 1 To n
  hc(j) = wsI.Cells(j + 1, col_in_player_skill)
  #If I_Want_Colors Then
  wsI.Range("A" & j + 1 & ":C" & j + 1).Interior.ColorIndex = _
    xlCILightYellow
  #End If
Next j
min_stdev = 1E+308

k = 1
Do
  v = UniqRandInt(n, n)
  For i = 1 To teamcount
```

```
      hc_sum(i) = 0
      For j = 1 To playersperteam
        hc_sum(i) = hc_sum(i) + hc(v((i - 1) * playersperteam + j))
      Next j
    Next i
    stdev_hc_sum = WorksheetFunction.StDev(hc_sum)
    If stdev_hc_sum < min_stdev Then
      For i = 1 To n
        mina(i) = v(i)
      Next i
      min_stdev = stdev_hc_sum
      Application.StatusBar = "Iteration " & k & _
        ", new min stdev =" & min_stdev
    End If
    k = k + 1
  Loop Until k > wsI.Range("SimCount")

  wsI.Range(wsI.Cells(2, col_out_team_no), _
    wsI.Cells(1000, col_stat_sum_skills)).ClearContents

  For i = 1 To teamcount
    s = 0#
    For j = 1 To playersperteam
      wsI.Cells(1 + (i - 1) * playersperteam + j, col_out_team_no) = i
      wsI.Cells(1 + (i - 1) * playersperteam + j, col_out_player_name) = _
        IIf("" = wsI.Cells(1 + mina((i - 1) * _
          playersperteam + j), col_in_player_name), _
          "[Empty]", wsI.Cells(1 + mina((i - 1) * _
          playersperteam + j), col_in_player_name))
      wsI.Cells(1 + (i - 1) * playersperteam + j, col_out_player_skill) = _
        CDbl(wsI.Cells(1 + mina((i - 1) * _
          playersperteam + j), col_in_player_skill))
      s = s + wsI.Cells(1 + mina((i - 1) * _
          playersperteam + j), col_in_player_skill)
      #If I_Want_Colors Then
      wsI.Range("I" & 1 + (i - 1) * playersperteam + j & ":K" & 1 + _
        (i - 1) * playersperteam + j).Interior.ColorIndex = xlCILightGreen
      #End If
    Next j
    wsI.Cells(1 + i, col_stat_team_no) = i
    wsI.Cells(1 + i, col_stat_sum_skills) = s
    #If I_Want_Colors Then
    wsI.Range("M" & i + 1 & ":N" & i + 1).Interior.ColorIndex = _
      xlCILightGreen
    #End If
  Next i
  wsI.Cells(2 + teamcount, col_stat_team_no) = "StDev"
  wsI.Cells(2 + teamcount, col_stat_sum_skills) = min_stdev
End Sub
```

3.4 Monte Carlo Simulation für einen Regatta Flight Plan – `sbRegattaFlightPlan`

Falls Sie einen Regatta Flight Plan für 12 Flights (Segelrunden) für 12 Einzelsegler bei 4 vorhandenen Booten erzeugen wollen, wobei

- kein Segler gegen einen anderen zu häufig antritt
- kein Segler ein Boot zu häufig zugewiesen bekommt
- möglichst kein Segler in aufeinanderfolgenden Flights segeln muss

dann hilft Ihnen hoffentlich dieses Programm, welches `UniqRandInt` und die Klasse `SystemState` verwendet.

Eingabe:

Figure 12: sbRegattaFlightPlan Eingabe

Ausgabe:

	D	E	F	G	H	I	J	K	L	M	N	O	P
1		Flight 1	Flight 2	Flight 3	Flight 4	Flight 5	Flight 6	Flight 7	Flight 8	Flight 9	Flight 10	Flight 11	Flight 12
2	Boot 1	Dan	Ida	Carl	Joe	Dan	Eve	Hanna	Carl	Fred	Hanna	Eve	Luke
3	Boot 2	Joe	Abe	Hanna	Luke	Hanna	Carl	Kim	Abe	Giles	Dan	Giles	Fred
4	Boot 3	Eve	Kim	Fred	Giles	Ben	Abe	Ben	Dan	Ida	Abe	Joe	Carl
5	Boot 4	Giles	Luke	Ben	Kim	Ida	Fred	Luke	Joe	Eve	Kim	Ida	Ben
6	Maximale Anzahl von sich erneut treffenden Seglern	3											
7	Maximale Anzahl von Booten die ein Segler erneut nutzt	2											
8	Anzahl von Seglern mit angrenzenden Flights	0											

Figure 13: sbRegattaFlightPlan Ausgabe

Programmcode `sbRegattaFlightPlan`

Dieses Programm benötigt `UniqRandInt` und die Klasse `SystemState`.

```
#Const I_Want_Colors = True

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

Sub sbRegattaFlightPlan()
'Performs a simple Monte Carlo simulation to create a regatta flight plan.
'(C) (P) by Bernd Plumhoff 07-Jan-2023 PB V0.3
Dim i As Long, j As Long, k As Long, m As Long
Dim lAdjacentFlights        As Long
Dim lBestSailorInBoat       As Long
Dim lBestSailorMeetSailor   As Long
Dim lBoatCount              As Long
Dim lFlightCount            As Long
Dim lLowestAdjacentFlights  As Long
Dim lMaxSailorInBoat        As Long
Dim lMaxSailorMeetSailor    As Long
Dim lSailorIndex            As Long
Dim lSailorCount            As Long
Dim lSimulationCount        As Long
Dim ws                      As Worksheet
Dim state                   As SystemState

With Application.WorksheetFunction

'Initialize
Set ws = ThisWorkbook.ActiveSheet
```

```
Set state = New SystemState
ws.Cells.Interior.Pattern = xlNone
ws.Cells.Interior.TintAndShade = 0
ws.Cells.Interior.PatternTintAndShade = 0
ws.Cells.Font.ColorIndex = xlAutomatic
ws.Cells.Font.TintAndShade = 0

#If I_Want_Colors Then
For i = 1 To 56: xlFC(i) = True: Next i
xlFC(xlCIBlack) = False: xlFC(xlCIRed) = False: xlFC(xlCIBlue) = False
xlFC(xlCIDarkRed) = False: xlFC(xlCIGreen) = False
xlFC(xlCIDarkBlue) = False: xlFC(xlCIDarkYellow) = False
xlFC(xlCIViolet) = False: xlFC(xlCIDarkPurple) = False
xlFC(xlCILightBrown) = False: xlFC(xlCIDarkPink) = False
xlFC(xlCIDarkBrown) = False: xlFC(xlCISeaBlue) = False
xlFC(xlCIBlueGray) = False: xlFC(xlCIDarkTeal) = False
xlFC(xlCIDarkGreen) = False: xlFC(xlCIGreenBrown) = False
xlFC(xlCIIndigo) = False: xlFC(xlCIGray80) = False
#End If

Randomize
i = Range("Sailors").Row + 1
Do While Not IsEmpty(ws.Cells(i + lSailorCount, 1))
  lSailorCount = lSailorCount + 1
Loop
ReDim sSailor(1 To lSailorCount) As String
i = Range("Sailors").Row
j = 1
Do While Not IsEmpty(ws.Cells(i + j, 1))
  sSailor(j) = ws.Cells(i + j, 1)
  #If I_Want_Colors Then
  k = (j Mod 56) + 1
  ws.Cells(i + j, 1).Interior.ColorIndex = k
  If xlFC(k) Then
    ws.Cells(i + j, 1).Font.ColorIndex = xlCIBlack
  Else
    ws.Cells(i + j, 1).Font.ColorIndex = xlCIWhite
  End If
  #End If
  j = j + 1
Loop

lBoatCount = Range("Boats")
lFlightCount = Range("Flights")
lSimulationCount = Range("Simulations")
lBestSailorMeetSailor = lSailorCount
lBestSailorInBoat = lBoatCount
lLowestAdjacentFlights = lFlightCount * lSailorCount

If lFlightCount * lBoatCount Mod lSailorCount <> 0 Then
  Call MsgBox("Number of flights" & vbCrLf & "times number of boats" & _
```

```
      vbCrLf & "needs to be divisible" & vbCrLf & "by number of sailors!", _
      vbOKOnly, "Error")
    Exit Sub
  End If
  If lBoatCount > lSailorCount Then
    Call MsgBox("Number of boats" & vbCrLf & "needs to be less or equal" & _
      vbCrLf & "to number of sailors!", vbOKOnly, "Error")
    Exit Sub
  End If

  Range("D:XFD").EntireColumn.Delete

  ReDim lBestBoatInFlight(1 To lBoatCount, 1 To lFlightCount) As Long
  For i = 1 To lSimulationCount
    ReDim lSailorInBoat(1 To lSailorCount, 1 To lBoatCount) As Long
    ReDim lSailorMeetSailor(1 To lSailorCount, 1 To lSailorCount) As Long
    ReDim lBoatInFlight(1 To lBoatCount, 1 To lFlightCount) As Long
    lAdjacentFlights = 0
    For j = 1 To lFlightCount
      ReDim lBoat(1 To lBoatCount) As Long
      For k = 1 To lBoatCount
        If lSailorIndex = 0 Then
          ReDim vSailor(1 To lSailorCount) As Variant
          vSailor = UniqRandInt(lSailorCount, lSailorCount)
          lSailorIndex = 1
        End If
        lBoat(k) = vSailor(lSailorIndex)
        lBoatInFlight(k, j) = vSailor(lSailorIndex)
        For m = 1 To k - 1
          lSailorMeetSailor(lBoat(k), lBoat(m)) = _
            lSailorMeetSailor(lBoat(k), lBoat(m)) + 1
          lSailorMeetSailor(lBoat(m), lBoat(k)) = _
            lSailorMeetSailor(lBoat(m), lBoat(k)) + 1
        Next m
        If j > 1 Then
          For m = 1 To lBoatCount
            If lBoatInFlight(k, j) = lBoatInFlight(m, j - 1) Then
              lAdjacentFlights = lAdjacentFlights + 1
            End If
          Next m
        End If
        lSailorInBoat(vSailor(lSailorIndex), k) = _
          lSailorInBoat(vSailor(lSailorIndex), k) + 1
        lSailorIndex = (lSailorIndex + 1) Mod (lSailorCount + 1)
      Next k
    Next j
    lMaxSailorMeetSailor = 0
    For j = 1 To lSailorCount - 1
      For m = j + 1 To lSailorCount
        If lMaxSailorMeetSailor < lSailorMeetSailor(j, m) Then
          lMaxSailorMeetSailor = lSailorMeetSailor(j, m)
```

```
        End If
      Next m
    Next j
    lMaxSailorInBoat = 0
    For j = 1 To lSailorCount
      For m = 1 To lBoatCount
        If lMaxSailorInBoat < lSailorInBoat(j, m) Then
          lMaxSailorInBoat = lSailorInBoat(j, m)
        End If
      Next m
    Next j
    If lBestSailorMeetSailor + lBestSailorInBoat + lLowestAdjacentFlights > _
      lMaxSailorMeetSailor + lMaxSailorInBoat + lAdjacentFlights Then
      For j = 1 To lBoatCount
        For m = 1 To lFlightCount
          lBestBoatInFlight(j, m) = lBoatInFlight(j, m)
        Next m
      Next j
      lBestSailorMeetSailor = lMaxSailorMeetSailor
      lBestSailorInBoat = lMaxSailorInBoat
      lLowestAdjacentFlights = lAdjacentFlights
    End If
  Next i

  For m = 1 To lFlightCount: ws.Cells(1, 4 + m) = "Flight " & m: Next m
  For j = 1 To lBoatCount
    ws.Cells(1 + j, 4) = "Boat " & j
    For m = 1 To lFlightCount
      ws.Cells(1 + j, 4 + m) = sSailor(lBestBoatInFlight(j, m))
      #If I_Want_Colors Then
      k = (lBestBoatInFlight(j, m) Mod 56) + 1
      ws.Cells(1 + j, 4 + m).Interior.ColorIndex = k
      If xlFC(k) Then
        ws.Cells(1 + j, 4 + m).Font.ColorIndex = xlCIBlack
      Else
        ws.Cells(1 + j, 4 + m).Font.ColorIndex = xlCIWhite
      End If
      #End If
    Next m
  Next j

  ws.Cells(j + 1, 4) = "Maximal meet of sailor pairs"
  ws.Cells(j + 1, 5) = lBestSailorMeetSailor
  ws.Cells(j + 2, 4) = "Maximal repetition of boat per sailor"
  ws.Cells(j + 2, 5) = lBestSailorInBoat
  ws.Cells(j + 3, 4) = "Number of sailors with adjacent flights"
  ws.Cells(j + 3, 5) = lLowestAdjacentFlights
  Range("D:XFD").EntireColumn.AutoFit

End With
End Sub
```

3.5 Chancen beim Brettspiel Risiko

Kennen Sie Ihre Chance, einen Angriff mit 15 Armeen auf einem Ihrer Länder gegen einen benachbarten Verteidiger mit 11 Armeen zu gewinnen? Nach den alten, originalen Regeln können Sie mit 14 Ihrer 15 Armeen angreifen und hätten eine etwa 32%-ige Chance, zu gewinnen, aber nach den neuen Regeln läge die Chance bei 79%: (siehe blaue Kreise)

Alte Version: Sowohl Angreifer als auch Verteidiger verwenden bis zu 3 Würfel.

Angreifer Armeen \ Verteidiger Armeen	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	0,41	0,11	0,02	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
3	0,76	0,36	0,12	0,05	0,02	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
4	0,92	0,66	0,32	0,22	0,12	0,06	0,03	0,02	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	0,97	0,79	0,45	0,31	0,20	0,11	0,07	0,04	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
6	0,99	0,89	0,57	0,41	0,28	0,17	0,11	0,07	0,04	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00
7	1,00	0,94	0,69	0,52	0,37	0,26	0,17	0,12	0,07	0,05	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00
8	1,00	0,97	0,76	0,61	0,46	0,33	0,24	0,16	0,11	0,07	0,05	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00
9	1,00	0,98	0,82	0,69	0,54	0,41	0,31	0,22	0,15	0,11	0,07	0,05	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00
10	1,00	0,99	0,87	0,75	0,62	0,49	0,36	0,28	0,20	0,15	0,11	0,07	0,05	0,04	0,02	0,02	0,01	0,01	0,00	0,00
11	1,00	0,99	0,90	0,80	0,68	0,55	0,45	0,34	0,25	0,19	0,14	0,10	0,07	0,05	0,03	0,02	0,01	0,01	0,01	0,00
12	1,00	1,00	0,93	0,84	0,73	0,62	0,51	0,40	0,32	0,24	0,17	0,13	0,09	0,07	0,05	0,03	0,02	0,02	0,01	0,01
13	1,00	1,00	0,95	0,88	0,78	0,68	0,57	0,47	0,36	0,28	0,22	0,16	0,12	0,09	0,06	0,04	0,03	0,02	0,01	0,01
14	1,00	1,00	0,96	0,90	0,83	0,73	0,62	0,52	0,43	0,34	0,26	0,20	0,16	0,11	0,08	0,06	0,04	0,03	0,02	0,01
15	1,00	1,00	0,97	0,93	0,86	0,77	0,68	0,58	0,48	0,39	0,32	0,25	0,19	0,15	0,11	0,08	0,05	0,04	0,03	0,02
16	1,00	1,00	0,98	0,94	0,89	0,81	0,72	0,63	0,54	0,44	0,37	0,29	0,24	0,18	0,13	0,10	0,07	0,06	0,04	0,03
17	1,00	1,00	0,98	0,96	0,90	0,85	0,78	0,68	0,58	0,50	0,42	0,34	0,27	0,22	0,17	0,13	0,10	0,07	0,05	0,04
18	1,00	1,00	0,99	0,97	0,93	0,88	0,80	0,73	0,64	0,55	0,47	0,39	0,31	0,26	0,21	0,15	0,12	0,09	0,07	0,05
19	1,00	1,00	0,99	0,98	0,94	0,89	0,83	0,76	0,68	0,60	0,52	0,44	0,35	0,30	0,23	0,19	0,15	0,12	0,08	0,06
20	1,00	1,00	1,00	0,98	0,96	0,91	0,86	0,80	0,71	0,64	0,56	0,50	0,41	0,34	0,28	0,22	0,17	0,14	0,11	0,08

Neue Version: Angreifer verwendet bis zu 3 Würfel, Verteidiger nur bis zu 2.

Angreifer Armeen \ Verteidiger Armeen	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	0,41	0,10	0,03	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
3	0,75	0,36	0,21	0,09	0,05	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
4	0,92	0,67	0,47	0,32	0,21	0,13	0,08	0,05	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	0,97	0,78	0,65	0,46	0,36	0,26	0,18	0,13	0,09	0,06	0,04	0,03	0,02	0,01	0,01	0,00	0,00	0,00	0,00	0,00
6	0,99	0,88	0,78	0,64	0,51	0,39	0,29	0,23	0,17	0,11	0,08	0,06	0,04	0,03	0,02	0,01	0,01	0,01	0,00	0,00
7	1,00	0,94	0,85	0,74	0,64	0,52	0,43	0,32	0,26	0,18	0,15	0,11	0,08	0,06	0,04	0,03	0,02	0,02	0,01	0,01
8	1,00	0,97	0,91	0,83	0,74	0,64	0,54	0,45	0,35	0,28	0,22	0,17	0,13	0,10	0,07	0,05	0,04	0,03	0,02	0,02
9	1,00	0,98	0,95	0,89	0,82	0,73	0,65	0,55	0,45	0,38	0,31	0,24	0,19	0,15	0,12	0,09	0,07	0,06	0,04	0,03
10	1,00	0,99	0,97	0,93	0,87	0,81	0,73	0,65	0,55	0,48	0,40	0,33	0,27	0,22	0,17	0,14	0,10	0,08	0,06	0,05
11	1,00	0,99	0,98	0,95	0,92	0,86	0,80	0,73	0,64	0,56	0,50	0,42	0,35	0,29	0,24	0,19	0,15	0,11	0,10	0,08
12	1,00	1,00	0,99	0,97	0,94	0,91	0,85	0,80	0,73	0,64	0,58	0,50	0,43	0,38	0,31	0,25	0,21	0,17	0,14	0,11
13	1,00	1,00	0,99	0,98	0,96	0,93	0,90	0,85	0,79	0,72	0,65	0,59	0,51	0,45	0,38	0,33	0,28	0,23	0,19	0,15
14	1,00	1,00	1,00	0,99	0,98	0,95	0,92	0,89	0,84	0,79	0,72	0,66	0,59	0,53	0,47	0,40	0,34	0,29	0,25	0,21
15	1,00	1,00	1,00	0,99	0,99	0,97	0,95	0,91	0,88	0,83	0,79	0,72	0,66	0,60	0,54	0,47	0,41	0,37	0,31	0,25
16	1,00	1,00	1,00	1,00	0,99	0,98	0,97	0,94	0,91	0,88	0,83	0,78	0,72	0,67	0,60	0,55	0,48	0,43	0,37	0,32
17	1,00	1,00	1,00	1,00	0,99	0,99	0,98	0,96	0,93	0,90	0,87	0,83	0,78	0,73	0,67	0,62	0,56	0,49	0,44	0,39
18	1,00	1,00	1,00	1,00	1,00	0,99	0,98	0,97	0,95	0,93	0,90	0,87	0,82	0,78	0,73	0,67	0,61	0,57	0,50	0,45
19	1,00	1,00	1,00	1,00	1,00	0,99	0,99	0,98	0,97	0,95	0,92	0,90	0,87	0,82	0,78	0,73	0,68	0,62	0,57	0,51
20	1,00	1,00	1,00	1,00	1,00	1,00	0,99	0,99	0,98	0,97	0,94	0,92	0,89	0,86	0,82	0,78	0,73	0,68	0,62	0,57

Figure 14: Brettspiel Risiko

Ein bedingtes Format färbt die Zellen rot für Chancen in Höhe von 50% oder weniger, ein gelber Hintergrund zeigt Chancen zwischen 50% und 75% an, und grüne Zellfarben zeigen Chancen von 75% oder höher an:

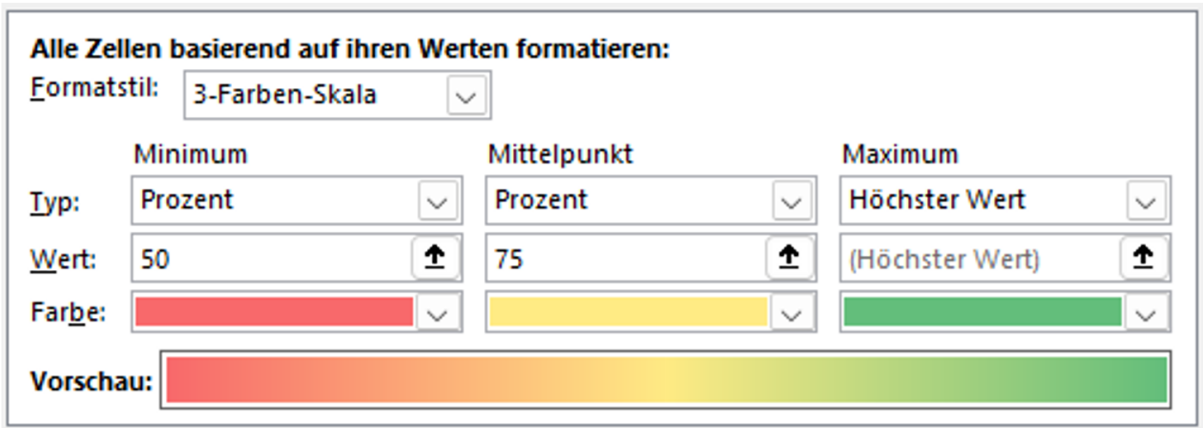


Figure 15: Brettspiel Risiko - Bedingte Zellformate

Theoretisch müssten beide Tabellen für die ersten 2 Spalten identisch sein. Kleine Differenzen werden durch die 'unvollständige' Zufälligkeit der endlichen Monte Carlo Simulation mit 10.000 Läufen verursacht.

Programmcode `Game of Risk`

Dieses Programm benötigt (verwendet) die Klasse `SystemState`.

```
Const GCMonteCarloRuns = 10000

Sub Schedule()
'Calculate chances for an attacker at the game of risk for both the
'original version (both attacker and defender roll up to 3 dice) and
'the new version (attacker rolls up to 3 dice, defender only up to 2).
'Calls parametrized sub Calculate_Chances twice.
'(C) (P) by Bernd Plumhoff 30-Sep-2012 PB V0.1
Dim ws As Worksheet
'See: https://jkp-ads.com/Articles/performanceclass.asp:
Dim cPerf As clsPerf
Dim state As SystemState

If gbDebug Then
  Set cPerf = New clsPerf
  cPerf.SetRoutine "Schedule"
End If
Application.StatusBar = False
Set state = New SystemState

'Preparation
Set ws = Sheets("Chances")
ws.Cells.ClearContents

Call Calculate_Chances("Old Version: Both Attacker and" & _
  defender roll up to" & " 3 dice.", 1, 3)
```

```
Call Calculate_Chances("New Version: Attacker rolls up " & _
  "to 3 dice, defender" & " only up to 2.", 23, 2)

Call ReportPerformance

End Sub

Sub Calculate_Chances(sTitle As String, _
    lOutputRow As Long, _
    lMaxDefenderArmies As Long)
'Calculate chances for an attacker at the game of risk.
'This sub calculates the chances for a matrix of 2 to 20 attacking armies
'against 1 to 20 defending armies.
'(C) (P) by Bernd Plumhoff V0.1 30-Sep-2012
Dim i As Long
Dim j As Long
Dim k As Long
Dim m As Long
Dim lAttackerDice As Long
Dim lAttackerThrow As Long
Dim lAttackerResult(1 To 3) As Long
Dim lAttackerWins As Long
Dim lDefenderDice As Long
Dim lDefenderThrow As Long
Dim lDefenderResult(1 To 3) As Long
Dim ws As Worksheet
'See: https://jkp-ads.com/Articles/performanceclass.asp:
Dim cPerf As clsPerf
Dim state As SystemState

If gbDebug Then
  Set cPerf = New clsPerf
  cPerf.SetRoutine "Calculate_Chances"
End If

Application.StatusBar = False
Set state = New SystemState

With Application.WorksheetFunction
'Preparation
Set ws = Sheets("Chances")
ws.Cells(lOutputRow, 1) = sTitle
ws.Cells(lOutputRow + 1, 1) = "Attacker armies \ Defender armies"
For i = 2 To 20
  Application.StatusBar = "Calculating " & i & " attackers for " & sTitle
  For j = 1 To 20
    ws.Cells(i + lOutputRow, 1) = i
    ws.Cells(1 + lOutputRow, j + 1) = j
    lAttackerWins = 0
    For k = 1 To GCMonteCarloRuns
      lAttackerDice = i - 1 'One army needs to occupy the land and
                            'cannot be used to attack
```

```
lDefenderDice = j
Do While lAttackerDice > 0 And lDefenderDice > 0
  lAttackerThrow = lAttackerDice
  If lAttackerThrow > 3 Then lAttackerThrow = 3
  lDefenderThrow = lDefenderDice
  If lDefenderThrow > lMaxDefenderArmies Then
    lDefenderThrow = lMaxDefenderArmies
  End If
  'Roll the dice
  For m = 2 To 3
    lAttackerResult(m) = 0
    lDefenderResult(m) = 0
  Next m
  For m = 1 To lAttackerThrow
    lAttackerResult(m) = Int(1 + Rnd * 6)
  Next m
  For m = 1 To lDefenderThrow
    lDefenderResult(m) = Int(1 + Rnd * 6)
  Next m
  'Sort results
  If lAttackerResult(1) < lAttackerResult(2) Then
    If lAttackerResult(1) < lAttackerResult(3) Then
        If lAttackerResult(2) < lAttackerResult(3) Then
            '3-2-1
           m = lAttackerResult(1)
            lAttackerResult(1) = lAttackerResult(3)
            lAttackerResult(3) = m
        Else
            '2-3-1
            m = lAttackerResult(1)
           lAttackerResult(1) = lAttackerResult(2)
           lAttackerResult(2) = lAttackerResult(3)
           lAttackerResult(3) = m
        End If
    Else
        '2-1-3
       m = lAttackerResult(1)
       lAttackerResult(1) = lAttackerResult(2)
       lAttackerResult(2) = m
    End If
  Else
    If lAttackerResult(1) < lAttackerResult(3) Then
        If lAttackerResult(2) < lAttackerResult(3) Then
            '3-1-2
           m = lAttackerResult(1)
            lAttackerResult(1) = lAttackerResult(3)
            lAttackerResult(3) = lAttackerResult(2)
            lAttackerResult(2) = m
        End If
    Else
        If lAttackerResult(2) < lAttackerResult(3) Then
```

```
            '1-3-2
            m = lAttackerResult(2)
            lAttackerResult(2) = lAttackerResult(3)
            lAttackerResult(3) = m
        End If
    End If
End If
If lDefenderResult(1) < lDefenderResult(2) Then
    If lDefenderResult(1) < lDefenderResult(3) Then
        If lDefenderResult(2) < lDefenderResult(3) Then
            '3-2-1
            m = lDefenderResult(1)
            lDefenderResult(1) = lDefenderResult(3)
            lDefenderResult(3) = m
        Else
            '2-3-1
            m = lDefenderResult(1)
            lDefenderResult(1) = lDefenderResult(2)
            lDefenderResult(2) = lDefenderResult(3)
            lDefenderResult(3) = m
        End If
    Else
        '2-1-3
        m = lDefenderResult(1)
        lDefenderResult(1) = lDefenderResult(2)
        lDefenderResult(2) = m
    End If
Else
    If lDefenderResult(1) < lDefenderResult(3) Then
        If lDefenderResult(2) < lDefenderResult(3) Then
            '3-1-2
            m = lDefenderResult(1)
            lDefenderResult(1) = lDefenderResult(3)
            lDefenderResult(3) = lDefenderResult(2)
            lDefenderResult(2) = m
        End If
    Else
        If lDefenderResult(2) < lDefenderResult(3) Then
            '1-3-2
            m = lDefenderResult(2)
            lDefenderResult(2) = lDefenderResult(3)
            lDefenderResult(3) = m
        End If
    End If
End If
'Analyze result and reduce armies
For m = 1 To 3
    If lAttackerResult(m) > 0 And lDefenderResult(m) > 0 Then
        If lAttackerResult(m) > lDefenderResult(m) Then
            lDefenderDice = lDefenderDice - 1
        Else
```

```
            lAttackerDice = lAttackerDice - 1
          End If
        Else
          Exit For
        End If
      Next m
    Loop
    If lAttackerDice > 0 Then
      lAttackerWins = lAttackerWins + 1
    End If
  Next k
  ws.Cells(i + lOutputRow, j + 1) = lAttackerWins / GCMonteCarloRuns
Next j
Next i
End With
End Sub
```

3.6 Eine simple Monte Carlo Simulation

Mit Excel/VBA kann man rasch eine Monte Carlo Simulation erzeugen, aber man sollte einige mögliche Fallstricke umgehen:
• Verwende die Klasse `SystemState`, um das Programm durch Ausschalten von ScreenUpdating und durch Setzen der Calculation auf xlCalculationManual zu beschleunigen.
• Halte den Anwender über den Prozess der Simulation auf dem Laufenden.
• Behandle unbekanntes Dimensionswachstum während des Programms effizient.
• Sei Dir bewusst, dass Excel im Allgemeinen nicht das beste (nicht das schnellste) Simulationstool ist.

	A	B	C	D	E	F	G
1	Number of Simulations	2.000.000		3 showed up after this many throws	How often	Theoretical Value (rounded)	
2				1	199029	200000	
3	Start Simulation			2	180354	180000	
4				3	162605	162000	
5				4	145455	145800	
6				5	131762	131220	
7				6	118324	118098	
8				7	106100	106288	
9				8	95572	95659	
10				9	85749	86094	
11				10	77749	77484	
12				11	69962	69736	
13				12	62811	62762	

Figure 16: Simple Monte Carlo Simulation

3.6.1 Literatur

Ab Excel 2010 scheint es ok zu sein, Excel für Monte Carlo Simulationen zu verwenden, wenn es nicht zu langsam ist:
Alexei Botchkarev, Assessing Excel VBA Suitability for Monte Carlo Simulation,
https://arxiv.org/ftp/arxiv/papers/1503/1503.08376.pdf

Programmcode `sbMontaCarloSimulation`

Dieses Programm benötigt (verwendet) die Klasse `SystemState`.

```
Sub Simulate()
'Creates a simple Monte Carlo simulation by counting how long it takes to
 throw a 3 with a die with 10 surfaces (likelihood for each to show is 1/10).
'(C) (P) by Bernd Plumhoff  23-Nov-2022 PB V0.2
Dim i                   As Long
Dim lSimulations        As Long
Dim lTries              As Long
Dim state               As SystemState

With Application.WorksheetFunction
Set state = New SystemState
Randomize
lSimulations = Range("Simulations")
ReDim lResult(1 To 1) As Long 'Error Handler will increase as needed
On Error GoTo ErrHdl
For i = 1 To lSimulations
    If i Mod 10000 = 1 Then Application.StatusBar = "Simulation " & _
        Format(i, "#,##0") 'Inform the user that program is still alive
    lTries = 0
    Do
        lTries = lTries + 1
    Loop Until .RandBetween(1, 10) = 3 'This is the simulation
    lResult(lTries) = lResult(lTries) + 1
Next i
On Error GoTo 0
Range("D:F").ClearContents
Range("D1:F1").FormulaArray = Array("3 showed up after this many throws", _
    "How often", "Theoretical Value (rounded)")
For i = 1 To UBound(lResult)
    Cells(i + 1, 4) = i:  Cells(i + 1, 5) = lResult(i)
    lTries = .Round(lSimulations * 0.1, 0): Cells(i + 1, 6) = lTries
    lSimulations = lSimulations - lTries
Next i
End With
Exit Sub

ErrHdl:
If Err.Number = 9 Then
   'Here we normally get if we breach Ubound(lResult)
   If lTries > UBound(lResult) Then
       'So we need to increase dimension
      ReDim Preserve lResult(1 To lTries)
       Resume 'Back to statement which caused error
   End If
End If
On Error GoTo 0 'Other error - terminate
Resume
End Sub
```

4 Gleitkomma-Zufallszahlen

4.1 Erzeuge eine ideale Normalverteilung – `sbGenNormDist`

Um eine Standardnormalverteilung zu generieren, verwenden Sie normalerweise =NORM.S.INV(ZUFALLSZAHL()). Wenn Sie also eine Standardnormalverteilung mit dem Mittelwert 7 und der Standardabweichung 10 benötigen, dann rufen Sie =NORM.INV(ZUFALLSZAHL();7;10) auf.

Aber: Ihre Normalverteilung wird nie einen Mittelwert von genau 7 (und nie eine Standard¬abweichung von genau 10) zeigen, wenn die Anzahl der Zufallszahlen nicht gegen unendlich geht. Falls Sie einen Mittelwert von genau 7 und eine Standardabweichung von genau 10 erreichen wollen, dann müssen Sie die erzeugte Menge von Zufallszahlen zum Mittelwert verschieben und sie dann zur gewünschten Standardabweichung stauchen oder dehnen. Sie können dies auf mindestens drei verschiedene Art und Weisen erreichen:

C24 | =sbGenNormDist(10;B6;B7)

	A	B	C	D	E	F
1	**Erzeuge eine Menge von Zufallszahlen mit dem Mittelwert M und der Standardabweichung S**					
3	**Ansatz mit Tabellenblattfunktionen**					
5				**Formeln in Spalte C**		**Formeln in Spalte E**
6	**Mean M**	7	6,7944131	=MITTELWERT(C8:C17)	7	=MITTELWERT(E8:E17)
7	**StDev S**	10	10,441055	=STABW.S(C8:C17)	10	=STABW.S(E8:E17)
8			-4,5794828	=NORM.INV(ZUFALLSZAHL();B	-3,893435039	=$B$6+(C8-$C$6)*$B$7/$C$7
9			26,986454		26,3390799	
10			10,405619		10,45865971	
11			12,504899	1. Schritt der 1. Lösung	12,46926097	1. Lösung
12		Daten	16,733573		16,51930603	
13			-3,5126697		-2,87168668	
14			0,6537503		1,11873367	
15			-2,9944088		-2,375318377	
16			11,265665		11,28237507	
17			0,4807327		0,953024758	
19	**Ansatz mit VBA**					
21				**Formeln in Spalte C**		**Formeln in Spalte E**
22			7	=MITTELWERT(C24:C33)	7	=MITTELWERT(E24:E33)
23			10	=STABW.S(C24:C33)	10	=STABW.S(E24:E33)
24			25,004199	=sbGenNormDist(10;B6;B7)	8,069886204	8,06988620352736
25			12,092367		12,16061155	
26			8,8120697		12,48134027	Erzeuge Zufalls-KONSTANTEN
27			1,8056338		20,43492779	
28			9,9119656		-11,11087134	
29			-1,9171841	2. Lösung	7,603025147	3. Lösung
30			0,2556426		-9,160180688	
31			2,1986687		4,412517183	
32			19,349192		13,92644218	
33			-7,5125533		11,18230171	

Figure 17: sbGenNormDist

Programmcode `sbGenNormDist`

```
Function sbGenNormDist(lCount As Long, _
    dMean As Double, _
    dStDev As Double) As Variant
'Generates lCount normally distributed random values
'with mean dMean and standard deviation dStDev.
'(C) (P) by Bernd Plumhoff 30-May-2024 V0.3
Dim i As Long
Dim dSampleMean As Double, dSampleStDev As Double

If lCount < 2 Then
  sbGenNormDist = CVErr(xlErrValue)
  Exit Function
End If
ReDim vR(1 To lCount) As Variant
With Application
For i = 1 To lCount
  vR(i) = .Norm_Inv(Rnd(), dMean, dStDev)
Next i
dSampleMean = .Average(vR)
dSampleStDev = .StDev_S(vR)
For i = 1 To lCount
  vR(i) = dMean + (vR(i) - dSampleMean) * dStDev / dSampleStDev
Next i
sbGenNormDist = .Transpose(vR)
End With
End Function
```

4.2 Verteilungen von Gleitkomma-Zufallszahlen

4.2.1 `sbRandGeneral`

Wenn Sie eine schrittweise lineare Verteilung von Zufallszahlen benötigen, dann empfehle ich meine benutzerdefinierte Funktion `sbRandGeneral`.

Bemerkung: Man kann jede beliebige Verteilung mit einer festgelegten Mindestgenauigkeit durch eine schrittweise lineare Verteilung wie hier angeboten approximieren.

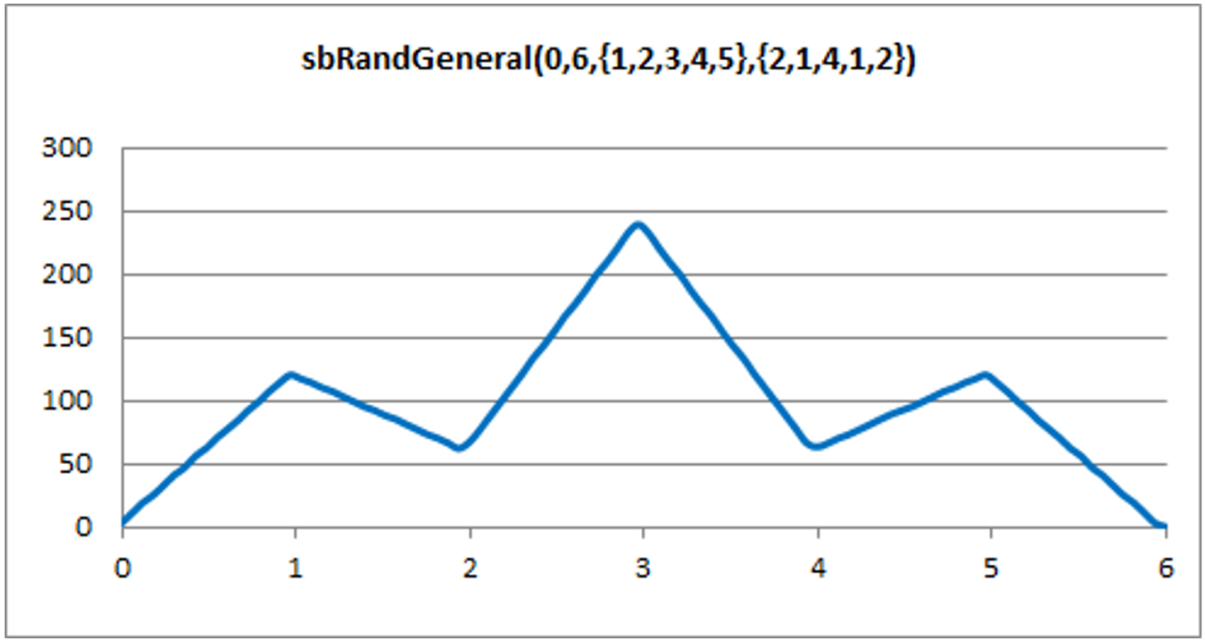
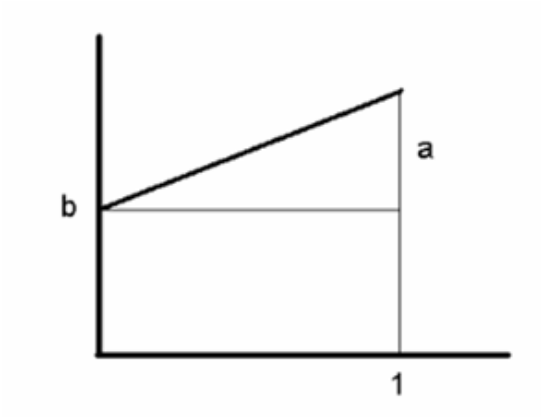


Figure 18: sbRandGeneral

Herleitung des Algorithmus



Given: Values a and b.

Needed: Inverse function of area below $f(x) = ax + b$.

Function F for area below f(x): $F(x) = \frac{a}{2}x^2 + bx$

$\Leftrightarrow$ [a≠0]

$$\frac{2}{a}F(x) = x^2 + \frac{2b}{a}x + \frac{b^2}{a^2} - \frac{b^2}{a^2} = \left(x + \frac{b}{a}\right)^2 - \frac{b^2}{a^2}$$

$\Leftrightarrow$

$$x = -\frac{b}{a} \pm \sqrt{\frac{2}{a}F(x) + \frac{b^2}{a^2}}$$

$\Leftrightarrow$

$$G(x) = -\frac{b}{a} \pm \sqrt{\frac{2}{a}x + \frac{b^2}{a^2}}$$

With $b = w_i$ and $a = \frac{w_{i+1} - w_i}{x_{i+1} - x_i}$ we get

$$G(x) = -\frac{w_i(x_{i+1} - x_i)}{w_{i+1} - w_i} \pm \sqrt{\frac{2(x_{i+1} - x_i)}{w_{i+1} - w_i}x + \left(\frac{w_i(x_{i+1} - x_i)}{w_{i+1} - w_i}\right)^2}$$

Figure 19: sbRandGeneral - Herleitung des Algorithmus

Programmcode `sbRandGeneral`

```
Function sbRandGeneral(dMin As Double, dMax As Double, vXi As Variant, _
    vWi As Variant, Optional dRandom As Double = 1#) As Double
'Generates a random number, General distributed.
'[see Vose: Risk Analysis, 2nd ed., p. 116]
'(C) (P) by Bernd Plumhoff  26-Jul-2020 PB V1.01
'Similar to @RISK's (C) RiskGeneral function.
Static bRandomized As Boolean
Dim i As Long, lWiCount As Long, lXiCount As Long
Dim dA As Double, dRand As Double, dSgn As Double

On Error GoTo ErrorLabelIsVariant
lXiCount = vXi.Count
lWiCount = vWi.Count
ErrorLabelWasVariant:
On Error GoTo 0
If lWiCount <> lXiCount Then
    sbRandGeneral = CVErr(xlErrValue)
    Exit Function
End If
If Not bRandomized Then Randomize: bRandomized = True
ReDim dX(0 To lXiCount + 1) As Double
ReDim dW(0 To lWiCount + 1) As Double

dX(0) = dMin
dX(UBound(dX)) = dMax
dW(0) = 0#
dW(UBound(dW)) = 0#
For i = 1 To lXiCount
    dX(i) = vXi(i)
    dW(i) = vWi(i)
Next i

'Calculate area
dA = 0#
For i = 0 To UBound(dX) - 1
    If dX(i) >= dX(i + 1) Or dW(i) < 0# Then
        sbRandGeneral = CVErr(xlErrValue)
        Exit Function
    End If
    dA = dA + (dX(i + 1) - dX(i)) * (dW(i + 1) + dW(i)) / 2#
Next i

'Normalise weights to set area to 1
For i = 1 To UBound(dW) - 1
    dW(i) = dW(i) / dA
Next i

ReDim dF(0 To UBound(dX)) As Double
'Calculate border points of value ranges for
```

```
'cumulative inverse function
dF(0) = 0#
dA = 0#
For i = 0 To UBound(dX) - 1
    dA = dA + (dX(i + 1) - dX(i)) * (dW(i + 1) + dW(i)) / 2#
    dF(i + 1) = dA
Next i
If dRandom = 1# Then
    dRand = Rnd()
Else
    dRand = dRandom
End If
i = 1
Do While dF(i) <= dRand
    i = i + 1
Loop
dSgn = Sgn(dW(i) - dW(i - 1))
If dSgn = 0# Then
    sbRandGeneral = dX(i - 1) + (dRand - dF(i - 1)) / _
                    (dF(i) - dF(i - 1)) * (dX(i) - dX(i - 1))
Else
    sbRandGeneral = dX(i - 1) + _
                    dSgn * Sqr((dRand - dF(i - 1)) * _
                    2# * (dX(i) - dX(i - 1)) / (dW(i) - dW(i - 1)) + _
                    (dW(i - 1) * (dX(i) - dX(i - 1)) / _
                    (dW(i) - dW(i - 1))) ^ 2#) - _
                    dW(i - 1) * (dX(i) - dX(i - 1)) / (dW(i) - dW(i - 1))
End If
Exit Function

ErrorLabelIsVariant:
lXiCount = UBound(vXi) - 1
lWiCount = UBound(vWi) - 1
Resume ErrorLabelWasVariant

End Function
```

4.2.2 sbRandHistogrm

Eine stufenweise konstante Verteilung ist die Histogram Verteilung:

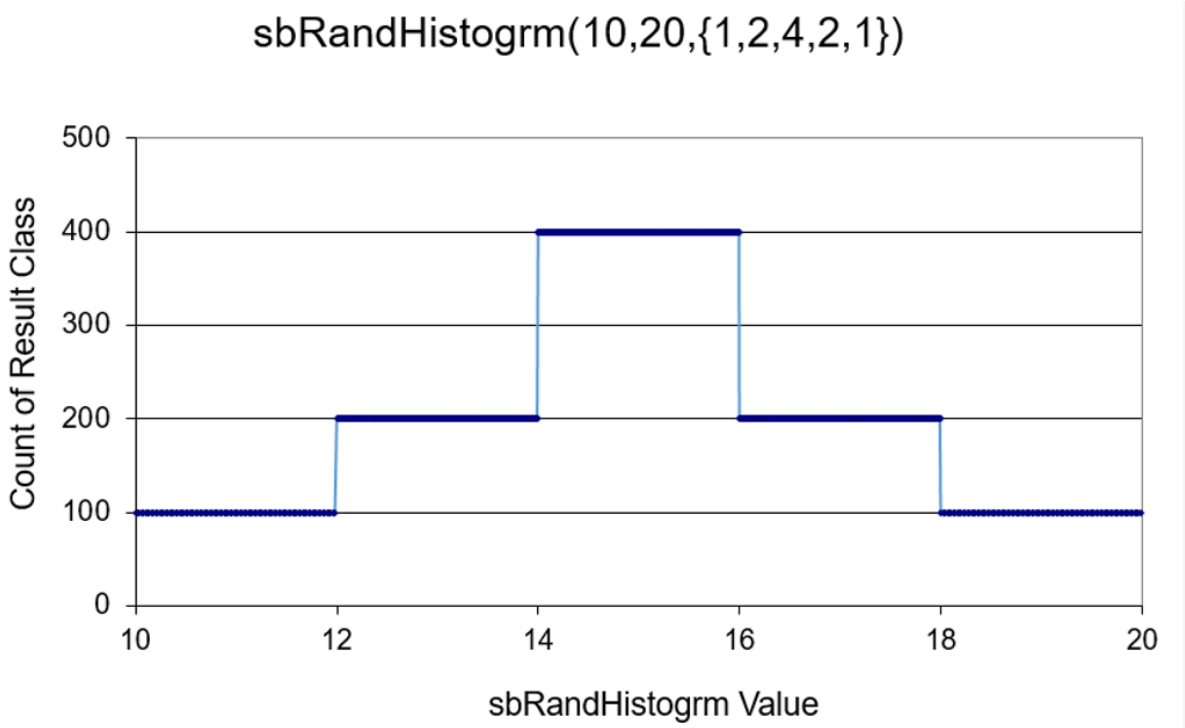


Figure 20: sbRandHistogrm

Programmcode `sbRandHistogrm`

```
Function sbRandHistogrm(dmin As Double, dMax As Double, _
            vWeight As Variant, Optional dRandom = 1#) As Double
'Specifies a histogram distribution with range dmin:dmax.
'This range is divided into vWeight.count classes. Each
'class has weight vWeight(i) reflecting the probability
'of occurrence of a value within the class.
'Similar to @Risk's function RiskHistogrm.
'(C) (P) by Bernd Plumhoff 18-Oct-2020 PB V1.01

Dim i As Long, n As Long, vW As Variant
Dim dRand As Double, dR As Double, dSumWeight As Double

With Application.WorksheetFunction
vW = .Transpose(.Transpose(vWeight))
End With

n = UBound(vW)
ReDim dSumWeightI(0 To n) As Double

dSumWeight = 0#
dSumWeightI(0) = 0#
For i = 1 To n
```

```
    If vW(i) < 0# Then 'A negative weight is an error
        sbRandHistogrm = CVErr(xlErrValue)
        Exit Function
    End If
    dSumWeight = dSumWeight + vW(i)  'Calculate sum of all weights
    dSumWeightI(i) = dSumWeight      'Calculate sum of weights till i
Next i

If dSumWeight = 0# Then  'Sum of weights has to be greater than zero
    sbRandHistogrm = CVErr(xlErrValue)
    Exit Function
End If

If dRandom = 1# Then
    dRand = Rnd()
Else
    dRand = dRandom
End If
dR = dSumWeight * dRand

i = n
Do While dR < dSumWeightI(i)
    i = i - 1
Loop

sbRandHistogrm = dmin + (dMax - dmin) * _
     (CDbl(i) + (dR - dSumWeightI(i)) / vW(i + 1)) / CDbl(n)

End Function
```

Ähnliche Verteilungen erzeugen die Programme `rww` und `redw`:

Programmcode `rww`

```
Function rww(ParamArray w() As Variant) As Double
'Produces random numbers with defined widths & weights
'Rww expects a vector of n random widths and weightings
'of type double and returns a random number of type double.
'This random number will lie in the given n-width-range of the
'[0,1)-interval with the given likelihood of the n weightings.
' Call Randomize before calling rww!
'(C) (P) by Bernd Plumhoff 06-Aug-2004 PB V0.50
'Examples:
'a) rww(1,0,1,1,8,0) will return a random number between 0.1 and 0.2
'b) rww(5,2,5,1) will return a random number between 0 and 0.5 twice as
'    often as a random number between 0.5 and 1.
'c) rww(1/3,0,1/3,1,1/3,0) will return a random number between
'    0.33333333333333 and 0.66666666666666.
'd) rww(5,15.4,3,7.7,2,0) would return a random value between
'    0 and 0.8, first 5 deciles with double likelihood than decile 6-8.
```

```
Dim i As Long
Dim swidths As Double
Dim sw As Double

If (UBound(w) + 1) Mod 2 <> 0 Then
  rww = -2       'No even number of arguments: Error
  Exit Function
End If

ReDim swidthsi(0 To (UBound(w) + 1) / 2 + 1) As Double
ReDim swi(0 To (UBound(w) + 1) / 2 + 1) As Double
ReDim weights(0 To (UBound(w) + 1) / 2) As Double
ReDim widths(0 To (UBound(w) + 1) / 2) As Double

swidths = 0#
sw = 0#
swi(0) = 0#
swidthsi(0) = 0#
For i = 0 To (UBound(w) - 1) / 2
  If w(2 * i) < 0# Then      'A negative width is an error
    rww = -3#
    Exit Function
  End If
  widths(i) = w(2 * i)
  swidths = swidths + widths(i)
  swidthsi(i + 1) = swidths
  If w(2 * i + 1) < 0# Then 'A negative weight is an error
    rww = -1#
    Exit Function
  End If
  weights(i) = w(2 * i + 1)
  If widths(i) > 0# Then
    sw = sw + weights(i)
  End If
  swi(i + 1) = sw
Next i
rww = sw * Rnd
'i = (UBound(w) - 1) / 2 + 1 'i already equals (UBound(w) - 1)/2 + 1,
                            'you may omit this statement.
Do While rww < swi(i)
  i = i - 1
Loop

rww = (swidthsi(i) + (rww - swi(i)) / weights(i) * widths(i)) / swidths

End Function
```

Programmcode `redw`

```
Function redw(ParamArray vWeights() As Variant) As Double
'Produces random numbers with equidistant weights. Redw expects a vector
'of n random weights of type double and returns a random number of type
'double. This random number will lie in the given equidistant n-split-range
'of the [0,1)-interval with the given likelihood of weightings.
'Call Randomize before calling redw!
'(C) (P) by Bernd Plumhoff 09-Dec-2009 PB V0.50
'Examples:
'a) redw(0,1,0,0,0,0,0,0,0,0) will return a random number d, 0.1 <= d < 0.2
'b) redw(2,1) will return a random number between 0 and 0.5 twice as
'    often as a random number between 0.5 and 1.
'c) redw(0,1,0) will return a random number d,
'    0.333333333333333 <= d < 0.666666666666666.
'd) redw(15.4,15.4,15.4,15.4,15.4,7.7,7.7,7.7,0,0) would return a random
'    value between 0 and 0.8, first 5 deciles with double likelihood than
'    decile 6-8.

Dim i As Long
Dim dw As Double
ReDim dwi(0 To UBound(vWeights) + 2) As Double

dw = 0#
dwi(0) = 0#
For i = 0 To UBound(vWeights)
  If vWeights(i) < 0# Then  'A negative weight is an error
    redw = CVErr(xlErrValue)
    Exit Function
  End If
  dw = dw + vWeights(i)      'Calculate sum of all weights
  dwi(i + 1) = dw            'Calculate sum of weights till i
Next i

redw = dw * Rnd

'i = UBound(vWeights) + 1 'i already equals UBound(vWeights) + 1,
                         'you may omit this statement.
Do While redw < dwi(i)
  i = i - 1
Loop

redw = (CDbl(i) + (redw - dwi(i)) / vWeights(i)) / _
       (CDbl(UBound(vWeights) + 1))

End Function
```

4.2.3 sbRandTriang

Die Dreiecksverteilung ist eine stetige Wahrscheinlichkeitsverteilung, deren Dichtefunktion wie ein Dreieck aussieht. Es ist eine einfache Verteilung, weil Sie lediglich ihr Minimum, ihren Median und ihr Maximum kennen müssen:

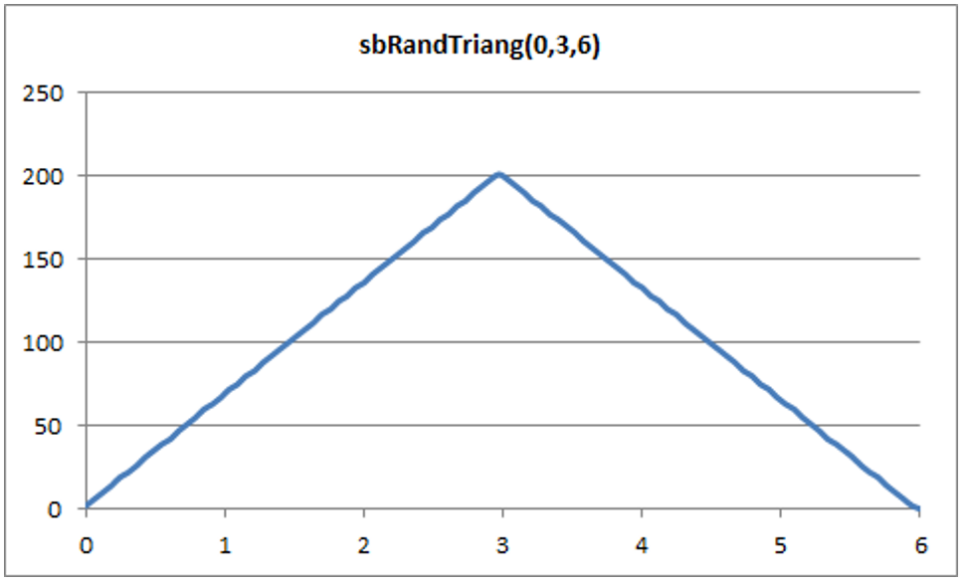


Figure 21: sbRandTriang

Im englischen Sprachraum wird diese Verteilung auch Distribution of Missing Data genannt, weil für ihre Bestimmung nur ein Minimum an Information benötigt wird. Diese Verteilung wird oft zur Simulation von Expertenwissen eingesetzt oder wenn eine genauere Datenermittlung zu schwierig oder zu teuer erscheint.

Programmcode `sbRandTriang`

```
Function sbRandTriang(dMin As Double, dMode As Double, _
    dMax As Double, Optional dRandom = 1#) As Double
'Generates a random number, Triang distributed
'[see Vose: Risk Analysis, 2nd ed., p. 128]
'(C) (P) by Bernd Plumhoff 30-Aug-2024 PB V0.32
'Similar to @RISK's (C) RiskTriang function.
'sbRandTriang(minimum,mode,maximum) specifies a triangular
'distribution with three points - a minimum, a mode and
'a maximum. The skew of the triangular distribution is
'driven by the distance of the mode from the minimum and
'from the maximum. Reducing the distance from mode to
'minimum will increase the skew.
'Please ensure that you execute Randomize before you call
'this function for the first time.
Dim dRand As Double, dc_a As Double, db_a As Double
Dim dc_b As Double

If dMode < dMin Or dMax < dMode Then
  sbRandTriang = CVErr(xlErrValue)
  Exit Function
```

```
End If
If dMin = dMax Then
  sbRandTriang = dMin 'Triangle is just one point
  Exit Function
End If
dc_a = dMax - dMin
db_a = dMode - dMin
dc_b = dMax - dMode
If dRandom = 1# Then
  dRand = Rnd()
Else
  dRand = dRandom
End If
If dRand < db_a / dc_a Then
  sbRandTriang = dMin + Sqr(dRand * db_a * dc_a)
Else
  sbRandTriang = dMax - Sqr((1# - dRand) * dc_a * dc_b)
End If
End Function
```

4.2.4 `sbRandTrigen`

`sbRandTrigen` bestimmt eine Dreiecksverteilung mit drei Punkten: einen mit der höchsten Wahr¬schein¬lichkeit und zwei weitere an den spezifizierten unteren und oberen Perzentilen. Diese Perzen¬tilparameter haben Werte zwischen 0 und 100 und repräsentieren die kumulierten Wahr¬schein¬lichkeiten für diese unteren und oberen Werte. Dies ist manchmal nützlich, wenn man die absolut kleinsten und größten Werte der Verteilung nicht kennt oder nicht leicht bestimmen kann.

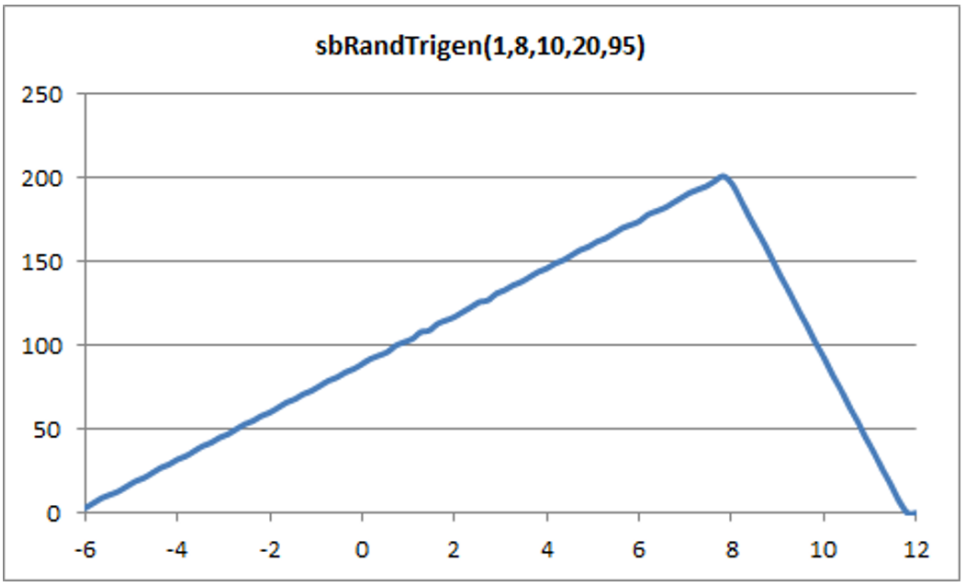


Figure 22: sbRandTrigen

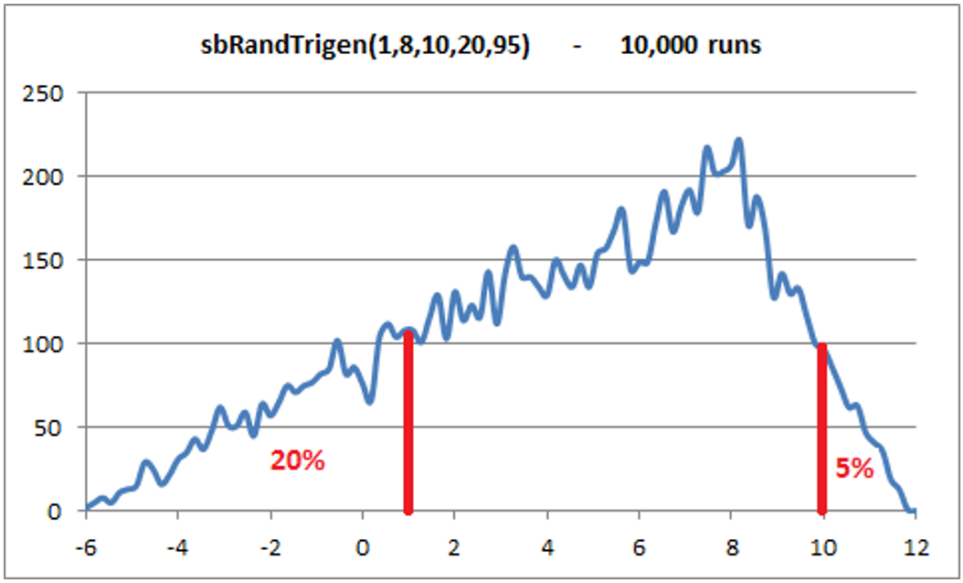
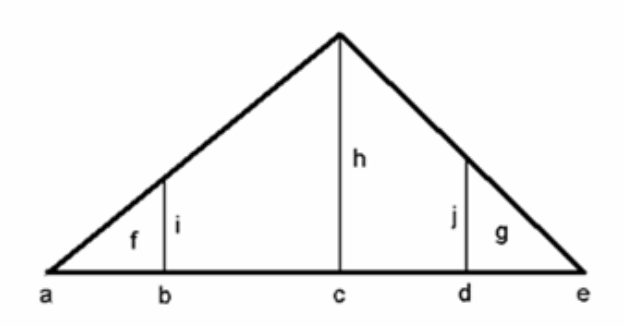


Figure 23: sbRandTrigen 10.000 Iterationen

Herleitung des Algorithmus



Given: Points b, c and d; Areas f and g. Area of the triangle is 1.

To be computed: Points a and e.

Triangle area is 1: [i] $h(e-a)=2 \Leftrightarrow h=\frac{2}{e-a}$

[ii] $\frac{h}{c-a}=\frac{i}{b-a} \Leftrightarrow i=h\frac{b-a}{c-a}=\frac{2(b-a)}{(e-a)(c-a)}$

[iii] $\frac{h}{e-c}=\frac{j}{e-d} \Leftrightarrow j=h\frac{e-d}{e-c}=\frac{2(e-d)}{(e-a)(e-c)}$

[iv] $2g=j(e-d)=\frac{2(e-d)^2}{(e-a)(e-c)} \Rightarrow a=e-\frac{(e-d)^2}{g(e-c)}$

[v] $2f=i(b-a)=\frac{2(b-a)^2}{(e-a)(c-a)}$

Substituting a in [v] and putting everything on one side gives:

$$\left\{b-e+\frac{(e-d)^2}{g(e-c)}\right\}^2-f\left\{e-e+\frac{(e-d)^2}{g(e-c)}\right\}\left\{c-e+\frac{(e-d)^2}{g(e-c)}\right\}=0$$

$\Leftrightarrow$

$$\frac{\left((b-e)g(e-c)+(e-d)^2\right)^2}{\left(g(e-c)\right)^2}-f\frac{(e-d)^2}{g(e-c)}\frac{(c-e)(g(e-c)+(e-d)^2}{g(e-c)}=0$$

$\Leftrightarrow$ [g≠0 and e≠c]

Figure 24: sbRandTrigen - Herleitung des Algorithmus Seite 1

$$\left(g(b-c)(e-c)+(e-d)^2\right)^2 - f(e-d)^2\left((e-d)^2 - g(e-c)^2\right)=0$$

$$\Leftrightarrow$$

$$g^2(b-e)^2(e-c)^2 + 2g(b-e)(e-c)(e-d)^2 + (e-d)^4 - f(e-d)^4 - fg(e-d)^2(e-c)^2 = 0$$

$$\Leftrightarrow$$

$$\begin{aligned}
&e^4(fg - f + 1 - 2g + g^2)\\
&+e^3(-2fgd - 2fgc - 4d + 4fd + 2gc + 4gd + 2gb - 2g^2c - 2g^2b)\\
&+e^2(fgd^2 + 4f\,\text{gcd} + fgc^2 - 6fd^2 + 6d^2 - 4\,\text{gcd} - 2gd^2 - 2gbc - 4gbd + g^2c^2 + 4g^2bc + g^2b^2)\\
&+e(-2f\,\text{gcd}^2 - 2fgc^2d + 4d^3f - 4d^3 + 2\,\text{gcd}^2 + 4gbcd + 2gbd^2 - 2g^2bc^2 - 2g^2b^2c)\\
&+(fgc^2d^2 - fd^4 + d^4 - 2gbcd^2 + g^2b^2c^2 = 0
\end{aligned}$$

The correct solution for e can now be calculated with a Newton iteration with a starting value > e, for example with the start value

$$d + \frac{d-c}{(1-g)^2}$$

If g=0 and f>0 then switch f and g, b and d and later the solution a and e.

Figure 25: sbRandTrigen - Herleitung des Algorithmus Seite 2

Programmcode `sbRandTrigen`

Bitte beachten Sie, daß `sbRandTrigen` `sbRandTriang` benötigt und aufruft.

```
Function sbRandTrigen(dBottom As Double, dMode As Double, _
    dTop As Double, dBottomPerc As Double, _
    dTopPerc As Double, Optional dRandom = 1#) As Double
'Generates dMin random number, Triang distributed
'with given first and last decile
'[see Vose: Risk Analysis, 2nd ed., p. 129]
'(C) (P) by Bernd Plumhoff 19-Nov-2011 PB V0.32
'Similar to @RISK's (C) RiskTrigen function.
'sbRandTrigen(bottom, mode, top, bottom percentile, top percentile)
'specifies a triangular distribution with three points - one
'at the mode and two at the specified bottom and top percentiles.
'The bottom percentile and top percentile are values between
'0 and 100. Each percentile value gives the percentile of the
'total area under the triangle that is on the left side of the
'given point.
'Example:
'sbRandTrigen(1,8,10,20,95) will call
'sbRandTriang(-6.13212712795534, 8, 11.8648937411641).
'Please ensure that you execute Randomize before you call
'this function for the first time.

Static dBottomLast As Double
Static dModeLast As Double
Static dTopLast As Double
Static dBottomPercLast As Double
Static dTopPercLast As Double
Static dMin As Double
```

```
Static dMax As Double
Dim dMaxNew As Double
Dim da0 As Double, da1 As Double, da2 As Double
Dim da3 As Double, da4 As Double
Dim dfe As Double, df1e As Double
Dim dBottomPerc2 As Double, dTopPerc2 As Double
Dim i As Long

If dBottom = dBottomLast And dMode = dModeLast And dTop = dTopLast _
   And dBottomPerc = dBottomPercLast And dTopPerc = dTopPercLast _
   And Not IsError(dMin) Then
    sbRandTrigen = sbRandTriang(dMin, dMode, dMax, dRandom)
    Exit Function
End If

dBottomLast = dBottom
dModeLast = dMode
dTopLast = dTop
dBottomPercLast = dBottomPerc
dTopPercLast = dTopPerc

dBottomPerc2 = dBottomPerc / 100#
dTopPerc2 = 1# - dTopPerc / 100#
If dMode <= dBottom Or dTop <= dMode Then
    dMin = CVErr(xlErrValue) 'Trigger rerun next time
    sbRandTrigen = CVErr(xlErrValue)
    Exit Function
End If
If dBottomPerc2 < 0# Or dTopPerc2 < 0# Then
    dMin = CVErr(xlErrDiv0) 'Trigger rerun next time
    sbRandTrigen = CVErr(xlErrValue)
    Exit Function
End If

If dTopPerc2 = 0# Then
    If dBottomPerc2 = 0# Then
        sbRandTrigen = sbRandTriang(dBottom, dMode, dTop, dRandom)
        Exit Function
    End If
    sbRandTrigen = sbRandTrigen(dBottom, dMode, dTop, _
                   dBottomPerc2, dTopPerc2)
    Exit Function
End If

da4 = dBottomPerc2 * dTopPerc2 - dBottomPerc2 + 1# - _
      2# * dTopPerc2 + dTopPerc2 ^ 2#
da3 = -2# * dBottomPerc2 * dTopPerc2 * dTop - _
      2# * dBottomPerc2 * dTopPerc2 * dMode - _
      4# * dTop + 4# * dBottomPerc2 * dTop + _
      2# * dTopPerc2 * dMode + 4# * dTopPerc2 * _
      dTop + 2# * dTopPerc2 * dBottom - 2# * dTopPerc2 ^ 2# * dMode - _
```

```
        2# * dTopPerc2 ^ 2# * dBottom
da2 = dBottomPerc2 * dTopPerc2 * dTop ^ 2# + _
        4# * dBottomPerc2 * dTopPerc2 * dMode * _
        dTop + dBottomPerc2 * dTopPerc2 * dMode ^ 2# - _
        6# * dBottomPerc2 * dTop ^ 2# + _
        6# * dTop ^ 2# - 4# * dTopPerc2 * dMode * dTop - _
        2# * dTopPerc2 * dTop ^ 2# - 2# * _
        dTopPerc2 * dBottom * dMode - 4# * dTopPerc2 * _
        dBottom * dTop + dTopPerc2 ^ 2# * _
        dMode ^ 2# + 4# * dTopPerc2 ^ 2# * dBottom * _
        dMode + dTopPerc2 ^ 2# * dBottom ^ 2#
da1 = -2# * dBottomPerc2 * dTopPerc2 * dMode * _
        dTop ^ 2# - 2# * dBottomPerc2 * dTopPerc2 * _
        dMode ^ 2# * dTop + 4# * dTop ^ 3# * dBottomPerc2 - _
        4# * dTop ^ 3# + 2# * dTopPerc2 * _
        dMode * dTop ^ 2# + 4# * dTopPerc2 * dBottom * _
        dMode * dTop + 2# * dTopPerc2 * _
        dBottom * dTop ^ 2# - 2# * dTopPerc2 ^ 2# * _
        dBottom * dMode ^ 2# - 2# * _
        dTopPerc2 ^ 2# * dBottom ^ 2# * dMode
da0 = dBottomPerc2 * dTopPerc2 * dMode ^ 2# * dTop ^ 2# - _
        dBottomPerc2 * dTop ^ 4# + dTop ^ 4# - _
        2# * dTopPerc2 * dBottom * dMode * dTop ^ 2# + _
        dTopPerc2 ^ 2# * dBottom ^ 2# * dMode ^ 2#

dMax = dTop + (dTop - dMode) / (1# - dTopPerc2) ^ 2#

'Newton iteration
Do While Abs(dMaxNew - dMax) > 0.000000000001
    i = i + 1
    If i > 30 Then
        If Abs(dfe) > 0.000000000001 Then
            dMin = CVErr(xlErrDiv0) 'Trigger rerun next time
            sbRandTrigen = CVErr(xlErrValue)
            Exit Function
        Else
            Exit Do
        End If
    End If
    dMaxNew = dMax
    dfe = da4 * dMaxNew ^ 4# + da3 * dMaxNew ^ 3# + _
          da2 * dMaxNew ^ 2# + da1 * dMaxNew + da0
    df1e = 4# * da4 * dMaxNew ^ 3# + 3# * da3# * _
           dMaxNew ^ 2# + 2# * da2 * dMaxNew + da1
    dMax = dMax - dfe / df1e
Loop

dMin = dMax - (dMax - dTop) ^ 2# / dTopPerc2 / (dMax - dMode)
sbRandTrigen = sbRandTriang(dMin, dMode, dMax, dRandom)

End Function
```

4.2.5 sbRandCauchy

Wenn Sie simulieren wollen, wie Partikel ausgehend von einem festen Punkt auf eine Gerade treffen, können Sie eine Cauchy Verteilung verwenden. Diese Verteilung wird manchmal auch Lorentz Verteilung genannt. Auch der Quotient zweier $(0, 1)$ Normalverteilungen führt zu einer Cauchy Verteilung.

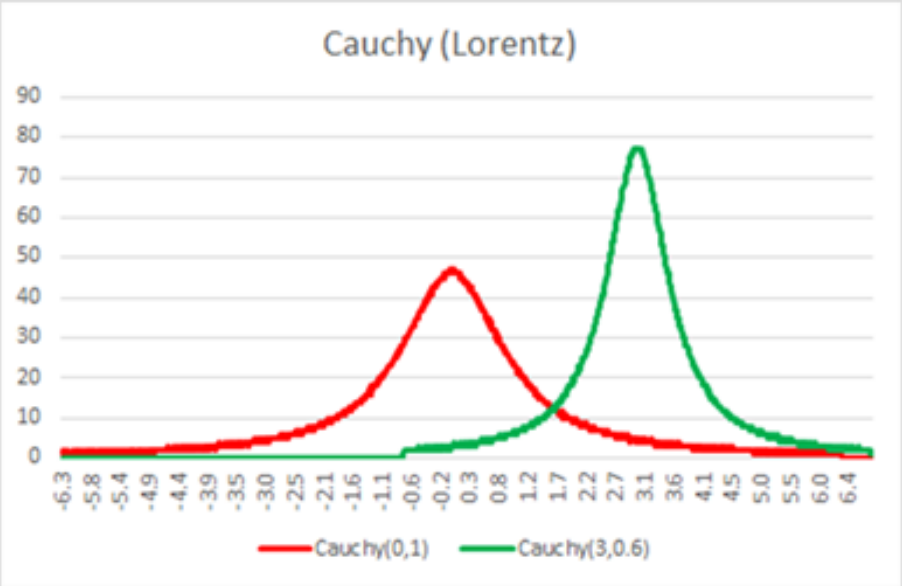


Figure 26: sbRandCauchy

Programmcode `sbRandCauchy`

```
Const GCPi = 3.14159265358979

Function sbRandCauchy(dLocation As Double, dScale As Double, _
    Optional dRandom = 1#) As Double
'(C) (P) by Bernd Plumhoff 03-Nov-2020 PB V0.2
Static bRandomized As Boolean
Dim dRand As Double
If dRandom < 0# Or dRandom > 1# Or dScale <= 0# Then
    sbRandCauchy = CVErr(xlErrValue)
    Exit Function
End If
If Not bRandomized Then
    Randomize
    bRandomized = True
End If
If dRandom = 1# Then
    dRand = Rnd()
Else
    dRand = dRandom
End If
sbRandCauchy = dLocation + dScale * Tan((dRand - 0.5) * GCPi)
End Function
```

4.2.6 sbRandCDFInv

Sie können Zufallszahlen mit einer gewünschten Verteilung leicht erzeugen, wenn die inverse Verteilungsfunktion explizit vorliegt.

Ein Beispiel einer geschichteten Stichprobe:

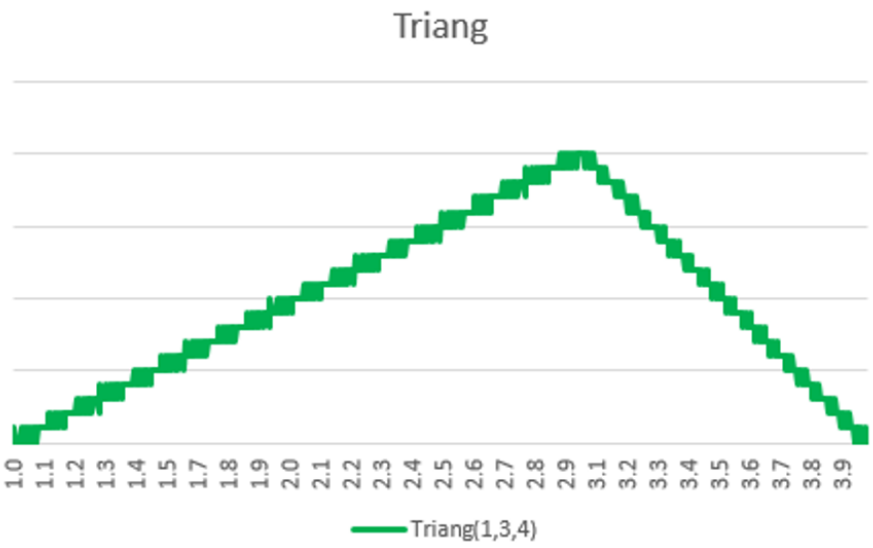


Figure 27: sbRandCDFInv

Ohne die explizit vorliegende inverse Verteilungsfunktion können Sie eine lineare Approximation mit der Function `sbRandPDF` anwenden.

Programmcode `sbRandCFDInv`

```
Function sbRandCDFInv(dParam1 As Double, dParam2 As Double, _
    dParam3 As Double, Optional dRandom = 1#) As Double
'(C) (P) by Bernd Plumhoff  03-Nov-2020 PB V0.2
Static bRandomized As Boolean
Dim dRand As Double
If dRandom < 0# Or dRandom > 1# Then
  sbRandCDFInv = CVErr(xlErrValue)
  Exit Function
End If
If Not bRandomized Then
  Randomize
  bRandomized = True
End If
If dRandom = 1# Then
    dRand = Rnd()
Else
    dRand = dRandom
End If
'Here you need to define the inverse of the cumulative distribution function
sbRandCDFInv = sbRandTriang(dParam1, dParam2, dParam3, dRand)
End Function
```

4.2.7 sbRandPDF

Mit einer expliziten Form der inversen Verteilungsfunktion sollten Sie `sbRandCDFInv` verwenden. Aber wenn eine solche nicht vorliegt, können Sie mit einer linearen Approximation die Funktion `sbRandPDF` nutzen. Unglücklicherweise ist dieser Ansatz sehr rechenintensiv, selbst wenn Sie die Anzahl der Punkte bei identischen oder nahezu identischen Steigungen reduzieren können.

Programmcode `sbRandPDF`

```
Function sbRandPDF(Optional dParam1, Optional dParam2, _
    Optional dParam3, Optional dRandom = 1#) As Double
'(C) (P) by Bernd Plumhoff  12-Sep-2014 PB V0.15
Dim dRand As Double
Dim i As Long
Static dPar1 As Double
Static dPar2 As Double
Static dPar3 As Double
Static vX(0 To 1000) As Variant
Static vY(0 To 1000) As Variant
If dRandom < 0# Or dRandom > 1# Then
  sbRandPDF = CVErr(xlErrValue)
  Exit Function
End If
If dRandom = 1# Then
  dRand = Rnd()
Else
  dRand = dRandom
End If
If dParam1 <> dPar1 Or dParam2 <> dPar2 Or dParam3 <> dPar3 Then
  dPar1 = dParam1
  dPar2 = dParam2
  dPar3 = dParam3
  'Initialize RandGeneral call parameters
  For i = 0 To 1000
    vX(i) = dPar1 + i * (dPar3 - dPar1) / 1000#
    'Now we can insert an arbitrary PDF function
    If vX(i) < dPar2 Then
      vY(i) = (vX(i) - dPar1) / ((dPar3 - dPar1) * (dPar2 - dPar1))
      If vY(i) < 0# Then vY(i) = 0#
    Else
      vY(i) = (dPar3 - vX(i)) / ((dPar3 - dPar1) * (dPar3 - dPar2))
      If vY(i) < 0# Then vY(i) = 0#
    End If
  Next i
End If
'Depending on the PDF input range you need to feed start
'and end values to sbRandGeneral
sbRandPDF = sbRandGeneral(dPar1, dPar3, vX, vY, dRand)
End Function
```

4.2.8 `sbRandCumulative`

Wenn Sie eine schrittweise kumulierte Verteilungsfunktion von Zufallszahlen erzeugen müssen, können Sie diese benutzerdefinierte Funktion `sbRandCumulative` verwenden. Die Herleitung des Algorithmus ist analog zu `sbRandGeneral`.

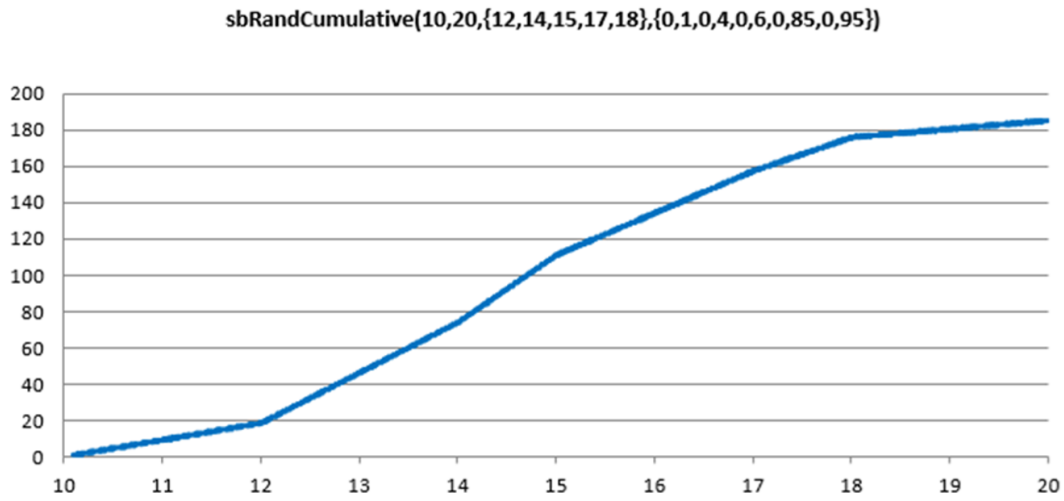


Figure 28: sbRandCumulative

Programmcode `sbRandCumulative`

```
Function sbRandCumulative(dMin As Double, dMax As Double, _
    vXi As Variant, vWi As Variant, Optional dRandom = 1#) As Double
'Generates a random number, Cumulative distributed.
'[see Vose: Risk Analysis, 2nd ed., p. 109]
'(C) (P) by Bernd Plumhoff 23-Dec-2020 PB V0.50
'Similar to @RISK's (C) RiskCumulative function.
Static bRandomized As Boolean, i As Long
Dim dA As Double, dRand As Double, dSgn As Double

If vWi.Count <> vXi.Count Then
    sbRandCumulative = CVErr(xlErrValue)
    Exit Function
End If
ReDim dX(0 To vXi.Count + 1) As Double
ReDim dW(0 To vWi.Count + 1) As Double

dX(0) = dMin
dX(UBound(dX)) = dMax
dW(0) = 0#
dW(UBound(dW)) = 1#
For i = 1 To vXi.Count
  dX(i) = vXi(i)
  dW(i) = vWi(i)
  If dW(i) < dW(i - 1) Then
    'Weights need to be monotonously increasing
```

```
    sbRandCumulative = CVErr(xlErrValue)
    Exit Function
  End If
Next i
If dW(UBound(dW)) < dW(UBound(dW) - 1) Then
  'Weights need to be monotonously increasing
  sbRandCumulative = CVErr(xlErrValue)
  Exit Function
End If

'Calculate area
dA = 0#
For i = 0 To UBound(dX) - 1
  If dX(i) >= dX(i + 1) Or dW(i) < 0# Then
    sbRandCumulative = CVErr(xlErrValue)
    Exit Function
  End If
  dA = dA + (dX(i + 1) - dX(i)) * (dW(i + 1) + dW(i)) / 2#
Next i

'Normalise weights to set area to 1
For i = 1 To UBound(dW)
  dW(i) = dW(i) / dA
Next i

ReDim dF(0 To UBound(dX)) As Double
'Calculate border points of value ranges for
'cumulative inverse function
dF(0) = 0#
dA = 0#
For i = 0 To UBound(dX) - 1
  dA = dA + (dX(i + 1) - dX(i)) * (dW(i + 1) + dW(i)) / 2#
  dF(i + 1) = dA
Next i

If dRandom = 1# Then
  If Not bRandomized Then
    Randomize
    bRandomized = True
  End If
  dRand = Rnd()
Else
  dRand = dRandom
End If

i = 1
Do While dF(i) <= dRand
  i = i + 1
Loop
dSgn = Sgn(dW(i) - dW(i - 1))
If dSgn = 0# Then
```

```
    sbRandCumulative = dX(i - 1) + (dRand - dF(i - 1)) / _
                       (dF(i) - dF(i - 1)) * (dX(i) - dX(i - 1))
Else
    sbRandCumulative = dX(i - 1) + _
                       dSgn * Sqr((dRand - dF(i - 1)) * _
                       2# * (dX(i) - dX(i - 1)) / (dW(i) - dW(i - 1)) + _
                       (dW(i - 1) * (dX(i) - dX(i - 1)) / _
                       (dW(i) - dW(i - 1))) ^ 2#) - _
                       dW(i - 1) * (dX(i) - dX(i - 1)) / (dW(i) - dW(i - 1))
End If

End Function
```

5 Praktische Anwendungen von Gleitkommazufallszahlen

5.1 Zufallszahlen mit der Summe 1 – `sbRandSum1`

Wir erzeugen n Zufallszahlen mit einer Bedingung: Die Summe aller erzeugten Zahlen soll 1 sein. Dies kann man mit vielen verschiedenen Ansätzen erreichen.

Drei mögliche Ansätze sind:

• Reduziere den Freiheitsgrad sukzessive: Erzeuge die erste Zufallszahl, danach die zweite im Bereich $[0, 1 - Erste\_Zahl)$, dann die dritte $[0, 1 - Erste\_Zahl - Zweite\_Zahl)$, ..., die letzte Zahl muss schließlich gleich $1 - Summe\_aller\_anderen\_Zahlen$ sein.

• Erzeuge n Zufallszahlen und dividiere sie durch ihre Summe.

• Simuliere die Teilung einer Torte: wo immer man schneidet, man kann nicht mehr und nicht weniger als die ganze Torte verteilen.

Die resultierenden Verteilungen sehen so aus:

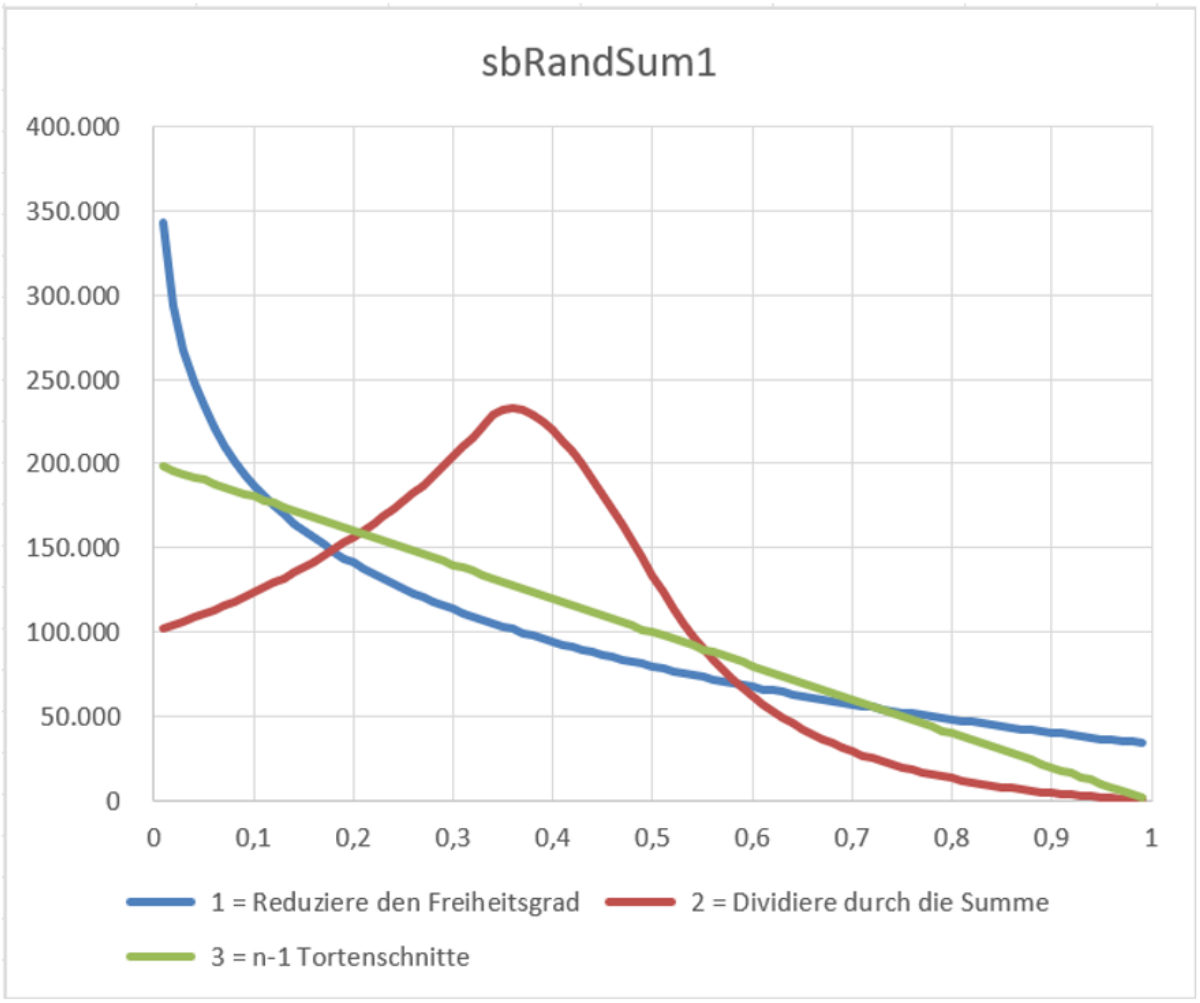


Figure 29: sbRandSum1

Sie können leicht erkennen, dass der weitverbreitete Ansatz mit n Zufallszahlen dividiert durch ihre Summe eine schlechte Wahl ist: Sie erhalten meist Zahlen zwischen 0,2 und 0,5 (siehe die rote Kurve).

Bemerkung: Ein genereller Ansatz wäre die Dirichlet Verteilung. Für eine Implementierung mit Python siehe numpy - für unsere obige Ausgabe müsste der Parameter size auf 1 gesetzt werden: `numpy.random.dirichlet`

Programmcode `sbRandSum1`

```
Function sbRandSum1(ByVal lDist As Long, Optional ByVal lCount As Long, _
    Optional bVolatile As Boolean = False) As Variant
'sbRandSum1 produces lCount (or the number of selected cells if
'called from a worksheet range) random numbers which sum up to 1.
'Possible values of lDist to specify desired distribution:
'        1 = reduce degree of freedom linearly
'        2 = divide lCount random numbers by their sum
'        3 = lCount-1 random cuts of (0,1)-interval
'If TypeName(Application.Caller) <> "Range" Then lCount has to be set.
'It specifies the count of summands which have to have the sum of 1.
'(C) (P) by Bernd Plumhoff 02-Aug-2020 PB V0.4
Static bRandomized As Boolean
Dim bRowWise As Boolean, vA As Variant, vT  As Variant
Dim i As Long, j As Long, dSum As Double

If bVolatile Then Application.Volatile
If Not bRandomized Then Randomize: bRandomized = True
If TypeName(Application.Caller) <> "Range" Then
  If lCount < 1 Then
    sbRandSum1 = CVErr(xlErrRef)
    Exit Function
  End If
  bRowWise = False
Else
  With Application.Caller
    lCount = .Rows.Count
    bRowWise = True
    If lCount < .Columns.Count Then
      lCount = .Columns.Count
      bRowWise = False
    End If
    If lCount = 1 Then
      sbRandSum1 = 1
      Exit Function
    End If
  End With
End If
ReDim vA(1 To lCount) As Variant
Select Case lDist
  Case 1
    ReDim nRand(1 To lCount) As Long
    For i = 1 To lCount
      nRand(i) = i
    Next i
    For i = 1 To lCount - 1
      j = Int(Rnd * (lCount - i + 1)) + i
      vA(nRand(j)) = Rnd * (1# - dSum)
      dSum = dSum + vA(nRand(j))
      nRand(j) = nRand(i)
```

```
    Next i
    vA(nRand(lCount)) = 1# - dSum
  Case 2
    For i = 1 To lCount
      vA(i) = Rnd
      dSum = dSum + vA(i)
    Next i
    For i = 1 To lCount
      vA(i) = vA(i) / dSum
    Next i
  Case 3
    For i = 1 To lCount - 1
      vA(i) = Rnd
      j = i - 1
      Do While j > 0
        If vA(j) > vA(j + 1) Then
          vT = vA(j + 1)
          vA(j + 1) = vA(j)
          vA(j) = vT
        End If
        j = j - 1
      Loop
    Next i
    vA(lCount) = 1# - vA(lCount - 1)
    i = lCount - 1
    Do While i > 1
      vA(i) = vA(i) - vA(i - 1)
      i = i - 1
    Loop
  Case Else
    sbRandSum1 = CVErr(xlErrValue)
    Exit Function
End Select
If bRowWise Then vA = Application.WorksheetFunction.Transpose(vA)
sbRandSum1 = vA
End Function
```

5.2 Beispiel für ein exaktes Verhältnis von Zufallszahlen - `sbExactRandHistogrm`

Es ist recht leicht, einen "unfairen" Würfel zu simulieren. Wenn wir z. B. im Mittel die 6 doppelt so häufig wie alle anderen Zahlen 1 bis 5 erhalten wollen, geben Sie in Zelle A1 ein: $= MIN(GANZZAHL(ZUFALLSZAHL() * 7 + 1); 6)$.

Aber wenn Sie einen Würfel 7 mal würfeln wollen und alle Zahlen von 1 bis 5 genau einmal und die 6 genau zweimal erscheinen soll?

Eine allgemeine Lösung:

	A	B	C	D	E	F	G	H	I	J	K	L
1				Nur statistische Wahrscheinlichkeit						Total		
2	Farbe	Häufigkeit	Pos / Iteration	Eins	Zwei	Drei	Vier	Fünf	Sechs	Grün	Gelb	Rot
3	Grün	50,00%	1	Grün	Grün	Grün	Grün	Grün	Rot	5	0	1
4	Gelb	33,33%	2	Gelb	Gelb	Grün	Grün	Gelb	Gelb	2	4	0
5	Rot	16,67%	3	Rot	Rot	Rot	Grün	Grün	Gelb	2	1	3
6			4	Gelb	Gelb	Grün	Grün	Gelb	Grün	3	3	0
7			5	Grün	Grün	Gelb	Grün	Gelb	Grün	4	2	0
8			6	Rot	Rot	Grün	Grün	Gelb	Grün	3	1	2
9			7	Gelb	Grün	Grün	Gelb	Grün	Rot	3	2	1
10			8	Grün	Gelb	Grün	Gelb	Grün	Gelb	3	3	0
11			9	Rot	Gelb	Gelb	Gelb	Gelb	Gelb	0	5	1
12			10	Grün	Grün	Grün	Gelb	Gelb	Rot	3	2	1
13									Total:	28	23	9
14									Sollte stochastisch ergeben:	30	20	10
16				Genaue Häufigkeit						Total		
17			Pos / Iteration	Eins	Zwei	Drei	Vier	Fünf	Sechs	Grün	Gelb	Rot
18			1	Grün	Grün	Gelb	Grün	Rot	Gelb	3	2	1
19			2	Gelb	Grün	Gelb	Grün	Grün	Rot	3	2	1
20			3	Grün	Grün	Rot	Gelb	Grün	Gelb	3	2	1
21			4	Rot	Grün	Grün	Grün	Gelb	Gelb	3	2	1
22			5	Gelb	Grün	Rot	Grün	Grün	Gelb	3	2	1
23			6	Rot	Grün	Grün	Grün	Gelb	Gelb	3	2	1
24			7	Grün	Grün	Gelb	Gelb	Grün	Rot	3	2	1
25			8	Grün	Grün	Rot	Gelb	Gelb	Grün	3	2	1
26			9	Gelb	Rot	Gelb	Grün	Grün	Grün	3	2	1
27			10	Grün	Rot	Grün	Gelb	Gelb	Grün	3	2	1
28									Total:	30	20	10
29									Sollte stochastisch ergeben:	30	20	10

Figure 30: sbExactRandHistogrm

Tabellenblattformeln		
Bereich	Formel	
D3:I12	**D3**	=INDEX($A$3:$A$5;GANZZAHL(sbRandHistogrm(1;4;$B$3:$B$5)))
J3:L12;J18:L27	**J3**	=ZÄHLENWENN($D3:$I3;J$2)
J13:L13;J28:L28	**J13**	=SUMME(J3:J12)
J14;J29	**J14**	=ANZAHL2($D$3:$I$12)*MTRANS($B$3:$B$5)
D18:D27	**D18**	=INDEX($A$3:$A$5;GANZZAHL(sbExactRandHistogrm(6;1;4;$B$3:$B$5)))

Figure 31: sbExactRandHistogrm Formeln

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 - siehe Appendix A.

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.

Programmcode `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
```

5.3 Zufallszahlen die sich nicht sofort wiederholen – `sbRandomNoRepeatBeforeN`

Wenn Sie eine zufällige Auswahl aus einigen Elementen (Namen, Zahlen, was auch immer) treffen möchten, und diese Elemente mehr als einmal auftreten können, aber nicht innerhalb der nächsten N Ziehungen wieder erscheinen sollen, hilft Ihnen die hier gezeigte Funktion weiter. Sie ruft `sbRandHistogrm` auf, das Sie in Ihren VBA-Code einfügen müssen. Diese Funktion ist ein relativ fortgeschrittenes Beispiel dafür, wie man mit Skripting-Dictionaries (assoziativen Arrays) arbeitet. Sie zeigt:

- Wie man Werte an Namen (Schlüssel) hängt
- Wie man Schlüssel/Wert-Paare hinzufügt, entfernt und ändert
- Wie man Werte nachschlägt
- Wie man auf das gesamte Werteset als Array zugreift
- Wie man testet, ob ein Dictionary leer ist (d.h. keine Einträge enthält)
- Wie man auf Werte mit numerischen Indizes zugreift (kann verwendet werden, um numerisch durch das Schlüssel- oder Werteset zu iterieren)

Ein Beispiel für N = 3:

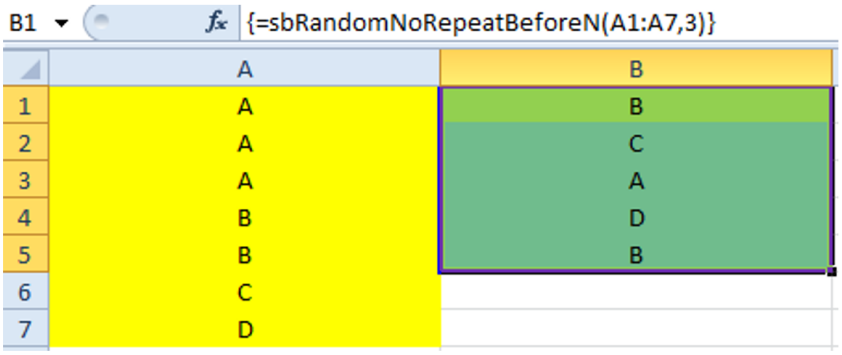


Figure 32: sbRandomNoRepeatBeforeN

Bitte beachten Sie, dass bei gewissen Ziehungsreihenfolgen keine Lösung existiert. Mit denselben Eingaben wie zuvor, aber bei einer anderen Zufallsreihenfolge, könnte man auch erhalten:

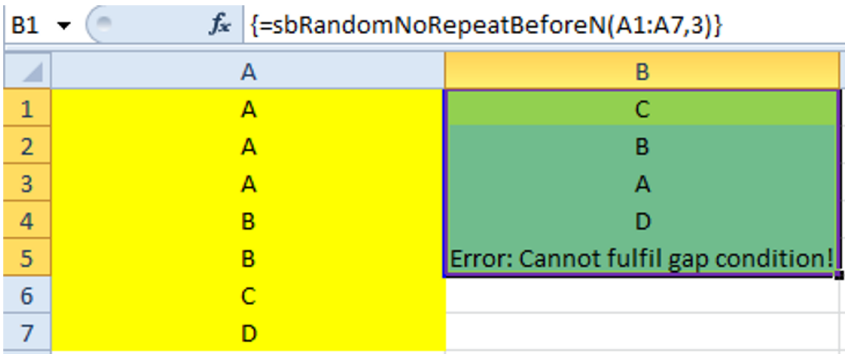


Figure 33: sbRandomNoRepeatBeforeN Fehlerbeispiel

Die einzelnen "C" und "D" Einträge wurden bereits gezogen und ein zusätzliches "A" or "B" würde gegen die Bedingung verstoßen, die nächsten drei Ziehungen nicht wieder (in B5) zu erscheinen.

Programmcode `sbRandomNoRepeatBeforeN`

Diese Funktion benötigt (ruft auf) die Funktion `sbRandHistoGrm`.

```
Function sbRandomNoRepeatBeforeN(rInput As Range, lN As Long) As Variant
'From names in rInput we create a random draw into the
'selected cells this function is called from as an array
'function (entered with CTRL + SHIFT + ENTER) so that no
'name re-appears in the next lN cells.
'This function needs / calls sbRandHistogrm.
'(C) (P) by Bernd Plumhoff 20-Dec-2012 PB V0.10

Dim i As Long, j As Long, lDrawn As Long, lCol As Long, lRow As Long
Dim obj As Object
ReDim vNames(1 To lN) As Variant
ReDim lCount(1 To lN) As Long

With Application
'Parameter check
If TypeName(.Caller) <> "Range" Then
  sbRandomNoRepeatBeforeN = CVErr(xlErrRef)
  Exit Function
End If
If .Caller.Rows.Count * .Caller.Columns.Count > rInput.Count Then
  sbRandomNoRepeatBeforeN = CVErr(xlErrValue)
  Exit Function
End If

ReDim vR(1 To .Caller.Rows.Count, 1 To .Caller.Columns.Count)

'First read in all names. They may appear more than once.
Set obj = CreateObject("Scripting.Dictionary")
```

```
For i = 1 To rInput.Count
  obj.Item(rInput(i).Value) = obj.Item(rInput(i).Value) + 1
Next i

'Now apply the draws. After each draw take drawn name out
'for next lN draws.
i = 1
For lRow = 1 To UBound(vR, 1)
  For lCol = 1 To UBound(vR, 2)
    If obj.Count > 0 Then
      lDrawn = sbRandHistogrm(0#, UBound(obj.Items), obj.Items)
      vR(lRow, lCol) = obj.Keys()(lDrawn)
      If vNames(1) <> "" Then
        'Need to add in again a name
        obj.Add vNames(1), lCount(1)
      End If
      For j = 1 To lN - 1
        vNames(j) = vNames(j + 1)
        lCount(j) = lCount(j + 1)
      Next j
      If obj.Items()(lDrawn) > 1 Then
        vNames(lN) = obj.Keys()(lDrawn)
        lCount(lN) = obj.Items()(lDrawn) - 1
      Else
        vNames(lN) = ""
        'lCount(lN) = 0 'Not necessary but clean
      End If
      obj.Remove obj.Keys()(lDrawn)
      i = i + 1
    Else
      vR(lRow, lCol) = "Error: Cannot fulfil gap condition!"
    End If
  Next lCol
Next lRow
sbRandomNoRepeatBeforeN = vR
End With
End Function
```

6 Brownsche Brücken

Eine brownsche Brücke ist eine brownsche (zufällige) Bewegung mit bestimmtem Start- und Endwert. Sie wird zur Modellierung von zufälligen Entwicklungen von Zeitreihen verwendet, deren Wert zu zwei Zeitpunkten bekannt ist.

Neben den hier vorgestellten Beispielen kann auch `sbRandIntFixSum` als brownsche Brücke angesehen werden.

6.1 `sbGrowthSeries`

Sie können Zufallszahlen mit einer kumulierten Wachstumsrate dblRate, mit einer maximalen relativen Änderungsrate pro Zeitschritt dblMaxRatePerStep und mit einem optionalen Startwert dblStartVal erzeugen. Die Anzahl der Zeitschritte (Perioden) wird implizit durch die Anzahl der ausgewählten Zellen gewählt, in die der Funktionsaufruf als Matrixformel mit STRG + SHIFT + ENTER eingegeben wird. Dies ist eine spezielle Art von Brownscher Brücke.

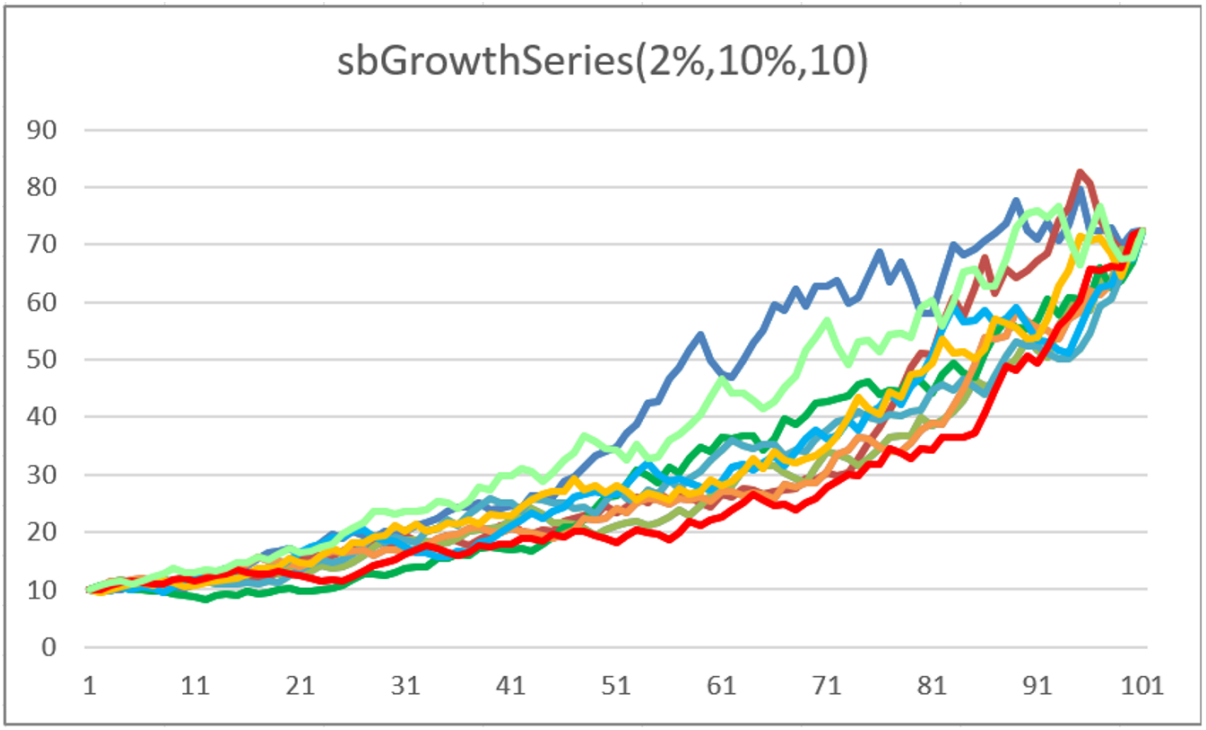


Figure 34: sbGrowthSeries

Programmcode `sbGrowthSeries`

```
Function sbGrowthSeries(dblRate As Double, _
        dblMaxRatePerStep As Double, _
        Optional dblStartVal As Double = 1#) As Variant
'Returns random data with a compound growth rate dblRate, with
'a maximal relative change rate per step of dblMaxRatePerStep
'and with a start value dblStartVal. The number of periods
'is implicitly chosen by the number of selected cells which
'call this function as an array formula (entered with
'CTRL + SHIFT + ENTER). This is sort of a brownian bridge.
'(C) (P) by Bernd Plumhoff 20-Mar-2011 PB V0.91
```

```
Dim vR As Variant
Dim lP As Long 'Periods
Dim lrow As Long
Dim lcol As Long
Dim dblCurrVal As Double
Dim dblCurrRate As Double
Dim dblCurrMin As Double
Dim dblCurrMax As Double
Dim dblRelMin As Double
Dim dblRelMax As Double
Dim dblEndVal As Double

If TypeName(Application.Caller) <> "Range" Then
  sbGrowthSeries = CVErr(xlErrRef)
  Exit Function
End If

If Application.Caller.Rows.Count <> 1 And _
  Application.Caller.Columns.Count <> 1 Then
  sbGrowthSeries = CVErr(xlErrValue)
  Exit Function
End If

If Abs(dblRate) > dblMaxRatePerStep Then
  sbGrowthSeries = CVErr(xlErrNum)
  Exit Function
End If

lP = Application.Caller.Count

ReDim vR(1 To Application.Caller.Rows.Count, _
    1 To Application.Caller.Columns.Count)

dblCurrVal = dblStartVal
dblEndVal = dblStartVal * (1# + dblRate) ^ CDbl(lP)
dblCurrMin = dblEndVal / (1# + dblMaxRatePerStep) ^ CDbl(lP)
dblCurrMax = dblEndVal / (1# - dblMaxRatePerStep) ^ CDbl(lP)
For lrow = 1 To UBound(vR, 1)
  For lcol = 1 To UBound(vR, 2)
    dblCurrRate = (dblEndVal / dblCurrVal) ^ (1# / _
        CDbl(lP - lcol * lrow + 1)) - 1#
    dblCurrMin = dblCurrMin * (1# + dblMaxRatePerStep)
    dblCurrMax = dblCurrMax * (1# - dblMaxRatePerStep)
    dblRelMin = (dblCurrMin - dblCurrVal) / dblCurrVal
    If dblRelMin < -dblMaxRatePerStep Then
      dblRelMin = -dblMaxRatePerStep
    End If
    dblRelMax = (dblCurrMax - dblCurrVal) / dblCurrVal
    If dblRelMax > dblMaxRatePerStep Then
      dblRelMax = dblMaxRatePerStep
```

```
    End If
    If dblCurrRate - dblRelMin < dblRelMax - dblCurrRate Then
      dblRelMax = 2# * dblCurrRate - dblRelMin
    Else
      dblRelMin = 2# * dblCurrRate - dblRelMax
    End If
    dblCurrVal = dblCurrVal * (1# + (dblRelMin + dblRelMax) / 2# + _
        (Rnd() - 0.5) * (dblRelMax - dblRelMin))
    vR(lrow, lcol) = dblCurrVal
  Next lcol
Next lrow

sbGrowthSeries = vR

End Function
```

6.2 Fixe Summe aus verschiedenen Zufallsbereichen

Sie benötigen sieben Zufallszahlen aus verschiedenen Zahlenbereichen, die zusammen genau 100 ergeben?

	A	B	C	D	E	F	G	H	I	J	K
1	**Zielwert**	100								**Summe**	**Check**
2	**Untergrenze**	0,27	2,8	15,3	43,3	15,1	9,8	2,7		89,27	
3	**Obergrenze**	0,44	4,9	17,4	48,5	18,2	13,5	5,8		108,74	
4	**Formellösungen:**										
5		0,40206022	3,51356284	16,5085186	44,8899007	17,3533623	11,5549935	5,77760189		100	
6		0,39120156	3,44220507	17,1827317	47,9229809	16,7717078	10,9405165	3,34865645		100	
7		0,40951144	3,79485159	15,8092308	47,7845392	15,2303171	11,6556437	5,31590611		100	
8		0,4239384	4,15342804	16,0116953	45,3050432	15,6557401	12,8155512	5,63460382		100	
9		0,3845167	3,64892537	15,6595322	43,5092385	18,00071	13,0754472	5,72162988		100	
10		0,30599978	3,54247014	17,0732974	47,3858742	17,8938342	10,0380864	3,76043783		100	
11		0,41335292	3,60723858	15,8304168	46,851858	18,1192622	11,5286128	3,64925862		100	
12		0,28130997	3,51554573	17,3374028	45,6987532	16,5240956	13,4954458	3,14744687		100	
13		0,28909348	3,73813934	16,4378585	47,4888238	17,6815508	10,2536789	4,11085527		100	
14		0,30455272	3,96973418	16,3162646	46,7935995	15,2630869	11,7514806	5,60128144		100	

Figure 35: Fixe Summe aus verschiedenen Zufallsbereichen

Tabellenblattformeln		
Bereich	Formel	
J2:J3;J5:J14	**J2**	=SUMME(B2:H2)
K2	**K2**	=WENN(J2>$B$1;"Keine Lösung, weil Summe der Untergrenzen größer als " & $B$1 & " ist.";"")
K3	**K3**	=WENN(J3<$B$1;"Keine Lösung, weil Summe der Obergrenzen kleiner als " & $B$1 & " ist.";"")
B5:H14	**B5**	=MAX(B$2;$B$1-SUMME($A5:A5)-SUMME(C$3:$I$3))+ZUFALLSZAHL()*(MIN(B$3;$B$1-SUMME($A5:A5)-SUMME(C$2:$I$2))-MAX(B$2;$B$1-SUMME($A5:A5)-SUMME(C$3:$I$3)))

Figure 36: Fixe Summe aus verschiedenen Zufallsbereichen - Formeln

Wichtiger Hinweis: Es existiert keine Lösung, wenn die Summe der Untergrenzen größer als 100 ist oder wenn die Summe der Obergrenzen kleiner ist als 100! Dies wird in den Zellen K2:K3 geprüft.

Mögliche Sortierreihenfolgen der Korridorbreiten

6.2.1 Ursprüngliche Sortierreihenfolger der Korridorbreiten

Die erzeugten Zufallszahlen im obigen Beispiel sind relativ gleichverteilt. Bei $1,048,572$ erzeugten Reihen von jeweils 7 Zahlen erhält man für die originale Sortierung der Grenzwert-Korridore:

	A	B	C	D	E	F	G	H	I	J	K
1	Zielwert	100								Summe	Check
2	Untergrenze	0,27	2,8	15,3	43,3	15,1	9,8	2,7		89,27	
3	Obergrenze	0,44	4,9	17,4	48,5	18,2	13,5	5,8		108,74	
4	Formellösungen:										
5		0,40206022	3,51356284	16,5085186	44,8899007	17,3533623	11,5549935	5,77760189		100	
6		0,39120156	3,44220507	17,1827317	47,9229809	16,7717078	10,9405165	3,34865645		100	
7		0,40951144	3,79485159	15,8092308	47,7845392	15,2303171	11,6556437	5,31590611		100	
8		0,4239384	4,15342804	16,0116953	45,3050432	15,6557401	12,8155512	5,63460382		100	
9		0,3845167	3,64892537	15,6595322	43,5092385	18,00071	13,0754472	5,72162988		100	
10		0,30599978	3,54247014	17,0732974	47,3858742	17,8938342	10,0380864	3,76043783		100	
11		0,41335292	3,60723858	15,8304168	46,851858	18,1192622	11,5286128	3,64925862		100	
12		0,28130997	3,51554573	17,3374028	45,6987532	16,5240956	13,4954458	3,14744687		100	
13		0,28909348	3,73813934	16,4378585	47,4888238	17,6815508	10,2536789	4,11085527		100	
14		0,30455272	3,96973418	16,3162646	46,7935995	15,2630869	11,7514806	5,60128144		100	

Figure 37: Fixe Summe aus verschiedenen Zufallsbereichen - Originale Korridorsortierung

6.2.2 Absteigende Reihenfolge der Korridorbreiten

Bei absteigender Sortierung der Grenzwerte nach Korridorbreite erhält man:

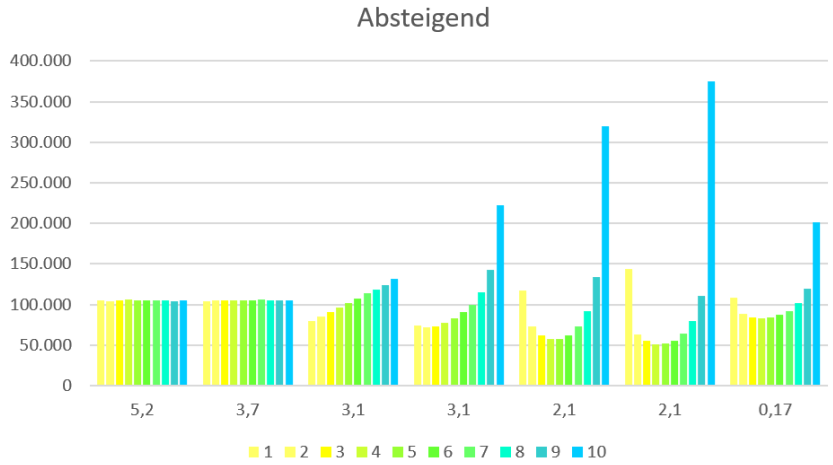


Figure 38: Fixe Summe aus verschiedenen Zufallsbereichen - Absteigende Korridorbreite

6.2.3 Aufsteigende Reihenfolge der Korridorbreiten

Bei aufsteigender Sortierung nach Korridorbreite:

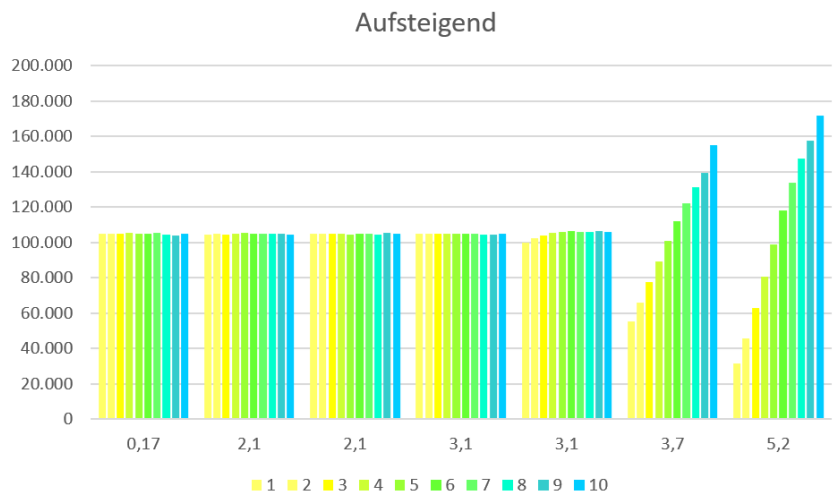


Figure 39: Fixe Summe aus verschiedenen Zufallsbereichen - Aufsteigende Korridorbreite

Um möglichst gleichverteilte Zufallszahlen zu erhalten, sollte man deshalb die Spalten nach aufsteigenden Korridorbreiten sortieren, weil die erzeugenden Formeln die Freiheitsgrade von links nach rechts reduzieren. Falls Sie - aus welchen Gründen auch immer - absteigend sortierte Grenz¬wertbereiche vorliegen haben, müssen Sie mit deutlich extremeren Verteilungen rechnen.

6.2.4 Verwendung einer Triang Verteilung

With the triangular distribution `sbRandTriang` you get with 10,000 generated rows:

The corresponding formula in cell B5: `=sbRandTriang (MAX(B$2,$B$1-SUM($A5:A5) - SUM(C$3:$I$3)),MIN(MAX(MAX(B$2,$B$1 - SUM($A5:A5) - SUM(C$3:$I$3)),B$2+ ($B$1 - (SUM($A5:A5) + SUM(B$2:$I$2))) / (SUM(B$3:$I$3) - SUM(B$2:$I$2)) * (B$3-B$2)),MIN(B$3,$B$1 - SUM($A5:A5) - SUM(C$2:$I$2))),MIN(B$3,$B$1 - SUM($A5:A5) - SUM(C$2:$I$2))).`

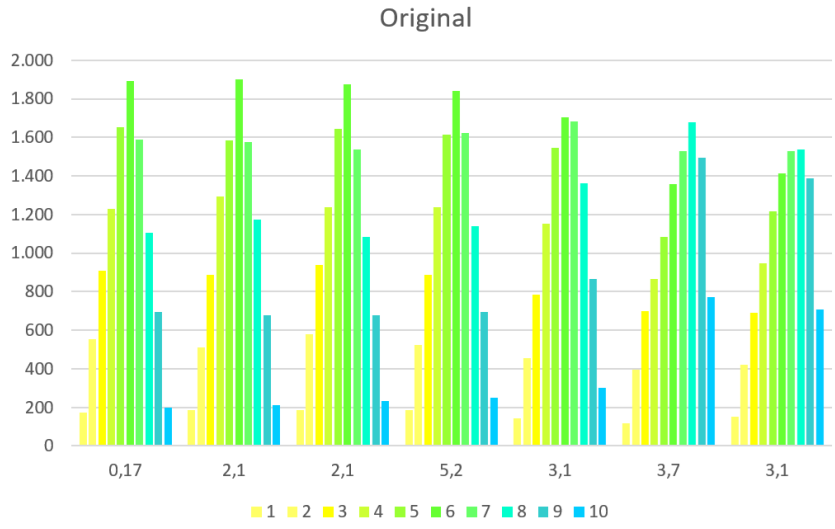


Figure 40: Fixe Summe aus Zufallsbereichen - sbRandTriang mit Originalkorridoren

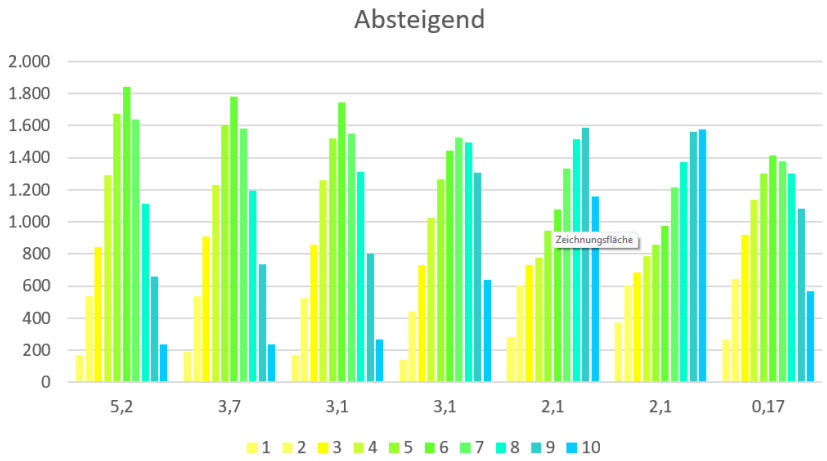


Figure 41: Fixe Summe aus Zufallsbereichen - sbRandTriang mit absteigender Breite

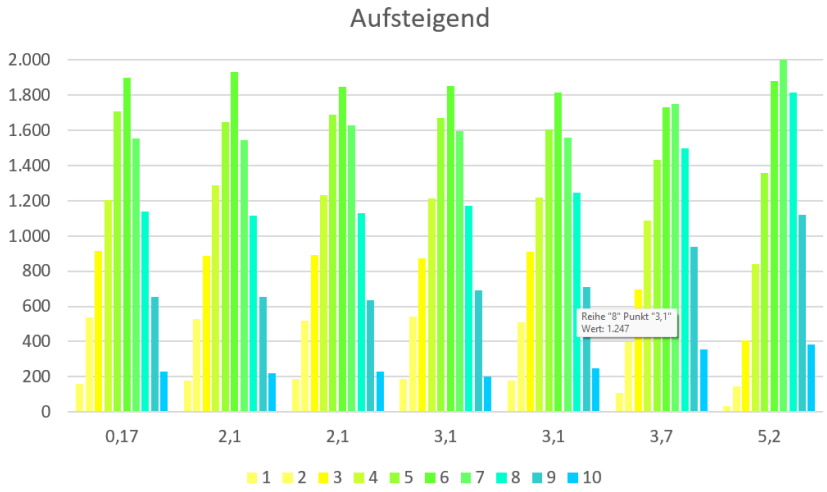


Figure 42: Fixe Summe aus Zufallsbereichen - sbRandTriang mit aufsteigender Breite

6.2.5 Gerundete Ergebnisse

Müssen die Ergebnisse auf eine bestimmte Anzahl von Nachkommastellen gerundet werden, dann kann die oben genannte allgemeine Formel z. B. für 2 Nachkommastellen in =RUNDEN(...; 2) eingebettet werden.

Wichtig ist nur, dass mindestens auf die maximale Anzahl der in den Grenzwerten verwendeten Nach¬kommastellen gerundet wird, damit

- die Ergebnisse nach Rundung noch im Korridorbereich sind
- nicht Teile des Korridorbereiches durch die Rundung unerreichbar werden
- das Zielergebnis immer erreicht wird.

7 Korrelierte Zufallszahlen

7.1 Cholesky Zerlegung

Mit der Cholesky (sprich: 'Koleski') Zerlegung können Sie korrelierte Zufallszahlen erzeugen:

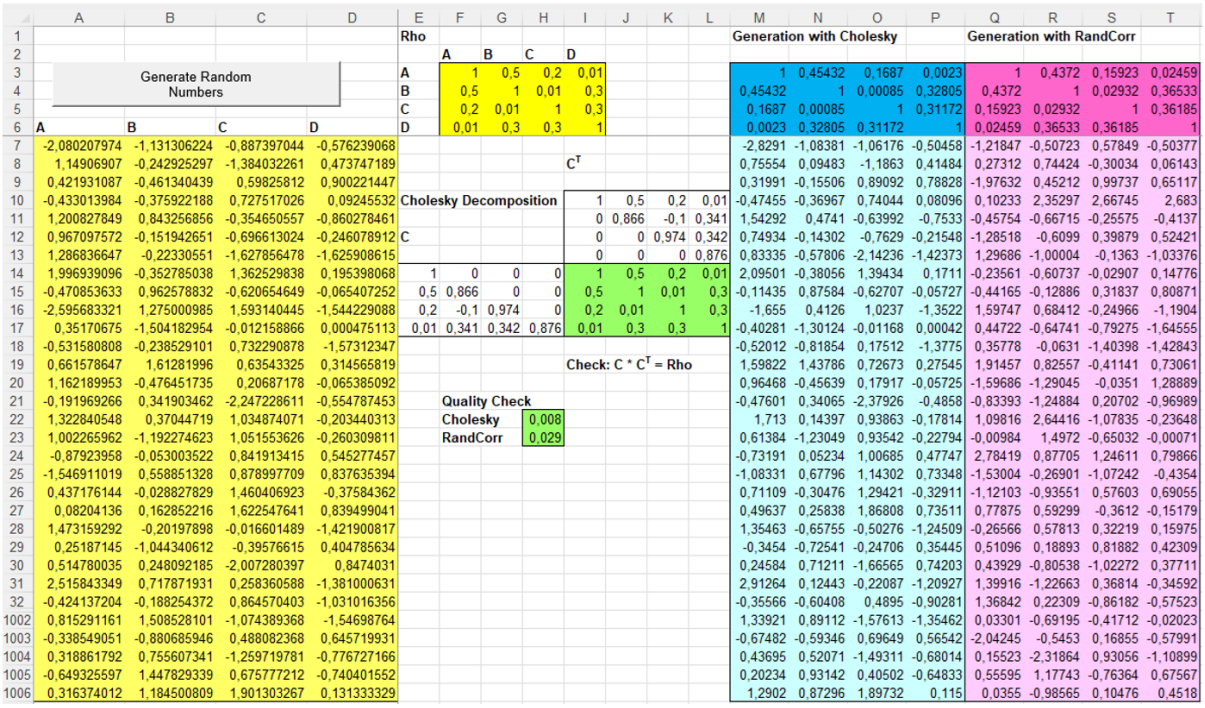


Figure 43: Cholesky und RandCorr

Tabellenblattformeln		
Bereich	Formel	
E14	**E14**	=Cholesky(F3:I6)
I10	**I10**	=MTRANS(E14:H17)
I14	**I14**	=MMULT(E14:H17;I10:L13)
Cholesky_Check	**H22**	=SUMME((F3:I6-M3:P6)^2)
RandCorr_Check	**H23**	=SUMME((F3:I6-Q3:T6)^2)
M3:P6	**M3**	=KORREL(INDEX($M$7:$P$1005;;ZEILE()-2);INDEX($M$7:$P$1005;;SPALTE()-12))
M7	**M7**	=MMULT(A7:D1006;E14:H17)
Q3:T6	**Q3**	=KORREL(INDEX($Q$7:$T$1005;;ZEILE()-2);INDEX($Q$7:$T$1005;;SPALTE()-16))
Q7	**Q7**	=RandCorr(1000;Rho)

Figure 44: Cholesky und RandCorr- Formeln

Programmcode `Cholesky`

```
Function Cholesky (vA As Variant) As Variant
'I suggest to use the Cholesky decomposition just for purposes of
'demonstration. Better options are (in this order): tred2, tqli, eigsrt
'from Numerical Recipes. SVD also works but is computationally
'more expensive by far since it does not make use of symmetry.
'(Thanks to my former colleague Glen R.)
'Bernd Plumhoff 02-Nov-2024 PB V1.1
Dim d As Double
Dim i As Long, j As Long, k As Long, n As Long
With Application.WorksheetFunction
On Error Resume Next
vA = .Transpose(.Transpose(vA))
On Error GoTo 0
n = UBound(vA, 1)
If n <> UBound(vA, 2) Then
  Cholesky = CVErr(xlErrRef)
  Exit Function
End If

ReDim dR(1 To n, 1 To n) As Double 'Zeroing all elements
For j = 1 To n
  d = 0#
  For k = 1 To j - 1
    d = d + dR(j, k) * dR(j, k)
  Next k
  dR(j, j) = vA(j, j) - d
  If dR(j, j) > 0# Then
    dR(j, j) = Sqr(dR(j, j))
    For i = j + 1 To n
      d = 0#
      For k = 1 To j - 1
        d = d + dR(i, k) * dR(j, k)
      Next k
      dR(i, j) = (vA(i, j) - d) / dR(j, j)
    Next i
  Else
    'Cannot continue with usual Cholesky
    'Fill this column with zeros. Idea: Glen R.
    For i = j To n
      dR(i, j) = 0#
    Next i
  End If
Next j
Cholesky = dR
End With
End Function
```

Programmcode `RandCorr`

```
Function RandCorr(n As Long, vVarCovar As Variant) As Variant
'Returns Ubound(vVarCovar,1) correlated random number vectors of
'length n. vVarCovar is a square matrix containing the variance /
'covariance matrix. Please notice that you will only get a "proxy"
'correlation, not an exact one.
'Bernd Plumhoff 06-Nov-2009 PB V0.2
Dim vA As Variant
Dim d As Double
Dim i As Long, j As Long, k As Long, m As Long

With Application.WorksheetFunction
vA = .Transpose(.Transpose(vVarCovar))
m = UBound(vA, 1)
If m <> UBound(vA, 2) Then
  RandCorr = CVErr(xlErrRef)
  Exit Function
End If
ReDim Db(1 To m, 1 To m) As Double
For j = 1 To m
  d = 0#
  For k = 1 To j - 1
    d = d + Db(j, k) * Db(j, k)
  Next k
  Db(j, j) = vA(j, j) - d
  If Db(j, j) <= 0 Then
    RandCorr = CVErr(xlErrNum)
    Exit Function
  End If
  Db(j, j) = Sqr(Db(j, j))

  For i = j + 1 To m
    d = 0#
    For k = 1 To j - 1
      d = d + Db(i, k) * Db(j, k)
    Next k
    Db(i, j) = (vA(i, j) - d) / Db(j, j)
  Next i
Next j

ReDim vR(1 To n, 1 To m) As Variant
For i = 1 To n
  For j = 1 To m
    vR(i, j) = .Norm_S_Inv(Rnd())
  Next j
Next i
vR = .MMult(vR, Db)
RandCorr = vR
End With
End Function
```

7.2 Iman-Conover Methode

Falls Sie korrelierte Zufallszahlen erzeugen müssen, ist die Iman-Conover Methode besser als die Cholesky Zerlegung.

1982 veröffentlichten Iman und Conover ihren ursprünglichen Artikel
`"A distribution-free approach to inducing rank correlation among input variables"`

Rick Wicklin schrieb dazu im Jahr 2021 "Simulate correlated variables by using the Iman-Conover transformation". Sein Artikel enthält eine SAS Implementierung der Iman Conover Methode:
`"Simulate correlated variables by using the Iman-Conover transformation"`

2005 veröffentlichte Stephen J. Mildenhall "Correlation and Aggregate Loss Distributions with an Emphasis on the Iman-Conover Method": `"Correlation and Aggregate Loss Distributions with an Emphasis on the Iman-Conover Method"`

Ich implementierte das Beispiel aus Mildenhall's Artikel sowohl mit Excel Tabellenblattfunktionen als auch mit Excel / VBA:

Die Eingabematrix X:

	A	B	C	D
1	123.567	44.770	15.934	13.273
2	126.109	45.191	16.839	15.406
3	138.713	47.453	17.233	16.706
4	139.016	47.941	17.265	16.891
5	152.213	49.345	17.620	18.821
6	153.224	49.420	17.859	19.569
7	153.407	50.686	20.804	20.166
8	155.716	52.931	21.110	20.796
9	155.780	54.010	22.728	20.968
10	161.678	57.346	24.072	21.178
11	161.805	57.685	25.198	23.236
12	167.447	57.698	25.393	23.375
13	170.737	58.380	30.357	24.019
14	171.592	60.948	30.779	24.785
15	178.881	66.972	32.634	25.000
16	181.678	68.053	33.117	26.754
17	184.381	70.592	35.248	27.079
18	206.940	72.243	36.656	30.136
19	217.092	86.685	38.483	30.757
20	240.935	87.138	39.483	35.108

Input_Matrix_X | Target_Correlation_S | Cholesky_C | Inter

Figure 45: Iman-Conover Methode - Eingabematrix X

Die Zielkorrelation S:

	A	B	C	D
1	1	0,8	0,4	0
2	0,8	1	0,3	-0,2
3	0,4	0,3	1	0,1
4	0	-0,2	0,1	1

... Target_Correlation_S | Cholesky_C | Intermediate_M | Covariance

Tabellenblattformeln		
Bereich	Formel	
A2;A3:B3;A4:C4	**A2**	=INDEX($A$1:$D$4;SPALTE();ZEILE())

Figure 46: Iman-Conover Methode - Zielkorrelation S

Die Cholesky Zerlegung C von S:

	A	B	C	D
1	1,0000	0,8000	0,4000	0,0000
2	0,0000	0,6000	-0,0333	-0,3333
3	0,0000	0,0000	0,9159	0,0970
4	0,0000	0,0000	0,0000	0,9378

... Cholesky_C | Intermediate_M | Covariance_E | Cholesky_F

Tabellenblattformeln		
Bereich	Formel	
A1	**A1**	=MTRANS(Cholesky(Target_Correlation_S!A1:D4))

Figure 47: Iman-Conover Methode - Cholesky Zerlegung C von S

Die Zwischenmatrix M (konstante Werte, identisch zu Mildenhall's Daten):

	A	B	C	D
1	-1,92062	1,22896	-1,00860	-0,49584
2	-1,50709	-1,50709	-1,50709	0,82015
3	-1,22896	1,92062	0,82015	-0,65151
4	-1,00860	-0,20723	1,00860	-1,00860
5	-0,82015	0,82015	0,34878	1,92062
6	-0,65151	-1,22896	-0,65151	0,20723
7	-0,49584	-0,65151	1,22896	-0,34878
8	-0,34878	-0,49584	-0,49584	-0,06874
9	-0,20723	-1,00860	0,20723	0,65151
10	-0,06874	0,49584	0,06874	-1,22896
11	0,06874	-0,34878	-1,22896	0,49584
12	0,20723	0,34878	0,65151	0,34878
13	0,34878	-0,06874	-0,20723	1,22896
14	0,49584	-1,92062	-0,82015	-0,20723
15	0,65151	0,20723	1,92062	-1,92062
16	0,82015	1,00860	1,50709	1,50709
17	1,00860	-0,82015	-1,92062	1,00860
18	1,22896	1,50709	0,49584	-1,50709
19	1,50709	0,06874	-0,06874	0,06874
20	1,92062	0,65151	-0,34878	-0,82015

... Intermediate_M | Covariance_E | Cholesky_F | Intermediat

Tabellenblattformeln		
Bereich	Formel	
I1	**I1**	=NORM.S.INV(SEQUENZ(20;1;1;1)/21)/STABWNA(NORM.S.INV(SEQUENZ(20;1;1;1)/21))
J1:L1	**J1**	=MTRANS(randomshuffle($A$1:$A$20))

Figure 48: Iman-Conover Methode - Zwischenmatrix M

Sie können analoge Daten automatisch mit dieser Formel in $A1 : A20$ erzeugen:
$= NORM.S.INV(SEQUENZ(20; 1; 1; 1)/21)/$
$STABWNA(NORM.S.INV(SEQUENZ(20; 1; 1; 1)/21))$
und mit dieser Formel
$= MTRANS(RandomShuffle(\$A\$1 : \$A\$20))$
in den Zellen $B1 : B20$ (kopieren Sie diese entsprechend in die Spalten C und D).

Nun erhalten Sie die Kovarianzmatrix E:

	A	B	C	D
1	1,0000	0,0486	0,0898	-0,0960
2	0,0486	1,0000	0,4504	-0,2408
3	0,0898	0,4504	1,0000	-0,3192
4	-0,0960	-0,2408	-0,3192	1,0000

... Covariance_E | Cholesky_F | Intermediate_T | Check_Cc ...

Tabellenblattformeln		
Bereich	Formel	
A1:D4	**A1**	=KOVAR(INDEX(Intermediate_M!$A$1:$D$20;;ZEILE());INDEX(Intermediate_M!$A$1:$D$20;;SPALTE()))

Figure 49: Iman-Conover Methode - Kovarianzmatrix E

Und ihre Cholesky Zerlegung F:

	A	B	C	D
1	1,0000	0,0486	0,0898	-0,0960
2	0,0000	0,9988	0,4466	-0,2364
3	0,0000	0,0000	0,8902	-0,2303
4	0,0000	0,0000	0,0000	0,9391

... Cholesky_F | Intermediate_T | Check_Correlation | Rank_T

Tabellenblattformeln		
Bereich	Formel	
A1	**A1**	=MTRANS(Cholesky(Covariance_E!A1:D4))

Figure 50: Iman-Conover Methode - Cholesky Zerlegung F von E

Die Zwischenmatrix T:

	A	B	C	D
1	-1,92062	-0,74214	-2,28105	-1,33232
2	-1,50709	-2,06697	-1,30678	0,54577
3	-1,22896	0,20647	-0,51141	-0,94465
4	-1,0086	-0,9019	0,80546	-0,65873
5	-0,82015	-0,13949	-0,31782	1,7696
6	-0,65151	-1,24042	-0,28	0,23988
7	-0,49584	-0,77356	1,42145	0,23612
8	-0,34878	-0,56669	-0,38117	-0,14744
9	-0,20723	-0,76561	0,64214	0,97494
10	-0,06874	0,24487	-0,19673	-1,33695
11	0,06874	-0,15653	-1,06954	0,14014
12	0,20723	0,36925	0,56695	0,51206
13	0,34878	0,22754	-0,06362	1,1955
14	0,49584	-0,77155	0,26828	0,03168
15	0,65151	0,62666	2,08987	-1,21744
16	0,82015	1,23804	1,32493	1,8568
17	1,0086	0,28474	-1,23688	0,59246
18	1,22896	1,85259	0,17411	-1,62428
19	1,50709	1,20294	0,39517	0,13931
20	1,92062	1,87176	-0,04335	-0,97245

... | Cholesky_F | **Intermediate_T** | Check_Correlation | Rank_T

Tabellenblattformeln		
Bereich	Formel	
A1	**A1**	=MMULT(MMULT(Intermediate_M!A1:D20;MINV(Cholesky_F!A1:D4));Cholesky_C!A1:D4)

Figure 51: Iman-Conover Methode - Zwischenmatrix T

Sie können die erzeugten Korrelationen prüfen:

	A	B	C	D
1	1,0000	0,8000	0,4000	0,0000
2	0,8000	1,0000	0,3000	-0,2000
3	0,4000	0,3000	1,0000	0,1000
4	0,0000	-0,2000	0,1000	1,0000
5				
6	**Difference to Target Correlation:**			
7				
8	0	0	0	-4,44089E-17
9	0	0	0	0
10	0	0	0	0
11	-4,44089E-17	0	0	0

... | Cholesky_F | Intermediate_T | **Check_Correlation** | Rank_T | Result_Y | Check_Correlatic

Tabellenblattformeln		
Bereich	Formel	
A1:D4	**A1**	=KORREL(INDEX(Intermediate_T!$A$1:$D$20;;ZEILE());INDEX(Intermediate_T!$A$1:$D$20;;SPALTE()))
A8	**A8**	=A1:D4-Target_Correlation_S!A1:D4

Figure 52: Iman-Conover Methode - Erzeugte Korrelationen

Berechnen Sie den Rang der Zahlen in den Spalten von T in Tabellenblatt Rank_T:

	A	B	C	D
1	1	7	1	3
2	2	1	2	15
3	3	11	5	6
4	4	3	17	7
5	5	10	7	19
6	6	2	8	13
7	7	4	19	12
8	8	8	6	8
9	9	6	16	17
10	10	13	9	2
11	11	9	4	11
12	12	15	15	14
13	13	12	10	18
14	14	5	13	9
15	15	16	20	4
16	16	18	18	20
17	17	14	3	16
18	18	19	12	1
19	19	17	14	10
20	20	20	11	5

Tabellenblattformeln		
Bereich	Formel	
A1:D1	**A1**	=RANG(Intermediate_T!A1:A20;Intermediate_T!A1:A20;1)

Figure 53: Iman-Conover Methode - Rang der Zahlen in den Spalten von T

Schließlich erhalten Sie im Tabellenblatt Result_Y:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	123.567	50.686	15.934	16.706		VBA	123.567	44.770	17.265	24.785		Worksheet formulae	123.567	50.686	15.934	16.706
2	126.109	44.770	16.839	25.000			126.109	45.191	33.117	24.019			126.109	44.770	16.839	25.000
3	138.713	57.685	17.620	19.569			138.713	50.686	17.233	13.273			138.713	57.685	17.620	19.569
4	139.016	47.453	35.248	20.166			139.016	57.685	20.804	30.136			139.016	47.453	35.248	20.166
5	152.213	57.346	20.804	30.757			152.213	57.346	30.357	21.178			152.213	57.346	20.804	30.757
6	153.224	45.191	21.110	24.019			153.224	49.420	16.839	20.968			153.224	45.191	21.110	24.019
7	153.407	47.941	38.483	23.375			153.407	47.941	15.934	26.754			153.407	47.941	38.483	23.375
8	155.716	52.931	17.859	20.796			155.716	47.453	22.728	19.569			155.716	52.931	17.859	20.796
9	155.780	49.420	33.117	27.079			155.780	52.931	25.393	35.108			155.780	49.420	33.117	27.079
10	161.678	58.380	22.728	15.406			161.678	49.345	25.198	23.236			161.678	58.380	22.728	15.406
11	161.805	54.010	17.265	23.236			161.805	86.685	36.656	15.406			161.805	54.010	17.265	23.236
12	167.447	66.972	32.634	24.785			167.447	66.972	30.779	16.706			167.447	66.972	32.634	24.785
13	170.737	57.698	24.072	30.136			170.737	54.010	35.248	20.796			170.737	57.698	24.072	30.136
14	171.592	49.345	30.357	20.968			171.592	68.053	17.859	18.821			171.592	49.345	30.357	20.968
15	178.881	68.053	39.483	16.891			178.881	57.698	38.483	27.079			178.881	68.053	39.483	16.891
16	181.678	72.243	36.656	35.108			181.678	70.592	17.620	16.891			181.678	72.243	36.656	35.108
17	184.381	60.948	17.233	26.754			184.381	58.380	24.072	20.166			184.381	60.948	17.233	26.754
18	206.940	86.685	25.393	13.273			206.940	72.243	32.634	30.757			206.940	86.685	25.393	13.273
19	217.092	70.592	30.779	21.178			217.092	60.948	39.483	23.375			217.092	70.592	30.779	21.178
20	240.935	87.138	25.198	18.821			240.935	87.138	21.110	25.000			240.935	87.138	25.198	18.821

Tabellenblattformeln		
Bereich	Formel	
A1:D1;M1:P1	**A1**	=INDEX(Input_Matrix_X!$A$1:$A$20;Rank_T!$A$1:$A$20)
G1	**G1**	=ImanConover(Input_Matrix_X!$A$1:$D$20;Target_Correlation_S!$A$1:$D$4)

Figure 54: Iman-Conover Methode - Ergebnis Y

Sie können die Differenz zur Zielkorrelation im Tabellenblatt Check_Correlation_Result prüfen:

	A	B	C	D	E	F
1	1,00	0,85	0,26	-0,11		
2	0,85	1,00	0,19	-0,20		
3	0,26	0,19	1,00	0,10		
4	-0,11	-0,20	0,10	1,00		
5						
6	**Difference to Target Correlation:**					**Maximal absolute error:**
7						
8	0	0,049673836	-0,13590681	-0,11360723		0,135906814
9	0,049673836	0	-0,11151318	0,000215604		
10	-0,13590681	-0,11151318	0	-0,00429182		
11	-0,11360723	0,000215604	-0,00429182	0		

Tabellenblattformeln		
Bereich	Formel	
A1:D4	**A1**	=KORREL(INDEX(Result_Y!$A$1:$D$20;;ZEILE());INDEX(Result_Y!$A$1:$D$20;;SPALTE()))
A8	**A8**	=A1:D4-Target_Correlation_S!A1:D4
F8	**F8**	=MAX(ABS(A8:D11))

Figure 55: Iman-Conover Methode - Differenz zwischen Ergebnis und Zielkorrelation

Programmcode `RandomShuffle`

```
Function RandomShuffle (vtemp As Variant) As Variant
Dim j As Long, k As Long, n As Long
Dim temp As Double, u As Double
'Application.Volatile 'Uncomment if you think you need this.
With Application.WorksheetFunction
On Error Resume Next 'Ignore error: VBA calls already with 1-dim array.
vtemp = .Transpose(vtemp)
On Error GoTo 0
n = UBound(vtemp)
j = n
Do While j > 0
   u = Rnd()
   k = Int(j * u + 1)
   temp = vtemp(j)
   vtemp(j) = vtemp(k)
   vtemp(k) = temp
   j = j - 1
Loop
RandomShuffle = vtemp
End With
End Function
```

Programmcode `IndexX`

```
Function IndexX (n As Long, arr As Variant, colNo As Long) As Variant
'Indexes an array arr[1..n], i.e., outputs the array indx[1..n] such
'that arr[indx[j]] is in ascending order for j = 1, 2, .. ,n. The
'input quantities n and arr are not changed.
Const m As Long = 7
Const NSTACK As Long = 50
Dim i As Long, indxt As Long, ir As Long, itemp As Long
Dim j As Long, k As Long, l As Long
Dim jstack As Long, istack(1 To NSTACK) As Long, a As Double

ir = n: l = 1
ReDim indx(1 To n) As Long
For j = 1 To n: indx(j) = j: Next j

Do While 1
  If (ir - l < m) Then
    For j = l + 1 To ir
      indxt = indx(j)
      a = arr(indxt, colNo)
      For i = j - 1 To l Step -1
        If (arr(indx(i), colNo) <= a) Then Exit For
        indx(i + 1) = indx(i)
      Next i
      indx(i + 1) = indxt
    Next j
    If (jstack = 0) Then Exit Do
    ir = istack(jstack)
    jstack = jstack - 1
    l = istack(jstack)
    jstack = jstack - 1
  Else
    k = (l + ir) / 2
    itemp = indx(k)
    indx(k) = indx(l + 1)
    indx(l + 1) = itemp
    If (arr(indx(l), colNo) > arr(indx(ir), colNo)) Then
      itemp = indx(l)
      indx(l) = indx(ir)
      indx(ir) = itemp
    End If
    If (arr(indx(l + 1), colNo) > arr(indx(ir), colNo)) Then
      itemp = indx(l + 1)
      indx(l + 1) = indx(ir)
      indx(ir) = itemp
    End If
    If (arr(indx(l), colNo) > arr(indx(l + 1), colNo)) Then
      itemp = indx(l)
      indx(l) = indx(l + 1)
      indx(l + 1) = itemp
```

```
      End If
      i = l + 1
      j = ir
      indxt = indx(l + 1)
      a = arr(indxt, colNo)
      Do While 1
        Do
          i = i + 1
        Loop While (arr(indx(i), colNo) < a)
        Do
          j = j - 1
        Loop While (arr(indx(j), colNo) > a)
        If (j < i) Then Exit Do
        itemp = indx(i)
        indx(i) = indx(j)
        indx(j) = itemp
      Loop
      indx(l + 1) = indx(j)
      indx(j) = indxt
      jstack = jstack + 2
      If (jstack > NSTACK) Then
        'STACK too small in indexx
        IndexX = CVErr(xlErrNum)
        Exit Function
      End If
      If (ir - i + 1 >= j - l) Then
        istack(jstack) = ir
        istack(jstack - 1) = i
        ir = j - 1
      Else
        istack(jstack) = j - 1
        istack(jstack - 1) = l
        l = i
      End If
    End If
  Loop
  IndexX = indx
End Function
```

Programmcode `ImanConover`

This is the VBA implementation of the Iman-Conover method. I use this code in `sbGenerateTestData`, too.

Please notice that the function `ImanConover` uses (calls) the user-defined functions `IndexX` and `RandomShuffle` mentioned above as well as the function `Cholesky`.

```
Function ImanConover(rInputMatrix As Range, _
        rTargetCorrelation As Range) As Variant
'Implements the Iman-Conover method to generate random
'number vectors with a given correlation.
'Algorithm as described in:
'Mildenham, November 27, 2005
'Correlation and Aggregate Loss Distributions With An
'Emphasis On The Iman-Conover Method
'V0.3 PB 02-Nov-2024 by Bernd Plumhoff
Dim vX As Variant    'Input matrix
Dim vS As Variant    'Target correlation matrix
Dim vC As Variant    'Cholesky decomposition of vS
Dim vM As Variant    'Intermediate matrix M
Dim vE As Variant    'Covariance matrix E
Dim vF As Variant    'Cholesky decomposition of vE
Dim vT As Variant    'Intermediate matrix T
Dim d As Double, dS As Double
Dim i As Long, j As Long, k As Long
Dim lRow As Long, lCol As Long
Dim state As SystemState

With Application
Set state = New SystemState
vX = .Transpose(.Transpose(rInputMatrix))
lRow = rInputMatrix.Rows.Count
lCol = rInputMatrix.Columns.Count

'#########################################################################
'#                          Check inputs                                 #
'#########################################################################

If lCol <> rTargetCorrelation.Columns.Count _
  And rTargetCorrelation.Rows.Count _
    <> rTargetCorrelation.Columns.Count Then
   'Structure of target correlation matrix needs to fit input matrix
   ImanConover = CVErr(xlErrNum)
   Exit Function
End If
vS = .Transpose(.Transpose(rTargetCorrelation))
For i = 1 To lCol
   If vS(i, i) <> 1# Then
      'Target correlation matrix not 1 on diagonal
      ImanConover = CVErr(xlErrValue)
      Exit Function
   End If
```

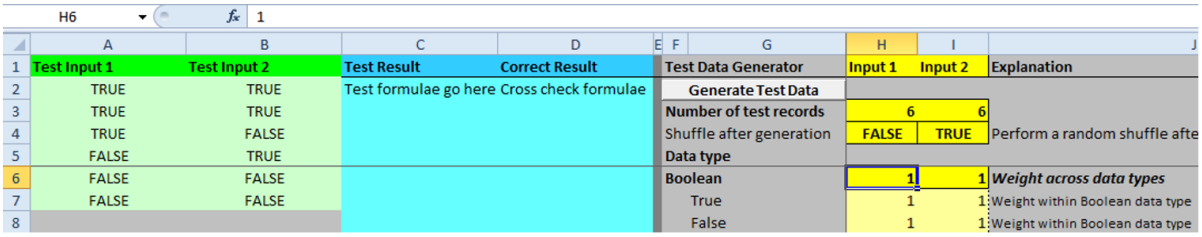
```
    For j = 1 To i - 1
      If vS(i, j) <> vS(j, i) Then
        'Target correlation matrix not symmetric
        ImanConover = CVErr(xlErrValue)
        Exit Function
      End If
    Next j
  Next i

  vC = .Transpose(Cholesky(vS))

  '#########################################################################
  '#                        Create intermediate matrix M                   #
  '#########################################################################
  ReDim vMV(1 To lRow) As Double
  d = 0#
  dS = 0#
  For i = 1 To Int(lRow / 2)
    vMV(i) = .NormSInv(i / (lRow + 1))
    vMV(lRow - i + 1) = -vMV(i)
    d = d + 2# * vMV(i) * vMV(i)
  Next i
  If lRow Mod 2 = 1 Then
    vMV((lRow + 1) / 2) = 0 'Just for clarity, it's already 0
  End If
  d = Sqr(d / lRow)
  For i = 1 To lRow
    vMV(i) = vMV(i) / d
  Next i

  vM = vX
  For i = 1 To lRow
    vM(i, 1) = vMV(i)
  Next i

  Dim vMW As Variant
  For i = 2 To lCol
    vMW = RandomShuffle(vMV)
    For j = 1 To lRow
      vM(j, i) = vMW(j)
    Next j
  Next i

  '#########################################################################
  '#                        Calculate covariance matrix E                  #
  '#########################################################################

  vE = vC
  For i = 1 To lCol
    vE(i, i) = .Covar(.Index(.Transpose(vM), i), _
      .Index(.Transpose(vM), i))
```

```
    For j = i + 1 To lCol
      vE(i, j) = .Covar(.Index(.Transpose(vM), i), _
        .Index(.Transpose(vM), j))
      vE(j, i) = vE(i, j)
    Next j
  Next i

  vF = .Transpose(Cholesky(vE))

  vT = .MMult(.MMult(vM, .MInverse(vF)), vC)

  '#########################################################################
  '#                        Compute ranks of matrix T                      #
  '#########################################################################

  Dim vRT As Variant, vR As Variant
  vRT = vX
  For j = 1 To lCol
    vR = IndexX(lRow, vT, j)
    For i = 1 To lRow
      vRT(i, j) = vR(i)
    Next i
    vR = IndexX(lRow, vX, j)
    For i = 1 To lRow
      vX(i, j) = vX(vR(i), j)
    Next i
  Next j

  '#########################################################################
  '#                        Calculate result matrix Y                      #
  '#########################################################################

  Dim vY As Variant
  vY = vX
  For i = 1 To lRow
    For j = 1 To lCol
      vY(i, j) = vX(vRT(i, j), j)
    Next j
  Next i

  ImanConover = vY
  End With
  End Function
```

8 Testdaten erzeugen – `sbGenerateTestData`

Wenn Sie eine Anwendung oder ein Programm auf Herz und Nieren testen wollen, benötigen Sie häufig Testdaten. Die Anwendung `sbGenerateTestData` soll Sie dabei unterstützen, zufällige Testdaten in numerischer Form oder als Text zu erzeugen.

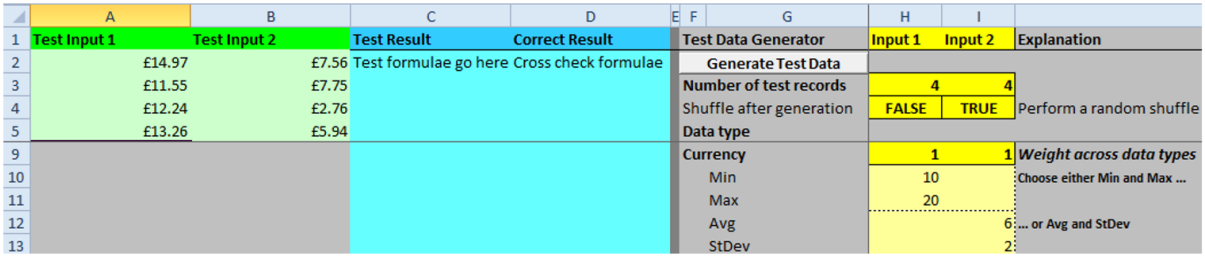
Wollen Sie beispielsweise sechs Wahrheitswerte, davon 50% WAHR und 50% FALSCH, einmal in der erzeugten Reihenfolge und einmal zufällig gemischt:



	A	B	C	D	F	G	H	I	J
1	Test Input 1	Test Input 2	Test Result	Correct Result	Test Data Generator		Input 1	Input 2	Explanation
2	TRUE	TRUE	Test formulae go here	Cross check formulae	Generate Test Data				
3	TRUE	TRUE			Number of test records		6	6	
4	TRUE	FALSE			Shuffle after generation		FALSE	TRUE	Perform a random shuffle afte
5	FALSE	TRUE			Data type				
6	FALSE	FALSE			Boolean		1	1	*Weight across data types*
7	FALSE	FALSE				True	1	1	Weight within Boolean data type
8						False	1	1	Weight within Boolean data type

Figure 56: sbGenerateTestData - 6 Wahrheitswerte

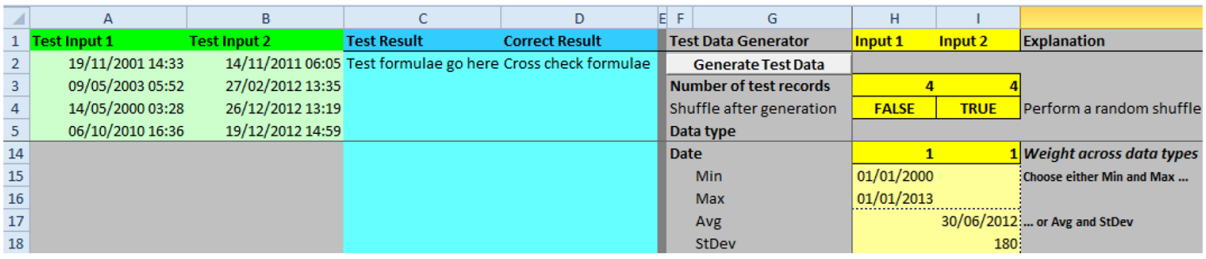
Oder Sie benötigen 4 Geldbeträge in britischen Pfund Sterling (GBP), die erste Serie zwischen 10 GBP und 20 GBP und die zweite mit einem Durchschnittswert von 6 GBP und einer Standardabweichung von 2 GBP:



	A	B	C	D	F	G	H	I	
1	Test Input 1	Test Input 2	Test Result	Correct Result	Test Data Generator		Input 1	Input 2	Explanation
2	£14.97	£7.56	Test formulae go here	Cross check formulae	Generate Test Data				
3	£11.55	£7.75			Number of test records		4	4	
4	£12.24	£2.76			Shuffle after generation		FALSE	TRUE	Perform a random shuffle
5	£13.26	£5.94			Data type				
9					Currency		1	1	*Weight across data types*
10						Min	10		Choose either Min and Max ...
11						Max	20		
12						Avg		6	... or Avg and StDev
13						StDev		2	

Figure 57: sbGenerateTestData - 4 GBP Werte

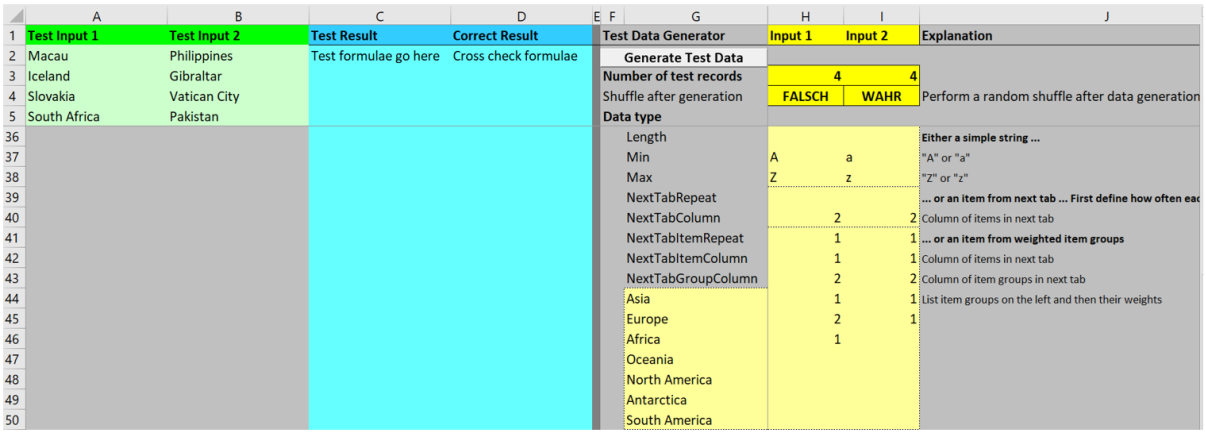
Falls Sie vier Daten zwischen $1 - Jan - 2000$ und $1 - Jan - 2013$ oder vier Daten mit dem Durschschnittswert $30 - Jun - 2012$ und einer Standardabweichung von 180 Tagen benötigen:



	A	B	C	D	F	G	H	I	
1	Test Input 1	Test Input 2	Test Result	Correct Result	Test Data Generator		Input 1	Input 2	Explanation
2	19/11/2001 14:33	14/11/2011 06:05	Test formulae go here	Cross check formulae	Generate Test Data				
3	09/05/2003 05:52	27/02/2012 13:35			Number of test records		4	4	
4	14/05/2000 03:28	26/12/2012 13:19			Shuffle after generation		FALSE	TRUE	Perform a random shuffle
5	06/10/2010 16:36	19/12/2012 14:59			Data type				
14					Date		1	1	*Weight across data types*
15						Min	01/01/2000		Choose either Min and Max ...
16						Max	01/01/2013		
17						Avg		30/06/2012	... or Avg and StDev
18						StDev		180	

Figure 58: sbGenerateTestData - 4 Datumswerte

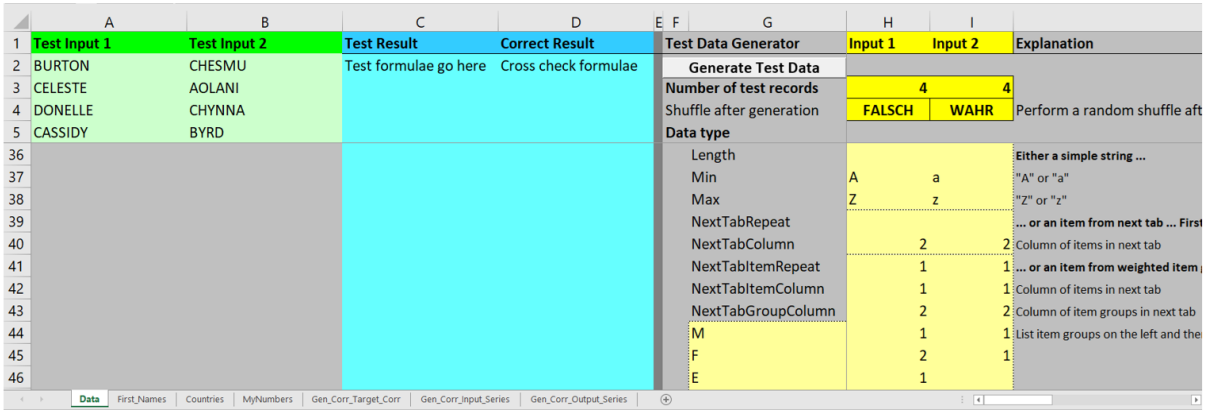
Wenn Sie 4 Ländernamen erzeugen wollen, davon einen aus Afrika, einen aus Asien, und zwei aus Europa; oder Sie brauchen 2 asiatische und 2 europäische Ländernamen (ziehen Sie das Tabellenblatt 'Countries' gleich rechts neben das Tabellenblatt 'Data' so dass es Blatt 2 ist:



	A	B	C	D	G	H	I	J
1	**Test Input 1**	**Test Input 2**	**Test Result**	**Correct Result**	**Test Data Generator**	**Input 1**	**Input 2**	**Explanation**
2	Macau	Philippines	Test formulae go here	Cross check formulae	**Generate Test Data**			
3	Iceland	Gibraltar			**Number of test records**	4	4	
4	Slovakia	Vatican City			Shuffle after generation	FALSCH	WAHR	Perform a random shuffle after data generation
5	South Africa	Pakistan			**Data type**			
36					Length			**Either a simple string ...**
37					Min	A	a	"A" or "a"
38					Max	Z	z	"Z" or "z"
39					NextTabRepeat			**... or an item from next tab ... First define how often eac**
40					NextTabColumn	2	2	Column of items in next tab
41					NextTabItemRepeat	1	1	**... or an item from weighted item groups**
42					NextTabItemColumn	1	1	Column of items in next tab
43					NextTabGroupColumn	2	2	Column of item groups in next tab
44					Asia	1	1	List item groups on the left and then their weights
45					Europe	2	1	
46					Africa	1		
47					Oceania			
48					North America			
49					Antarctica			
50					South America			

Figure 59: sbGenerateTestData - 4 Ländernamen

Falls Sie Vornamen zufällig aus einer gegebenen Liste ziehen wollen, ziehen SIe das Tabellenblatt 'First_Names' rechts neben das 'Data' Blatt. Sie erhalten nach erneutem Drücken des Knopfes 'Generate Test Data' eine Warnung. Drücken Sie dann 'Ok':



	A	B	C	D	G	H	I	
1	**Test Input 1**	**Test Input 2**	**Test Result**	**Correct Result**	**Test Data Generator**	**Input 1**	**Input 2**	**Explanation**
2	BURTON	CHESMU	Test formulae go here	Cross check formulae	**Generate Test Data**			
3	CELESTE	AOLANI			**Number of test records**	4	4	
4	DONELLE	CHYNNA			Shuffle after generation	FALSCH	WAHR	Perform a random shuffle aft
5	CASSIDY	BYRD			**Data type**			
36					Length			**Either a simple string ...**
37					Min	A	a	"A" or "a"
38					Max	Z	z	"Z" or "z"
39					NextTabRepeat			**... or an item from next tab ... First**
40					NextTabColumn	2	2	Column of items in next tab
41					NextTabItemRepeat	1	1	**... or an item from weighted item**
42					NextTabItemColumn	1	1	Column of items in next tab
43					NextTabGroupColumn	2	2	Column of item groups in next tab
44					M	1	1	List item groups on the left and the
45					F	2	1	
46					E	1		

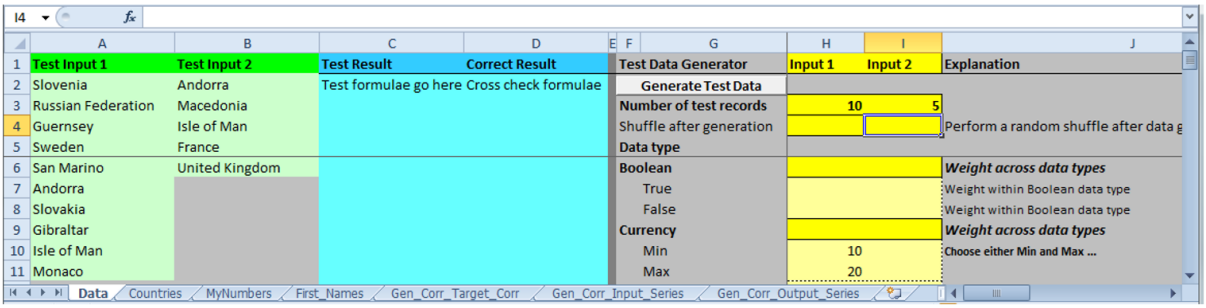
Data | First_Names | Countries | MyNumbers | Gen_Corr_Target_Corr | Gen_Corr_Input_Series | Gen_Corr_Output_Series

Figure 60: sbGenerateTestData - 4 Vornamen

Bemerkung: Die Spalten für die Listenelemente und deren Gruppen sind hier nicht zufällig identisch. Dies wurde so absichtlich gestaltet, damit man durch Ziehen des entsprechenden Tabellenblatts neben das 'Data' Blatt die gewünschten Werte ändern kann, entweder zu Vornamen oder zu Ländernamen.

Sie können mit dieser Anwendung auch korrelierte Pseudozufallszahlen erzeugen. Ich implementierte die Iman-Conover Methode mit VBA.

Die Tabellenblätter:



	A	B	C	D	E	F	G	H	I	J
1	**Test Input 1**	**Test Input 2**	**Test Result**	**Correct Result**			**Test Data Generator**	**Input 1**	**Input 2**	**Explanation**
2	Slovenia	Andorra	Test formulae go here	Cross check formulae			**Generate Test Data**			
3	Russian Federation	Macedonia					**Number of test records**	10	5	
4	Guernsey	Isle of Man					Shuffle after generation			Perform a random shuffle after data g
5	Sweden	France					**Data type**			
6	San Marino	United Kingdom					**Boolean**			***Weight across data types***
7	Andorra						True			Weight within Boolean data type
8	Slovakia						False			Weight within Boolean data type
9	Gibraltar						**Currency**			***Weight across data types***
10	Isle of Man						Min	10		**Choose either Min and Max ...**
11	Monaco						Max	20		

Data | Countries | MyNumbers | First_Names | Gen_Corr_Target_Corr | Gen_Corr_Input_Series | Gen_Corr_Output_Series

Figure 61: sbGenerateTestData - Verwendete Tabellenblätter

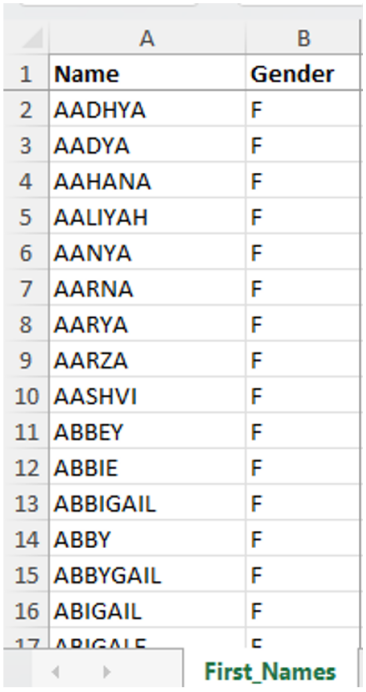
Tabellenblatt `Countries`:

	A	B
1	**Country Name**	**Continent Name**
2	Afghanistan	Asia
3	Åland	Europe
4	Albania	Europe
5	Algeria	Africa
6	American Samoa	Oceania
7	Andorra	Europe
8	Angola	Africa
9	Anguilla	North America
10	Antarctica	Antarctica
11	Antigua and Barbuda	North America
12	Argentina	South America
13	Armenia	Asia
14	Aruba	North America
15	Australia	Oceania
16	Austria	Europe
17	Azerbaijan	Asia

Data | **Countries** | First_Names | Gen_Corr_

Figure 62: sbGenerateTestData - Tabellenblatt Countries

Tabellenblatt First_Name, female names male names:

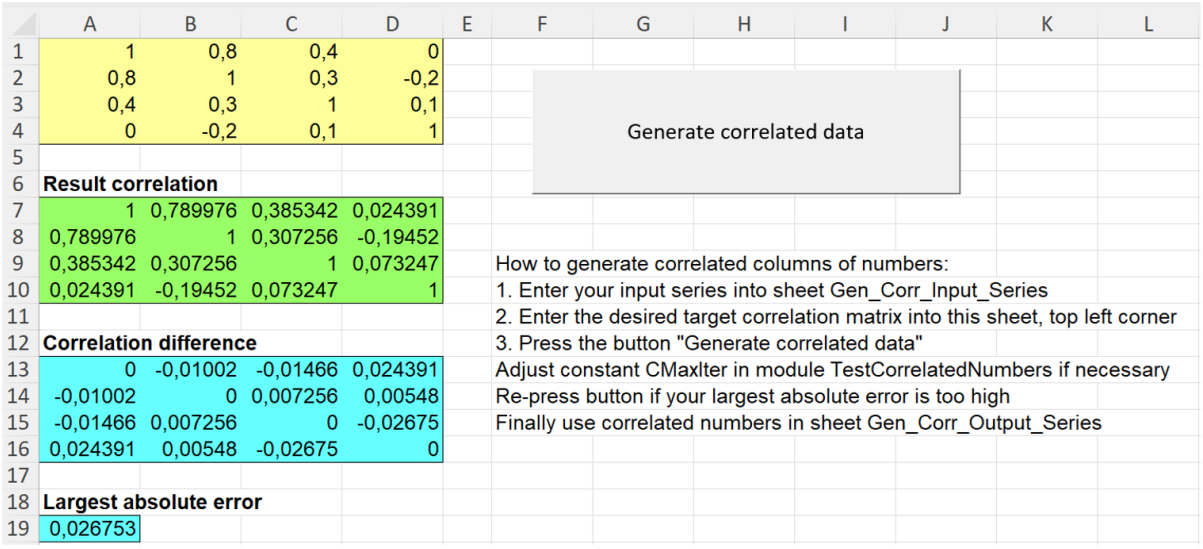


	A	B
1	**Name**	**Gender**
2	AADHYA	F
3	AADYA	F
4	AAHANA	F
5	AALIYAH	F
6	AANYA	F
7	AARNA	F
8	AARYA	F
9	AARZA	F
10	AASHVI	F
11	ABBEY	F
12	ABBIE	F
13	ABBIGAIL	F
14	ABBY	F
15	ABBYGAIL	F
16	ABIGAIL	F

First_Names

Figure 63: sbGenerateTestData - Tabellenblatt First Names

Korrelierte Zufallszahlen erzeugen Sie mit dieser Anwendung wie folgt:



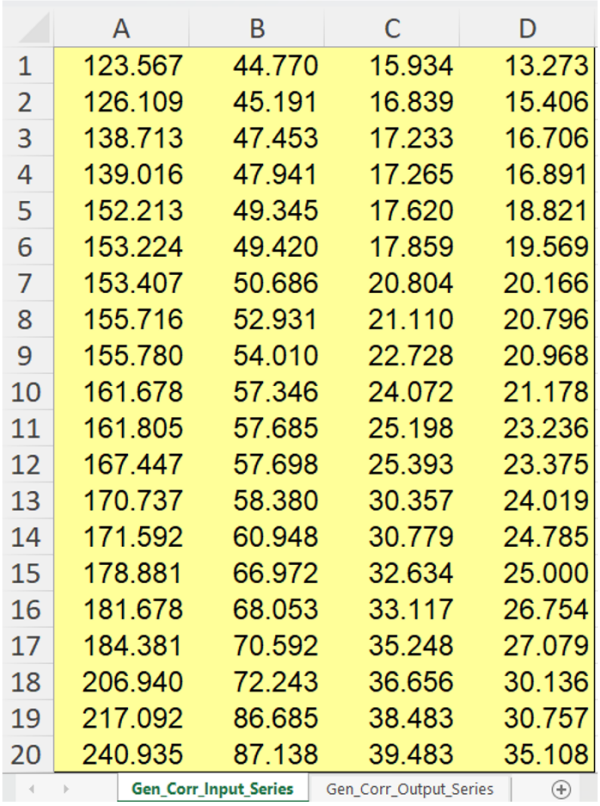
	A	B	C	D
1	1	0,8	0,4	0
2	0,8	1	0,3	-0,2
3	0,4	0,3	1	0,1
4	0	-0,2	0,1	1
5				
6	**Result correlation**			
7	1	0,789976	0,385342	0,024391
8	0,789976	1	0,307256	-0,19452
9	0,385342	0,307256	1	0,073247
10	0,024391	-0,19452	0,073247	1
11				
12	**Correlation difference**			
13	0	-0,01002	-0,01466	0,024391
14	-0,01002	0	0,007256	0,00548
15	-0,01466	0,007256	0	-0,02675
16	0,024391	0,00548	-0,02675	0
17				
18	**Largest absolute error**			
19	0,026753			

Generate correlated data

How to generate correlated columns of numbers:
1. Enter your input series into sheet Gen_Corr_Input_Series
2. Enter the desired target correlation matrix into this sheet, top left corner
3. Press the button "Generate correlated data"
Adjust constant CMaxIter in module TestCorrelatedNumbers if necessary
Re-press button if your largest absolute error is too high
Finally use correlated numbers in sheet Gen_Corr_Output_Series

Worksheet Formulas		
Range	Formula	
A7:D10	**A7**	=CORREL(INDEX(Gen_Corr_Output_Series!$A$1:$D$20,,ROW()-6),INDEX(Gen_Corr_Output_Series!$A$1:$D$20,,COLUMN()))
A13	**A13**	=A7 - A1
A19	**A19**	=MAX(ABS(A13:D16))

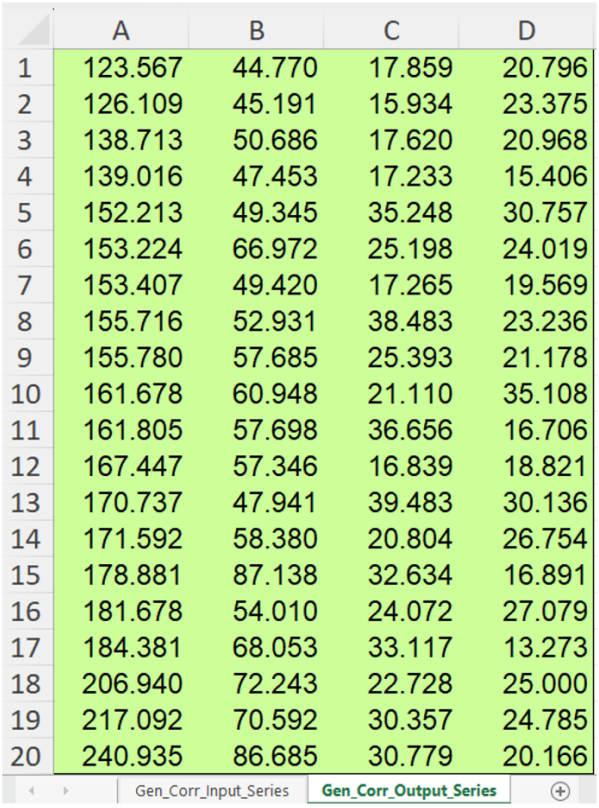
Figure 64: sbGenerateTestData - Eingabe für korrelierte Zufallszahlen



	A	B	C	D
1	123.567	44.770	15.934	13.273
2	126.109	45.191	16.839	15.406
3	138.713	47.453	17.233	16.706
4	139.016	47.941	17.265	16.891
5	152.213	49.345	17.620	18.821
6	153.224	49.420	17.859	19.569
7	153.407	50.686	20.804	20.166
8	155.716	52.931	21.110	20.796
9	155.780	54.010	22.728	20.968
10	161.678	57.346	24.072	21.178
11	161.805	57.685	25.198	23.236
12	167.447	57.698	25.393	23.375
13	170.737	58.380	30.357	24.019
14	171.592	60.948	30.779	24.785
15	178.881	66.972	32.634	25.000
16	181.678	68.053	33.117	26.754
17	184.381	70.592	35.248	27.079
18	206.940	72.243	36.656	30.136
19	217.092	86.685	38.483	30.757
20	240.935	87.138	39.483	35.108

Gen_Corr_Input_Series | Gen_Corr_Output_Series

Figure 65: sbGenerateTestData -Input Series for correlated number generation



	A	B	C	D
1	123.567	44.770	17.859	20.796
2	126.109	45.191	15.934	23.375
3	138.713	50.686	17.620	20.968
4	139.016	47.453	17.233	15.406
5	152.213	49.345	35.248	30.757
6	153.224	66.972	25.198	24.019
7	153.407	49.420	17.265	19.569
8	155.716	52.931	38.483	23.236
9	155.780	57.685	25.393	21.178
10	161.678	60.948	21.110	35.108
11	161.805	57.698	36.656	16.706
12	167.447	57.346	16.839	18.821
13	170.737	47.941	39.483	30.136
14	171.592	58.380	20.804	26.754
15	178.881	87.138	32.634	16.891
16	181.678	54.010	24.072	27.079
17	184.381	68.053	33.117	13.273
18	206.940	72.243	22.728	25.000
19	217.092	70.592	30.357	24.785
20	240.935	86.685	30.779	20.166

Gen_Corr_Input_Series | Gen_Corr_Output_Series

Figure 66: sbGenerateTestData - Output Series of correlated number generation

Programmcode `sbGenerateTestData`

Dieses Programm benötigt (verwendet) die Klasse SystemState.

```
Enum types
   ty_start = 0 'So that we can iterate from ty_start + 1 to ty_end - 1
   ty_boolean
   ty_currency
   ty_date
   ty_decimal
   ty_double
   ty_long
   ty_string
   ty_end   'So that we can iterate from ty_start + 1 to ty_end - 1
End Enum 'types

Enum param_rows
   pr_records = 3
   pr_shuffle
   pr_Boolean = 6
      pr_bTrue
      pr_bFalse
   pr_Currency
      pr_ccyMin
      pr_ccyMax
      pr_ccyAvg
      pr_ccyStDev
   pr_Date
      pr_dtMin
      pr_dtMax
      pr_dtAvg
      pr_dtStDev
   pr_Decimal
      pr_decMin
      pr_decMax
      pr_decAvg
      pr_decStDev
   pr_Double
      pr_dMin
      pr_dMax
      pr_dAvg
      pr_dStDev
   pr_Long
      pr_lSum
      pr_lMin1
      pr_lMin2
      pr_lMax
      pr_lMaxRepeat
   pr_String
      pr_sLength
      pr_sMin
      pr_sMax
```

```
    pr_sNextTabRepeat
    pr_sNextTabColumn
    pr_sNextTabItemRepeat
    pr_sNextTabItemColumn
    pr_sNextTabGroupColumn
    pr_sNextTabGroupWeights 'Item group weights start
    'from here and can go down any number
End Enum 'param_rows

Enum param_columns
  pc_Output1 = 1
  pc_Output2
  pc_ItemGroups = 7
  pc_Input1 = 8
  pc_Input2
End Enum 'param_columns

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

Sub sbGenerateTestData()
'Randomly generate test data as specified in input area.
'Bernd Plumhoff 06-Apr-2021 PB V0.2

Dim bGroupsUpToDate As Boolean
Dim dAvg As Double
Dim dmax As Double
Dim dmin As Double
Dim dStDev As Double
Dim dSumWeights As Double
ReDim dTypeWeight(ty_start + 1 To ty_end - 1) As Double
ReDim sTypeName(ty_start + 1 To ty_end - 1) As String
Dim i As Long
```

```
Dim j As Long
Dim k As Long
Dim lCol As Long
Dim lLength As Long
Dim lRecord As Long
Dim lRow As Long
Dim lIdx As Long
Dim lTypeSum As Long
Dim objItem As Object
Dim objGroup As Object
Dim s As String
Dim sErrMsg As String
Dim v As Variant
Dim vThisType As Variant
Dim vType As Variant
Dim vGroup As Variant
Dim wsItem As Worksheet
Dim state As SystemState

Set state = New SystemState
Randomize

With Application.WorksheetFunction

'Clear input
wsD.Range("A:A").Offset(, pc_Output1 - 1).ClearContents
wsD.Range("A:A").Offset(, pc_Output2 - 1).ClearContents
wsD.Range("A:A").Offset(, pc_Output1 - 1).ClearFormats
wsD.Range("A:A").Offset(, pc_Output2 - 1).ClearFormats
wsD.Range("A:A").Offset(, pc_Output1 - 1).Interior.ColorIndex = xlCIGray25
wsD.Range("A:A").Offset(, pc_Output2 - 1).Interior.ColorIndex = xlCIGray25
With wsD.Range("A1").Offset(, pc_Output1 - 1)
   .Formula = "Test-Input-1"
   .Font.Bold = True
   .Interior.ColorIndex = xlCIBrightGreen
End With
With wsD.Range("A1").Offset(, pc_Output2 - 1)
   .Formula = "Test-Input-2"
   .Font.Bold = True
   .Interior.ColorIndex = xlCIBrightGreen
End With

sTypeName(ty_boolean) = "Boolean"
sTypeName(ty_currency) = "Currency"
sTypeName(ty_date) = "Date"
sTypeName(ty_decimal) = "Decimal"
sTypeName(ty_double) = "Double"
sTypeName(ty_long) = "Long"
sTypeName(ty_string) = "String"

For lCol = pc_Input1 To pc_Input2
```

```
sErrMsg = ""
lRecord = wsD.Cells(pr_records, lCol)
If lRecord <= 0 Then
  Call MsgBox("Number of test records must be greater zero!" & _
    vbCrLf, vbOKOnly, "Error")
  Exit Sub
End If
wsD.Cells(2, lCol - pc_Input1 + _
  pc_Output1).Resize(lRecord).Interior.ColorIndex = xlCILightGreen
ReDim vInput(1 To lRecord) As Variant
lIdx = 1
dTypeWeight(ty_boolean) = wsD.Cells(pr_Boolean, lCol)
dTypeWeight(ty_currency) = wsD.Cells(pr_Currency, lCol)
dTypeWeight(ty_date) = wsD.Cells(pr_Date, lCol)
dTypeWeight(ty_decimal) = wsD.Cells(pr_Decimal, lCol)
dTypeWeight(ty_double) = wsD.Cells(pr_Double, lCol)
dTypeWeight(ty_long) = wsD.Cells(pr_Long, lCol)
dTypeWeight(ty_string) = wsD.Cells(pr_String, lCol)
dSumWeights = 0#
For i = LBound(dTypeWeight) To UBound(dTypeWeight)
  If dTypeWeight(i) < 0 Then sErrMsg = sErrMsg & _
    "Weight for data type " & sTypeName(i) & _
      " must be greater equal zero!" & vbCrLf
  dSumWeights = dSumWeights + dTypeWeight(i)
Next i
If dSumWeights <= 0 Then sErrMsg = sErrMsg & _
  "Sum of weights for data types (Boolean, ..., String) " & _
  "must be greater zero!" & vbCrLf

If Len(sErrMsg) > 0 Then
  Call MsgBox(sErrMsg & vbCrLf, vbOKOnly, "Error")
  Exit Sub
End If
For i = LBound(dTypeWeight) To UBound(dTypeWeight)
  dTypeWeight(i) = dTypeWeight(i) / dSumWeights * lRecord
Next i
'Decide how many records to generate for each data type
vType = RoundToSum(dTypeWeight, 0)

For i = LBound(vType, 1) To UBound(vType, 1)
  If vType(i) > 0 Then
    Select Case i
    Case ty_boolean
      ReDim dThisTypeWeight(1 To 2) As Double
      If Abs(wsD.Cells(pr_bTrue, lCol) + wsD.Cells(pr_bFalse, lCol)) _
        < 0.0000000000001 Then
        'No weights means equal weights
        dThisTypeWeight(1) = vType(i) / 2
        dThisTypeWeight(2) = dThisTypeWeight(1)
      Else
        dThisTypeWeight(1) = wsD.Cells(pr_bTrue, lCol) / _
```

```
                                (wsD.Cells(pr_bTrue, lCol) + _
                                wsD.Cells(pr_bFalse, lCol)) * _
                                vType(i)
        dThisTypeWeight(2) = wsD.Cells(pr_bFalse, lCol) / _
                                (wsD.Cells(pr_bFalse, lCol) + _
                                wsD.Cells(pr_bTrue, lCol)) * _
                                vType(i)
    End If
    vThisType = RoundToSum(dThisTypeWeight, 0)
    For j = 1 To vThisType(1)
      vInput(lIdx) = True
      lIdx = lIdx + 1
    Next j
    For j = 1 To vThisType(2)
      vInput(lIdx) = False
      lIdx = lIdx + 1
    Next j
  Case ty_currency
    If IsEmpty(wsD.Cells(pr_ccyAvg, lCol)) Or _
      IsEmpty(wsD.Cells(pr_ccyStDev, lCol)) Then
      'Work with Min and Max
      dmin = wsD.Cells(pr_ccyMin, lCol)
      dmax = wsD.Cells(pr_ccyMax, lCol)
      For j = 1 To vType(i)
        vInput(lIdx) = CCur(dmin + Rnd() * (dmax - dmin))
        lIdx = lIdx + 1
      Next j
    Else
      'Work with Avg and StDev
      ReDim dThisDouble(1 To vType(i)) As Double
      For j = 1 To vType(i)
        dThisDouble(j) = Rnd()
      Next j
      dAvg = .Average(dThisDouble)
      dStDev = .StDevP(dThisDouble)
      If dStDev < 0.0000000000001 Then
        If vType(i) = 1 Then
          vInput(lIdx) = CCur(dAvg)
          lIdx = lIdx + 1
        Else
          Call MsgBox("StDev of data type " & _
              sTypeName(ty_currency) & _
              " must not be zero!", vbOKOnly, "Error!")
          Exit Sub
        End If
      End If
      For j = 1 To vType(i)
        vInput(lIdx) = CCur(wsD.Cells(pr_ccyAvg, lCol) + _
                        (dThisDouble(j) - dAvg) * _
                        wsD.Cells(pr_ccyStDev, lCol) / dStDev)
        lIdx = lIdx + 1
```

```
      Next j
    End If
  Case ty_date
    If IsEmpty(wsD.Cells(pr_dtAvg, lCol)) Or _
      IsEmpty(wsD.Cells(pr_dtStDev, lCol)) Then
      'Work with Min and Max
      dmin = wsD.Cells(pr_dtMin, lCol)
      dmax = wsD.Cells(pr_dtMax, lCol)
      For j = 1 To vType(i)
        vInput(lIdx) = CDate(dmin + Rnd() * (dmax - dmin))
        lIdx = lIdx + 1
      Next j
    Else
      'Work with Avg and StDev
      ReDim dThisDouble(1 To vType(i)) As Double
      For j = 1 To vType(i)
        dThisDouble(j) = Rnd()
      Next j
      dAvg = .Average(dThisDouble)
      dStDev = .StDevP(dThisDouble)
      If dStDev < 0.0000000000001 Then
        If vType(i) = 1 Then
          vInput(lIdx) = CDate(dAvg)
          lIdx = lIdx + 1
        Else
          Call MsgBox("StDev of data type " & _
               sTypeName(ty_date) & _
               " must not be zero!", vbOKOnly, "Error!")
          Exit Sub
        End If
      End If
      For j = 1 To vType(i)
        vInput(lIdx) = CDate(wsD.Cells(pr_dtAvg, lCol) + _
                       (dThisDouble(j) - dAvg) * _
                       wsD.Cells(pr_dtStDev, lCol) / dStDev)
        lIdx = lIdx + 1
      Next j
    End If
  Case ty_decimal
    If IsEmpty(wsD.Cells(pr_decAvg, lCol)) Or _
      IsEmpty(wsD.Cells(pr_decStDev, lCol)) Then
      'Work with Min and Max
      dmin = wsD.Cells(pr_decMin, lCol)
      dmax = wsD.Cells(pr_decMax, lCol)
      For j = 1 To vType(i)
        vInput(lIdx) = CDec(dmin + Rnd() * (dmax - dmin))
        lIdx = lIdx + 1
      Next j
    Else
      'Work with Avg and StDev
      ReDim dThisDouble(1 To vType(i)) As Double
```

```
      For j = 1 To vType(i)
        dThisDouble(j) = Rnd()
      Next j
      dAvg = .Average(dThisDouble)
      dStDev = .StDevP(dThisDouble)
      If dStDev < 0.0000000000001 Then
        If vType(i) = 1 Then
          vInput(lIdx) = CDec(dAvg)
          lIdx = lIdx + 1
        Else
          Call MsgBox("StDev of data type " & _
               sTypeName(ty_decimal) & _
               " must not be zero!", vbOKOnly, "Error!")
          Exit Sub
        End If
      End If
      For j = 1 To vType(i)
        vInput(lIdx) = CDec(wsD.Cells(pr_decAvg, lCol) + _
                       (dThisDouble(j) - dAvg) * _
                       wsD.Cells(pr_decStDev, lCol) / dStDev)
        lIdx = lIdx + 1
      Next j
    End If
  Case ty_double
    If IsEmpty(wsD.Cells(pr_dAvg, lCol)) Or _
      IsEmpty(wsD.Cells(pr_dStDev, lCol)) Then
      'Work with Min and Max
      dmin = wsD.Cells(pr_dMin, lCol)
      dmax = wsD.Cells(pr_dMax, lCol)
      For j = 1 To vType(i)
        vInput(lIdx) = CDbl(dmin + Rnd() * (dmax - dmin))
        lIdx = lIdx + 1
      Next j
    Else
      'Work with Avg and StDev
      ReDim dThisDouble(1 To vType(i)) As Double
      For j = 1 To vType(i)
        dThisDouble(j) = Rnd()
      Next j
      dAvg = .Average(dThisDouble)
      dStDev = .StDevP(dThisDouble)
      If dStDev < 0.0000000000001 Then
        If vType(i) = 1 Then
          vInput(lIdx) = CDbl(dAvg)
          lIdx = lIdx + 1
        Else
          Call MsgBox("StDev of data type " & _
               sTypeName(ty_double) & _
               " must not be zero!", vbOKOnly, "Error!")
          Exit Sub
        End If
```

```
      End If
      For j = 1 To vType(i)
        vInput(lIdx) = CDbl(wsD.Cells(pr_dAvg, lCol) + _
                       (dThisDouble(j) - dAvg) * _
                       wsD.Cells(pr_dStDev, lCol) / dStDev)
        lIdx = lIdx + 1
      Next j
    End If
  Case ty_long
    If IsEmpty(wsD.Cells(pr_lSum, lCol)) Then
      If IsEmpty(wsD.Cells(pr_lMaxRepeat, lCol)) Then
        'Work with arbitrary repetitions
        dmin = wsD.Cells(pr_lMin2, lCol)
        dmax = wsD.Cells(pr_lMax, lCol)
        For j = 1 To vType(i)
          vInput(lIdx) = Int(dmin + Rnd() * (dmax - dmin + 1))
          lIdx = lIdx + 1
        Next j
      Else
        If (wsD.Cells(pr_lMax, lCol) - wsD.Cells(pr_lMin2, lCol) _
          + 1) * wsD.Cells(pr_lMaxRepeat, lCol) < vType(i) Then
          Call MsgBox("Not enough random numbers for data type " & _
              sTypeName(ty_long) & _
              "!", vbOKOnly, "Error!")
          Exit Sub
        End If
        v = sbRandInt(CLng(vType(i)), wsD.Cells(pr_lMin2, lCol), _
            wsD.Cells(pr_lMax, lCol), _
            wsD.Cells(pr_lMaxRepeat, lCol))
        For j = 1 To vType(i)
          vInput(lIdx) = v(j)
          lIdx = lIdx + 1
        Next j
      End If
    Else
      v = sbLongRandSumN(wsD.Cells(pr_lSum, lCol), vType(i), _
          wsD.Cells(pr_lMin1, lCol))
      For j = 1 To vType(i)
        vInput(lIdx) = v(j)
        lIdx = lIdx + 1
      Next j
    End If
  Case ty_string
    If Not IsEmpty(wsD.Cells(pr_sLength, lCol)) Then
      'Simple string
      lLength = wsD.Cells(pr_sLength, lCol)
      If lLength <= 0 Then lLength = 1
      dmin = Asc(wsD.Cells(pr_sMin, lCol))
      dmax = Asc(wsD.Cells(pr_sMax, lCol))
      For j = 1 To vType(i)
        s = ""
```

```
      For k = 1 To lLength
        s = s & Chr(dmin + Rnd() * (dmax - dmin))
      Next k
      vInput(lIdx) = s
      lIdx = lIdx + 1
    Next j
  ElseIf Not IsEmpty(wsD.Cells(pr_sNextTabRepeat, lCol)) Then
    'Simple items from next tab
    Set wsItem = Sheets(2)
    If (wsItem.Cells(1, wsD.Cells(pr_sNextTabColumn, _
      lCol)).End(xlDown).Row - 1) * _
      wsD.Cells(pr_sNextTabRepeat, lCol) < vType(i) Then
      Call MsgBox("Not enough random numbers for data type " & _
          sTypeName(ty_string) & "!", vbOKOnly, "Error!")
      Exit Sub
    End If
    v = sbRandInt(CLng(vType(i)), 2, _
        wsItem.Cells(1, wsD.Cells(pr_sNextTabColumn, _
        lCol)).End(xlDown).Row, _
        wsD.Cells(pr_sNextTabRepeat, lCol))
    For j = 1 To vType(i)
      vInput(lIdx) = _
        wsItem.Cells(1, wsD.Cells(pr_sNextTabColumn, lCol))(v(j))
      lIdx = lIdx + 1
    Next j
  Else
    'Items from weighted groups from next tab
    Set wsItem = Sheets(2)
    Set objGroup = CreateObject("Scripting.Dictionary")
    j = 2
    Do While Not IsEmpty(wsItem.Cells(j, _
      wsD.Cells(pr_sNextTabGroupColumn, lCol)))
      objGroup.Item(wsItem.Cells(j, _
          wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value) = _
          objGroup.Item(wsItem.Cells(j, _
          wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value) + 1
      j = j + 1
    Loop
    'Are the item groups still identical
    'to the ones in the param list?
    bGroupsUpToDate = True
    j = 0
    Do While Not IsEmpty(wsD.Cells(pr_sNextTabGroupWeights + j, _
      pc_ItemGroups))
      If objGroup.Item(wsD.Cells(pr_sNextTabGroupWeights + j, _
        pc_ItemGroups).Value) > 0 Then
        objGroup.Item(wsD.Cells(pr_sNextTabGroupWeights + j, _
          pc_ItemGroups).Value) = 0
      Else
        Set objGroup = Nothing
        Set objGroup = CreateObject("Scripting.Dictionary")
```

```
      j = 2
      Do While Not IsEmpty(wsItem.Cells(j, _
        wsD.Cells(pr_sNextTabGroupColumn, lCol)))
        objGroup.Item(wsItem.Cells(j, _
          wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value) = _
            objGroup.Item(wsItem.Cells(j, _
              wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value) + 1
        j = j + 1
      Loop
      bGroupsUpToDate = False
      Exit Do
    End If
    j = j + 1
  Loop
  If j <> objGroup.Count Then bGroupsUpToDate = False
  If Not bGroupsUpToDate Then
    Range(wsD.Cells(pr_sNextTabGroupWeights, pc_ItemGroups), _
      wsD.Cells(pr_sNextTabGroupWeights, _
        pc_ItemGroups).End(xlDown)).ClearContents
    wsD.Cells(pr_sNextTabGroupWeights, _
      pc_ItemGroups).Resize(objGroup.Count).FormulaArray = _
      .Transpose(objGroup.keys)
    If vbCancel = MsgBox("Item groups from next tab" & _
      " are not up to date!" & vbCrLf & _
      vbCrLf & "OK to continue anyway" & _
      vbCrLf & "Cancel to stop", vbOKCancel, "Warning") Then
      Exit Sub
    End If
  End If
  dSumWeights = 0#
  j = 0
  Do While Not IsEmpty(wsD.Cells(pr_sNextTabGroupWeights + j, _
    pc_ItemGroups))
    dSumWeights = dSumWeights + _
      wsD.Cells(pr_sNextTabGroupWeights + j, lCol)
    j = j + 1
  Loop
  ReDim dGroupWeights(1 To j) As Double
  For j = LBound(dGroupWeights) To UBound(dGroupWeights)
    dGroupWeights(j) = wsD.Cells(pr_sNextTabGroupWeights + _
      j - 1, lCol) / dSumWeights * vType(i)
  Next j
  'Decide how many records to generate for each item group
  vGroup = RoundToSum(dGroupWeights, 0)
  For j = LBound(vGroup, 1) To UBound(vGroup, 1)
    If vGroup(j) > 0 Then
      Set wsItem = Sheets(2)
      Set objItem = CreateObject("Scripting.Dictionary")
      lRow = 2
      Do While Not IsEmpty(wsItem.Cells(lRow, _
        wsD.Cells(pr_sNextTabGroupColumn, lCol)))
```

```
                If wsItem.Cells(lRow, _
                  wsD.Cells(pr_sNextTabGroupColumn, lCol)).Value = _
                  objGroup.keys()(j - 1) Then
                  objItem.Item(wsItem.Cells(lRow, _
                      wsD.Cells(pr_sNextTabItemColumn, lCol)).Value) = _
                      objItem.Item(wsItem.Cells(lRow, _
                      wsD.Cells(pr_sNextTabItemColumn, lCol)).Value) + 1
                End If
                lRow = lRow + 1
              Loop
              If objItem.Count * wsD.Cells(pr_sNextTabItemRepeat, _
                 lCol) < vGroup(j) Then
                Call MsgBox("Not enough random numbers for" & _
                     " data type string, item group " & _
                     wsD.Cells(pr_sNextTabGroupWeights + j, _
                     pc_ItemGroups).Value & _
                     "!", vbOKOnly, "Error!")
                Exit Sub
              End If
              v = sbRandInt(CLng(vGroup(j)), 1, objItem.Count, _
                wsD.Cells(pr_sNextTabItemRepeat, lCol))
              For k = 1 To vGroup(j)
                vInput(lIdx) = objItem.keys()(v(k) - 1)
                lIdx = lIdx + 1
              Next k
              Set objItem = Nothing
            End If
          Next j
          Set objGroup = Nothing
        End If
      End Select
    End If
  Next i
  'Now shuffle the result vector into random order if specified
  If wsD.Cells(pr_shuffle, lCol) Then
    lRow = 2
    For Each v In UniqRandInt(lRecord, lRecord)
      wsD.Cells(lRow, lCol - pc_Input1 + pc_Output1) = vInput(v)
      lRow = lRow + 1
    Next v
  Else
    For lRow = 2 To lRecord + 1
      wsD.Cells(lRow, lCol - pc_Input1 + pc_Output1) = vInput(lRow - 1)
    Next lRow
  End If
Next lCol
wsD.Calculate
End With

End Sub
```

A RoundToSum

```
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
```