

VBA code for Simtools.xla (3.31) and Formlist.xla (1.5). Copyright 1996-2000 by R. B. Myerson. All rights reserved.
See <http://home.uchicago.edu/~rmyerson/addins.htm>

```

Sub SIMTABLE()
Dim c As Integer, r As Integer, rng As Object, goon As Variant, mess As String, k As Long, rr As Integer
Set rng = Selection
c = rng.Columns.Count
r = rng.Rows.Count - 1
rr = r - 1
k = rng.Cells(2, 1).Row
Randomize
If (c = 1 Or r = 0) Then GoTo 1
If Application.Calculation <> xlAutomatic Then
    mess = "OK to set Calculation to Automatic?" & Chr(10) & "(To reset, see the Tools:Options menu.)"
    goon = MsgBox(Prompt:=mess, Buttons:=vbOKCancel)
    If goon = vbCancel Then Exit Sub
    Application.Calculation = xlAutomatic
End If
rng.Cells(1, 1).Value = "SimTable"
With rng.Cells(1, 1).Resize(1, c).Borders(xlBottom)
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
mess = "=(ROW()-" & k & ")/" & rr
rng.Cells(2, 1).Resize(r, 1).FormulaR1C1 = mess
rng.Table , rng.Cells(1, 1)
rng.Cells(2, 1).Resize(r, c).Copy
rng.Cells(2, 1).PasteSpecial Paste:=xlValues
Application.CutCopyMode = False
rng.Cells(2, 2).Resize(r, c - 1).Select
Exit Sub
1 MsgBox Prompt:="Select a range with simulation output in the top row, but not in the top-left cell. Recalculated values will fill the lower rows." _
    & " A percentile index will fill the leftmost column.", Title:="SIMULATION TABLE"
End Sub

Function POISINV(ByVal probability As Double, ByVal mean As Double)

```

```

On Error GoTo 16
Dim n As Long, v As Double, cumv As Double
If probability > 0.999999 Then probability = 0.999999
n = mean - 5 * (mean ^ 0.5) - 1
If n > 100 Then
    v = Exp(n - mean - n * Application.Ln(n / mean)) / (Application.Pi() * (2 * n + 1 / 3)) ^ 0.5
    cumv = v * mean / (mean - n)
Else
    n = 0
    v = Exp(-mean)
    cumv = v
End If
If v = 0 Then GoTo 16
Do While probability > cumv
    n = n + 1
    v = v * mean / n
    cumv = cumv + v
Loop
POISINV = n
Exit Function
16 POISINV = CVErr(xlErrNum)
End Function

```

Function GENLINV(ByVal probability As Single, ByVal quart1 As Single, ByVal quart2 As Single, ByVal quart3 As Single, Optional lowest, Optional highest)

```

On Error GoTo 3
Dim b As Single, norml As Single
If probability > 0.999999 Then probability = 0.999999
If probability < 0.000001 Then probability = 0.000001
If quart1 > quart3 Then GoTo 3
norml = Application.NormSInv(probability) / 0.67449
b = (quart3 - quart2) / (quart2 - quart1)
If b = 1 Then
    GENLINV = (quart3 - quart2) * norml + quart2
ElseIf b > 0 Then
    GENLINV = (quart3 - quart2) * (b ^ norml - 1) / (b - 1) + quart2
Else GoTo 3

```

```

End If
If Not IsMissing(lowest) Then
  If quart1 < lowest Then GoTo 3
  If GENLINV < lowest Then GENLINV = lowest
End If
If Not IsMissing(highest) Then
  If quart3 > highest Then GoTo 3
  If GENLINV > highest Then GENLINV = highest
End If
Exit Function
3 GENLINV = CVErr(xlErrNum)
End Function

```

```

Function TRIANINV(ByVal probability As Single, ByVal lowerbound As Single, ByVal mostlikely As Single, ByVal upperbound As Single)
On Error GoTo 18
Dim x As Single
If probability > 1 Or probability < 0 Then GoTo 18
If lowerbound >= upperbound Then GoTo 18
x = (mostlikely - lowerbound) / (upperbound - lowerbound)
If (x > 1 Or x < 0) Then GoTo 18
If probability <= x Then TRIANINV = lowerbound + (upperbound - lowerbound) * ((probability * x) ^ 0.5)
If probability > x Then TRIANINV = upperbound - (upperbound - lowerbound) * (((1 - probability) * (1 - x)) ^ 0.5)
Exit Function
18 TRIANINV = CVErr(xlErrNum)
End Function

```

```

Function EXPOINV(ByVal probability As Double, ByVal mean As Double)
On Error GoTo 2
EXPOINV = -mean * Application.Ln(1 - probability)
Exit Function
2 EXPOINV = CVErr(xlErrNum)
End Function

```

```

Function UTIL(ByVal income As Double, ByVal RiskTolConst As Double, Optional RiskTolSlope)
On Error GoTo 7
If Not IsMissing(RiskTolSlope) Then

```

```

If RiskTolSlope <> 0 Then
    UTIL = Application.Ln(RiskTolConst + RiskTolSlope * income)
    If RiskTolSlope <> 1 Then UTIL = Exp(UTIL * (1 - 1 / RiskTolSlope)) / (RiskTolSlope - 1)
    Exit Function
End If
End If
UTIL = -Exp(-income / RiskTolConst)
If RiskTolConst < 0 Then UTIL = -UTIL
Exit Function
7 UTIL = CVErr(xlErrNum)
End Function

Function UINV(ByVal utility As Double, ByVal RiskTolConst As Double, Optional RiskTolSlope)
On Error GoTo 36
If Not IsMissing(RiskTolSlope) Then
If RiskTolSlope <> 0 Then
    If RiskTolSlope <> 1 Then utility = Application.Ln((RiskTolSlope - 1) * utility) / (1 - 1 / RiskTolSlope)
    UINV = (Exp(utility) - RiskTolConst) / RiskTolSlope
    Exit Function
End If
End If
If RiskTolConst < 0 Then utility = -utility
UINV = -RiskTolConst * Application.Ln(-utility)
Exit Function
36 UINV = CVErr(xlErrNum)
End Function

Function BINOMINV(ByVal probability As Double, ByVal n As Integer, ByVal p As Double)
On Error GoTo 17
Dim x As Integer, pptr As Double, cumv As Double, rev As Integer
rev = 0
If p > 0.5 Then
    rev = 1
    probability = 1 - probability
    p = 1 - p
End If

```

```

If n <= 0 Then
  BINOMINV = 0
  Exit Function
End If
If p < 0 Then p = 0
x = 0
ptpr = (1 - p) ^ n
If ptp = 0 Then GoTo 17
cumv = ptp
Do While (probability > cumv And x < n)
  x = x + 1
  ptp = ptp * (n + 1 - x) * p / (x * (1 - p))
  cumv = cumv + ptp
Loop
BINOMINV = (1 - rev) * x + rev * (n - x)
Exit Function
17 BINOMINV = CVErr(x!ErrNum)
End Function

```

```

Function ARGMAX(labels As Object, values As Object, Optional testCells, Optional criterion)
Dim i As Integer, j As Integer, k As Integer, r As Integer, c As Integer, y As Double, x As Variant, crit As Variant
On Error GoTo 5
r = labels.Rows.Count
c = labels.Columns.Count
If (values.Rows.Count <> r Or values.Columns.Count <> c) Then GoTo 5
If IsMissing(testCells) Then
  For i = 1 To r
    For j = 1 To c
      y = values.Cells(i, j).Value
      If IsEmpty(x) Or y > x Then
        x = y
        ARGMAX = labels.Cells(i, j).Value
      End If
    Next j
  Next i
Exit Function

```

```

End If
If (testCells.Rows.Count <> r Or testCells.Columns.Count <> c Or IsMissing(criterion)) Then GoTo 5
crit = criterion
For i = 1 To r
For j = 1 To c
If Application.CountIf(testCells.Cells(i, j), crit) = 1 Then
y = values.Cells(i, j).Value
If IsEmpty(x) Or y > x Then
x = y
ARGMAX = labels.Cells(i, j).Value
End If
End If
Next j
Next i
If IsEmpty(x) Then ARGMAX = CVErr(xlErrNull)
Exit Function
5 ARGMAX = CVErr(xlErrValue)
End Function

```

```

Function CEPR(values As Object, probabilities As Object, ByVal RiskTolConst As Double, Optional testCells, Optional criterion)
Dim i As Integer, j As Integer, k As Integer, r As Integer, c As Integer, p As Double, v As Double, prval As Double, proby As Double
Dim crit As Variant, linr As Boolean, havtes As Boolean, goon As Boolean
On Error GoTo 9
linr = (0 = RiskTolConst)
proby = 0
prval = 0
r = values.Rows.Count
c = values.Columns.Count
If (probabilities.Rows.Count <> r Or probabilities.Columns.Count <> c) Then GoTo 9
havtes = Not IsMissing(testCells)
If havtes Then
If (testCells.Rows.Count <> r Or testCells.Columns.Count <> c Or IsMissing(criterion)) Then GoTo 9
crit = criterion
Else
goon = True
End If

```

```

For i = 1 To r
For j = 1 To c
  If havtes Then
    goon = (Application.CountIf(testCells.Cells(i, j), crit) = 1)
  End If
  If goon Then
    p = probabilities.Cells(i, j).Value
    v = values.Cells(i, j).Value
    If p < 0 Then p = 0
    proby = proby + p
    If linr Then prval = prval + p * v Else prval = prval + p * Exp(-v / RiskTolConst)
  End If
Next j
Next i
8 If proby = 0 Then
  CEPR = CVErr(xlErrDiv0)
Else
  CEPR = prval / proby
  If Not linr Then CEPR = -RiskTolConst * Application.Ln(CEPR)
End If
Exit Function
9 CEPR = CVErr(xlErrValue)
End Function

Sub COMBINE()
Dim r1 As Integer, r2 As Integer, c1 As Integer, c2 As Integer, tp1 As Long, tp2 As Long, i As Integer, mess0 As String, mess1 As String
Dim insrt As Variant, rng1 As Object, rng1A As Object, mess2 As String, rng2 As Object, rng2A As Object, mess3 As String, goon As Variant, calx As Long
mess0 = "Is it OK to insert new rows into the worksheet? (Otherwise, cells below the combination range will be overwritten.)"
insrt = MsgBox(Prompt:=mess0, Title:="INSERT NEW ROWS?", Buttons:=vbYesNo)
calx = Application.Calculation
On Error GoTo 38
Application.EnableCancelKey = xlErrorHandler
Application.Calculation = xlManual
mess1 = "Select a range of rows to be extended by combinations. "
If insrt = vbYes Then
  mess1 = mess1 & "New worksheet rows will be inserted at the bottom of this range."

```

```

Else
    mess1 = mess1 & "Cells below this range may be overwritten."
End If
10 Set rng1A = Application.InputBox(Prompt:=mess1, Title:="SELECT RANGE TO BE EXTENDED", default:=Selection.Address, Top:=32, Type:=8)
r1 = rng1A.Rows.Count
c1 = rng1A.Columns.Count
Set rng1 = rng1A.Cells(1, 1)
tp1 = rng1.Row
rng1.Worksheet.Activate
mess2 = "Select another range. Each row in this range will be copied to the right of a copy of each row in the previously selected range."
rng1.Offset(0, c1).Select
Set rng2A = Application.InputBox(Prompt:=mess2, Title:="SELECT EXTENDING RANGE", default:=Selection.Address, Top:=32, Type:=8)
r2 = rng2A.Rows.Count
c2 = rng2A.Columns.Count
Set rng2 = rng2A.Cells(1, 1)
tp2 = rng2.Row
If insrt = vbYes And r2 > 1 Then rng1.Offset(r1, 0).Resize(r1 * (r2 - 1), 1).EntireRow.Insert
rng2.Resize(r2, c2).Copy (rng1.Offset(0, c1))
If insrt = vbYes And rng1.Worksheet.Name = rng2.Worksheet.Name Then
    If tp2 + r2 > tp1 + r1 And tp2 < tp1 + r1 And r2 > 1 Then
        rng2.Offset(r1 * r2 + tp1 - tp2, 0).Resize(r2 + tp2 - (r1 + tp1), c2).Copy (rng1.Offset(tp1 + r1 - tp2, c1))
    End If
End If
For i = 1 To r1
    rng1.Offset(r1 - i, 0).Resize(1, c1).Copy (rng1.Offset(r2 * (r1 - i), 0).Resize(r2, 1))
    rng1.Offset(0, c1).Resize(r2, c2).Copy (rng1.Offset(r2 * (r1 - i), c1))
Next i
r1 = r1 * r2
c1 = c1 + c2
rng1.Resize(r1, c1).Select
mess3 = "A range containing " & r1 & " combined rows has been made. Continue making combinations?"
goon = MsgBox(Prompt:=mess3, Title:="COMBINE ROWS", Buttons:=vbYesNo + vbDefaultButton2)
If goon = vbYes Then GoTo 10
Application.Calculation = calx
mess2 = "In the selected range, any cell that is the same as the cell above will displayed in yellow half-size font."
Set rng2 = Application.InputBox(Prompt:=mess2, Title:="HIGHLIGHT CHANGES IN TABLE", default:=Selection.Address, Top:=32, Type:=8)

```

```

i = (rng2.Cells(1, 1).Font.Size + 1) / 2
r1 = rng2.Rows.Count
c1 = rng2.Columns.Count
For r2 = 2 To r1
  For c2 = 1 To c1
    If rng2.Cells(r2, c2).Value = rng2.Cells(r2 - 1, c2).Value Then
      rng2.Cells(r2, c2).Font.Color = RGB(255, 255, 0)
      rng2.Cells(r2, c2).Font.Size = i
    End If
  Next c2
Next r2
Exit Sub
38 Application.Calculation = calx
End Sub

Function LGTINV(p)
On Error GoTo 13
Dim n As Integer, j As Integer, y() As Double, yy As Double, c As Double
n = 1
Do While (Not IsError(Application.Index(p, 1, n + 1)))
  n = n + 1
Loop
ReDim y(1 To 1, 1 To n)
c = Exp(-50)
yy = Application.Index(p, 1, n)
If n = 1 Then yy = 1 - yy
If (yy < c And yy >= 0) Then yy = c
For j = 1 To n
  y(1, j) = Application.Index(p, 1, j)
  If (y(1, j) < c And y(1, j) >= 0) Then y(1, j) = c
  y(1, j) = Application.Ln(y(1, j) / yy)
Next j
LGTINV = y
Exit Function
13 LGTINV = CVErr(xlErrValue)
End Function

```

```

Function LGT(x)
On Error GoTo 15
Dim n As Integer, j As Integer, y() As Double, yy As Double
Do While (Not IsError(Application.Index(x, 1, n + 1)))
    n = n + 1
Loop
ReDim y(1 To 1, 1 To n)
yy = 0
For j = 1 To n
    y(1, j) = Exp(Application.Index(x, 1, j))
    yy = yy + y(1, j)
Next j
If n = 1 Then yy = yy + 1
For j = 1 To n
    y(1, j) = y(1, j) / yy
Next j
LGT = y
Exit Function
15 LGT = CVErr(xlErrValue)
End Function

```

```

Function YHATSTE(XDataRange As Object, NewXRow As Object, ByVal RegrSSnStdErr As Double)
On Error GoTo 11
Dim r As Integer, c As Integer, i As Integer, j As Integer, num As Double, x() As Double, w() As Double
r = XDataRange.Rows.Count
c = XDataRange.Columns.Count
If NewXRow.Columns.Count <> c Then GoTo 11
ReDim x(1 To r, 1 To c + 1), w(1 To c + 1)
For j = 1 To c
    w(j) = NewXRow.Cells(1, j).Value
Next j
w(c + 1) = 1
For i = 1 To r
    For j = 1 To c
        x(i, j) = XDataRange.Cells(i, j).Value
    Next j

```

```

    x(i, c + 1) = 1
Next i
num = Application.SumProduct(w, Application.MMult(w, Application.MInverse(Application.MMult(Application.Transpose(x), x))))
YHATSTE = RegrSsnStdErr * (num ^ 0.5)
Exit Function
11 YHATSTE = CVer(xlErrValue)
End Function

Function REGRESSN(XDataRange As Object, YDataRange As Object) As Variant
On Error GoTo 45
Dim r As Integer, c As Integer, i As Integer, j As Integer, b As Variant, y As Double
Dim out() As Variant, x() As Double, sumer As Double, YX() As Double, k As Integer, xx() As Double
r = XDataRange.Rows.Count
c = XDataRange.Columns.Count
If (YDataRange.Rows.Count <> r Or YDataRange.Columns.Count <> 1 Or r <= (c + 1)) Then GoTo 45
ReDim x(1 To c), xx(1 To c + 1, 1 To c + 1), YX(1 To c + 1), out(1 To 7, 1 To c)
sumer = 0
For i = 1 To r
    y = YDataRange.Cells(i, 1).Value
    YX(c + 1) = YX(c + 1) + y
    sumer = sumer + y * y
    For j = 1 To c
        x(j) = XDataRange.Cells(i, j).Value
        xx(j, c + 1) = xx(j, c + 1) + x(j)
        YX(j) = YX(j) + x(j) * y
        For k = 1 To j
            xx(j, k) = xx(j, k) + x(j) * x(k)
        Next k
    Next j
Next i
xx(c + 1, c + 1) = r
For j = 1 To c
    For k = 1 To j - 1
        xx(k, j) = xx(j, k)
    Next k
    xx(c + 1, j) = xx(j, c + 1)

```

```

Next j
b = Application.MMult(YX, Application.MInverse(xx))
sumer = sumer - Application.SumProduct(YX, b)
For j = 1 To c
    out(1, j) = ""
    out(2, j) = "X(" & j & ") Coefficient"
    out(3, j) = b(j)
Next j
out(1, 1) = "Regressn Array ... 7 Rows x " & c & " Columns"
out(4, 1) = "Intercept"
out(5, 1) = b(c + 1)
out(6, 1) = "Std Error"
out(7, 1) = (sumer / (r - (c + 1))) ^ 0.5
For i = 4 To 7
    For j = 2 To c
        out(i, j) = ""
    Next j
Next i
REGRESSN = out
Exit Function
45 REGRESSN = CVErr(xlErrValue)
End Function

Function MSQRT(squareArray) As Variant
On Error GoTo 14
Dim n As Integer, i As Integer, j As Integer, k As Integer, x() As Double
If IsObject(squareArray) Then
    n = squareArray.Columns.Count
    If n <> squareArray.Rows.Count Then GoTo 14
ElseIf IsArray(squareArray) Then
    n = UBound(squareArray, 1) + 1 - LBound(squareArray, 1)
    If n <> UBound(squareArray, 2) + 1 - LBound(squareArray, 2) Then GoTo 14
Else n = 1
End If
ReDim x(1 To n, 1 To n)
For j = n To 1 Step -1

```

```

x(j, j) = Application.Index(squareArray, j, j)
For k = n To j + 1 Step -1
  x(k, j) = Application.Index(squareArray, k, j)
  x(j, k) = 0
  For i = n To k + 1 Step -1
    x(k, j) = x(k, j) - x(i, j) * x(i, k)
  Next i
  If x(k, j) <> 0 Then x(k, j) = x(k, j) / x(k, k)
  x(j, j) = x(j, j) - (x(k, j) ^ 2)
Next k
x(j, j) = x(j, j) ^ 0.5
Next j
MSQRT = x
Exit Function
14 MSQRT = CVErr(xlErrValue)
End Function

```

```

Function CORAND(correlArray, Optional RANDsource) As Variant
Application.Volatile (IsError(Application.Index(RANDsource, 1, 2)))
On Error GoTo 19
Dim n As Integer, i As Integer, j As Integer, k As Integer, x() As Single, y() As Single, a() As Single, m As Integer, L As Single
If IsObject(correlArray) Then
  n = correlArray.Columns.Count
ElseIf IsArray(correlArray) Then
  n = UBound(correlArray, 2) + 1 - LBound(correlArray, 2)
Else n = 1
End If
ReDim a(1 To n, 1 To n)
If n > 2 Then
  m = n
  L = 0.999999
Else
  m = 2
  L = 1
End If
ReDim y(1 To m), x(1 To m)

```

```

If IsMissing(RANDsource) Then
  For i = 1 To m - 1
    x(i) = Application.NormSInv(Rnd)
  Next i
  y(m) = Rnd
ElseIf IsError(Application.Index(RANDsource, 1, 2)) Then
  For i = 1 To m - 1
    x(i) = Application.NormSInv(Rnd)
  Next i
  y(m) = RANDsource
Else
  For i = 1 To m - 1
    x(i) = Application.NormSInv(Application.Index(RANDsource, 1, i))
  Next i
  y(m) = Application.Index(RANDsource, 1, m)
  If Not IsError(Application.Index(RANDsource, 1, m + 1)) Then GoTo 19
End If
a(n, n) = Application.Index(correlArray, n, n)
x(m) = Application.NormSInv(y(m))
If n = 1 Then
  x(1) = a(1, 1) * x(2) + ((1 - a(1, 1) ^ 2) ^ 0.5) * x(1)
  y(1) = Application.NormSDist(x(1))
  CORAND = y
  Exit Function
End If
If a(n, n) <> 1 Then GoTo 19
For j = n - 1 To 1 Step -1
  a(j, j) = Application.Index(correlArray, j, j)
  If a(j, j) <> 1 Then GoTo 19
  y(j) = 0
  For k = n To j + 1 Step -1
    a(k, j) = Application.Index(correlArray, k, j) * L
    For i = k + 1 To n
      a(k, j) = a(k, j) - a(i, j) * a(i, k)
    Next i
    If a(k, j) <> 0 Then a(k, j) = a(k, j) / a(k, k)
  
```

```

    y(j) = y(j) + a(k, j) * x(k)
    a(j, j) = a(j, j) - a(k, j) ^ 2
Next k
a(j, j) = a(j, j) ^ 0.5
y(j) = Application.NormSDist(y(j) + a(j, j) * x(j))
Next j
CORAND = y
Exit Function
19 CORAND = CVErr(xlErrValue)
End Function

```

```

Function GAMINV(ByVal probability As Single, ByVal mean As Single, ByVal stdevn As Single)
On Error GoTo 29
GAMINV = Application.GammaInv(probability, (mean / stdevn) ^ 2, (stdevn ^ 2) / mean)
Exit Function
29 GAMINV = CVErr(xlErrNum)
End Function

```

```

Function BETINV(ByVal probability As Single, ByVal mean As Single, ByVal stdevn As Single, Optional lowerbound, Optional upperbound)
On Error GoTo 30
Dim u As Single, L As Single, m As Single, s As Single, n As Single
u = 1
L = 0
If Not IsMissing(upperbound) Then u = upperbound
If Not IsMissing(lowerbound) Then L = lowerbound
If u <= L Then GoTo 30
m = (mean - L) / (u - L)
s = stdevn / (u - L)
n = m * (1 - m) / (s ^ 2) - 1
BETINV = L + (u - L) * Application.BetaInv(probability, m * n, (1 - m) * n)
Exit Function
30 BETINV = CVErr(xlErrNum)
End Function

```

```

Function LNORMINV(ByVal probability As Single, ByVal mean As Single, ByVal stdevn As Single)
On Error GoTo 31

```

```

Dim sig2 As Single
sig2 = Application.Ln(1 + ((stdevn / mean) ^ 2))
LNORMINV = Exp(Application.NormInv(probability, Application.Ln(mean) - (0.5 * sig2), sig2 ^ 0.5))
Exit Function
31 LNORMINV = CVErr(xlErrNum)
End Function

```

```

Sub ABOUTSIM()
MsgBox Prompt:="Simtools (3.31) was developed at the Kellogg Graduate School of Management, Northwestern University." & Chr(10) & Chr(169) _
& " 1996-2000 by R. B. Myerson.", Title:="ABOUT SIMTOOLS"
End Sub

```

```

Function COVARPR(values1 As Object, values2 As Object, probabilities As Object)
On Error GoTo 40
Dim c As Integer, r As Integer, sump As Double, mean1 As Double, mean2 As Double, PRODS As Double, p As Double, i As Integer, j As Integer
c = probabilities.Columns.Count
r = probabilities.Rows.Count
If (values1.Rows.Count <> r Or values1.Columns.Count <> c) Then GoTo 40
If (values2.Rows.Count <> r Or values2.Columns.Count <> c) Then GoTo 40
sump = 0
mean1 = 0
mean2 = 0
PRODS = 0
For i = 1 To r
For j = 1 To c
p = probabilities.Cells(i, j).Value
If p < 0 Then p = 0
sump = sump + p
mean1 = mean1 + values1.Cells(i, j).Value * p
mean2 = mean2 + values2.Cells(i, j).Value * p
PRODS = PRODS + values1.Cells(i, j).Value * values2.Cells(i, j).Value * p
Next j
Next i
If (sump > 1.01 Or sump < 0.99) Then GoTo 40
COVARPR = (PRODS - (mean1 * mean2 / sump)) / sump
Exit Function

```

```
40 COVARPR = CVErr(xlErrValue)
```

```
End Function
```

```
Function STDEVPR(values As Object, probabilities As Object)
```

```
On Error GoTo 41
```

```
STDEVPR = COVARPR(values, values, probabilities) ^ 0.5
```

```
Exit Function
```

```
41 STDEVPR = CVErr(xlErrValue)
```

```
End Function
```

```
Function CORRELPR(values1 As Object, values2 As Object, probabilities As Object)
```

```
On Error GoTo 42
```

```
CORRELPR = COVARPR(values1, values2, probabilities) / ((COVARPR(values1, values1, probabilities) * COVARPR(values2, values2, probabilities)) ^ 0.5)
```

```
Exit Function
```

```
42 CORRELPR = CVErr(xlErrValue)
```

```
End Function
```

```
Sub MARKOV()
```

```
Dim oldst As Object, newst As Object, out As Object, topout As Object, r As Integer, c As Integer, i As Integer, calx As Long
```

```
Set oldst = Application.InputBox(Title:="SELECT STATE RANGE", Prompt:="Select a range of cells containing the values that represent the current " _  
& "state of the process.", default:=Selection.Cells(1, 1).Address, Top:=32, Type:=8)
```

```
Set newst = Application.InputBox(Title:="SELECT UPDATE RANGE", Prompt:="Select a range containing formulas that compute the next state of the " & _  
"process. At each iteration, this range will be copied and pasted-special-values onto the state range.", default:=Selection.Cells(1, 1).Address, _  
Top:=32, Type:=8)
```

```
If (oldst.Columns.Count <> newst.Columns.Count Or oldst.Rows.Count <> newst.Rows.Count Or newst.Areas.Count > 1) Then
```

```
MsgBox Prompt:="State range and update range must have the same size."
```

```
Exit Sub
```

```
End If
```

```
Set out = Application.InputBox(Title:="SELECT OUTPUT TABLE", _
```

```
Prompt:="Select a range with model output in the top row, but with the top-left cell unused. Output will be tabulated below at each " _  
& "iteration, with iteration numbers in the left column.", default:=Selection.Address, Top:=32, Type:=8)
```

```
r = out.Rows.Count
```

```
c = out.Columns.Count
```

```
Randomize
```

```
If (r <= 1 Or c <= 1) Then
```

```
MsgBox Prompt:="Output range must contain at least 2 rows and at least 2 columns.", Title:="ITERATIVE PROCESS"
```

```

Exit Sub
End If
Set topout = out.Cells(1, 2).Resize(1, c - 1)
calx = Application.Calculation
On Error GoTo 39
Application.EnableCancelKey = xlErrorHandler
Application.Calculation = xlManual
With out.Cells(1, 1).Resize(1, c).Borders(xlBottom)
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
Application.ScreenUpdating = False
Application.Calculate
For i = 2 To r
    topout.Copy
    out.Cells(i, 2).PasteSpecial Paste:=xlValues
    out.Cells(i, 1).Value = i - 1
    Application.StatusBar = "Iteration " & (i - 1) & ". First output cell = " & topout.Cells(1, 1).Value
    newst.Copy
    oldst.Cells(1, 1).PasteSpecial Paste:=xlValues
    Application.Calculate
Next i
out.Cells(1, 1).Value = "Iterations (" & newst.Address(rowabsolute:=False, columnabsolute:=False) & " to " _
    & oldst.Address(rowabsolute:=False, columnabsolute:=False) & ")"
out.Worksheet.Activate
out.Cells(1, 1).Select
39 Application.Calculation = calx
Application.ScreenUpdating = True
Application.StatusBar = False
End Sub

```

```

Function CE(incomes As Object, ByVal RiskTolConst As Double, Optional RiskTolSlope)

```

```

On Error GoTo 44

```

```

Dim y As Double, x As Double, s As Double, t As Double, u As Double, k As Integer, n As Integer, cnst As Boolean, strt As Boolean, cell As Object

```

```

cnst = True

```

```

If Not IsMissing(RiskTolSlope) Then

```

```

s = RiskTolSlope
cnst = (CSng(1 + s) = 1)
End If
If cnst Then
t = RiskTolConst
ElseIf CSng(s) <> 1 Then
t = s / (1 - s)
Else
t = 0
End If
If t < 0 Then k = -1 Else k = 1
t = k * t
strt = True
If t <> 0 Then
For Each cell In incomes
If Application.Count(cell) = 1 Then
n = n + 1
If cnst Then x = k * cell.Value Else x = k * Application.Ln(RiskTolConst + s * cell.Value)
If strt Then
y = x
strt = False
End If
If y <= x Then
u = u + Exp((y - x) / t)
Else
u = u * Exp((x - y) / t) + 1
y = x
End If
End If
Next cell
CE = k * (y - t * Application.Ln(u / n))
Else
For Each cell In incomes
If Application.Count(cell) = 1 Then
n = n + 1
If cnst Then x = cell.Value Else x = Application.Ln(RiskTolConst + s * cell.Value)

```

```

    u = u + x
  End If
Next cell
CE = u / n
End If
If Not cnst Then CE = (Exp(CE) - RiskTolConst) / s
Exit Function
44 CE = CVErr(xlErrValue)
End Function

```

```

Function DIRICH(alphaArray As Object, Optional RANDsource) As Variant
Application.Volatile (IsMissing(RANDsource))
On Error GoTo 59
Dim m As Integer, n As Integer, i As Integer, j As Integer, x() As Double, y As Double, z As Double, rp As Double, miss As Boolean
m = alphaArray.Rows.Count
n = alphaArray.Columns.Count
ReDim x(1 To m, 1 To n)
y = 0
miss = IsMissing(RANDsource)
If Not miss Then
  If RANDsource.Rows.Count <> m Or RANDsource.Columns.Count <> n Then GoTo 59
End If
For i = 1 To m
For j = 1 To n
  If miss Then rp = Rnd Else rp = RANDsource.Cells(i, j).Value
  z = alphaArray.Cells(i, j).Value
  If z > 340 Then
    x(i, j) = Application.NormInv(rp, z, z ^ 0.5)
  Else x(i, j) = Application.GammaInv(rp, z, 1)
  End If
  y = y + x(i, j)
Next j
Next i
For i = 1 To m
For j = 1 To n
  x(i, j) = x(i, j) / y

```

```

Next j
Next i
DIRICH = x
Exit Function
59 DIRICH = CVErr(xlErrValue)
End Function

```

```

Function DIRALPHA(dataRange As Object) As Variant
On Error GoTo 44
Dim r As Integer, c As Integer, i As Integer, j As Integer, x() As Double, y As Double
r = dataRange.Rows.Count
c = dataRange.Columns.Count
ReDim x(1 To c)
y = 1
For j = 1 To c
    x(j) = 1
    For i = 1 To r
        x(j) = x(j) * dataRange.Cells(i, j).Value
    Next i
    x(j) = x(j) ^ (1 / r)
    y = y - x(j)
Next j
For j = 1 To c
    x(j) = 0.5 * ((c - 1) * x(j) + y) / y
Next j
DIRALPHA = x
Exit Function
44 DIRALPHA = CVErr(xlErrValue)
End Function

```

```

Function DISCRINV(ByVal randprob As Double, values As Object, probabilities As Object)
On Error GoTo 63
Dim i As Integer, cumv As Double, cel As Object
If values.Count <> probabilities.Count Then GoTo 63
For Each cel In probabilities
    i = i + 1

```

```

cumv = cumv + cel.Value
If randprob < cumv Then
  DISCRINV = values.Cells(i).Value
  Exit Function
End If
Next cel
If randprob < cumv + 0.001 Then
  DISCRINV = values.Cells(i).Value
  Exit Function
End If
63 DISCRINV = CVErr(xlErrValue)
End Function

```

```

Function RISKTOL(ByVal HighIncome As Double, ByVal LowIncome As Double, ByVal CertainEquiv As Double)
On Error GoTo 48
Dim x As Double, y As Double, z As Double, rat As Double, i As Integer, a As Double, b As Double, k As Integer
If HighIncome <= LowIncome Then GoTo 48
rat = (CertainEquiv - LowIncome) / (HighIncome - LowIncome)
k = 1
If CSng(rat) = 0.5 Then
  RISKTOL = CVErr(xlErrDiv0)
  Exit Function
ElseIf rat > 0.5 Then
  rat = 1 - rat
  k = -1
End If
If rat < 0 Then GoTo 48
x = 0.1251 / (0.5 - rat)
z = rat / 0.6932
For i = 1 To 15
  y = (x + z) / 2
  If -y * Application.Ln(0.5 * Exp(-1 / y) + 0.5) > rat Then x = y Else z = y
Next i
a = -x * Application.Ln(0.5 * Exp(-1 / x) + 0.5)
b = -z * Application.Ln(0.5 * Exp(-1 / z) + 0.5)
If a <> b Then

```

```

y = z + (x - z) * (rat - b) / (a - b)
RISKTOL = k * (HighIncome - LowIncome) * y
Else
  RISKTOL = k * (HighIncome - LowIncome) * (x + z) / 2
End If
Exit Function
48 RISKTOL = CVErr(xlErrValue)
End Function

Function MCCRRELS(dataRange As Object) As Variant
On Error GoTo 20
Dim r As Integer, n As Integer, rr As Integer, i As Integer, j As Integer, k As Integer, doit As Integer
Dim x() As Variant, mc() As Double, ss() As Double, m() As Double, ob As Object
r = dataRange.Rows.Count
n = dataRange.Columns.Count
ReDim ss(1 To n, 1 To n), m(1 To n), x(1 To n), mc(1 To n, 1 To n)
rr = r
For i = 1 To r
  doit = 0
  For j = 1 To n
    Set ob = dataRange.Cells(i, j)
    doit = doit + Application.Count(ob)
    x(j) = ob.Value
  Next j
  If doit = n Then
    For j = 1 To n
      m(j) = m(j) + x(j)
      For k = 1 To j
        ss(j, k) = ss(j, k) + x(j) * x(k)
      Next k
    Next j
  Else
    rr = rr - 1
  End If
Next i
For j = 1 To n

```

```

ss(j, j) = (ss(j, j) - m(j) * m(j) / rr) ^ 0.5
mc(j, j) = 1
For k = 1 To j - 1
    ss(j, k) = ss(j, k) - m(j) * m(k) / rr
    mc(j, k) = ss(j, k) / (ss(j, j) * ss(k, k))
    mc(k, j) = mc(j, k)
Next k
Next j
MCORRELS = mc
Exit Function
20 MCORRELS = CVErr(xlErrValue)
End Function

Function NORMIZE(datacolumn As Object) As Variant
On Error GoTo 53
Dim src() As Integer, rnk() As Integer, data() As Double, temps() As Integer
Dim n As Integer, m As Integer, i As Integer, r() As Variant
n = datacolumn.Count
If Application.Caller.Rows.Count < n Or Application.Caller.Columns.Count > 1 Then
    NORMIZE = "NORMIZE array requires " & n & " rows x 1 column."
Exit Function
End If
ReDim src(1 To n), rnk(1 To n), data(1 To n), temps(1 To n), r(1 To n, 1 To 1)
For i = 1 To n
    If Application.Count(datacolumn.Cells(i)) = 1 Then
        m = m + 1
        src(m) = i
        data(i) = datacolumn.Cells(i).Value
    Else
        r(i, 1) = "..."
    End If
Next i
QRANK data(), src(), rnk(), 1, m, temps()
For i = 1 To m
    r(src(i), 1) = Application.NormSInv(rnk(i) / (2 * m))
Next i

```

```

NORMIZE = r
Exit Function
53 NORMIZE = CVErr(xlErrValue)
End Function

```

```

Private Sub QRANK(data() As Double, src() As Integer, rnk() As Integer, ByVal low As Integer, ByVal hi As Integer, temps() As Integer)
Dim k As Integer, i As Integer, j1 As Integer, j2 As Integer, j3 As Integer, L As Integer, midval As Double
midval = data(src(low + Int(Rnd * (hi + 1 - low))))
For i = low To hi
  Select Case (data(src(i)) - midval)
    Case Is < 0
      src(low + j1) = src(i)
      j1 = j1 + 1
    Case Is > 0
      temps(hi - j3) = src(i)
      j3 = j3 + 1
    Case Else
      j2 = j2 + 1
      temps(j2) = src(i)
  End Select
Next i
k = low + j1 - 1
L = 2 * k + j2
For i = k + 1 To k + j2
  src(i) = temps(i - k)
  rnk(i) = L
Next i
For i = low + j1 + j2 To hi
  src(i) = temps(i)
Next i
If j1 > 1 Then
  QRANK data(), src(), rnk(), low, low + j1 - 1, temps()
ElseIf j1 = 1 Then rnk(low) = 2 * low - 1
End If
If j3 > 1 Then
  QRANK data(), src(), rnk(), hi + 1 - j3, hi, temps()

```

```

ElseIf j3 = 1 Then rnk(hi) = 2 * hi - 1
End If
End Sub

```

```

Function PRODS(values As Object) As Variant
On Error GoTo 21
Dim n As Integer, x() As Double, y() As Double, i As Integer, j As Integer, cell As Object
n = values.Count
If n + 1 > values.Rows.Count + values.Columns.Count Then GoTo 21
ReDim x(1 To n), y(1 To n, 1 To n)
For Each cell In values
    i = i + 1
    x(i) = cell.Value
Next cell
For i = 1 To n
    y(i, i) = x(i) * x(i)
    For j = 1 To i - 1
        y(i, j) = x(i) * x(j)
        y(j, i) = y(i, j)
    Next j
Next i
PRODS = y
Exit Function
21 PRODS = CVErr(xlErrValue)
End Function

```

```

Function XTREMINV(ByVal probability As Single, ByVal mean As Single, ByVal stdevn As Single)
On Error GoTo 33
If stdevn < 0 Then probability = 1 - probability
XTREMINV = 0.4500535 + 0.7796968 * Application.Ln(-Application.Ln(probability))
XTREMINV = mean - stdevn * XTREMINV
Exit Function
33 XTREMINV = CVErr(xlErrNum)
End Function

```

```

Function MIDRAND(ByVal correlation As Double, ByVal givenCoValue As Double)

```

```

On Error GoTo 54
If Abs(correlation) > 1 Then GoTo 54
MIDRAND = Application.NormSDist(correlation * Application.NormSInv(givenCoValue))
Exit Function
54 MIDRAND = CVErr(xlErrNum)
End Function

```

```

Function SHUFFLE(ByVal n As Integer, Optional RANDsource) As Variant
Application.Volatile (IsMissing(RANDsource))
On Error GoTo 37
Dim i As Integer, r As Double, k As Integer, eps As Double, d() As Integer, sh() As Integer
ReDim d(1 To n), sh(1 To 1, 1 To n)
If IsMissing(RANDsource) Then r = Rnd Else r = RANDsource
eps = (10 ^ -10) * r
For i = 1 To n
    r = (n + 1 - i) * (r + eps)
    k = Int(r)
    r = r - k
    If i + k > n Then k = 0
    sh(1, i) = i + k + d(i + k)
    d(i + k) = d(i) - k
Next i
SHUFFLE = sh
Exit Function
37 SHUFFLE = CVErr(xlErrValue)
End Function

```

```

Sub FORMLIST()
Dim i As Integer, here As String, mess1 As String, mess2 As String, shtdif As Boolean
Dim inrng As Object, inrng2 As Object, outrng As Object, topout As Object
Dim doit As Long, cell As Object, calx As Long, ask0 As Boolean, clr0 As Boolean, nm As Object
calx = Application.Calculation
On Error GoTo 6
Application.EnableCancelKey = xlErrorHandler

```

```

here = Selection.AddressLocal
mess1 = "Select the range of cells whose formulas are to be listed."
mess1 = mess1 & Chr(10) & "(FormList 1.5 " & Chr(169) & " 1996-1999 by R. B. Myerson.)"
mess2 = "SELECT RANGE OF FORMULAS TO LIST"
Set inrng = Application.InputBox(prompt:=mess1, Title:=mess2, default:=here, Type:=8)
inrng.Worksheet.Activate
inrng.Select
Set inrng2 = Application.Intersect(inrng, inrng.Worksheet.UsedRange)
mess1 = "Where should the list of formulas be written?" & Chr(10) & "(Keeping the same range does a formula/text toggle.)"
mess2 = "SELECT OUTPUT RANGE"
Set outrng = Application.InputBox(prompt:=mess1, Title:=mess2, default:=inrng.AddressLocal, Type:=8)
Application.Calculation = xlManual
If outrng.AddressLocal = inrng.AddressLocal Then
    mess1 = "In the selected range, formulas (except arrays) will be converted to text beginning '=', and all text beginning '=' will be converted back to formulas."
    mess2 = "FORMULA/TEXT TOGGLE"
    doit = MsgBox(prompt:=mess1, Buttons:=vbOKCancel, Title:=mess2)
    If doit = vbCancel Then GoTo 6
    ask0 = True
    clr0 = True
    For Each cell In inrng2
        If cell.HasFormula Then
            If Not cell.HasArray Then cell.Value = "" & cell.FormulaLocal
        ElseIf Application.IsText(cell) Then
            If Left(cell.Value, 1) = "=" Then
                cell.FormulaLocal = cell.Value
                cell.HorizontalAlignment = xlGeneral
            ElseIf clr0 Then
                If "" = cell.Value Then
                    If ask0 Then
                        doit = MsgBox(prompt:="FORMLIST found cells containing zero-length strings. Is it OK to clear them?", Buttons:=vbYesNo, Title:=mess2)
                        clr0 = (vbYes = doit)
                        If clr0 Then cell.ClearContents
                        ask0 = False
                    Else
                        cell.ClearContents
                    End If
                End If
            End If
        End If
    Next cell
End If

```

```

        End If
    End If
End If
Next cell
Application.Calculation = calx
Exit Sub
End If
Set topout = outrng.Cells(1, 1)
here = topout.AddressLocal(rowabsolute:=False, columnabsolute:=False)
mess1 = "OK to make list of formulas, with top in cell " & here & ", and continuing down the column?"
mess2 = "FORMULA LIST"
doit = MsgBox(prompt:=mess1, Buttons:=vbOKCancel, Title:=mess2)
If doit = vbCancel Then GoTo 6
i = 0
For Each cell In inrng2
    If cell.HasFormula Then
        i = i + 1
        Do While topout.Offset(i, 0).HasFormula
            i = i + 1
        Loop
        If cell.HasArray Then
            mess1 = cell.AddressLocal(rowabsolute:=False, columnabsolute:=False) & " in "
            topout.Offset(i, 0).Value = mess1 & cell.CurrentArray.AddressLocal(rowabsolute:=False, columnabsolute:=False) & ". {" & cell.FormulaLocal & "}"
        Else
            topout.Offset(i, 0).Value = cell.AddressLocal(rowabsolute:=False, columnabsolute:=False) & ". " & cell.FormulaLocal
        End If
    End If
End If
Next cell
If ActiveWorkbook.Names.Count > 0 Then
    doit = MsgBox(prompt:="Do you want to list also the names defined in this workbook?", Buttons:=vbYesNo, Title:="LIST NAMES?")
    If doit = vbYes Then
        For Each nm In ActiveWorkbook.Names
            i = i + 1
            Do While topout.Offset(i, 0).HasFormula
                i = i + 1
            Loop
        End For
    End If
End If

```

```

        topout.Offset(i, 0).Value = "" & nm.Name & ". " & nm.RefersToLocal
    Next nm
End If
End If
Application.Calculation = calx
shtdif = (topout.Worksheet.Name <> inrng2.Worksheet.Name)
topout.Value = "FORMULAS FROM RANGE " & inrng2.AddressLocal(rowabsolute:=False, columnabsolute:=False, external:=shtdif)
topout.Worksheet.Activate
topout.Select
Exit Sub
6 Application.Calculation = calx
MsgBox prompt:="Formulas were not found, or an error occurred.", Title:="FORMULA LIST"
End Sub

Function FORMULAS(auditRange As Object) As Variant
On Error GoTo 7
Dim n As Integer, i As Integer, x() As String, cell As Object
n = auditRange.Cells.Count + 1
ReDim x(1 To n, 1 To 1)
i = 1
For Each cell In auditRange
If cell.HasFormula Then
    If cell.HasArray Then
        x(i, 1) = cell.AddressLocal(rowabsolute:=False, columnabsolute:=False) & ". {" & cell.FormulaLocal & "}"
    Else
        x(i, 1) = cell.AddressLocal(rowabsolute:=False, columnabsolute:=False) & ". " & cell.FormulaLocal
    End If
    i = i + 1
End If
Next cell
If i > 1 Then x(i, 1) = "...FORMULAS FROM RANGE " & auditRange.AddressLocal(rowabsolute:=False, columnabsolute:=False)
FORMULAS = x
Exit Function
7 FORMULAS = CVErr(xlErrValue)
End Function

```

```
Function FORMRC(auditCell As Object)
On Error GoTo 29
Dim refcell As Object
Set refcell = auditCell.Cells(1, 1)
If refcell.HasFormula Then
    FORMRC = refcell.FormulaR1C1Local
    If refcell.HasArray Then FORMRC = "{" & FORMRC & "}"
ElseIf Not IsEmpty(refcell) Then
    FORMRC = refcell.Value
Else FORMRC = ""
End If
Exit Function
29 FORMRC = CVErr(xlErrValue)
End Function
```