# Contents

# VBA

## VBA Function : return array 2 dimentions

```
Function getData() As Variant()
    Dim ar(2, 2) As Variant

    ar(0, 0) = "x"
    ar(0, 1) = 1

    ar(1 0) = "y"
    ar(1, 1) = 2

    getData= ar

End Function
//How to use formula : use like array formula by “Ctrl+Shift+Enter”.
```

## Extended Euclidean algorithm F1

```
Function extGCDof_a_b_to_5_2(a As LongLong, b As LongLong) As Variant()
    Dim ar(5, 2) As Variant
    Dim s As LongLong, old_s As LongLong, t As LongLong, old_t As
LongLong, r As LongLong, old_r As LongLong, Quotient As LongLong, prov
As LongLong
        s = 0
        old_s = 1
        t = 1
        old_t = 0
        r = b
        old_r = a
    While r <> 0
        Quotient = Int(old_r / r)
        '(old_r, r) := (r, old_r - quotient * r)
        prov = r
        r = old_r - Quotient * prov
        old_r = prov
        '(old_s, s) := (s, old_s - quotient * s)
        prov = s
        s = old_s - Quotient * prov
        old_s = prov
        '(old_t, t) := (t, old_t - quotient * t)
        prov = t
        t = old_t - Quotient * prov
        old_t = prov
    Wend

    'output "greatest common divisor:", old_r
    ar(0, 0) = "GCD"
    ar(0, 1) = old_r
    'output "Bézout coefficients:", (old_s, old_t)
    ar(1, 0) = "Bezout coeff of a"
    ar(1, 1) = old_s
    ar(2, 0) = "Bezout coeff of b"
    ar(2, 1) = old_t

    'output "quotients by the gcd:", (t, s)
    ar(3, 0) = "quotients of a"
    ar(3, 1) = t
    ar(4, 0) = "quotients of b"
    ar(4, 1) = s

    extGCDof_a_b_to_5_2 = ar
End Function
```

## Extended Euclidean algorithm F2

```
Function extGCD2of_a_b_to_3_2(a As LongLong, b As LongLong) As Variant()
    Dim ar(3, 2) As Variant
    Dim s As LongLong, old_s As LongLong, r As LongLong, old_r As
LongLong, Quotient As LongLong, prov As LongLong, bezout_t As LongLong
        s = 0
        old_s = 1
        r = b
        old_r = a
    While r <> 0
        Quotient = Int(old_r / r)
        '(old_r, r) := (r, old_r - quotient * r)
        prov = r
        r = old_r - Quotient * prov
        old_r = prov
        '(old_s, s) := (s, old_s - quotient * s)
        prov = s
        s = old_s - Quotient * prov
        old_s = prov
    Wend

    If b <> 0 Then
        bezout_t = Int((old_r - old_s * a) / b)
    Else
        bezout_t = 0
    End If
    'output "greatest common divisor:", old_r
    ar(0, 0) = "GCD"
    ar(0, 1) = old_r
    'output "Bézout coefficients:", (old_s, bezout_t)
    ar(1, 0) = "Bezout coeff of a"
    ar(1, 1) = old_s
    ar(2, 0) = "Bezout coeff of b"
    ar(2, 1) = bezout_t

    extGCD2of_a_b_to_3_2 = ar
End Function
```

## multiplicative inverses in modular

```
Function Mod_mult_inv(a As Long, n As Long)
'at =_ 1 mod n
'https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm#Computing_mu
ltiplicative_inverses_in_modular_structures
Dim t As Long, newt As Long, r As Long, newr As Long, Quotient As Long,
prov As Long
    t = 0
    newt = 1
    r = n
    newr = a

    While newr <> 0
        Quotient = Int(r / newr)
        '(t, newt) = (newt, t - quotient * newt)
        prov = newt
        newt = t - Quotient * prov
        t = prov
        '(r, newr) = (newr, r - quotient * newr)
        prov = newr
        newr = r - Quotient * prov
        r = prov
    Wend
    If r > 1 Then
        'return "a is not invertible"
        t = -9999999
    End If
    If t < 0 Then
        t = t + n
    End If

    Mod_mult_inv = t
End Function
```

## Use Loop with Range vba

```
‘https://stackoverflow.com/questions/18168151/vba-pass-a-group-of-cells-as-range
-to-function
Function myAdd(Arg1 As Range, ParamArray Args2() As Variant) As Double
     Dim elem As Variant
     Dim i As Long
     For Each elem In Arg1
```

```
        myAdd = myAdd + elem.Value
      Next elem
      For i = LBound(Args2) To UBound(Args2)
           For Each elem In Args2(i)
                myAdd = myAdd + elem.Value
           Next elem
      Next i
End Function

‘----------------------------------------------------------------
by zam007
Function test(R As Range) As Double
     Dim n As Double, sum As Double
        n = R.Count
     Dim i As Long
     For i = 1 To n
        sum = sum + R(i, 1)  ‘sum of column range
     Next i

     test = sum
End Function
```

## Kernel Bandwidth Selection

‘https://www.di.ubi.pt/~lfbaa/pubs/tecrep2008.pdf

```
Function Find_h_sortedData(R As Range, e As Double) As Double
    'R.Sort order1:=xlAscending, Header:=xlNo
    Dim n As Double
        n = R.Count
    Dim i As Long, j As Long, k4 As Double, h0 As Double, h1 As Double,
S1 As Double, sum As Double  ', e As Double

    'e = 0.00001   '0.001
     '=0.9*POWER(5,-1/5)*MIN(
(QUARTILE.EXC(C6:C10,3)-QUARTILE.EXC(C6:C10,1))/1.34,STDEV.S(C6:C10))
    S1 = 0.9 * (n ^ (-0.2)) *
Application.WorksheetFunction.Min((Application.WorksheetFunction.Quartil
e_Exc(R, 3) - Application.WorksheetFunction.Quartile_Exc(R, 1)) / 1.34,
Application.WorksheetFunction.StDev_S(R))
    h0 = S1
    h1 = h0 + e
    While Abs(h1 - h0) > e

        h0 = h1
        '+++++++++++++++++++++++++++++++++

            sum = 0
            For i = 1 To n
                For j = 1 To i - 1
                    sum = sum + (((R(i, 1) - R(j, 1)) ^ 2 - 6 * h0 ^ 2)
^ 2 - 24 * h0 ^ 4) * (Exp(-((R(i, 1) - R(j, 1)) / 2 / h0) ^ 2))
                Next j
            Next i

        k4 = 3 * n * h0 + (1 / 2 / (h0 ^ 3)) * sum
        '**************************************
        h1 = ((4 * n * (h0 ^ 6)) / k4) ^ 0.2
        '//////////////////////////////////////

        '----------------
        h1 = (h0 + h1) / 2

    Wend

    Find_h_sortedData = h1
End Function

‘+++++++++++++++++++++++++++++
‘--------- V 2 ———————-
Function Find_h_sortedData(R As Range, e As Double, itrMax As Long) As
Double
    'R.Sort order1:=xlAscending, Header:=xlNo
    Dim n As Double
        n = R.Count
    Dim ifor As Long, i As Long, j As Long, k4 As Double, h0 As Double,
h1 As Double, S1 As Double, sum As Double ', e As Double

    'e = 0.00001   '0.001
     '=0.9*POWER(5,-1/5)*MIN(
(QUARTILE.EXC(C6:C10,3)-QUARTILE.EXC(C6:C10,1))/1.34,STDEV.S(C6:C10))
    S1 = 0.9 * (n ^ (-0.2)) *
Application.WorksheetFunction.Min((Application.WorksheetFunction.Quartil
e_Exc(R, 3) - Application.WorksheetFunction.Quartile_Exc(R, 1)) / 1.34,
Application.WorksheetFunction.StDev_S(R))
    h0 = S1
    h1 = h0 + e
    sum = 0
    'While Abs(h1 - h0) >= e  'old is > ...new is >=  '' Abs(h1 - h0) >=
e
    For ifor = 1 To itrMax
        h0 = h1
        '+++++++++++++++++++++++++++++++++

            sum = 0
            For i = 1 To n
                For j = 1 To i - 1
                    sum = sum + (((R(i, 1) - R(j, 1)) ^ 2 - 6 * h0 ^ 2)
^ 2 - 24 * h0 ^ 4) * (Exp(-((R(i, 1) - R(j, 1)) / 2 / h0) ^ 2))
                Next j
            Next i

        k4 = 3 * n * h0 + (1 / 2 / (h0 ^ 3)) * sum
        '**************************************
        h1 = ((4 * n * (h0 ^ 6)) / k4) ^ 0.2
        '//////////////////////////////////////

        '----------------
        h1 = (h0 + h1) / 2    ' not enable is ok too.

        'newwwwwww
        If (Abs(h1 - h0) <= e) Then Exit For



    Next ifor   'wend

    Find_h_sortedData = h1
End Function
```

# Excel Formula

## Quantile function of   Standard normal distribution

```
=10*LOG( 1 - LOG( -LOG(A1,2),22),41)
=10/LOG10(41)*LOG10(1-LOG10(-LOG10(A1)/LOG10(2))/LOG10(22))
```

## Zam switch function

```
=MOD(POWER(2,CEILING.MATH(LOG10(ABS(A1)+9))-1),2)
***answer iis  0,1
```

https://script.google.com/macros/s/AKfycbzw2jpiPunqe-Miwd6dlbNZsXqidQi-uKA7mHWL/exec