**Programming Col Sig (means): Classical Student Test (with and without estimator)**

In this article we will look at programming colsig for means.

* Classical Student Test
* Classical Student Test (using estimator)

The .qes file and portfolio containing the examples discussed is attached: Table Arithmetic Examples 15.rar.



For each test I will show how to set the syntax up for:

1. a table that shows regular means - i.e. ‘Mean’ calculation at the bottom of a scaled rating question in the rows

See tab definitions: *ColSigTests - Regular Means Tab (95%)*

1. a mean summary table - i.e. ‘Script (mean)’ calculation which references a scaled question to be filtered by the rows and columns of the tab definition



See tab definitions: *ColSigTests - Means Summary Tab (90%)*



**Classical Student Test**

Let’s refresh ourselves on the formula for this:





Since the formula references the standard deviation, we’ll need to add a calculation for this as well.



Then in (# Col Sig Num) we add the script:

Dim numX1 = CurrentTable.GetCell(CurrentCell.X,CurrentCell.y-3).Value

Dim numSD1 = CurrentTable.GetCell(CurrentCell.X,CurrentCell.y-2).Value

Dim dN1 = CurrentCell.Row.Question.Data.FilterByX().Size()

Dim DA1 = ( numSD1 \* numSD1 ) / dN1

Dim StrColSig = ""

Dim i

 For i = CurrentTable.StartX to CurrentTable.MaxX

 Dim numX2 = CurrentTable.GetCell(i,CurrentCell.y-3).Value

 Dim numSD2 = CurrentTable.GetCell(i,CurrentCell.y-2).Value

 Dim dN2 = CurrentCell.Row.Question.Data.FilterByX(i).Size()

 Dim DA2 = ( numSD2 \* numSD2 ) / dN2

 Dim numNumerator = numX1 - numX2

 Dim numDenom = pow(( DA1 + DA2 ),0.5)

 Dim tValue = numNumerator / numDenom

 If abs(tValue) > 1.96 Then

 StrColSig = StrColSig + CurrentTable.GetColSigLetter(i)

 End if

 Next i

Return StrColSig

The points to note here are:

Dim numX1 = CurrentTable.GetCell(CurrentCell.X,CurrentCell.y-3).Value

 This grabs the value from the calculation 3 cells above the current calculation’s cell - Mean

Dim numSD1 = CurrentTable.GetCell(CurrentCell.X,CurrentCell.y-2).Value

This grabs the value from the calculation 2 cells above the current calculation’s cell - Sd

Dim dN1 = CurrentCell.Row.Question.Data.FilterByX().Size()

This grabs the base – we could use the previous syntax replacing the Y co-ordinate reference with: CurrentCell.y-4 but it’s preferable to use ‘FilterBy’ keyword as it’s more dynamic (doesn’t require update if calculations added or moved).

You’ll note that we have defined numX1, numSD1 and dN1 all outside of a loop and then defined their counterparts numX2, numSD2 and dN2 inside of a loop (iterates through each data column). So each column (1) is tested against all other columns (2) in turn.

For the mean summary table we need to add:



You’ll note for the script in this tab definition we use: (If abs(tValue) > 1.645 Then) the instead of: (If abs(tValue) > 1.96 Then). That’s because this test is done at the 90% confidence level rather than 95%.

**Classical Student Test (using estimator)**

The formula for this:

****



The tables look like this:



And the script in *# ColSig (Est)*:

Dim numX1 = CurrentTable.GetCell(CurrentCell.X,CurrentCell.y-3).Value

Dim numSD1 = CurrentTable.GetCell(CurrentCell.X,CurrentCell.y-2).Value

Dim dN1 = CurrentCell.Row.Question.Data.FilterByX().Size()

Dim numDenPartSD1 = (numSD1 \* numSD1)\*(dN1 - 1)

Dim StrColSig = ""

Dim i

 For i = CurrentTable.StartX to CurrentTable.MaxX

 Dim numX2 = CurrentTable.GetCell(i,CurrentCell.y-3).Value

 Dim numSD2 = CurrentTable.GetCell(i,CurrentCell.y-2).Value

 Dim dN2 = CurrentCell.Row.Question.Data.FilterByX(i).Size()

 Dim numDenPartSD2 = (numSD2 \* numSD2)\*(dN2 - 1)

 Dim numDenPartNa = (dN1 + dN2 - 2)

 Dim numDenPartNb = ((1/dN1) + (1/dN2))

 Dim numNumerator = numX1 - numX2

 Dim numDenom = pow(((numDenPartSD1 + numDenPartSD2) \* numDenPartNb / numDenPartNa),0.5)

 Dim tValue = numNumerator / numDenom

 If abs(tValue) > 1.96 Then

 StrColSig = StrColSig + CurrentTable.GetColSigLetter(i)

 End if

 Next i

Return StrColSig