overview of charting

Charting will be a major chapter in this book since it's the focus of a lot of what I do. It also is a spot where a significant amount of time can be saved since the chart options are awful to mess with.

Some high level topics:

- Creating a chart
- Formatting a chart
- Manipulating the series on a chart
- Changing the layout of charts on a page (make grid code)
- Common patterns when working through charts (`ForEach` loops wherever possible)

introduction to charting

[handwritten note: ALSO EXCEL HAS AN AWFUL UI.]

Charting is the second most important aspect of automatic Excel behind manipulating `Ranges`. There is a bias when saying that because a lot of what I do after engineering calculations is chart the results. In particular, Excel can be used to great effect to chart time series of data. THe other reason charts are so amenable to VBA is that very often you are applying the same actions to the charts. In that sense, the VBA related to charts is doing a lot of changing settings and formats so that the charts look the way you want. This ahs the immediate effect of making your charts look less like "they came from Excel" which si a common knock in some circles.

When working with `Charts`, there is a `Range` of difficulties depending on what you are trying to do. In some cases, working with an existing `chart` is much easier than creating a new one. In other instances, it can be much simpler to create a new chart rather, starting from a default, rather than change all the settings back. One other major difference between `Charts` and `Ranges` is that working with charts is much more about knowing the object model than knowing how to program. The vast majority of your code related to charts is simple iterating through objects to find the one property that you want to change. This makes it easier to write chart VBA once you have the basics of `For Each` loops down. It also means that you need to spend some time getting comfortable with the object model.

There is one oddity related to Charts that is worth mentioning now. Charts can either be embedded as an object on a `Worksheet`, or they can be their own `Sheets`. I personally never use the latter case, but it is common enough that it needs to be on your mind when working with Charting code.

(I don't use the Chart as a Sheet model because I find that it is not necessary in terms of displaying data. In particular, you are at the mercy of your window size and cannot easily change the dimensions. Also, it

complicates the VBA side of things to work in both formats all the time, so I just decided to always put my Charts on Sheets. Your mileage may vary so I'll touch on both approaches in the code samples.)

a quick overview of the object model

- ChartObjects -> ChartObject - this derives from Shape and exists when the Chart is on a Worksheet
    - Chart
        * SeriesCollection -> Series
        * Axes -> Axis
        * ChartArea
        * PlotArea
- ActiveChart -> Chart - this works whether you have a Worksheet or Chart on a sheet
- Selection -> Variant - this one can be useful but is often not of the type that you want.

obtaining a reference to a Chart

When working with Charts, the first task is typically to get a reference to an existing chart – unless you are creating a new chart. To obtain a reference to a chart, there are a handful of ways of doing it depending on what your spreadsheet contains and how it's structured.

THe main ways to do it are:

- Use the `ActiveChart` object
- Use the `Selection` object – this is highly depending on what is selected
- Use the `ChartObjects` object
    - If you know which chart you want, you can supply an index; this works great if there is only a single chart - `ChartObjects(1)`
    - If you want to do something to all charts, you can iterate this object
    - If you have named the chart (more on that later) you can supply the name as the index - `ChartObjects("SomeChart")`
- The `Workbook.Sheets` object if your charts are contained in their own sheets
    - Same as above, you can access via a numeric index, name, or iterate through all of them

ActiveChart `ActiveChart` is similar to the other `Active` objects in that it does about what you expect. The one difference is that the Chart actually has to be selected or have focus in order to be

considered "active". This is similar but also different to something like `ActiveWorkbook` where having the workbook open makes it active.

Note that ActiveChart will work for a Chart that is contained on a Worksheet or also for one that is its own Sheet. If the latter case, then ActiveSheet and ActiveChart will refer to the same object. Side note: this technicality is why you will not get proper Intellisense when using ActiveSheet – that Sheet could technically be a Chart.

The nice thing about ActiveChart is that it gives you the Chart object which then gives you immediate access to the Chart related details you are like to want to change. The downside is that unless you have a single Chart that is already selected, ActiveChart has limited application when using VBA. Again, the goal is to avoid selecting objects in order to access them via VBA so ActiveChart has this limitation.

Selection   The Selection object is probably the greatest catch all for an object. It literally holds anything, and this means that using the object requires knowing what is selected, or checking vigorously before using the object. Technically, you also let your code error out if the wrong object is selected, and this works well at times. This works well because you are unlikely to be using Selection in a complicated workflow because, again, you should not be selecting objects to access them. This means that Selection is really limited to one-off and helper code where you can more tightly dictate that this code only works if you select a Chart. You should still add some error handling, but sometimes that step is skipped.

Since the Selection can hold anything, it's important to know what could be Selected. Related to charts, the following can all live in the Selection:

- ChartObjects
- Chart
- ChartArea
- PlotArea
- Legend
- ChartTitle
- Series

If you are writing VBA to work on Charts, you can technically require the user to select the correct part of the chart and always use `Selection`. You will quickly grow tired of having to remember which part of the Chart to select in order to make the code work. To avoid this scenario, it is helpful to remember the object model and know how to work your way around a Chart.

My approach has always been to convert the Selection to a Collection of ChartObjects. I can then always

iterate that resulting Collection to process the Charts. If only a single Chart was selected, the code works all the same. The downside to this approach is that a Chart as a Sheet cannot live inside a ChartObject. This is a large part of why I always put Charts on a Worksheet.

Below is the helper function I use in order to convert a possibly Chart containing selection into a Collection of ChartObjects. It works for all objects except for the Axis related ones.

TODO: consider improving this code if it is included as a de facto reference

```vba
Public Function Chart_GetObjectsFromObject(ByVal inputObject As Object) As
    Variant

    Dim chartObjectCollection As New Collection

    'NOTE that this function does not work well with Axis objects.  Excel
        does not return the correct Parent for them.

    Dim targetObject As Variant
    Dim inputObjectType As String
    inputObjectType = TypeName(inputObject)

    Select Case inputObjectType

        Case "DrawingObjects"
            'this means that multiple charts are selected
            For Each targetObject In inputObject
                If TypeName(targetObject) = "ChartObject" Then
                    'add it to the set
                    chartObjectCollection.Add targetObject
                End If
            Next targetObject

        Case "Worksheet"
            For Each targetObject In inputObject.ChartObjects
                chartObjectCollection.Add targetObject
            Next targetObject

        Case "Chart"
            chartObjectCollection.Add inputObject.Parent
```

```
30          Case "ChartArea", "PlotArea", "Legend", "ChartTitle"
31              'parent is the chart, parent of that is the chart targetObject
32              chartObjectCollection.Add inputObject.Parent.Parent
33
34          Case "Series"
35              'need to go up three levels
36              chartObjectCollection.Add inputObject.Parent.Parent.Parent
37
38          Case "Axis", "Gridlines", "AxisTitle"
39              'these are the oddly unsupported objects
40              MsgBox "Axis/gridline selection not supported.  This is an Excel
                  bug.  Select another element on the chart(s)."
41
42          Case Else
43              MsgBox "Select a part of the chart(s), except an axis."
44
45      End Select
46
47      Set Chart_GetObjectsFromObject = chartObjectCollection
48  End Function
```

ChartObjects   If you are working on a Worksheet, then that Worksheet will have the ChartObjects object. This object is great because it contains all of the Charts in their own collection (separate from any other Shapes or buttons). This ChartObjects collection contains object of type ChartObject. The ChartObject derives from Shape which means it contains all of the properties related to on-sheet position and size.

A typical workflow is included below since it is a pattern that shows up all the time in VBA code related to charts. At a high level the steps are:

- Use ActiveSheet or a Worksheet reference to access the ChartObjects
- Iterate through each ChartObject, storing a reference to the underlying Chart
- You then setup sections to work through the parts of the Chart you want
  - Iterate through the SeriesCollection
  - Iterate through the Axes
  - Touch the other top level properties including ChartTile, Legend, etc.

This workflow is quite powerful because it can quickly be wrapped with a loop to go through all Worksheets and even possible all Workbooks. It's also powerful because you can be quite comfortable learning this

pattern and then adding in the parts that you actually want to change. The only downside is that it can be quite tedious to type out all the loops every time, but there's not a good way around that other than to use the clipboard.

Another approach to using ChartObjects is to not iterate through all of them but instead to select a single ChartObject and work with it. There are two ways to do this:

- Use an integer index for the Chart – this is quite easy to do if there are only a few charts
- Name the chart and use that name

When using either of these approaches, it is quite helpful to show the `Selection Pane` window in Excel. This pane will pop out and tell you the order and the names of all the objects on the sheet (this includes comments, shapes, and Charts). From this pane, you can rearrange the charts into the order you want or rename them.

Although `For Each` loops are generally preferred when working with Charts, sometimes you simply know that you want to change one chart and an index just lets you do that. If you are in the habit of using loops however, you can easily do that with the helper code included above which stick a single chart into a Collection.

Workbook.Sheets to get Chart references   The final approach to obtaining a Chart reference is to use the Sheets object. Aside from ActiveChart, this is the only way to deal with Charts that are their own Sheet. Again, you can either use an index or a Name. Here, the Name is easily changed on the Sheet tab so it's much more common to use a Name when doing this. The other approach is to iterate through all the Sheets and pick off the ones that are Charts.

There are two key points when working with Charts as Sheets:

- You must use the Workbook.Sheets object to access them and not Workbook.Worksheets. The latter object contains only those Worksheets that are not Charts. The former contains both Charts and Worksheets.
- It's possible that your Sheet is not actually a Chart. You should check the type of the object is you are going to iterate through all Worksheets. Also be aware that some sheets can be hidden which might lead to unexpected results.

TODO: is there a Charts object on Workbook?

common objects/properties for a Chart

This section will focus on the common formatting changes that can be made to a Chart. the next section focuses on creating a Chart from scratch if you want to see that. These common changes will be grouped by the type that they affect, but this is not meant to be an exhaustive list. Instead, this is a list that will cover the objects nad functions that are actually used in regular code. There will be several other things that you will need to check the reference for (or record a macro), but this listing will get you started with the regular things.

To organize this section, we will focus on the different parts of a Chart in turn along with how to access the things you need. This section is meant to be a one stop shop for working on the common parts of a Chart. This will cover:

- ChartObject
  - Top, Left, Height, Width - control the location of a chart
- Chart
  - ChartType
  - Access the other objects and controls whether some things exist
    * HasLegend
    * HasTitle
- Legend
- Series – accessed the the Chart.SeriesCollection
  - ChartType
- Axis – accessed through Chart.Axes
  - Display the axis
  - Change the text
  - Change the min/max scale including automatic values
  - Change the number format of the axis
  - Change the format and display of the Gridlines
- Point – accessed through a Series
  - Change display of individual points
  - Control the DataLabels (HasLabel and then DataLabel)
- Trendline

TODO: go through bUTL and find other commonly appearing things

common changes to the ChartObject

The ChartObject is the main container for a Chart that is on a Worksheet. The common changes then are related to the position and size of the Chart on the Worksheet. The common properties to change here are:

- Top
- Left
- Height
- Width
- Placement (controls the move with cells option)

All of these are of type Double which means you can use decimal calculations to determine the size or position. In Excel, the 0,0 point is at the upper left hand corner (upper left of cell A1) and the Top and Left increase going to the right and down. If you are familiar with 0,0 being the center of the XY plane, then Excel will be a tad unfamiliar. Once you get used to it, you will realize that there is not really a better way to arrange the coordinate system since the spreadsheet can extend to the right and down nearly infinitely.

TODO: are there Bottom and Right properties too?

TODO: add a comment about Points vs. inches here and the function to convert them

The most common application of changing these properties is to either standardize the size of several charts or to arrange the charts in a grid (which standardizes the size and then position).

That code is included below:

TODO: clean up this code to only the required parts

```
Public Sub Chart_GridOfCharts( _
    Optional columnCount As Long = 3, _
    Optional chartWidth As Double = 400, _
    Optional chartHeight As Double = 300, _
    Optional offsetVertical As Double = 80, _
    Optional offsetHorizontal As Double = 40, _
    Optional shouldFillDownFirst As Boolean = False, _
    Optional shouldZoomOnGrid As Boolean = False)

    Dim targetObject As ChartObject

    Dim targetSheet As Worksheet
    Set targetSheet = ActiveSheet

```

```vba
15      Application.ScreenUpdating = False

16

17      Dim countOfCharts As Long
18      countOfCharts = 0

19

20      For Each targetObject In targetSheet.ChartObjects
21          Dim left As Double, top As Double

22

23          If shouldFillDownFirst Then
24              left = (countOfCharts \ columnCount) * chartWidth +
                    offsetHorizontal
25              top = (countOfCharts Mod columnCount) * chartHeight +
                    offsetVertical
26          Else
27              left = (countOfCharts Mod columnCount) * chartWidth +
                    offsetHorizontal
28              top = (countOfCharts \ columnCount) * chartHeight +
                    offsetVertical
29          End If

30

31          targetObject.top = top
32          targetObject.left = left
33          targetObject.Width = chartWidth
34          targetObject.Height = chartHeight

35

36          countOfCharts = countOfCharts + 1

37

38      Next targetObject

39

40      'loop through columns to find how far to zoom
41      'Cells.Left property returns a variant in points
42      If shouldZoomOnGrid Then
43          Dim columnToZoomTo As Long
44          columnToZoomTo = 1
45          Do While targetSheet.Cells(1, columnToZoomTo).left < columnCount *
                chartWidth
46              columnToZoomTo = columnToZoomTo + 1
47          Loop

48
```

```
49        targetSheet.Range("A:A", targetSheet.Cells(1, columnToZoomTo - 1).
              EntireColumn).Select
50        ActiveWindow.Zoom = True
51        targetSheet.Range("A1").Select
52     End If
53
54     Application.ScreenUpdating = True
55
56  End Sub
```

common properties of the Chart

The Chart object is mostly a container for the other more useful properties of the Chart, but there are a couple of common properties that live at this top level. Those include:

- The HasXXX: HasTitle, HasLegend (TODO: any others?) - control the display of these things
- ChartType
- Delete
- Copy (TODO: this on ChartObject also?)

TODO: find more of these

In addition to those properties, the Chart object provides access to other useful things via the common accessors:

- SeriesCollection
- Axes
- Legend
- ChartTitle
- ChartArea
- PlotArea

TODO: is this list complete?

common properties of the Series

One of the two most used Chart objects is the Series (other is the Axis). The Series ends up being powerful because it provides access to the data of the Chart along with the major formatting choices since the Series

is the prominent feature of a Chart.

The common things to go after for a series are:

- Data related
    - Name
    - XValues
    - Values
    - Formula
- Formatting related
    - Format
        * Line
    - MarkerSize
    - MarkerStyle
    - MarkerForegroundColor, MarkerBackgroundColor

Also, from a Series you can access the following other objects:

- Points
- Trendlines

common properties of the Axis

The Axis is the second most common object to work with (behind the Series). This is largely because the Axis controls or provides access to a lot of the formatting related aspects of the Chart. The Axis also controls the scale of the Axis and in that regard, is a critical part of making or editing a Chart.

The first part of the Axis is accessing the correct one. This is slightly tricky the first time because the Axes are stored in the Chart.Axes object. THe real trick is that this object is indexed by the xlAxisType (TODO: check that) which can be xlCategory (for the x-axis) or xlValue/xlValue2 (for the y-axis, left and right).

Once you have an Axis object, you can set to work changing the common properties:

- Scale related
    - MinimumScale/MaximumScale
    - MinimumScaleIsAuto/MaximumScaleIsAuto
- Formatting related (most of these are accessors to a different object)
    - GridLines (Major/minor and the HasXXX)
    - Ticks (TODO: that right?)

– HasTitle and AxisTitle

Chart_Axis_AutoX.md
```
1  Public Sub Chart_Axis_AutoX()
2
3      Dim targetObject As ChartObject
4      For Each targetObject In Chart_GetObjectsFromObject(Selection)
5          Dim targetChart As Chart
6          Set targetChart = targetObject.Chart
7
8          Dim xAxis As Axis
9          Set xAxis = targetChart.Axes(xlCategory)
10         xAxis.MaximumScaleIsAuto = True
11         xAxis.MinimumScaleIsAuto = True
12         xAxis.MajorUnitIsAuto = True
13         xAxis.MinorUnitIsAuto = True
14
15     Next targetObject
16
17 End Sub
```

Chart_Axis_AutoY.md
```
1  Public Sub Chart_Axis_AutoY()
2
3      Dim targetObject As ChartObject
4      For Each targetObject In Chart_GetObjectsFromObject(Selection)
5          Dim targetChart As Chart
6          Set targetChart = targetObject.Chart
7
8          Dim yAxis As Axis
9          Set yAxis = targetChart.Axes(xlValue)
10         yAxis.MaximumScaleIsAuto = True
11         yAxis.MinimumScaleIsAuto = True
12         yAxis.MajorUnitIsAuto = True
13         yAxis.MinorUnitIsAuto = True
14
15     Next targetObject
16
17 End Sub
```

Chart_AxisTitleIsSeriesTitle.md

```vba
Public Sub Chart_AxisTitleIsSeriesTitle()

    Dim targetObject As ChartObject
    Dim targetChart As Chart
    For Each targetObject In Chart_GetObjectsFromObject(Selection)
        Set targetChart = targetObject.Chart

        Dim butlSeries As bUTLChartSeries
        Dim targetSeries As series

        For Each targetSeries In targetChart.SeriesCollection
            Set butlSeries = New bUTLChartSeries
            butlSeries.UpdateFromChartSeries targetSeries

            targetChart.Axes(xlValue, targetSeries.AxisGroup).HasTitle = True
            targetChart.Axes(xlValue, targetSeries.AxisGroup).AxisTitle.Text
                = butlSeries.name

            '2015 11 11, adds the x-title assuming that the name is one cell
                above the data
            '2015 12 14, add a check to ensure that the XValue exists
            If Not butlSeries.XValues Is Nothing Then
                targetChart.Axes(xlCategory).HasTitle = True
                targetChart.Axes(xlCategory).AxisTitle.Text = butlSeries.
                    XValues.Cells(1, 1).Offset(-1).Value
            End If

        Next targetSeries
    Next targetObject
End Sub
```

Chart_FitAxisToMaxAndMin.md

```vba
Public Sub Chart_FitAxisToMaxAndMin(ByVal axisType As XlAxisType)

    Dim targetObject As ChartObject
```

```
4    For Each targetObject In Chart_GetObjectsFromObject(Selection)
5        '2015 11 09 moved first inside loop so that it works for multiple
             charts
6        Dim isFirst As Boolean
7        isFirst = True
8
9        Dim targetChart As Chart
10       Set targetChart = targetObject.Chart
11
12       Dim targetSeries As series
13       For Each targetSeries In targetChart.SeriesCollection
14
15           Dim minSeriesValue As Double
16           Dim maxSeriesValue As Double
17
18           If axisType = xlCategory Then
19
20               minSeriesValue = Application.Min(targetSeries.XValues)
21               maxSeriesValue = Application.Max(targetSeries.XValues)
22
23           ElseIf axisType = xlValue Then
24
25               minSeriesValue = Application.Min(targetSeries.Values)
26               maxSeriesValue = Application.Max(targetSeries.Values)
27
28           End If
29
30           Dim targetAxis As Axis
31           Set targetAxis = targetChart.Axes(axisType)
32
33           Dim isNewMax As Boolean, isNewMin As Boolean
34           isNewMax = maxSeriesValue > targetAxis.MaximumScale
35           isNewMin = minSeriesValue < targetAxis.MinimumScale
36
37           If isFirst Or isNewMin Then targetAxis.MinimumScale =
                 minSeriesValue
38           If isFirst Or isNewMax Then targetAxis.MaximumScale =
                 maxSeriesValue
39
```

```
40            isFirst = False
41        Next targetSeries
42    Next targetObject
43
44 End Sub
```

Chart_YAxisRangeWithAvgAndStdev.md

```
1 Public Sub Chart_YAxisRangeWithAvgAndStdev()
2
3     Dim numberOfStdDevs As Double
4
5     numberOfStdDevs = CDbl(InputBox("How many standard deviations to include?
        "))
6
7     Dim targetObject As ChartObject
8
9     For Each targetObject In Chart_GetObjectsFromObject(Selection)
10
11        Dim targetSeries As series
12        Set targetSeries = targetObject.Chart.SeriesCollection(1)
13
14        Dim avgSeriesValue As Double
15        Dim stdSeriesValue As Double
16
17        avgSeriesValue = WorksheetFunction.Average(targetSeries.Values)
18        stdSeriesValue = WorksheetFunction.StDev(targetSeries.Values)
19
20        targetObject.Chart.Axes(xlValue).MinimumScale = avgSeriesValue -
            stdSeriesValue * numberOfStdDevs
21        targetObject.Chart.Axes(xlValue).MaximumScale = avgSeriesValue +
            stdSeriesValue * numberOfStdDevs
22
23    Next
24
25 End Sub
```

common properties of the Legend

The Legend is a simple affair compared to the others. There really only two things to do with it: remove it or move it. Both of these are simple enough:

- HasLegend (on the Chart)
- Delete
- Position

TODO: add an example of these in action

common properties of a Point

The Point represents the lowest level when it comes to how the data and formatting of a Chart is built. In general, you do not have to actively go editing Points. This is because you will typically edit the appearance of the Series and the Axes to get the Chart that you want. There are however times when you get down to the metal and edit the properties of the individual points. Before describing how to do this, it may help to give an example or two for why you want to get down to this level:

- Delete a data point without touching the Series
- Add a DataLabel to the point if the value is below some threshold (or if some other Range has a value)
- Hide a Point from one series because you want it to show up in another one

Of the tasks above, only one of them (the second) has to be accomplished via the Points. The others *could* be done via a different method, but you might find yourself in a spot where iterating some Points will save a ton of headache elsewhere. A cautionary note is that typically you should not be editing the properties of a Point; there is nearly always a better way to do these things. Part of the problem is that the settings you change will be quickly overwritten by changes in Excel or VBA. If you know you just need something done however, Points can be a quick way to make it happen.

TODO: look into ErrorBars here?

WHen thinking about working through the Points of a Series, consider the common properties you can change:

- HasLabel / DataLabel
- Value
- Formatting? (TODO: what are these)
- Hidden

TODO: finish this list

Note that in addition to the common properties, you can also change anything that can be changed from the normal Excel settings/properties window.

common properties of the TrendLine

The TrendLine is one of the lesser used properties, but it can be a real time saver when using VBA if you need to. The problem with the trendline normally is that you are required to work through a ton of menus to configure the properties. This is even more painful when you've got to do the same thing to multiple Series in a Chart or across multiple Charts. Similar to the other objects here, you can use VBA to quickly do the task that is otherwise a pain.

The most likely properties you'll use:

- Creating one off of a series
- Type
- Parameter

TODO: confirm these are correct

TODO: add an example showing how to add a Trendline for every Series

Chart_AddTrendlineToSeriesAndColor.md

```
1  Public Sub Chart_AddTrendlineToSeriesAndColor()
2
3      Dim targetObject As ChartObject
4
5      For Each targetObject In Chart_GetObjectsFromObject(Selection)
6          Dim chartIndex As Long
7          chartIndex = 1
8
9          Dim targetSeries As series
10         For Each targetSeries In targetObject.Chart.SeriesCollection
11
12             Dim butlSeries As New bUTLChartSeries
13             butlSeries.UpdateFromChartSeries targetSeries
14
15             'clear out old ones
16             Dim j As Long
```

```
17            For j = 1 To targetSeries.Trendlines.Count
18                targetSeries.Trendlines(j).Delete
19            Next j
20
21            targetSeries.MarkerBackgroundColor = Chart_GetColor(chartIndex)
22
23            Dim newTrendline As Trendline
24            Set newTrendline = targetSeries.Trendlines.Add()
25            newTrendline.Type = xlLinear
26            newTrendline.Border.Color = targetSeries.MarkerBackgroundColor
27
28            '2015 11 06 test to avoid error without name
29            '2015 12 07 dealing with multi-cell Names
30            'TODO: handle if the name is not a range also
31            If Not butlSeries.name Is Nothing Then
32                newTrendline.name = butlSeries.name.Cells(1, 1).Value
33            End If
34
35            newTrendline.DisplayEquation = True
36            newTrendline.DisplayRSquared = True
37            newTrendline.DataLabel.Format.TextFrame2.TextRange.Font.Fill.
                  ForeColor.RGB = Chart_GetColor(chartIndex)
38
39            chartIndex = chartIndex + 1
40        Next targetSeries
41
42    Next targetObject
43 End Sub
```

creating charts from scratch

The previous section discussed how to work with existing Charts. This section will focus on how to create those Charts from scratch if you are coming into a blank Worksheet or if you simply need to add a chart to existing data. At the start, it's worth mentioning that creating Charts from scratch falls into one of two categories:

- Library/helper type code where you want to quickly create a Chart in a common way. This type of code works best in an addin and typically provides functionality that you wish Excel had from the

start

- One-off code for a specific application. This involves creating a Chart with some sort of odd manipulation or formatting or other detail where automation saves time.

The two types of category will end up with code that looks similar, but the goals of the former category will be slightly different than the latter. Typically when making code for a one-off application, you can make more assumptions about how the data is structured and what sorts of actions need to be taken. When working with helper code, you will spend more time asking for user input, and handling the different cases that might come up.

Another key point to make is that the type of work that is being done in a chart can vary as well. The splitting line here is whether the Chart creation is data heavy or formatting heavy (or possibly both). For a data heavy Chart, you will spend a lot of time collecting Ranges, creating Series, and possibly manipulating individual Points. For a formatting heavy chart, you will spend a lot of time iterating through the Series to apply formatting, label the Axes, set the number formats, and generally modify the Excel defaults. Both of these tasks are very time intensive if you are doing them without VBA, so both lend themselves to being automated if possible.

Excel provides two means of creating a Chart depending on how you want to handle things. Those two commands are:

- ChartObjects.Add
- TODO: what is the other method

I always prefer to use ChartObjects.Add because of it consistent application. The other approach tends to put you at the mercy of how Excel interprets your data and its layout.

TODO: add more detail here

The general process for creating a chart looks like this:

- Create a new ChartObject via ChartObjects.Add - store that reference
    - If you know where you want the Chart to go, you can use that information here
- Access the Chart of that object
- Change the properties of the Chart that you know – namely ChartType
- Access the SeriesCollection of the Chart and call NewSeries for each Series you want - store a reference to that Series
    - This is typically done inside a loop that is iterating through Ranges in some way
    - If you need to apply Series specific formatting, do that here
- Access the Axes collection and modify any specific parts of the Axes that you want

- This may show up in the loop above if you want the Axis to draw information from the Series (maybe set the max to the max of the data?)

At this point, you will have a Chart with the Series you want along with the major formatting taken care of. Even better, this general framework lends itself nicely to adding new commands where needed. If you need to go after some of the finer details of the Chart, you can add those commands where the objects are being reference, or at the end of the code. The main thing to consider is whether you need to work inside loops (per Series) or if you can process the extra stuff at the end.

The other upside of this approach is that you can quickly wrap all of this code with another loop to create multiple Charts. You can then wrap that code with another loop to do it on multiple Worksheets, etc. When you write code that can cleanly live inside a loop, you make it easy to use the code elsewhere.

One other aspect of Charts that is somewhat unique is that you can typically reuse a lot of the code by creating new Subs. These can be called from the inside of a loop to create a chain of commands to process a Chart. This approach is highly effective if you work in an environment where the same or similar things need to be done. For example: you have a monthly report to create each month for multiple departments. Standardizing as much of that work into modules makes it easy to apply the code in multiple spots with minor changes. This is relevant to Charts because most of the work of Charts is changing the values of specific properties. There is typically far less logic that is unique to an application (like trying to build a Range based on the layout of data).

Once you have this general framework mastered, you can quickly use it to make more charts.

TODO: add some examples of creating Charts

specific charting examples

This section will focus on some specific applications of applying VBA to charts. The code here can be quickly reused for your own application. These examples include:

- Creating a grid of XY scatter plots (a scatter matrix) based on a block of data
- Creating a panel of time series, one chart per each value with a common x-axis

TODO: identify the examples to include here

creating an XY scatter matrix

ChartCreateXYGrid.md

```vba
Public Sub ChartCreateXYGrid()

    On Error GoTo ChartCreateXYGrid_Error

    DeleteAllCharts
    'VBA doesn't allow a constant to be defined using a function (rgb) so we
        use a local variable rather than
    'muddying it up with the calculated value of the rgb function
    Dim majorGridlineColor As Long
    majorGridlineColor = RGB(200, 200, 200)
    Dim minorGridlineColor As Long
    minorGridlineColor = RGB(220, 220, 220)

    Const CHART_HEIGHT As Long = 300
    Const CHART_WIDTH As Long = 400
    Const MARKER_SIZE As Long = 3
    'dataRange will contain the block of data with titles included
    Dim dataRange As Range
    Set dataRange = Application.InputBox("Select data with titles", Type:=8)

    Application.ScreenUpdating = False

    Dim rowIndex As Long, columnIndex As Long
    rowIndex = 0

    Dim xAxisDataRange As Range, yAxisDataRange As Range
    For Each yAxisDataRange In dataRange.Columns
        columnIndex = 0

        For Each xAxisDataRange In dataRange.Columns
            If rowIndex <> columnIndex Then
                Dim targetChart As Chart
                Set targetChart = ActiveSheet.ChartObjects.Add(columnIndex *
                    CHART_WIDTH, _
                                                                rowIndex *
                                                                CHART_HEIGHT
                                                                + 100, _
```

```
34                                                             CHART_WIDTH,
                                                               CHART_HEIGHT
                                                               ).Chart
35
36            Dim targetSeries As series
37            Dim butlSeries As New bUTLChartSeries
38
39            'offset allows for the title to be excluded
40            Set butlSeries.XValues = Intersect(xAxisDataRange,
                  xAxisDataRange.Offset(1))
41            Set butlSeries.Values = Intersect(yAxisDataRange,
                  yAxisDataRange.Offset(1))
42            Set butlSeries.name = yAxisDataRange.Cells(1)
43            butlSeries.ChartType = xlXYScatter
44
45            Set targetSeries = butlSeries.AddSeriesToChart(targetChart)
46
47            targetSeries.MarkerSize = MARKER_SIZE
48            targetSeries.MarkerStyle = xlMarkerStyleCircle
49
50            Dim targetAxis As Axis
51            Set targetAxis = targetChart.Axes(xlCategory)
52            targetAxis.HasTitle = True
53            targetAxis.AxisTitle.Text = xAxisDataRange.Cells(1)
54            targetAxis.MajorGridlines.Border.Color = majorGridlineColor
55            targetAxis.MinorGridlines.Border.Color = minorGridlineColor
56
57            Set targetAxis = targetChart.Axes(xlValue)
58            targetAxis.HasTitle = True
59            targetAxis.AxisTitle.Text = yAxisDataRange.Cells(1)
60            targetAxis.MajorGridlines.Border.Color = majorGridlineColor
61            targetAxis.MinorGridlines.Border.Color = minorGridlineColor
62
63            targetChart.HasTitle = True
64            targetChart.ChartTitle.Text = yAxisDataRange.Cells(1) & " vs.
                  " & xAxisDataRange.Cells(1)
65            'targetChart.ChartTitle.Characters.Font.Size = 8
66            targetChart.Legend.Delete
67        End If
```

```
68
69            columnIndex = columnIndex + 1
70        Next xAxisDataRange
71
72        rowIndex = rowIndex + 1
73    Next yAxisDataRange
74
75    Application.ScreenUpdating = True
76
77    dataRange.Cells(1, 1).Activate
78
79    On Error GoTo 0
80    Exit Sub
81
82 ChartCreateXYGrid_Error:
83
84    MsgBox "Error " & Err.Number & " (" & Err.Description & _
85            ") in procedure ChartCreateXYGrid of Module Chart_Format"
86    MsgBox "This is most likely due to Range issues"
87
88 End Sub
```

creating a panel of time series plots

Chart_TimeSeries.md

```
1 Public Sub Chart_TimeSeries(ByVal rangeOfDates As Range, ByVal dataRange As
    Range, ByVal rangeOfTitles As Range)
2
3    Application.ScreenUpdating = False
4    Const MARKER_SIZE As Long = 3
5    Dim majorGridlineColor As Long
6    majorGridlineColor = RGB(200, 200, 200)
7    Dim chartIndex As Long
8    chartIndex = 1
9
10    Dim titleRange As Range
11    Dim targetColumn As Range
12
13    For Each titleRange In rangeOfTitles
```

```vbnet
14
15          Dim targetObject As ChartObject
16          Set targetObject = ActiveSheet.ChartObjects.Add(chartIndex * 300, 0,
                300, 300)
17
18          Dim targetChart As Chart
19          Set targetChart = targetObject.Chart
20          targetChart.ChartType = xlXYScatterLines
21          targetChart.HasTitle = True
22          targetChart.Legend.Delete
23
24          Dim targetAxis As Axis
25          Set targetAxis = targetChart.Axes(xlValue)
26          targetAxis.MajorGridlines.Border.Color = majorGridlineColor
27
28          Dim targetSeries As series
29          Dim butlSeries As New bUTLChartSeries
30
31          Set butlSeries.XValues = rangeOfDates
32          Set butlSeries.Values = dataRange.Columns(chartIndex)
33          Set butlSeries.name = titleRange
34
35          Set targetSeries = butlSeries.AddSeriesToChart(targetChart)
36
37          targetSeries.MarkerSize = MARKER_SIZE
38          targetSeries.MarkerStyle = xlMarkerStyleCircle
39
40          chartIndex = chartIndex + 1
41
42      Next titleRange
43
44      Application.ScreenUpdating = True
45  End Sub
```

applying common formatting to all Charts

ChartDefaultFormat.md
```vbnet
1  Public Sub ChartDefaultFormat()
2
```

```vba
    Const MARKER_SIZE As Long = 3
    Dim majorGridlineColor As Long
    majorGridlineColor = RGB(242, 242, 242)
    Const TITLE_FONT_SIZE As Long = 12
    Const SERIES_LINE_WEIGHT As Single = 1.5

    Dim targetObject As ChartObject

    For Each targetObject In Chart_GetObjectsFromObject(Selection)
        Dim targetChart As Chart

        Set targetChart = targetObject.Chart

        Dim targetSeries As series
        For Each targetSeries In targetChart.SeriesCollection

            targetSeries.MarkerSize = MARKER_SIZE
            targetSeries.MarkerStyle = xlMarkerStyleCircle

            If targetSeries.ChartType = xlXYScatterLines Then targetSeries.
                Format.Line.Weight = SERIES_LINE_WEIGHT

            targetSeries.MarkerForegroundColorIndex = xlColorIndexNone
            targetSeries.MarkerBackgroundColorIndex = xlColorIndexAutomatic

        Next targetSeries


        targetChart.HasLegend = True
        targetChart.Legend.Position = xlLegendPositionBottom

        Dim targetAxis As Axis
        Set targetAxis = targetChart.Axes(xlValue)

        targetAxis.MajorGridlines.Border.Color = majorGridlineColor
        targetAxis.Crosses = xlAxisCrossesMinimum

        Set targetAxis = targetChart.Axes(xlCategory)
```

```
41        targetAxis.HasMajorGridlines = True
42
43        targetAxis.MajorGridlines.Border.Color = majorGridlineColor
44
45        If targetChart.HasTitle Then
46            targetChart.ChartTitle.Characters.Font.Size = TITLE_FONT_SIZE
47            targetChart.ChartTitle.Characters.Font.Bold = True
48        End If
49
50        Set targetAxis = targetChart.Axes(xlCategory)
51
52    Next targetObject
53
54 End Sub
```

Chart_AddTitles.md

```
1 Public Sub Chart_AddTitles()
2
3     Dim targetObject As ChartObject
4     Const X_AXIS_TITLE As String = "x axis"
5     Const Y_AXIS_TITLE As String = "y axis"
6     Const SECOND_Y_AXIS_TITLE As String = "2and y axis"
7     Const CHART_TITLE As String = "chart"
8
9     For Each targetObject In Chart_GetObjectsFromObject(Selection)
10        With targetObject.Chart
11            If Not .Axes(xlCategory).HasTitle Then
12                .Axes(xlCategory).HasTitle = True
13                .Axes(xlCategory).AxisTitle.Text = X_AXIS_TITLE
14            End If
15
16            If Not .Axes(xlValue, xlPrimary).HasTitle Then
17                .Axes(xlValue).HasTitle = True
18                .Axes(xlValue).AxisTitle.Text = Y_AXIS_TITLE
19            End If
20
21            '2015 12 14, add support for 2and y axis
```

```
22          If .Axes.Count = 3 Then
23              If Not .Axes(xlValue, xlSecondary).HasTitle Then
24                  .Axes(xlValue, xlSecondary).HasTitle = True
25                  .Axes(xlValue, xlSecondary).AxisTitle.Text =
                        SECOND_Y_AXIS_TITLE
26              End If
27          End If
28
29          If Not .HasTitle Then
30              .HasTitle = True
31              .ChartTitle.Text = CHART_TITLE
32          End If
33      End With
34   Next targetObject
35
36 End Sub
```

Chart_ApplyFormattingToSelected.md

```
1  Public Sub Chart_ApplyFormattingToSelected()
2
3      Dim targetObject As ChartObject
4      Const MARKER_SIZE As Long = 5
5
6      For Each targetObject In Chart_GetObjectsFromObject(Selection)
7
8          Dim targetSeries As series
9
10         For Each targetSeries In targetObject.Chart.SeriesCollection
11             targetSeries.MarkerSize = MARKER_SIZE
12         Next targetSeries
13     Next targetObject
14
15 End Sub
```

Chart_ApplyTrendColors.md

```
1   Public Sub Chart_ApplyTrendColors()
2
3       Dim targetObject As ChartObject
4       For Each targetObject In Chart_GetObjectsFromObject(Selection)
5
6           Dim targetSeries As series
7           For Each targetSeries In targetObject.Chart.SeriesCollection
8
9               Dim butlSeries As New bUTLChartSeries
10              butlSeries.UpdateFromChartSeries targetSeries
11
12              targetSeries.MarkerForegroundColorIndex = xlColorIndexNone
13              targetSeries.MarkerBackgroundColor = Chart_GetColor(butlSeries.
                    SeriesNumber)
14
15              targetSeries.Format.Line.ForeColor.RGB = targetSeries.
                    MarkerBackgroundColor
16
17          Next targetSeries
18      Next targetObject
19  End Sub
```

Chart_CreateChartWithSeriesForEachColumn.md

```
1   Public Sub Chart_CreateChartWithSeriesForEachColumn()
2       'will create a chart that includes a series with no x value for each
            column
3
4       Dim dataRange As Range
5       Set dataRange = GetInputOrSelection("Select chart data")
6
7       'create a chart
8       Dim targetObject As ChartObject
9       Set targetObject = ActiveSheet.ChartObjects.Add(0, 0, 300, 300)
10
11      targetObject.Chart.ChartType = xlXYScatter
12
```

```
13     Dim targetColumn As Range
14     For Each targetColumn In dataRange.Columns
15
16         Dim chartDataRange As Range
17         Set chartDataRange = RangeEnd(targetColumn.Cells(1, 1), xlDown)
18
19         Dim butlSeries As New bUTLChartSeries
20         Set butlSeries.Values = chartDataRange
21
22         butlSeries.AddSeriesToChart targetObject.Chart
23     Next targetColumn
24
25 End Sub
```

Chart_CreateDataLabels.md

```
1  Public Sub Chart_CreateDataLabels()
2
3      Dim targetObject As ChartObject
4      On Error GoTo Chart_CreateDataLabels_Error
5
6      For Each targetObject In Chart_GetObjectsFromObject(Selection)
7
8          Dim targetSeries As series
9          For Each targetSeries In targetObject.Chart.SeriesCollection
10
11             Dim dataPoint As Point
12             Set dataPoint = targetSeries.Points(2)
13
14             dataPoint.HasDataLabel = False
15             dataPoint.DataLabel.Position = xlLabelPositionRight
16             dataPoint.DataLabel.ShowSeriesName = True
17             dataPoint.DataLabel.ShowValue = False
18             dataPoint.DataLabel.ShowCategoryName = False
19             dataPoint.DataLabel.ShowLegendKey = True
20
21         Next targetSeries
22     Next targetObject
```

```
23
24    On Error GoTo 0
25    Exit Sub
26
27 Chart_CreateDataLabels_Error:
28
29    MsgBox "Error " & Err.Number & " (" & Err.Description & ") in procedure
          Chart_CreateDataLabels of Module Chart_Format"
30
31 End Sub
```

Chart_ExtendSeriesToRanges.md

```
1  Public Sub Chart_ExtendSeriesToRanges()
2
3      Dim targetObject As ChartObject
4
5      For Each targetObject In Chart_GetObjectsFromObject(Selection)
6
7          Dim targetSeries As series
8
9          'get each series
10         For Each targetSeries In targetObject.Chart.SeriesCollection
11
12             'create the bUTL obj and manipulate series ranges
13             Dim butlSeries As New bUTLChartSeries
14             butlSeries.UpdateFromChartSeries targetSeries
15
16             If Not butlSeries.XValues Is Nothing Then
17                 targetSeries.XValues = RangeEnd(butlSeries.XValues.Cells(1),
                       xlDown)
18             End If
19             targetSeries.Values = RangeEnd(butlSeries.Values.Cells(1), xlDown
                   )
20
21         Next targetSeries
22     Next targetObject
23 End Sub
```

Chart_GoToXRange.md

```
Public Sub Chart_GoToXRange()


    If TypeName(Selection) = "Series" Then
        Dim b As New bUTLChartSeries
        b.UpdateFromChartSeries Selection

        b.XValues.Parent.Activate
        b.XValues.Activate
    Else
        MsgBox "Select a series in order to use this."
    End If

End Sub
```

Chart_SortSeriesByName.md

```
Public Sub Chart_SortSeriesByName()
    'this will sort series by names
    Dim targetObject As ChartObject
    For Each targetObject In Chart_GetObjectsFromObject(Selection)

        'uses a simple bubble sort but it works... shouldn't have 1000 series
            anyways
        Dim firstChartIndex As Long
        Dim secondChartIndex As Long
        For firstChartIndex = 1 To targetObject.Chart.SeriesCollection.Count
            For secondChartIndex = (firstChartIndex + 1) To targetObject.
                Chart.SeriesCollection.Count

                Dim butlSeries1 As New bUTLChartSeries
                Dim butlSeries2 As New bUTLChartSeries
```

```
14
15              butlSeries1.UpdateFromChartSeries targetObject.Chart.
                    SeriesCollection(firstChartIndex)
16              butlSeries2.UpdateFromChartSeries targetObject.Chart.
                    SeriesCollection(secondChartIndex)
17
18              If butlSeries1.name.Value > butlSeries2.name.Value Then
19                  Dim indexSeriesSwap As Long
20                  indexSeriesSwap = butlSeries2.SeriesNumber
21                  butlSeries2.SeriesNumber = butlSeries1.SeriesNumber
22                  butlSeries1.SeriesNumber = indexSeriesSwap
23                  butlSeries2.UpdateSeriesWithNewValues
24                  butlSeries1.UpdateSeriesWithNewValues
25              End If
26
27          Next secondChartIndex
28        Next firstChartIndex
29    Next targetObject
30 End Sub
```

ChartFlipXYValues.md

```
1  Public Sub ChartFlipXYValues()
2
3      Dim targetObject As ChartObject
4      Dim targetChart As Chart
5      For Each targetObject In Chart_GetObjectsFromObject(Selection)
6          Set targetChart = targetObject.Chart
7
8          Dim butlSeriesies As New Collection
9          Dim butlSeries As bUTLChartSeries
10
11         Dim targetSeries As series
12         For Each targetSeries In targetChart.SeriesCollection
13             Set butlSeries = New bUTLChartSeries
14             butlSeries.UpdateFromChartSeries targetSeries
15
16             Dim dummyRange As Range
```

```vba
17
18              Set dummyRange = butlSeries.Values
19              Set butlSeries.Values = butlSeries.XValues
20              Set butlSeries.XValues = dummyRange
21
22          'need to change the series name also
23          'assume that title is same offset
24          'code blocked for now
25          If False And Not butlSeries.name Is Nothing Then
26              Dim rowsOffset As Long, columnsOffset As Long
27              rowsOffset = butlSeries.name.Row - butlSeries.XValues.Cells
                    (1, 1).Row
28              columnsOffset = butlSeries.name.Column - butlSeries.XValues.
                    Cells(1, 1).Column
29
30              Set butlSeries.name = butlSeries.Values.Cells(1, 1).Offset(
                    rowsOffset, columnsOffset)
31          End If
32
33          butlSeries.UpdateSeriesWithNewValues
34
35      Next targetSeries
36
37      ''need to flip axis labels if they exist
38      ''three cases: X only, Y only, X and Y
39
40      If targetChart.Axes(xlCategory).HasTitle And Not targetChart.Axes(
            xlValue).HasTitle Then
41
42          targetChart.Axes(xlValue).HasTitle = True
43          targetChart.Axes(xlValue).AxisTitle.Text = targetChart.Axes(
                xlCategory).AxisTitle.Text
44          targetChart.Axes(xlCategory).HasTitle = False
45
46      ElseIf Not targetChart.Axes(xlCategory).HasTitle And targetChart.Axes
            (xlValue).HasTitle Then
47          targetChart.Axes(xlCategory).HasTitle = True
48          targetChart.Axes(xlCategory).AxisTitle.Text = targetChart.Axes(
                xlValue).AxisTitle.Text
```

```
49          targetChart.Axes(xlValue).HasTitle = False
50
51      ElseIf targetChart.Axes(xlCategory).HasTitle And targetChart.Axes(
           xlValue).HasTitle Then
52          Dim swapText As String
53
54          swapText = targetChart.Axes(xlCategory).AxisTitle.Text
55
56          targetChart.Axes(xlCategory).AxisTitle.Text = targetChart.Axes(
               xlValue).AxisTitle.Text
57          targetChart.Axes(xlValue).AxisTitle.Text = swapText
58
59      End If
60
61      Set butlSeriesies = Nothing
62
63    Next targetObject
64
65 End Sub
```

ChartMergeSeries.md

```
 1 Public Sub ChartMergeSeries()
 2
 3     Dim targetObject As ChartObject
 4     Dim targetChart As Chart
 5     Dim firstChart As Chart
 6
 7     Dim isFirstChart As Boolean
 8     isFirstChart = True
 9
10     Application.ScreenUpdating = False
11
12     For Each targetObject In Chart_GetObjectsFromObject(Selection)
13
14         Set targetChart = targetObject.Chart
15         If isFirstChart Then
16             Set firstChart = targetChart
```

```vba
            isFirstChart = False
        Else
            Dim targetSeries As series
            For Each targetSeries In targetChart.SeriesCollection

                Dim newChartSeries As series
                Dim butlSeries As New bUTLChartSeries

                butlSeries.UpdateFromChartSeries targetSeries
                Set newChartSeries = butlSeries.AddSeriesToChart(firstChart)

                newChartSeries.MarkerSize = targetSeries.MarkerSize
                newChartSeries.MarkerStyle = targetSeries.MarkerStyle

                targetSeries.Delete

            Next targetSeries

            targetObject.Delete

        End If
    Next targetObject

    Application.ScreenUpdating = True

End Sub
```

ChartSplitSeries.md

```vba
Public Sub ChartSplitSeries()

    Dim targetObject As ChartObject
    Dim targetChart As Chart

    Dim targetSeries As series
    For Each targetObject In Chart_GetObjectsFromObject(Selection)

        For Each targetSeries In targetObject.Chart.SeriesCollection
```

```
10
11          Dim newChartObject As ChartObject
12          Set newChartObject = ActiveSheet.ChartObjects.Add(0, 0, 300, 300)
13
14          Dim newChartSeries As series
15          Dim butlSeries As New bUTLChartSeries
16
17          butlSeries.UpdateFromChartSeries targetSeries
18          Set newChartSeries = butlSeries.AddSeriesToChart(newChartObject.
                Chart)
19
20          newChartSeries.MarkerSize = targetSeries.MarkerSize
21          newChartSeries.MarkerStyle = targetSeries.MarkerStyle
22
23          targetSeries.Delete
24
25      Next targetSeries
26
27
28      targetObject.Delete
29
30    Next targetObject
31 End Sub
```

DeleteAllCharts.md

```
1 Public Sub DeleteAllCharts()
2
3     If MsgBox("Delete all charts?", vbYesNo) = vbYes Then
4         Application.ScreenUpdating = False
5
6         Dim chartObjectIndex As Long
7         For chartObjectIndex = ActiveSheet.ChartObjects.Count To 1 Step -1
8
9             ActiveSheet.ChartObjects(chartObjectIndex).Delete
10
11         Next chartObjectIndex
12
```

```
13        Application.ScreenUpdating = True
14
15    End If
16 End Sub
```

RemoveZeroValueDataLabel.md

```
1  Public Sub RemoveZeroValueDataLabel()
2
3      'uses the ActiveChart, be sure a chart is selected
4      Dim targetChart As Chart
5      Set targetChart = ActiveChart
6
7      Dim targetSeries As series
8      For Each targetSeries In targetChart.SeriesCollection
9
10         Dim seriesValues As Variant
11         seriesValues = targetSeries.Values
12
13         'include this line if you want to reestablish labels before deleting
14         targetSeries.ApplyDataLabels xlDataLabelsShowLabel, , , , True, False
              , False, False, False
15
16         'loop through values and delete 0-value labels
17         Dim pointIndex As Long
18         For pointIndex = LBound(seriesValues) To UBound(seriesValues)
19             If seriesValues(pointIndex) = 0 Then
20                 With targetSeries.Points(pointIndex)
21                     If .HasDataLabel Then .DataLabel.Delete
22                 End With
23             End If
24         Next pointIndex
25     Next targetSeries
26 End Sub
```

UpdateFromChartSeries.md

```vba
Public Sub UpdateFromChartSeries(targetSeries As series)


    'this will work for the simple case where all items are references
    Const FIND_STRING As String = "SERIES("
    Const COMMA As String = ","
    Const CLOSE_BRACKET As String = ")"

    Set series = targetSeries

    Dim targetForm As Variant

    '=SERIES("Y",Sheet1!$C$8:$C$13,Sheet1!$D$8:$D$13,1)

    'pull in the formula
    targetForm = targetSeries.Formula

    'uppercase to remove match errors
    targetForm = UCase(targetForm)

    'remove the front of the formula
    targetForm = Replace(targetForm, FIND_STRING, vbNullString)

    'find the first foundPosition
    Dim foundPosition As Long
    foundPosition = InStr(targetForm, COMMA)

    If foundPosition > 1 Then
        'need to catch an error here if a text name is used instead of a
            valid range
        On Error Resume Next
        Set Me.name = Range(left(targetForm, foundPosition - 1))
        If Err <> 0 Then pName = left(targetForm, foundPosition - 1)
        On Error GoTo 0
    End If

    'pull out the title from that
    targetForm = Mid(targetForm, foundPosition + 1)
```

```vba
38
39      foundPosition = InStr(targetForm, COMMA)
40
41      If foundPosition > 1 Then Set Me.XValues = Range(left(targetForm,
            foundPosition - 1))
42
43      targetForm = Mid(targetForm, foundPosition + 1)
44
45      foundPosition = InStr(targetForm, COMMA)
46      Set Me.Values = Range(left(targetForm, foundPosition - 1))
47      targetForm = Mid(targetForm, foundPosition + 1)
48
49      foundPosition = InStr(targetForm, CLOSE_BRACKET)
50      Me.SeriesNumber = left(targetForm, foundPosition - 1)
51
52      Me.ChartType = targetSeries.ChartType
53  End Sub
```