Base R Replication, Part 2: Using ggplot2

Sam Eckhardt

9/16/2014

Purpose

Ggplot2 is a fantastic tool for graph building. while in the last guide, we talked more about what the graphs are for, this time we will talk more about the code involved. Ggplot2 is actually a coding language, simplifying how you design and build your graphs. This is basically a mini Rosetta Stone program on ggplot2. I am your guide, Alfonso.

Data

Lets use the same data from the last guide.

```
## Simulate some data

## 3 Factor Variables
FacVar1=as.factor(rep(c("level1","level2"),25))
FacVar2=as.factor(rep(c("levelA","levelB","levelC"),17)[-51])
FacVar3=as.factor(rep(c("levelI","levelII","levelIII","levelIV"),13)[-c(51:52)])

## 4 Numeric Vars
set.seed(123)
NumVar1=round(rnorm(n=50,mean=1000,sd=50),digits=2) ## Normal distribution
set.seed(123)
NumVar2=round(runif(n=50,min=500,max=1500),digits=2) ## Uniform distribution
set.seed(123)
NumVar3=round(rexp(n=50,rate=.001)) ## Exponential distribution
NumVar4=2001:2050

simData=data.frame(FacVar1,FacVar2,FacVar3,NumVar1,NumVar2,NumVar3,NumVar4)
```

Next, comes the new part

```
library(ggplot2)
library(reshape2)
```

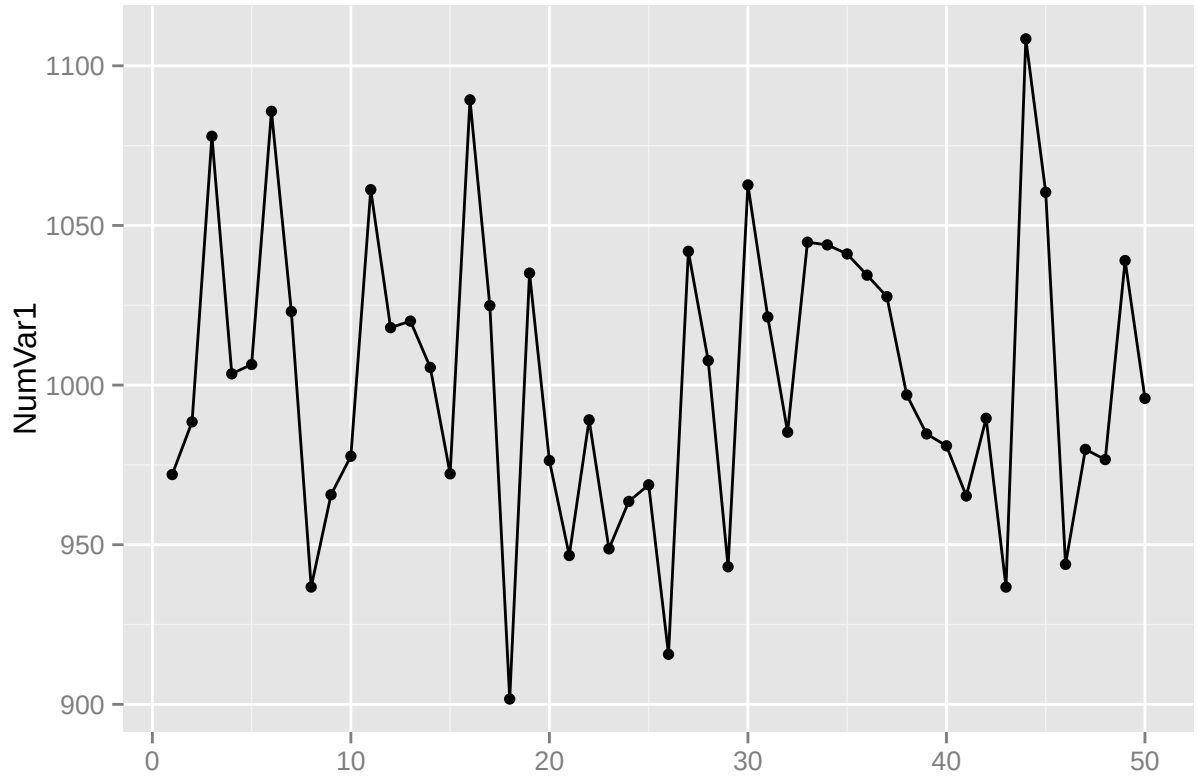
What we have done is activated ggplot2 in our page. This allows us to utilize the coding language in ggplot2. This must be downloaded first; use the packages tab and search to download.

Now, lets see how it works

One Numeric Variable

Index Chart

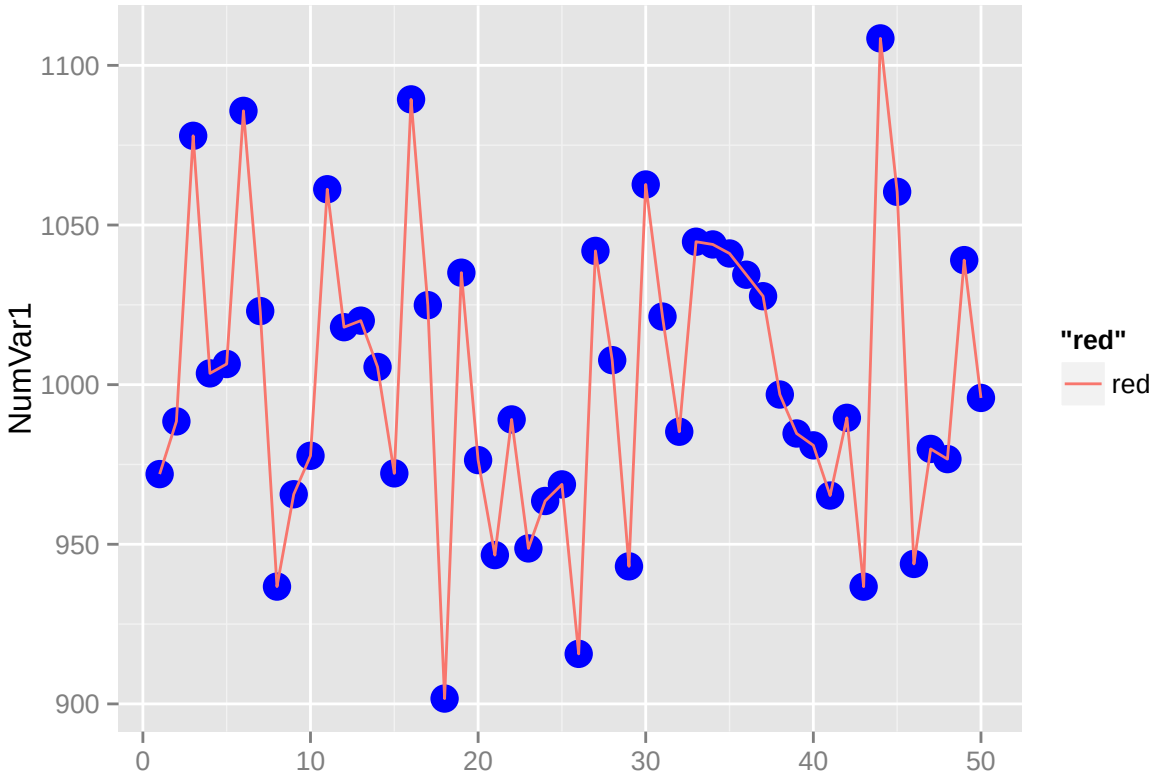
```
ggplot(simData,aes(y=NumVar1,x=1:nrow(simData),group="NumVar1"))+geom_point()+geom_line()+ xlab("") ## l
```



This is the same line graph that we had in the last guide, but now in ggplot2. Basically how it works, is we ask it to ggplot our sim data. then aesthetically, we are going to have Y=numvar1, and X equal each entry row in simdata. We then add on the end as a point chart, a line chart, and leave xlab blank.

Now find the differences in the graph, and the code.

```
ggplot(simData,aes(y=NumVar1,x=1:nrow(simData),group="NumVar1", color="red"))+geom_point(size=5,color="
```

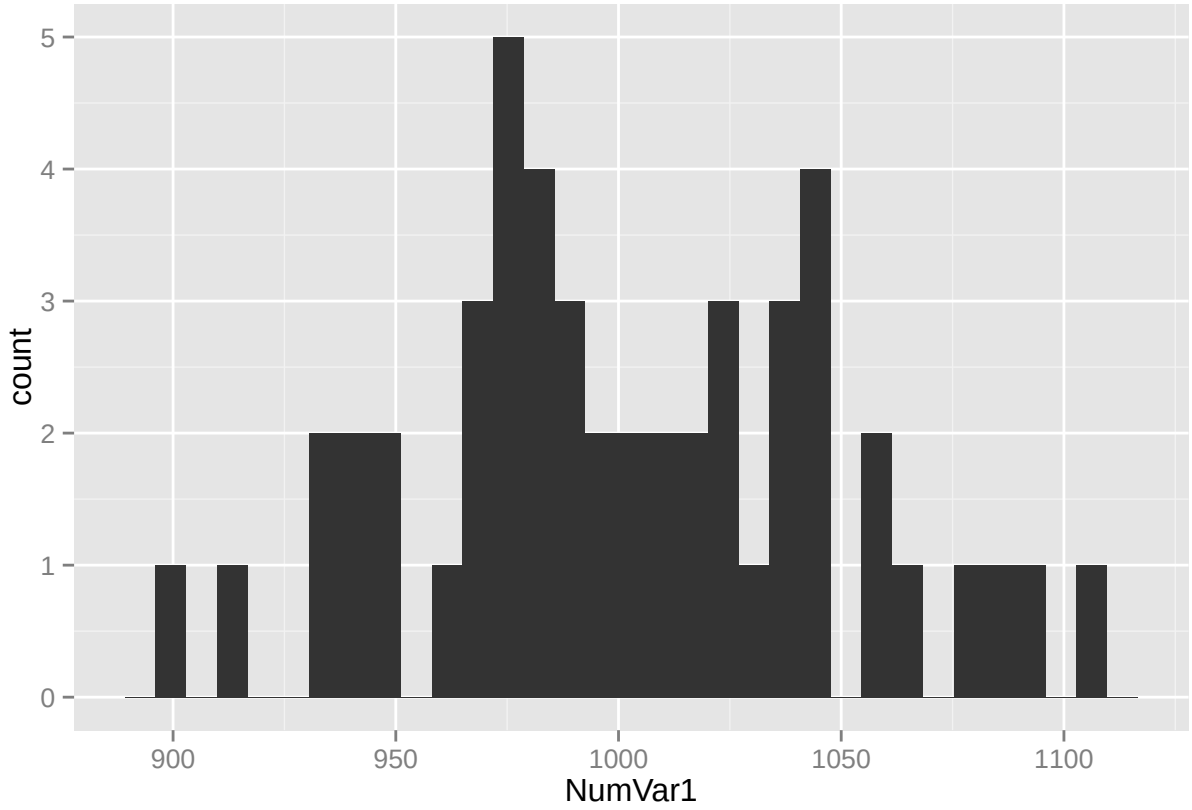


The best part about ggplot2 is what I have done here. I added colors, changed sizes, with a very simple command of size= and color=. Depending on where you place these commands changes the graph as a whole; if I do not put the color of the points as blue following geom_point, they become red, since I commanded the entire graph to be red in the aes command. There are dozens of different things that you can use to adjust the graphs, from the size and color of the lines, to the background type and more. This language is very easy and simple to use compared to just base R, so lets continue on and see what kind of trouble we can get into.

Histogram

```
ggplot(simData,aes(x=NumVar1))+geom_histogram() ## histogram
```

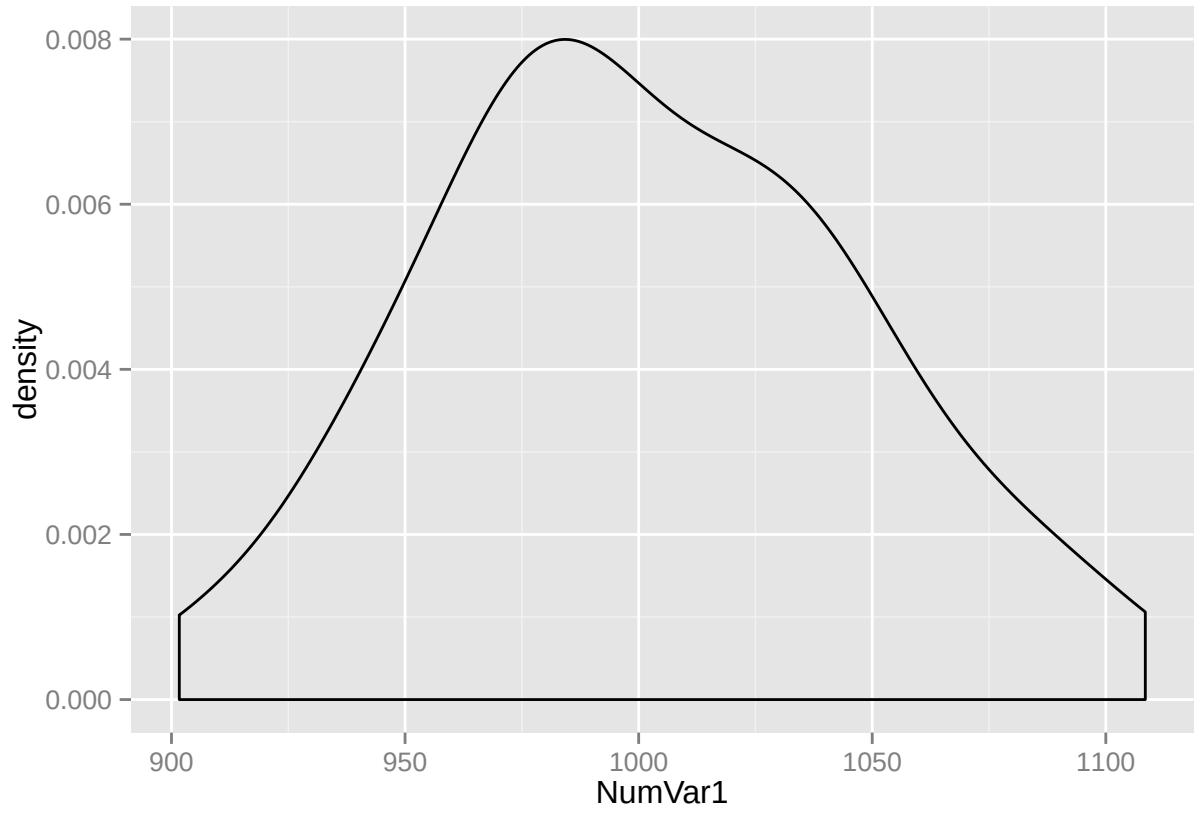
```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



And that is how easy it is to make a histogram. One line of code, that is very easy to understand what is going on. We are using ggplot on the sim data, aesthetically using numvar1, and using the command for geom_histogram, to make a histogram. Very easy.

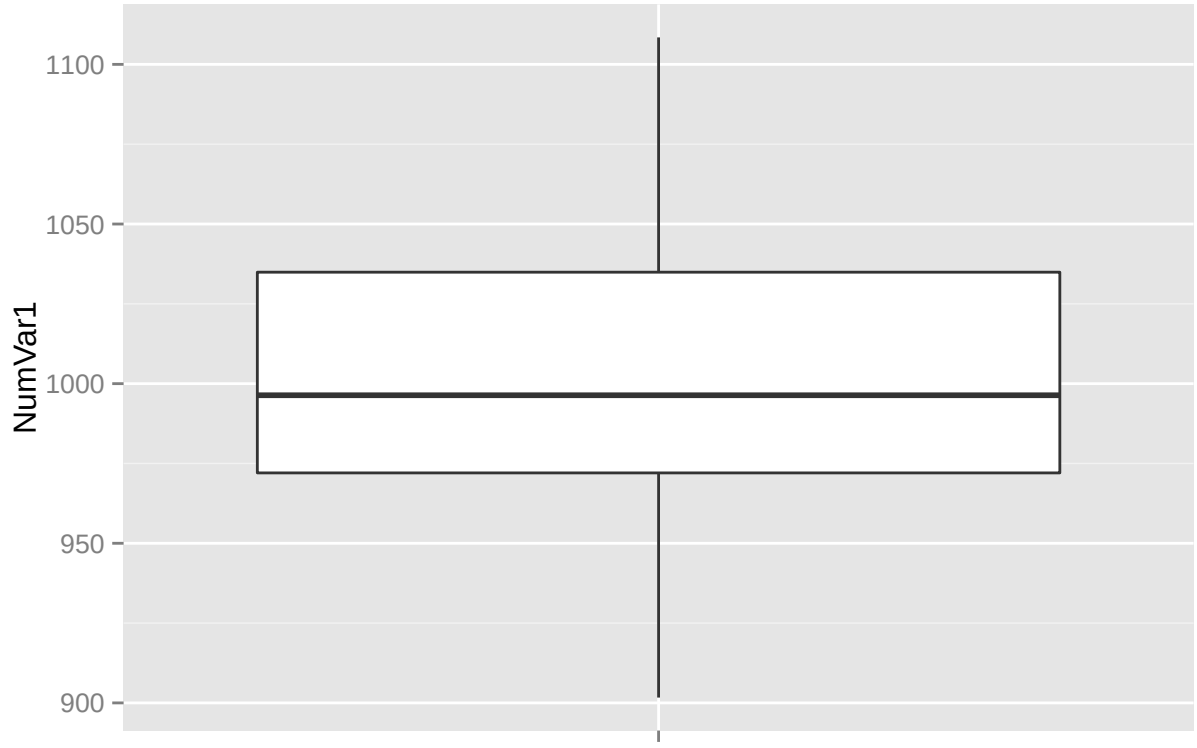
Density Plot

```
ggplot(simData,aes(x=NumVar1))+geom_density() ## Kernel density plot
```



Box PLot

```
ggplot(simData,aes(x=factor(""),y=NumVar1))+geom_boxplot()+ xlab("") ## box plot
```



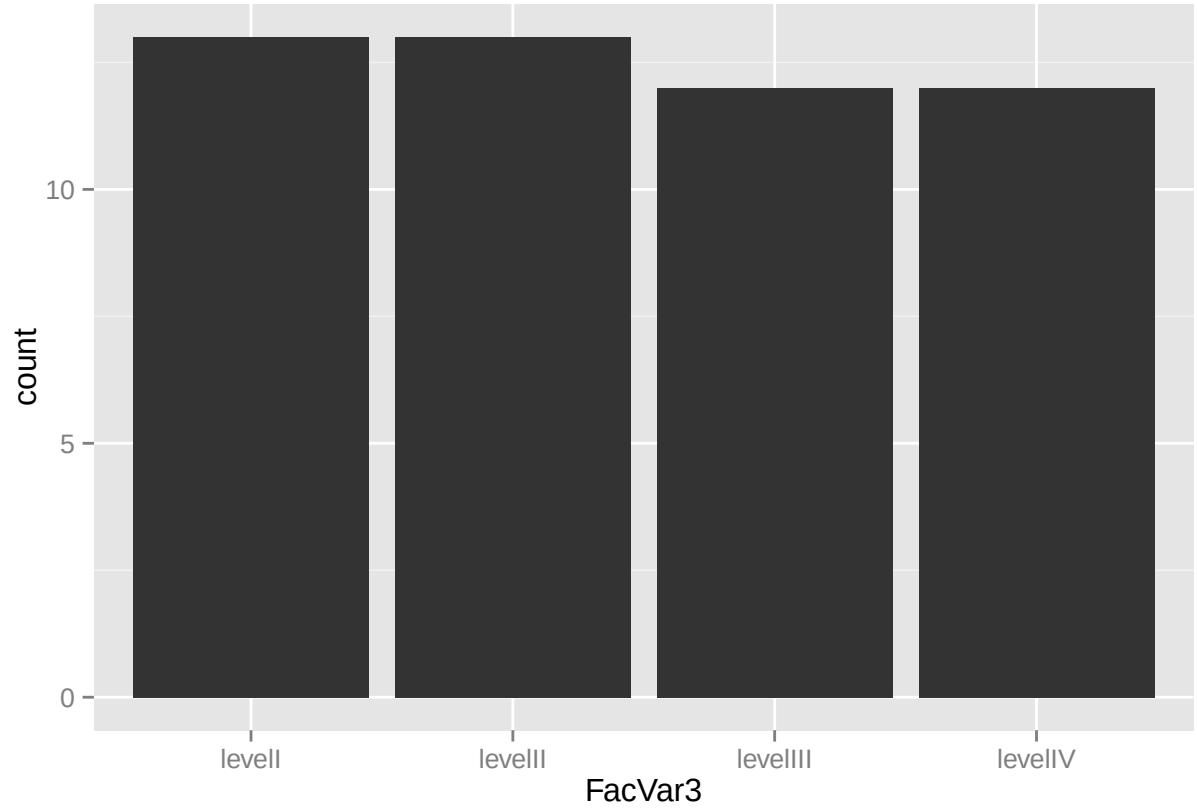
All of these graphs are a simple command, fill in for what you want, use placers in the space of something we want to keep blank, and ggplot2 fills in the rest. All of them are more customizable and easier to use than just base R.

One Factor Variable

Factor variables aren't going to be any different than the numeric ones; they just use different graphs

Bar Chart

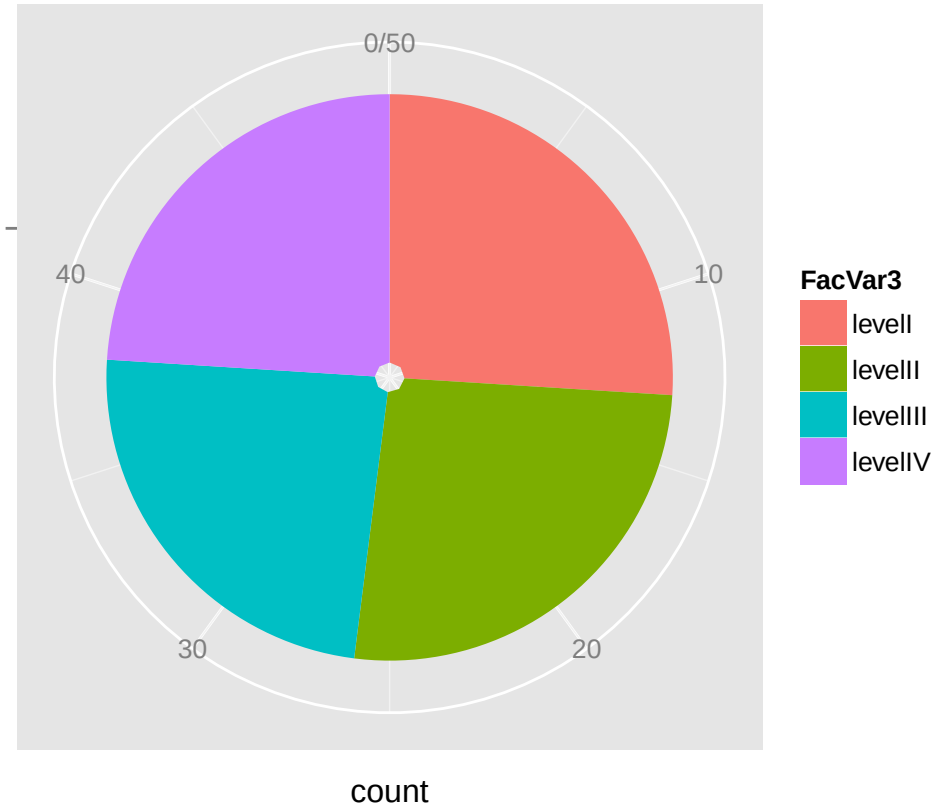
```
## barplot
ggplot(simData,aes(x=FacVar3))+geom_bar()
```



Plot sim data, aesthetically using factor variable 3, then making it a bar plot. Nothing too complicated.

Pie Chart

```
## pie chart - Not the best graph --- use with caution
ggplot(simData,aes(x = factor(""), fill=FacVar3, label=FacVar3))+geom_bar()+ coord_polar(theta = "y")
```



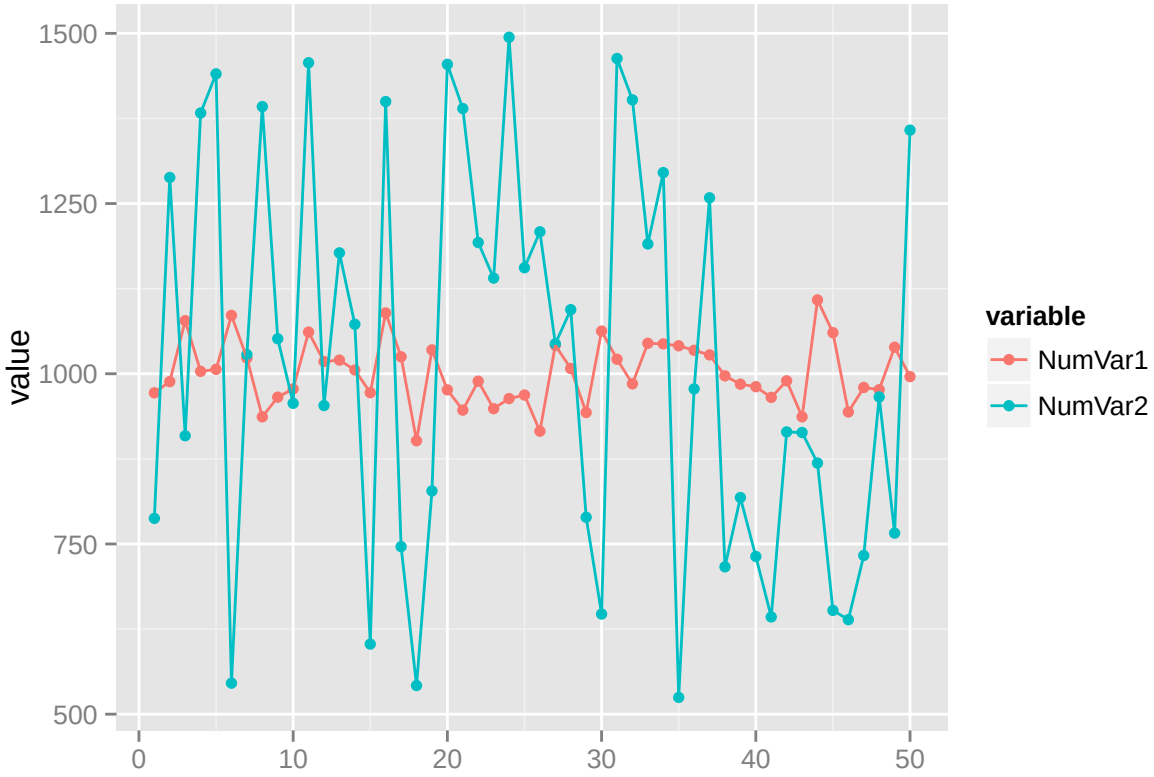
Pie charts are more complicated to use, since there is no specific code in ggplot that is geom_pie. Pie charts are hated by those using R, so to make one is not a simple task by any means. Even then, this graph has its problems, its not the clearest of things, and its not very exact.

Two Variables: Two Numeric

Index

```
simtmp=simData[,c(4:5)] ## 4th and 5th columns are NumVar1 and NumVar2
simtmp$index=1:nrow(simtmp)
simtmpmelt=melt(simtmp,id=c("index"))

## line plots with observation number as index
ggplot(simtmpmelt,aes(y=value,x=index,color=variable))+geom_point()+geom_line()+xlab("")
```

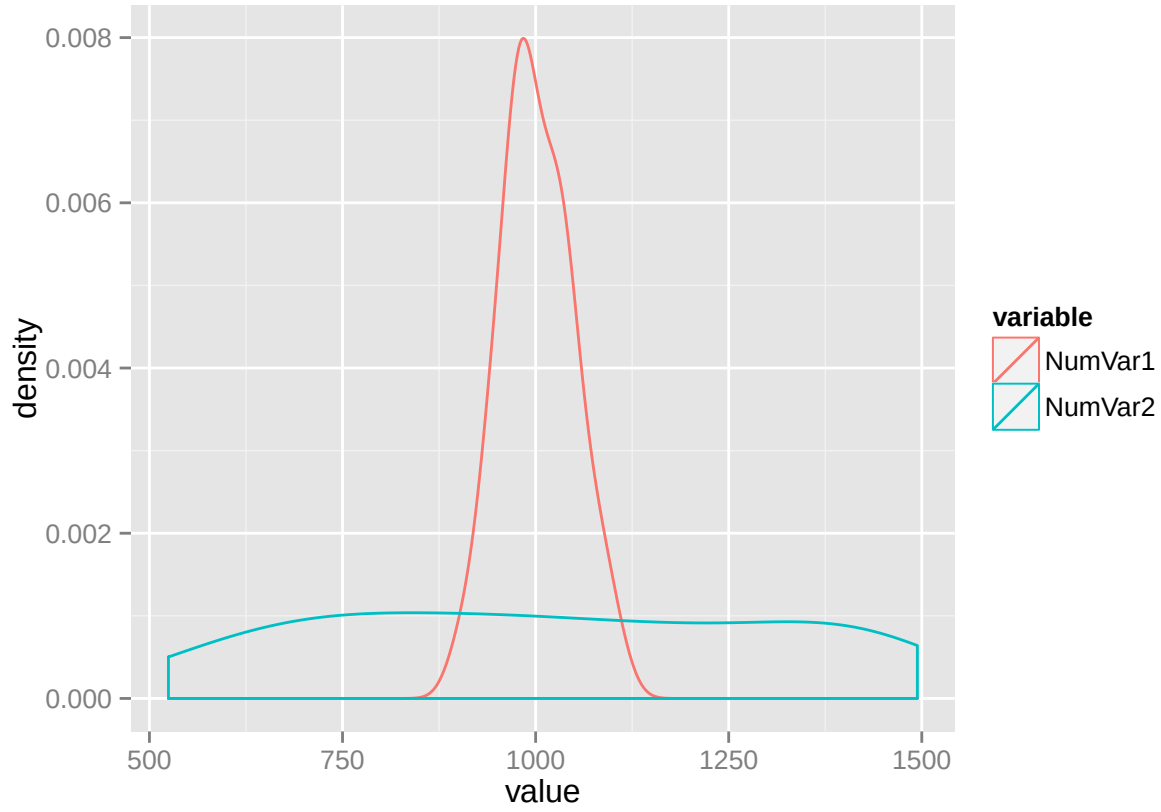


For more than one variable, we cannot just call the Y axis as one variable. We first have to clarify what we want each variable to be, and that is what the first lines of code are. We then call this code "index", and use that as our x axis values. Since we are using two lines in index, we have 2 lines in the graph. Then, we do a line plot just like the last one. Color=variable makes each variable in the model a different color, easily making the lines two different colors.

This first line of code is now usable throughout the guide, known as simtmpmelt. Lets look at another graph

Density Plot

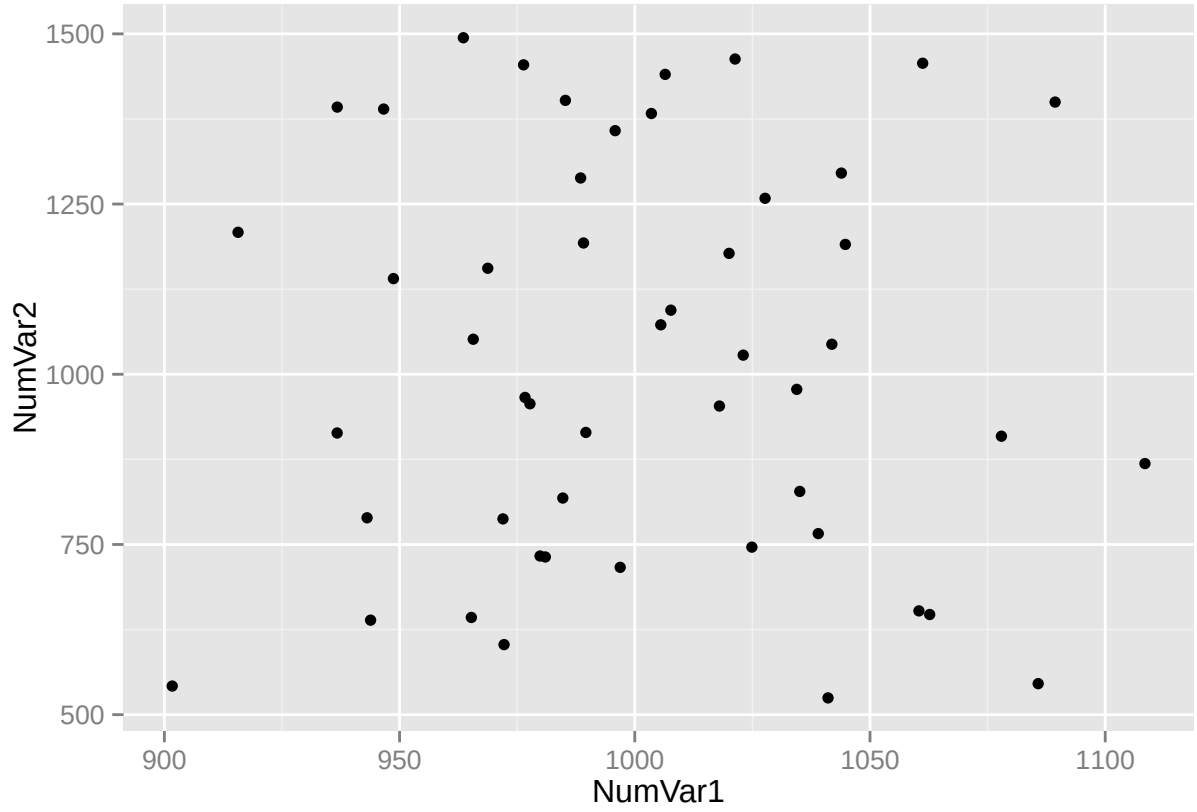
```
## Let's draw density functions for NumVar1 & NumVar2
ggplot(simtmpmelt,aes(x=value,color=variable))+geom_density()
```



Now, we use the simtmpmelt as our guide, and use the 2 variables in this density graph.

Scatterplot

```
## scatter plot
ggplot(simData,aes(x=NumVar1,y=NumVar2))+geom_point()
```



This time, we can just call each axis one variable. Since scatterplots look at that correlation, it is much easier to do that than try to use simtmpmelt. Once again, we only use a point, just like the index graph, but we aren't using a line, making the difference.

Two Factor Variables

Mosaic

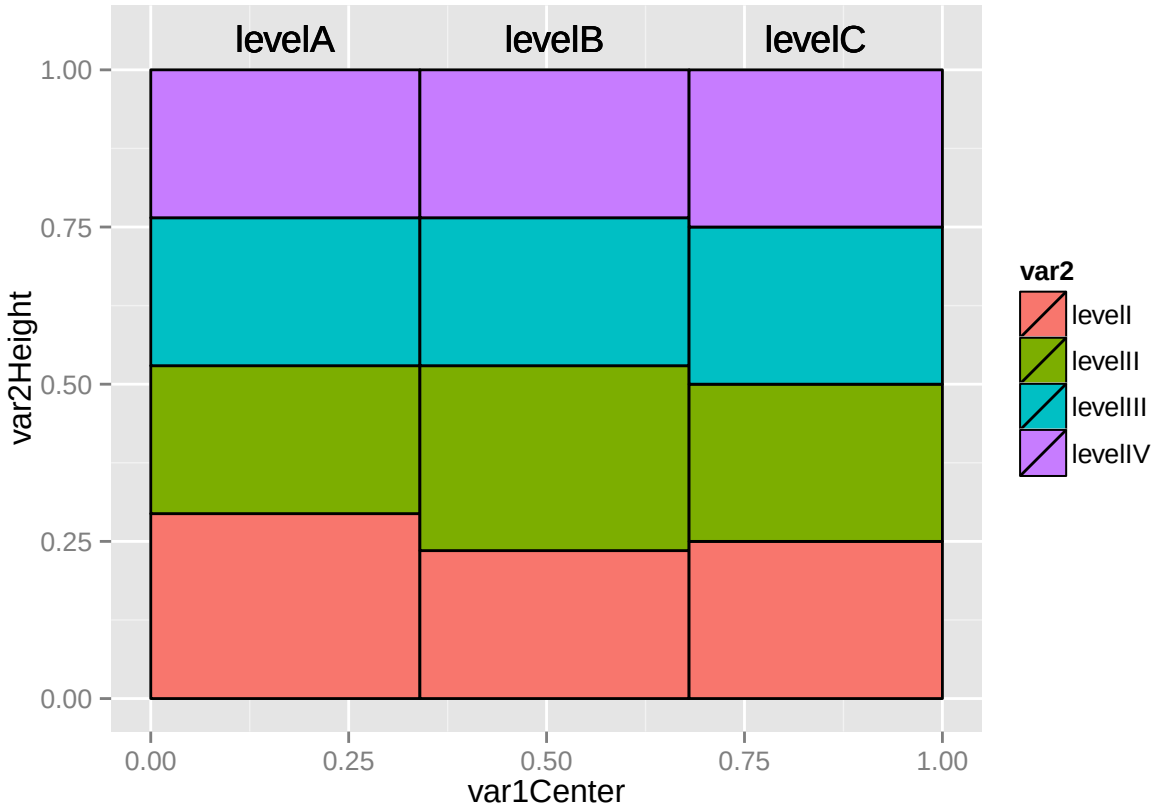
```
## Mosaic plot: ggMMplot function - thanks to Edwin on Stackoverflow: http://stackoverflow.com/questions

ggMMplot <- function(var1, var2){
  require(ggplot2)
  levVar1 <- length(levels(var1))
  levVar2 <- length(levels(var2))

  jointTable <- prop.table(table(var1, var2))
  plotData <- as.data.frame(jointTable)
  plotData$marginVar1 <- prop.table(table(var1))
  plotData$var2Height <- plotData$Freq / plotData$marginVar1
  plotData$var1Center <- c(0, cumsum(plotData$marginVar1)[1:levVar1 -1]) +
    plotData$marginVar1 / 2

  ggplot(plotData, aes(var1Center, var2Height)) +
    geom_bar(stat = "identity", aes(width = marginVar1, fill = var2), col = "Black") +
    geom_text(aes(label = as.character(var1), x = var1Center, y = 1.05))
}
ggMMplot(simData$FacVar2, simData$FacVar3)
```

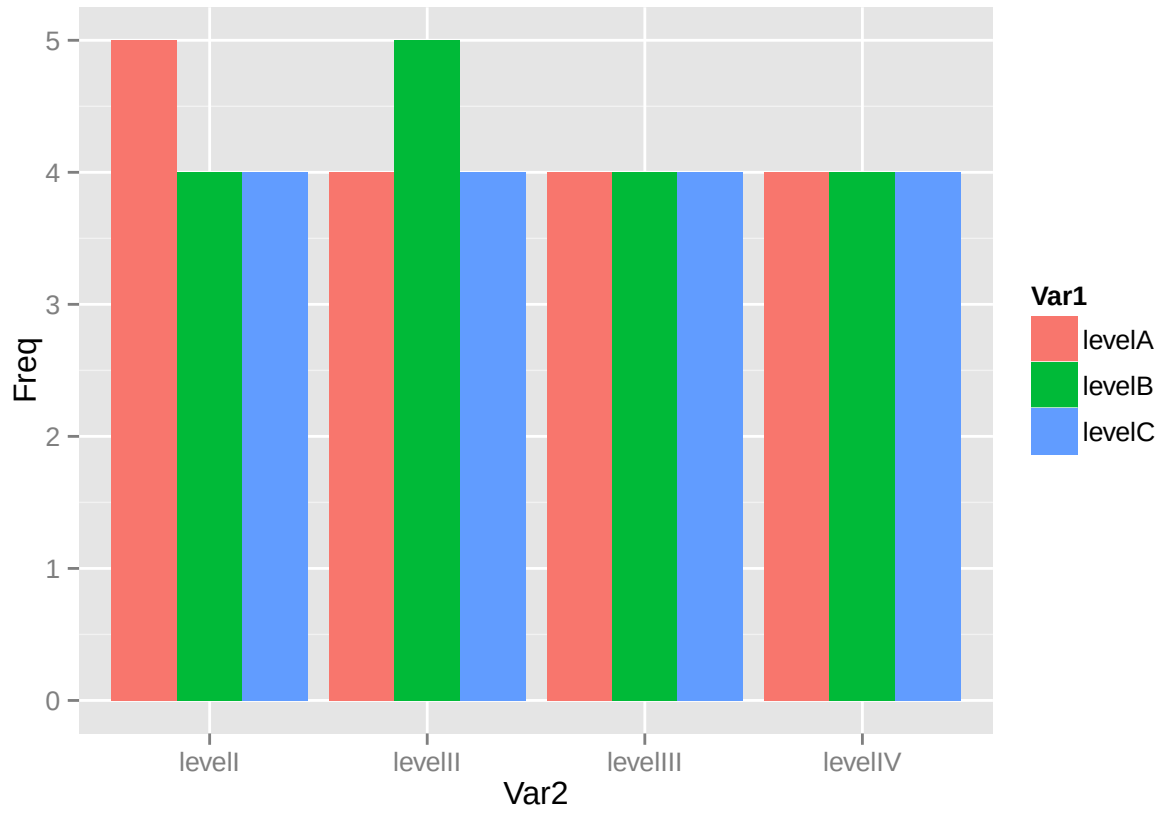
```
## Warning: position_stack requires constant width: output may be incorrect
```



That is some code. Since R doesn't like area graphs, and this is exactly what a mosaic graph is, this becomes very complicated. We could not find it from the basic R guide, we had to search out how to do it from an outside source.

BarPlot

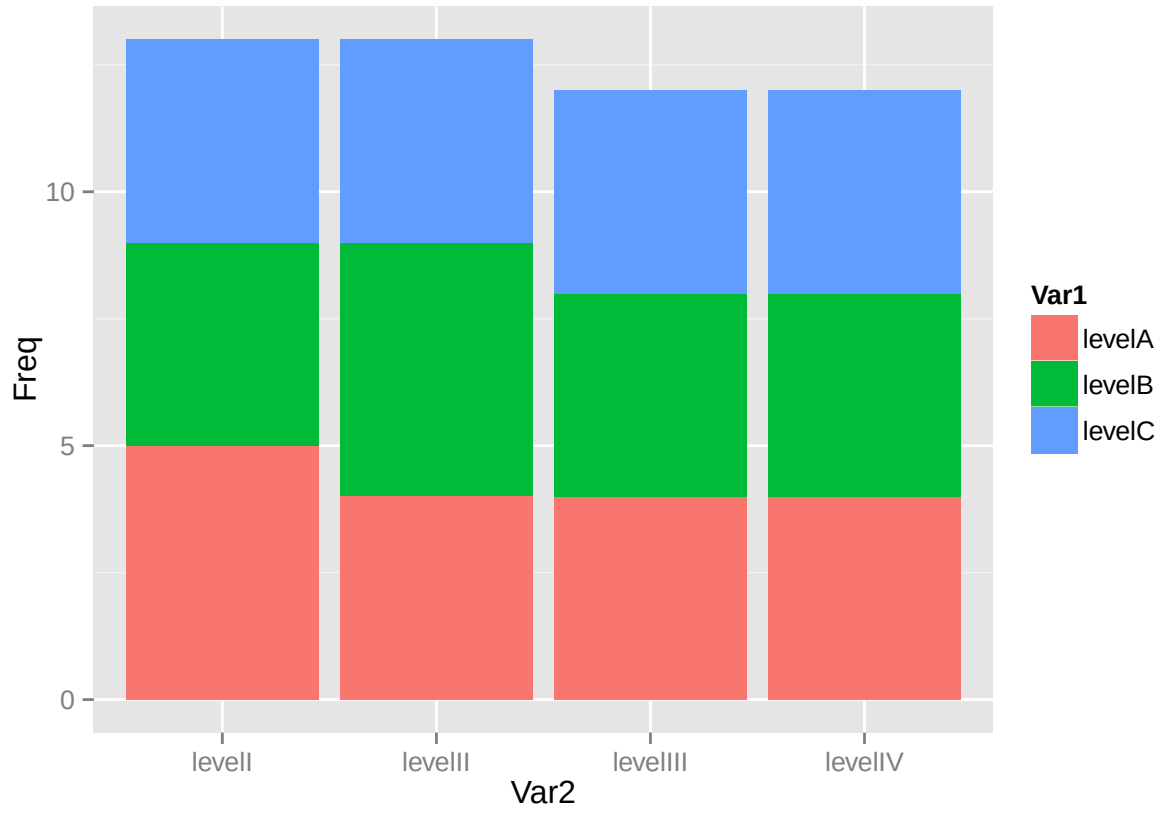
```
bartabledat = as.data.frame(table(simData$FacVar2, simData$FacVar3)) ## get the cross tab
ggplot(bartabledat,aes(x=Var2,y=Freq,fill=Var1))+geom_bar(position="dodge",stat="identity")## plot
```



We have now created a data frame for the bar plot, with bartabledat, and we utilize that to create our bar plot. This code is much easier to use than the mosaic graph, and as well, looks better and gets data across easier.

Stacked Bar

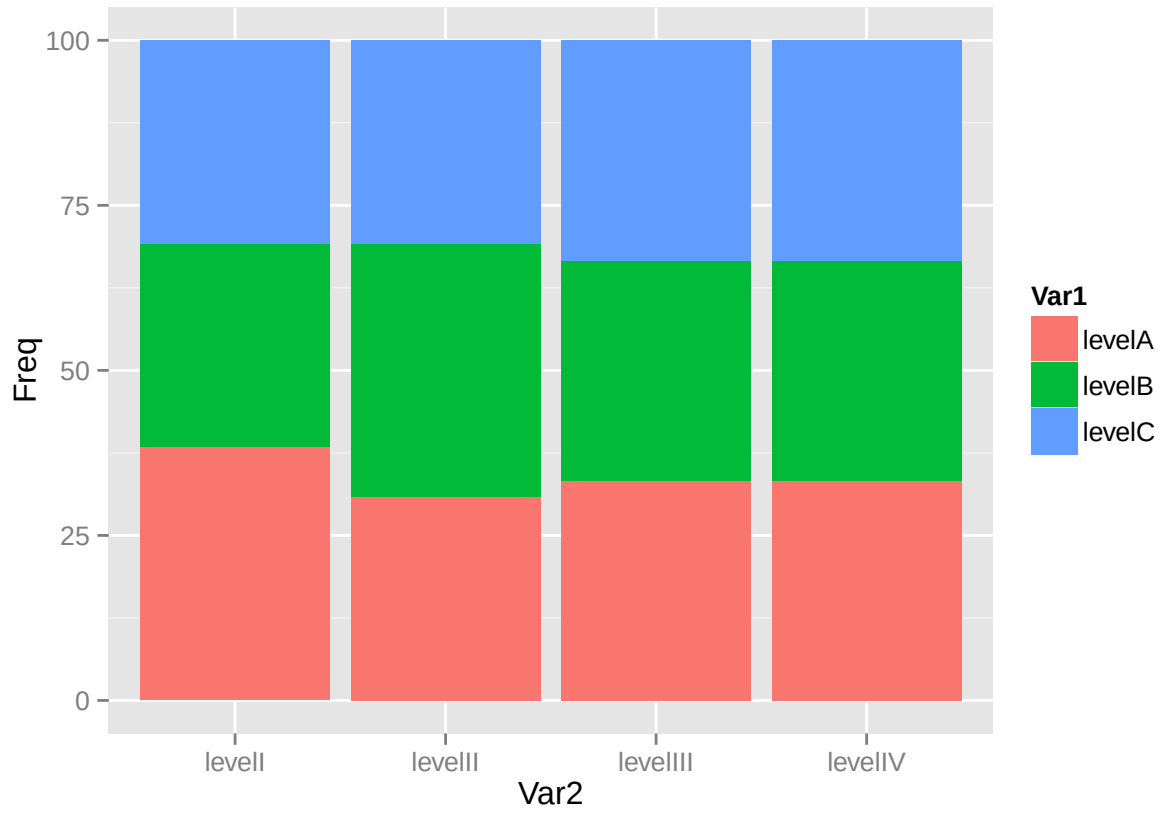
```
ggplot(bartabledat,aes(x=Var2,y=Freq,fill=Var1))+geom_bar(,stat="identity") ## stacked
```



In this graph, we do not have 3 bars, but one whole one for each level, showing the numbers. The difference in code between these two is the position="dodge" function. This is the basic multiple variable bar graph.

Stacked 100%

```
bartableprop =as.data.frame(prop.table(table(simData$FacVar2, simData$FacVar3),2)*100)
ggplot(bartableprop,aes(x=Var2,y=Freq,fill=Var1))+geom_bar(stat="identity") ## Stacked 100%
```

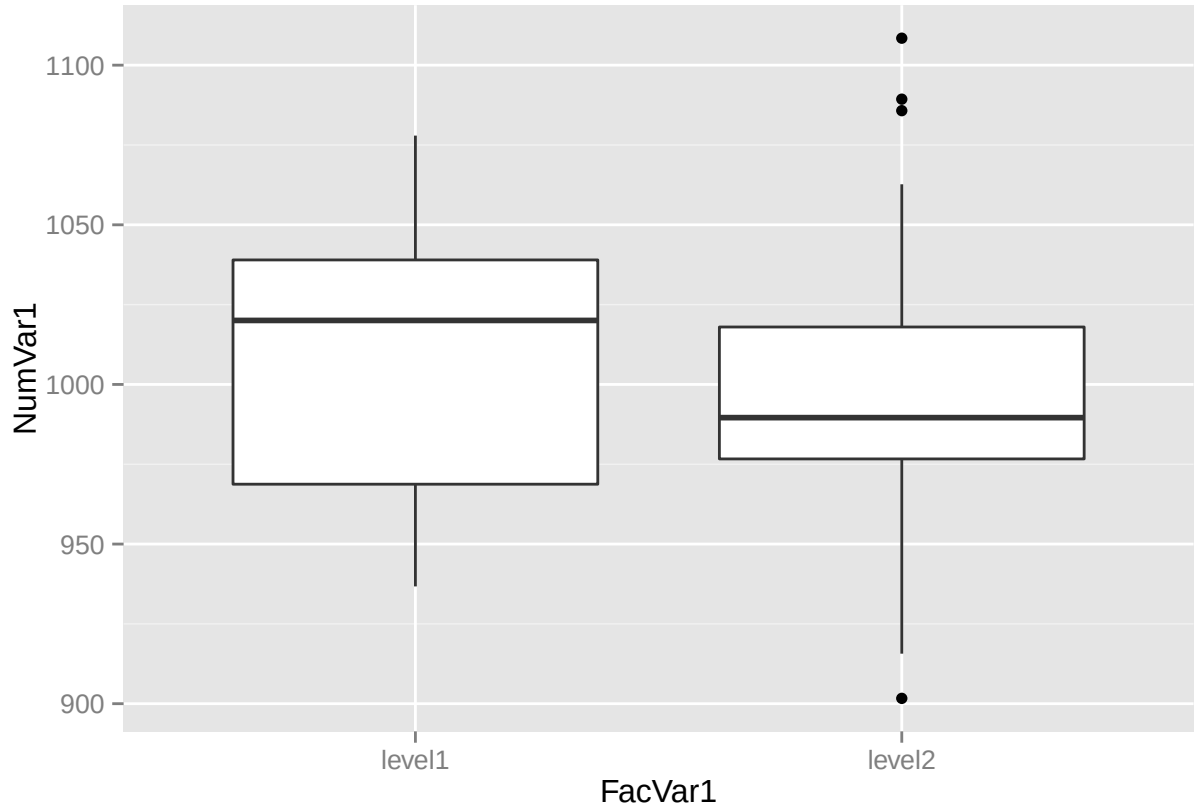


For this graph, we have created another table as a percentage of the whole instead of just the numbers. we then use this as our data set, and enter that into the ggplot code. Now, we have a graph that is maxed out at 100% of each bar, and we see the percentages of each level.

Two Variables: One Factor, One Numeric

Box Plot

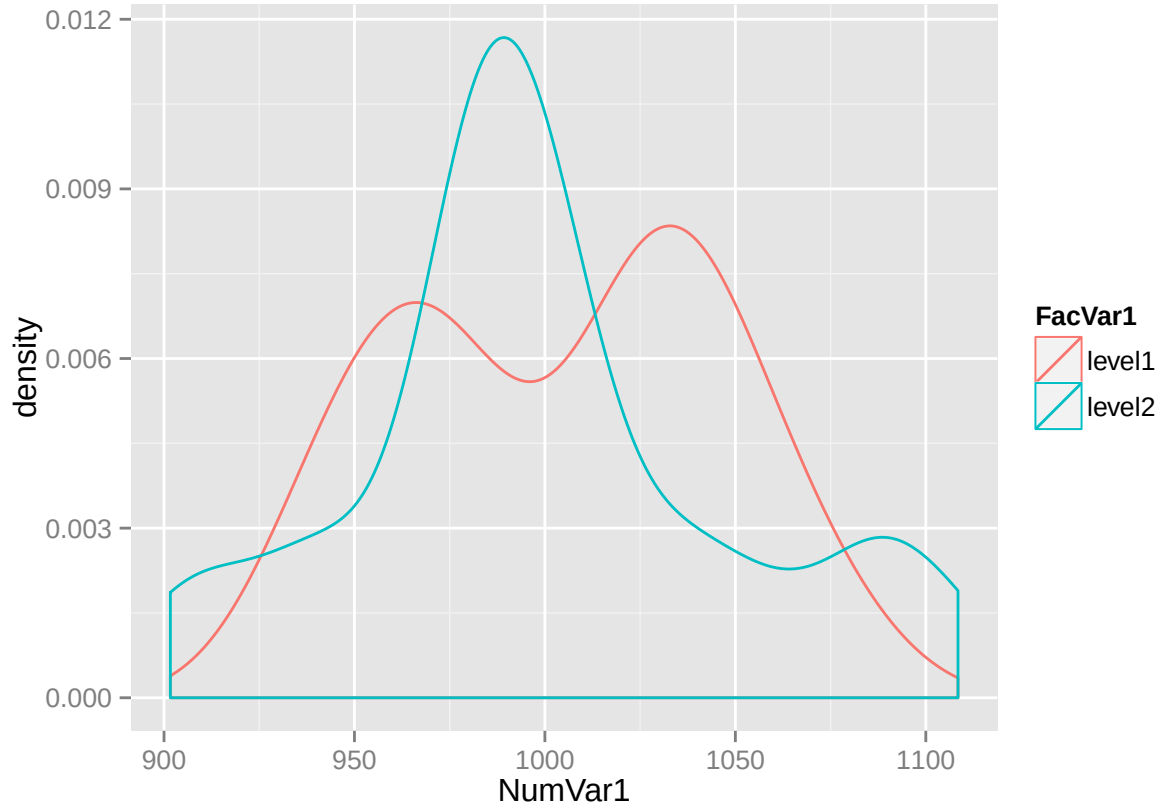
```
## Box plots for the numeric var over the levels of the factor var
ggplot(simData,aes(x=FacVar1,y=NumVar1))+geom_boxplot()
```



This box plot does not need to have a data set built because we can now compare the numbers in numvar1 by the factor variable, instead of the blank frequency like last time.

Frequency Distribution

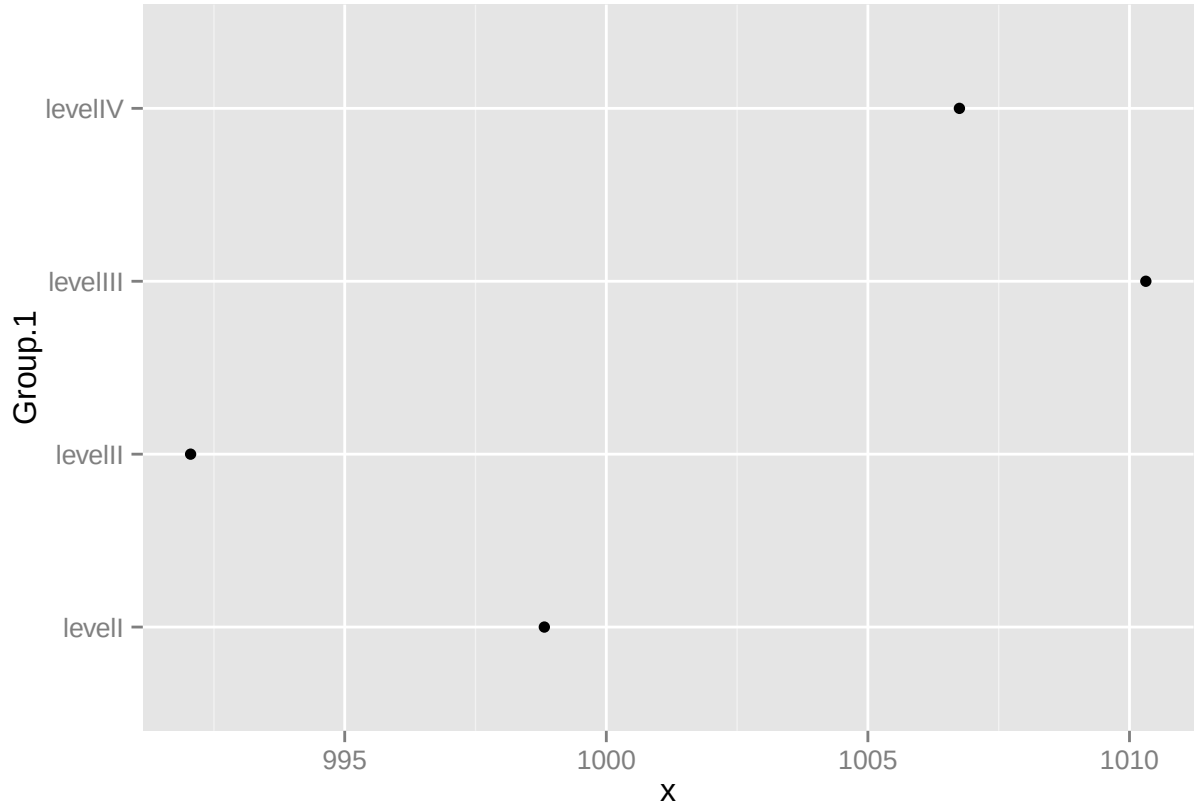
```
## density plot of numeric var across multiple levels of the factor var
ggplot(simData,aes(x=NumVar1,color=FacVar1))+geom_density()
```



What we did with the last graph, we do with this one, for the same reasons.

Mean Plot

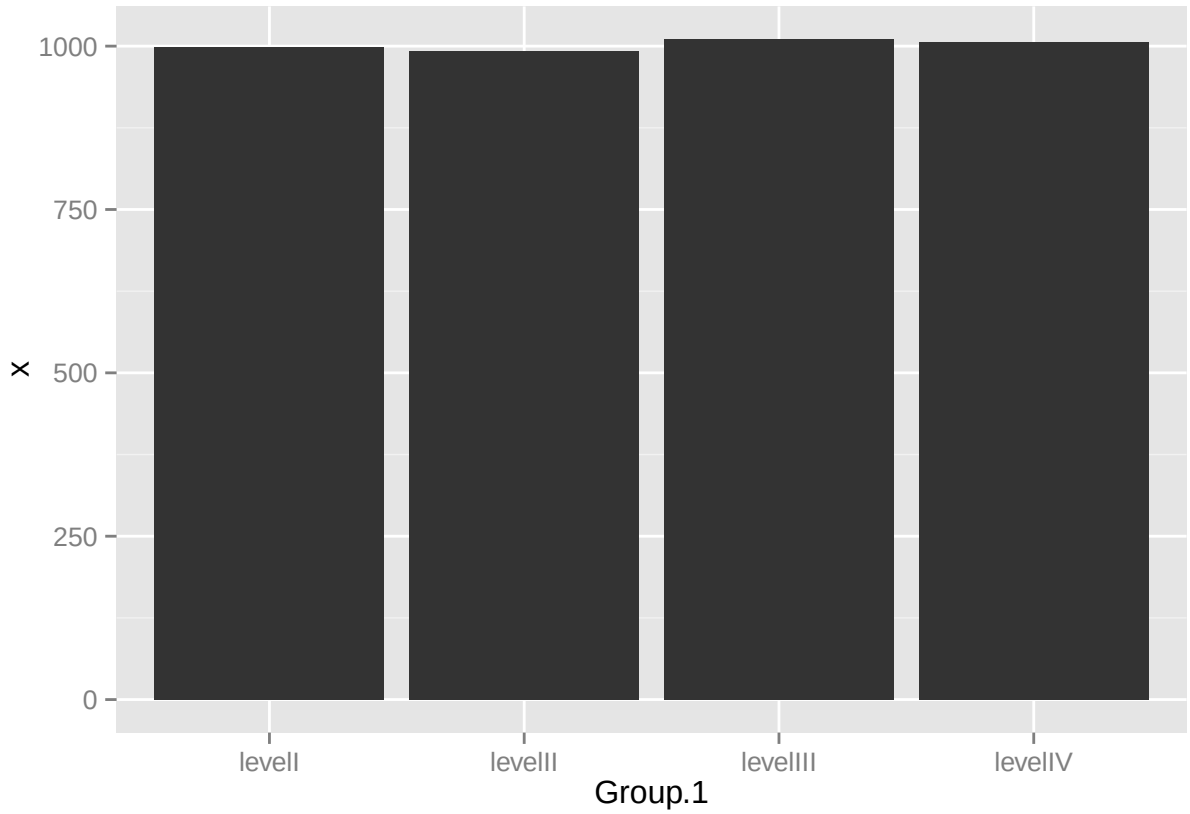
```
## Mean of one numeric var over levels of one factor var
meanagg = aggregate(simData$NumVar1, list(simData$FacVar3), mean)
ggplot(meanagg,aes(x=Group.1,y=x))+geom_point()+coord_flip() ## Dot Chart equivalent
```



Meanagg is the mean of the plots, which we then use as the data set for the dot plot. We use coord_flip to have it sideways, instead of vertical means. You can do this with bar, scatter, or any other plot you wish if it makes your data easier to read.

Mean Bar

```
ggplot(meanagg,aes(x=Group.1,y=x))+geom_bar(stat="identity") ## Bar plot
```

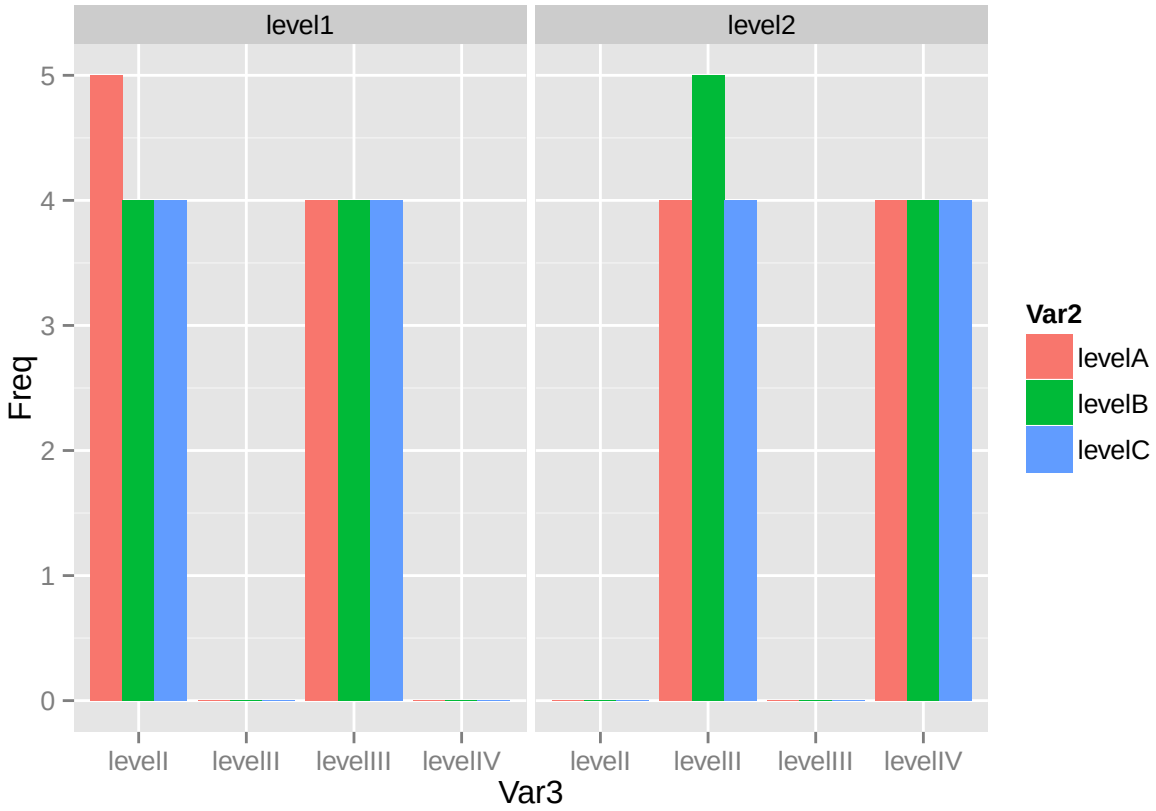


Same data as the last, just a bar plot this time. Using Meanagg as the source of the data.

Three Factor Variables

Bar

```
Threebartable = as.data.frame(table(simData$FacVar1, simData$FacVar2, simData$FacVar3)) ## CrossTab
ggplot(Threebartable,aes(x=Var3,y=Freq,fill=Var2))+geom_bar(position="dodge",stat="identity")+facet_wra
```

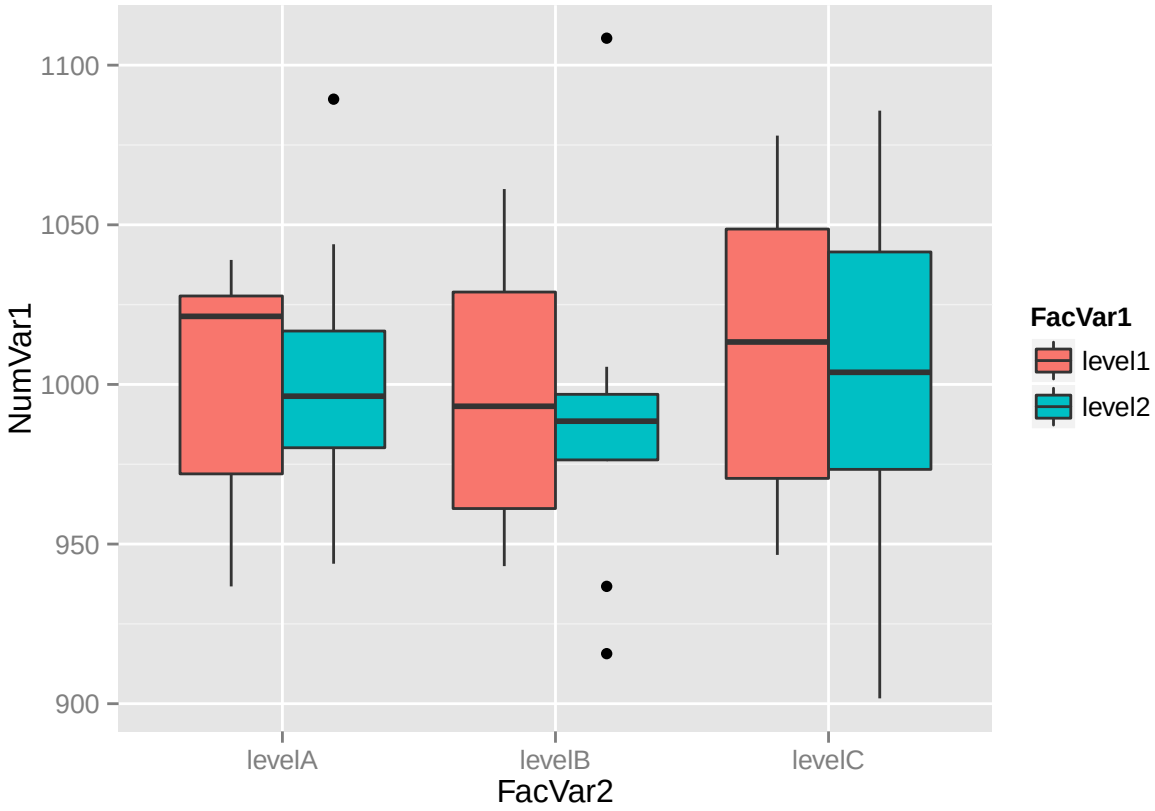


Facet_wrap adds the variable Var1 as a second graph, and the fill for the aesthetic is now the different variables in Var2. This uses the dataframe for Threebartable, which we create in the first row.

Three Variable: One Numerical, Two Factor

Box Plot

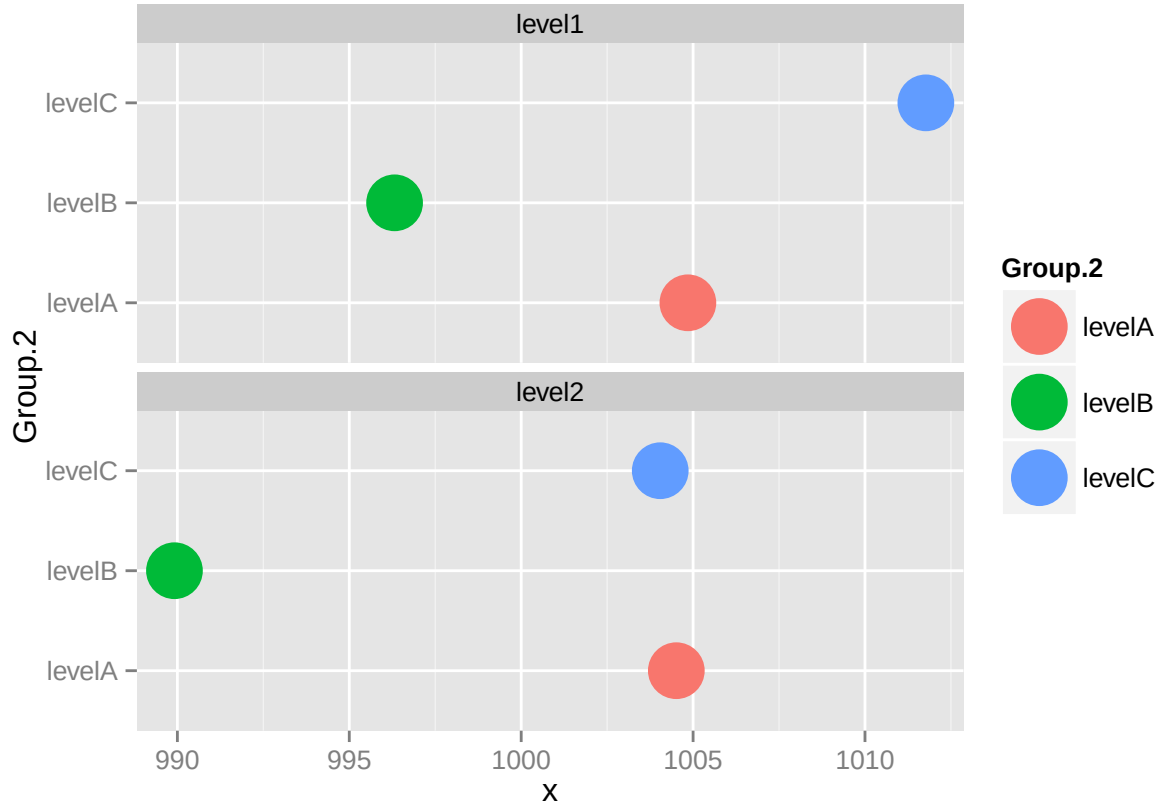
```
## boxplot of NumVar1 over an interaction of 6 levels of the combination of FacVar1 and FacVar2
ggplot(simData,aes(x=FacVar2,y=NumVar1, fill=FacVar1))+geom_boxplot()
```



This graph uses numvar as the Y axis, fills with FacVar1, and has the graphs separated by facvar2. Using Fill to put in another variable is a very valuable tool, and something that makes ggplot2 a worthwile tool to learn.

Mean Plot

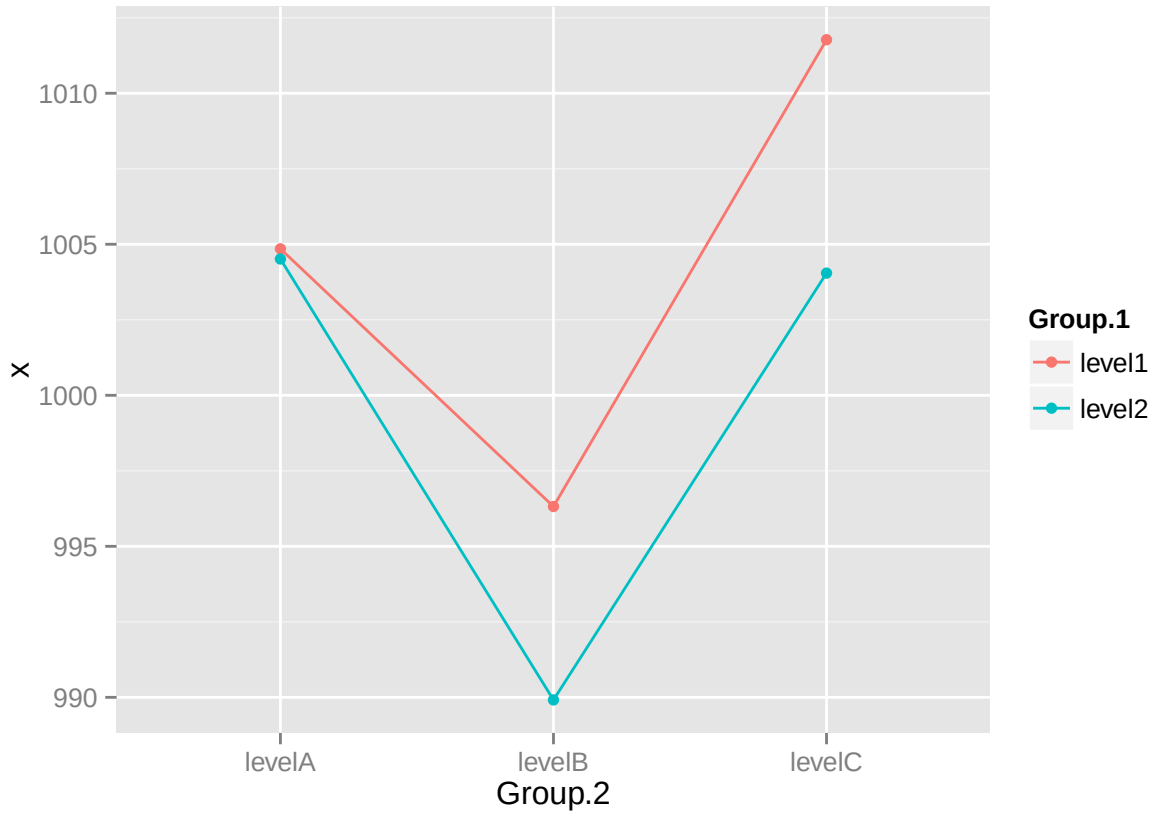
```
## Mean of 1 Numeric over levels of two factor vars
meanaggg = aggregate(simData$NumVar1, list(simData$FacVar1, simData$FacVar2), mean)
## Dot Chart equivalent
ggplot(meanaggg,aes(x=Group.2,y=x,color=Group.2,))+geom_point(size=10)+coord_flip()+facet_wrap(~Group.1
```



For this one, we use a data set with the means, then make a chart with the data. Color is used to differentiate. Made larger with size=10 to make it easier to see.

Line Chart

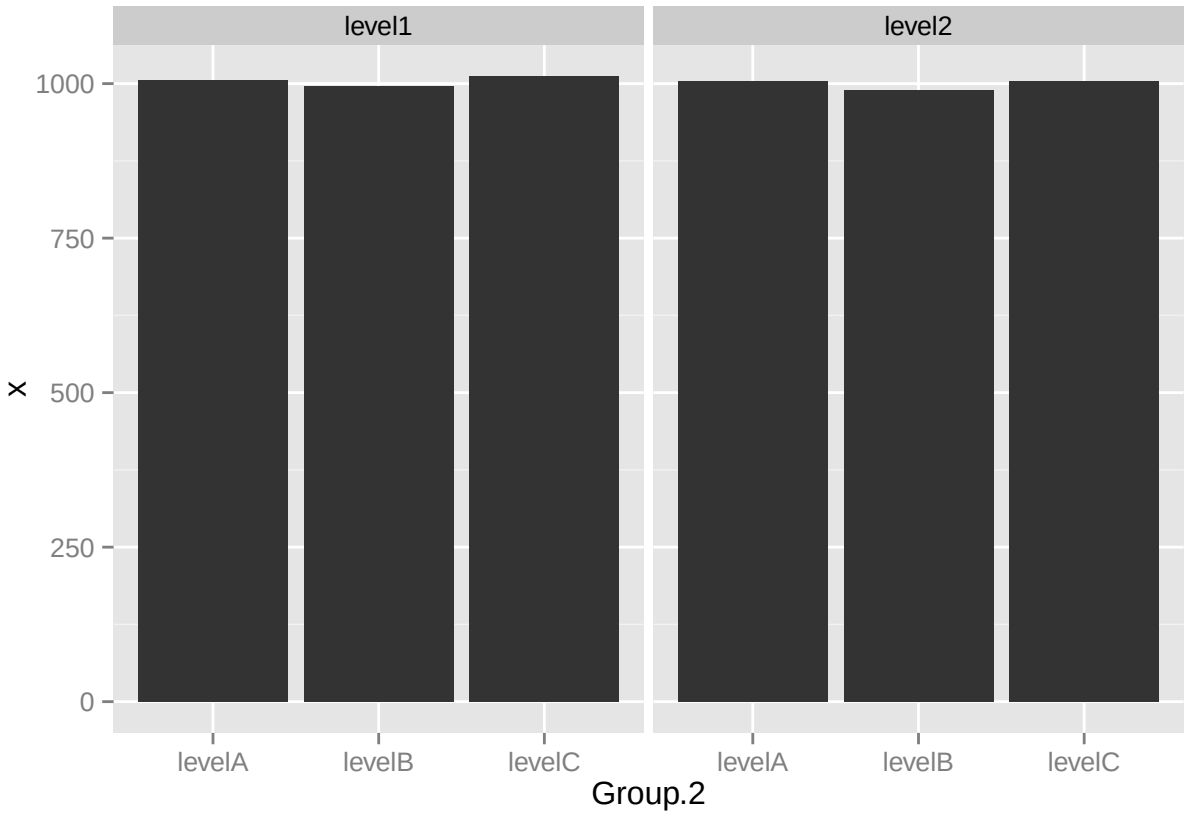
```
## Interaction chart - line chart
ggplot(meanaggg,aes(x=Group.2,y=x,color=Group.1, group=Group.1))+geom_point()+geom_line()
```



Using the meanaggg set, then using lines and points

Bar Plot

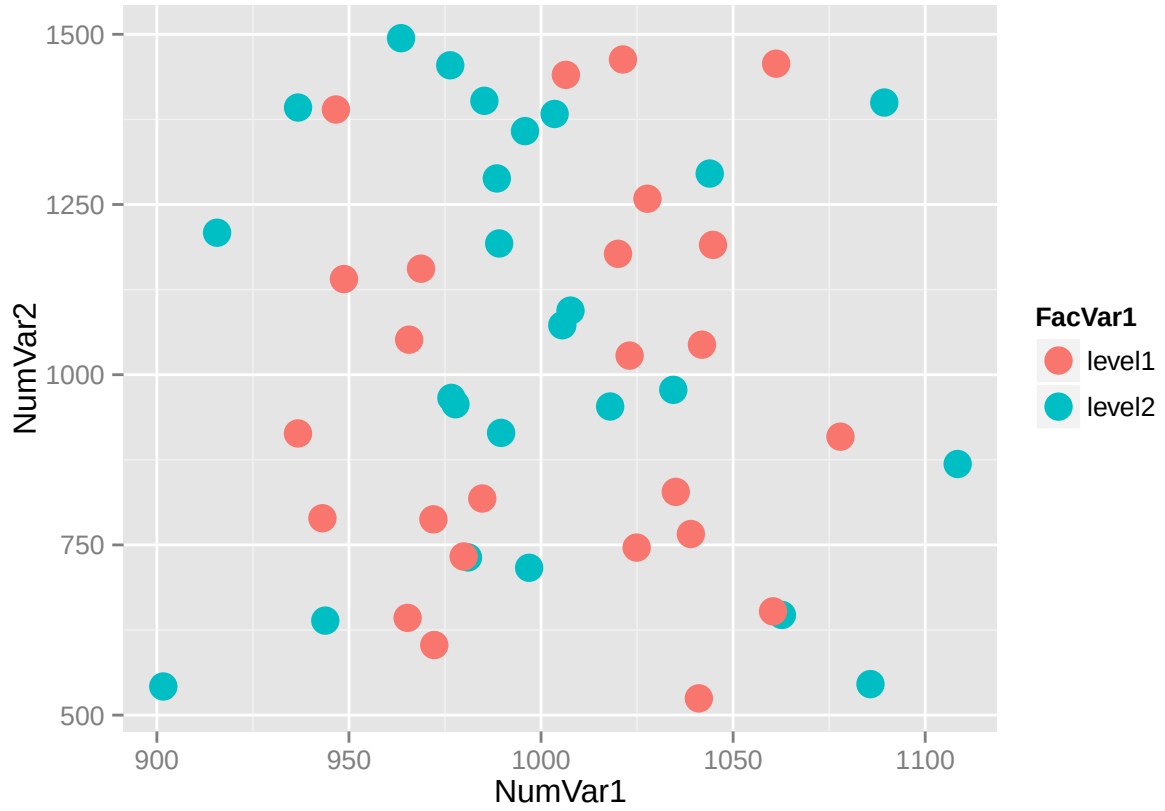
```
## And bar plot
ggplot(meanaggg,aes(x=Group.2,y=x))+geom_bar(stat="identity")+facet_wrap(~Group.1)
```



Two Numeric, One Factor

Scatter

```
## Scatter plot with color identifying the factor variable
ggplot(simData,aes(x=NumVar1,y=NumVar2,color=FacVar1))+geom_point(size=5)
```

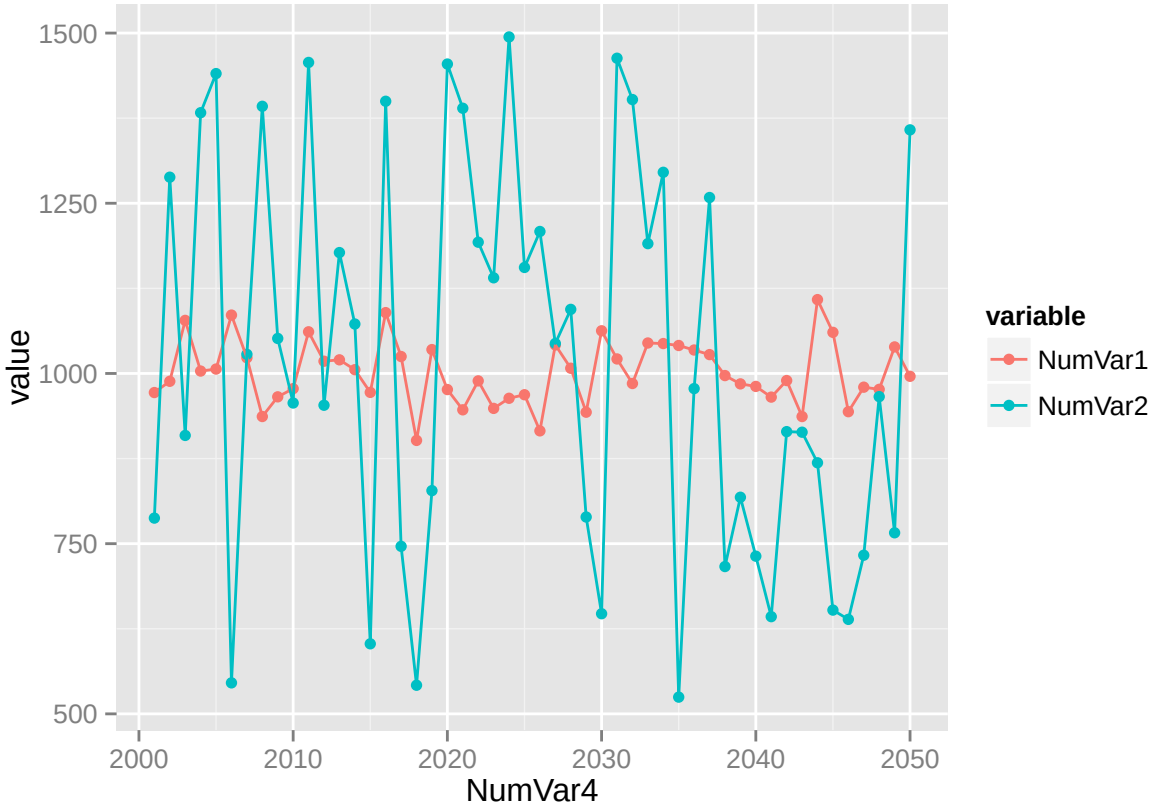


Once again, using color as a variable can add another layer of information to your graph, and with ggplot2, its very easy

Three Numeric

Line Plot

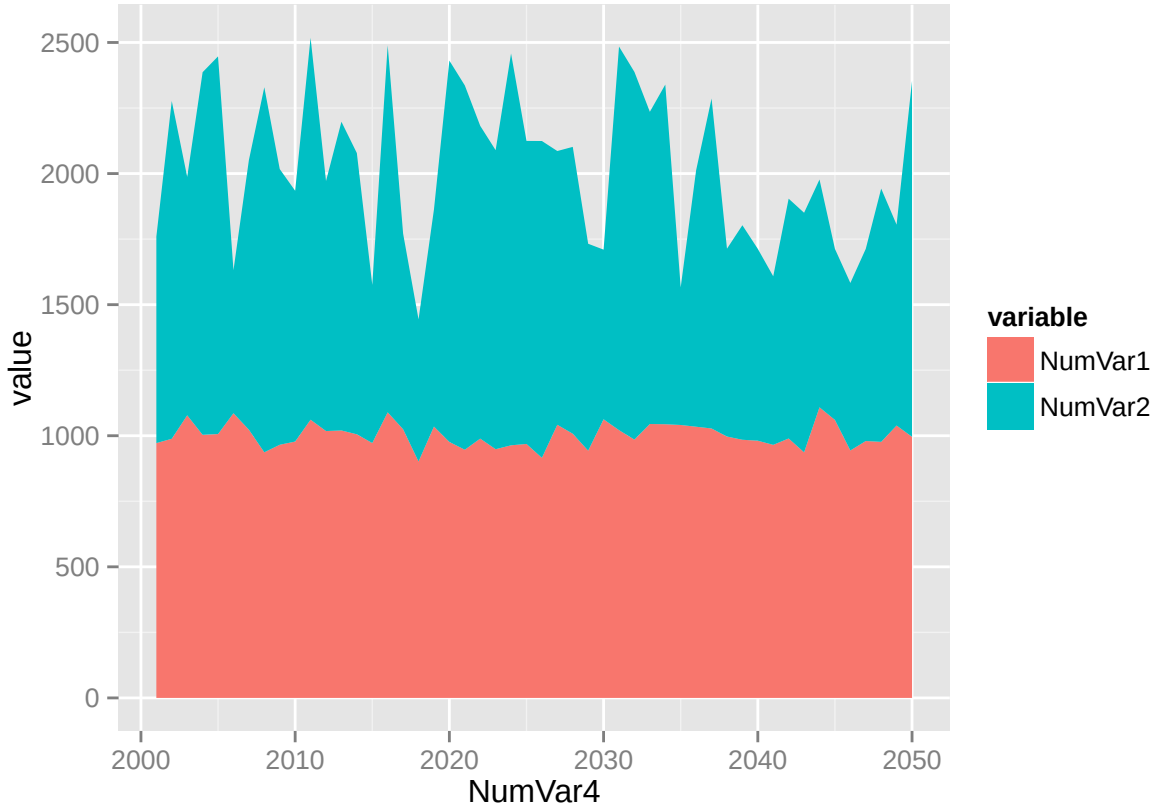
```
## NumVar4 is 2001 through 2050... possibly, a time variable - use that as the x-axis
simtmpp=simData[,c(4,5,7)]
simtmppmelt=melt(simtmpp,id=c("NumVar4"))
ggplot(simtmppmelt,aes(x=NumVar4,y=value,color=variable,group=variable))+geom_point()+geom_line()
```



Using NumVar4, which is basically time, we now have time series data. Still very easy to use.

Stacked Area Graph

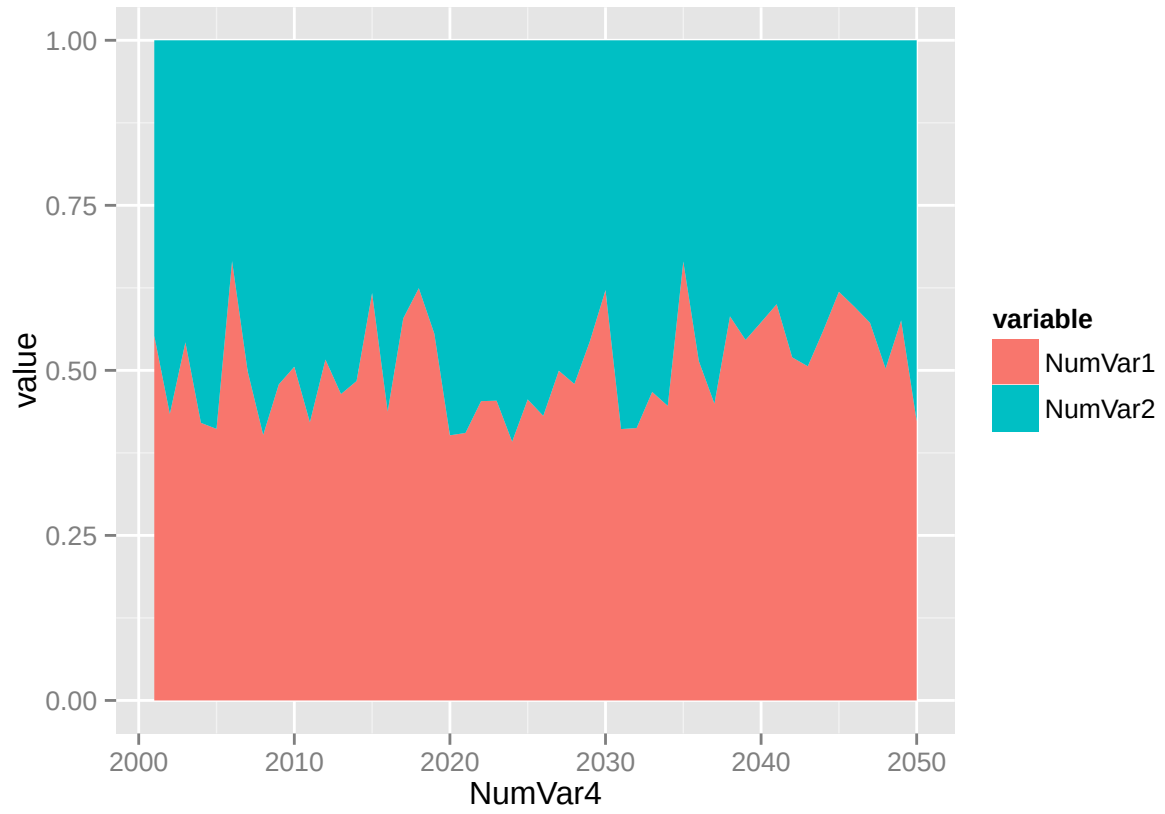
```
## Extra: Stacked Area Graph
ggplot(simtmppmelt,aes(x=NumVar4,y=value,fill=variable))+geom_area(position="stack")
```



Stacks the level of Numvar2 on top of Numvar1, which is not a very good idea to do, since the peak levels of numvar2 are not on a level plane, which makes it hard to tell a true peak or trough.

Stacked 100% Graph

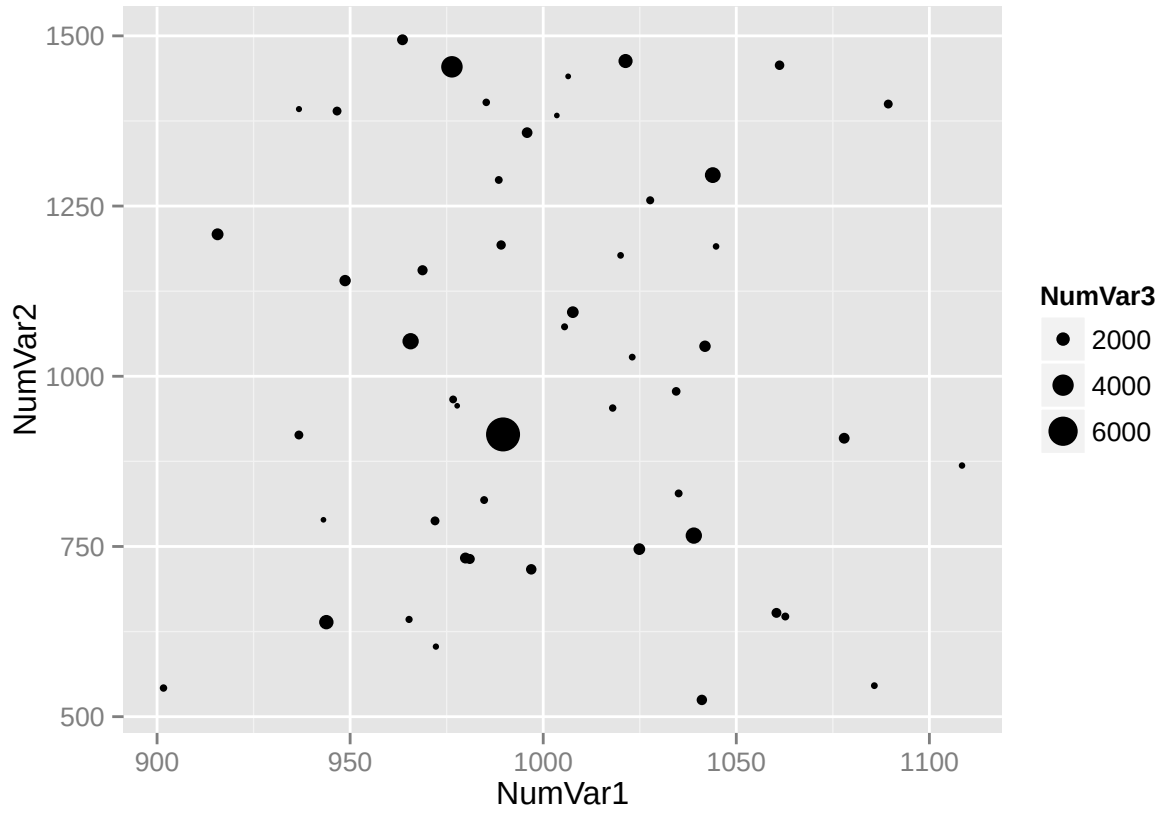
```
## Extra: 100% stacked area graph
ggplot(simtmppmelt,aes(x=NumVar4,y=value,fill=variable))+geom_area(position="fill")
```



I have no idea what this would be used for. Looks fancy, though.

Bubble Plot

```
## ## Bubble plot - scatter plot of NumVar1 and NumVar2 with individual observations sized by NumVar3
ggplot(simData,aes(x=NumVar1,y=NumVar2,size=NumVar3))+geom_point()
```



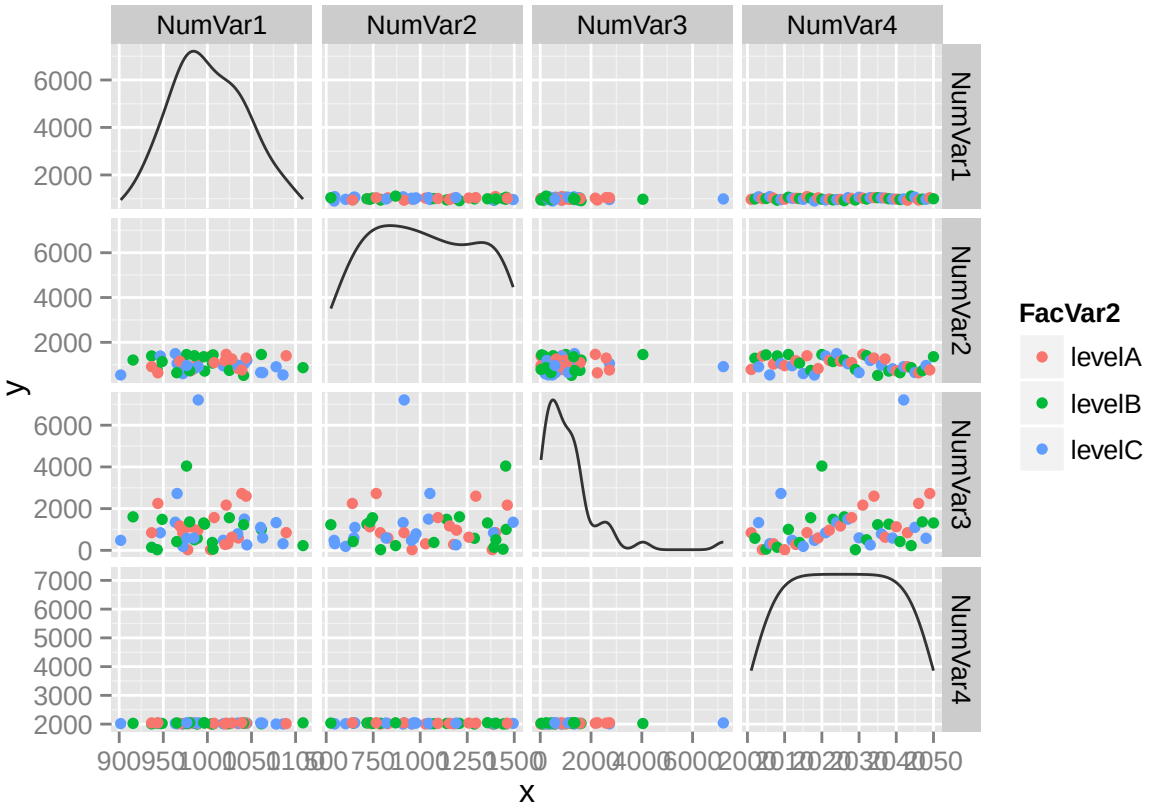
Matrix of Scatterplots

```
#Thanks to Gaston Sanchez for the function: http://gastonsanchez.wordpress.com/2012/08/27/scatterplot-m
 makePairs <- function(data)
{
  grid <- expand.grid(x = 1:ncol(data), y = 1:ncol(data))
  grid <- subset(grid, x != y)
  all <- do.call("rbind", lapply(1:nrow(grid), function(i) {
    xcol <- grid[i, "x"]
    ycol <- grid[i, "y"]
    data.frame(xvar = names(data)[ycol], yvar = names(data)[xcol],
               x = data[, xcol], y = data[, ycol], data)
  }))
  all$xvar <- factor(all$xvar, levels = names(data))
  all$yvar <- factor(all$yvar, levels = names(data))
  densities <- do.call("rbind", lapply(1:ncol(data), function(i) {
    data.frame(xvar = names(data)[i], yvar = names(data)[i], x = data[, i])
  }))
  list(all=all, densities=densities)
}

## expanding numeric columns for pairs plot
gg1 = makePairs(simData[,4:7])

## new data frame
simDatabig = data.frame(gg1$all,simData[,1:3])

## pairs plot
ggplot(simDatabig, aes_string(x = "x", y = "y")) +
  facet_grid(xvar ~ yvar, scales = "free") +
  geom_point(aes(colour=FacVar2), na.rm = TRUE) +
  stat_density(aes(x = x, y = ..scaled.. * diff(range(x)) + min(x)),
               data = gg1$densities, position = "identity",
               colour = "grey20", geom = "line")
```



That is a graph. And some code. There is a ton goingon here, making all of these graphs, how they stack, the colors, and more. There is a ton of information here, and it could be used in many ways.

Conclusion

As you can see, ggplot2 is a helpful tool to making your graphs. With its simplicity, customizability, and applicability, it makes a ton of sense to use in your work life.