Data Visualisation in R with ggplot2

Recommended reading before we begin :

http://byrneslab.net/classes/biol607/readings/wickham_layered-grammar.pdf

What you are **NOT** expected to gather from the reading :

- Learn ggplot2
- Learn any form of coding

What you **ARE** expected to gather :

- Understand the layered grammar fundamentals of graphics

I'm assuming once you start reading this you have given at least a cursory reading to the text mentioned above. You must have realized that layered grammar of graphics is not a complex frame work for statistical graphics. Instead its an attempt to streamline traditional way of having a distinct name for each kind of graphics. Basic idea behind this is that almost all kind of graphics are manifestations of different values taken by underlying layers which are :

- data layer and Aesthetic Mapping
- Geometric Object Layer
- Scales and grouping in the data
- Statistical Transformation
- Coordinate Axes

We'll start with data layer and aesthetics mapping, subsequent layer discussion will not be discussed under separate subheadings, rather our discussion will evolve into usage of those layers and we'll discuss details as and when we come to it. From your reading of the recommended article, you might have also come across idea of defaults and their inheritance between layers, we'll take up that once we are done with basic graphics in ggplot2. Lets begin.

Before starting the discussion, you need to understand that graphics using ggplot2 are not going to be some alien plots, they are still going to be same old scatter plots, bar plots , histograms etc, we are still going to have x -axes , y-axes etc, our plots are going to be two dimensional. Our approach to arrive at those plots has just become more generic now , that's all.

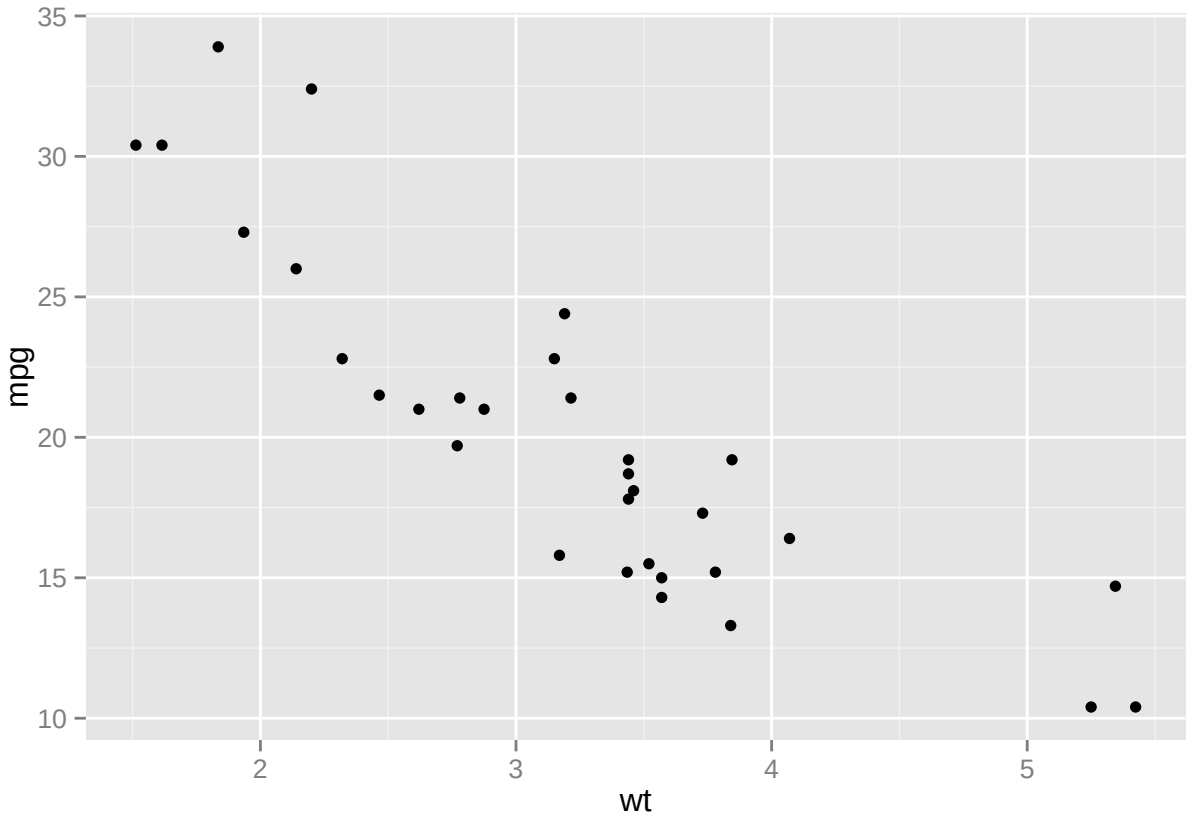
All graphics that you generate are made up of some aesthetics and data mapped on to those aesthetics. For example a scatter plot is nothing but a variable mapped to x axis and another to y axis. Lets define such a data layer with only these mappings and make a scatter plot.

```
library(ggplot2)
p=ggplot(mtcars,aes(x=wt,y=mpg))
```

There are no graphics element yet, we have only created our data layer with some basic aesthetic mapping. If you simply write `p` and try to run, you'll get an appropriate warning stating `Error: No layers in plot`.

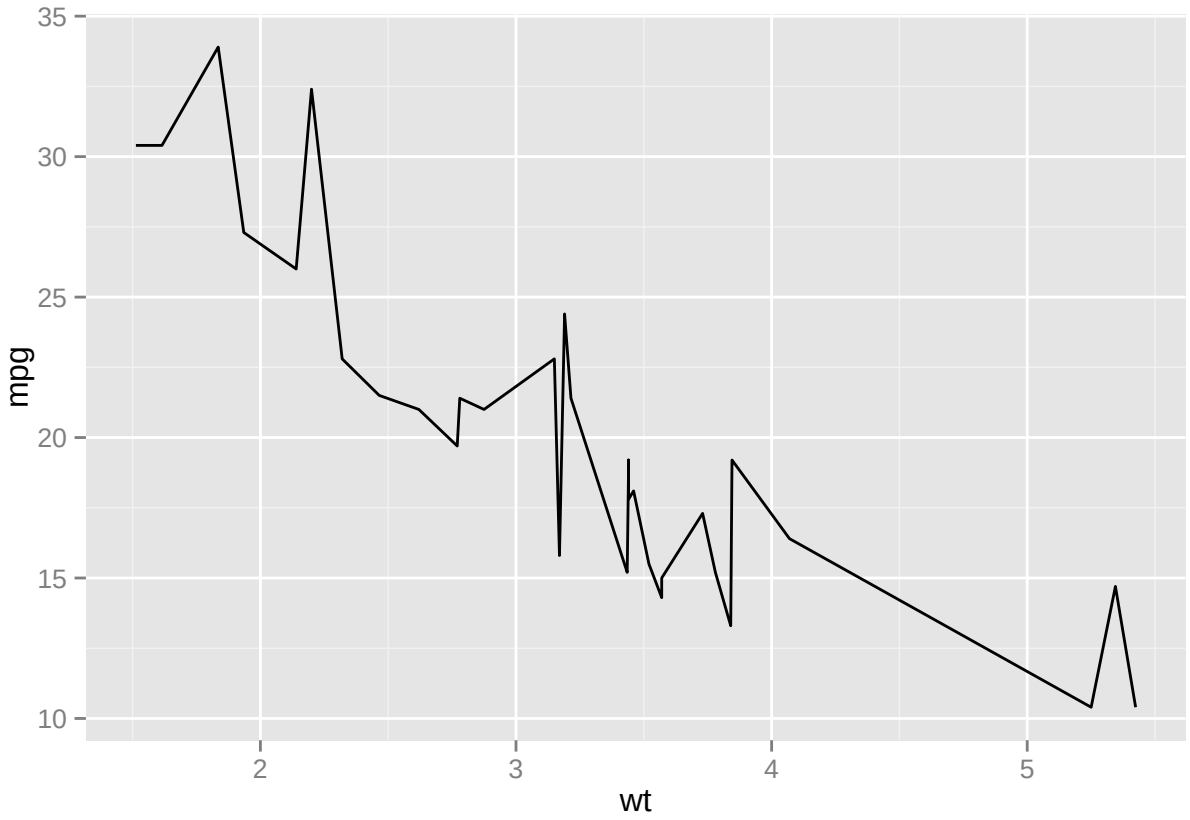
We need to add a geometric object layer on top of this to create a graphic. Lets start with a simple scatter plot by showing our data using points.

```
p+geom_point()
```



And we have our good old scatter plot. While typing `geom_point` , you might have got a long list of these `geom` options, indeed there are many. But not all of them will make sense with current aesthetic mapping, in fact in some cases you'll get error stating some missing aesthetics mapping. Some will produce a plot which is useless. Lets see more examples of geometric object which make sense *[or not]* with current aesthetics mapping.

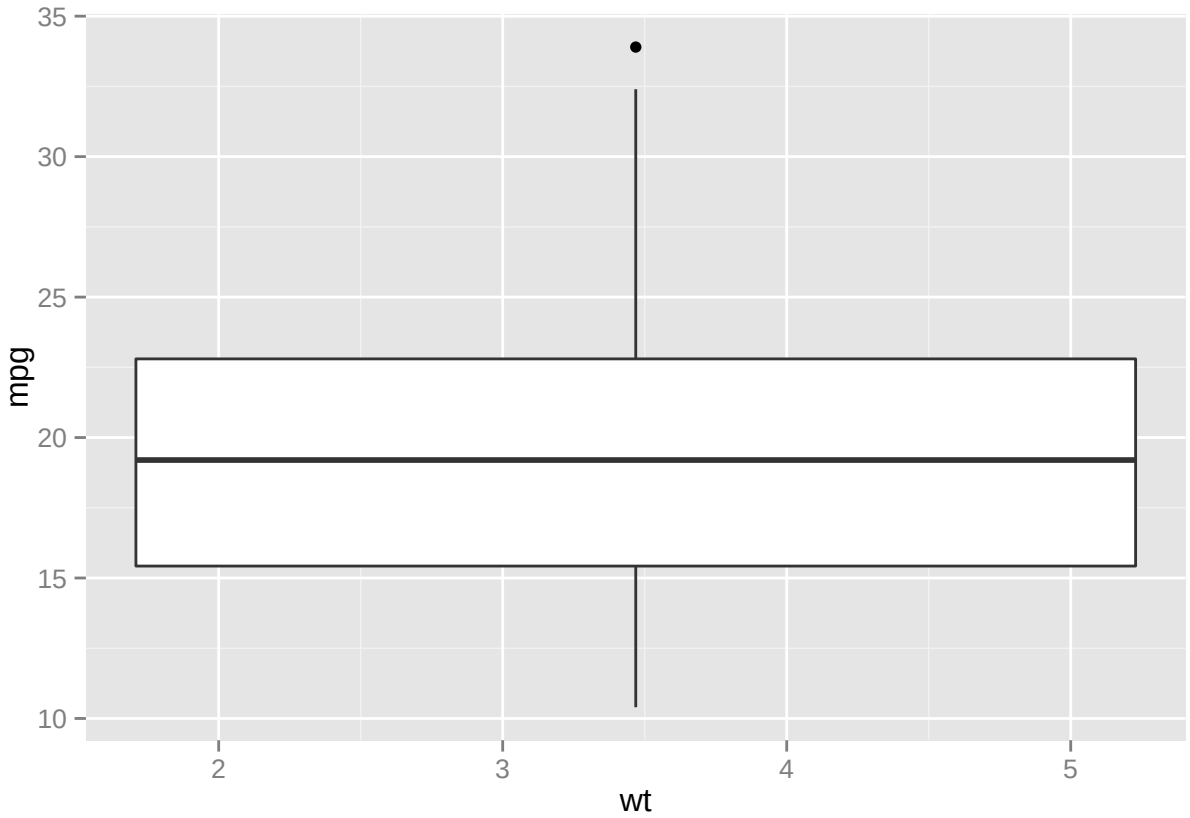
```
p+geom_line()
```




```
p+geom_rect()
```

```
## Error: geom_rect requires the following missing aesthetics: xmin, xmax, ymin, ymax
```

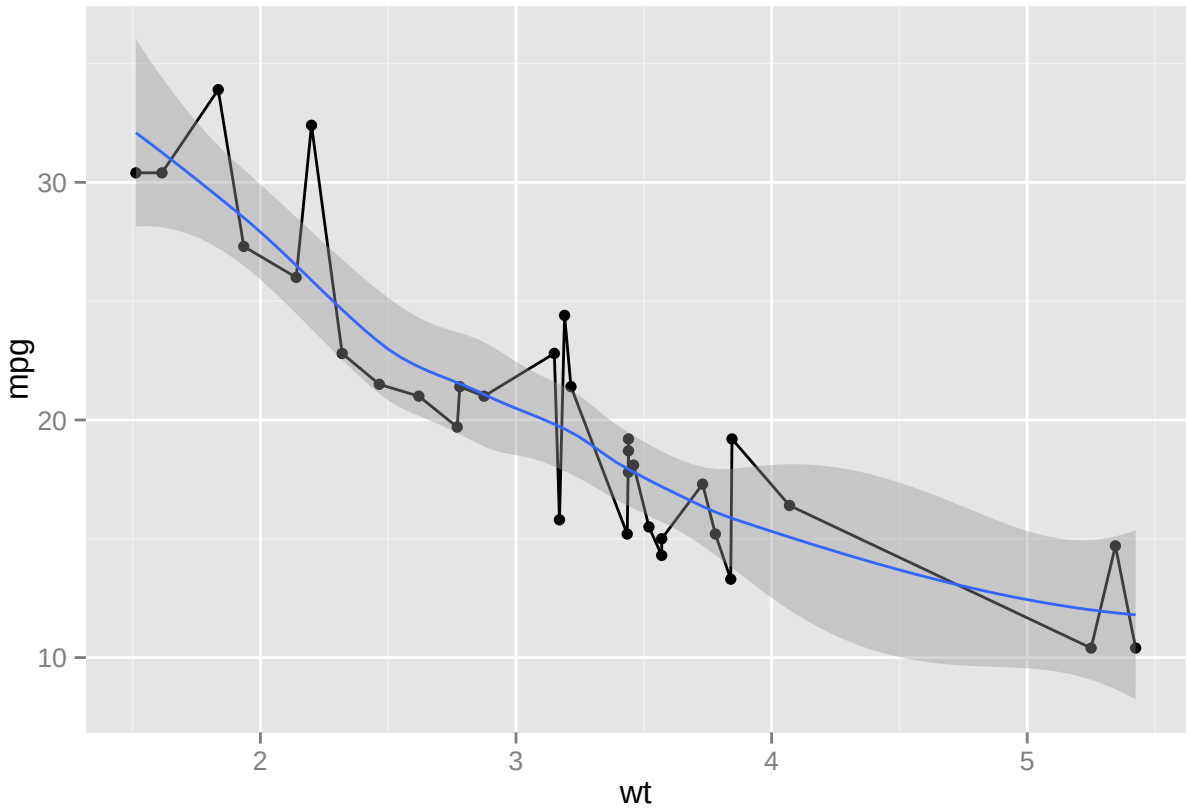
```
p+geom_boxplot()
```



Example of boxplot is one which gives you a result which is nonsensical in this context. You can not really have a boxplot taken two continuous numeric variables mapped to either axes. What did we learn here ? Although `"Colorless green ideas sleep furiously"` is grammatically correct but a useless nonsense sentence none the less.

You can add multiple kind of geometric object layers at once as well, as long as it makes sense to add them together.

```
p=ggplot(mtcars,aes(x=wt,y=mpg))
p+geom_point()+geom_line()+geom_smooth()
```

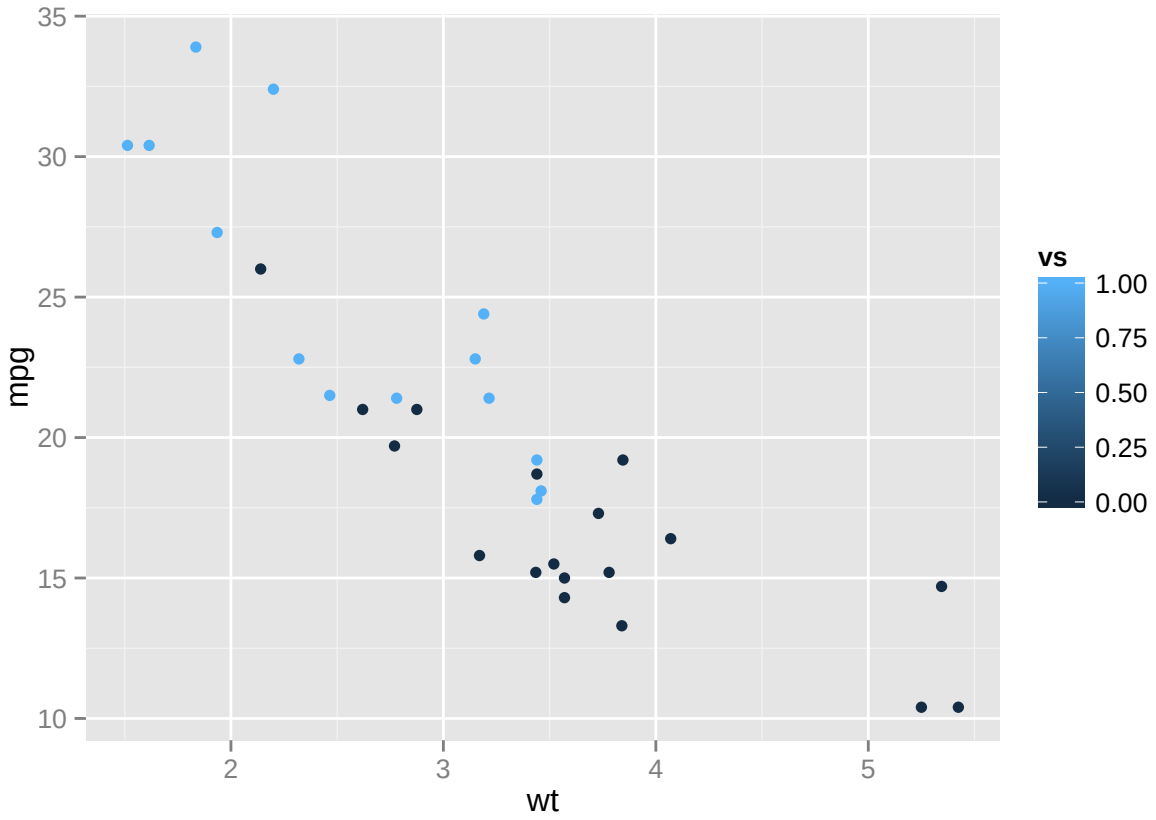


In addition to x,y axes there are other common aesthetics in a plot:

- Color
- Shape
- Size

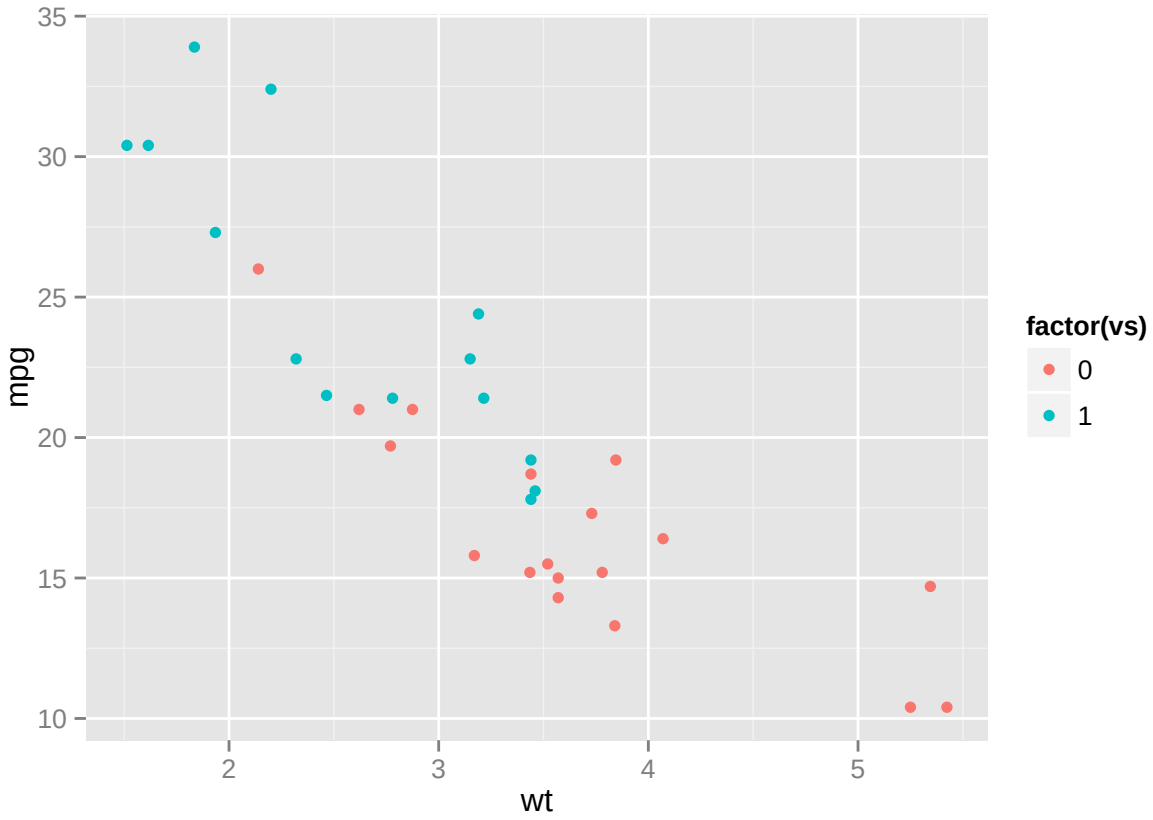
Again, these aesthetics are going to be mapped to some data. Lets build on our example of scatter plot.

```
p=ggplot(mtcars,aes(x=wt,y=mpg,color=vs))
p+geom_point()
```

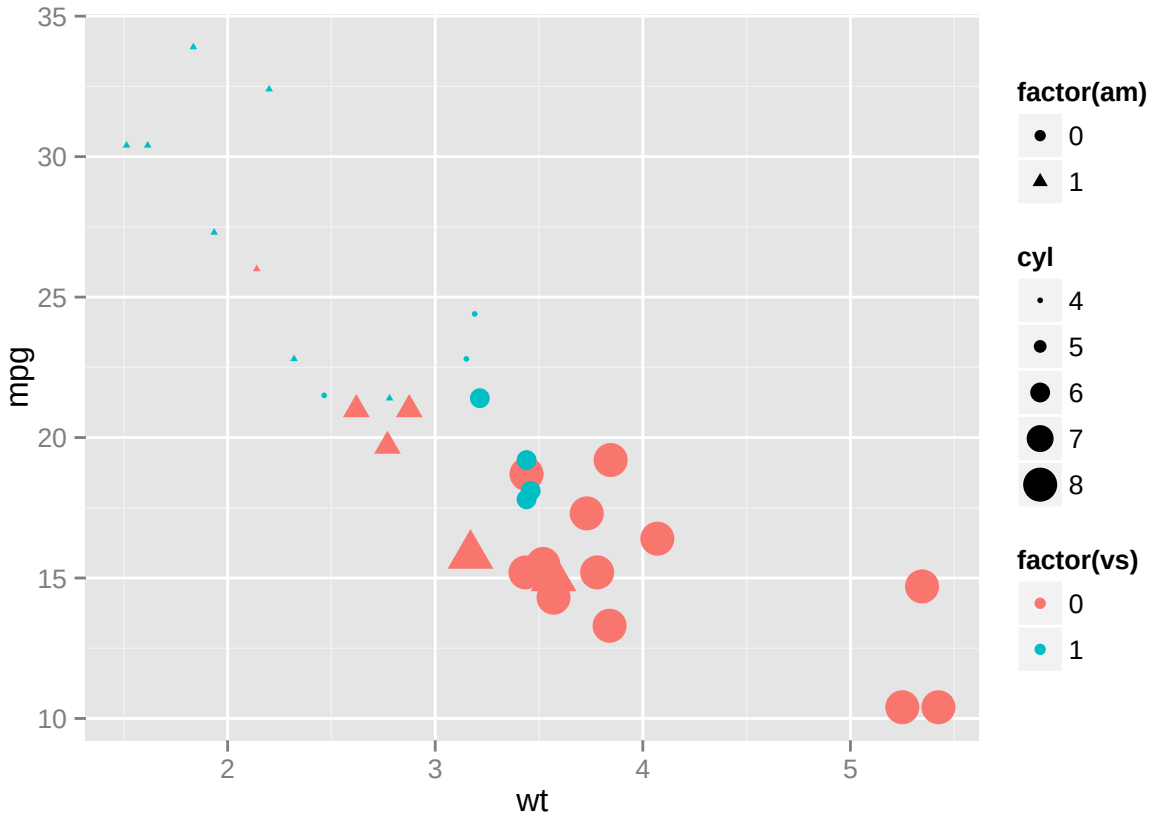


You can see that points have been colored according to value of variable `vs`. A legend has been added to plot automatically, which also tells you that variable `vs` is taken as continuous numeric hence the color scale. We can convert variable `vs` to factor type and legend will change.

```
p=ggplot(mtcars,aes(x=wt,y=mpg,color=factor(vs)))
p+geom_point()
```




```
p=ggplot(mtcars,aes(x=wt,y=mpg,color=factor(vs),shape=factor(am),size=cyl))
p+geom_point()
```



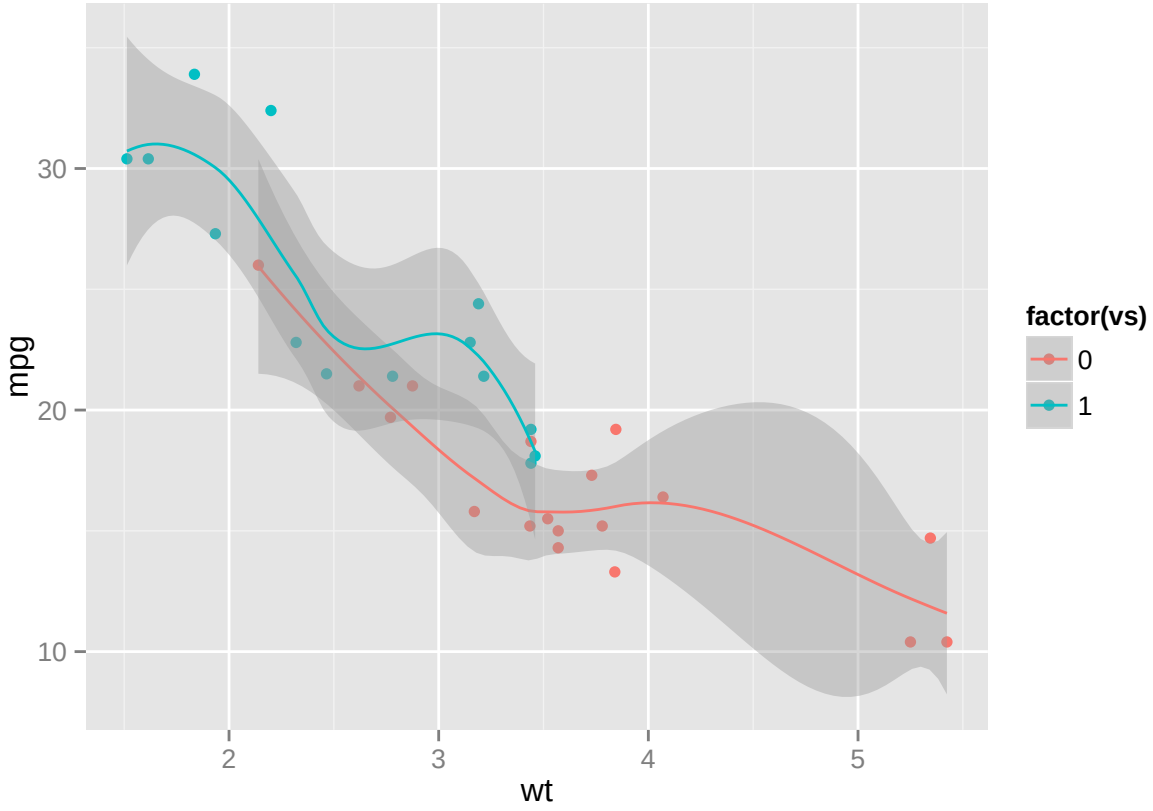
Now these aesthetic that we see, can be mapped to different geometric objects, few examples of which we have seen earlier. As you have seen, there are many geometric objects. All of these objects can have there own aesthetics as well. If you haven't noticed yet, so-far all `geom_*` that we have added, had no inputs, they were inheriting all the aesthetic mapping from the data layer.

I am listing down some aesthetics of the common geometric objects

- points
- – point shape
- – point size
- lines
- – line type
- – line weight
- bars
- – y minimum
- – y maximum
- – fill color
- – outline color
- text
- – label value

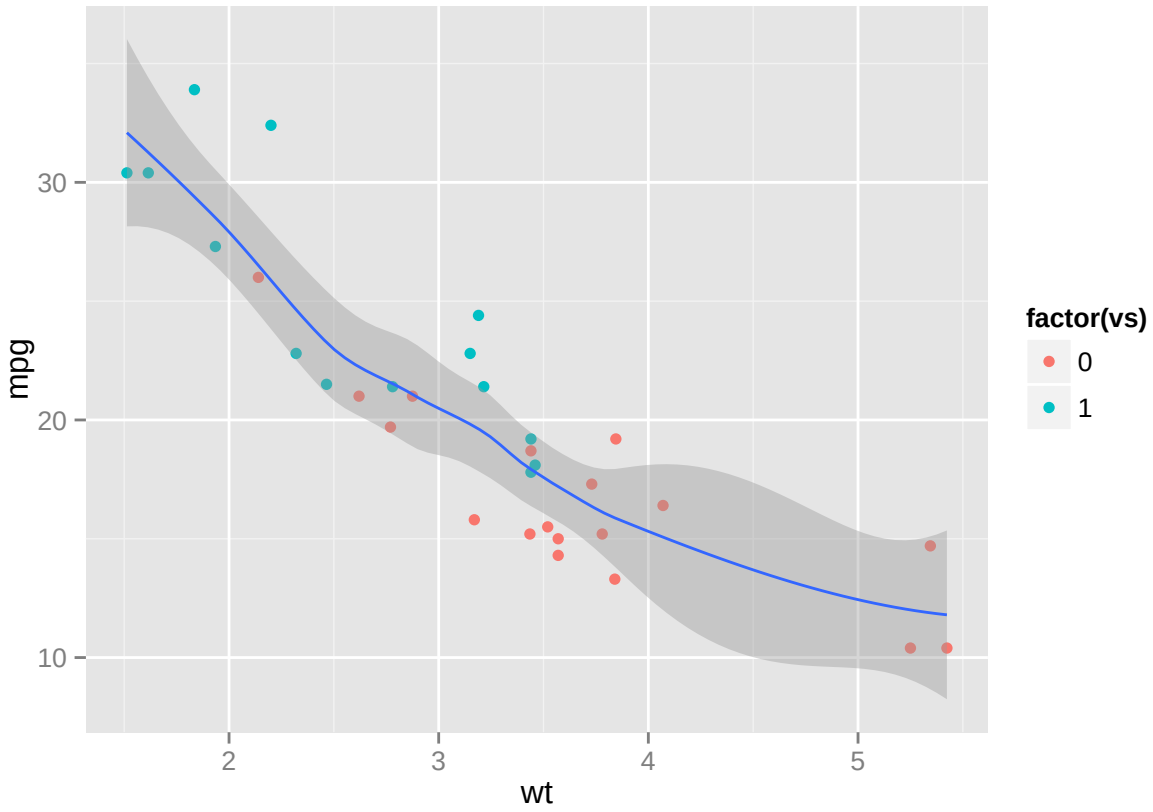
To understand better about inheritance of aesthetics, lets assign color to our data layer add both point and a smoothing layer to it.

```
p=ggplot(mtcars,aes(x=wt,y=mpg,color=factor(vs)))
p+geom_point()+geom_smooth()
```



You can see that data is automatically grouped according to colors. This grouping is passed to both the geometric layers by default. If you assign these aesthetics to geometric layers individually in stead putting it in top layer, data is grouped for those layers only.

```
p=ggplot(mtcars,aes(x=wt,y=mpg))
p+geom_point(aes(color=factor(vs)))+geom_smooth()
```

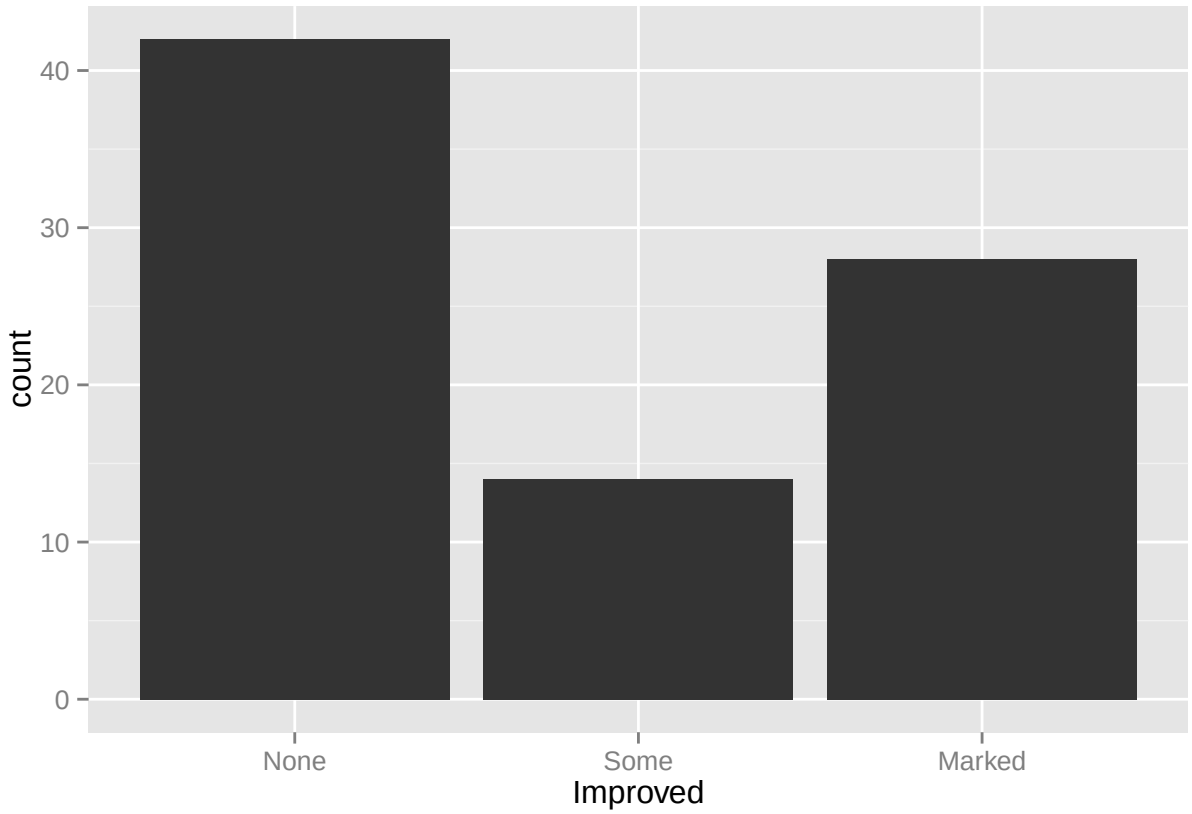


you can see that points are colored by the groups defined by color aesthetic but there is no grouping in smoothing layer. Lets look at few more example with bar geometric object using a new data `Arthritis` present in package `vcd`.

```
##   ID Treatment  Sex Age Improved
## 1 57   Treated Male  27     Some
## 2 46   Treated Male  29     None
## 3 77   Treated Male  30     None
## 4 17   Treated Male  32   Marked
## 5 36   Treated Male  46   Marked
## 6 23   Treated Male  58   Marked
```

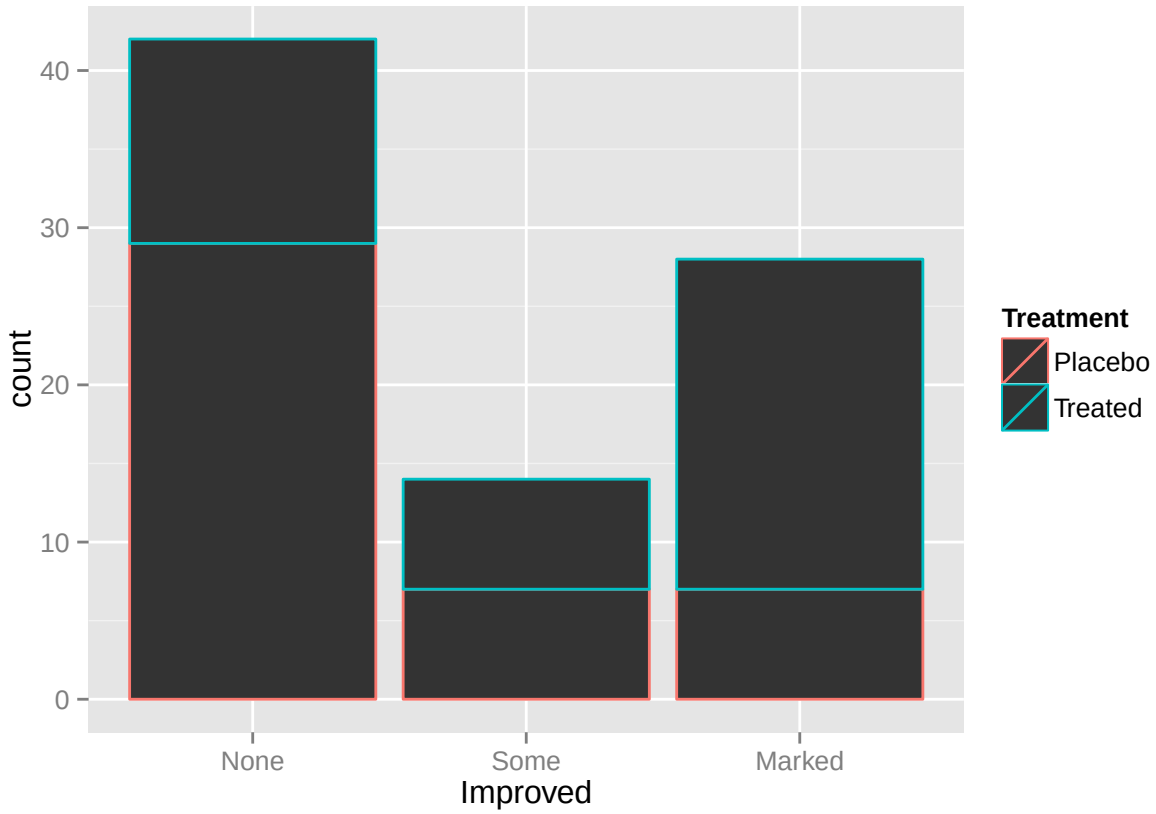
From here on wards we are not going to define data layer separately, we'll add it in the same line

```
ggplot(Arthritis, aes( Improved))+geom_bar()
```

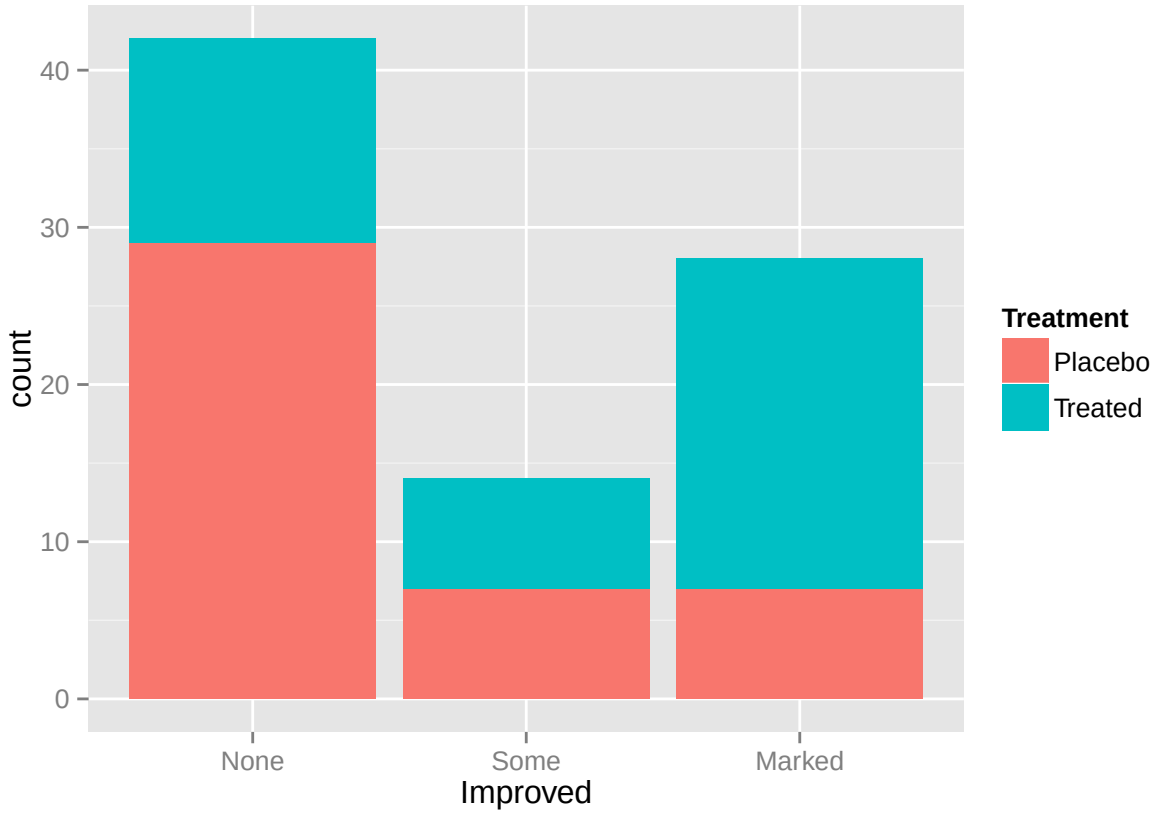


For geom_bar color only controls color of outline, to change color of the fill, you'll have to use fill aesthetic.

```
ggplot(Arthritis, aes(Improved))+geom_bar(aes(color=Treatment))
```

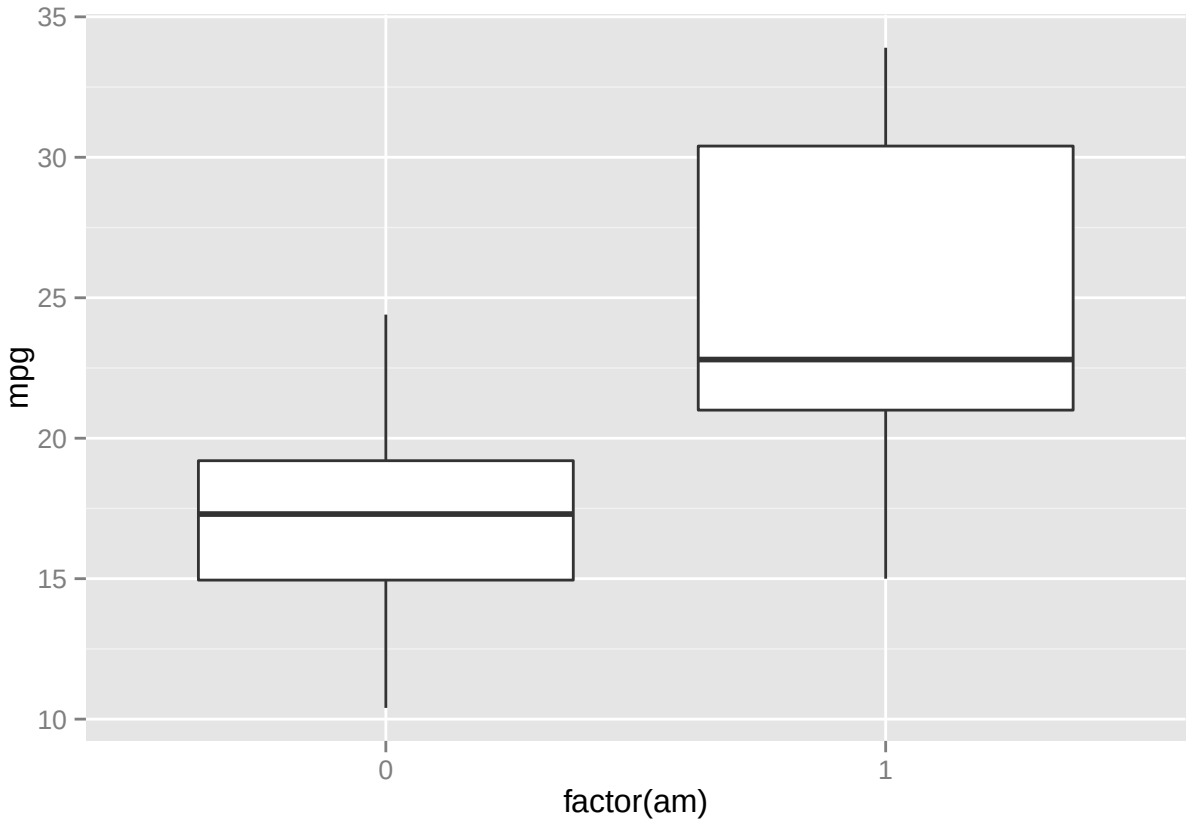



```
ggplot(Arthritis, aes(Improved))+geom_bar(aes(fill=Treatment))
```

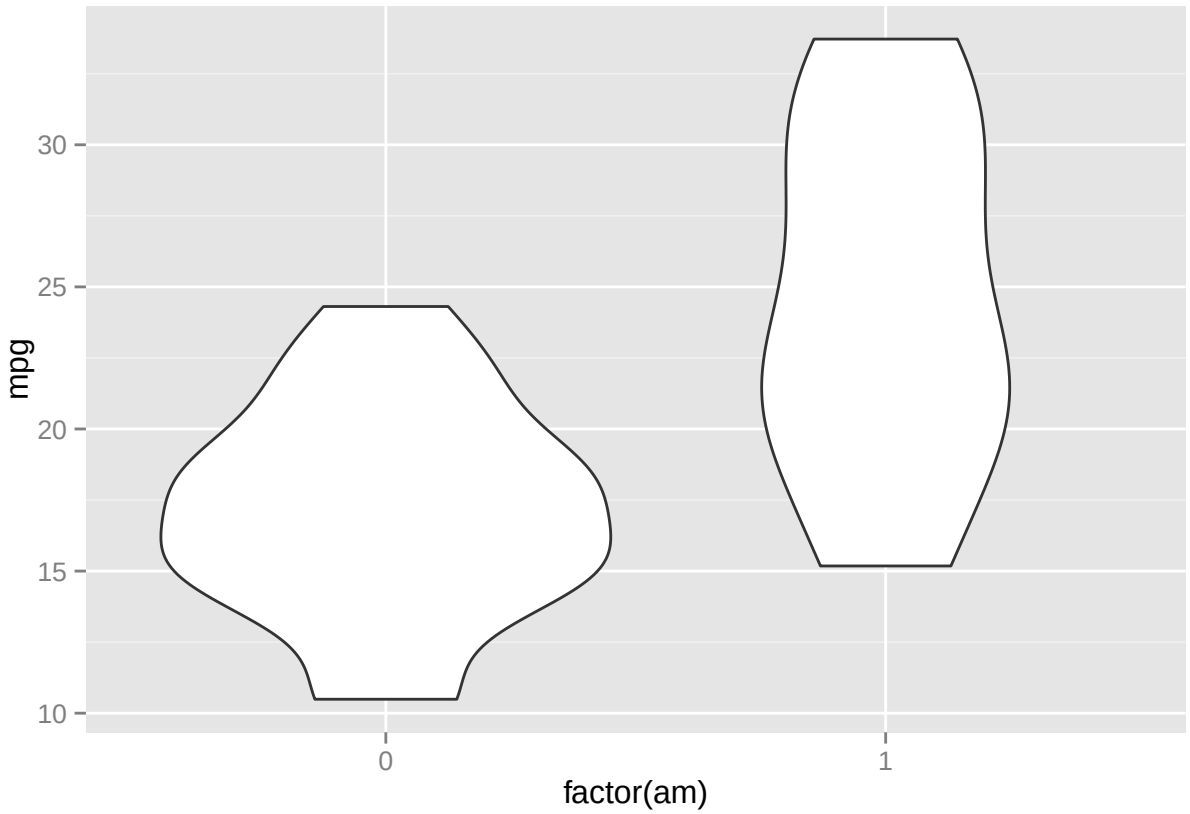


We'll take another example of boxplot. By now you must start to think what aesthetic will map to which data point in order to generate desired plot. For example to generate a boxplot y axes should map to the variable in question and x axes should map to the grouping variable .

```
ggplot(mtcars,aes(y=mpg,x=factor(am)))+geom_boxplot()
```

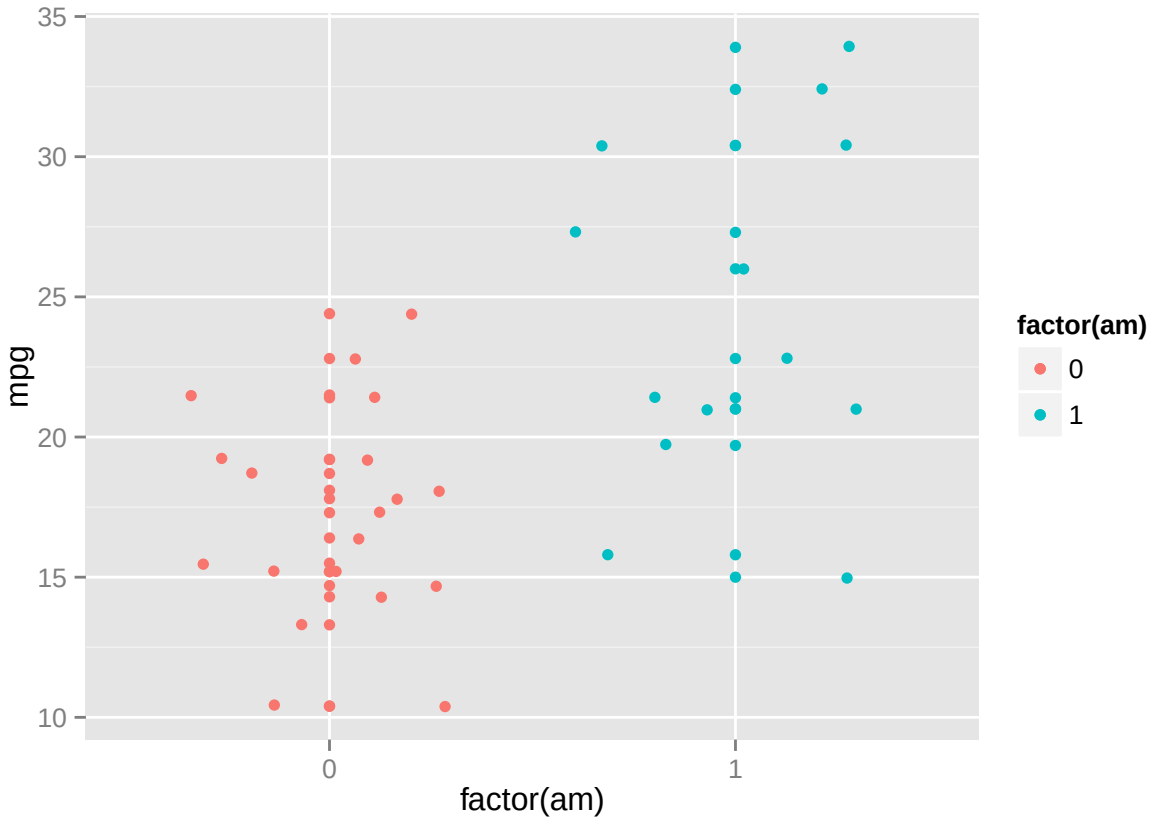



```
ggplot(mtcars,aes(y=mpg,x=factor(am)))+geom_violin()
```



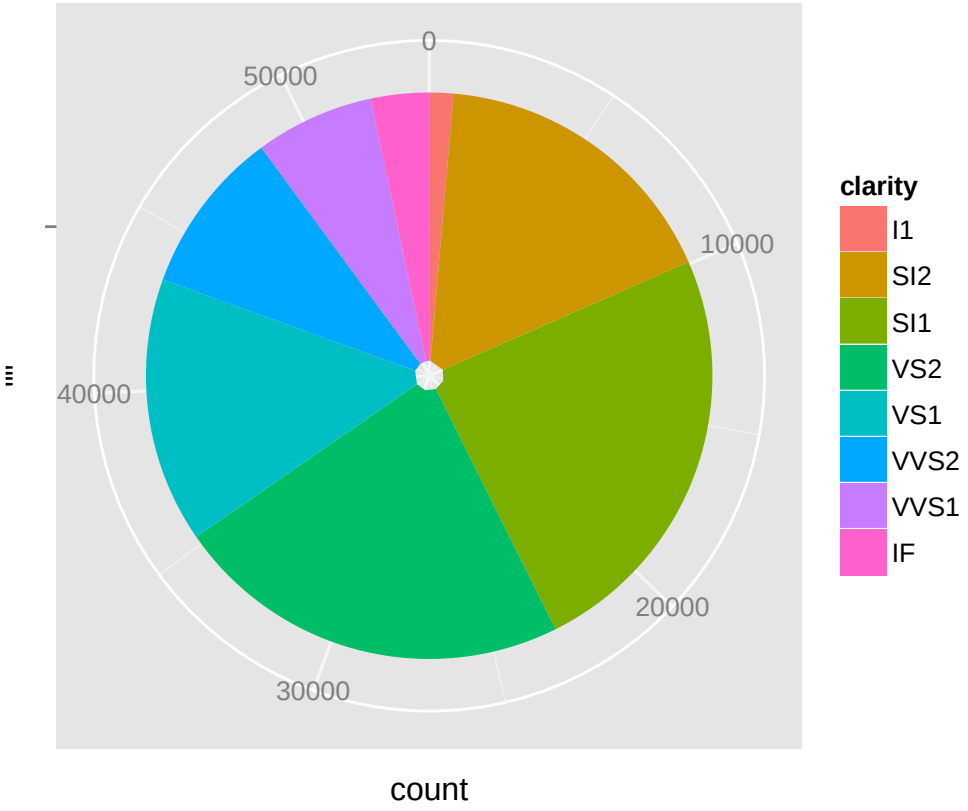
Here violin plots , in addition to your boxplot , provide how dense your points as well. Even better will be to simply add a point layer and add little jitter to it.

```
ggplot(mtcars,aes(y=mpg,x=factor(am),color=factor(am)))+geom_point()+geom_jitter()
```



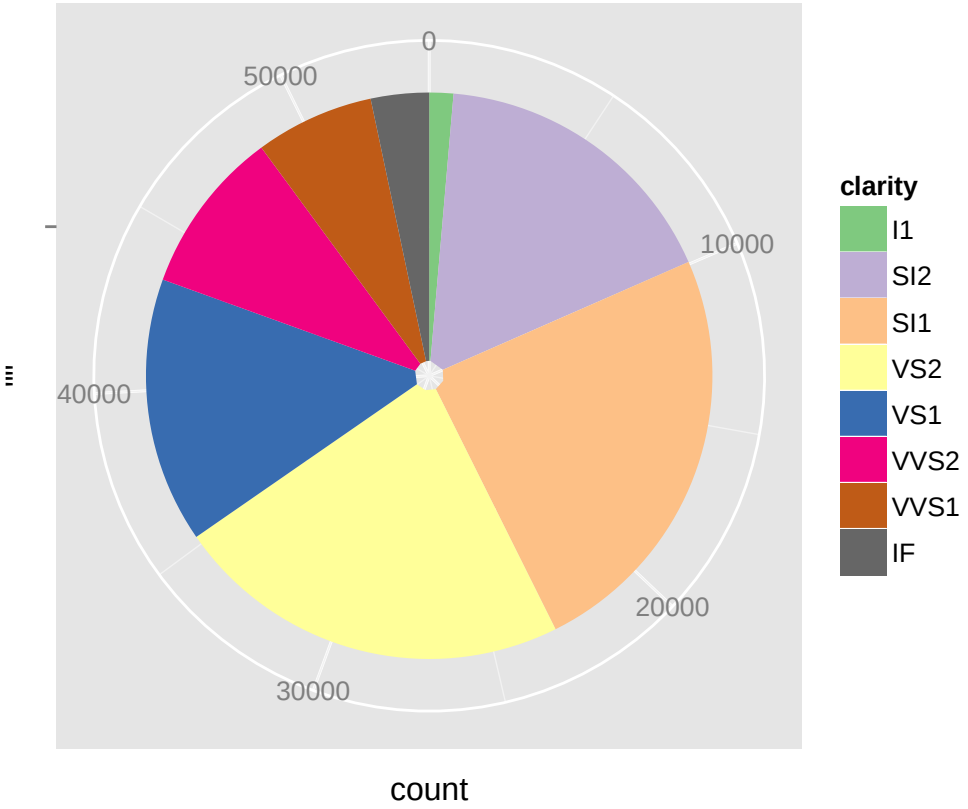
Lets see how to make different kind of plots by playing around with coordinate layers. So far the plots that we have seen , all have simple x-y coordinate system. By adding a polar coordinate system we can convert our bar plot to pie charts. This makes you realize that , pie charts are nothing but bar charts in a polar coordinate system. Lets see. We'll be using dataset `diamond` here which comes loaded with the package `ggplot2`. What you need to remember here is that in the simple bar plot x axis belongs to different values taken by a categorical variable and y axis belongs to counts of those values in the data. By assigning `theta` to y or [count], you are essentially making share of each type in the polar bands getting determined by their counts

```
ggplot(diamonds, aes(x = "", fill = clarity)) + geom_bar() + coord_polar(theta = "y")
```



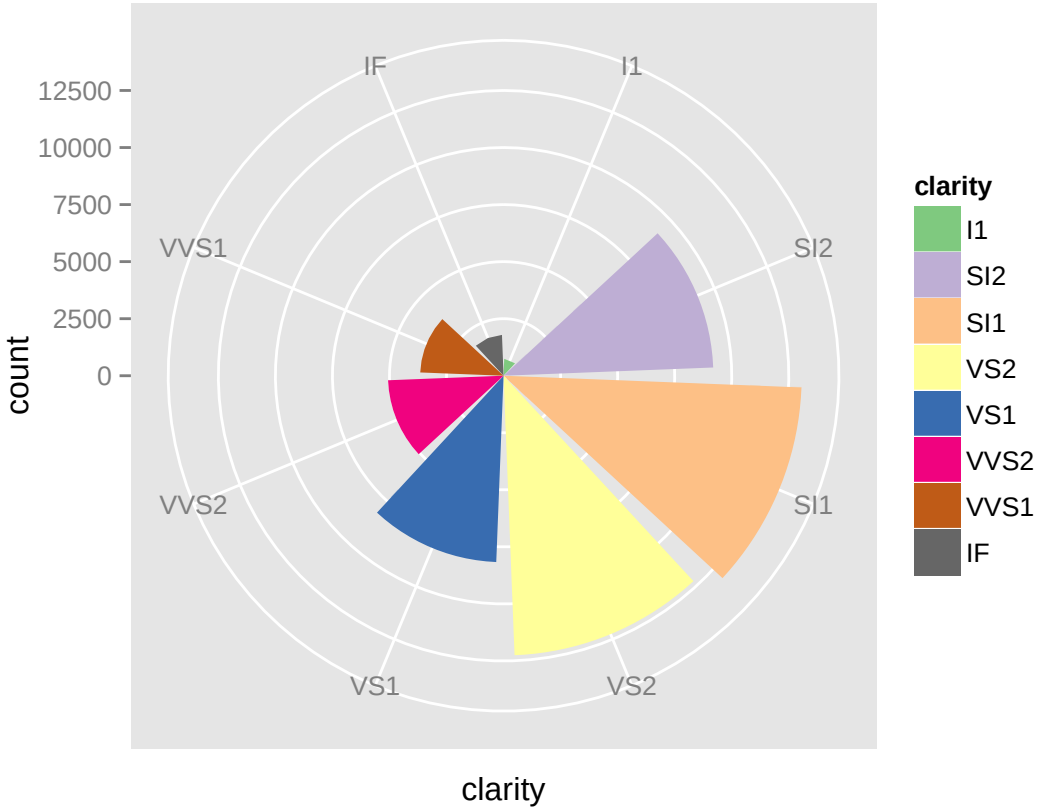
you can try a different color palette also

```
ggplot(diamonds, aes(x = "", fill = clarity)) + geom_bar() +
  coord_polar(theta ="y")+scale_fill_brewer(palette = "Accent")
```



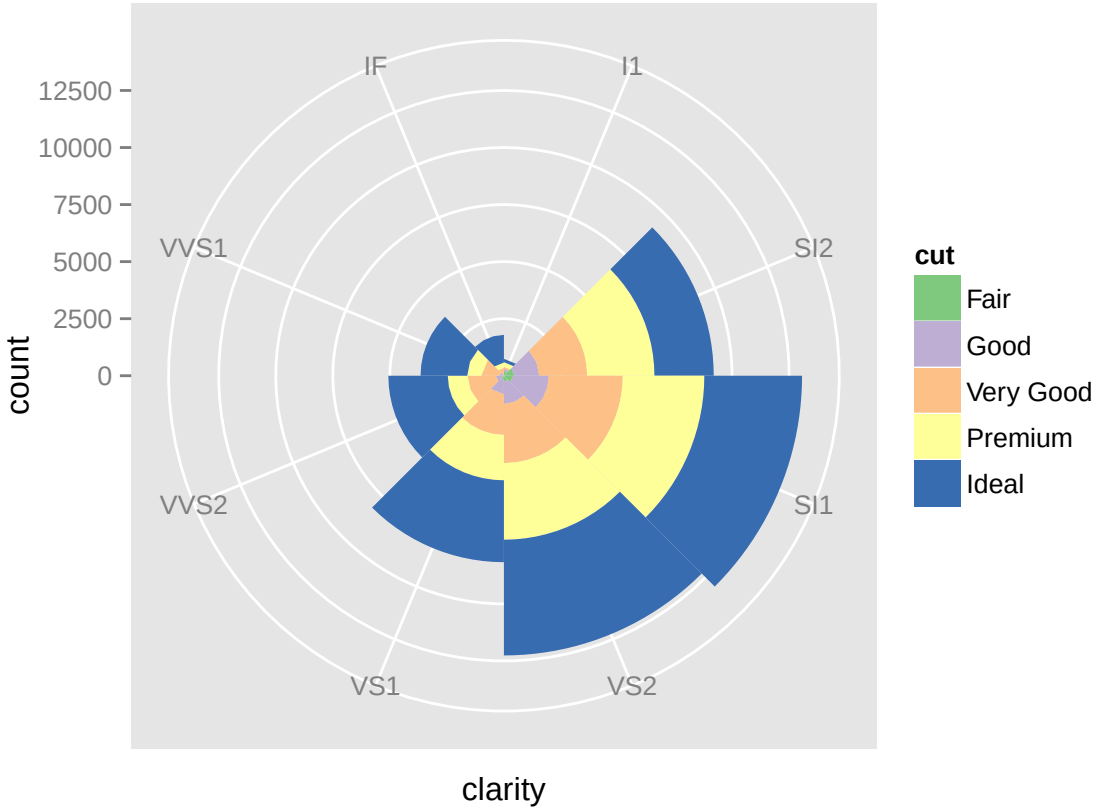
Now if you map x to clarity , which basically becomes radius in polar coordinates, then instead of equal bar width (or equal radius share), it will determined by values of data mapped to x aesthetic. Lets see.

```
ggplot(diamonds, aes(x = clarity, fill = clarity)) + geom_bar() +
  coord_polar(theta = "x") + scale_fill_brewer(palette = "Accent")
```



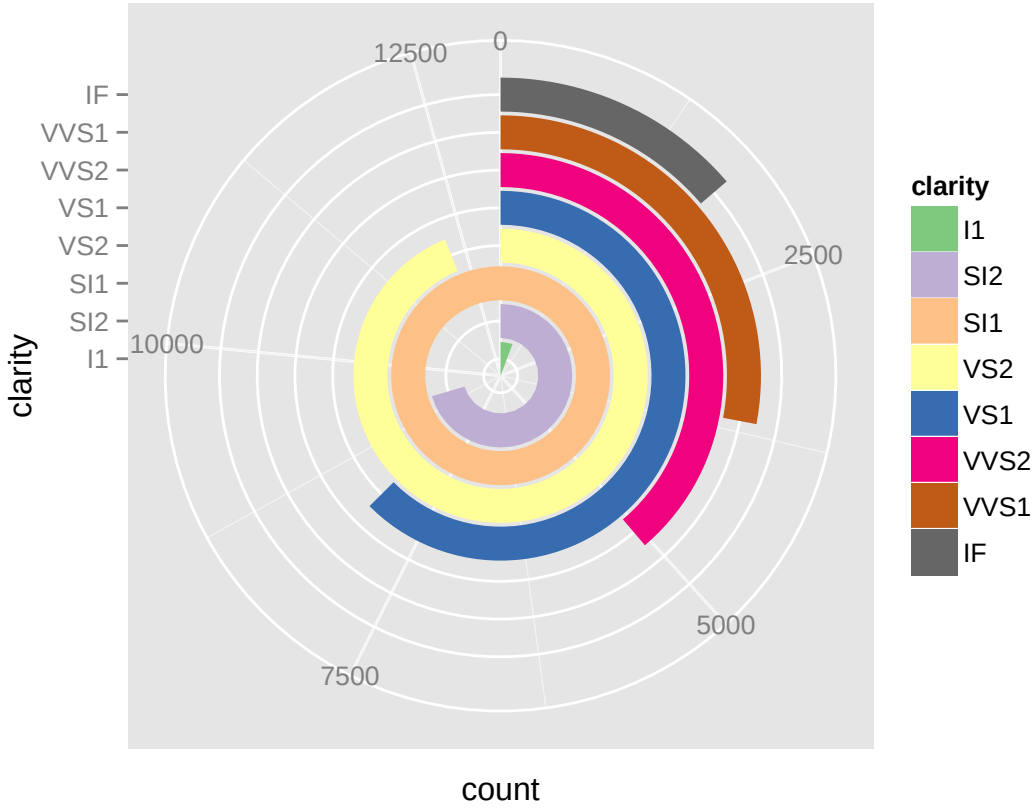
Incorporating another variable on fill gives the 'Wind Rose' plot:

```
ggplot(diamonds, aes(x = clarity, fill = cut)) + geom_bar(width = 1) +
  coord_polar(theta = "x") + scale_fill_brewer(palette = "Accent")
```



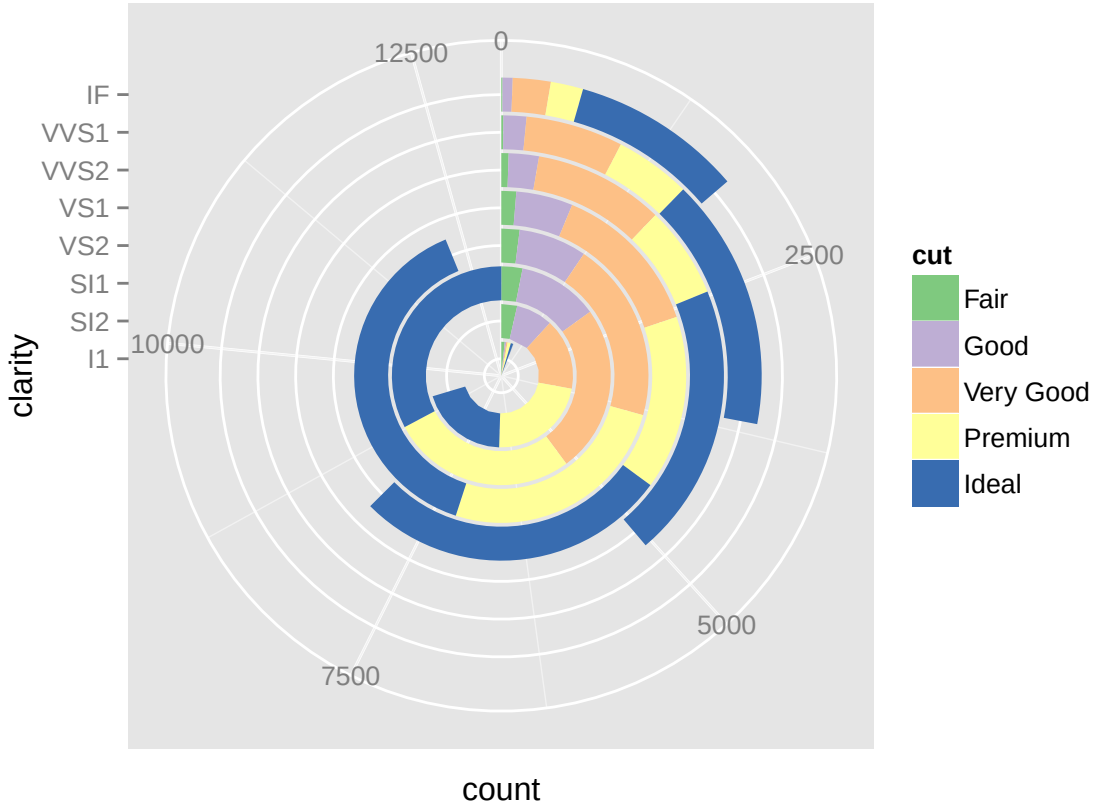
Putting theta on 'Y' gives a weird sort of bulls-eye plot. Use a width of .9 to produce space between bars.

```
ggplot(diamonds, aes(x = clarity, fill = clarity)) + geom_bar(width =0.9) +
  coord_polar(theta = "y") + scale_fill_brewer(palette = "Accent")
```



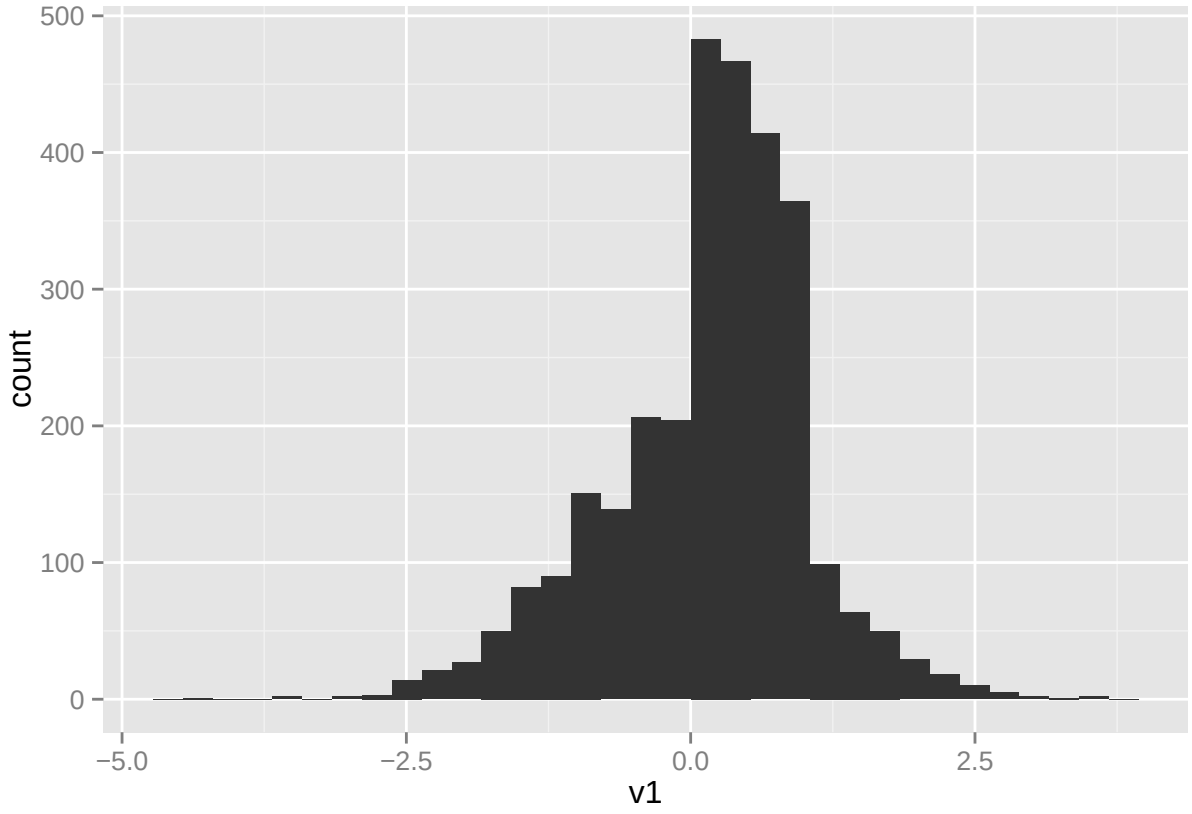
This becomes more interesting with a different fill:

```
ggplot(diamonds, aes(x = clarity, fill = cut)) + geom_bar(width = 0.9) +
  coord_polar(theta = "y") +  scale_fill_brewer(palette = "Accent")
```

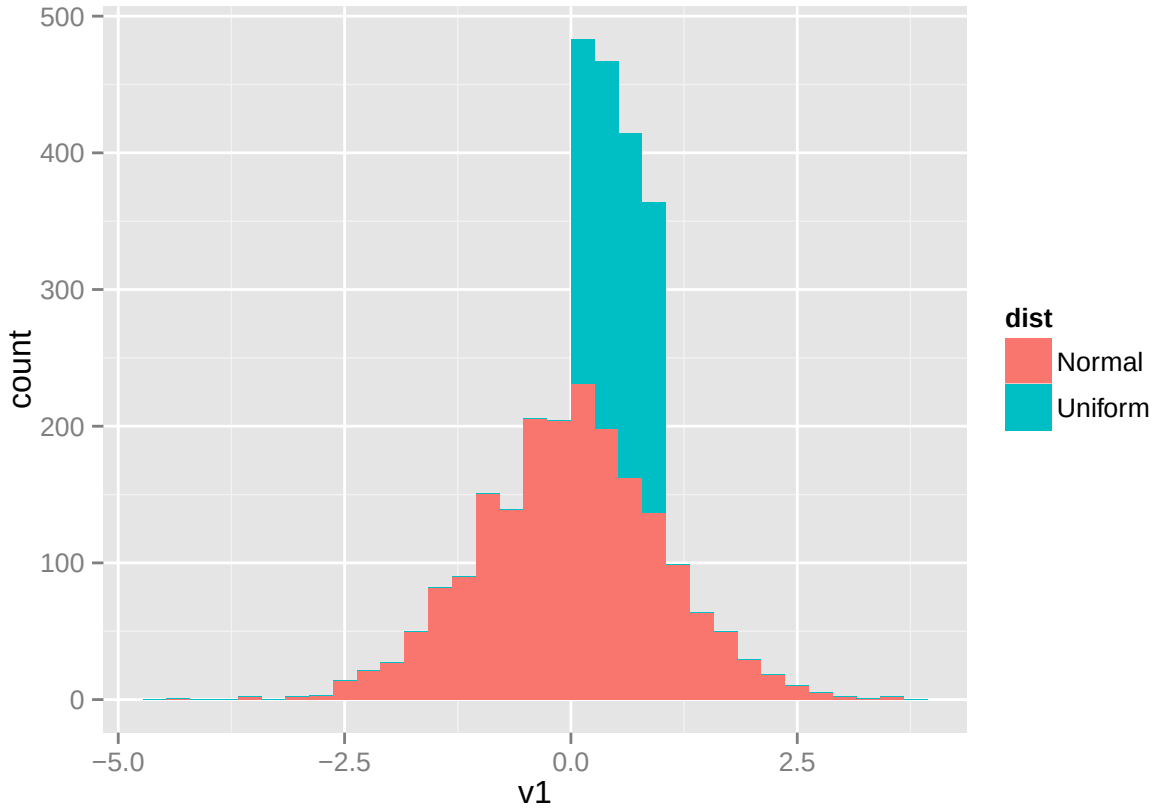


You can also play around with scale layer to transform your axes to log or any other scales. I'll leave that for you to explore on your own. Another important visualization that you would like to look at will be distribution/histogram of your variable. Lets do that. Remember that histogram is nothing but a special case of bar charts for continuous numeric variables where x axis is made up bins depending on values of the data mapped to x aesthetic. Y axis still represent counts.

```
mydata=data.frame(v1=c(rnorm(2000),runif(1000)),
                  dist=c(rep("Normal",2000),rep("Uniform",1000)))
ggplot(mydata,aes(x=v1))+geom_histogram()
```

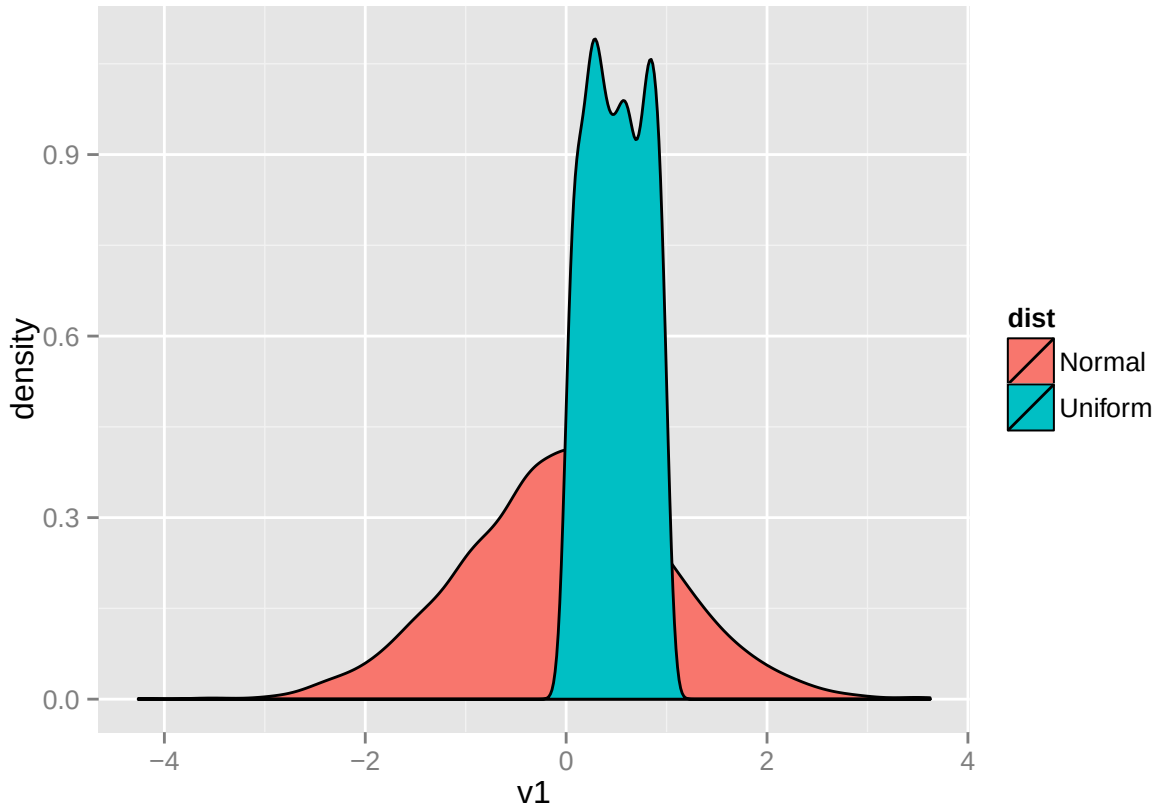



```
ggplot(mydata,aes(x=v1,fill=dist))+geom_histogram()
```



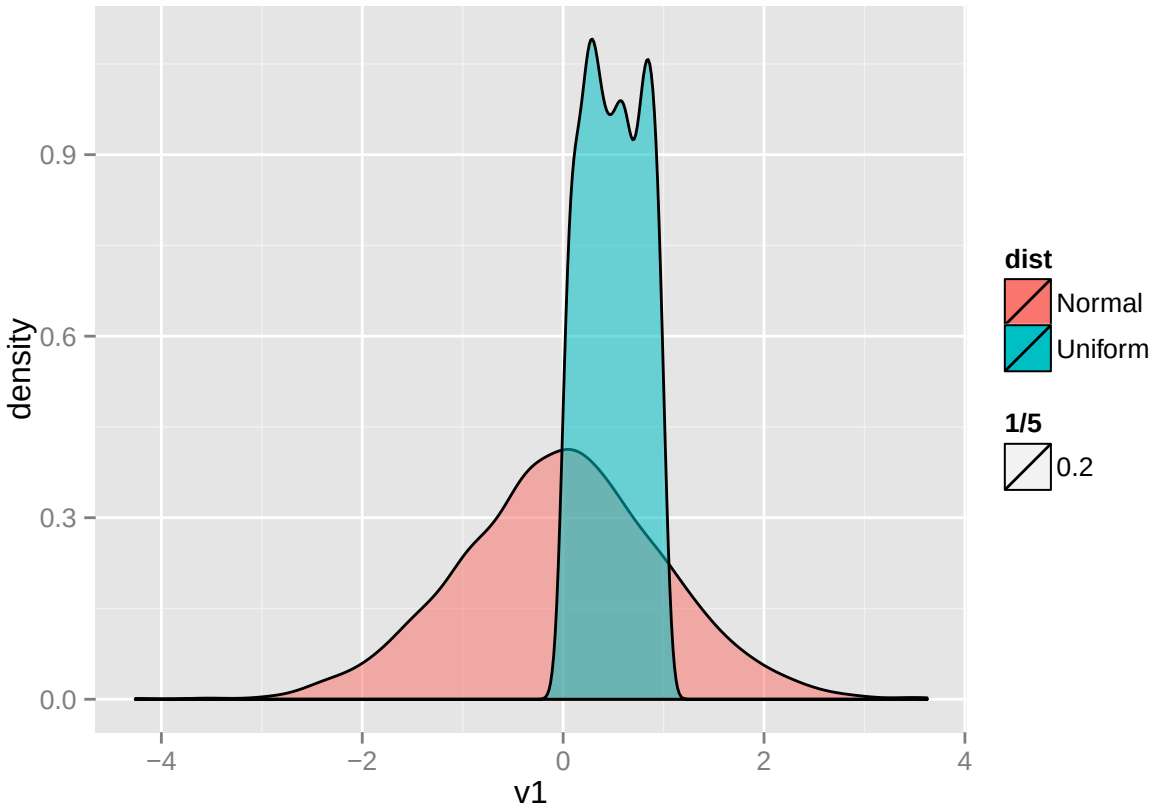
You can directly plot density curves also.

```
ggplot(mydata,aes(x=v1,fill=dist))+geom_density()
```



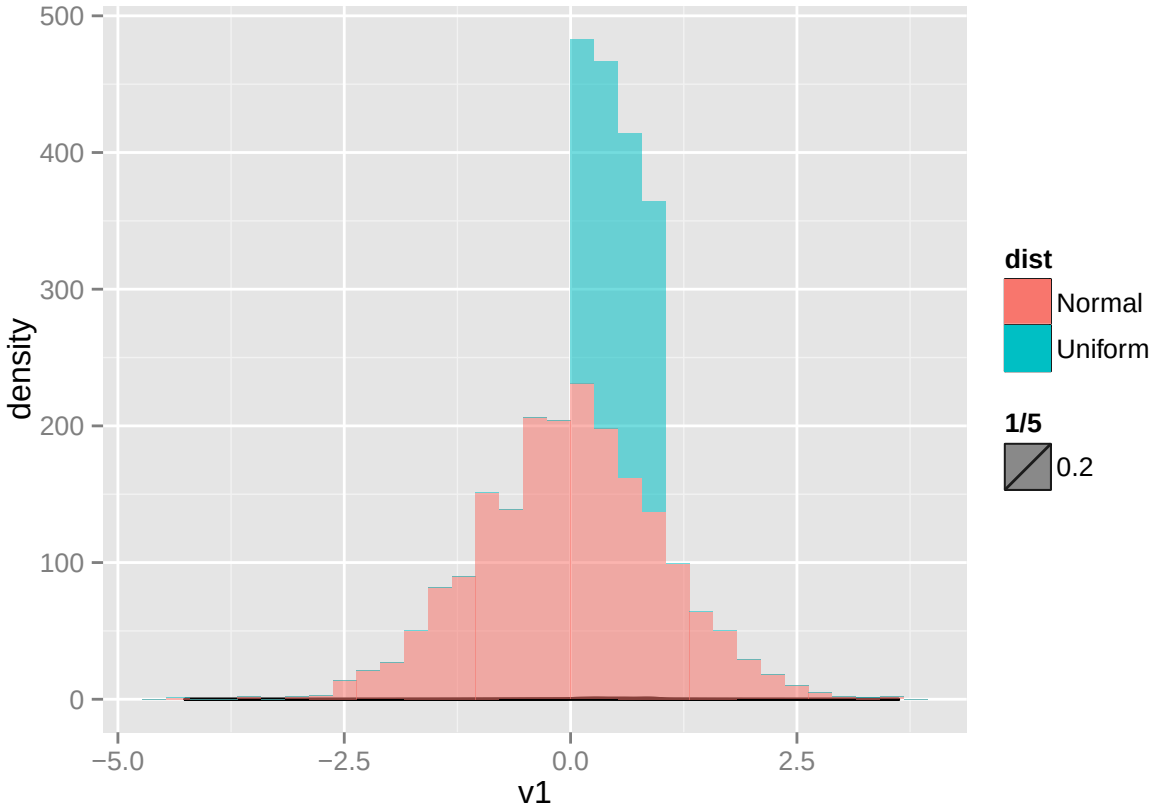
you can see that density curves are overlapping. This can be dealt with setting transparency of fill by using aesthetic `alpha`.

```
ggplot(mydata,aes(x=v1,fill=dist,alpha=1/5))+geom_density()
```



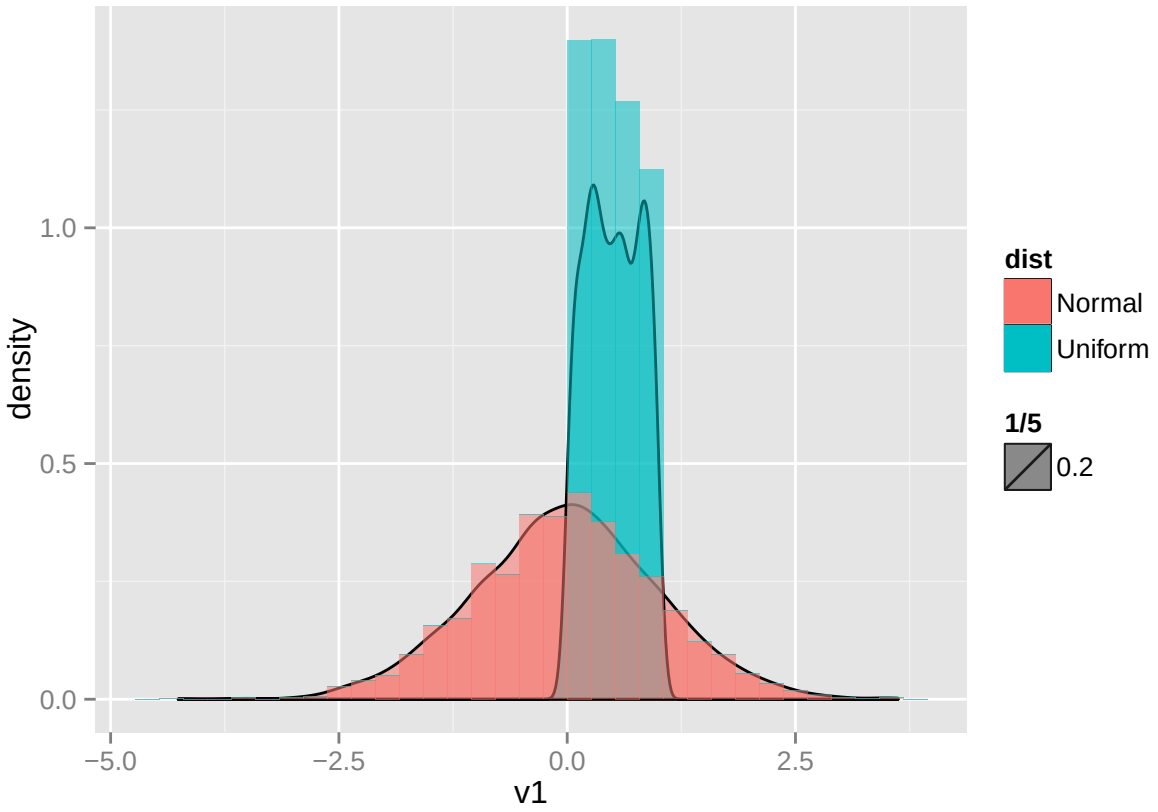
You can add histogram and density together, but there is an issue which you'll realize soon.

```
ggplot(mydata,aes(x=v1,fill=dist,alpha=1/5))+geom_density()+geom_histogram()
```



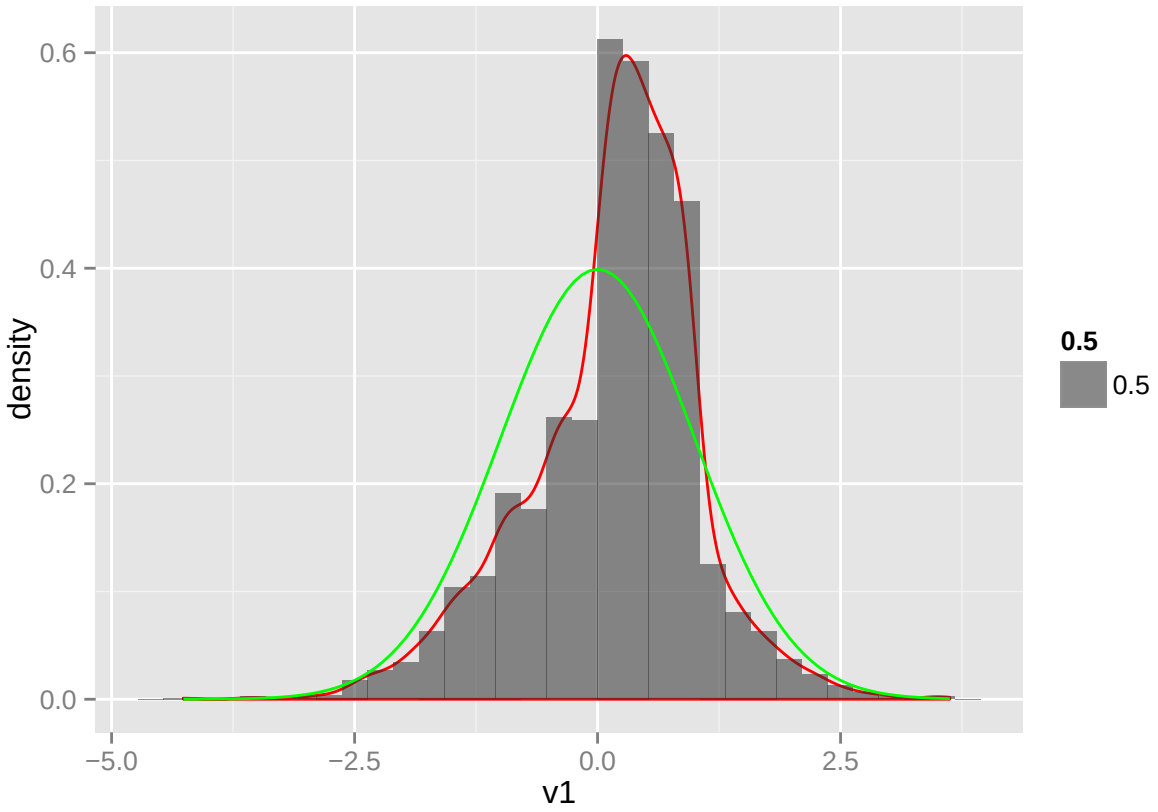
See here, density curves are no where to be seen. Believe me, they are there just that the y axis values for density are in percentages where as for histogram these are counts which obviously dominate. Hence your density curves are there but they are merely hugging x axis, this can be taken care of by modifying y aesthetic of histograms.

```
ggplot(mydata,aes(x=v1,fill=dist,alpha=1/5))+geom_density()+
  geom_histogram(aes(y=..density..))
```



Now lets say we wanted to check whether a given data conforms to a normal distribution or not. Clearly we'd like to plot a normal density curve assuming the data is from normal distribution [getting μ and σ from the data] and compare it with the natural density curve of the data. Lets see how we can achieve that.

```
ggplot(mydata,aes(x=v1))+geom_density(color="red")+
  geom_histogram(aes(y=..density..,alpha=0.5))+
  stat_function(fun=dnorm,aes(x=v1),color="green")
```



We'll conclude here . By now you must understand the layering philosophy of graphics, there is a lot of option still to try and you can explore that by looking at various examples available on the net. Here are few sources to look at.

http://r4stats.com/examples/graphics-ggplot2/

http://www.cookbook-r.com/Graphs/Bar_and_line_graphs_(ggplot2)/

A slightly advanced source of examples can be found here :

https://plot.ly/ggplot2/

Few sources that i have provided are a good starting point for you to explore , by no means that's an exhaustive list. Learn and Explore, Explore and learn.

Prepared By: Lalit Sachan

Contact : lalit.sachan@edvancer.in

In case of any doubts or need for clarification , please take to Q&A forum on LMS.