post01-Sharon-Huang

ggplot2

Introduction

In the class stat133, I learned a little bit about ggplot2. From the lectures and labs, I realized that ggplot2 is a wonderful R package for graphing instead of base graphics because ggplot2 can describe the data with more visual perception. The graphs from ggplot2 can become very beautiful. In this post, I will introduce some backgound of ggplot2 first, such as the history of ggplot2, and the pros and cons of ggplot2. Second, I will show about the basic structure of ggplot2. Third, I will show some amazing plots from ggplot for us to enjoy. Now, let's know some background about ggplot2 together first.

Background

ggplot2 is an R package for the statistical programming language R to produce statistical graphics, created by Hadley Wickham in 2005.
ggplot2 is an implementation of Leland Wilkinson's Grammar of Graphics-a general scheme for data visualization which breaks up graphs into semantic components such as scales and layers. The gg in "ggplot2" stands for Grammar of Graphics.
ggplot2 can serve as a replacement for the base graphics in R and contains a number of defaults for web and print display of common scales.
Since 2005, ggplot2 has grown in use to become one of the most popular R packages. It is licensed under GNU GPL v2.

Advantages of ggplot2
- The package provides a framework based on Leland Wilkinson's Grammar of Graphics.
- The package "ggplot2" provides beautiful plots while taking care of fiddly details like legends, axes, colors, and so on. It includes mature and complete graphics system, and it also includes theme system for polishing plot appearance.
- The package "ggplot2" is very flexible for annotating, editing, and embedding output.
- The package "ggplot2" is specification is at a high level of abstraction.

Disadvantages of ggplot2
- The package "ggplot2" can not provide 3-dimensional graphics.
- The package "ggplot2" does not have Graph-theory type graphs.

Basic Structure of ggplot2

```
# Before using ggplot2 package,
# we must install.packages first,
# but we just need to install a package once,
# so we do not need to type the below code in the code chunk.
# We just need to install the package in Console.
install.packages("ggplot2")
```

```
# After installing the package,
# we need to load the package.
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.2
```

After loading the "ggplot2", we need a dataset to discuss some examples about "ggplot2."
Be careful, if we want to use ggplot2, the data must be in a **data frame**.

```
# see the data sets in package "datasets"
data()
```

```
# choose a simple data for us to anaylise
data("iris")
# know about the data
head(iris)
```

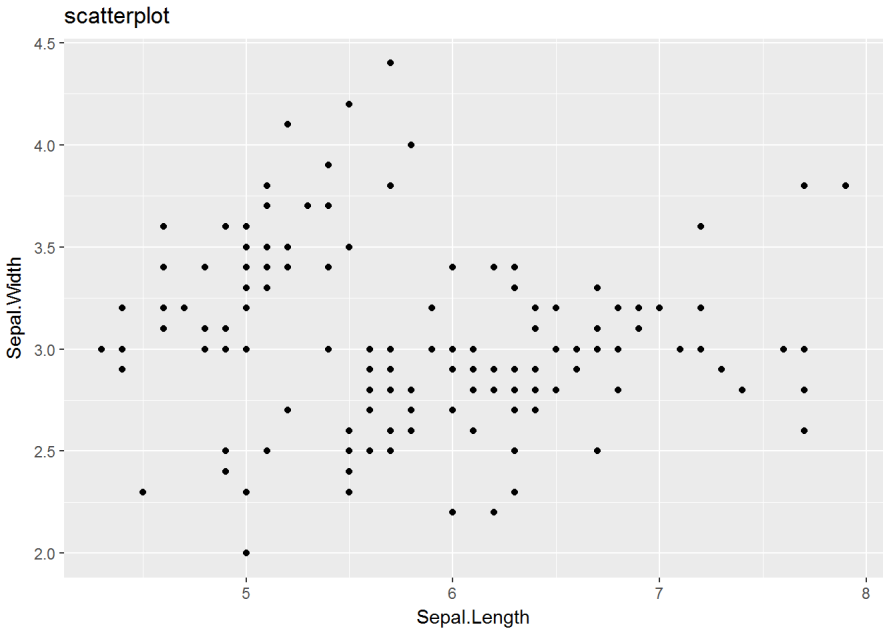
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

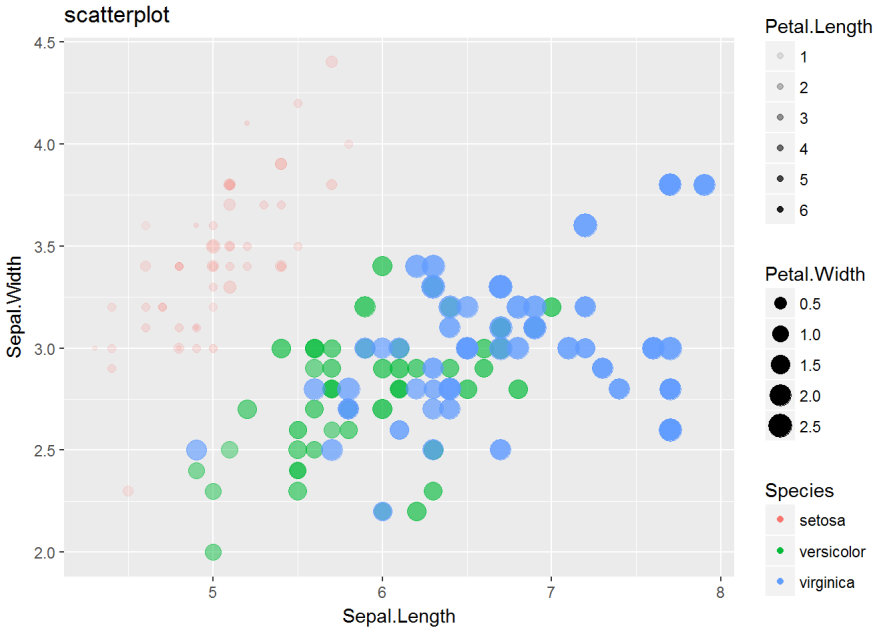
Now, we need to learn about the plot basics.
1. The main function in "ggplot2" is `ggplot()`. In the `ggplot()`, we need to specify which dataset we are using, and we need to clarify the X axis and Y axis inside the `aes()` argument. Also, we can add whatever aesthetics we want to apply to our ggplot inside 'aes()' argument, such as color, size, shape, stroke, and fill.
2. Once the based setup is done, we need layers, which is `geom_XXX`. The plot can not be displayed until we add a layer. We use "+" to combine `ggplot()` and layers together.
There are some examples showing below:

```
# geom_point() describes the points in ggplot2.
# scatterplot of Sepal.Length and Sepal.Width
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_point() +
  labs(title="scatterplot")
```



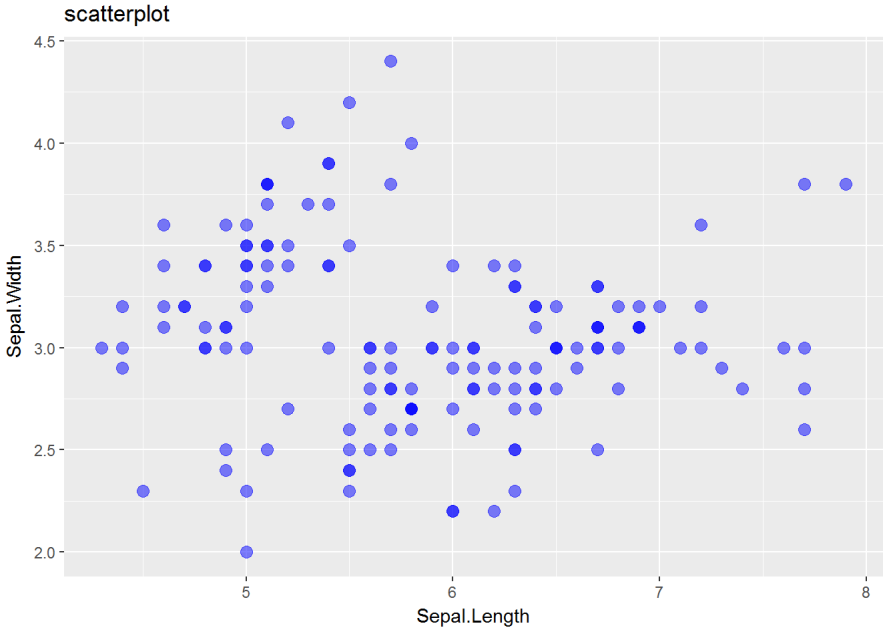
`geom_point()` Also, `geom_point()` can describe alpha, color, fill, shape, group, stroke and size:

```
# we just need to type the one we want to use and the corresponding variables inside the 'aes()' argument.
# very easy and convinent to apply.
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_point(aes(color = Species, size = Petal.Width, alpha = Petal.Length)) +
  labs(title="scatterplot")
```



We can also describle the alpha, color, fill, shape, and size for all points:

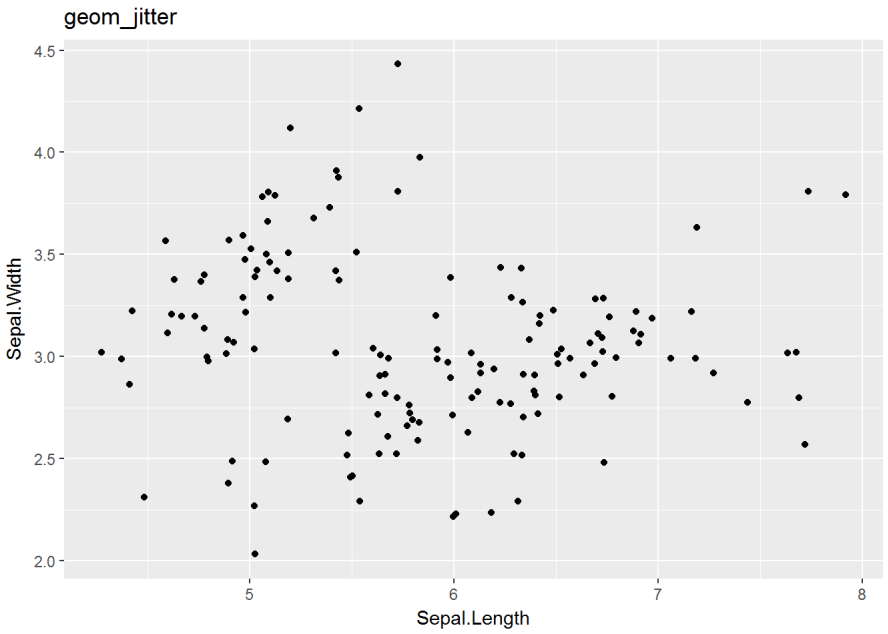
```
# For this situation, we do not need to type inside the 'aes()' argument because we are not using any specific var
iables for the values.
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_point(color = "blue", size = 3, alpha = 0.5) +
  labs(title="scatterplot")
```



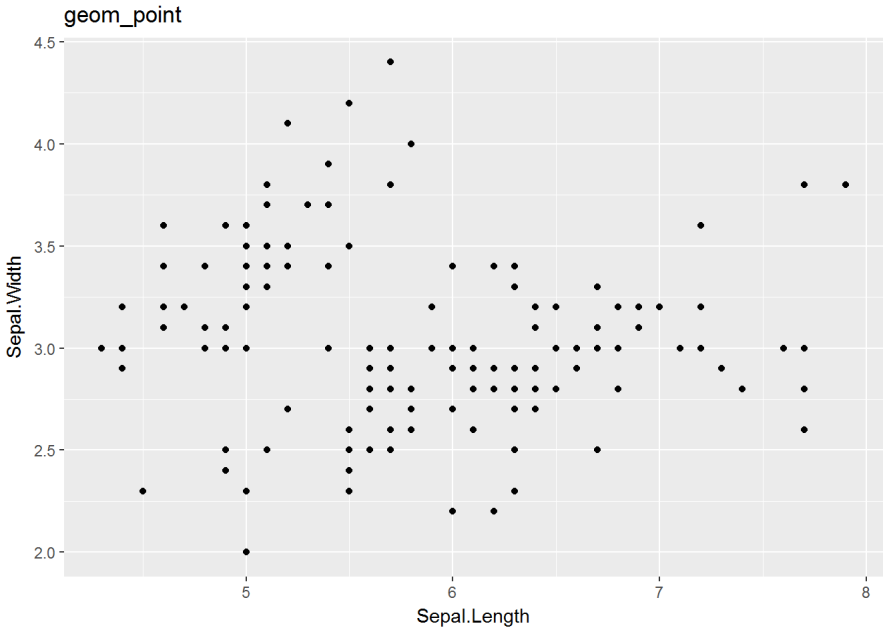

```
# If we want to know more information about geom_point(),
# we can see the document of geom_point()
?geom_point
```

`geom_jitter` *VS* `geom_point` In the 'Data Visualization with ggplot2 Cheat Sheet', I realize that the `geom_jitter` and `geom_point` is very similar. Actually, they are different. Let's see the difference between them:

```
# `geom_jitter` and `geom_point`
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_jitter() +
  labs(title="geom_jitter")
```



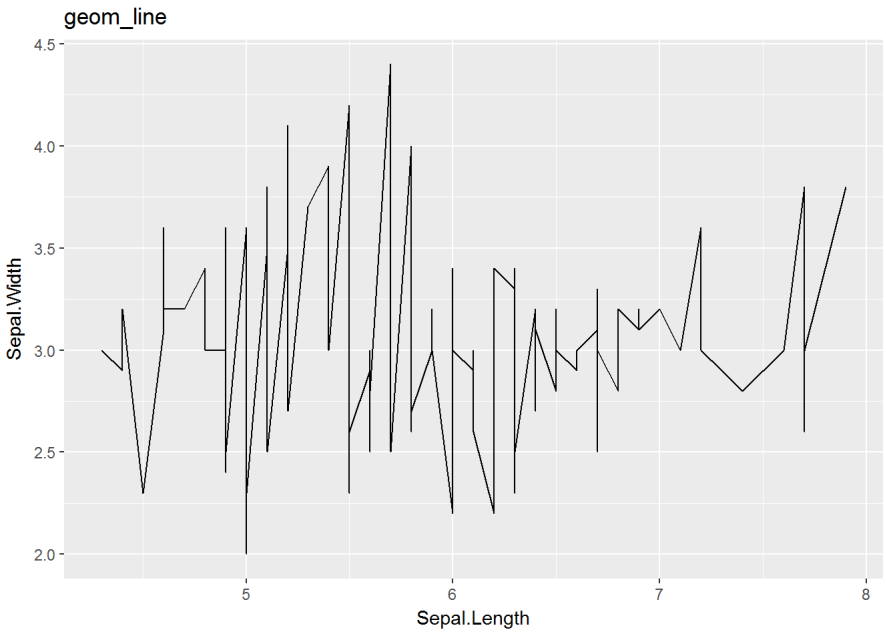

```
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_point() +
  labs(title="geom_point")
```



We can see that `geom_jitter` has more point that `geom_point`.
`geom_jitter` shows all data including overplotted points. It is only effective in the non-continuous data case where overplotted points typically are surrounded by whitespace. `geom_jitter` does not help with high density data.

`geom_line()`
We can use line instead of point:

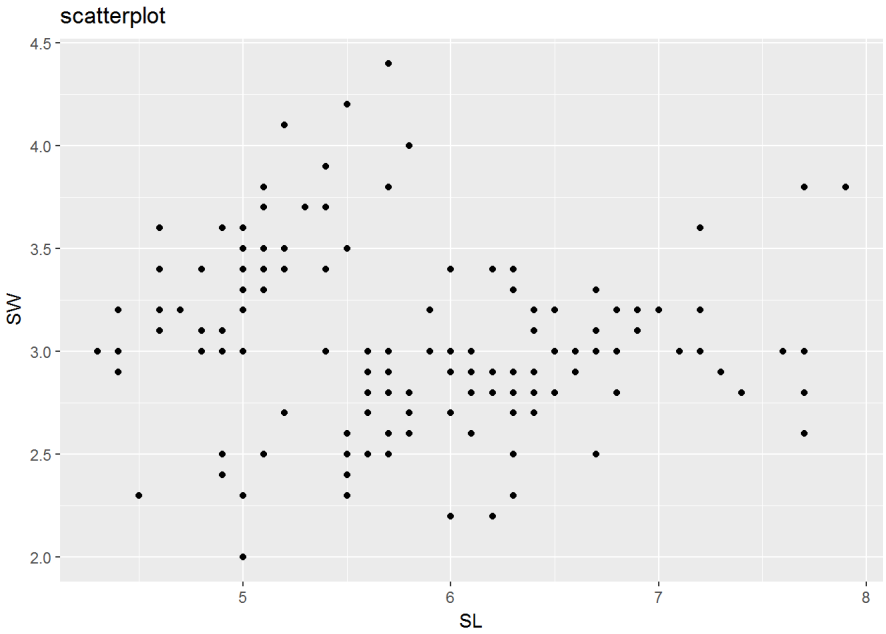
```
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_line() +
  labs(title="geom_line")
```



Actually, the histogram, boxplot, density, bar, and many other graphs have the similar structure to `geom_point()` and `geom_line()`. We can use geom_XXX() to tell the RStudio which graph we want to use. And inside the (), we can type the variables and others just like creating scatterplot. However, we need to be careful that density and histogram are just including one variable, but we still can use size, shape and others to show other variables. There are more examples in the "amazing examples from ggplot2"

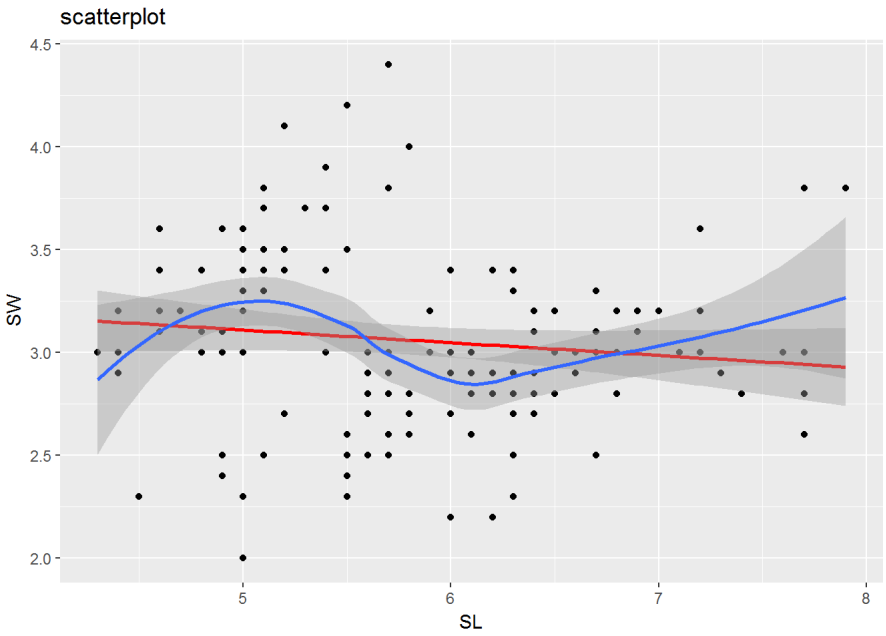
Label Also, we might want to label plot's main title and change the X and Y axis titles. For this situation, we just need to type title in `labs()`:

```
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_point() +
  labs(title="scatterplot", x="SL", y="SW")
```



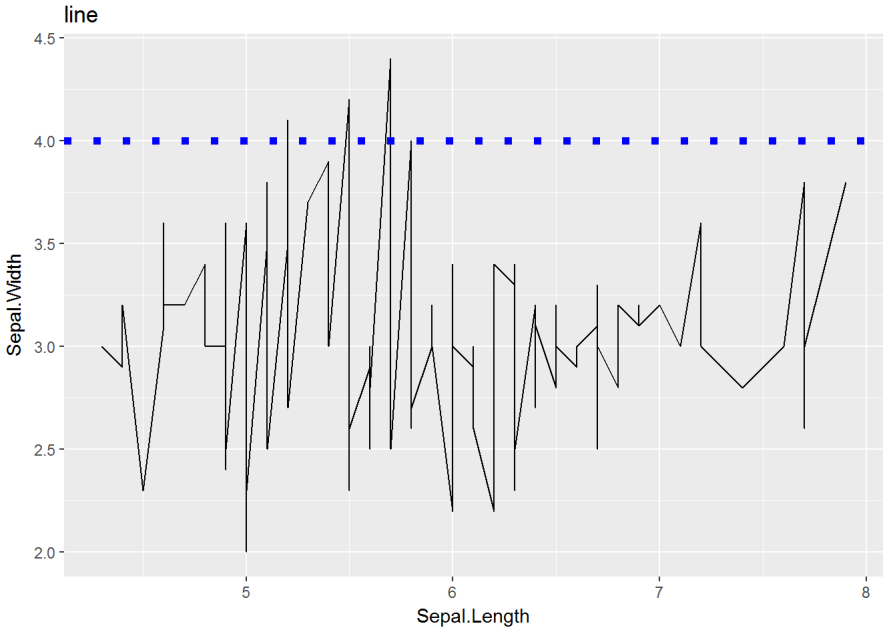
lowess line and regression line We can also apply `geom_smooth()` to lowess line and regression line:

```
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_point() +
  labs(title="scatterplot", x="SL", y="SW") +
  geom_smooth(method = lm, color="red") +
  geom_smooth(method = loess)
```



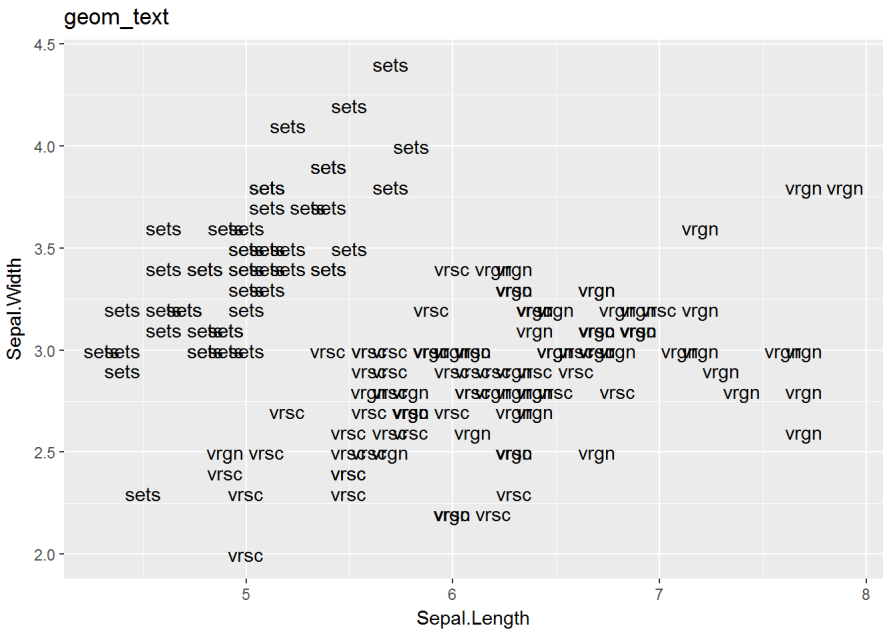
vertical line and horizontal line We can also use `geom_vline()` and `geom_hline()` to add vertical line and horizontal line:

```
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_line() +
  geom_hline(yintercept=4, size=2, linetype="dotted", color="blue") +
  labs(title="line")
```



geom_text We can use `geom_text()` to build a label mapping:

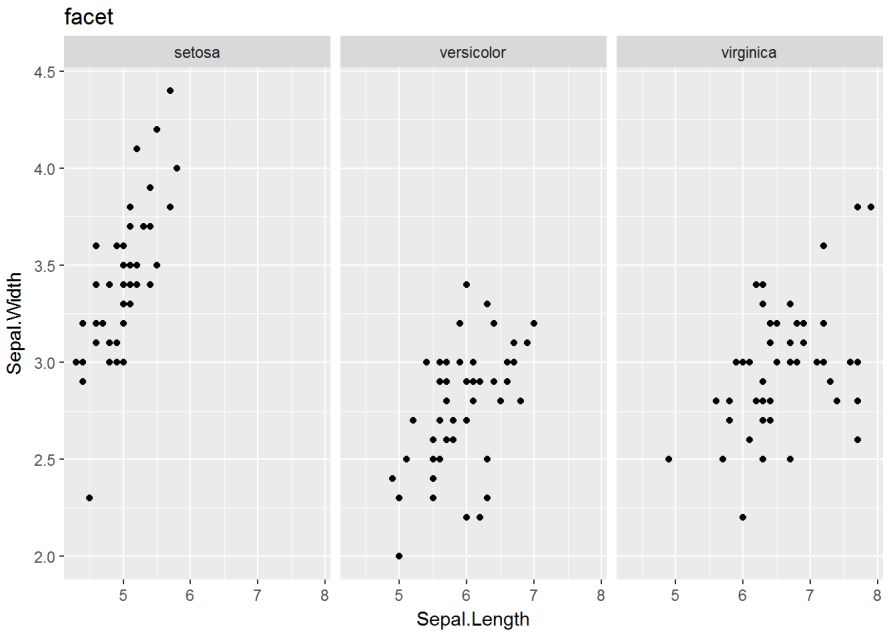
```
ggplot(data = iris, aes(x= Sepal.Length, y= Sepal.Width)) +
  geom_text(aes(label=abbreviate(Species))) +
  labs(title="geom_text")
```



Facet

One of the reason that I like "ggplot2" and want to share with you because "ggpolt2" has a feature to display multiple facets. `facet_warp()` can divide a plot into many different subplots based on the variables.

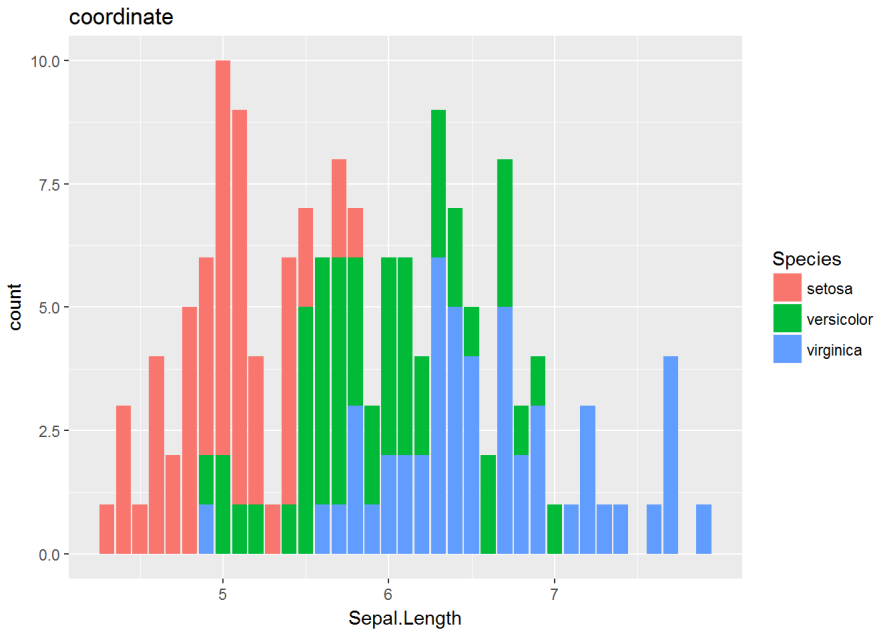
```
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point() +
  facet_wrap(~ Species) +
  labs(title="facet")
```



Coordinate systems The coordinate system determines how the x and y aesthetics combine to position elements in the plot. The default coordinate system is Cartesian, but we can teweaked it to others with:

`coord_map()`
`coord_fixed()`
`coord_flip()`
`coord_trans()`
`coord_polar()`

```
ggplot(iris)+
  geom_bar(aes(x=Sepal.Length, fill=Species))+
  coord_trans() +
  labs(title="coordinate")
```



Amazing graphs from ggplot2

According to the simple examples above, we know that ggplot2 is better than base graphic. Actually, when we have data including more variables and more complicated, ggplot2 is more useful because ggplot2 can show the graph with color, size, shape and so on beside x-axis and y-axis. We can analyse the data clearly and easily. We can see that even though the dataset is huge, it is not complicated for us to analyse the data through “ggplot2”.

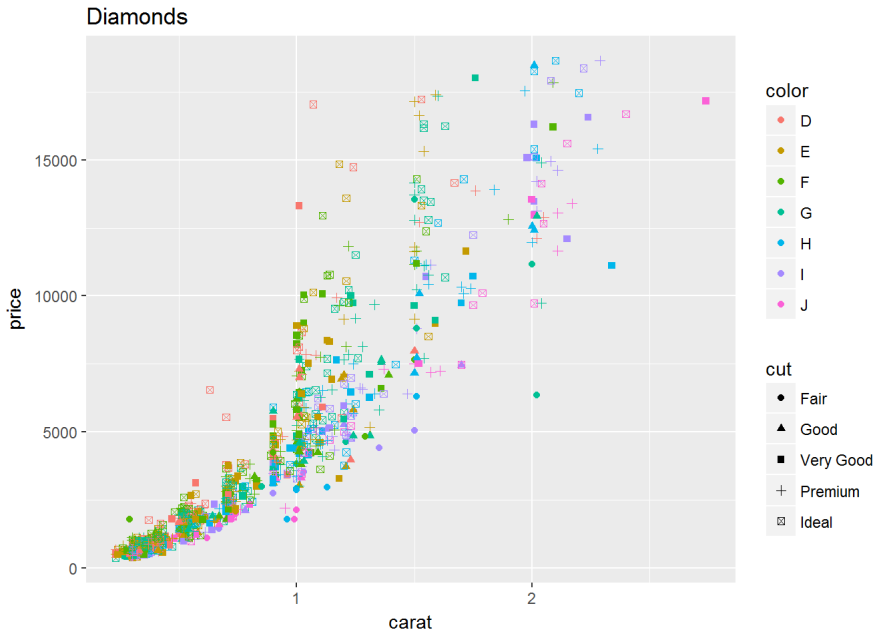
```
# Let's get a more complicated data.
data("diamonds")
summary(diamonds)
```

```
##      carat               cut        color        clarity
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1    :13065
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2    :12258
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194
##  Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2   : 5066
##  Max.   :5.0100                     I: 5422   VVS1   : 3655
##                                     J: 2808   (Other): 2531
##      depth           table           price             x
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740
##
##        y                z
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.710   Median : 3.530
##  Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :58.900   Max.   :31.800
##
```

```
# This data is super large, so it is more convinent for us to use if we just choose a small data in this dataset.
small <- diamonds[sample(nrow(diamonds), 1000), ]
```

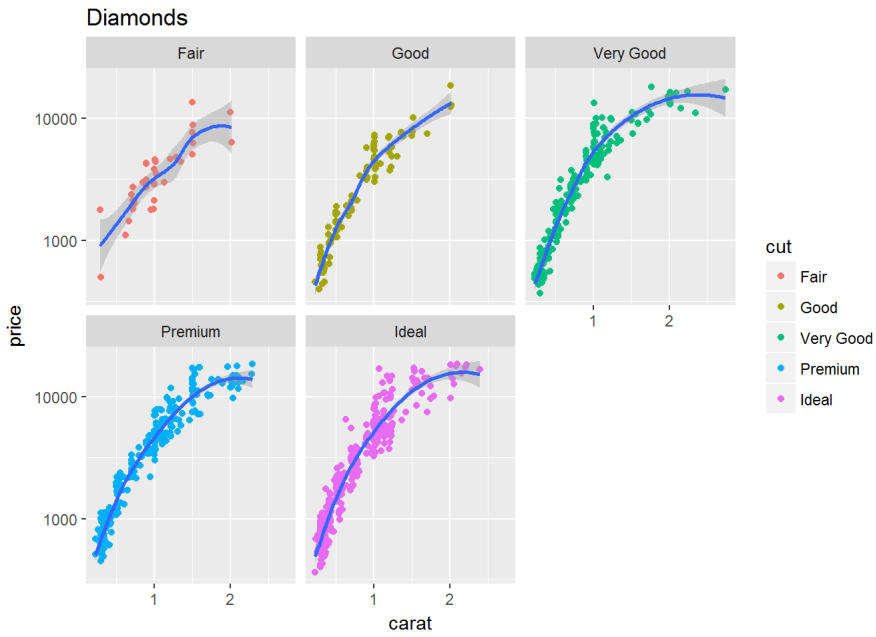
In this data, the different shape and the different color in the graph show the cut variable and color variable in the ‘diamonds’ dataset, respectively. Therefore, this graph expresses the data very well:

```
p <- ggplot(data=small, aes(x=carat, y=price, shape=cut, colour=color))
p+geom_point()+labs(title="Diamonds")
```



Even though the graph above are very beautiful, it is kind of messy and it is hard to anayse. If we separate the cutting quality, it would be more clear and easier to analize our data.
Also, we can compare the same variables in a different quanlity of cutting.

```
ggplot(small, aes(x=carat, y=price)) +
  geom_point(aes(colour=cut)) +
  scale_y_log10() +
  facet_wrap(~cut) +
  stat_smooth(method = loess) +
  labs(title="Diamonds")
```

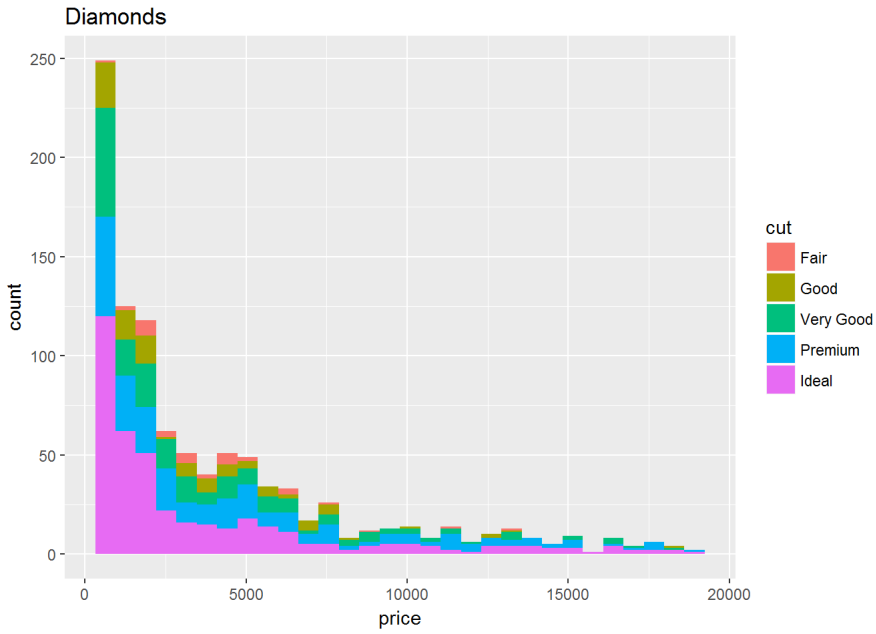


tips: *Scale*
The above graph uses scale to rescale the y-axis. It is good for us to see the data more accurate. We can also control the position, color and fill, size, shape, line type by scales in “ggplot2”.

In this graph, the different fill shows the different kinds of cut of dimonds vividly:

```
ggplot(small, aes(x=price, fill=cut))+
  geom_histogram() +
  labs(title="Diamonds")
```

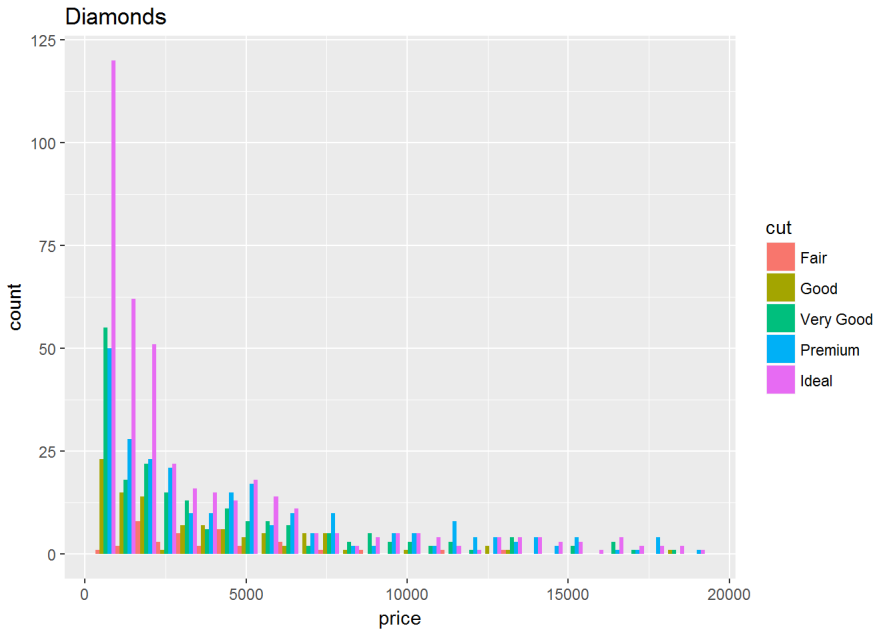
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Also, we can separate the color and change the fill to side by side:

```
ggplot(small, aes(x=price, fill=cut)) +
  geom_histogram(position="dodge") +
  labs(title="Diamonds")
```

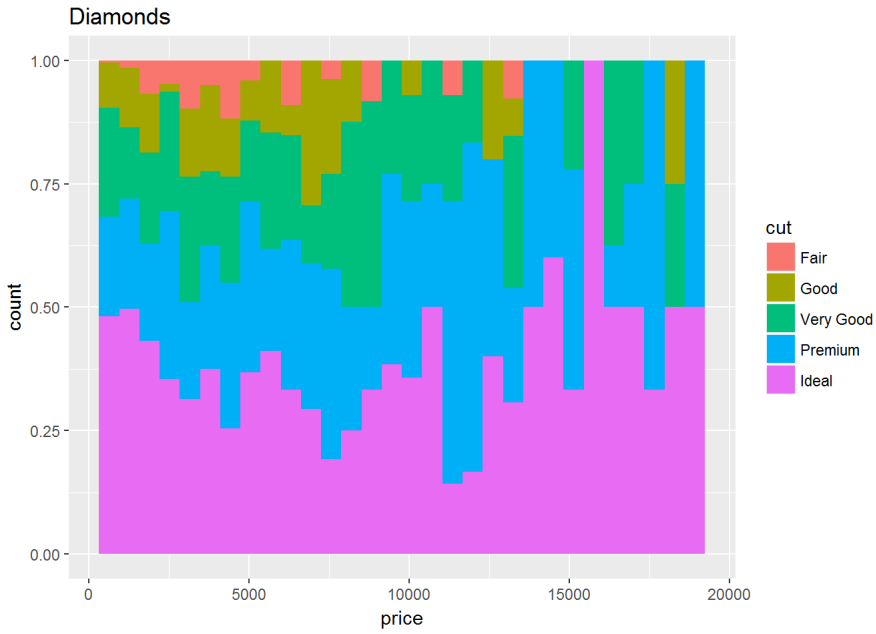
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can also show the different kinds of cut by ratio:

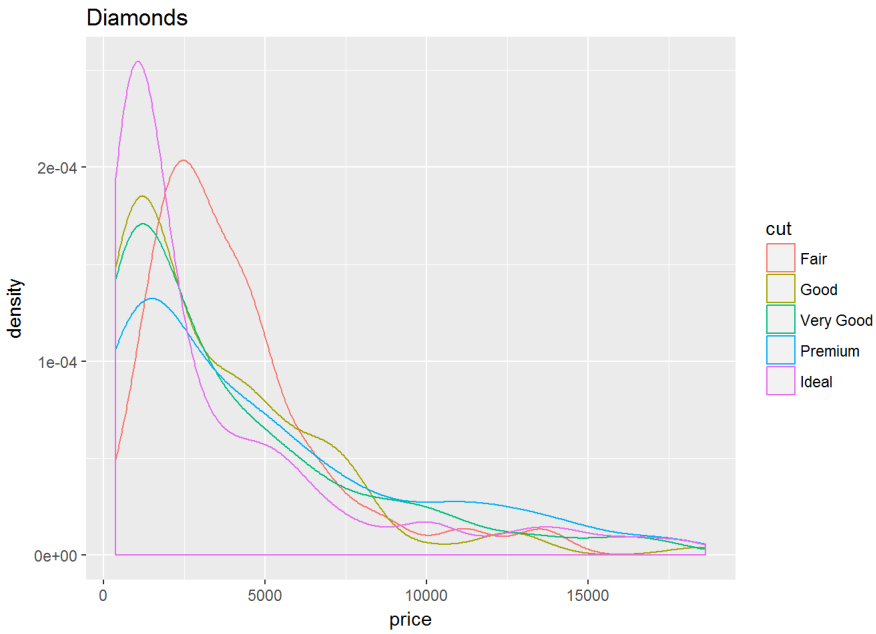
```
ggplot(small, aes(x=price, fill=cut))+
  geom_histogram(position="fill") +
  labs(title="Diamonds")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



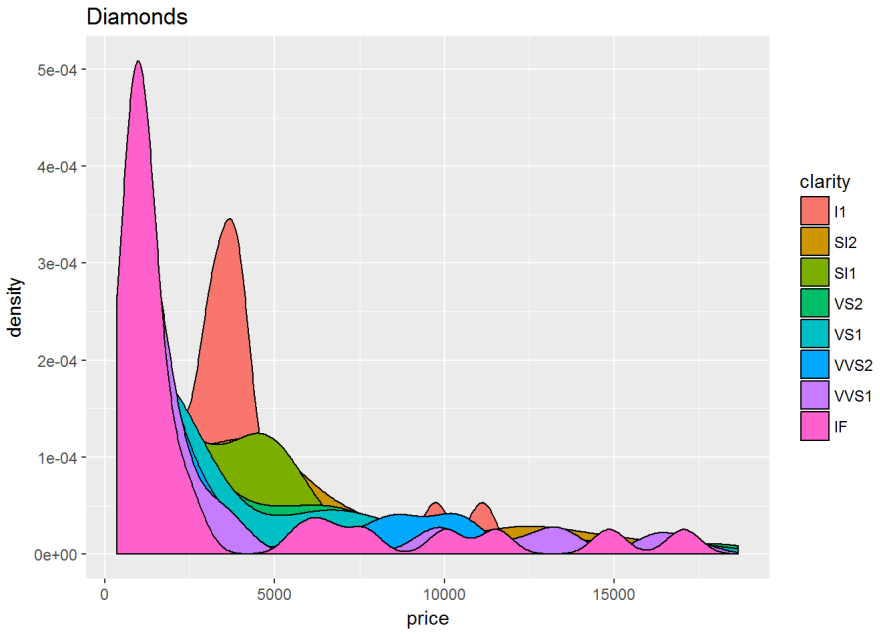
We all know the `geom_density` based on the simple example above, but this one is more beautiful:

```
ggplot(small, aes(x=price, colour=cut))+
  geom_density() +
  labs(title="Diamonds")
```

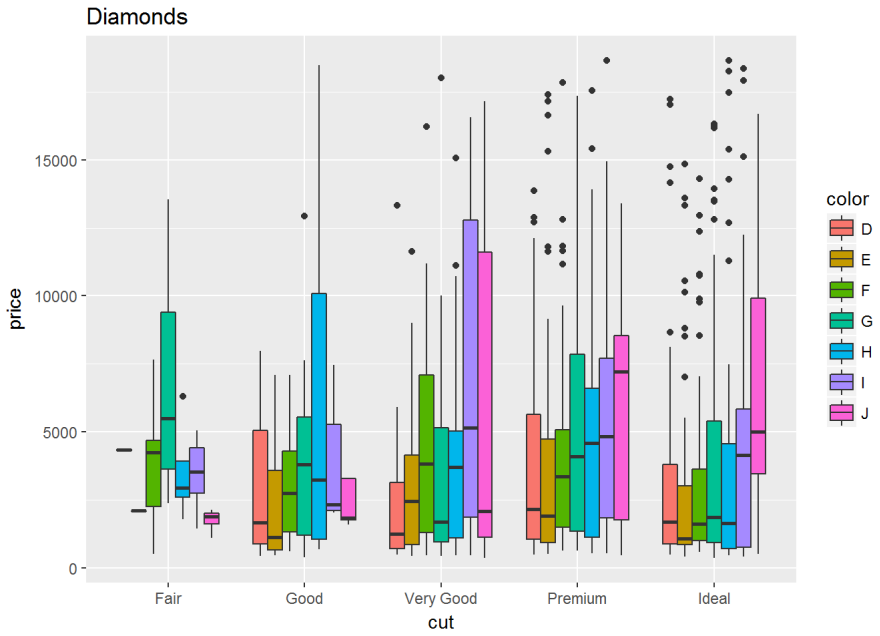


We can use the fill in geom_density also:

```
ggplot(small, aes(x=price,fill=clarity))+
  geom_density() +
  labs(title="Diamonds")
```

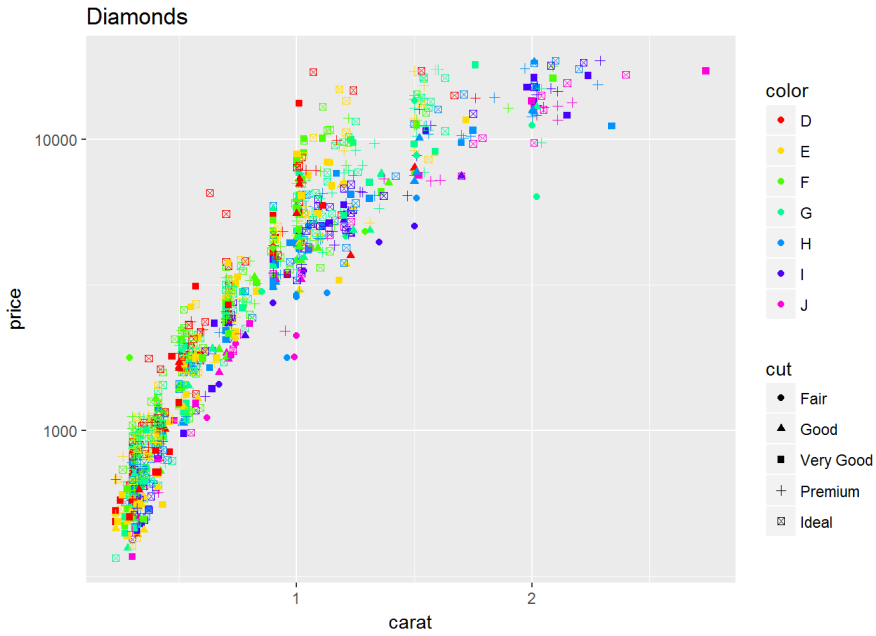



```
# boxplot
ggplot(small, aes(x=cut, y=price,fill=color))+
  geom_boxplot() +
  labs(title="Diamonds")
```

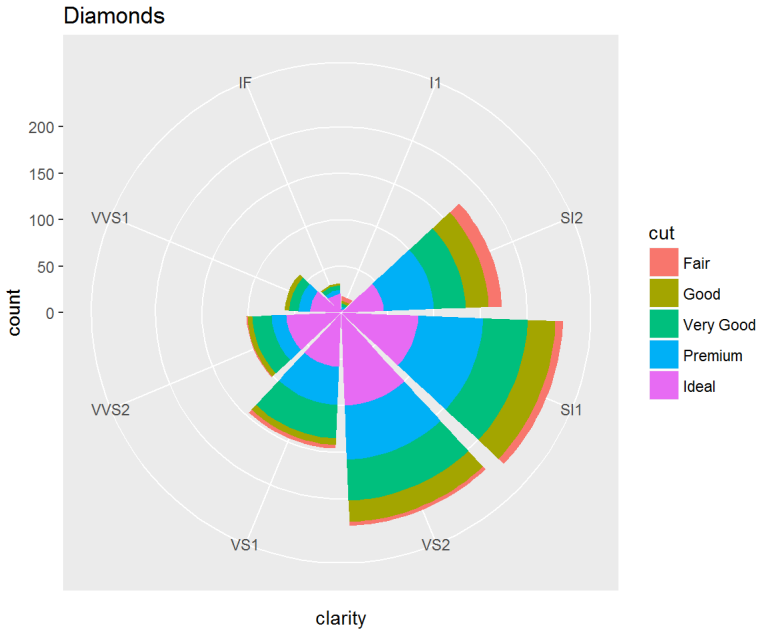


Isn't it very beautiful like a rainbow?

```
ggplot(small)+
  geom_point(aes(x=carat, y=price, shape=cut, colour=color))+
  scale_y_log10()+
  scale_colour_manual(values=rainbow(7)) +
  labs(title="Diamonds")
```

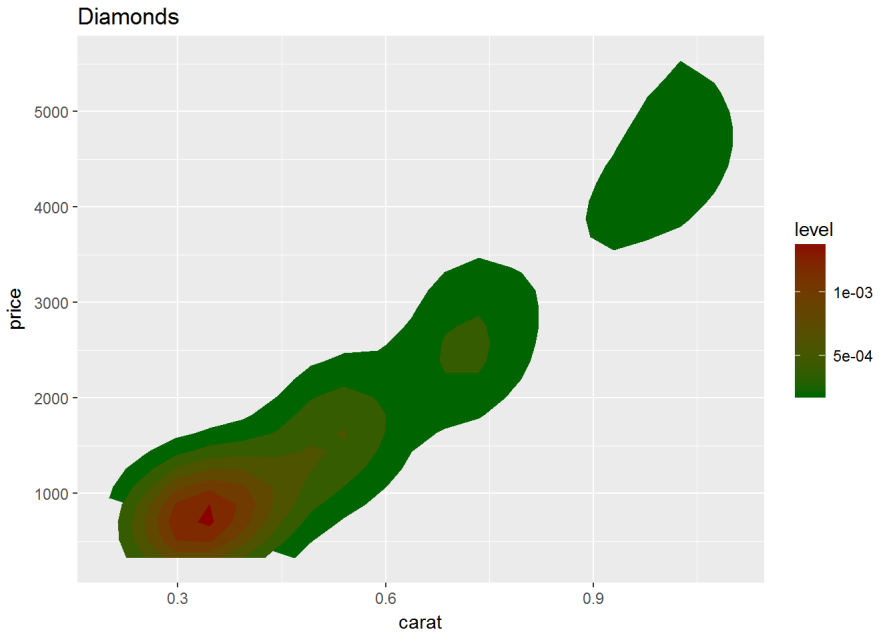



```
# Windrose
ggplot(small)+
  geom_bar(aes(x=clarity, fill=cut))+
  coord_polar() +
  labs(title="Diamonds")
```



When the data is too large, we can use densiry2d to show:

```
ggplot(diamonds, aes(carat, price))+
  stat_density2d(aes(fill = ..level..), geom="polygon") +
  scale_fill_continuous(high="darkred",low="darkgreen") +
  labs(title="Diamonds")
```



Conclusions

In this post, we talk about
- Background.
- Basic Structure of ggplot2.
- Amazing graphs from ggplot2.

Obviously, ggplot2 is a very easy, convenient, and useful tool for us to analyse our data even though the data is huge like data “diamonds”. I hope you would like this post, and more importantly, know how to use ggplot2 to draw your graphs and like to use ggplot2 from now on.

Reference

1. https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
2. https://stackoverflow.com/questions/39255781/what-is-difference-between-geom-point-and-geom-jitter-in-simple-language-in-r
3. https://www.r-bloggers.com/part-3a-plotting-with-ggplot2/
4. https://github.com/ucb-stat133/stat133-fall-2017/blob/master/slides/14-ggplot-lecture.pdf
5. http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html
6. http://r-statistics.co/ggplot2-Tutorial-With-R.html
7. https://en.wikipedia.org/wiki/Ggplot2
8. http://ggplot2.tidyverse.org/reference/#section-aesthetics