Post01

Zishen Cheng

10/28/2017

Graphing system in R, and how they help with interpreting data

Introduction

Before taking stat133 I had a little experience on programing and manage with data. In all my economic class and cs class I used java for programing and excel to deal with data. Both these two tools are good and sufficient at that time I was using it, but not very efficient. However, after get in touch with R, I realize that R can make my life much easier when analyzing and manipulating data. I'll first start with the comparison between R and java. java is a really rigorous programing tool that everything follows strict logical rule. Though the language is not very intelligent, the strict logic it follows made me very happy with that. When manipulating data, R just make everything so easy and more beautiful. I was impressed every time in all of our labs and lectures by R that how it simplified difficult programing tasks. So far in our labs and lectures we've learn what are vectors, what structure are factor, list and data frame, how should we manipulate data frame using bracket notation and dollar operator, how to efficiently use package ggplot2 and dplyr. In this post I'll summarize and further explore some feature and a package that would be useful in all area of study using R. In this post, we are using NBA players data and a random time series data called ts1.

Useful graphing function and package from our class

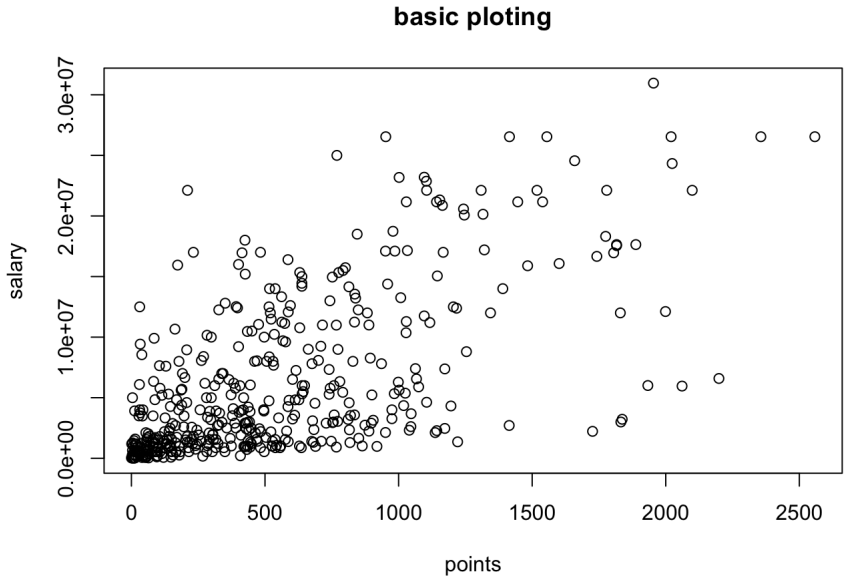
First here I'm going to do a summary about what we've covered in labs and in lecture about graphing. Two main ways we are using in lectures and labs are plot function and ggplot2 pacakges.

Simple plot

```
# load data in your R session

load('~/nba2017-salary-points.RData')
#head(player)
```

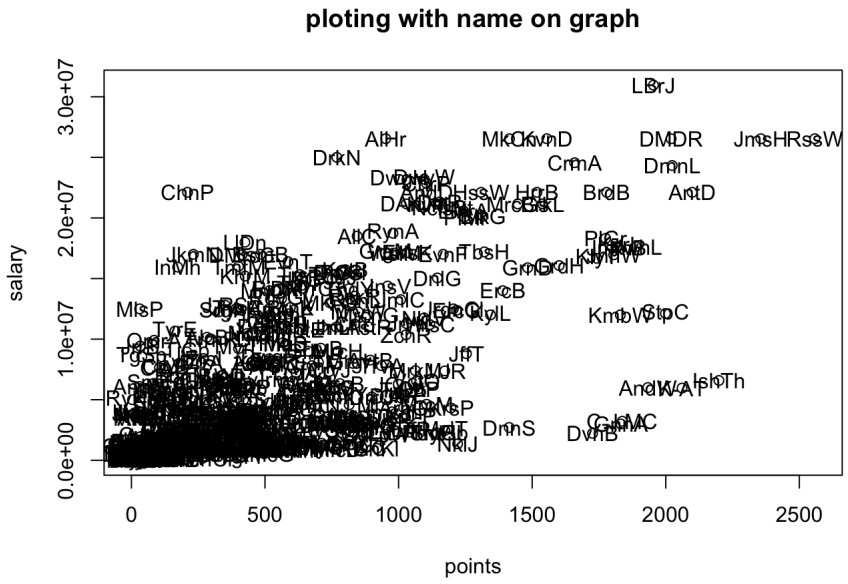
This is the most basic ploting how should we assign our x axis and y axis.

```
plot(points, salary)
title('basic ploting')
```



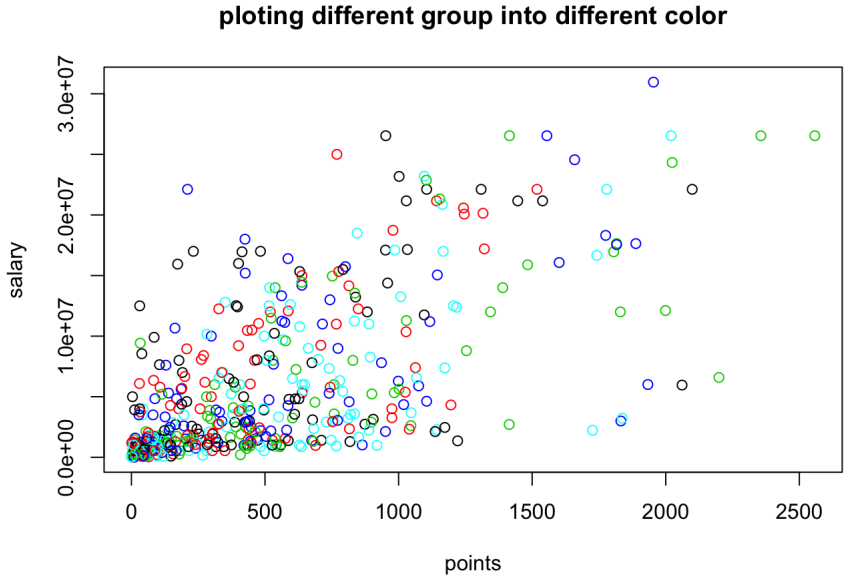
With additional function text, wen can also provide what should be printed for the name of x-axis, y-axis and each data point. Note that the text function can not run by it self. When we using this function we should always have a plot function prior to it (on top of it adjacently). So It might be better if we put plot and text in the same row.

```
plot(points, salary)
text(points, salary, labels = abbreviate(player))
title("ploting with name on graph")
```



"col" is one argument took in plot function, this argument can plot different data into different color base on how we group them. Therefore we can get a more clear interpretation from colored data.

```
# your colored scatterplot
position_fac <- factor(position)
plot(points, salary, col = position_fac)
title("ploting different group into different color")
```



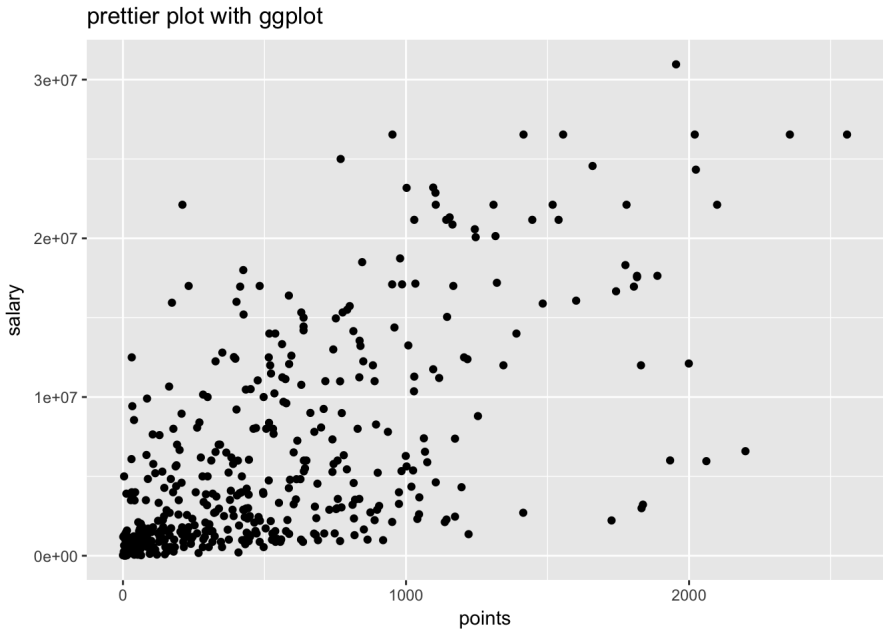
ggplot2 package

The package `"ggplot2"` is probably the most popular package in R to create beautiful graphics. In contrast to the functions in the base package `"graphcics"`, the package `"ggplot2"` follows a somewhat different philosophy, and it tries to be more consistent and modular as possible.

```
dat <- read.csv('nba2017-players.csv', stringsAsFactors = FALSE)
```

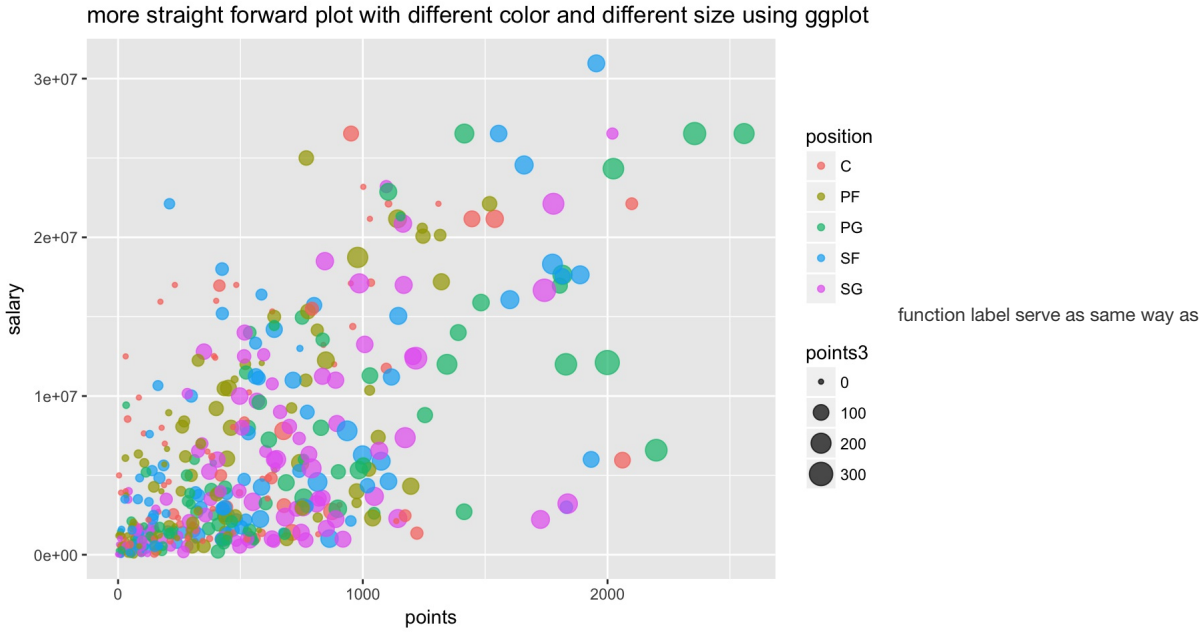
`"geom_point"` gives us most basic scatter plot for the data. in the bracket of `"ggplot"`, it takes our data frame as argument. and in the bracket of `"aes"` we can tell the function what is our x axis and what is our y axis.

```
ggplot(data = dat) +
  geom_point(aes(x = points, y = salary)) +
  ggtitle("prettier plot with ggplot")
```



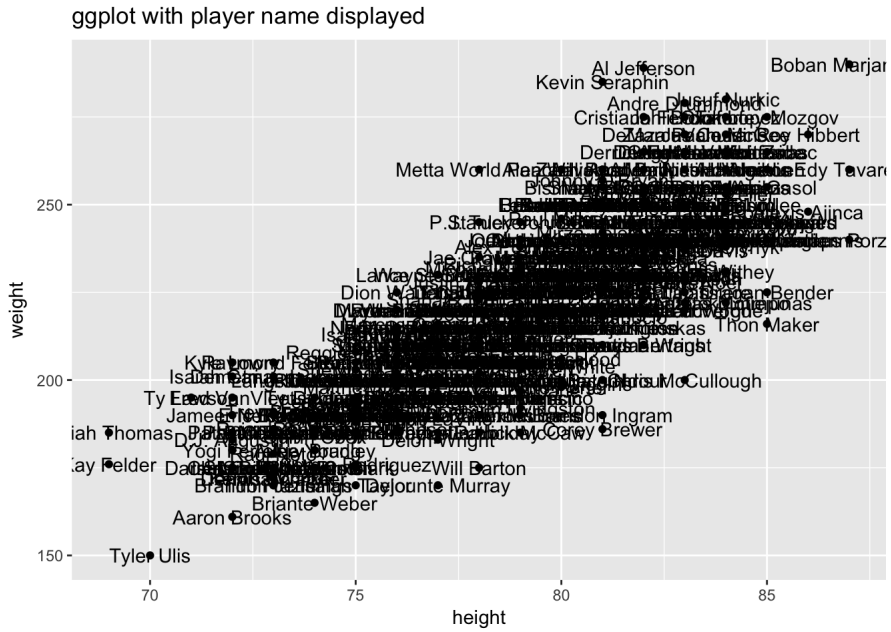
Since we noted that `"ggplot"` can provide us a more beautiful diagram, the we can infer that it can definately change the color and the size of the points. Further to make these points even more prettier, we can let these points transparently shown on the graph.

```
# sized and colored scatterplot
ggplot(data = dat, aes(x = points, y = salary)) +
  geom_point(aes(color = position, size = points3), alpha = 0.7) +
  ggtitle("more straight forward plot with different color and different size using ggplot")
```



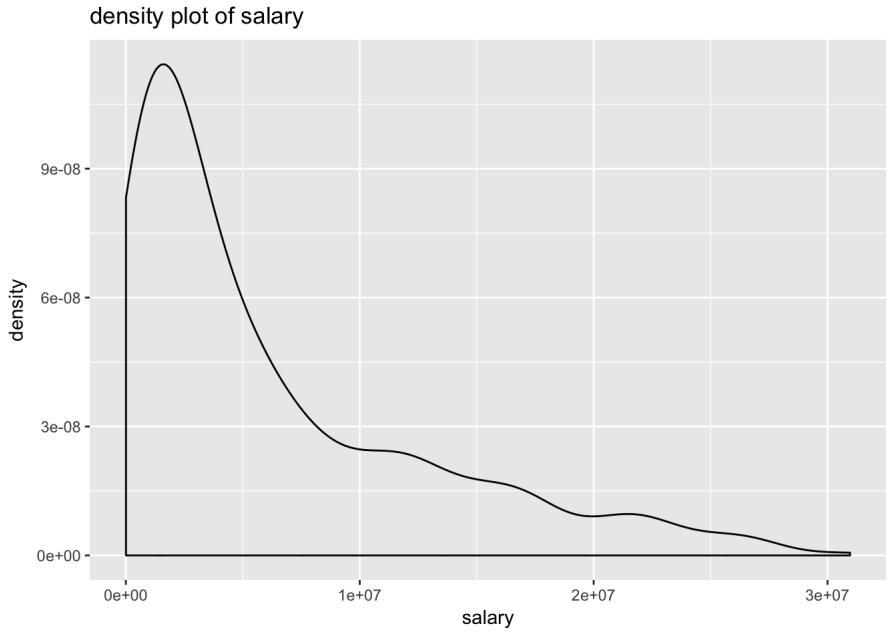
function label serve as same way as label in the regular plot. Using `"geo_text"` is ggplot version of changing points into text that represent those points.

```
ggplot(data = dat, aes(x = height, y = weight)) +
  geom_point() +
  geom_text(aes(label=player)) +
  ggtitle("ggplot with player name displayed")
```



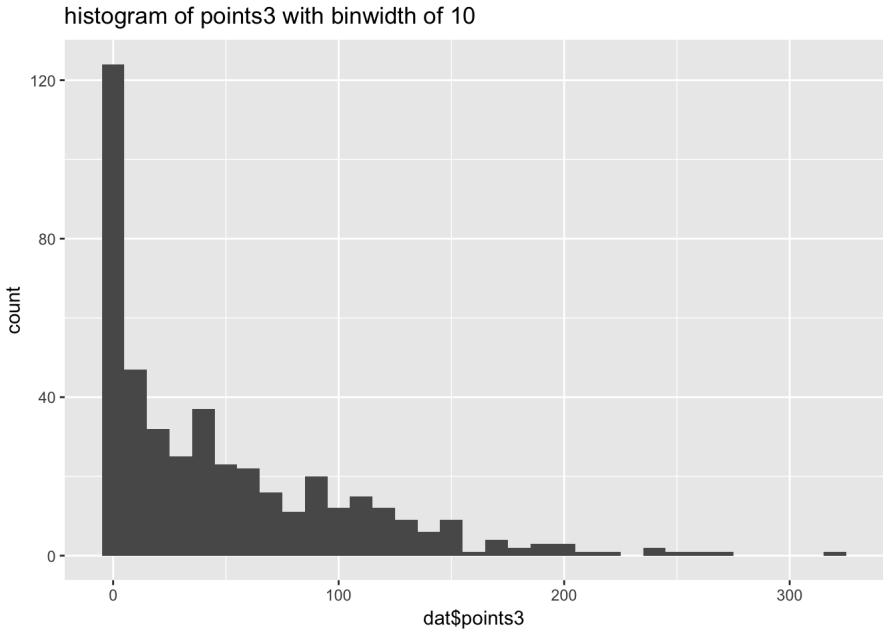
Other than scatter plot there also lots of different kinds of plot can be drawn by ggplot package. These other ploting function is well organized and much easier for us to visualize and anaylize data. density plot of `salary` (for all NBA players)

```
ggplot(data = dat, aes(salary)) +
  geom_density() +
  ggtitle("density plot of salary")
```



histogram of `points3` with binwidth of 10 (for all NBA players)

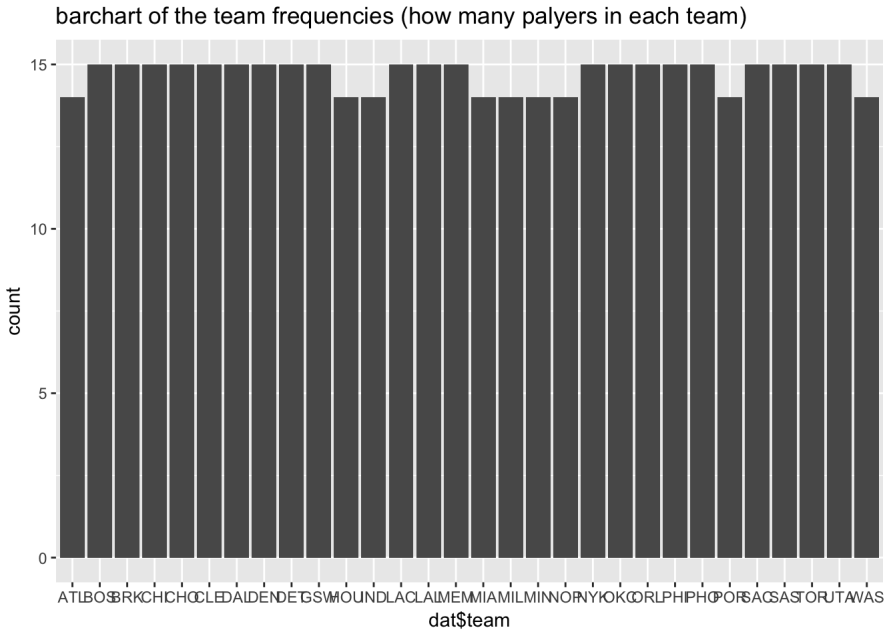
```
ggplot(data = dat, aes(dat$points3)) +
  geom_histogram(binwidth = 10) +
  ggtitle("histogram of points3 with binwidth of 10")
```



barchart of the `team` frequencies

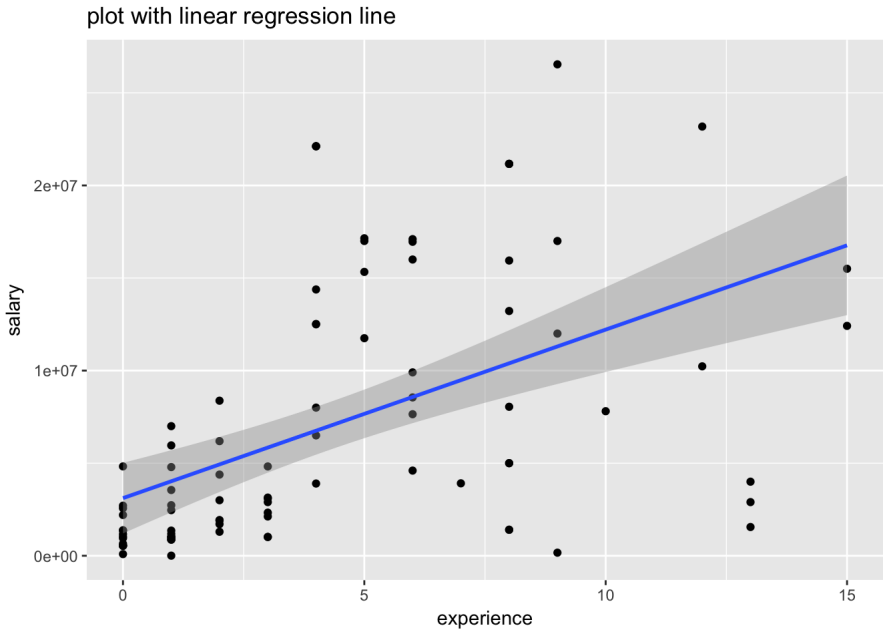
(for all NBA players)

```
ggplot(data = dat, aes(dat$team)) +
  geom_bar() +
  ggtitle("barchart of the team frequencies (how many palyers in each team)")
```



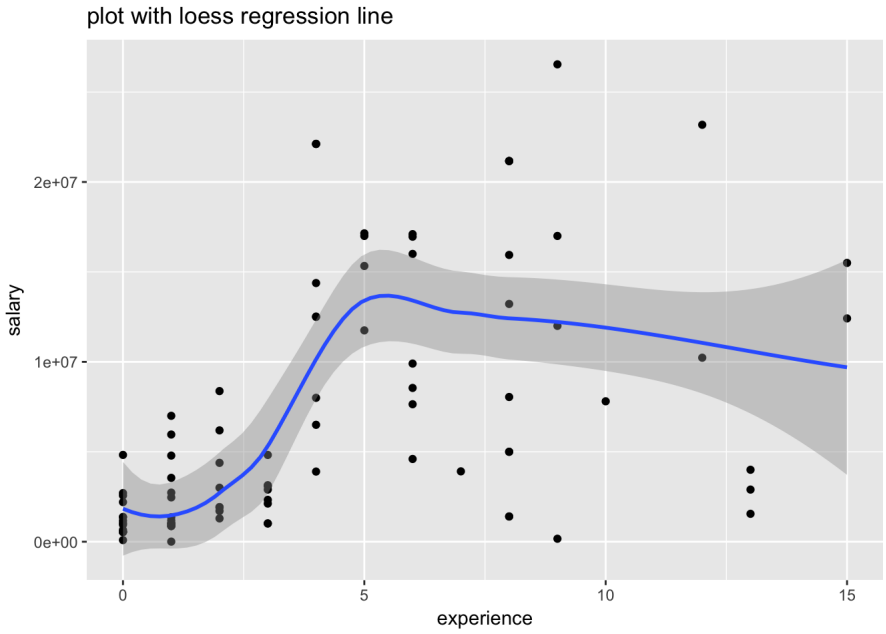
The gg plot package also provide another feature that allows user to add lines to the graph scatterplot of `experience` and `salary` of all centers, and use geom_smooth to add regression line to our scatter plot.

```
ggplot(data = filter(dat, position == 'C'), aes(x = experience, y = salary)) +
  geom_point() +
  geom_smooth(method = 'lm') +
  ggtitle("plot with linear regression line")
```



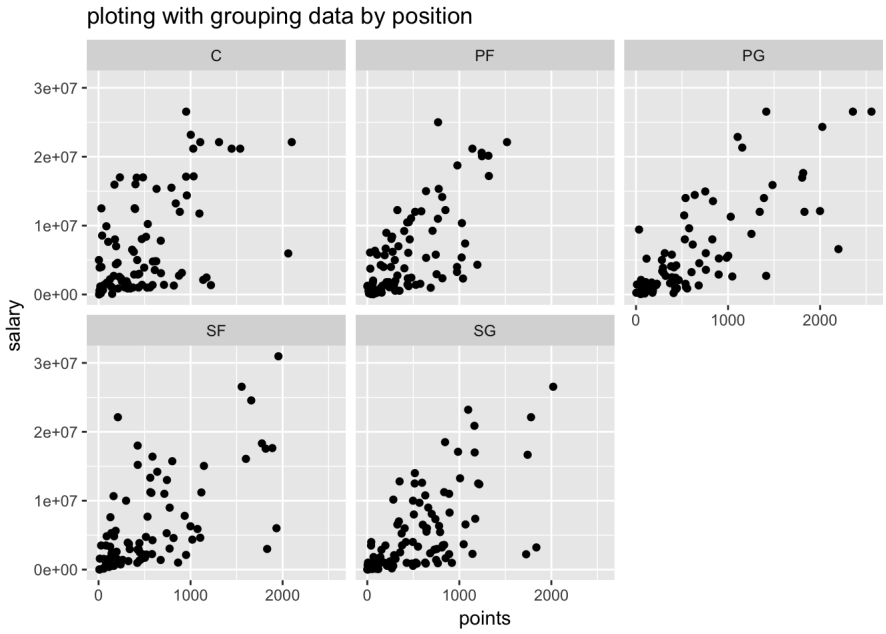
We can also try adding another kind of regression line wich is loess(locally weighted scatterplot smoothing)

```
ggplot(data = filter(dat, position == 'C'), aes(x = experience, y = salary)) +
  geom_point() +
  geom_smooth(method = loess) +
  ggtitle("plot with loess regression line")
```



faceting is one feature of ggplot we covered in class and I found it to be really useful. This feature group up and ploting these different property datas in different chart. By which it provide a very clear interpretation for data.

```
# scatterplot by position
ggplot(data = dat, aes(x = points, y = salary)) +
  geom_point() +
  facet_wrap(~ position) +
  ggtitle("ploting with grouping data by position")
```



extracurricular ploting that helps analysing data. (useful plot involved in time series analysis.)

background

time series package In this section of the post, I'm going to introduce some other useful plots that can help us interpreting data. time series is a very important statistic concept. Since, analyzing data, almost all fields of study have data that relate to time difference. Time series can be use in various of subjects. There are some other ploting functions that provide very intuitive understanding for data that relate to time series. Also another really important concept in time series is after we fitting data with a model, we need a residual analysis.

ARMA(p,q) model: This model is the combination of two ways we relate current data point to previous values and errors, Namely AR(p) (autoregressive model) and MA(q)(moving average model) the number in the bracket refers to how many time lag each model depend on.

In time series we always care about the correlation between different time segment. And in time series case, we call the correlation between different time segment autocorrelation. And later when we want to analyze residuals, we are hoping these residuals can be distributed normally. Therefore we also have some really good plots that can help us assess the normality of variables.

```
library(ggplot2)
library(tseries)
library(TSA)
```

```
## Loading required package: leaps
```

```
## Loading required package: locfit
```

```
## locfit 1.5-9.1    2013-03-22
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

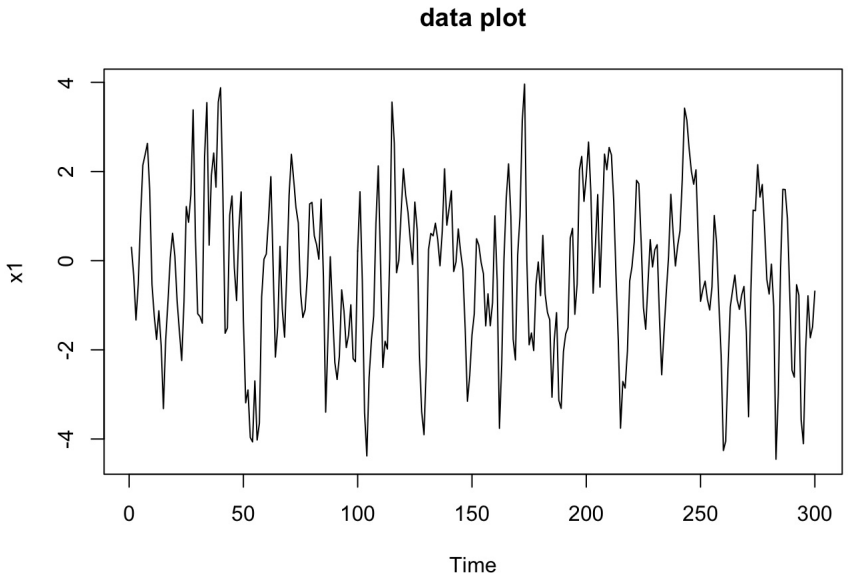
```
## This is mgcv 1.8-15. For overview type 'help("mgcv-package")'.
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

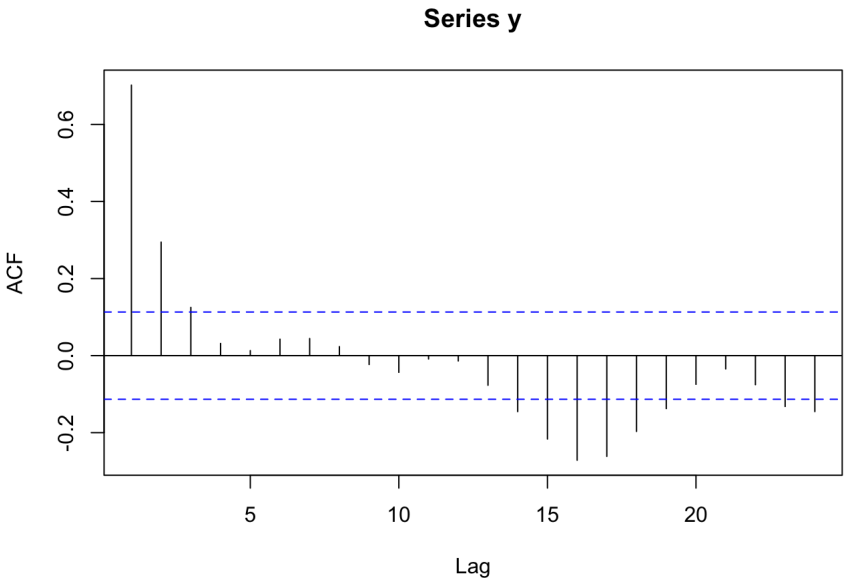
```
## The following object is masked from 'package:utils':
##
##     tar
```

```
unloadNamespace("dplyr")
load('ts1.RData')
dat <- data.frame(y)
plot(dat)
title('data plot')
```

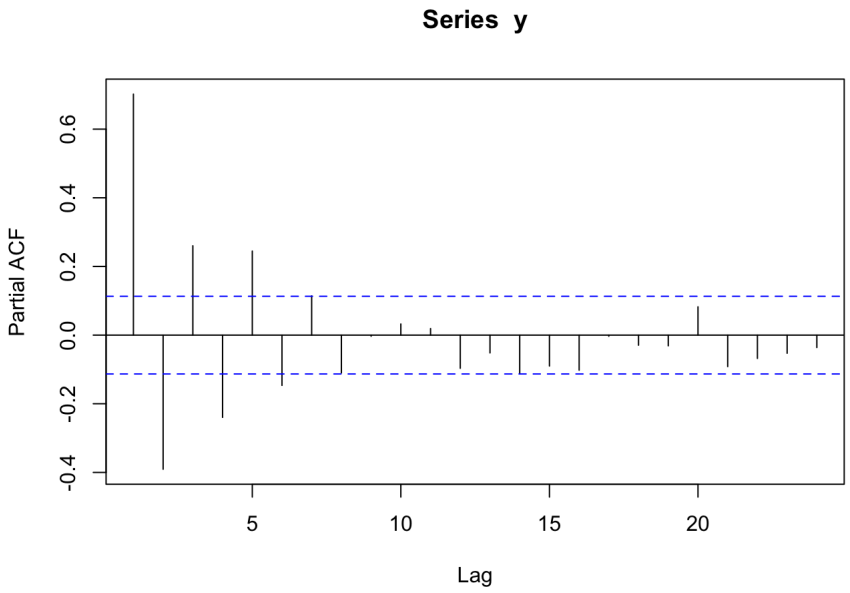


Dealing with my data, I'm using auto regressive moving average model, which the characteristic of this model assumes that current data point has autocorrelation with value of previous time points and has correlation with error of previous time points. Therefore, we need to look and the graph of the auto correlation function that can help us to determine which time lag is significant.

```
acf(y)
```




```
## using acf to find how many time lag our Yt is depend on error term , the correlation with timelag 1 is much mor
e significant than other, so we may conclude this is MA(1)

pacf(y)
```



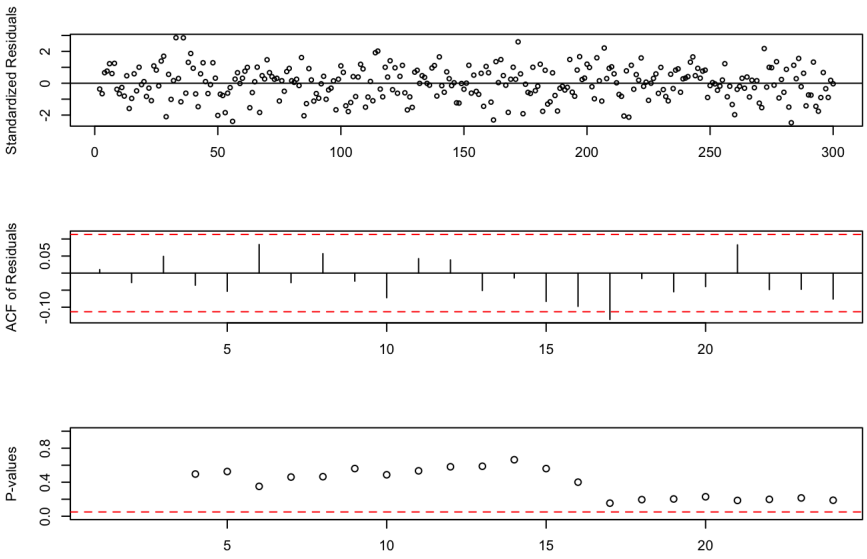

```
## using pacf to find how many time lag our Yt is depend on previous Yt-k, the correlation with timelag 1 is much more significant than other, so we may conclude this is AR(1)

eacf(y)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x o o o o o o o o  o  o  x
## 1 x o o o o o o o o o o  o  o  o
## 2 x o o o o o o o o o o  o  o  o
## 3 x o x o x o o o o o o  o  o  o
## 4 x o x x o o o o o o o  o  o  o
## 5 x x x x o o o o o o o  o  o  o
## 6 x x o x o o o o o o o  o  o  o
## 7 x o x x o o x o o o o  o  o  o
```

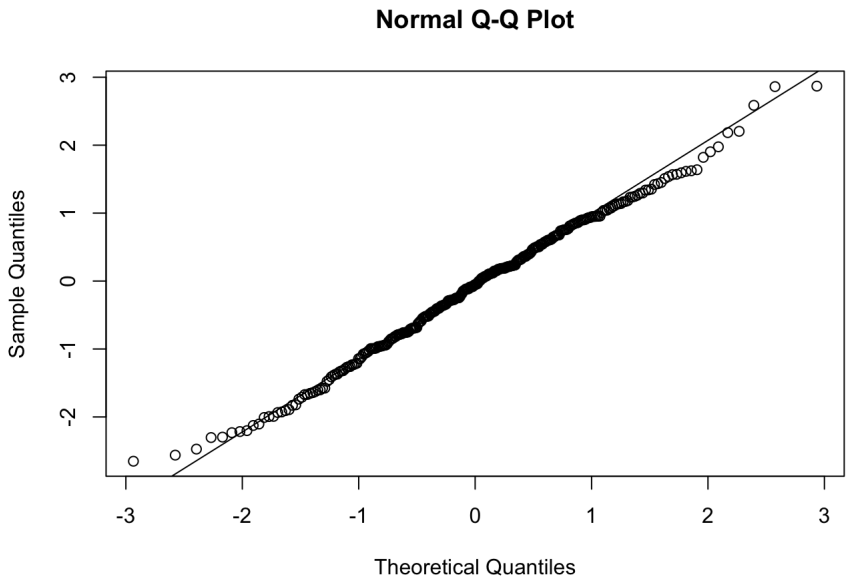
```
## the eacf chart shows that the first circle shows at 1,1 which refers that this timeseries data can be discribe as ARMA(1,1)
```

Function `"tsdiag"` help us analyzing the residues we got when we fit our value into our model. There will be three graph generated. The first one shows how standardized residual distribute along time. If they seemes random, we may interpret our residual is normally distributed and therefore is a nois. The second graph we are making sure that there are no autocorrelation between each error term, because we are expecting them to be iid. The last graph ploted by tsdiag is the pvalue of our residues. As these p-values are aboe the red line, we can infer that our residuals are noise. `"qqnorm"` is another good way to interpreting erros. It compare quantilewise how our residuals fit with a normal distribution. Therefore, if we get our qqnorm plot which all the points are close to the line. We can say that our residuals are normally distributed whitenoise.

```
x <- arima(y, order = c(1, 0, 1))
tsdiag(x, include.mean = FALSE)
```



```
qqnorm(residuals(arima(y, order = c(1, 0, 1), include.mean = FALSE)));qqline(residuals(arima(y, order = c(1, 0, 1)
, include.mean = FALSE)))
```




```
##from function tsdiag() we cna see that acf are mostly within the confident interval and pvalue is quiet signific
ant. Standard residual distribute normally. Further look at qqnorm and the qqline, our qqplot is pretty much a str
aight line
```

Conclusion

In conclusion, R really makes our life easier when analyzing data. With only looking at how convinience it graphing system is, I, myself, would prefer using R when interpreting data. Visualizing our data is one very important ways to interpret our data. And thus why it is so important to have a good graphing system or graphing package for our programing language. Including the most standard one, All graphing package are really advance. ggplot2 and tsa packages make us able to easily interpreting our data.

Reference:

-1.http://uc-r.github.io/quickplots#replication
-2.http://uc-r.github.io/ggplot
-3.[Jonathan_D._Cryer,_Kung-Sik_Chan]_Time_Series_Ana(BookFi)-edition2
-4.http://www.etsii.upm.es/ingor/estadistica/Carol/TSAtema4petten.pdf
-5.http://blog.minitab.com/blog/adventures-in-statistics-2/why-you-need-to-check-your-residual-plots-for-regression-analysis
-6.https://github.com/ucb-stat133/stat133-fall-2017/blob/master/labs/lab02-vector-basics.Rmd
-7.https://github.com/ucb-stat133/stat133-fall-2017/blob/master/labs/lab05-dplyr-ggplot-basics.Rmd