# post01-haoyu-chen

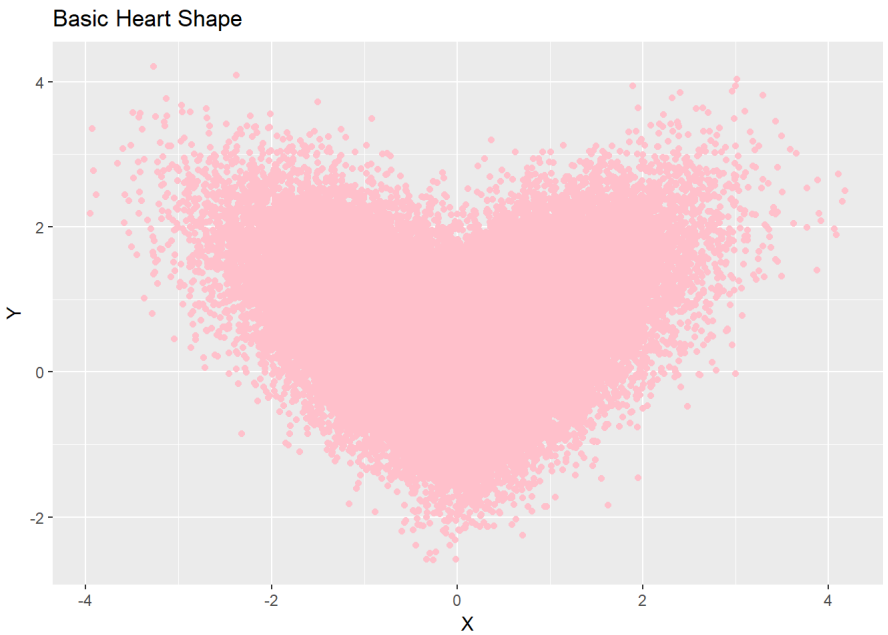
Fabulous Graphs with ggplot in RStudio

We have been so far using the function ggplot() a lot in class. Our first contact with function ggplot() in homework 05 has already been very fascinating. We generated basic scatterplots about the relationship between height and weight of Golden State Warriors players, density plots with respect to NBA players' salary, histograms of 2-point field goal distribution, barplots for positions frequency, and so forth. Later on, we moved a little step further by introducing more complex coding syntax in order to be able to generate enriched graphics, for example, graphics with regression lines and loess lines or with facet functions to divide plots into several subsections. With those skills, we were able to perform a deeper analysis by applying ggplot function to data frames. Examples include: barchart of players' efficiency with respect to points, rebounds, steals, turnovers, etc., scatterplots of efficiency and salary with regressions, ranking of teams based on their total salary, total points, total efficiency, and rescaled PC1. ggplot has facinated me by showing me how the raw, plain, and uncorrelated (at very first look at) data from basketball court could be interpreted, modified, and turned into useful, intuitive, and interest-grasping graphs, which could be used for further analysis for player trainings, strategies planning, or contract and salary determining. However, other than the practical usefulness of the graphs generated by function of ggplot, the readible codes, the charming colors, and the ease of revising and editing evrything are all the reasons that the ggplot looks one of the most powerful tools in RStudio. Therefore, I am going to do some more deeper explorations on the function of ggplot that may help me know more about it. I will show some very interesting graphs by using ggplot function here in this post.

1. Artistical Graphs by Basic R Codings

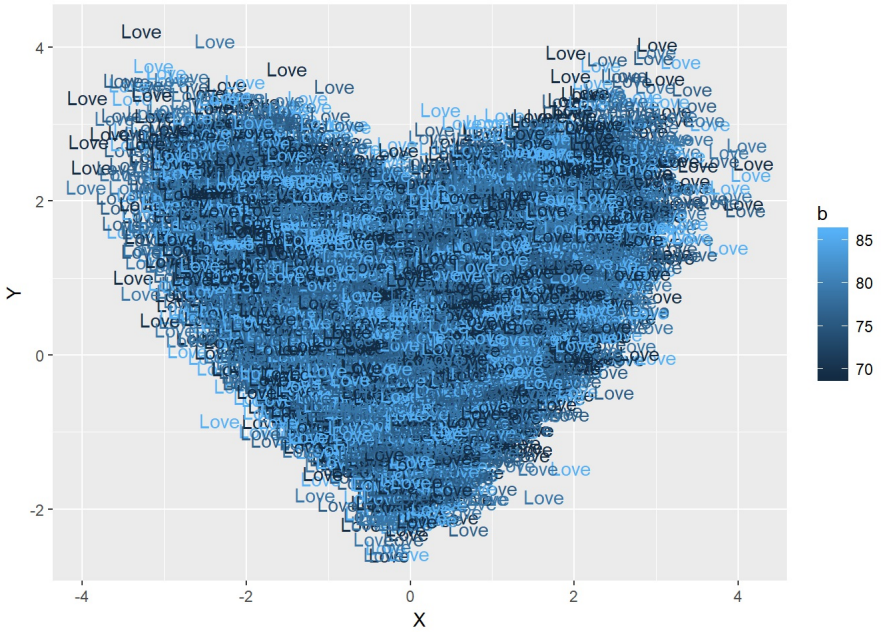
To generate nice pictures doesn't require much coding skills. Based on what we have learned in the class, we can plot very nice and interesting graphs with very basic codes. As long as users are talented, creative, and probably romantic, it is very easy to plot facinating pictures. Here I will plot a image of heart by using only ggplot and its associated function: geom_point. The idea here is, we let RStudio generate random points in a given area of shape of a heart and all points adding up to make it a shape which looks like a heart.

```
library(ggplot2)          #loading ggplot2
n=60000                   #6000 random points to generate
r=0.7;r_e=(1-r*r)^.5      #the radius of range for each point to be generated
X=rnorm(n)                #to generate points with normal distribution
Y=X*r+r_e*rnorm(n)        #setting the range (a circle area around each point)
Y=ifelse(X>0,Y,-Y)        #let y to be positive when x is positive and y to be negative when x is so
heart <- data.frame(X, Y) #combine X, Y and get a new data frame heart
ggplot(heart) +             #apply ggplot
  geom_point(aes(x = X, y = Y), col = 'pink') +
  ggtitle("Basic Heart Shape")
```




I can then make some tiny adjustments which will definitely make the heart look even better. Keeping everything else unchanged, we replace the geom_point with geom_text, and set the text to be characters of "LOVE". for example, replacing the points with labels of "love", and plain pink color to be gradient?

```
b <- sample(c(76,79,86,69),60000,T) #colors appear repeatedly for points
heart <- data.frame(X, Y)
ggplot(heart) +
  geom_text(aes(x = X, y = Y, label = "Love", col = b))
```




The point for these two ggplots is: ggplot does not have to be given data in order to generate plots. Like in both of these two cases, function rnorm serves to generate random variates that go into the function of ggplot. And also, R doen't have to be scientific; it could be fun and romantic!

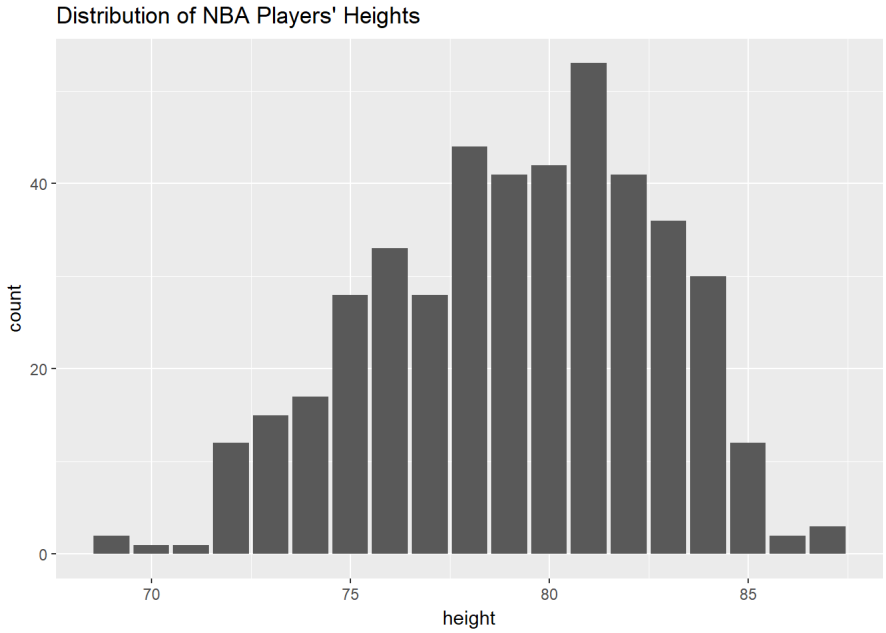
2. Next I am going to move on to some scientific graphs generation. we have done plots involving scatter points, lines, bars, histograms, etc in class, but the power of ggplot function is far beyond what we have tasted so far. I will present some different graphics that are helpful in statistical analysis. Also from this section, I will use the file nba2017-players, which was used for homework in class, as my data resources.

```
setwd("C:/Users/Haoyu/stat133/stat133-hws-fall17/Post01")
dat <- read.csv("nba2017-players.csv", stringsAsFactors = FALSE)
```

1. More about barplot

We always want tom make our barplot nicer. Filling bars with colors, reordering the bars, or arranging the relative positions of bars are all good choices. I first got a barplot based on players' heights like this:

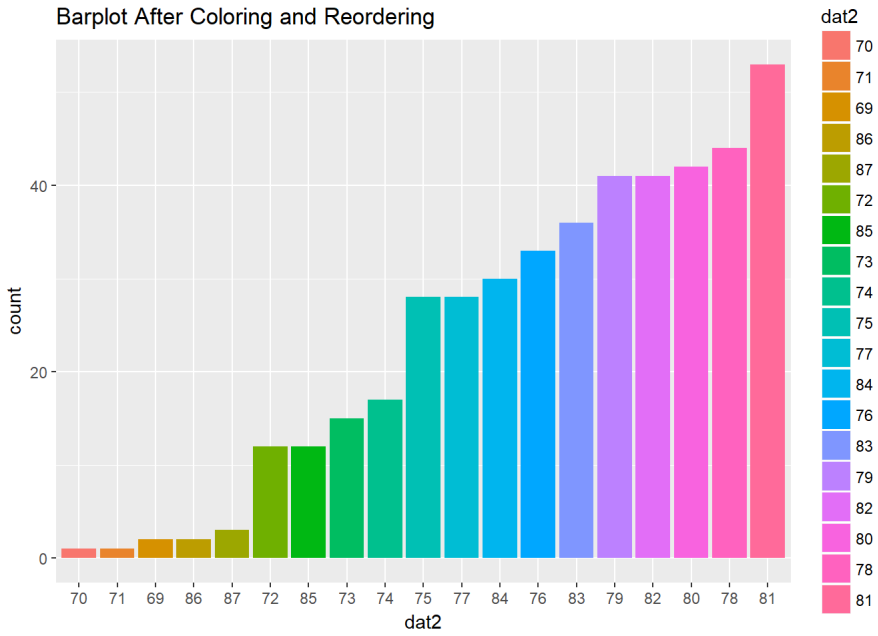
```
ggplot(dat, aes(x = height)) +
  geom_bar() +
  ggtitle("Distribution of NBA Players' Heights")
```




Reordering and coloring:

The picture looks really dull and is not very revealing. I will reorder the bars in ascending order, and fill them with different colors. The function I used for reordering is reorder(), and by using the statement "fill = dat2", we can achieve the goal of coloring. The code and the graphic are shown below:

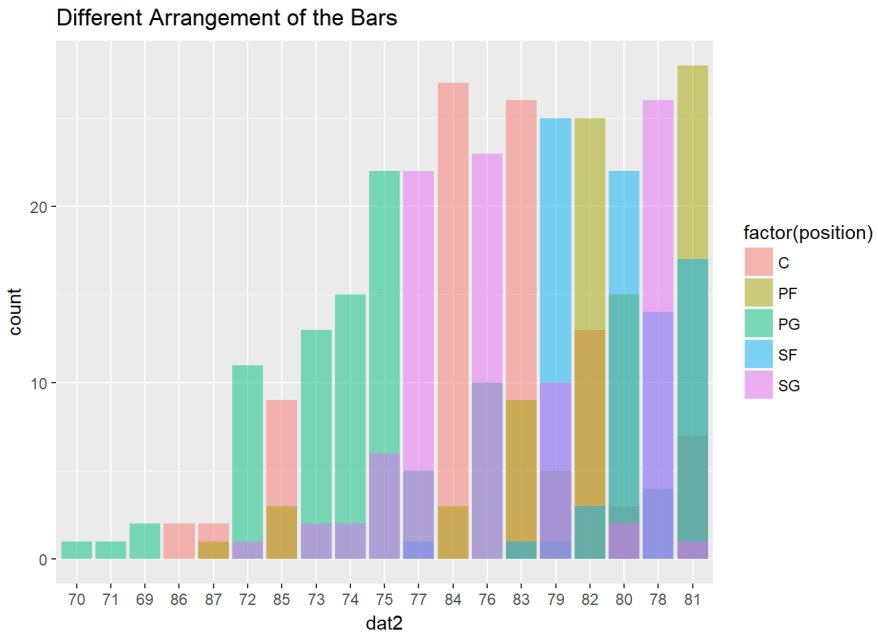
```
dat2<-dat$height
dat2<-reorder(dat2,dat2,length)  #reordering the bars in ascending order
dat$dat2<-dat2
ggplot(dat,aes(x=dat2)) +
  geom_bar(aes(fill=dat2)) +     #coloring
  ggtitle("Barplot After Coloring and Reordering")
```



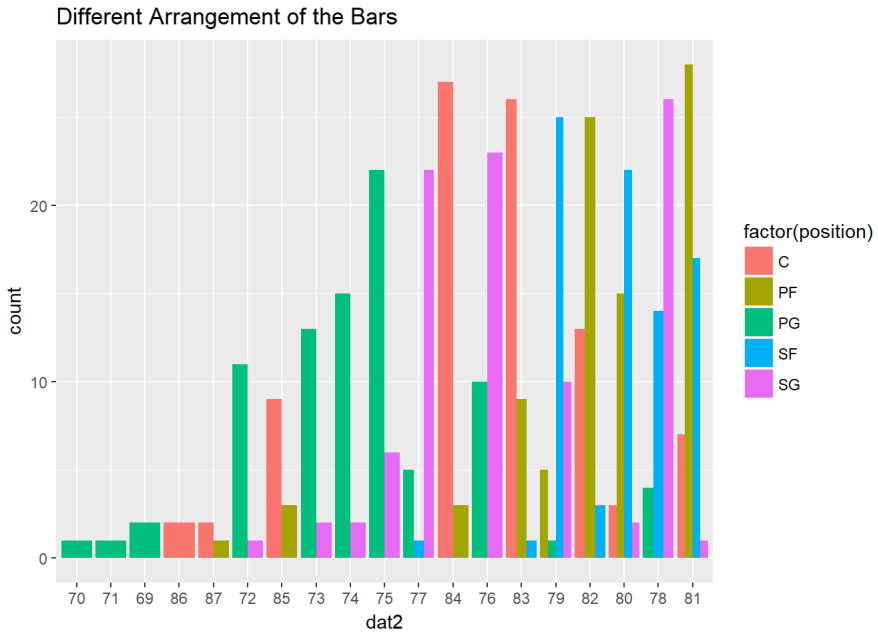

Arranging positions of bars:

Except for reordering, we can consider to arrange the relative position of bars to make them meet specific data analysis requirement or meet personal preferences. Here are three sample positions how the bar of two different groups can take up the position. Choosing position = “idnetity” gives the basic graph. Letting position = “dodge” adjusts bars horizontal position.

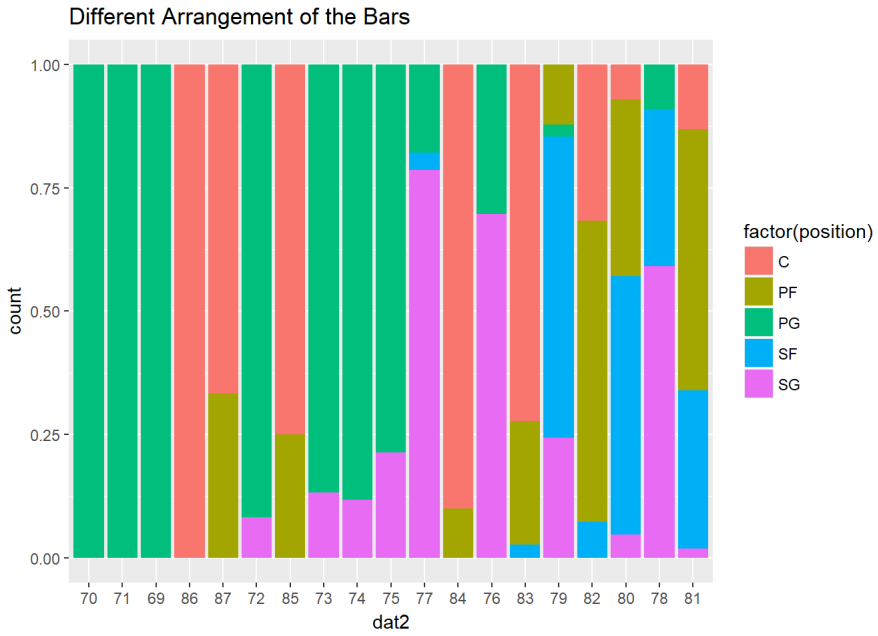
```
p<-ggplot(dat,aes(dat2,fill=factor(position))) +
  ggtitle("Different Arrangement of the Bars")
p+geom_bar(position="identity",alpha=0.5)     #identity
```




```
p+geom_bar(position="dodge")                  #dodge
```



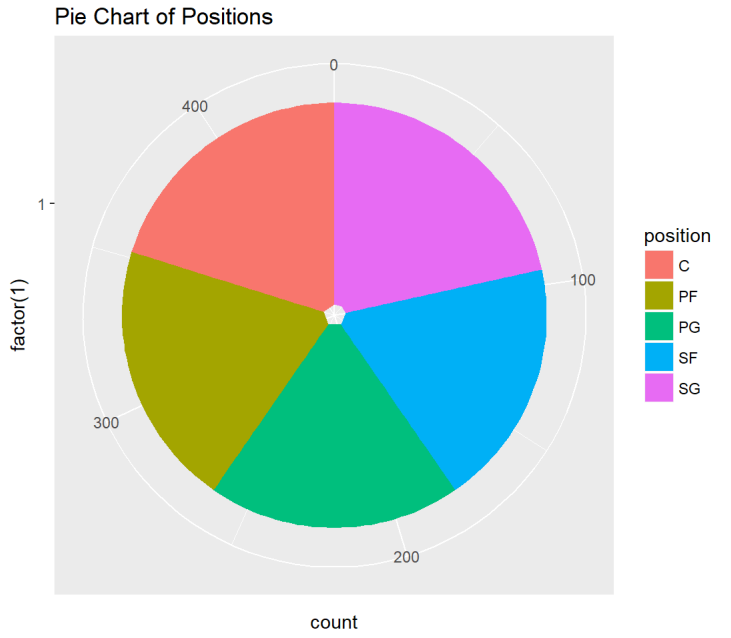

```
p+geom_bar(position="fill")                    #fill
```




2. Pie Chart

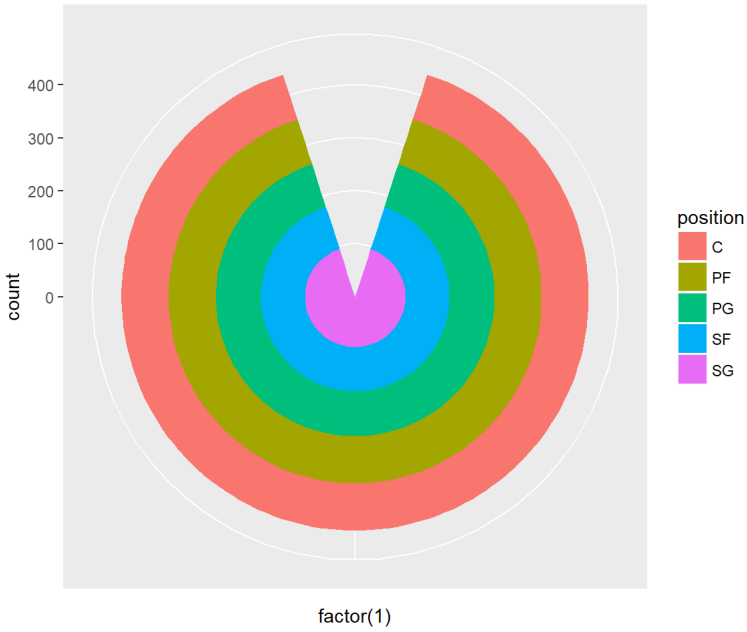
Pie is very useful in presenting what percentage each category takes up the the data set. To plot a pie chart, user just need to introduce geom_bar after applying function ggplot. The graph below is the relative percentage of the number of each position (Center, Power Forward, Small Forward, Shooting Guard, Point Guard) among all NBA players.

```
ggplot(dat) +
  geom_bar(aes(x=factor(1), fill= position)) +
  coord_polar(theta="y") +
  ggtitle("Pie Chart of Positions")
```

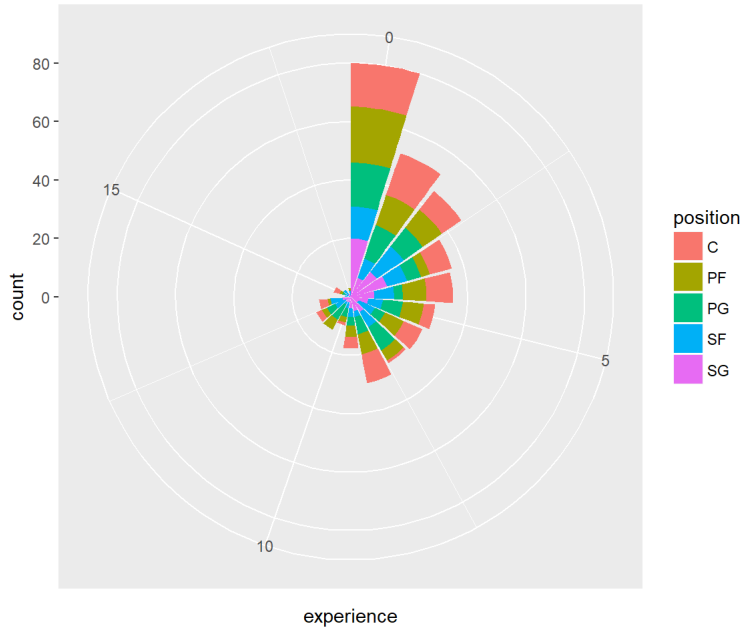



Let's make some changes in the code. We then can get donut plot and windrose plot. These can be easily achieved by changing the statement inside of the parenthesis of aes() as the followings:

```
ggplot(dat)+geom_bar(aes(x=factor(1), fill=position))+coord_polar()
```



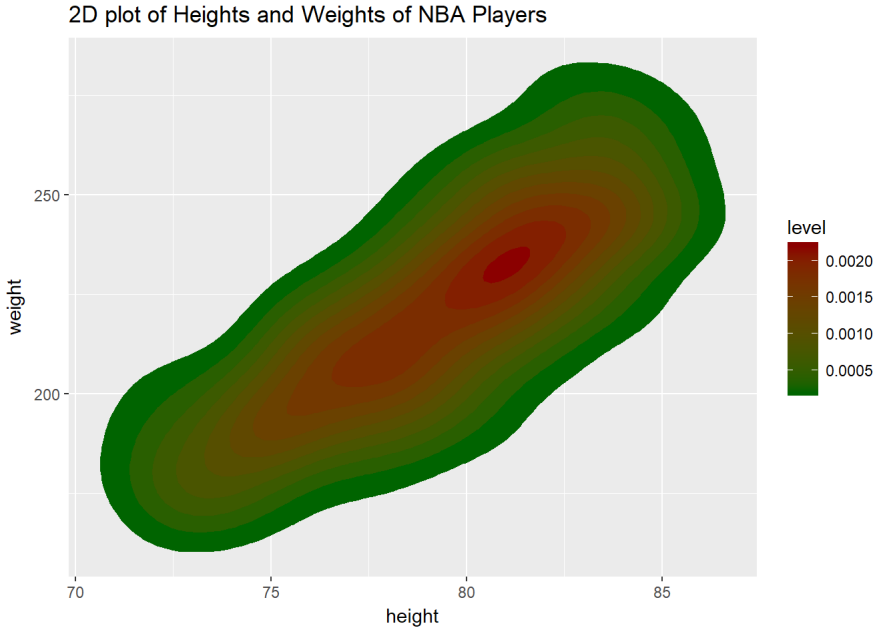

```
ggplot(dat)+geom_bar(aes(x=experience, fill=position))+coord_polar()
```




3. 2-D Density Plot

We have tried to use the function geom_density() to add density lines to a scatterplot. That motivates my thought about if there is a function similar to geom_density that will generate 2-D Density plot. There is! The function stat_density2d is designed to work for this, as the following code shows:

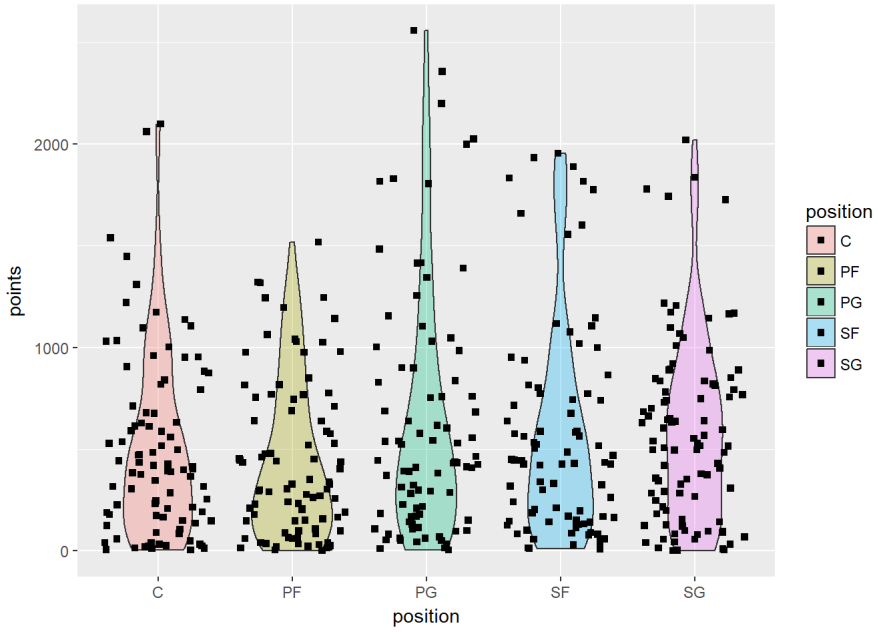
```
ggplot(dat, aes(height, weight))+
  stat_density2d(aes(fill = ..level..), geom="polygon") +  #filling colorings based on the level
  scale_fill_continuous(high='darkred',low='darkgreen') +
  ggtitle("2D plot of Heights and Weights of NBA Players")
```




4. Violin Plot

Violin plot is very similar to boxplot. However, it also shows the probability density of the data at different values. It is more informative than boxplot because it shows not only the summary statistics (i.e. means, average, range), it displays the full distribution of data, especially when it comes to mulimodal data distribution. Here is how RStudio shows a violin plot:

```
ggplot(dat,aes(position,points, fill=position)) +
  geom_violin(alpha=0.3,width=0.6)+geom_jitter(shape=15) #setting the width and the shape
```




In this section, we went over some other commonly-used ggplot functions, which gave users very useful ggplot graphics in data analysis, such as bar plots, pie chart plots, 2-D density plots, and violin plots. We can learn from this section that the function for a particular graphic in ggplot (i.e. violin graphic) is always in the form of geom_ + graphic name, for example geom_violin. As such, it is very user-friendly that people can easily remember what a function should be like when the functions are needed.

3. Drawing Maps with R

Next, I am going to show a very interesting way we can play with ggplot2, which is graphing a map. There are, of course, many different ways in RStudio to plot maps, but ggplot provides users with the easiest coding expereince and more aesthetic graphics when plotting a map. Here I will do a simple plotting about Chinese map in order to show the idea. First, we load some necessary functions for mapping.

```
library(maptools)
```

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

```
## Loading required package: sp
```

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

```
## Checking rgeos availability: FALSE
##      Note: when rgeos is not available, polygon geometry     computations in maptools depend on gpclib,
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

```
library(mapdata)
```

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

```
## Loading required package: maps
```

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

```
library(plyr)
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:maps':
##
##     ozone
```

Plotting maps require much knowledge about R coding which will go beyond our ability if we want a wonderful map. It also requires some data files which is hardly to get for free from website. As such, I refered to and modified the code on website http://bbs.pinggu.org/forum.php?mod=viewthread&tid=4182165&page=1 into the form that we students can read and understand it. The first few lines are just simple manipulations such as reading data, converting data into data frame, and combining two data frames. The second part is where we applied ggplot.

```
china_map <- readShapePoly("bou2_4p.shp")
```

```
## Warning: use rgdal::readOGR or sf::st_read
```

```
china_map1 <- china_map@data
china_map1 <- data.frame(china_map1,id=seq(0:924)-1)
china_map2 <- fortify(china_map)                    #convert to data frame
```

```
## Regions defined for each Polygons
```

```
china_map3 <- join(china_map2, china_map1, type="full", by ="id")


p <- ggplot (china_map3,aes(x=long,y=lat),colour="gray40")
p <- p+
  theme(
    panel.grid = element_blank(),
    panel.background = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.title = element_blank(),     #setting some lables to be blank
    legend.position = "none"
    )

p + geom_polygon(aes(group=group, fill = NAME), colour="grey60")  #Filling provinces with colors
```

Of course, we can further improve the map, for example, by adding provinces names and capital cities, and even adding circles to represent the coordinates of the cities. However, those steps are easy because we can technically use function geom_text to add province names and use function geom_point to add coordinates of capital cities.

Conclusion: This post exibits a powerful tool, namely ggplot, in RStudio for graphics. ggplot has been prefered by more and more users because of its convinience for users to remember the associated functions, more aesthetic graphining qualities, and its usefulness in presenting data in different ways in order to meet different data analysis requirements, which we have seen in this post. To be more familiar with ggplot, users just need to get more practice and it will definitely make things easier for us to generate more fun, aethestic, and revealing graphics.

References:


https://www.plob.org/article/1221.html http://rstudio-pubs-static.s3.amazonaws.com/3355_d3f08cb2f71f44f2bbec8b52f0e5b5e7.html
http://www.cnblogs.com/nxld/p/6059603.html http://www.r-graph-gallery.com/263-ggplot2-boxplot-parameters/
https://www.youtube.com/watch?v=EtJ-iTZeqTg https://gist.github.com/expersso/944f3d4aad15f71b192fff254d4ac5b9
http://eriqande.github.io/rep-res-web/lectures/making-maps-with-R.html https://en.wikipedia.org/wiki/Violin_plot
http://bbs.pinggu.org/forum.php?mod=viewthread&tid=4182165&page=1