intro_to_ggplot_by_evelyn_zou

Title: Introduction to ggplot

1) Introdcution

- Content: An introductory instruction for students to learn about and practice using ggplot.
- Purpose: ggplot is a series of functions in R for visualizing data. This post aims to introduce the basic skills and functions of ggplot in R to students. Since data visualization is significant in many areas, students that are interested in data analysis and majoring in different fields, such as Biology and Physics (like myself), can benefit from this post and apply it to their own learnings.
- Motivation: I'm interested in presenting data in an aesthetic way. With ggplot, I can visualize data with great efficiency and diversity of methods. Also, the codes of ggplot can be applied and organized easily. Therefore, I would like to share my learning with other students interested.
- Background: I'm intended physics and stats majors. Though I have no prior experience in coding, stat133 and other online resources provide me with the opportunity to learn about presenting data with R.

2) Learn about ggplot

(a)Download R & Create Rmd File

First and foremost, to use ggplot to display the data beautifully and structurally in R, you need to download R (https://www.r-project.org/) and Rstudio (https://www.rstudio.com/) from links provided. You will use R Studio to create R Markdown for this post.

You can practice R more using online resources including "Stackoverflow R questions(www.stackoverflow.com/questions/tagged/r)", "R bloggers(www.r-bloggers.com)", etc.

After downloading softwares, you can open R Studio and click the "File" on the top left bar. Then, you can click "New File" to create and name a new "R Markdown" file, in which you make and run your own code.

For Rmd file, you have to type codes in a chunk; to make a chunk, you can copy the one below. Later on, the examples of different codes will be in the chunk as well.

(b)Intro to Function in R

In R, you write down functions as your codes. Thus, you need to type in different functions and put input inside the bracket, "()". Then, you can click the green button on the top right corner of chunk to run functions. For instance, the basic function "() <- ()" defines values. In the example below, we define x as value 3. To see the definition, you can type in x and run the code.

```
#function(input)

# (input) <- (definition)

x <- 3
x
```

```
## [1] 3
```

(c)Download Data & ggplot Functions

Now you know the basics of R. To learn about ggplot, let's take the data about NBA Player as an example, which includes players' teams, weights, etc. So the next step is to download the data. You can use the codes below to download the NBA Player data and define it as "dat". After that, you can use the green button to run these codes.

```
# Download Data
# *When you type the function, don't include the # sign*
github <- "https://github.com/ucb-stat133/stat133-fall-2017/raw/master/"
csv <- "data/nba2017-players.csv"
download.file(url = paste0(github, csv), destfile = 'nba2017-players.csv')
dat <- read.csv('nba2017-players.csv', stringsAsFactors = FALSE)
```

After downloading the data, you also need to download the ggplot functions. The function "install.packages()" allows you to install functions like ggplot2, and the function "library()" enables you to load your package:

```
# Download ggplot function
# *When you type the function, don't include the # sign*
# install.packages(c("dplyr", "ggplot2"))
library(dplyr)
```

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

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

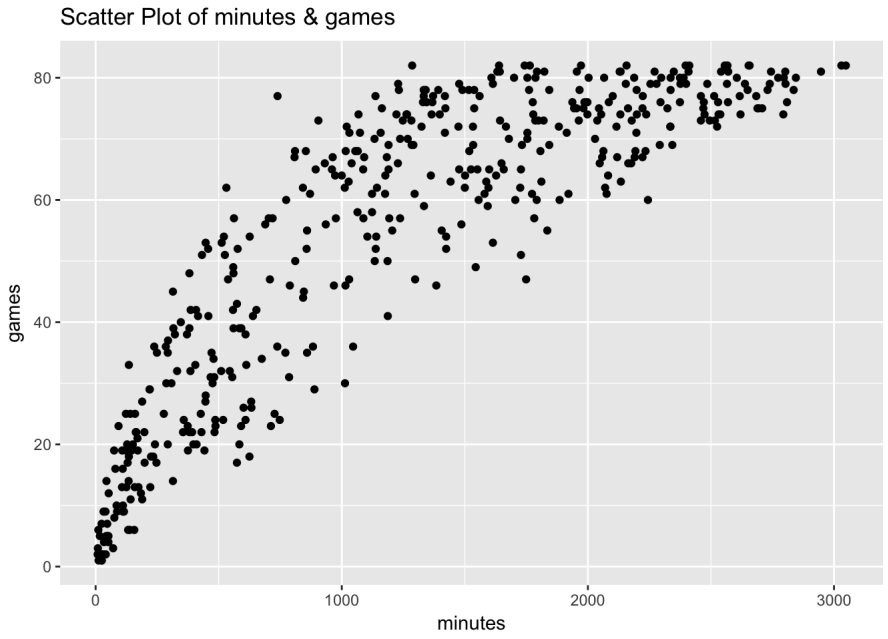
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

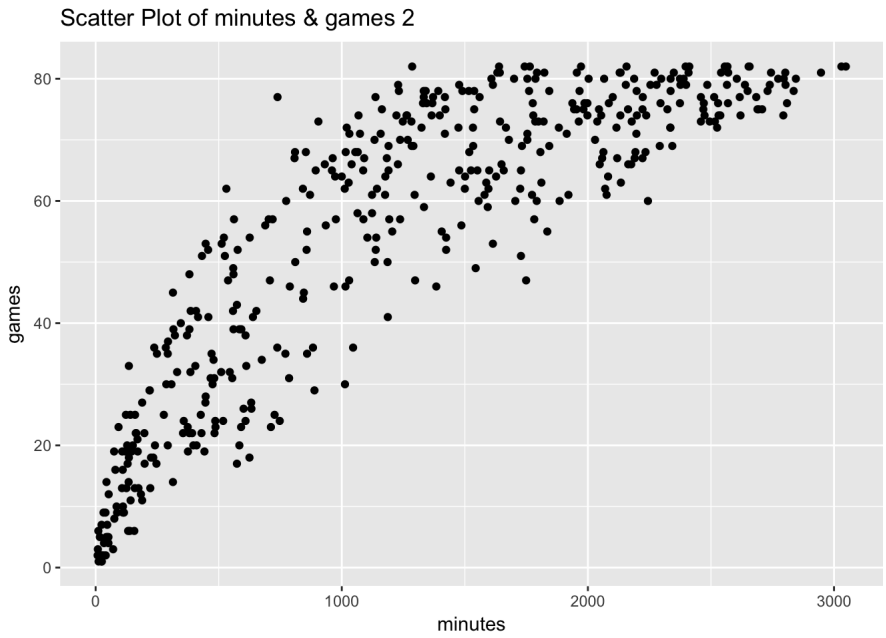
(d)Make Scatterplot with ggplot

Finishing the steps above, you are ready to make plots with functions. You can check different ggplot functions out in the cheatsheet from references. The fundamental function of ggplot is “ggplot()”, by which you can build a scatterplot. For instance, you can construct a scatterplot to demonstrate the correlation between NBA players’ minutes and number of games as below:

```
# Basic ggplot function
#ggplot(data = 'data', aes(x = 'indepedent variable', y = 'dependent variable')) +
# other ggplot functions, such as 'geom_point()', 'ggtitle()'

# Scatter Plot of minutes & games
ggplot(data = dat, aes(x = minutes, y = games)) +
  geom_point() +
  ggtitle("Scatter Plot of minutes & games")
```



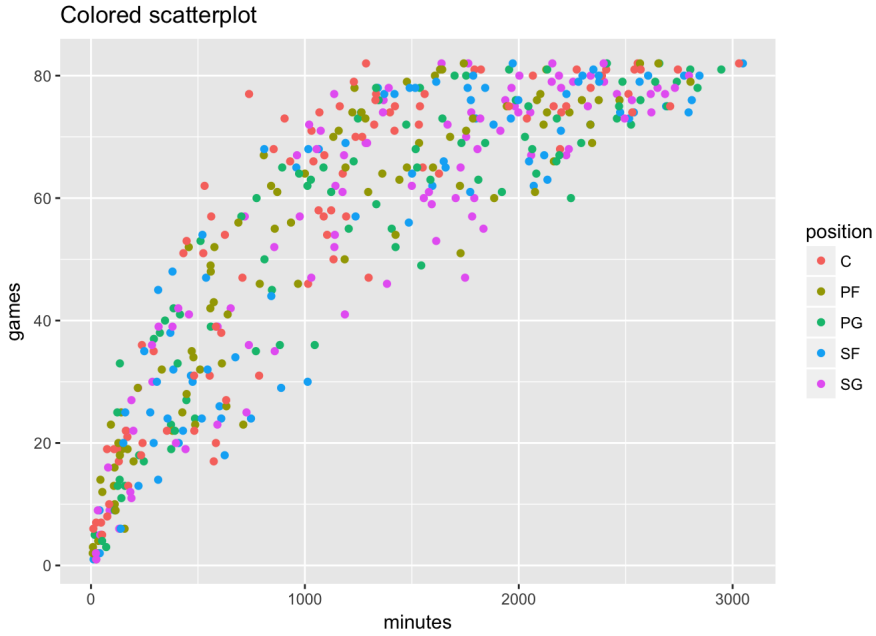

```
# or you can also code like this:
ggplot(data = dat) +
  geom_point(aes(x = minutes, y = games)) +
  ggtitle("Scatter Plot of minutes & games 2")
```



(e)Modify Scatterplot

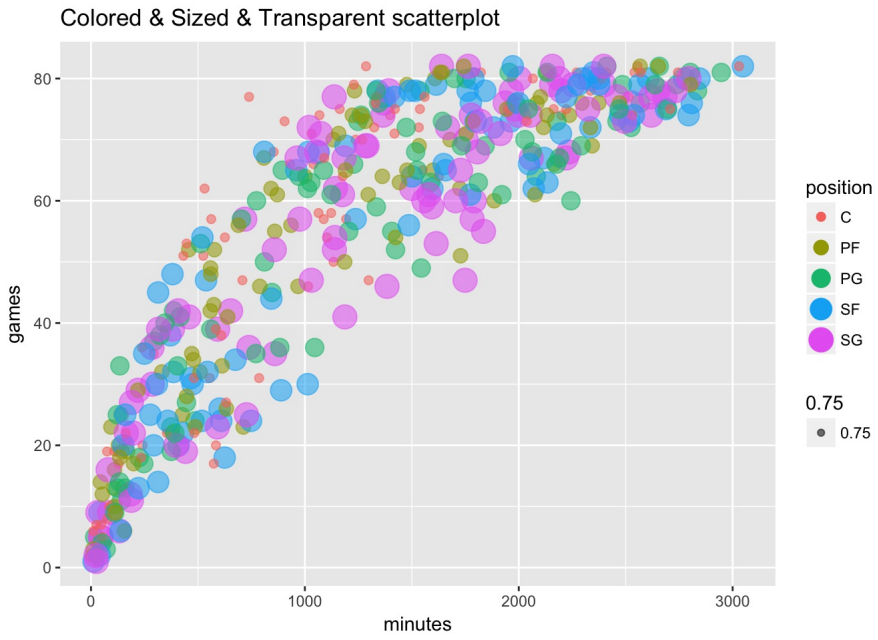
Modify the previous code, you can color the dots based on positions of players. Also, you can change their sizes and transparencies as below:

```
# ggplot function
#ggplot(data = 'data', aes(x = 'indepedent variable', y = 'dependent variable')) +
#  geom_point(aes(color = 'grouping variable'), size = 'another grouping variable', alpha = transparency value) +
#  ggtitle("title")

# Colored scatterplot based on different positions
ggplot(data = dat, aes(x = minutes, y = games)) +
  geom_point(aes(color = position)) +
  ggtitle("Colored scatterplot")
```




```
# Colored & Sized & Transparent scatterplot
ggplot(data = dat, aes(x = minutes, y = games)) +
  geom_point(aes(color = position, size = position, alpha = 0.75)) +
  ggtitle("Colored & Sized & Transparent scatterplot")
```

```
## Warning: Using size for a discrete variable is not advised.
```

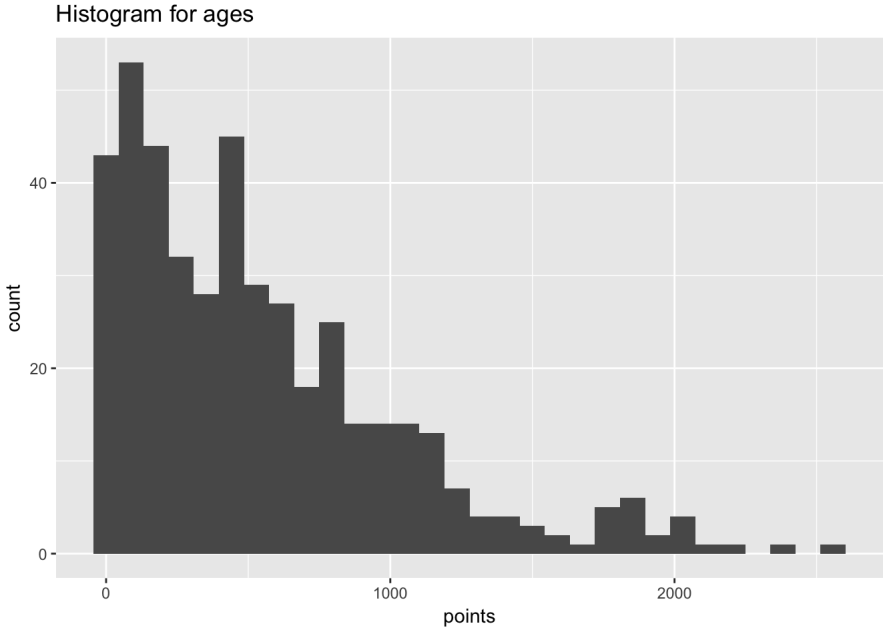


(f)Make Histogram with ggplot

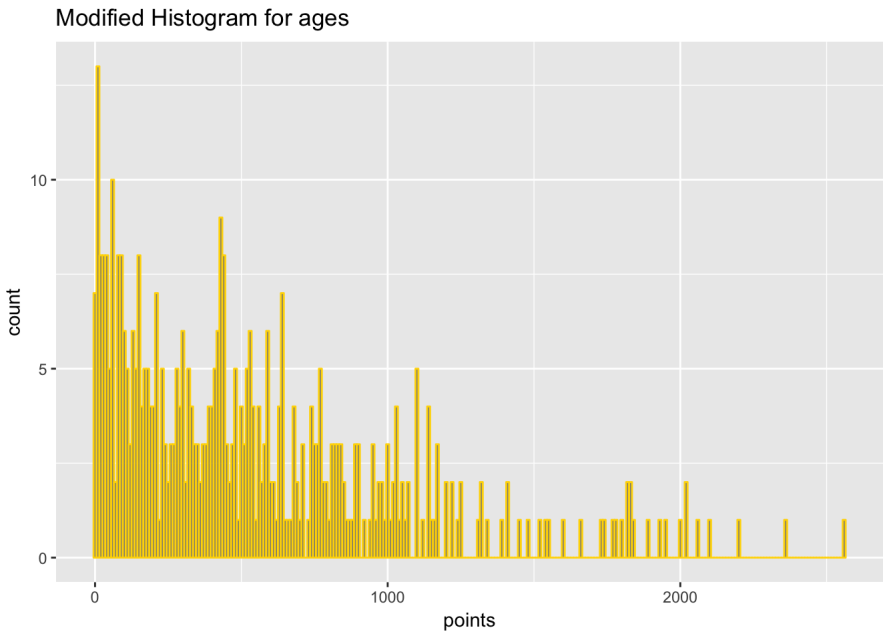
Similarly, you can create a histogram with ggplot function “geom_histogram()”, to display the frequency of ages. You can modify the color, width and transparency of the histogram as well:

```
# ggplot function for histogram
#ggplot(data = 'data', aes(x = 'variable')) +
#  geom_histogram() +
#  ggtitle("title")

# Histogram for ages
ggplot(dat, aes(x = points)) +
  geom_histogram() +
  ggtitle("Histogram for ages")
```

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



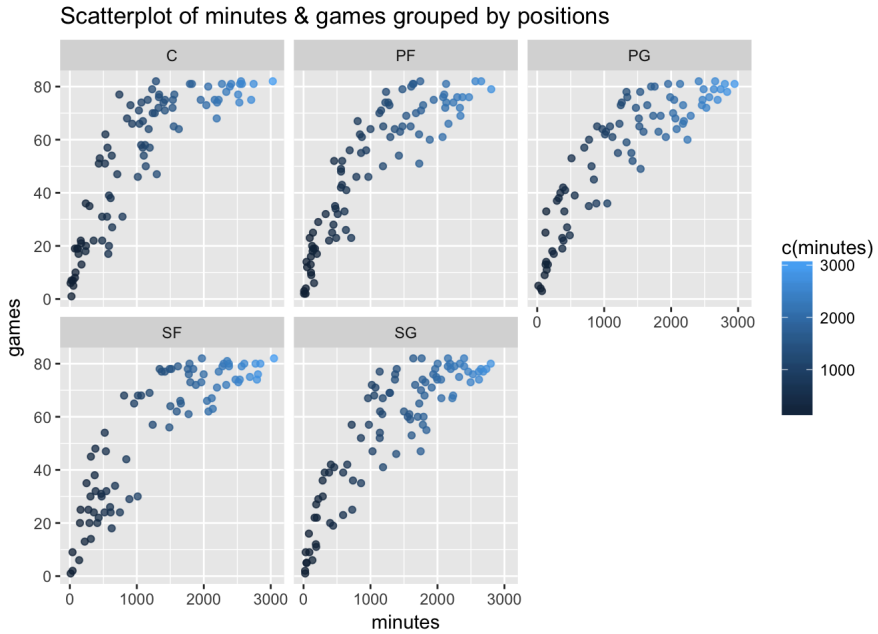

```
# Like the previous example, you can adjust factors of histogram, such as width, transparency, etc:
ggplot(dat, aes(x = points)) +
  geom_histogram(binwidth = 10, color = 'gold', alpha = 0.7) +
  ggtitle("Modified Histogram for ages")
```



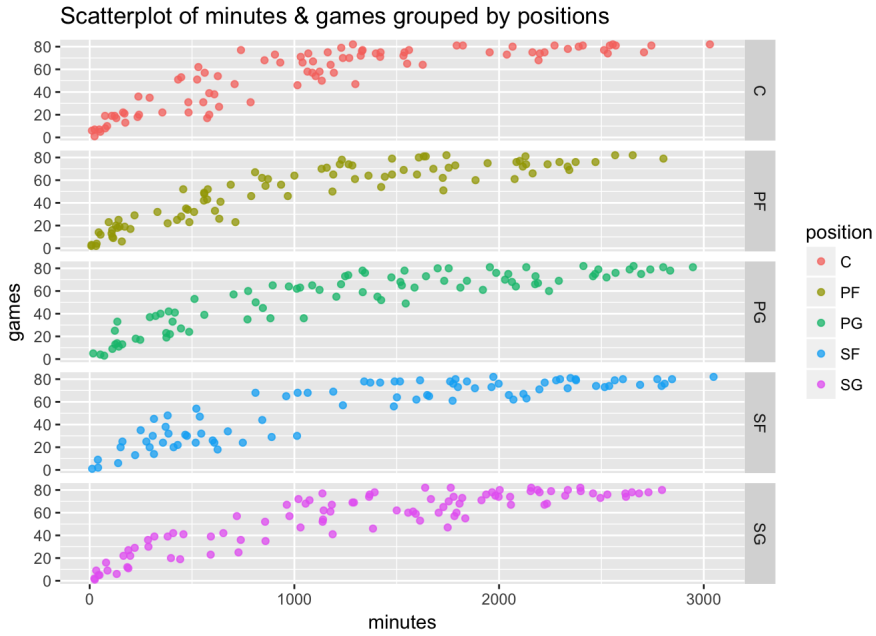
(g)Group Plot with ggplot

Moreover, by using “facet_warp()” function, you can group the scatterplots of minutes and games, such as grouping it by players’ positions. The shape of faceting can be changed by function “facet_grid()”. You can change the direction of faceting as well:

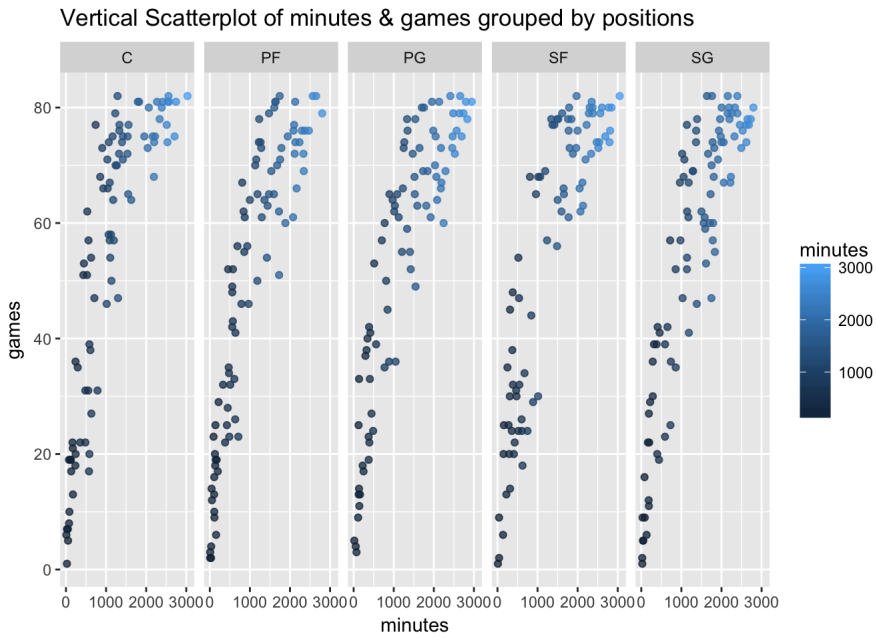
```
# faceted ggplot
#ggplot(data = 'data', aes(x = 'variable')) +
#  geom_point() +
#  faceting functions(variable ~ .) or faceting functions( ~ variable) or , such as facet_wrap(~ team)

# Scatterplot of minutes & games grouped by positions
ggplot(data = dat, aes(x = minutes, y = games)) +
  geom_point(aes(color = c(minutes)), alpha = 0.75) +
  facet_wrap(~ position) +
  ggtitle("Scatterplot of minutes & games grouped by positions")
```




```
# Horizonal Scatterplot of minutes & games grouped by positions
ggplot(data = dat, aes(x = minutes, y = games)) +
  geom_point(aes(color = position), alpha = 0.75) +
  facet_grid(position ~ .) +
  ggtitle("Scatterplot of minutes & games grouped by positions")
```



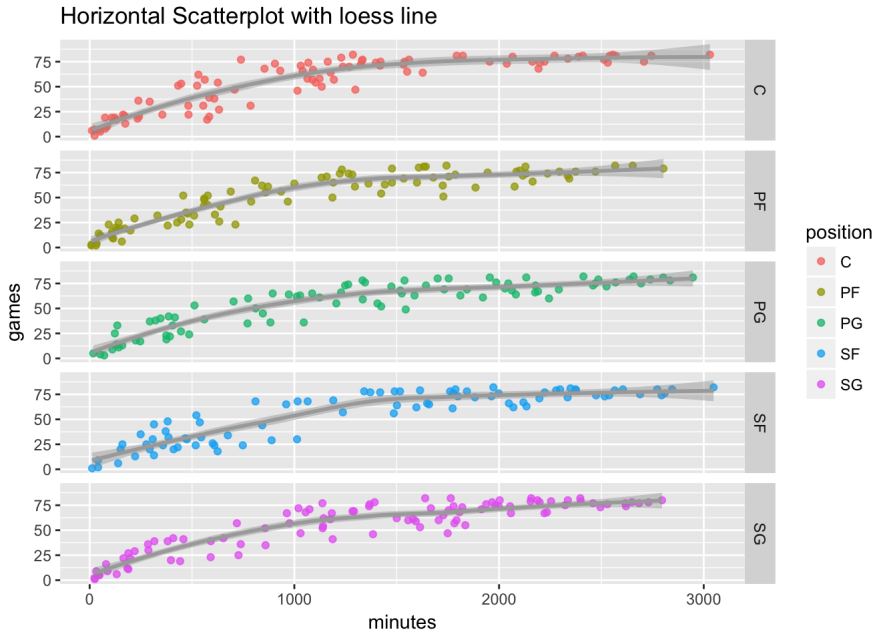

```
# To make the previous scatterplot vertical, you can put variable in faceting function to another side:
ggplot(data = dat, aes(x = minutes, y = games)) +
  geom_point(aes(color = minutes), alpha = 0.75) +
  facet_grid(~ position) +
  ggtitle("Vertical Scatterplot of minutes & games grouped by positions")
```



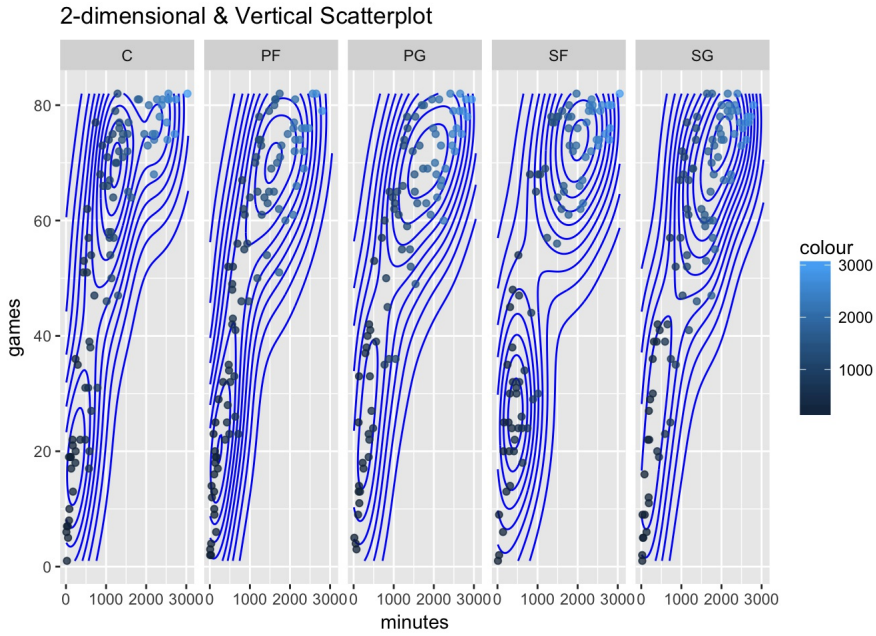
(h)Improve Plot with ggplot

To better the visualization, you can add loess line(locally weighted scatterplot smoothing) with function “geom_smooth(method = loess)”, and 2-dimensional density with function “geom_density2d()” to the plot, as below:

```
# Horizontal Scatterplot with loess line
ggplot(data = dat, aes(x = minutes, y = games)) +
  geom_point(aes(color = position), alpha = 0.75) +
  facet_grid(position ~ .) +
  geom_smooth(method = loess, color="dark grey") +
  ggtitle("Horizontal Scatterplot with loess line")
```




```
# 2-dimensional & Vertical Scatterplot
ggplot(data = dat, aes(x = minutes, y = games)) +
  geom_density2d(color = "blue") +
  geom_point(aes(color = minutes), alpha = 0.75) +
  facet_grid(~ position) +
  ggtitle("2-dimensional & Vertical Scatterplot")
```

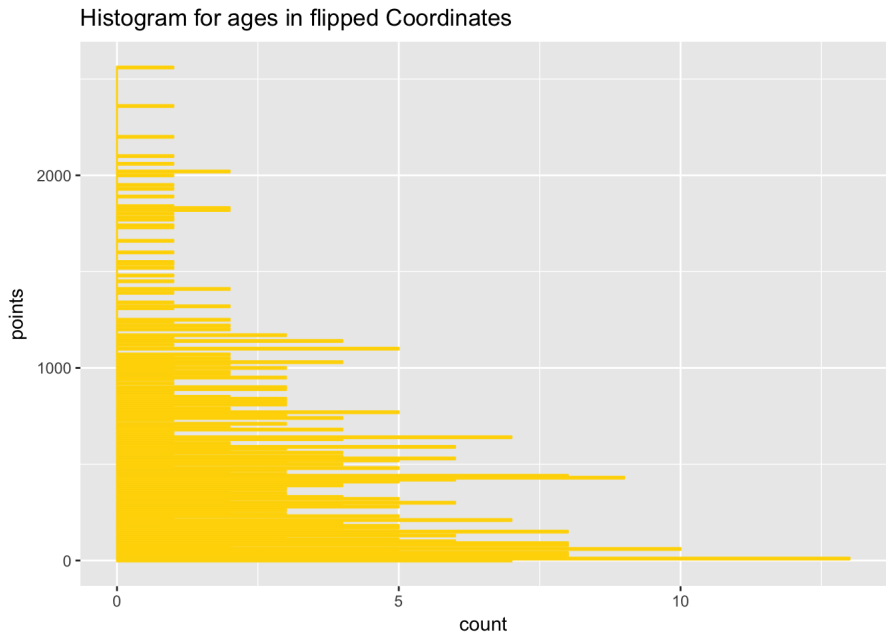


Besides the functions taught in stat133, this post will go over types of useful functions beyond lectures and discussions.

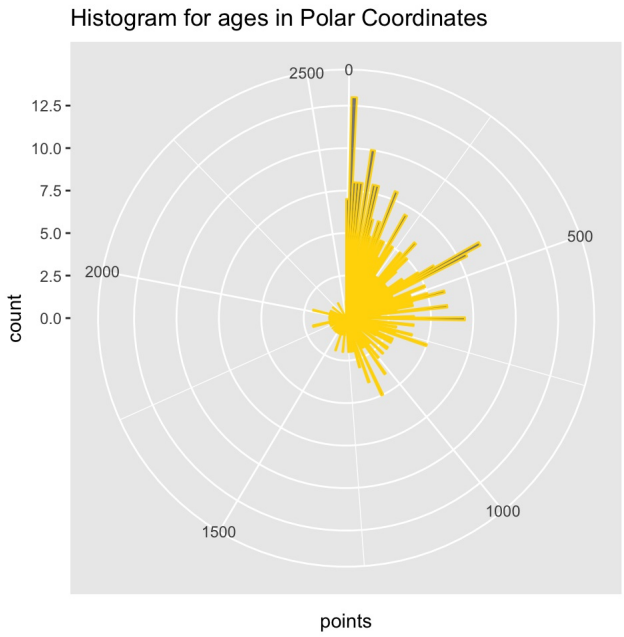
(i)Modify Plot Coordinates

For example, more than adjusting faceting, you can adjust the coordinate system of diagrams. For the previous example of histogram of ages, You can flip the coordinates with function “coord_flip()”, or apply other coordinate systems to your diagram as bellow, such as using polar coordinate with function “coord_polar()”:

```
# Histogram for ages in flipped Coordinates
ggplot(dat, aes(x = points)) +
  geom_histogram(binwidth = 10, color = 'gold', alpha = 0.7) +
  ggtitle("Histogram for ages in flipped Coordinates") +
  coord_flip()
```




```
# Histogram for ages in Polar Coordinate
ggplot(dat, aes(x = points)) +
  geom_histogram(binwidth = 10, color = 'gold', alpha = 0.7) +
  ggtitle("Histogram for ages in Polar Coordinates") +
  coord_polar(theta = "x", direction = 1)
```



(j)Make Violin Plot & Dotplot & Other Plots

ggplot comes in handy when presenting the correlation between one continuous variable and one discrete variable. Following the cheatsheet from references, you can create and adjust a violin plot/dotplot of players' positions and ages as below:

```
# Violin Plot of position & age
ggplot(data = dat, aes(x = position, y = age)) +
  geom_violin(scale="area", fill="orange", color="yellow") +
  ggtitle("Violin Plot of position & age")
```


```
# Dotplot of position & age
ggplot(data = dat, aes(x = position, y = age)) +
  geom_dotplot(binaxis="y",stackdir="center", binwidth=0.5, fill="green", color="dark green") +
  ggtitle("Dotplot of position & age")
```

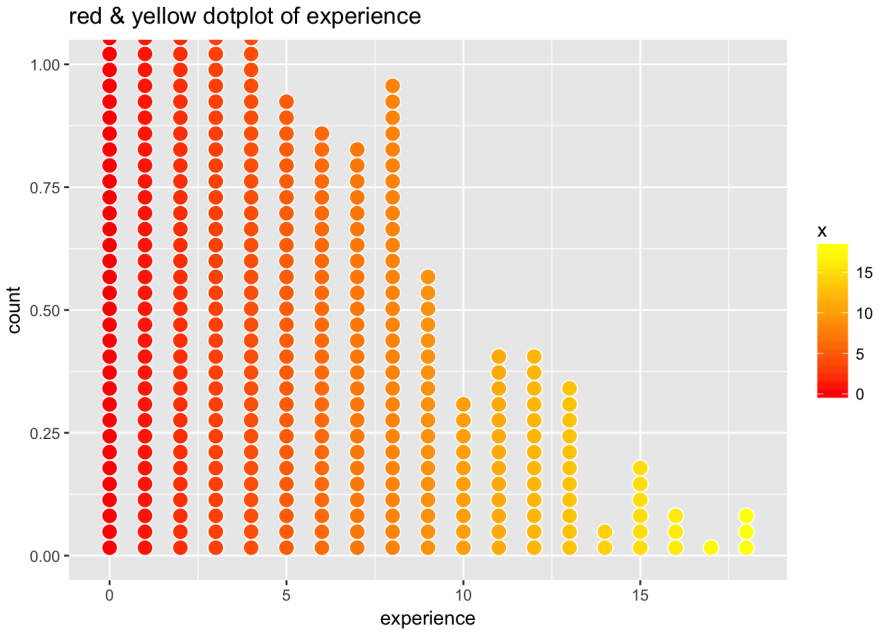
Likewise, you can make the plot of two discrete variables, such as the counting plot of position and age:

```
# counting plot of position & age
ggplot(data = dat, aes(x = position, y = age)) +
  geom_count(aes(color=age)) +
  ggtitle("counting plot of position & age")
```

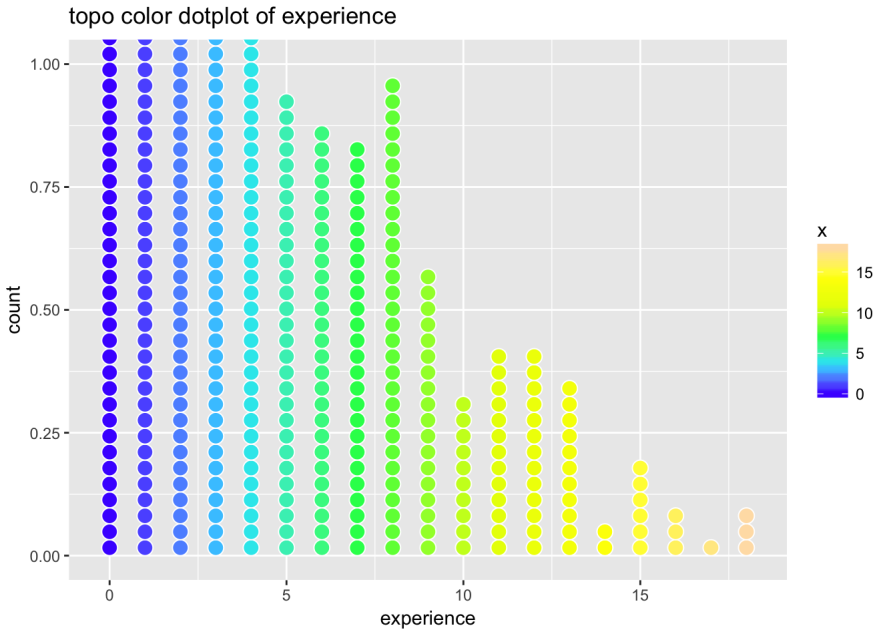
(k)Color the Plot

In addition to coloring with respect to another variable, you can color the graph with different methods. For example, when making dotplot of a single continuous variable, you can add gradient colors to it by using scale functions like “scale_fill_gradient()” and “scale_fill_gradientn()” as bellow:

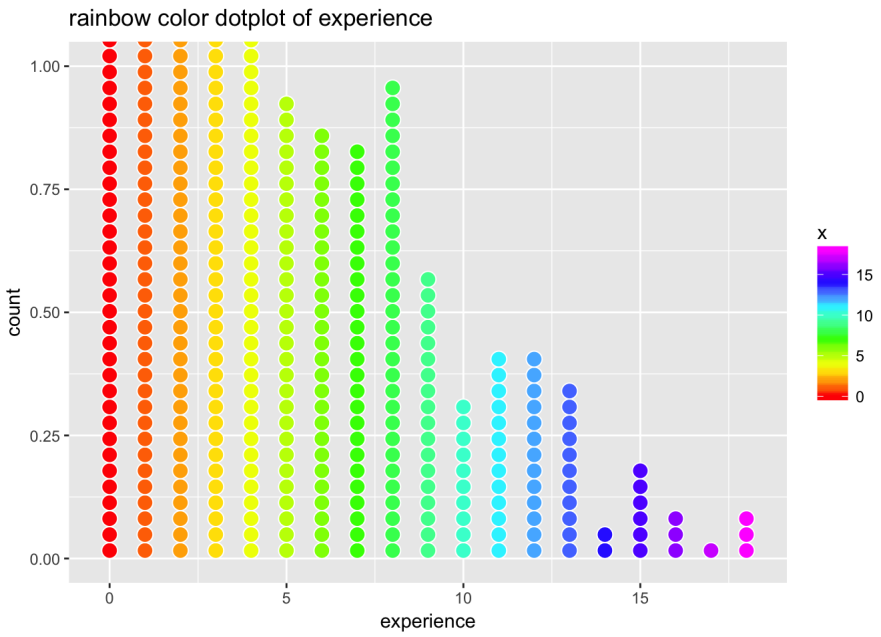
```
# Using function "scale_fill_gradient()", you can input the low and high colors.

# gradient red & yellow color dotplot of experience
ggplot(dat, aes(x = experience)) +
  geom_dotplot(binwidth=0.45, color="white", aes(fill=..x..)) +
  scale_fill_gradient(low="red",high="yellow") +
  ggtitle("red & yellow dotplot of experience")
```




```
# When using function "scale_fill_gradientn()", you can input the set of colors, such as "rainbow()" and "topo.col
ors()".

# topo color dotplot of experience
ggplot(dat, aes(x = experience)) +
  geom_dotplot(binwidth=0.45, color="white", aes(fill=..x..)) +
  scale_fill_gradientn(colours=topo.colors(6)) +
  ggtitle("topo color dotplot of experience")
```



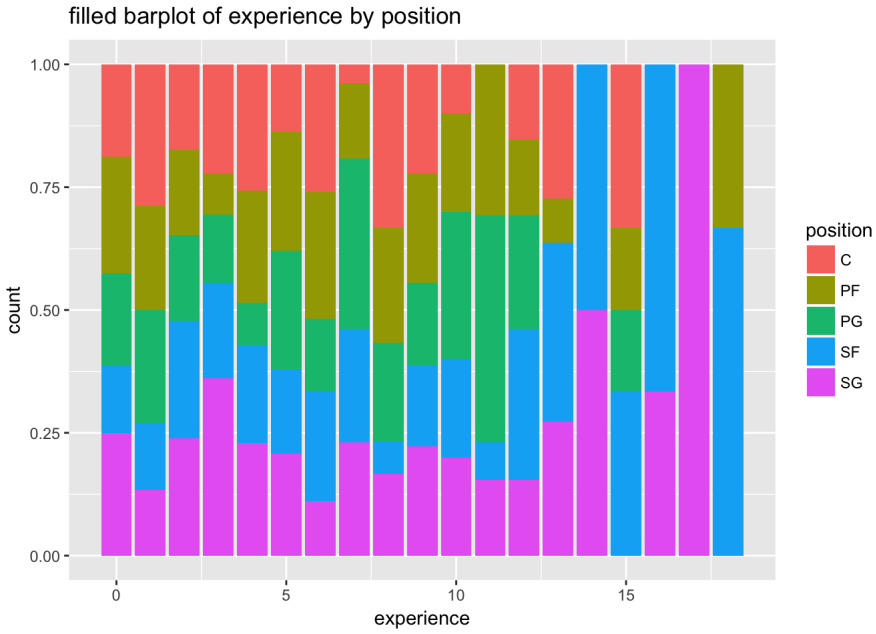

```
# rainbow color dotplot of experience
ggplot(dat, aes(x = experience)) +
  geom_dotplot(binwidth=0.45, color="white", aes(fill=..x..)) +
  scale_fill_gradientn(colours=rainbow(6)) +
  ggtitle("rainbow color dotplot of experience")
```



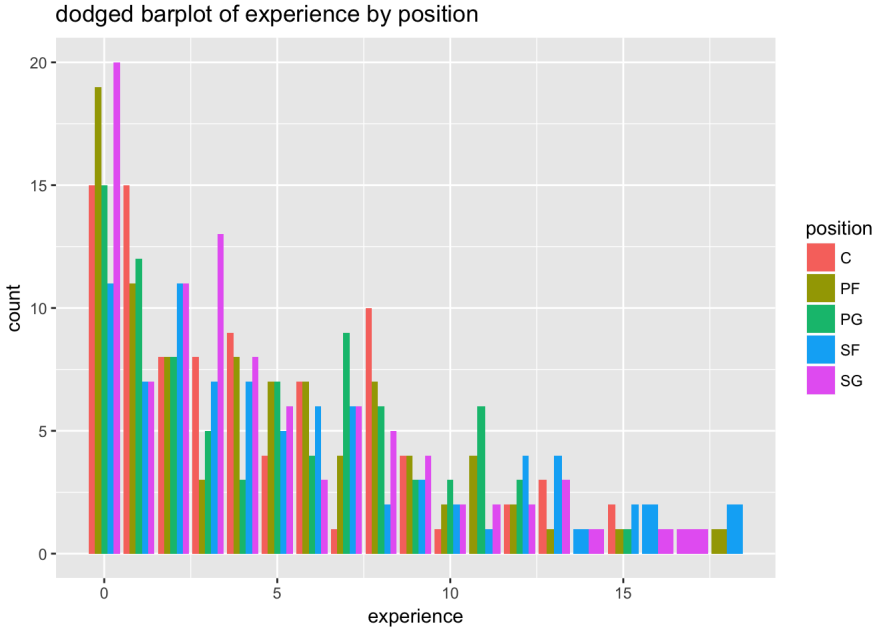
(l)Position Adjustment in Plot

Another useful function for ggplot is position adjustment. When you are working on barplot for one discrete variable and one continuous variable, you can modify the bars by changing the position input in function “geom_bar(position=)”:

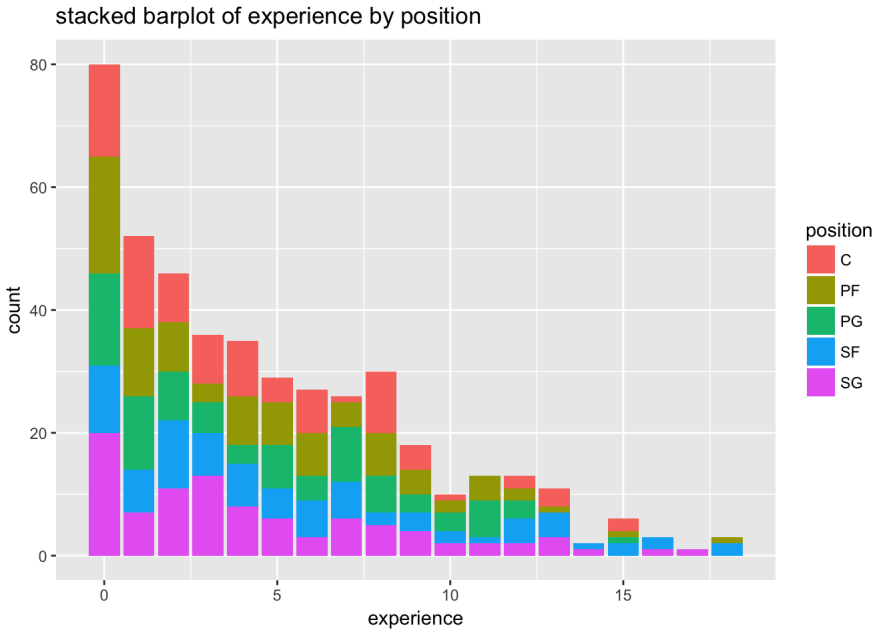
```
# barplot with position adjustment
#ggplot(dat, aes(x = 'independent variable', fill = position)) +
#  geom_bar(position=" ") + ggtitle("title")

#  filled barplot of experience by position
ggplot(dat, aes(x = experience, fill = position)) +
  geom_bar(position="fill") + ggtitle("filled barplot of experience by position")
```




```
# dodged barplot of experience by position
ggplot(dat, aes(x = experience, fill = position)) +
  geom_bar(position="dodge") + ggtitle("dodged barplot of experience by position")
```



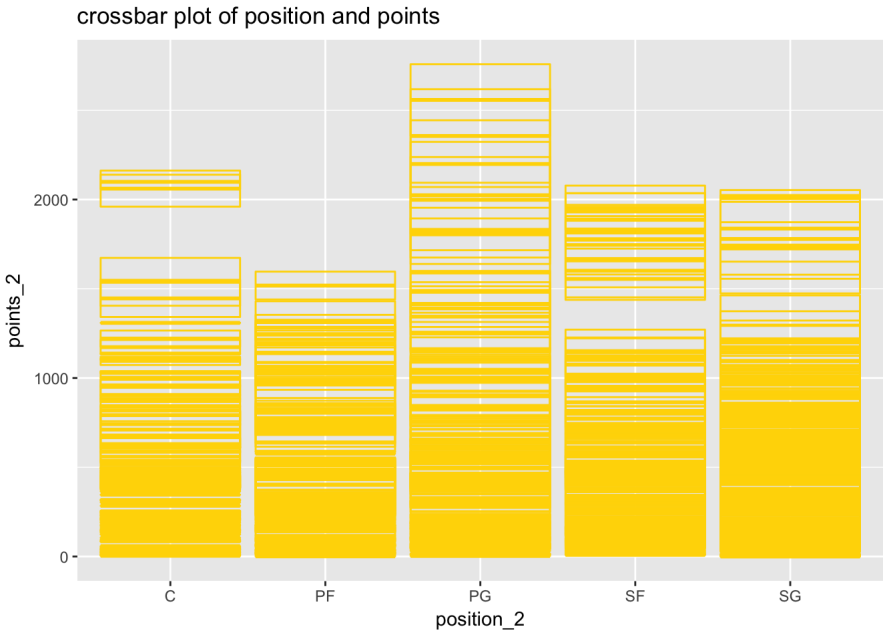

```
# stacked barplot of experience by position
ggplot(dat, aes(x = experience, fill = position)) +
  geom_bar(position="stack") + ggtitle("stacked barplot of experience by position")
```



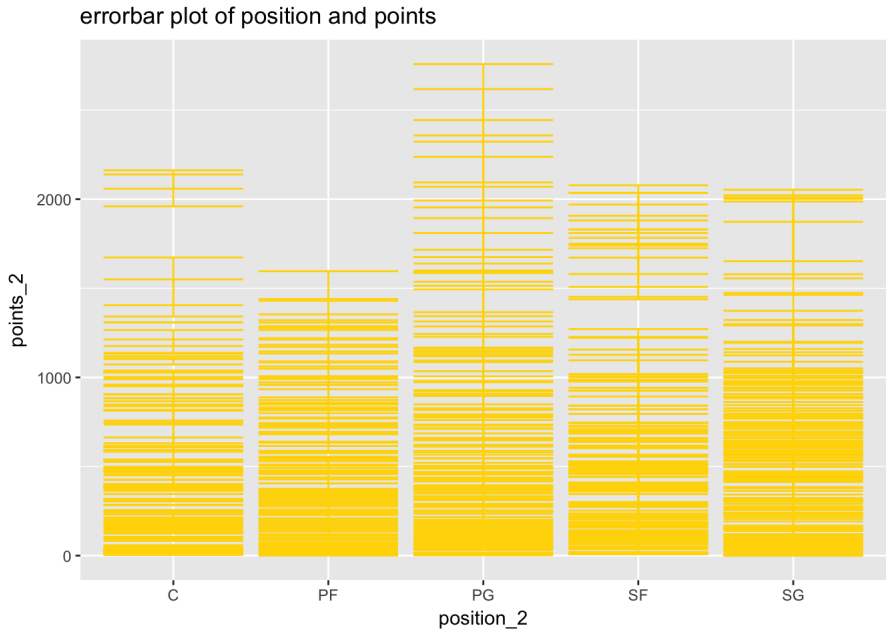
(m)Visualize Error in Plot

Furthermore, you can even visualize the error of data with ggplot. Assume that you're plotting the correlation between NBA players' points and positions. You can estimate the error in players' points by points3, which assumes that for each player, the maximum and minimum of points is off by player's points3. Then you can use crossbar/errorbar/pointrange to plot it. So first, you need to create another data set with function "data.frame()", with respect to players' positions, points, and points3:

```
# create another data frame as data
dat2 <- data.frame(
  position_2 = dat$position,
  points_2 = dat$points,
  points3_2 = dat$points3
)

# Then you can use ggplot functions to present your data as below:
# crossbar plot of position and points
ggplot(data = dat2, aes(position_2 , points_2, ymin = points_2 - points3_2, ymax = points_2 + points3_2)) +
  geom_crossbar(fatten = 2, color="gold") + ggtitle("crossbar plot of position and points")
```




```
# errorbar plot of position and points
ggplot(data = dat2, aes(position_2 , points_2, ymin = points_2 - points3_2, ymax = points_2 + points3_2)) +
  geom_errorbar(color="gold") + ggtitle("errorbar plot of position and points")
```

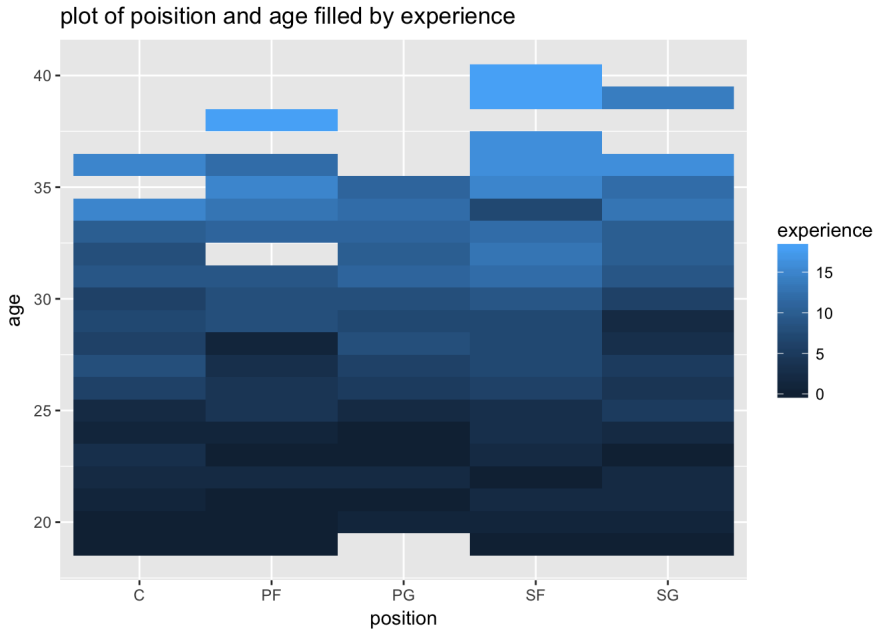



```
# pointrange plot of position and points
ggplot(data = dat2, aes(position_2 , points_2, ymin = points_2 - points3_2, ymax = points_2 + points3_2)) +
  geom_pointrange(color="gold") + ggtitle("pointrange plot of position and points")
```

(n)Three-variable ggplot

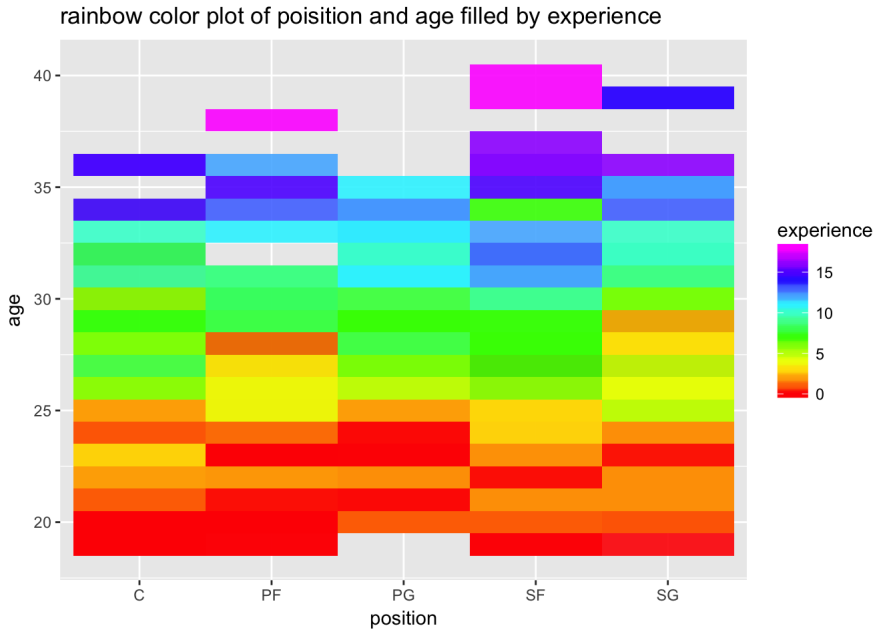
Last but not least, other than two variables, plots of three variable can be made with ggplot functions “geom_raster()”. For instance, you can create a plot of players’ positions, ages, and experience as below:

```
# ggplot of three variables
#ggplot(data = dat, aes(x = 'variable1', y = 'variable2', fill = 'variable3')) +
#  geom_raster() + ggtitle("title")

# plot of poisition and age filled by experience
ggplot(data = dat, aes(x = position, y = age, fill = experience)) +
  geom_raster() + ggtitle("plot of poisition and age filled by experience")
```




```
# Change the function "geom_raster()" to "geom_tile()", you can adjust its transparency, or color it with scale fu
nctions mentioned before:
ggplot(data = dat, aes(x = position, y = age, fill = experience)) +
  geom_tile(alpha=0.9) +
  scale_fill_gradientn(colours=topo.colors(6)) +
  ggtitle("modified plot of poisition and age filled by experience")
```


```
# rainbow color plot of poisition and age filled by experience
ggplot(data = dat, aes(x = position, y = age, fill = experience)) +
  geom_tile(alpha=0.9) +
  scale_fill_gradientn(colours=rainbow(6)) +
  ggtitle("rainbow color plot of poisition and age filled by experience")
```



3) Discussion:

Besides those mentioned above, there're other methods to make and improve the ggplot in data analysis. Therefore, there's more for you to explore, and you can start by trying answering the questions below: * How to use scale function to change the dot shape in dotplot? * How to create a three-variable plot with function "geom_contour()"? * Are the colors really necessary, or are they just confusing? What can be deleted? * Can variables in the plot be clarified by labeling the axes? …

4) Conclusion:

To summarize, I hope this post could bring you a basic understanding of ggplot, a very handy tool in R that allows you to present data clearly and beautifully. You can learn more about ggplot by using the references below, and other resources provided in this post. :) Have fun!

5) References:

- "R for Biologists: Several plot types in just a few minutes" by Maria Nattestad: https://www.youtube.com/watch?v=pibGllDeBPM
- "Introduction to Data Visualization with R and ggplot2" by Data Science Dojo: https://www.youtube.com/watch?v=49fADBfcDD4
- "Plotting in R tutorial: Gorgeous graphs with ggplot2" by deltaDNA: https://www.youtube.com/watch?v=rsG-GgR0aEY
- ggplot cheatsheat: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
- "ggplot with three y-variables R": https://stackoverflow.com/questions/41172451/ggplot-with-three-y-variables-r
- "Summarizing 3 categorical variables using R (and ggplot)": https://www.youtube.com/watch?v=TJKskHT_zQs
- "ggplot2 tutorial: Multiple Groups and Variables": https://www.youtube.com/watch?v=-jO2wuu7GUQ