more_about_ggplot_by_evelyn_zou

Title: More about ggplot Function

1) Introdcution

- Content: An instruction for students to learn about and practice using basic and advanced ggplot functions.
- Purpose: ggplot is a broad range of functions in R for visualizing data. Besides introducing the basic ggplot in R, this post aims to expose students to more advanced ggplot functions. Due to the significance of data visualization in many fields, students majoring in varied areas, such as Biology and Psychology, can learn from this post and apply ggplot to their own learnings.
- Motivation: After seeing the use of data analysis in different classes, I realize the importance of visualizing and presenting data. I'm also interested in presenting data in an aesthetic and concise way, which can be done with the help of ggplot. Therefore, I would like to share my learning with other students interested.
- Background: I plan to major in 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 5. To see the definition, you can type in x and run the code.

```
#function(input)

# (input) <- (definition)

x <- 5
x
```

```
## [1] 5
```

(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
```

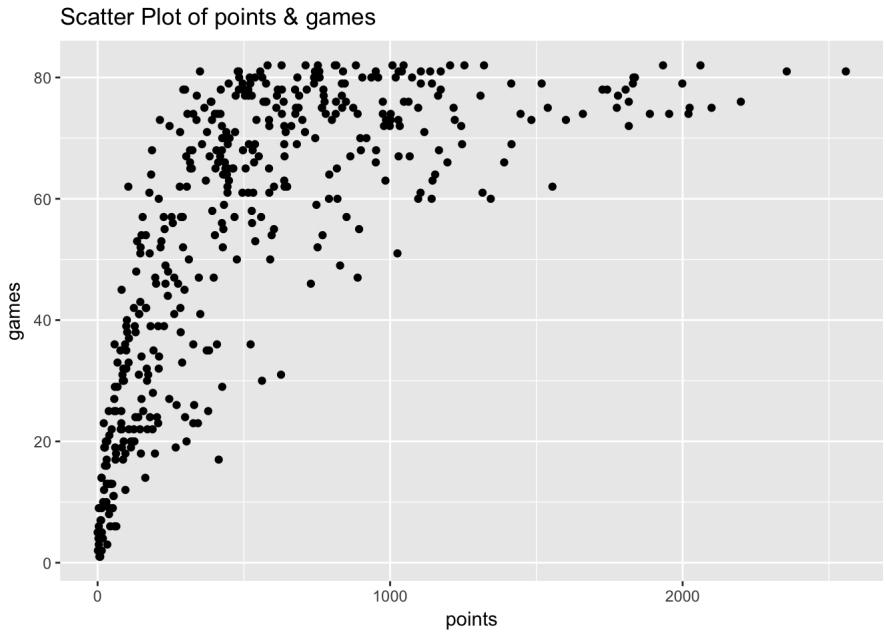
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

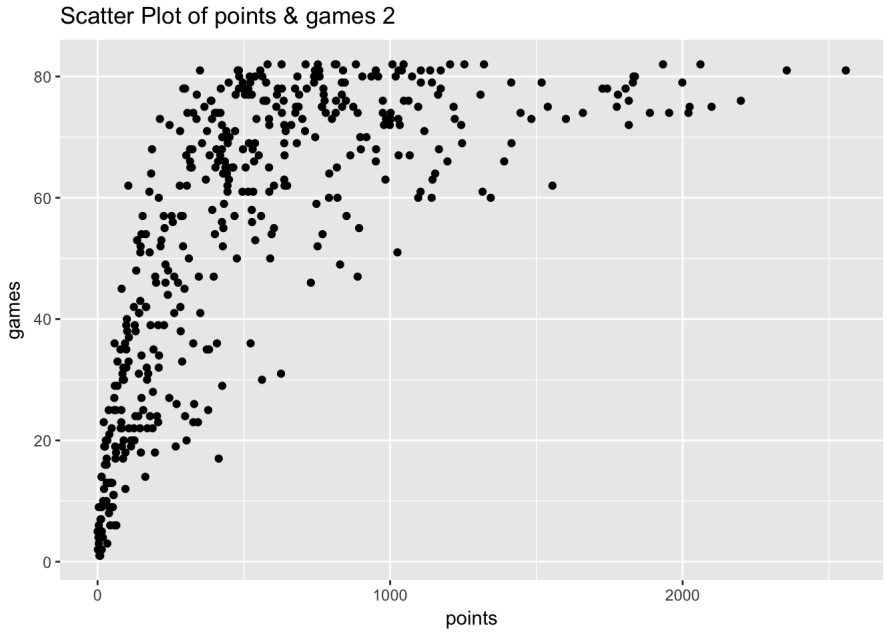
(d)Make Basic Scatterplot with ggplot

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’ points 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 = points, y = games)) +
  geom_point() +
  ggtitle("Scatter Plot of points & games")
```



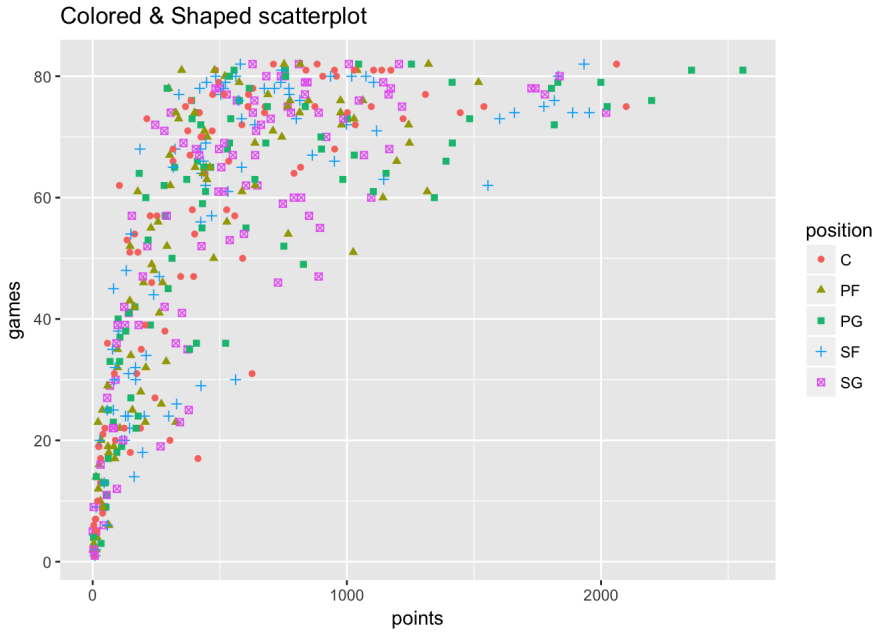

```
# or you can also code like this:
ggplot(data = dat) +
  geom_point(aes(x = points, y = games)) +
  ggtitle("Scatter Plot of points & games 2")
```



(e)Modify Scatterplot

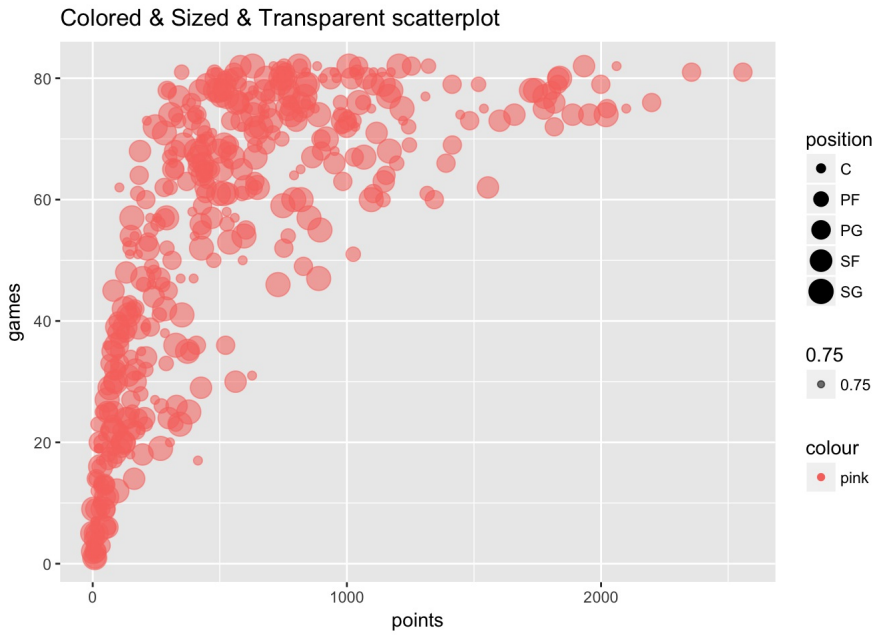
By modifying the previous code, you can change the attributes of the plot. For example, you can color and shape the dots based on players' different positions. Also, you can change their sizes and transparencies as below:

```
# 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 & Shaped scatterplot
ggplot(data = dat, aes(x = points, y = games)) +
  geom_point(aes(color = position, shape = position)) +
  ggtitle("Colored & Shaped scatterplot")
```




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

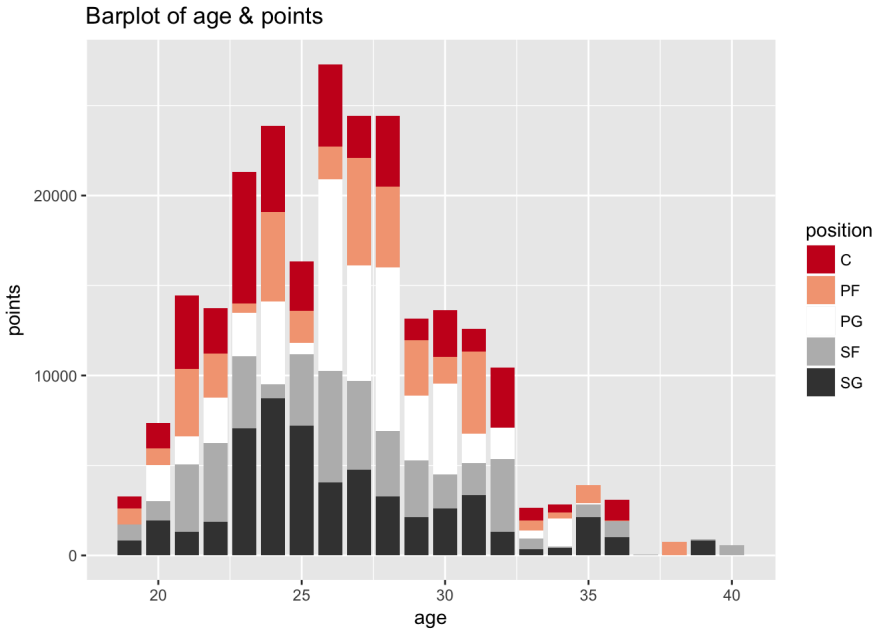
```
## Warning: Using size for a discrete variable is not advised.
```



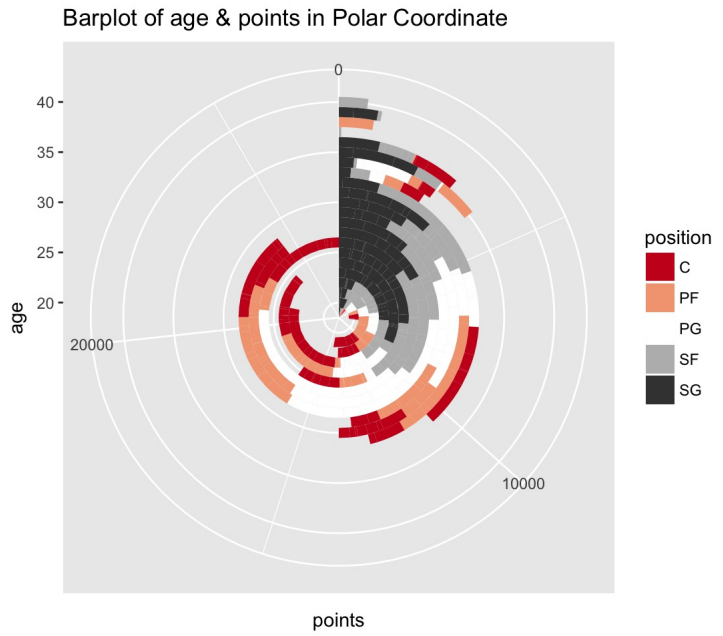
(h)Barplot & Polar Coordinate

The most common plot in R is the barplot. You can make a barplot between NBA players' age and points with function "geom_bar", and edit its color with "scale_fill_brewer()" as below:

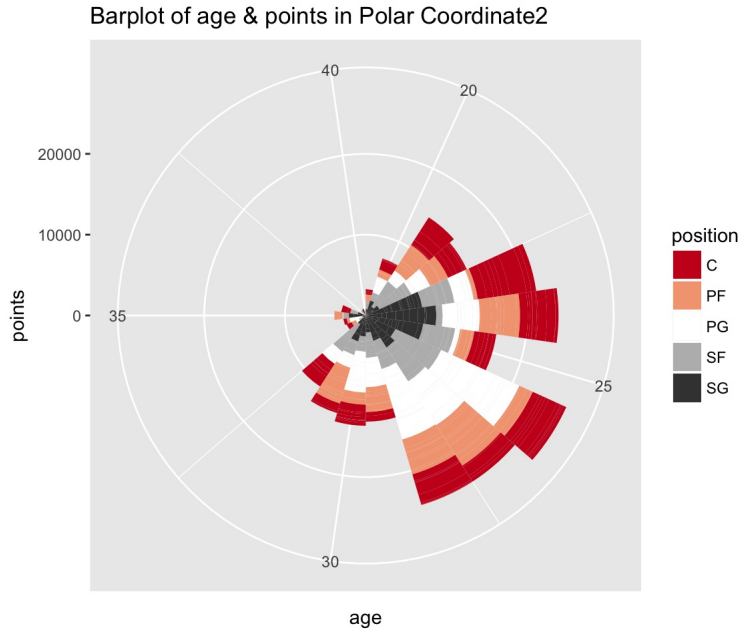
```
# Similar to the previous example, you can modify the attribute of plot by modifying the function.

# Barplot of age & points
ggplot(dat, aes(x = age, y = points, fill = position)) +
    geom_bar(width = 0.85, stat="identity") +
    scale_fill_brewer(palette = "RdGy") +
    ggtitle("Barplot of age & points")
```




```
# Using the function "coord_polar()", you can convert it into polar coordinate:
ggplot(dat, aes(x = age, y = points, fill = position)) +
    geom_bar(width = 1, stat="identity") +
    coord_polar(theta = "y") +
    scale_fill_brewer(palette = "RdGy") +
    ggtitle("Barplot of age & points in Polar Coordinate")
```




```
# You can also change the parameter of polar coordinate:
ggplot(dat, aes(x = age, y = points, fill = position)) +
    geom_bar(width = 1, stat="identity") +
    coord_polar(theta = "x") +
    scale_fill_brewer(palette = "RdGy") +
    ggtitle("Barplot of age & points in Polar Coordinate2")
```

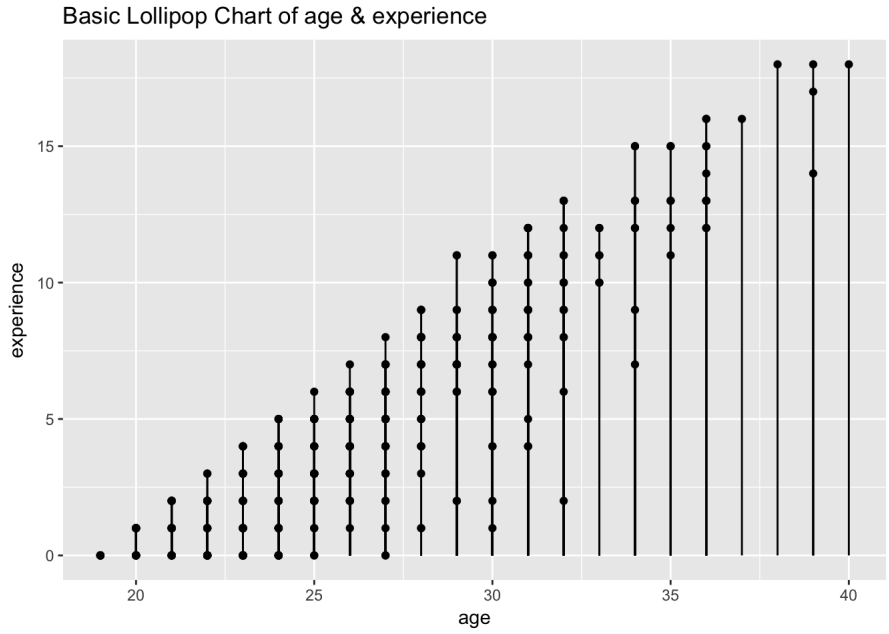


(f)Lollipop Chart

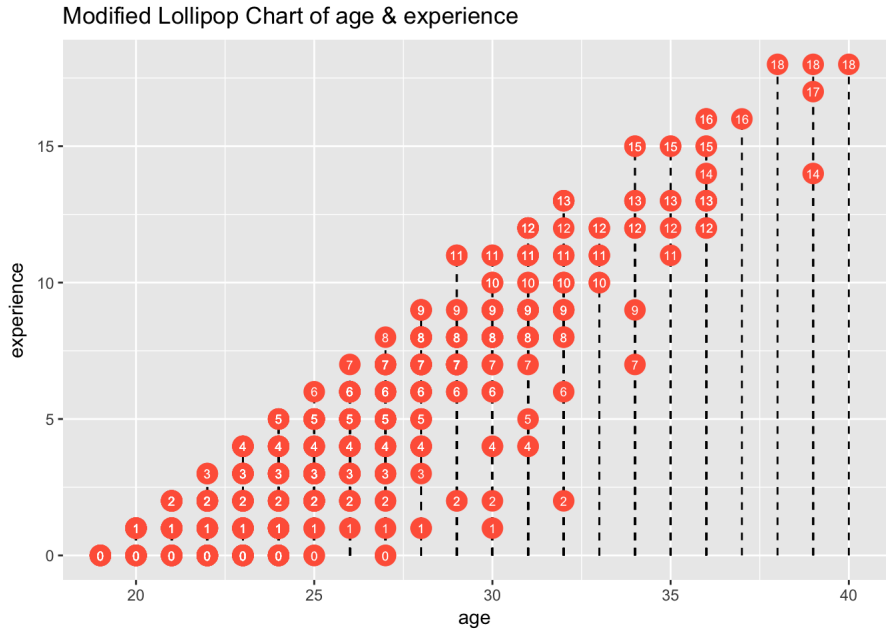
Now you've learnt how to use and modify basic ggplot function, let's move to more advanced ggplot function.

When visualizing the correlation between two continuous variables, Lollipop Chart is a good option. It not only presents the correlation clearly, but also groups the data with lollipop-shaped bar. For example, you can create a Lollipop Chart between players' age and experience with function "geom_segment()" as below:

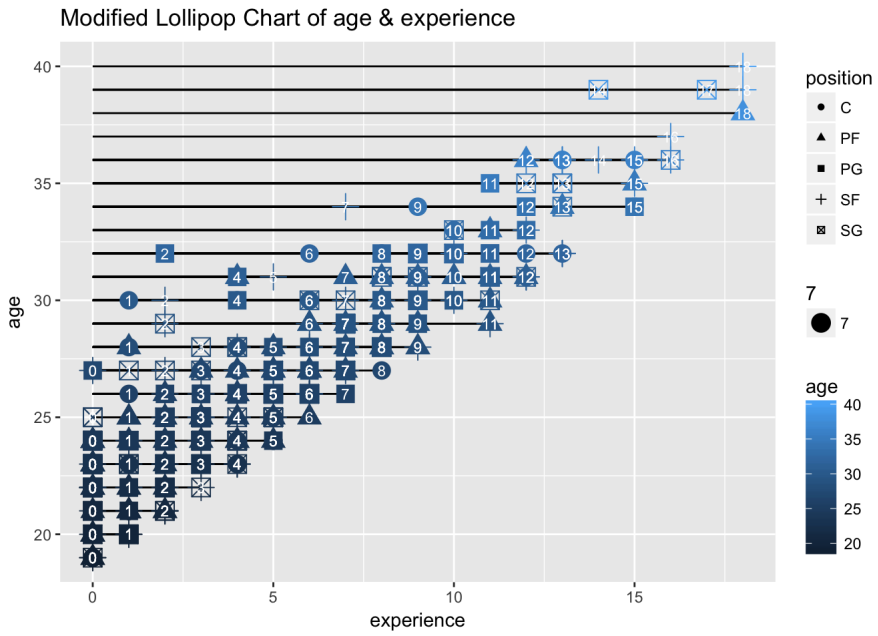
```
# Basic Lollipop Chart of age & experience
ggplot(dat, aes(x=age, y=experience)) +
        geom_segment(aes(y = 0, x = age, yend = experience, xend = age)) +
        geom_point() +
        ggtitle("Basic Lollipop Chart of age & experience")
```




```
# To better the visualization, you can enlarge the dots and add text on them:
ggplot(dat, aes(x=age, y=experience, label=experience)) +
  geom_segment(aes(y = 0, x = age, yend = experience, xend = age), linetype="dashed") +
  geom_point(stat='identity', size = 5, color = "tomato")  + # enlarge dots
  geom_text(color="white", size=2.5) + # add text
  ggtitle("Modified Lollipop Chart of age & experience")
```



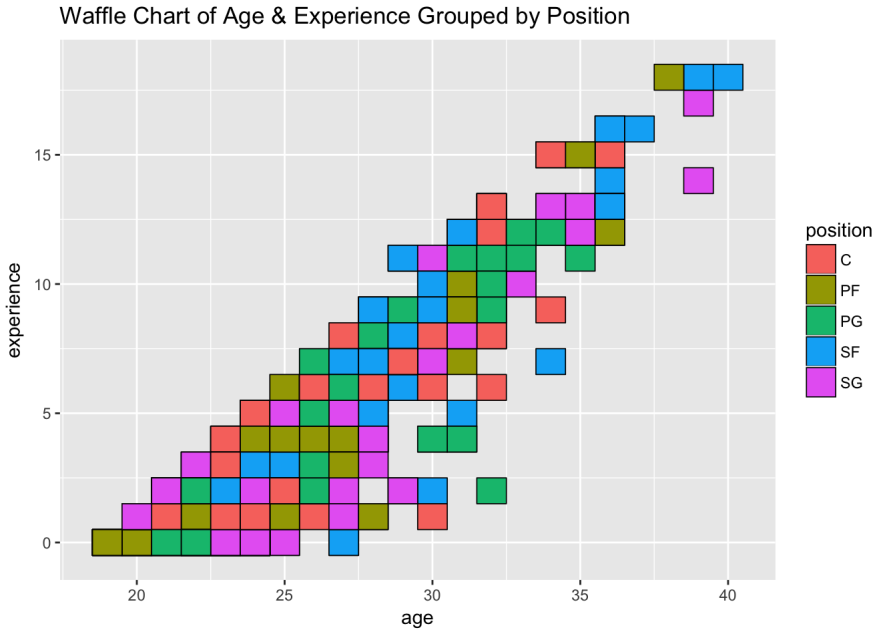

```
# You can also color and shape the dots, and flip the coordinate with "coord_flip()" :
ggplot(dat, aes(x=age, y=experience, label=experience)) +
  geom_segment(aes(y = 0, x = age, yend = experience, xend = age)) +
  geom_point(stat='identity', aes(size=7, color=age, shape = position))  +
  geom_text(color="white", size=3) +
  ggtitle("Modified Lollipop Chart of age & experience") +
  coord_flip()
```



(g)Waffle Chart

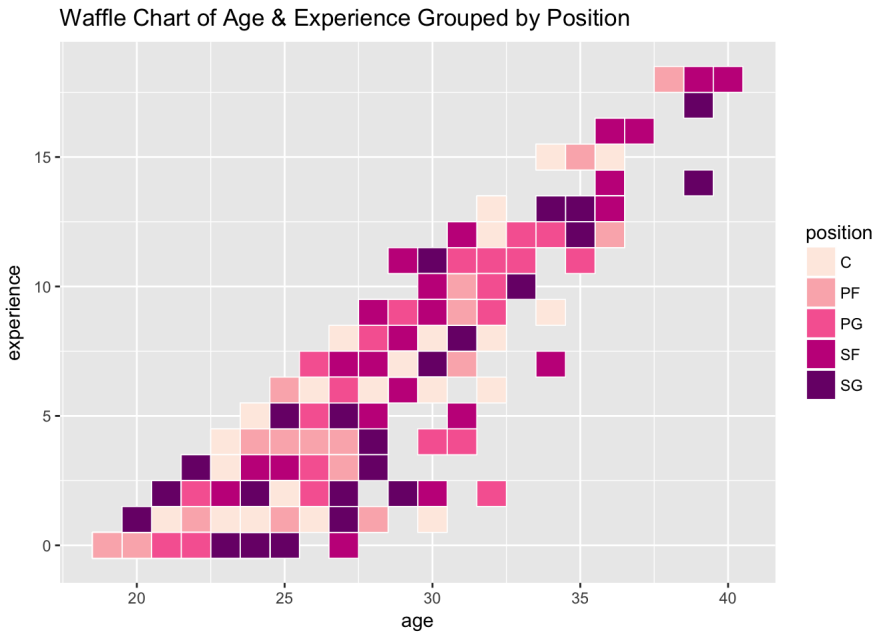
When dealing with two continuous variables, you can likewise apply waffle chart to visualize and group them. For the previous example, you can also use function “geom_tile()” to group them based on position as below:

```
# Waffle Chart of Age & Experience Grouped by Position
ggplot(dat, aes(x = age, y = experience, fill = position)) +
        geom_tile(color = "black", size = 0.3) +
        labs(title="Waffle Chart of Age & Experience Grouped by Position")
```




```
# You can change the color to B&W graident with function "scale_fill_grey()" :
ggplot(dat, aes(x = age, y = experience, fill = position)) +
        geom_tile(color = "black", size = 0.3) +
        scale_fill_grey(start = 0.1, end = 0.9) +
        labs(title="Waffle Chart of Age & Experience Grouped by Position")
```


```
# Similarly, you can change the graident into the color set in R with "scale_fill_brewer()" :
ggplot(dat, aes(x = age, y = experience, fill = position)) +
        geom_tile(color = "white", size = 0.3) +
        scale_fill_brewer(palette = "RdPu") +
        labs(title="Waffle Chart of Age & Experience Grouped by Position")
```




```
# To see all the color sets in R, you can type "RColorBrewer::display.brewer.all()" in Console.
```

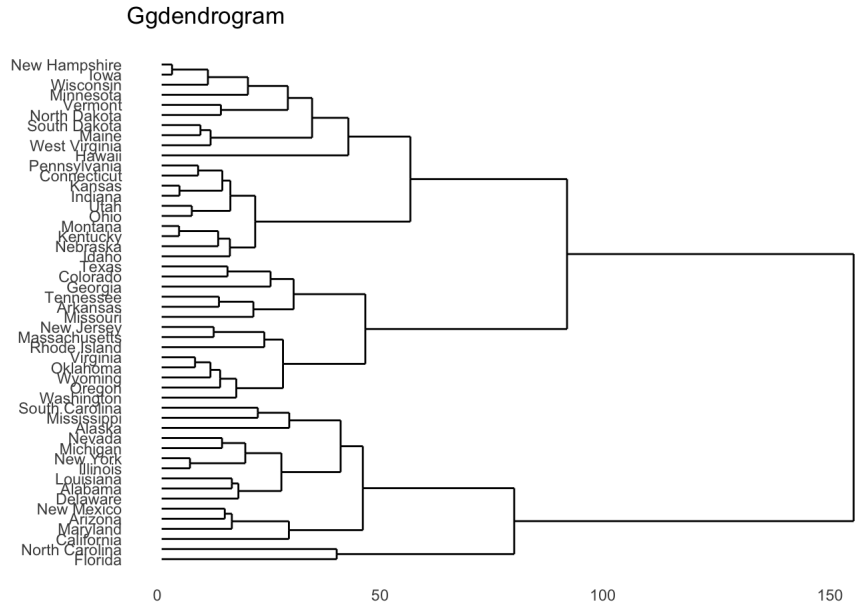
(i)Hierarchical Dendrogram

For the next part, we will be using the data of “USArrests”, a dataset in R that includes types of crimes in different states. To present the correlation between clusters of crimes and states in US, you can use the Hierarchical Dendrogram with “ggdendrogram()” as below:

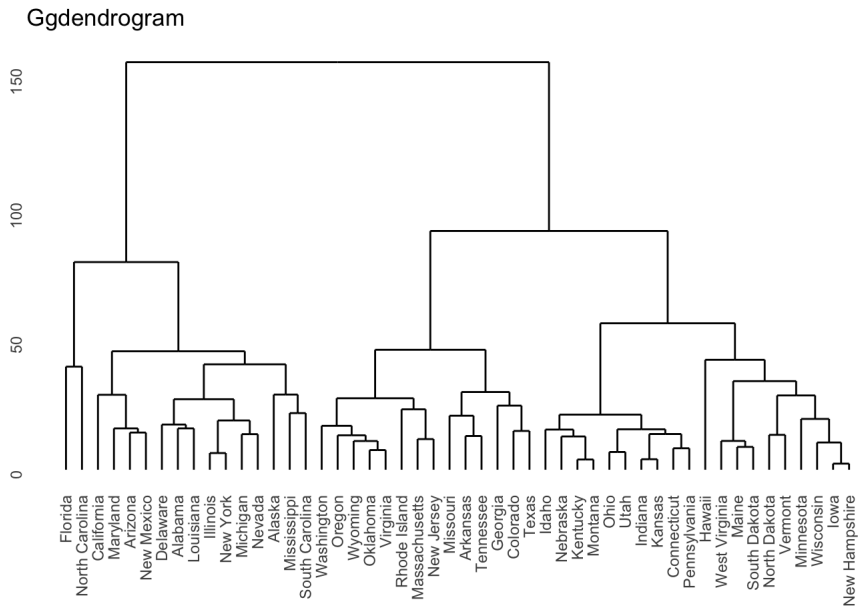
```
# First, you need to install and load the package of "ggdendrogram()":
# *When you type the function, don't include the # sign*
# install.packages('ggdendro')
library(ggdendro)

# Now let's take a look at the "USArrests" by function "head()":
head(USArrests)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```
# You can arrange the data into Hierarchical Clustering with "hclust()" and "dist()":
hc <- hclust(dist(USArrests), "ave")

# Ggdendrogram
ggdendrogram(hc, rotate = TRUE, size = 2) +
  ggtitle("Ggdendrogram")
```




```
# You certainly can flip the coordinate by changing the rotate parameter:
ggdendrogram(hc, rotate = FALSE, size = 3) +
  ggtitle("Ggdendrogram")
```



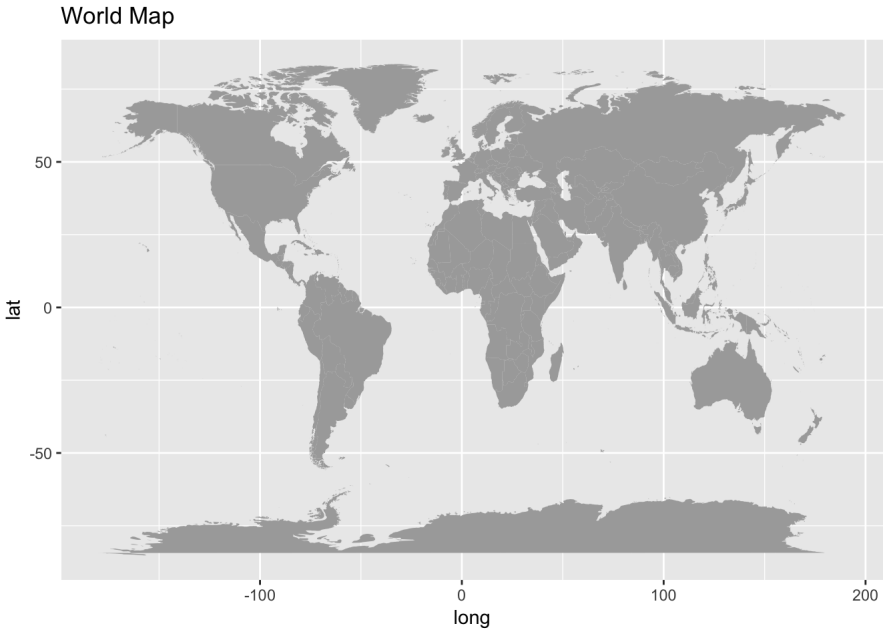
(j)maps

To better visualize the distribution of crimes in different states, you can present data directly in the US map with ggmap and ggplot. Before doings so, you need to download and load necessary functions like before:

```
library(maps)
library(ggmap)
library(mapproj)
```

After downloading, you can now load different maps including world map and US map, with functions “dfworldmap()” and “map_data()” as below:

```
# Load World Map
dfworldmap = map_data("world")
ggplot() +
  geom_polygon(aes(x=long,y=lat, group=group), fill="dark grey", data=dfworldmap) +
  ggtitle("World Map")
```




```
# Load US Map and check it with "head()" function:
us = map_data("state")
head(us)
```

```
##        long      lat group order  region subregion
## 1 -87.46201 30.38968     1     1 alabama      <NA>
## 2 -87.48493 30.37249     1     2 alabama      <NA>
## 3 -87.52503 30.37249     1     3 alabama      <NA>
## 4 -87.53076 30.33239     1     4 alabama      <NA>
## 5 -87.57087 30.32665     1     5 alabama      <NA>
## 6 -87.58806 30.32665     1     6 alabama      <NA>
```

```
# In the list of states in US, "long" represents the longitude and "lat" is for latitude.
```

Before visualizing the data of “USArrests” in different states, you need to first define the regions of states as below:

```
# Add the row of state names to the data set with operation "%>%":
arrest = USArrests %>%
add_rownames("region") %>%
mutate(region=tolower(region)) # Make them lowercase with function "tolower()"
```

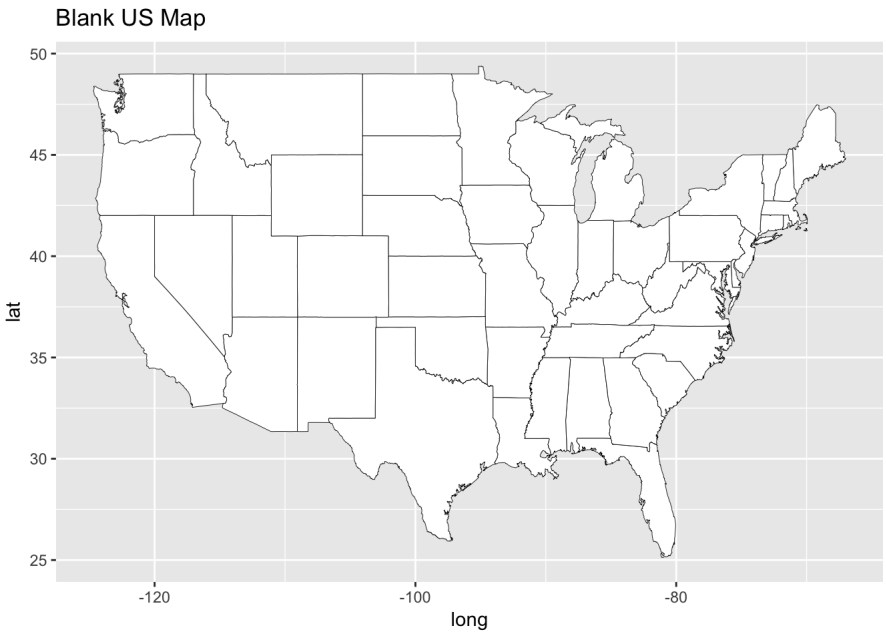
```
## Warning: Deprecated, use tibble::rownames_to_column() instead.
```

Now, you can present the type of crime in different stats with ggplot functions. For example, you can visualize the distribution of murder in US as below:

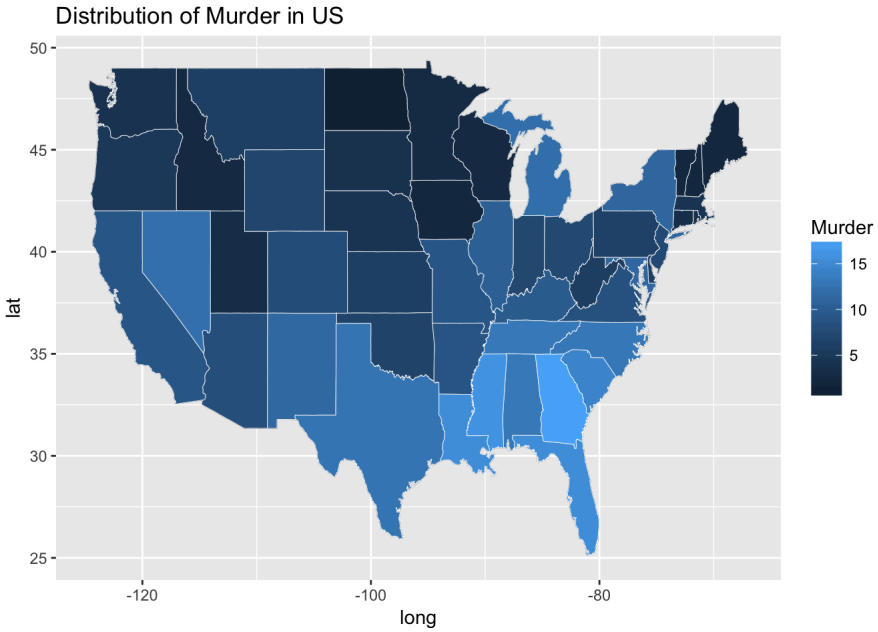
```
# Fist, you need to define a blank map with "geom_map()":
us_map = ggplot() +
  geom_map(data=us, map=us,aes(x=long, y=lat, map_id=region),
           fill="white", color="black", size=0.15) +
  ggtitle("Blank US Map")
```

```
## Warning: Ignoring unknown aesthetics: x, y
```

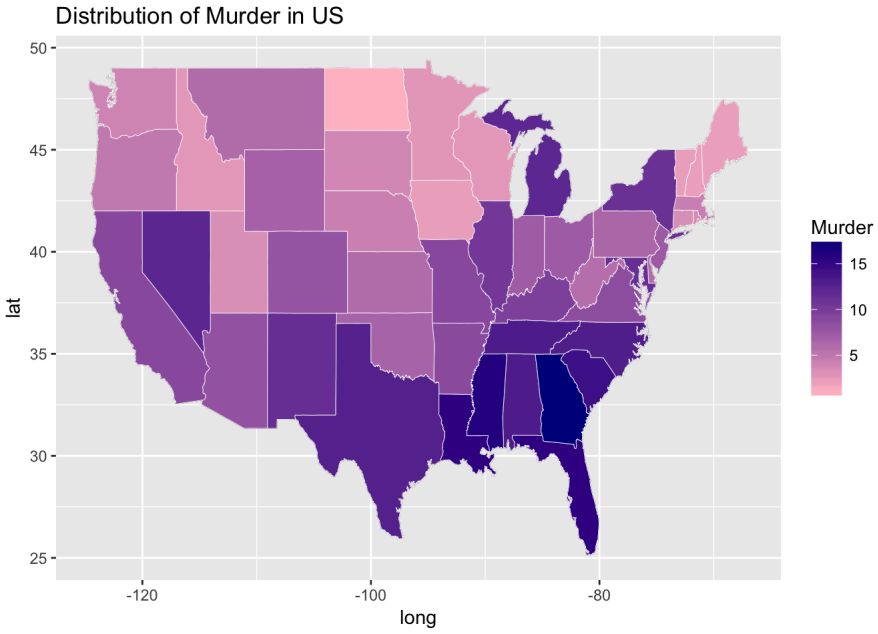
```
us_map # Check the blank map
```



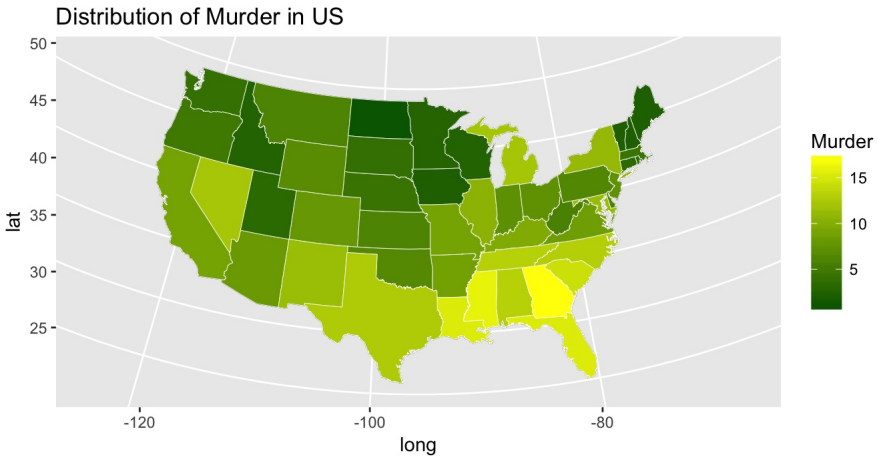

```
# Then, you can fill the blank map with the numbers of Murder in different regions :
us_map +
  geom_map(data=arrest, map=us, aes(fill= Murder, map_id=region), color="white", size=0.15) +
  ggtitle("Distribution of Murder in US")
```




```
# You can modify the gradient color of map with function "scale_fill_continuous" :
us_map +
  geom_map(data=arrest, map=us, aes(fill= Murder, map_id=region), color="white", size=0.15) +
  scale_fill_continuous(low='pink', high='darkblue', guide='colorbar') +
  ggtitle("Distribution of Murder in US")
```




```
# By using "coord_map()", you can project the map into a spherical coordinate:
us_map +
  geom_map(data=arrest, map=us, aes(fill= Murder, map_id=region), color="white", size=0.15) +
  scale_fill_continuous(low='dark green', high='yellow', guide='colorbar') +
  coord_map("albers", lat0 = 39, lat1 = 45) +
  ggtitle("Distribution of Murder in US")
```



(k)ggmap

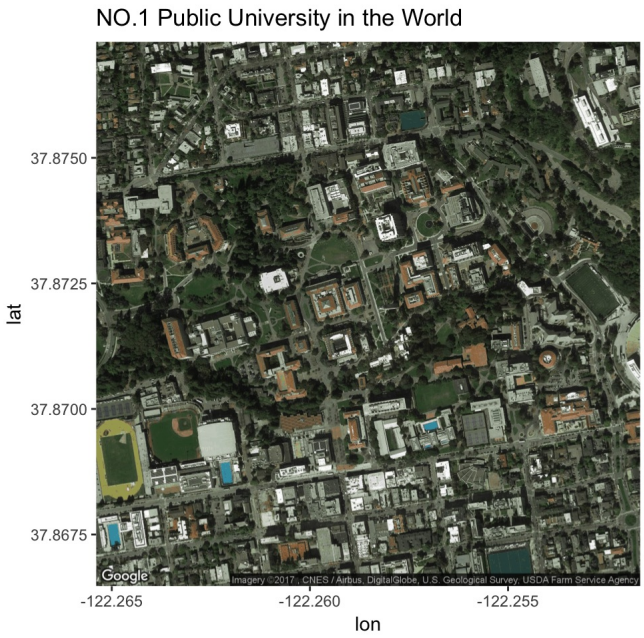
To visualize the data in a smaller range, you can use the function “ggmap()”, which allows you to get the google map of different locations. With “ggmap()” and “get_map()” functions, you can present the location of University of California as below:

```
# Satellite Map of UC Berkeley
map_berkeley = get_map(location = 'University of California, Berkeley', zoom = 16, maptype="satellite")
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=University+of+California,+Berkeley&zoom=16&
size=640x640&scale=2&maptype=satellite&language=en-EN&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=University%20of%20California,%2
0Berkeley&sensor=false
```

```
ggmap(map_berkeley) +
  ggtitle("NO.1 Public University in the World")
```




```
# Road Map of UC Berkeley
map_berkeley = get_map(location = 'University of California, Berkeley', zoom = 16, maptype="roadmap")
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=University+of+California,+Berkeley&zoom=16&
size=640x640&scale=2&maptype=roadmap&language=en-EN&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=University%20of%20California,%2
0Berkeley&sensor=false
```

```
ggmap(map_berkeley) +
  ggtitle("NO.1 Public University in the World")
```


```
# US Google Map
map = suppressMessages(get_map(location = 'United States', zoom = 4))
ggmap(map) +
  ggtitle("US Google Map")
```

By modifying the function, you can mark and circle locations on the map. You can mark places of interest in US as below:

```
# Make a List of Places of Interest
place_of_interest = suppressMessages(geocode(c(
"Space Center Houston",
"Lick Observatory",
"Universal Orlando",
"Moma"), source="google"))

place_of_interest # Check the list
```

```
##          lon      lat
## 1  -95.09799 29.55119
## 2 -121.64291 37.34141
## 3  -81.46782 28.47432
## 4  -73.97762 40.76143
```

```
# Mark the Places of Interest on US Map
ggmap(map) +
  geom_point(aes(x=place_of_interest$lon, y = place_of_interest$lat), lwd = 4, colour = "red") +
  ggtitle("Places of Interest in US")
```

```r
# Circle the Places of Interest on US Map

# You need to first download and load related functions:
library(ggalt)

# Circle the Locations on US Map
ggmap(map) +
  geom_point(aes(x=place_of_interest$lon, y = place_of_interest$lat),lwd = 4, colour = "red") +
   geom_encircle(aes(x=lon, y=lat), data = place_of_interest, size = 2, color = "red") +
   ggtitle("Places of Interest in US")
```

3) Discussion:

In addition to the methods above, there're other approaches to making and improving the ggplot visualization. Therefore, there's more for you to explore, and you can start by answering the questions below: * How to mark and circle locations in a ggplot of World Map? * How to organize the data before visualizing it, like arranging USArrests data based on regions? * Can I modify the attribute in a Hierarchical Dendrogram, such as labels? * How do different attributes including "zoom" and "legend" in ggmap work? …

4) Conclusion:

To summarize, I hope this post allows you to learn about and practice using ggplot, a very useful tool in R that helps you present data beautifully and effectively. For a better understanding of ggplot, you can try reproducing and exploring the codes in this post. Also, You can learn more about ggplot with the references below, and other resources provided in this post. :) Have fun!

5) References:

- ggplot cheatsheat https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
- The Complete ggplot2 Tutorial http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Animated%20Bubble%20Plot
- UC Business Analytics R Programming Guide: Cleveland Dot Plots http://uc-r.github.io/cleveland-dot-plots
- UC Business Analytics R Programming Guide: Lollipop Charts http://uc-r.github.io/lollipop
- Introduction to ggplot http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html
- Top 50 ggplot2 Visualizations http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Animated%20Bubble%20Plot
- Advanced Data Visualization in R https://web.stanford.edu/~imalone/VAM/VAMSlides.pdf