Data Wrangling Final Project

Yitao Liu

2020/5/3

https://github.com/YitaoLiu1996/DataWrangling-Final-Project.git

Introduction:

The National Basketball Association (NBA) is a men's professional basketball league is North America. I like watching NBA since I was a child, so the first idea that comes to my mind is to do something about NBA. Not only because I like NBA, one of the biggest advantages that NBA has is that it provides sufficient and useful data for me to analyze, and the data is not difficult to find and access.

With sufficient data, I find that athlete salaries compared to their stats is an interesting topic for me. So, I decided to wrangle two kinds of data: **The salary data** and **the player stats**, and I want to see if there is any pattern between them. The goal of this project is to scrap the data from different website, wrangle and clean the data, and to see if there's any connection between them.

Data Resources:

There are many different NBA data resources that are available online, but they are all stored in different way. So, I have to choose my resource carefully. After making my goal, there are two kinds of data that are required in my project: **the salary data** and **the stats data**. I will talk about the resources of these two data separately.

1. *Stats Data*

After doing some research, I find that **NBA official site** (https://stats.nba.com/) provides very detailed players stats. So, this will be my resource of players stats data.

2. *Salary Data*

For salary data, I find that there's a well scraped data of salary from **Kaggle** website (https://www.kaggle.com/hultm28/nba-player-salary-data-2002-2017). This dataset includes NBA player salary data between the 2002/03 and 2017/18 seasons - includes player, team, position, and salary amount. This will be one of my resource of players salary data.

However, the data from Kaggle has not being updated for two years. Therefore, I need to find another resource to get the salary data of 2018/19 and 2019/2020 season. The first resource that I find is through **Basketball-Reference** Website (https://www.basketball-reference.com/contracts/players.html). After I scrap the table from the website, I find that there is a problem about this resource: the names of the players do not integrate with the stats data that I found. For example, the following is the name of player Nikola Vucevic on Basketball-Reference:

23	Nikola Vučević	ORL	$28,000,000	$26,000,000	$24,000,000	$22,000,000			Bird Rights	$100,000,000

You can notice that the name contains some letters from other language, where the stats data that I found before is only in English:

34	Nikola Vucevic	54	32.6	19.5	7.9	16.9	47.0	1.5	4.6	32.9	2.1	2.7	78.1	2.4

Therefore, Basketball-Reference is not a good resource for my salary data.

Nevertheless, Basketball-Reference is not totally useless for this project. Later, I find that there is a contract table on it (https://www.basketball-reference.com/contracts/players.html) that could be helpful for making prediction:

545 Contracts Share & more ▾ Glossary

			Salary							
Rk	Player	Tm	2019-20	2020-21	2021-22	2022-23	2023-24	2024-25	Signed Using	Guaranteed
1	Stephen Curry	GSW	$40,231,758	$43,006,362	$45,780,966				Bird Rights	$129,019,086
2	Chris Paul	OKC	$38,506,482	$41,358,814	$44,211,146				Bird Rights	$79,865,296
3	Russell Westbrook	HOU	$38,178,000	$41,006,000	$43,848,000	$46,662,000			Bird Rights	$123,032,000
4	John Wall	WAS	$37,800,000	$40,824,000	$43,848,000	$46,872,000			Bird Rights	$122,472,000
5	James Harden	HOU	$37,800,000	$40,824,000	$43,848,000	$46,872,000			Bird Rights	$122,472,000
6	LeBron James	LAL	$37,436,858	$39,219,565	$41,002,273				Cap Space	$76,656,423
7	Kevin Durant	BRK	$37,199,000	$39,058,950	$40,918,900	$42,778,850			Sign and Trade	$117,176,850
8	Blake Griffin	DET	$34,234,964	$36,595,996	$38,957,028				Bird Rights	$70,830,960
9	Kyle Lowry	TOR	$33,296,296	$30,000,000					Bird Rights	$63,296,296
10	Paul George	LAC	$33,005,556	$35,450,412	$37,895,268				Maximum Salary	$68,455,968

So, I also decide to scrap this table as the resource of players future salary table. However, building statistical model is not the focus of my project, so now I just scrap it for practice.

Next, I find that **ESPN** website (http://www.espn.com/nba/salaries) provides perfect salary data for me that the names of the players well match my stats data. Finally, I decide to choose ESPN website as another resource of player salary data.

Data Collecting:

Now it is time to actually collect the data using R code. The code for data scraping is saved in data.Rmd . First, load some packages that are needed for this project:

```
```{r setup, include=FALSE}
library(readxl)
library(tidyverse)
library(rvest)
library(httr)
library(jsonlite)
library(curl)
```
```

Like above, I will talk about the collecting process of **salary data** and **stats data** separately.

1. *Salary Data*

Firstly, I will collect the player salary data between the 2002/03 and 2017/18 seasons, from **Kaggle**. The data is store in xlsx form: NBASalaryData03-17.xlsx

	A	B	C	D	E
1	team	salary	player	position	season
2	Minnesota Timberwolves	25200000	Kevin Garnett	PF	2002-2003
3	Portland Trail Blazers	13500000	Damon Stoudamire	PG	2002-2003
4	Seattle SuperSonics	13080000	Gary Payton	PG	2002-2003
5	Seattle SuperSonics	12375000	Ray Allen	SG	2002-2003
6	New York Knicks	12375000	Latrell Sprewell	SG	2002-2003
7	Boston Celtics	12375000	Antoine Walker	PF	2002-2003
8	Phoenix Suns	12375000	Stephon Marbury	PG	2002-2003

Next, read the excel file into R:

```
```{r}
## data of 02-03 to 17-18 seasons
salarydata1 <- read_excel('NBASalaryData03-17.xlsx')
salarydata1

```
```

The table looks like this, there are 6255 rows:

team <chr>	salary <dbl>	player <chr>	position <chr>	season <chr>
Minnesota Timberwolves	25200000	Kevin Garnett	PF	2002-2003
Portland Trail Blazers	13500000	Damon Stoudamire	PG	2002-2003
Seattle SuperSonics	13080000	Gary Payton	PG	2002-2003
Seattle SuperSonics	12375000	Ray Allen	SG	2002-2003
New York Knicks	12375000	Latrell Sprewell	SG	2002-2003
Boston Celtics	12375000	Antoine Walker	PF	2002-2003
Phoenix Suns	12375000	Stephon Marbury	PG	2002-2003
San Antonio Spurs	12072000	Tim Duncan	C	2002-2003
Orlando Magic	12072000	Tracy McGrady	SG	2002-2003
Orlando Magic	12072000	Grant Hill	SF	2002-2003

1-10 of 6,255 rows Previous 1 2 3 4

The collecting of salary data between the 2002/03 and 2017/18 seasons is done.

Next step is to scrap the salary data of 2018/19 and 2019/2020 season from **ESPN**. Because some URL address and API-keys are needed from now on, I create a R file that contains those strings: NBA api-keys.R

Let's take a look at the data on ESPN. For each season, there are many tables being on different pages. The URL addresses of 18/19 season are written like this:

http://www.espn.com/nba/salaries/_/year/2019

http://www.espn.com/nba/salaries/_/year/2019/page/2

http://www.espn.com/nba/salaries/_/year/2019/page/3

...

So, there is a pattern in the addresses for different pages. Therefore, I implement a code using a vector for different pages and concatenate them with the common part:

```
## Get the salary data of season 18-19 and season 19-20
source("D:/data wrangling/Data Wrangling Final Project/NBA api-keys.R") ## My api key is saved in this file

## 18-19 season data website address

# Concatenate the url for different pages.
url18
pages18 <- c('','/page/2','/page/3','/page/4','/page/5','/page/6','/page/7','/page/8','/page/9','/page/10','/page/11','/page/12',
'/page/13')
url18 <- paste(url18,pages18,sep='')
url18
```

```
 [1] "http://www.espn.com/nba/salaries/_/year/2019"          "http://www.espn.com/nba/salaries/_/year/2019/page/2"
 [3] "http://www.espn.com/nba/salaries/_/year/2019/page/3"   "http://www.espn.com/nba/salaries/_/year/2019/page/4"
 [5] "http://www.espn.com/nba/salaries/_/year/2019/page/5"   "http://www.espn.com/nba/salaries/_/year/2019/page/6"
 [7] "http://www.espn.com/nba/salaries/_/year/2019/page/7"   "http://www.espn.com/nba/salaries/_/year/2019/page/8"
 [9] "http://www.espn.com/nba/salaries/_/year/2019/page/9"   "http://www.espn.com/nba/salaries/_/year/2019/page/10"
[11] "http://www.espn.com/nba/salaries/_/year/2019/page/11" "http://www.espn.com/nba/salaries/_/year/2019/page/12"
[13] "http://www.espn.com/nba/salaries/_/year/2019/page/13"
```

Do the same thing on 19/20 season address:

```
## 19-20 season data website address

# Concatenate the url for different pages.
url19
pages19 <- c('','/_/page/2','/_/page/3','/_/page/4','/_/page/5','/_/page/6','/_/page/7','/_/page/8','/_/page/9','/_/page/10','/_/
page/11','/_/page/12','/_/page/13','/_/page/14')
url19 <- paste(url19,pages19,sep='')
url19
```

```
 [1] "http://www.espn.com/nba/salaries"                    "http://www.espn.com/nba/salaries/_/page/2/_/page/2"
 [3] "http://www.espn.com/nba/salaries/_/page/3/_/page/3"  "http://www.espn.com/nba/salaries/_/page/4/_/page/4"
 [5] "http://www.espn.com/nba/salaries/_/page/5/_/page/5"  "http://www.espn.com/nba/salaries/_/page/6/_/page/6"
 [7] "http://www.espn.com/nba/salaries/_/page/7/_/page/7"  "http://www.espn.com/nba/salaries/_/page/8/_/page/8"
 [9] "http://www.espn.com/nba/salaries/_/page/9/_/page/9"  "http://www.espn.com/nba/salaries/_/page/10/_/page/10"
[11] "http://www.espn.com/nba/salaries/_/page/11/_/page/11" "http://www.espn.com/nba/salaries/_/page/12/_/page/12"
[13] "http://www.espn.com/nba/salaries/_/page/13/_/page/13" "http://www.espn.com/nba/salaries/_/page/14/_/page/14"
```

Next, write a function **getSalary()** that read these addresses. Let's take a look on the website:

2019-2020 Player Salaries

RK	NAME	TEAM	SALARY
1	Stephen Curry, PG	Golden State Warriors	$40,231,758
2	Chris Paul, PG	Oklahoma City Thunder	$38,506,482
3	Russell Westbrook, PG	Houston Rockets	$38,506,482
4	John Wall, PG	Washington Wizards	$38,199,000
5	Kevin Durant, SF	Brooklyn Nets	$38,199,000
6	James Harden, SG	Houston Rockets	$38,199,000
7	LeBron James, SF	Los Angeles Lakers	$37,436,858
8	Kyle Lowry, PG	Toronto Raptors	$34,996,296
9	Blake Griffin, PF	Detroit Pistons	$34,449,964
10	Tobias Harris, SF	Philadelphia 76ers	$32,742,000
RK	NAME	TEAM	SALARY
11	Jimmy Butler, SF	Miami Heat	$32,742,000
12	Kawhi Leonard, SF	LA Clippers	$32,742,000
13	Klay Thompson, SG	Golden State Warriors	$32,742,000

For this table, there are several things that must be included in the function:

a. Change the column names so that it's easy to join with the stats data later: RK, player, team, salary.

b. Remove rows such like:

RK	NAME	TEAM	SALARY

c. Under the Name column, separate the name with its position by the comma sign (,), and add a new column called position.

d. Change the salary from character into numeric.

e. Add a new column called season to indicate the season.

The code of **getSalary()** is shown below:

```
## Write a function that get the salary data from website
getSalary <- function(x,season=2019){

  # Get data
  salary <- x %>%
    read_html() %>%
    html_table(fill = TRUE)
  salary <- data.frame(salary)

  # Change column name
  names(salary)[names(salary) == "X1"] <- "RK"
  names(salary)[names(salary) == "X2"] <- "player"
  names(salary)[names(salary) == "X3"] <- "team"
  names(salary)[names(salary) == "X4"] <- "salary"

  # Filter - Remove useless rows
  salarydata <- salary %>% filter(!RK == 'RK')

  # Separate name and position
  splitplayer <- strsplit(salarydata$player,',')

  position <- splitplayer %>% sapply('[',2)
  position <- gsub(" ", "", position, fixed = TRUE)

  playername <- splitplayer %>% sapply('[',1)

  # Add name and position column
  salarydata <- salarydata %>% mutate('position' = position) %>% mutate('player' = playername)

  # Change salary into numeric number
  salarydata$salary <- as.numeric(gsub("[\\$,]", "", salarydata$salary))

  # Adding a season column to indicate season.
  finaldata <- select(salarydata, 'team', 'salary', 'player', 'position')
  finaldata <- finaldata %>% mutate('season' = paste(as.character(season),as.character(season+1),sep='-'))
  finaldata
}
```

Next, use the function **getSalary()** to scrap the data from ESPN and concatenate the data of different page together.

18/19 season has 503 rows:

```
# Use the getSalary() function to get the data and concatenate the data together.
salarydata18 <- rbind(getSalary(url18[1],2018),getSalary(url18[2],2018),getSalary(url18[3],2018),getSalary(url18[4],2018),getSalary(url18[5],2018),getSalary(url18[6],2018),getSalary(url18[7],2018),getSalary(url18[8],2018),getSalary(url18[9],2018),getSalary(url18[10],2018),getSalary(url18[11],2018),getSalary(url18[12],2018),getSalary(url18[13],2018))
salarydata18
```

team <chr>	salary <dbl>	player <chr>	position <chr>	season <chr>
Golden State Warriors	37457154	Stephen Curry	PG	2018-2019
Houston Rockets	35654150	Chris Paul	PG	2018-2019
Los Angeles Lakers	35654150	LeBron James	SF	2018-2019
Oklahoma City Thunder	35654150	Russell Westbrook	PG	2018-2019
Detroit Pistons	32088932	Blake Griffin	PF	2018-2019
Boston Celtics	31214295	Gordon Hayward	SF	2018-2019
Toronto Raptors	31200000	Kyle Lowry	PG	2018-2019
Oklahoma City Thunder	30560700	Paul George	SG	2018-2019
Memphis Grizzlies	30521115	Mike Conley	PG	2018-2019
Houston Rockets	30421854	James Harden	SG	2018-2019

1-10 of 503 rows Previous 1 2 3 4 5 6 … 51 N

19/20 season has 525 rows:

```
# Use the getSalary() function to get the data and concatenate the data together.
salarydata19 <- rbind(getSalary(url19[1]),getSalary(url19[2]),getSalary(url19[3]),getSalary(url19[4]),getSalary(url19[5]),getSala
ry(url19[6]),getSalary(url19[7]),getSalary(url19[8]),getSalary(url19[9]),getSalary(url19[10]),getSalary(url19[11]),getSalary(url1
9[12]),getSalary(url19[13]),getSalary(url19[14]))
salarydata19
```

team <chr>	salary <dbl>	player <chr>	position <chr>	season <chr>
Golden State Warriors	40231758	Stephen Curry	PG	2019-2020
Oklahoma City Thunder	38506482	Chris Paul	PG	2019-2020
Houston Rockets	38506482	Russell Westbrook	PG	2019-2020
Washington Wizards	38199000	John Wall	PG	2019-2020
Brooklyn Nets	38199000	Kevin Durant	SF	2019-2020
Houston Rockets	38199000	James Harden	SG	2019-2020
Los Angeles Lakers	37436858	LeBron James	SF	2019-2020
Toronto Raptors	34996296	Kyle Lowry	PG	2019-2020
Detroit Pistons	34449964	Blake Griffin	PF	2019-2020
Philadelphia 76ers	32742000	Tobias Harris	SF	2019-2020

1-10 of 525 rows Previous 1 2 3 4 5 6 … 53 N

The collecting of salary data of 2018/19 and 2019/2020 seasons is done.

Finally, concatenate the salary data of 2018/19 and 2019/2020 season together with the salary data between the 2002/03 and 2017/18 seasons. The number of rows is correct, there are total of 7283 salary data:

team <chr>	salary <dbl>	player <chr>	position <chr>	season <chr>
Minnesota Timberwolves	25200000	Kevin Garnett	PF	2002-2003
Portland Trail Blazers	13500000	Damon Stoudamire	PG	2002-2003
Seattle SuperSonics	13080000	Gary Payton	PG	2002-2003
Seattle SuperSonics	12375000	Ray Allen	SG	2002-2003
New York Knicks	12375000	Latrell Sprewell	SG	2002-2003
Boston Celtics	12375000	Antoine Walker	PF	2002-2003
Phoenix Suns	12375000	Stephon Marbury	PG	2002-2003
San Antonio Spurs	12072000	Tim Duncan	C	2002-2003
Orlando Magic	12072000	Tracy McGrady	SG	2002-2003
Orlando Magic	12072000	Grant Hill	SF	2002-2003

1-10 of 7,283 rows Previous 1 2 3 4 5 6 … 100 Next

I save the final salary data into csv file: salaryData0203-1920.csv

In addition, for the future salary data, I use the following code to get it from **Basketball-Reference**. The code is shown below:

```
## Future salary data

source("D:/data wrangling/Data Wrangling Final Project/NBA api-keys.R") ## My api key is saved in this file

urlslry

futureSalary <- urlslry %>%
    read_html() %>%
    html_table(fill = TRUE)
futureSalary <- data.frame(futureSalary)
futureSalary
```

Here is the table:

Var.1 <chr>	Var.2 <chr>	Var.3 <chr>	Salary <chr>	Salary.1 <chr>	Salary.2 <chr>	Salary.3 <chr>	Salary.4 <chr>	Salary.5 <chr>
Rk	Player	Tm	2019-20	2020-21	2021-22	2022-23	2023-24	2024-25
1	Stephen Curry	GSW	$40,231,758	$43,006,362	$45,780,966			
2	Chris Paul	OKC	$38,506,482	$41,358,814	$44,211,146			
3	Russell Westbrook	HOU	$38,178,000	$41,006,000	$43,848,000	$46,662,000		
4	John Wall	WAS	$37,800,000	$40,824,000	$43,848,000	$46,872,000		
5	James Harden	HOU	$37,800,000	$40,824,000	$43,848,000	$46,872,000		
6	LeBron James	LAL	$37,436,858	$39,219,565	$41,002,273			
7	Kevin Durant	BRK	$37,199,000	$39,058,950	$40,918,900	$42,778,850		
8	Blake Griffin	DET	$34,234,964	$36,595,996	$38,957,028			
9	Kyle Lowry	TOR	$33,296,296	$30,000,000				

1-10 of 600 rows | 1-9 of 11 columns

Previous 1 2 3 4 5 6 … 60 Next

Then, I made several changes:

a. Change the columns name.

b. Remove the useless row by using filter() and select()

```
# Change columns name
names(futureSalary)[names(futureSalary) == "Var.2"] <- "player"
names(futureSalary)[names(futureSalary) == "Var.3"] <- "team"
names(futureSalary)[names(futureSalary) == "Salary"] <- "2019-2020"
names(futureSalary)[names(futureSalary) == "Salary.1"] <- "2020-2021"
names(futureSalary)[names(futureSalary) == "Salary.2"] <- "2021-2022"
names(futureSalary)[names(futureSalary) == "Salary.3"] <- "2022-2023"
names(futureSalary)[names(futureSalary) == "Salary.4"] <- "2023-2024"
names(futureSalary)[names(futureSalary) == "Salary.5"] <- "2024-2025"

# Get useful data
finalFutureSalary <- futureSalary %>% filter(!Var.1 == 'Rk') %>%
  filter(!player == '') %>%
  select('player', 'team', '2019-2020', '2020-2021', '2021-2022', '2022-2023', '2023-2024', '2024-2025')
finalFutureSalary
```

The final table of future salary is saved in futureSalary.csv , and it is shown below:

player <chr>	team <chr>	2019-2020 <chr>	2020-2021 <chr>	2021-2022 <chr>	2022-2023 <chr>	2023-2024 <chr>	2024-2025 <chr>
Stephen Curry	GSW	$40,231,758	$43,006,362	$45,780,966			
Chris Paul	OKC	$38,506,482	$41,358,814	$44,211,146			
Russell Westbrook	HOU	$38,178,000	$41,006,000	$43,848,000	$46,662,000		
John Wall	WAS	$37,800,000	$40,824,000	$43,848,000	$46,872,000		
James Harden	HOU	$37,800,000	$40,824,000	$43,848,000	$46,872,000		
LeBron James	LAL	$37,436,858	$39,219,565	$41,002,273			
Kevin Durant	BRK	$37,199,000	$39,058,950	$40,918,900	$42,778,850		
Blake Griffin	DET	$34,234,964	$36,595,996	$38,957,028			
Kyle Lowry	TOR	$33,296,296	$30,000,000				
Paul George	LAC	$33,005,556	$35,450,412	$37,895,268			

1-10 of 545 rows

Previous 1 2 3 4 5 6 _ 55 Next

2. *Stats Data*

The stats data is scraped from **NBA official site**:

#	PLAYER	GP	MIN	PTS	FGM	FGA	FG%	3PM	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST	STL	BLK	TOV	EFF
1	James Harden	61	36.7	34.4	9.9	22.7	43.5	4.4	12.6	35.2	10.1	11.8	86.1	1.0	5.3	6.4	7.4	1.7	0.9	4.5	31.8
2	Bradley Beal	57	36.0	30.5	10.4	22.9	45.5	3.0	8.4	35.3	6.8	8.0	84.2	0.9	3.3	4.2	6.1	1.2	0.4	3.4	25.4
3	Giannis Antetokounmpo	57	30.9	29.6	10.9	20.0	54.7	1.5	4.8	30.6	6.3	10.0	63.3	2.3	11.5	13.7	5.8	1.0	1.0	3.7	34.8
4	Trae Young	60	35.3	29.6	9.1	20.8	43.7	3.4	9.5	36.1	8.0	9.3	86.0	0.5	3.7	4.3	9.3	1.1	0.1	4.8	26.6

For every season, I choose to get the data of Regular Season, Per Game, and sort the data by PTS. Unlike the salary data, NBA official site stores the data in JSON format:


```
{"resource":"leagueleaders","parameters":{"LeagueID":"00","PerMode":"PerGame","StatCategory":"PTS","Season":"2002-03","SeasonType":"Regular Season","Scope":"S","ActiveFlag":null},"resultSet":
{"name":"LeagueLeaders","headers":
["PLAYER_ID","RANK","PLAYER","TEAM","GP","MIN","FGM","FGA","FG_PCT","FG3M","FG3A","FG3_PCT","FTM","FTA","FT_PCT","OREB","DREB","REB","AST","STL","BLK","TOV","PTS","EFF"],"rowSet":[[1503,1,"Tracy
McGrady","ORL",75,39.400000,11.100000,24.200000,0.457000,2.300000,6.000000,0.386000,7.700000,9.700000,0.793000,1.600000,4.900000,6.500000,5.500000,1.700000,0.800000,2.600000,32.100000,28.800000],[977,2,"Kobe
Bryant","LAL",82,41.500000,10.600000,23.500000,0.451000,1.500000,4.000000,0.383000,7.300000,8.700000,0.843000,1.300000,5.600000,6.900000,5.900000,2.200000,0.800000,3.500000,30.000000,28.000000],[947,3,"Allen
Iverson","PHI",82,42.500000,9.800000,23.700000,0.414000,1.000000,3.700000,0.277000,7.000000,9.000000,0.774000,0.800000,3.400000,4.200000,5.500000,2.700000,0.200000,3.500000,27.600000,20.900000],
[406,4,"Shaquille
O'Neal","LAL",67,37.800000,10.400000,18.100000,0.574000,0.000000,0.000000,0.000000,6.700000,10.800000,0.622000,3.900000,7.200000,11.100000,3.100000,0.600000,2.400000,2.900000,27.500000,29.900000],[1718,5,"Paul
Pierce","BOS",79,39.200000,8.400000,20.200000,0.416000,1.500000,4.900000,0.302000,7.600000,9.500000,0.802000,1.300000,6.000000,7.300000,4.400000,1.800000,0.800000,3.600000,25.900000,22.900000],[1717,6,"Dirk
Nowitzki","DAL",80,39.000000,8.600000,18.600000,0.463000,1.900000,4.900000,0.379000,6.000000,6.900000,0.881000,1.000000,8.900000,9.900000,3.000000,1.400000,1.000000,1.900000,25.100000,27.700000],[1495,7,"Tim
Duncan","SAS",81,39.300000,8.800000,17.200000,0.513000,0.100000,0.300000,0.273000,5.600000,7.800000,0.710000,3.200000,9.700000,12.900000,3.900000,0.700000,2.900000,3.100000,23.300000,29.900000],[185,8,"Chris
Webber","SAC",67,39.100000,9.900000,21.400000,0.461000,0.100000,0.300000,0.238000,3.200000,5.300000,0.607000,2.400000,8.100000,10.500000,5.400000,1.600000,1.300000,3.200000,23.000000,25.000000],[708,9,"Kevin
Garnett","MIN",82,40.500000,9.100000,18.100000,0.502000,0.200000,0.900000,0.282000,4.600000,6.100000,0.751000,3.000000,10.500000,13.400000,6.000000,1.400000,1.600000,2.800000,23.000000,32.100000],[951,10,"Ray
Allen","SEA",76,37.900000,7.900000,17.900000,0.439000,2.600000,7.000000,0.377000,4.200000,4.500000,0.916000,1.200000,3.800000,5.000000,4.400000,1.400000,0.200000,2.600000,22.500000,20.400000],[275,11,"Allan
```


The API-keys are written in this form:

https://stats.nba.com/stats/leagueLeaders?LeagueID=00&PerMode=PerGame&Scope=S&Season=2002-03&SeasonType=Regular+Season&StatCategory=PTS

Simply changing the year number after "Season=" will give you the stats data for different season. By finding the pattern of the API-keys, I divide the seasons into 3 parts:

a. 02/03 to 08/09 seasons.
b. 09/10 season.
c. 10/11 to 19/20 seasons.

Next, using the same technique, save the API-keys into NBA api-keys.R, implement a code using a vector for different seasons and concatenate them with the common part.

For 02/03 to 08/09 season:

```
## Get the stats data of season 02-03 to 08-09

source("D:/data wrangling/Data Wrangling Final Project/NBA api-keys.R") ## My api key is saved in this file.

# The website address of season 02-03 to 08-09
statsUrl1head
yearN1 <- c(2,3,4,5,6,7,8)
statsUrl1tail
statsUrl1 <- paste(statsUrl1head, as.character(yearN1), "-0", as.character((yearN1+1)), statsUrl1tail, sep='')
statsUrl1
```

```
[1] "https://stats.nba.com/stats/leagueLeaders?LeagueID=00&PerMode=PerGame&Scope=S&Season=2002-
03&SeasonType=Regular+Season&StatCategory=PTS"
[2] "https://stats.nba.com/stats/leagueLeaders?LeagueID=00&PerMode=PerGame&Scope=S&Season=2003-
04&SeasonType=Regular+Season&StatCategory=PTS"
[3] "https://stats.nba.com/stats/leagueLeaders?LeagueID=00&PerMode=PerGame&Scope=S&Season=2004-
05&SeasonType=Regular+Season&StatCategory=PTS"
[4] "https://stats.nba.com/stats/leagueLeaders?LeagueID=00&PerMode=PerGame&Scope=S&Season=2005-
06&SeasonType=Regular+Season&StatCategory=PTS"
[5] "https://stats.nba.com/stats/leagueLeaders?LeagueID=00&PerMode=PerGame&Scope=S&Season=2006-
07&SeasonType=Regular+Season&StatCategory=PTS"
[6] "https://stats.nba.com/stats/leagueLeaders?LeagueID=00&PerMode=PerGame&Scope=S&Season=2007-
08&SeasonType=Regular+Season&StatCategory=PTS"
[7] "https://stats.nba.com/stats/leagueLeaders?LeagueID=00&PerMode=PerGame&Scope=S&Season=2008-
09&SeasonType=Regular+Season&StatCategory=PTS"
```

Then write a function **getStats1()** to access and read the JSON file:

```
getStats1 <- function(x){
  stats <- x %>%
    curl() %>%
    readLines() %>%
    fromJSON()
  stats$resultSet$headers

  # Create database
  statsDt <- as.tibble(stats$resultSet$rowSet)
  names(statsDt) <- stats$resultSet$headers
  statsDt
}
```

Read those files:

```
getStats1(statsUrl1[1]) #season 02-03
getStats1(statsUrl1[2]) #season 03-04
getStats1(statsUrl1[3]) #season 04-05
getStats1(statsUrl1[4]) #season 05-06
getStats1(statsUrl1[5]) #season 06-07
getStats1(statsUrl1[6]) #season 07-08
getStats1(statsUrl1[7]) #season 08-09
```

The part of the data of season 02-03 is shown below:

PLAYER_ID <chr>	RANK <chr>	PLAYER <chr>	TEAM <chr>	GP <chr>	MIN <chr>	FGM <chr>	FGA <chr>	FG_PCT <chr>	FG3M <chr>
1503	1	Tracy McGrady	ORL	75	39.4	11.1	24.2	0.457	2.3
977	2	Kobe Bryant	LAL	82	41.5	10.6	23.5	0.451	1.5
947	3	Allen Iverson	PHI	82	42.5	9.8	23.7	0.414	1
406	4	Shaquille O'Neal	LAL	67	37.8	10.4	18.1	0.574	0
1718	5	Paul Pierce	BOS	79	39.2	8.4	20.2	0.416	1.5
1717	6	Dirk Nowitzki	DAL	80	39	8.6	18.6	0.463	1.9
1495	7	Tim Duncan	SAS	81	39.3	8.8	17.2	0.513	0.1
185	8	Chris Webber	SAC	67	39.1	9.9	21.4	0.461	0.1
708	9	Kevin Garnett	MIN	82	40.5	9.1	18.1	0.502	0.2
951	10	Ray Allen	SEA	76	37.9	7.9	17.9	0.439	2.6

1-10 of 189 rows | 1-10 of 24 columns Previous 1 2 3 4 5 6 .. 19 Next

For the rest of the seasons, the approaches are basically the same, so I will not show the codes and images. I have saved the stats data of every season into csv file, under the folder NBAstats Files .

Next, I want to concatenate all the files in NBAstats Files together to become a whole data table from 02/03 to 19/20 season. Write a function **getStatsFile()** to read the saved stats file from the folder, in the function we also need to add a new season column as we did with salary data:

```
## Function that read stats csv files and add season column
getStatsFile <- function(season){
  data <- read_csv(file = paste('NBAstats Files/stats-',season,'.csv',sep = ''))

  firstN = substr(season,1,2)
  secondN = substr(season,3,4)
  data <- data %>% mutate('season' = paste('20',firstN,'-20',secondN,sep=''))
}
```

Finally, read and concatenate the stats data, this is the final stats data:

```
s0203 <- getStatsFile('0203')
s0304 <- getStatsFile('0304')
s0405 <- getStatsFile('0405')
s0506 <- getStatsFile('0506')
s0607 <- getStatsFile('0607')
s0708 <- getStatsFile('0708')
s0809 <- getStatsFile('0809')
s0910 <- getStatsFile('0910')
s1011 <- getStatsFile('1011')
s1112 <- getStatsFile('1112')
s1213 <- getStatsFile('1213')
s1314 <- getStatsFile('1314')
s1415 <- getStatsFile('1415')
s1516 <- getStatsFile('1516')
s1617 <- getStatsFile('1617')
s1718 <- getStatsFile('1718')
s1819 <- getStatsFile('1819')
s1920 <- getStatsFile('1920')

statsData <- rbind(s0203,s0304,s0405,s0506,s0607,s0708,s0809,s0910,s1011,s1112,s1213,s1314,s1415,s1516,s1617,s1718,s1819,s1920)
statsData
```

I save the final stats data into statsData0203-1920.csv . The table is shown below, there are total of 3873 stats data:

PLAYER_ID <dbl>	RANK <dbl>	PLAYER <chr>	TEAM <chr>	GP <dbl>	MIN <dbl>	FGM <dbl>	FGA <dbl>	FG_PCT <dbl>	FG3M <dbl>
1503	1	Tracy McGrady	ORL	75	39.4	11.1	24.2	0.457	2.3
977	2	Kobe Bryant	LAL	82	41.5	10.6	23.5	0.451	1.5
947	3	Allen Iverson	PHI	82	42.5	9.8	23.7	0.414	1.0
406	4	Shaquille O'Neal	LAL	67	37.8	10.4	18.1	0.574	0.0
1718	5	Paul Pierce	BOS	79	39.2	8.4	20.2	0.416	1.5
1717	6	Dirk Nowitzki	DAL	80	39.0	8.6	18.6	0.463	1.9
1495	7	Tim Duncan	SAS	81	39.3	8.8	17.2	0.513	0.1
185	8	Chris Webber	SAC	67	39.1	9.9	21.4	0.461	0.1
708	9	Kevin Garnett	MIN	82	40.5	9.1	18.1	0.502	0.2
951	10	Ray Allen	SEA	76	37.9	7.9	17.9	0.439	2.6

1-10 of 3,873 rows | 1-10 of 25 columns

Previous 1 2 3 4 5 6 … 100 Next

FG3A <dbl>	FG3_PCT <dbl>	FTM <dbl>	FTA <dbl>	FT_PCT <dbl>	OREB <dbl>	DREB <dbl>	REB <dbl>	AST <dbl>	STL <dbl>
6.0	0.386	7.7	9.7	0.793	1.6	4.9	6.5	5.5	1.7
4.0	0.383	7.3	8.7	0.843	1.3	5.6	6.9	5.9	2.2
3.7	0.277	7.0	9.0	0.774	0.8	3.4	4.2	5.5	2.7
0.0	0.000	6.7	10.8	0.622	3.9	7.2	11.1	3.1	0.6
4.9	0.302	7.6	9.5	0.802	1.3	6.0	7.3	4.4	1.8
4.9	0.379	6.0	6.9	0.881	1.0	8.9	9.9	3.0	1.4
0.3	0.273	5.6	7.8	0.710	3.2	9.7	12.9	3.9	0.7
0.3	0.238	3.2	5.3	0.607	2.4	8.1	10.5	5.4	1.6
0.9	0.282	4.6	6.1	0.751	3.0	10.5	13.4	6.0	1.4
7.0	0.377	4.2	4.5	0.916	1.2	3.8	5.0	4.4	1.4

1-10 of 3,873 rows | 11-20 of 25 columns

Previous 1 2 3 4 5 6 … 100 Next

BLK <dbl>	TOV <dbl>	PTS <dbl>	EFF <dbl>	season <chr>
0.8	2.6	32.1	28.8	2002-2003
0.8	3.5	30.0	28.0	2002-2003
0.2	3.5	27.6	20.9	2002-2003
2.4	2.9	27.5	29.9	2002-2003
0.8	3.6	25.9	22.9	2002-2003
1.0	1.9	25.1	27.7	2002-2003
2.9	3.1	23.3	29.9	2002-2003
1.3	3.2	23.0	25.0	2002-2003
1.6	2.8	23.0	32.1	2002-2003
0.2	2.6	22.5	20.4	2002-2003

Previous 1 2 3 4 5 6 … 100 Next

The number of stats data is smaller than the number of salary data is because that there is a Statistical Minimums to qualify for NBA League Leader, which mean a player must play a certain amount of games to get counted in stats data. So, not all the players are recorded in the stats data.

Last step is to get our final whole data. First, read the final **salary data** and the final **stats data** from the csv files:

```
## Get our final data

# Read stats data
stdata <- read_csv('statsData0203-1920.csv')
stdata


# Read salary data
sldata <- read_csv('salaryData0203-1920.csv')
selected_sldata <- sldata %>% select(player, position,salary,season)
sldata
```

Second, integrate the column names so that they are easy to join. Notice that we need to join two data by player name and season:

```
# Change the column name so that its easy to join the data.
colnames(selected_sldata)[which(names(selected_sldata) == "player")] <- "PLAYER"
selected_sldata

# Join the salary data and stats data
finaldata <- merge(selected_sldata,stdata,by = c("PLAYER","season"))
finaldata
```

After joining, we successfully get the data we want:

PLAYER <chr>	season <chr>	position <chr>	salary <dbl>	PLAYER_ID <dbl>	RANK <dbl>	TEAM <chr>	GP <dbl>	MIN <dbl>	FGM <dbl>
Aaron Brooks	2008-2009	G	1045560	201166	89	HOU	80	25.0	4.0
Aaron Brooks	2009-2010	G	1118520	201166	19	HOU	82	35.6	7.0
Aaron Brooks	2015-2016	G	2250000	201166	191	CHI	69	16.1	2.7
Aaron Brooks	2016-2017	G	2700000	201166	246	IND	65	13.8	1.9
Aaron Gordon	2015-2016	PF	4171680	203932	141	ORL	78	23.9	3.5
Aaron Gordon	2016-2017	PF	4351320	203932	88	ORL	80	28.7	4.9
Aaron Gordon	2018-2019	PF	21590909	203932	56	ORL	78	33.8	6.0
Aaron Gordon	2019-2020	PF	19863636	203932	81	ORL	58	33.0	5.4
Aaron Holiday	2019-2020	PG	2329200	1628988	152	IND	58	23.6	3.5
Aaron McKie	2002-2003	SG	4500000	243	108	PHI	80	29.7	3.6

1-10 of 3,287 rows | 1-10 of 27 columns

Previous 1 2 3 4 5 6 … 100 Next

FGA <dbl>	FG_PCT <dbl>	FG3M <dbl>	FG3A <dbl>	FG3_PCT <dbl>	FTM <dbl>	FTA <dbl>	FT_PCT <dbl>	OREB <dbl>	DREB <dbl>
9.8	0.404	1.4	3.9	0.366	1.9	2.2	0.866	0.4	1.6
16.2	0.432	2.5	6.4	0.398	3.0	3.6	0.822	0.7	2.0
6.8	0.401	1.0	2.7	0.357	0.7	0.9	0.766	0.3	1.2
4.6	0.403	0.7	2.0	0.375	0.5	0.6	0.800	0.3	0.8
7.4	0.473	0.5	1.8	0.296	1.7	2.5	0.668	2.0	4.5
10.8	0.454	1.0	3.3	0.288	2.0	2.7	0.719	1.5	3.6
13.4	0.449	1.6	4.4	0.349	2.4	3.2	0.731	1.7	5.7
12.5	0.433	1.2	3.9	0.301	2.4	3.5	0.675	1.8	5.8
8.6	0.407	1.4	3.5	0.394	1.1	1.2	0.861	0.4	1.9
8.3	0.429	0.5	1.4	0.330	1.4	1.7	0.836	0.8	3.6

1-10 of 3,287 rows | 11-20 of 27 columns

Previous 1 2 3 4 5 6 … 100 Next

REB <dbl>	AST <dbl>	STL <dbl>	BLK <dbl>	TOV <dbl>	PTS <dbl>	EFF <dbl>
2.0	3.0	0.6	0.1	1.6	11.2	9.1
2.6	5.3	0.8	0.2	2.8	19.6	15.8
1.5	2.6	0.4	0.1	1.2	7.1	6.3
1.1	1.9	0.4	0.1	1.0	5.0	4.6
6.5	1.6	0.8	0.7	0.8	9.2	13.2
5.1	1.9	0.8	0.5	1.1	12.7	13.2
7.4	3.7	0.7	0.7	2.1	16.0	18.2
7.6	3.7	0.9	0.6	1.6	14.4	17.3
2.3	3.3	0.8	0.2	1.3	9.4	9.5
4.4	3.5	1.6	0.1	1.4	9.0	12.2

Previous 1 2 3 4 5 6 … 100 Next

Save this table into finalData0203-1920.csv , this is our final data!

Data Analysis:

Now it is time to investigate the data using R code. The code for data analyzing is saved in data_dist_cor.Rmd . This is not the main focus of our project, so I will display shortly.

First, load some packages that are needed for this project:

```
library("ggpubr")
library(Hmisc)
library(corrplot)
library(tidyverse)
```

Second, read the file:

PLAYER <chr>	season <chr>	salary <dbl>	PLAYER_ID <dbl>	RANK <dbl>	TEAM <chr>	GP <dbl>	MIN <dbl>	FGM <dbl>	FGA <dbl>
Aaron Brooks	2008-2009	1045560	201166	89	HOU	80	25.0	4.0	9.8
Aaron Brooks	2009-2010	1118520	201166	19	HOU	82	35.6	7.0	16.2
Aaron Brooks	2015-2016	2250000	201166	191	CHI	69	16.1	2.7	6.8
Aaron Brooks	2016-2017	2700000	201166	246	IND	65	13.8	1.9	4.6
Aaron Gordon	2015-2016	4171680	203932	141	ORL	78	23.9	3.5	7.4
Aaron Gordon	2016-2017	4351320	203932	88	ORL	80	28.7	4.9	10.8
Aaron Gordon	2018-2019	21590909	203932	56	ORL	78	33.8	6.0	13.4
Aaron Gordon	2019-2020	19863636	203932	81	ORL	58	33.0	5.4	12.5
Aaron Holiday	2019-2020	2329200	1628988	152	IND	58	23.6	3.5	8.6
Aaron McKie	2002-2003	4500000	243	108	PHI	80	29.7	3.6	8.3

1-10 of 3,287 rows | 1-10 of 26 columns Previous 1 2 3 4 5 6 _ 100 Next

With these data, we can do what we want.

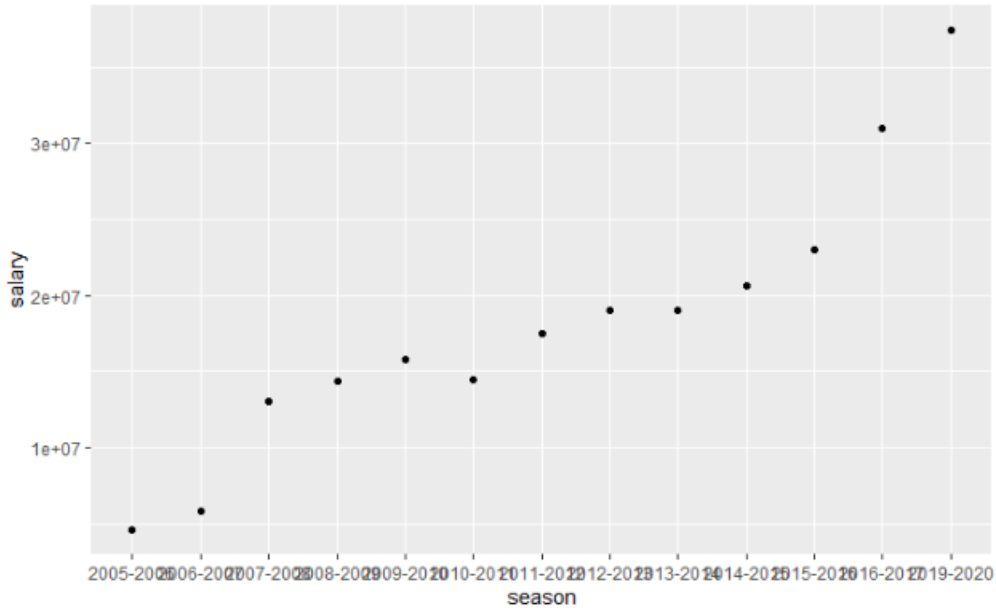
1. Find the teams with the most total salary in 18/19 season:

```
## Find the teams with the most total salary
finaldata %>% filter(season == '2018-2019') %>%
  group_by(TEAM) %>%
  summarise(totalSalary = sum(salary)) %>%
  arrange(desc(totalSalary))
```

TEAM <chr>	totalSalary <dbl>
GSW	135520914
TOR	132186986
OKC	125659399
POR	123035976
HOU	114785798
BOS	113577301
MIL	97107192
SAS	95125945
PHI	93939984
DET	89992738

2. Get the salary graph of LeBron James:

```
## Get the salary graph of LeBron James
finaldata %>% filter(PLAYER == 'LeBron James') %>%
  ggplot(aes(season,salary)) + geom_point()
```



3. Get the graphs of distribution of different stats:

   First, apply the log transformation to the salary data, and select the stats for us to analyze:

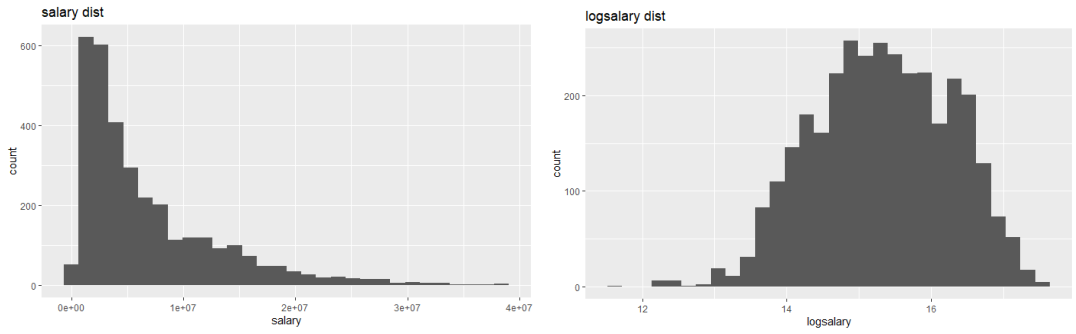
```
## Add a log transformation of salary.
logSalary <- log(finaldata$salary)
logSalary

finaldata <- finaldata %>% mutate('logsalary' = logSalary)
finaldata

##get useful column variables in order to analyse.
model_data <- finaldata %>% select(salary,logsalary,GP,MIN,FGM,FGA,FG_PCT,FG3M,FG3A,FG3_PCT,FTM,FTA,FT_PCT,OREB,DREB,REB,AST,STL,
BLK,TOV,PTS,EFF)
model_data
```

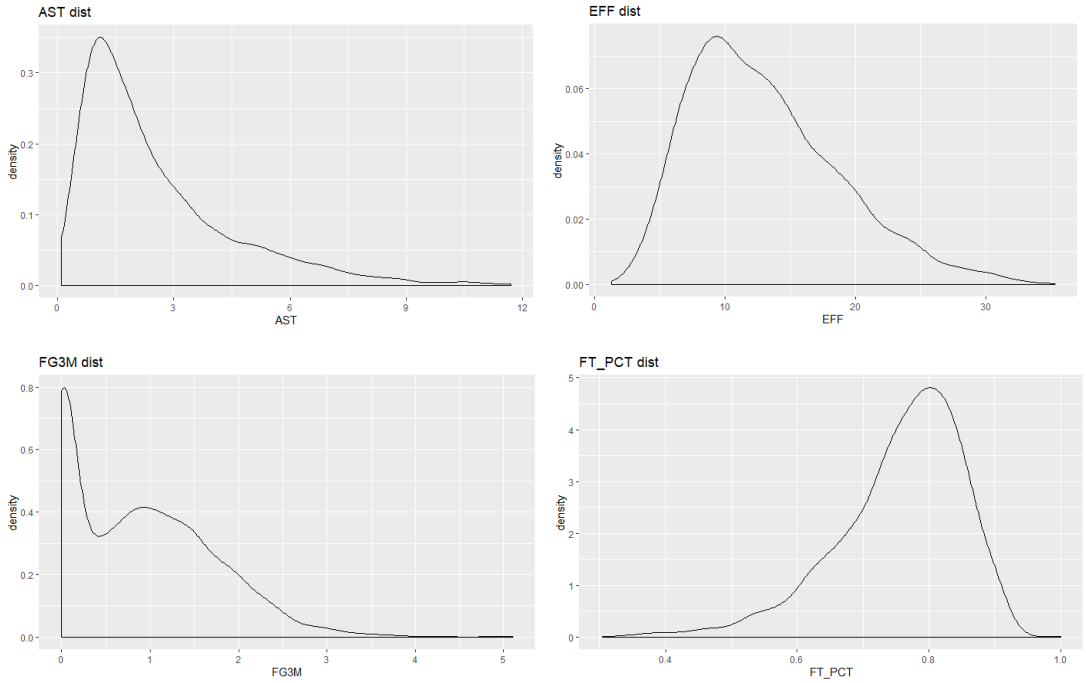
   Second, try histogram on salary data and log transformation of salary data:

```
ggplot(model_data, aes(x = salary)) + geom_histogram() + ggtitle("salary dist")
ggplot(model_data, aes(x = logsalary)) + geom_histogram() + ggtitle("logsalary dist")
```



   Third, try density plots of different stats:

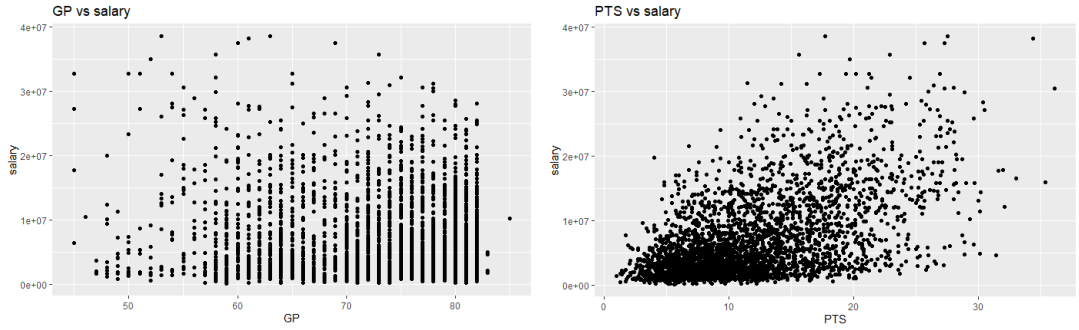
```
ggplot(model_data, aes(x = AST)) + geom_density() + ggtitle("AST dist")
ggplot(model_data, aes(x = EFF)) + geom_density() + ggtitle("EFF dist")
ggplot(model_data, aes(x = FG3M)) + geom_density() + ggtitle("FG3M dist")
ggplot(model_data, aes(x = FT_PCT)) + geom_density() + ggtitle("FT_PCT dist")
```



Next, investigate the relationship between salary and other stats:

```
ggplot(model_data, aes(x = GP, y = salary)) + geom_point() + ggtitle("GP vs salary")
```

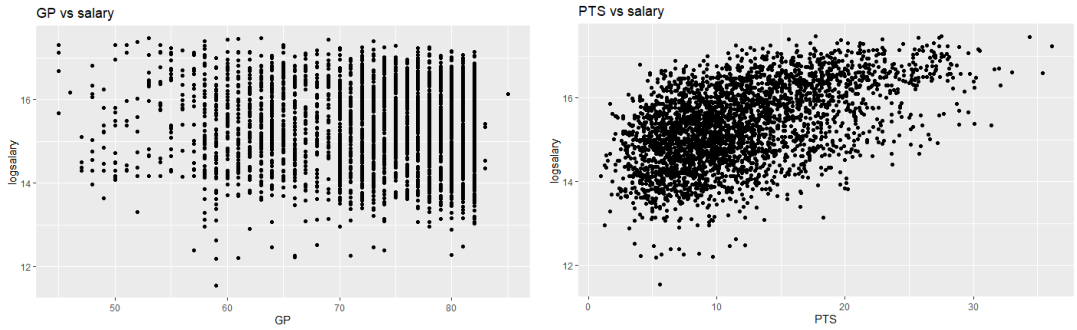
```
ggplot(model_data, aes(x = PTS, y = salary)) + geom_point() + ggtitle("PTS vs salary")
```



Try log salary:

```
ggplot(model_data, aes(x = GP, y = logsalary)) + geom_point() + ggtitle("GP vs salary")
```

```
ggplot(model_data, aes(x = PTS, y = logsalary)) + geom_point() + ggtitle("PTS vs salary")
```



4. Find some correlation between salary and stats:

```
## Find correlation using log salary

selected_model_data <- model_data %>% select(logsalary, GP, MIN, FGM, FGA, FG_PCT, FG3M, FG3A, FG3_PCT, FTM, FTA, FT_PCT, OREB,
DREB, REB, AST, STL, BLK, TOV, PTS, EFF)

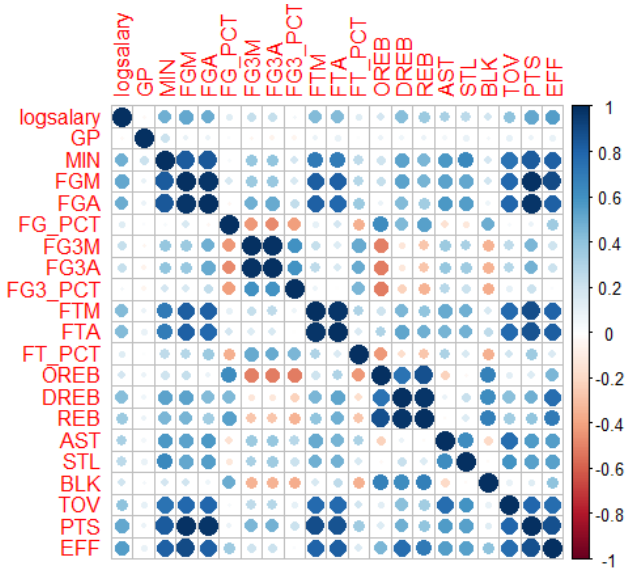
data_corr1 <- rcorr(as.matrix(selected_model_data))
data_corr1
```

```
          logsalary    GP  MIN  FGM   FGA FG_PCT  FG3M  FG3A FG3_PCT  FTM   FTA FT_PCT  OREB  DREB  REB  AST  STL  BLK
logsalary      1.00 -0.03 0.48 0.52  0.49   0.13  0.23  0.24    0.05 0.43  0.43   0.13  0.18  0.42 0.36 0.32 0.27 0.19
GP            -0.03  1.00 0.21 0.09  0.08   0.03 -0.05 -0.07   -0.06 0.09  0.10   0.02  0.09  0.07 0.08 0.06 0.09 0.06
MIN            0.48  0.21 1.00 0.84  0.84   0.04  0.39  0.40    0.18 0.70  0.69   0.26  0.21  0.53 0.45 0.58 0.64 0.18
FGM            0.52  0.09 0.84 1.00  0.97   0.16  0.36  0.38    0.16 0.82  0.81   0.29  0.23  0.54 0.47 0.53 0.52 0.19
FGA            0.49  0.08 0.84 0.97  1.00  -0.04  0.48  0.51    0.25 0.81  0.78   0.36  0.09  0.44 0.34 0.58 0.56 0.08
FG_PCT         0.13  0.03 0.04 0.16 -0.04   1.00 -0.44 -0.48   -0.42 0.11  0.19  -0.35  0.63  0.45 0.54 -0.16 -0.12 0.50
FG3M           0.23 -0.05 0.39 0.36  0.48  -0.44  1.00  0.99    0.60 0.23  0.14   0.51 -0.50 -0.13 -0.27 0.36 0.32 -0.34
FG3A           0.24 -0.07 0.40 0.38  0.51  -0.48  0.99  1.00    0.59 0.26  0.17   0.50 -0.51 -0.13 -0.27 0.39 0.36 -0.34
FG3_PCT        0.05 -0.06 0.18 0.16  0.25  -0.42  0.60  0.59    1.00 0.07 -0.01   0.46 -0.52 -0.22 -0.34 0.27 0.22 -0.37
FTM            0.43  0.09 0.70 0.82  0.81   0.11  0.23  0.26    0.07 1.00  0.98   0.31  0.20  0.45 0.39 0.51 0.48 0.16
FTA            0.43  0.10 0.69 0.81  0.78   0.19  0.14  0.17   -0.01 0.98  1.00   0.15  0.31  0.53 0.49 0.47 0.47 0.26
FT_PCT         0.13  0.02 0.26 0.29  0.36  -0.35  0.51  0.50    0.46 0.31  0.15   1.00 -0.43 -0.18 -0.28 0.31 0.17 -0.36
OREB           0.18  0.09 0.21 0.23  0.09   0.63 -0.50 -0.51   -0.52 0.20  0.31  -0.43  1.00  0.74 0.88 -0.23 -0.04 0.67
DREB           0.42  0.07 0.53 0.54  0.44   0.45 -0.13 -0.13   -0.22 0.45  0.53  -0.18  0.74  1.00 0.97 0.08 0.22 0.63
REB            0.36  0.08 0.45 0.47  0.34   0.54 -0.27 -0.27   -0.34 0.39  0.49  -0.28  0.88  0.97 1.00 -0.02 0.14 0.68
AST            0.32  0.06 0.58 0.53  0.58  -0.16  0.36  0.39    0.27 0.51  0.47   0.31 -0.23  0.08 -0.02 1.00 0.63 -0.20
STL            0.27  0.09 0.64 0.52  0.56  -0.12  0.32  0.36    0.22 0.48  0.47   0.17 -0.04  0.22 0.14 0.63 1.00 -0.02
BLK            0.19  0.06 0.18 0.19  0.08   0.50 -0.34 -0.34   -0.37 0.16  0.26  -0.36  0.67  0.63 0.68 -0.20 -0.02 1.00
TOV            0.41  0.10 0.74 0.78  0.79   0.04  0.26  0.29    0.10 0.77  0.77   0.19  0.13  0.42 0.34 0.77 0.60 0.10
PTS            0.51  0.08 0.84 0.98  0.98   0.09  0.46  0.47    0.22 0.89  0.86   0.36  0.15  0.49 0.40 0.56 0.54 0.13
EFF            0.55  0.09 0.81 0.89  0.82   0.36  0.21  0.21    0.02 0.81  0.82   0.16  0.46  0.77 0.71 0.56 0.55 0.41
           TOV  PTS  EFF
logsalary 0.41 0.51 0.55
GP        0.10 0.08 0.09
MIN       0.74 0.84 0.81
FGM       0.78 0.98 0.89
FGA       0.79 0.98 0.82
FG_PCT    0.04 0.09 0.36
FG3M      0.26 0.46 0.21
FG3A      0.29 0.47 0.21
FG3_PCT   0.10 0.22 0.02
FTM       0.77 0.89 0.81
FTA       0.77 0.86 0.82
FT_PCT    0.19 0.36 0.16
OREB      0.13 0.15 0.46
DREB      0.42 0.49 0.77
REB       0.34 0.40 0.71
AST       0.77 0.56 0.56
STL       0.60 0.54 0.55
BLK       0.10 0.13 0.41
TOV       1.00 0.80 0.76
PTS       0.80 1.00 0.88
EFF       0.76 0.88 1.00
```

Make correlation graph:

```
## Correlation graph

data_corr2 <- cor(selected_model_data)
corrplot(corr = data_corr2)
```



Conclusion:

In order to finish this project, I use a lot of knowledges that learned from the class, including:

1. Loading different packages for different uses.
2. Filtering, selecting, grouping, arranging, concatenating, joining, and summarizing the data.
3. Import and export different files such as csv file and JSON file.
4. Write some function to reduce the repeated work.
5. Manipulate strings such as URL address and API-keys.
6. Scraping tables and data from the website using URL address and API-keys.
7. Interact with different R files.
8. Make graphs by using ggplot.
9. Project in R.
10. Github.

In conclusion, by doing this project, I have not only reviewed my class notes, practiced my skill, but also learned a lot of new stuff. It is a really good experience for me.