Homework 3 for STA 141B
Xinyu Xiong

Before coping with questions, we need the following codes to connect the data base "lahman2013.sqlite" and find the names of tables and their columns:

```
    db = dbConnect(SQLite(), "lahman2013.sqlite")
    dbListTables(db)
    lapply(dbListTables(db), function(tbl) dbListFields(db, tbl))
```

1. What years does the data cover? are there data for each of these years?

The Appearance table identifies information of players every appearance, including the year of appearance (yearID), the team of players, etc. Since there must be some appearances of players in each season, we can find the range of years of this data by focusing on the yearID in Appearance table.

We can find the range by figuring out the maximum and minimum of the year value which is stored as numerical data. Note that the function "dbGetQuery()" is a R function for SELECT query in SQL. The rest of the codes in this report also apply this function to achieve SELECT query in SQL.

```
    P = dbGetQuery(db, "SELECT MIN(yearID), MAX(yearID) FROM Appearances")
```

```
##   MIN(yearID) MAX(yearID)
## 1        1871        2013
```

To know if there are data for each of these years, we can list all the distinct yearID data from Appearance table, and then count the length of data by using function nrow() in R. The result shows us that there are 143 different years, meaning that there are data for each of these years.

```
    D = dbGetQuery(db, "SELECT DISTINCT yearID FROM Appearances")
    nrow(D)
```

2&3. How many (unique) people are included in the database? How many are players, managers, etc? How many players became managers?

People in this database can be divided as players and managers. To know the number of unique people, we need to know the number of people who only played as managers, players, and those players who became managers. Therefore, this question should be answered in the end of this answer after we figured out the three numbers mentioned above.

For find out the number of unique players, we can query the table Master, which contains all information of players. Since we want "unique people", we should look at the playerID, which is a unique code assigned to each player, and SELECT them as DISTINCT values. The total number of unique players is 18354.

```
    Playernum = dbGetQuery(db, "SELECT COUNT(DISTINCT playerID) FROM Master")
```

To figure out the number of unique managers, we need to JOIN the table of Managers and Master to find out the identical playerID in the two fields, which indicates the number of people who has been played as both a player and a manager (M1). Then we figure out the total number of unique managers in Managers table (M2), so that the difference between M2 and M1 would be the number of people who are only managers. We can figure out that M1 is 679 and M2 is 683, meaning that there are 679 players became managers and only 4 managers were not players before. Note that we will not look at the column “plyrMgr” in Managers table, because player managers are those who played as managers and players simultaneously.

```
   M1 = dbGetQuery(db, "SELECT COUNT(DISTINCT Master.playerID) FROM Master
                        INNER JOIN Managers
                         on Master.playerID = Managers.playerID")
   M2 = dbGetQuery(db, "SELECT COUNT(DISTINCT playerID) FROM Managers")
```

Therefore, the total number of unique people in the dataset is 18354 + 4 = 18358; The number of players is 18354; The number of managers is 683; The number of players who became managers is 679.

4. How many players are there in each year, from 2000 to 2013? Do all teams have the same number of players?

To count the number of players in each year, we need to COUNT DISTINCT playerID in the Appearances table, GROUPing BY yearID. Specifically, we also need to set the range of yearID BETWEEN 2000 AND 2013.

```
   app = dbGetQuery(db, "SELECT COUNT (DISTINCT playerID), yearID FROM
                     Appearances
                     WHERE yearID BETWEEN 2000 AND 2013
                     GROUP BY yearID")  ##number of players from 2000 to 2013
```

The tail of result is shown below. We can tell the number of players for each year is not exactly the same, but they have similar values from 2008 to 2013.

```
##    COUNT (DISTINCT playerID) yearID
## 9                       1291   2008
## 10                      1266   2009
## 11                      1249   2010
## 12                      1295   2011
## 13                      1284   2012
## 14                      1304   2013
```

To count the sum of number of players for all years, we use a similar method as we mentioned above. One difference here is that we need to query the Teams table, which contains both teamID and yearID. Another difference is that we no longer COUNT for DISTINCT playerID, because we are counting the sum of players for all years, so that we are supposed to count a player for more than one times.

```
   Teamember = dbGetQuery(db, "SELECT COUNT(playerID), teamID FROM
                          Appearances
                           GROUP BY teamID")
```

The tail of result is shown below, we can tell that the sum of the number of players for all years are very different:

```
##     COUNT(playerID) teamID
## 145              19    WS6
## 146              25    WS7
## 147             112    WS8
## 148              38    WS9
## 149             238    WSN
```

5. What team won the World Series in 2010? Include the name of the team, the league and division.

There is a specific column that contains data of which team won the World Series – WSWin. Therefore, what we are going to do here is to SELECT the teamID, lgID (the name of the league) and divID (The name of the division) from a row WHERE the column of WSWin says “This team won the World Series” and the yearID of winning is 2010. Here we also SELECT the yearID and WSWin to show that we have choosed the correct row where WSwin is Y and yearID is 2010.

```
    Win2010 = dbGetQuery(db, "SELECT name, yearID, WSWin, lgID AS League,
                     divID AS division  FROM Teams
                      WHERE yearID = 2010 AND WSWin IS 'Y'")
```

The result is as what we expected – there is only one team that won the 2010 World Series, which is San Francisco Giants.

```
##                   name yearID WSWin League division
## 1 San Francisco Giants   2010     Y     NL        W
```

6. What team lost the World Series each year? Again, include the name of the team, league and division.

We may expect the result of this question would be a large table, because each year there could be only one winner and all other teams lost the World Series. We can use a similar method in question 5 to deal with this question. The only two differences is that here we look for rows WHERE the WSWin column has the value of Y (lose the World Series) and don’t specify the yearID as we want to know the losing teams for each year.

```
    Win2010 = dbGetQuery(db, "SELECT name, yearID, WSWin, lgID AS League,
                     divID AS division  FROM Teams
                      WHERE yearID = 2010 AND WSWin IS 'Y'
                      GROUP BY WSWin")
```

There are 2274 rows of results. Here is the tail of the result:

```
##                       name yearID WSWin League division
## 2269 Philadelphia Phillies   2013     N     NL        E
## 2270    Pittsburgh Pirates   2013     N     NL        C
## 2271      San Diego Padres   2013     N     NL        W
## 2272  San Francisco Giants   2013     N     NL        W
## 2273   St. Louis Cardinals   2013     N     NL        C
## 2274  Washington Nationals   2013     N     NL        E
```

7. Compute the table of World Series winners for all years, again with the name of the team, league and division.

The question has a solution that is almost the same as the solution of question 5. Here we just don’t need to specify the yearID, so that the result would return all the World Series winner for each year.

```
    winners = dbGetQuery(db, "SELECT name, yearID, WSWin, lgID AS League,
                    divID AS division FROM Teams
                    WHERE WSWin IS 'Y'")
```

However, the number of results we get is 114 rows of data, while the "Teams" table contains data for 143 years, from 1871 to 2013. By listing the yearID WHERE WSWin is Y (there were World Series winners) and comparing it with the yearID that has been recorded in "Teams" table, we found that there is no record of World Series winner from 1871 to 1883, which partially capture the missing data of World Series winners.

Here is the tail of the table of World Series winners for all years.

```
##                       name yearID WSWin League division

## 109 Philadelphia Phillies   2008     Y     NL        E

## 110      New York Yankees   2009     Y     AL        E

## 111  San Francisco Giants   2010     Y     NL        W

## 112   St. Louis Cardinals   2011     Y     NL        C

## 113  San Francisco Giants   2012     Y     NL        W

## 114        Boston Red Sox   2013     Y     AL        E
```

8. Compute the table that has both the winner and runner-up for the World Series in each tuple/row for all years, again with the name of the team, league and division, and also the number games the losing team won in the series.

The information of winner and runner-up for the World Series for all years are from the table "SeriesPost" WHERE the round column is WS (the level of playoffs is the World Series). However, we don't have all information for each team in SeriesPost, hence we need to join it with the data of teams' information from Teams table.

Since we need information from both the winner teams and runners-up, we need to JOIN the Teams table twice. In the first JOIN, we extract data of winner teams and then we extract data of runners-up for the second JOIN. To distinguish the two JOINs, we can name the first JOIN with Teams table as T1 and the second JOIN as T2. Specifically, we use the code "INNER JOIN" here so that it will only return rows when both tables have a match between the columns.

Then we need to clarify the ways to JOIN these tables. Apparently, we need to make sure that the winner's teamID in SeriesPost (represented as teamIDwinner) table is the same as the teamID in Teams table. Also, we need to sure that the yearIDs are matched, because there are multiple yearIDs corresponding to each teamID in the Teams table. If we don't clarify the match of yearIDs, we wouldn't see a unique row for each year's information.

To make it looks exactly the same as the given example result, we also arrange the rows in a DESCendent ORDER with respect to the value of yearID. Also, we LIMIT our result to the first 3 rows.

```
    winandlost = dbGetQuery(db, "SELECT SeriesPost.yearID, T1.teamID, T1.name,
T1.lgID AS League, T1.divID AS division,
                        T2.teamID, T2.name, T2.lgID AS League, T2.divID AS b
                         division, losses as 'Lost Team Won'
                        FROM SeriesPost
                        INNER JOIN Teams AS T1
                        INNER JOIN Teams AS T2
                        on SeriesPost.teamIDwinner = T1.teamID
                         AND SeriesPost.teamIDloser = T2.teamID
                         AND SeriesPost.yearID = T1.yearID
                         AND SeriesPost.yearID = T2.yearID
                        WHERE round = 'WS'
                        ORDER BY T1.yearID DESC
                        LIMIT 3")
```

Now, regardless of the format, we find our result is exactly the same as the example has.

Example result:
2013|BOS|Boston Red Sox|AL|E|SLN|St. Louis Cardinals|NL|C|3
2012|SFN|San Francisco Giants|NL|W|DET|Detroit Tigers|AL|C|0
2011|SLN|St. Louis Cardinals|NL|C|TEX|Texas Rangers|AL|W|3

My result:

```
##   yearID teamID                 name League division teamID                name
## 1   2013    BOS       Boston Red Sox     AL        E    SLN St. Louis Cardinals
## 2   2012    SFN San Francisco Giants     NL        W    DET      Detroit Tigers
## 3   2011    SLN  St. Louis Cardinals     NL        C    TEX       Texas Rangers
##   League division Lost Team Won
## 1     NL        C             3
## 2     AL        C             0
## 3     AL        W             3
```

9. Do you see a relationship between the number of games won in a season and winning the World Series?

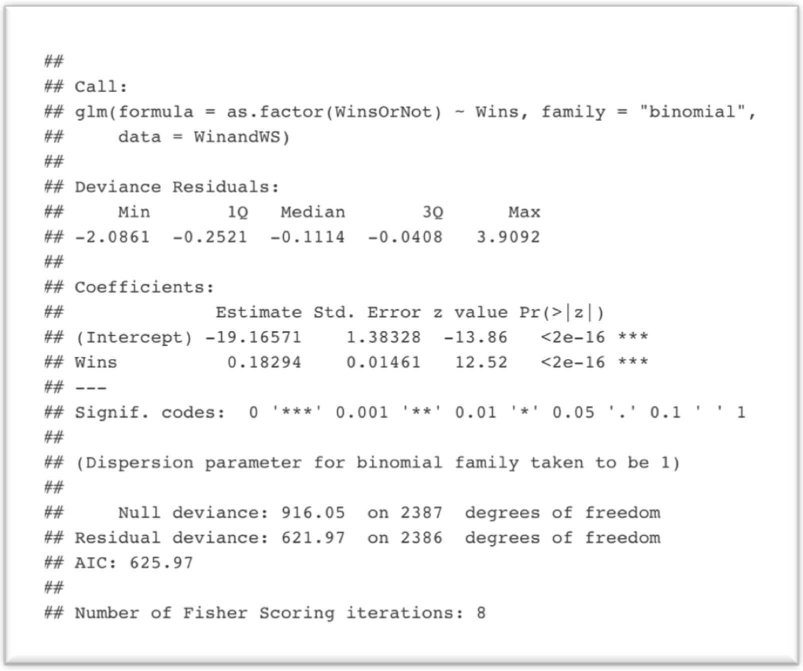
For this question, I will try to construct a logit model where the dependent variable is wining/not winning the World Series, and the independent variable is the number of games won. We firstly SELECT the relevant variables from the Teams table, including WSWin (wining/not winning) and W (number of games won).

```
    WinandWS = dbGetQuery(db, "SELECT yearID, name, W as Wins, WSWin as
                               'WinsOrNot' FROM Teams ORDER BY yearID ")
```

Then we construct a logit model where we also set the WSWin as a categorical variable which has the value either "Y" or "N". We use the function "*glm()*" to construct a logit model and use "*as.factor(WSWin)*" to make WSWinn a categorical variable. Also, please note that I omitted all rows with NA values in the table WinandWS which is created by myself.

```
    WinandWS = na.omit(WinandWS)
    summary(glm(data = WinandWS, as.factor(WinsOrNot) ~ Wins,family =
            "binomial"))
```

The result shows that there is a significant relationship between the championship of World Series and the number of games won. Specifically, as a team wins more games, it is more likely for it to win the World Series.



```
##
## Call:
## glm(formula = as.factor(WinsOrNot) ~ Wins, family = "binomial",
##     data = WinandWS)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0861  -0.2521  -0.1114  -0.0408   3.9092
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -19.16571    1.38328  -13.86   <2e-16 ***
## Wins          0.18294    0.01461   12.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 916.05  on 2387  degrees of freedom
## Residual deviance: 621.97  on 2386  degrees of freedom
## AIC: 625.97
##
## Number of Fisher Scoring iterations: 8
```

10. In 2003, what were the three highest salaries? (We refer here to unique salaries, i.e., there may be several players getting the exact same amount.) Find the players who got any of these 3 salaries with all of their details?

We can SELECT salary information from Salaries table and INNER JOIN it with the Master table to gain some personal details. We can ORDER the information by setting a DESCendent order and LIMIT the rows of data to be 10 (because there may be several players getting the exact same amount, but we are looking for unique salaries). Therefore, I SELECT salary, yearID, and all information from Master table (Master.*).

```
    Salary = dbGetQuery(db,"SELECT yearID, salary, Master.* FROM Salaries
                       INNER JOIN Master
                        on Salaries.playerID = Master.playerID
                        ORDER BY salary DESC
                        LIMIT 10")
```

The result has quite a large number of columns, but only from the yearID, salary and playerID columns, we can know that the top three unique salaries are all paid to one person, who has playerID “rodrial01” and whose name is Alex Rodriguez.

11. For 2010, compute the total payroll of each of the different teams. Next compute the team payrolls for all years in the database for which we have salary information. Display these in a plot.

We calculate the total payroll of each team in 2010 by SELECTING the SUM of salary FROM the salary table, and then set the yearID to be 2010. Also note that we need to GROUP our data BY teamIDs, so that the result will be classified into each team's data.

```
    salary2010 = dbGetQuery(db,"SELECT teamID, SUM(salary) FROM Salaries
                             WHERE yearID = 2010
                             GROUP BY teamID")
```

The tail of result is shown below:

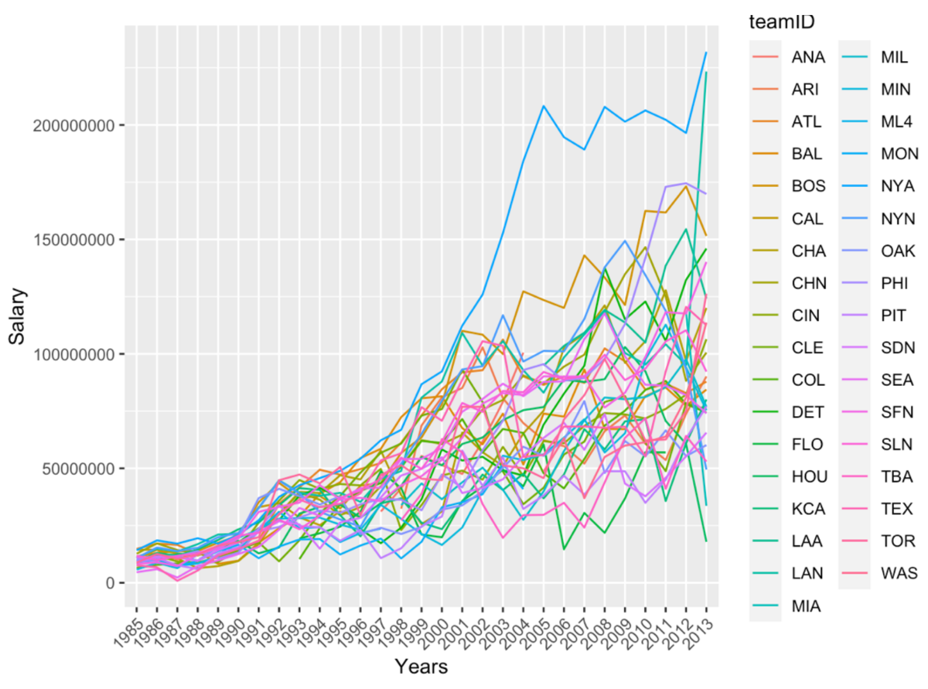
```
##    teamID SUM(salary)
## 25    SFN    98641333
## 26    SLN    93540751
## 27    TBA    71923471
## 28    TEX    55250544
## 29    TOR    62234000
## 30    WAS    61400000
```

To compute the total payroll for each team in each year, we no longer need to set the value of yearID. Since we need to know the variation of payrolls, we would also SELECT yearID in our table.

```
    salaryall = dbGetQuery(db,"SELECT teamID, SUM(salary) AS sumsalary,
                             yearID FROM Salaries
                             GROUP BY teamID, yearID")
```

Now we can draw a plot to represent the variation of payrolls for all the teams through several years.

```
    ggplot(data = salaryall, aes(x = as.factor(yearID), y = sumsalary,
      group = teamID)) + geom_line(aes(col = teamID)) +
      theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1)) +
      labs(x = "Years", y = "Salary")
```



14. Which player has hit the most home runs? Show the number per year.

For this question, we need to focus on the Batting table and SELECT playerID, yearID and the MAX of HR (Home runs). Since we are going to show the number of home runs per year, we need to also GROUP the data BY yearID.

```
    HRnum = dbGetQuery(db, "SELECT playerID, yearID, MAX(HR) AS HR FROM
                          Batting GROUP BY yearID")
```

The head of the result is shown below:

```
##    playerID yearID HR

## 1 meyerle01   1871  4

## 2  pikeli01   1872  6

## 3  pikeli01   1873  4

## 4 orourji01   1874  5

## 5 orourji01   1875  6

## 6  hallge01   1876  5
```

15. Has the distribution of home runs for players increased over the years?

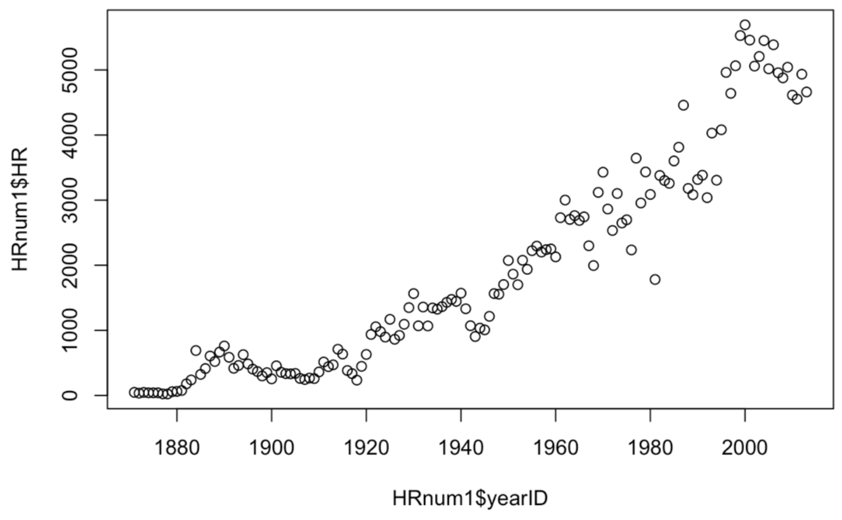
For this question we can refer to the table "HRnum" created by myself in question 14, which shows the yearID and the MAX of home runs. The only difference is that here, we no longer focus on the MAX number but the SUM of HR for each year.

```
    HRnum1 = dbGetQuery(db, "SELECT playerID, yearID, SUM(HR) AS HR FROM
                              Batting GROUP BY yearID")
```

Since we are going to know the trend of home runs for players over the years, we can simply plot the "HRnum1" data, where x = HRnum1$yearID and y = HRnum1$HR.

```
    plot(HRnum1$yearID, HRnum1$HR)
```

The result shows that the distribution of home runs has increased over the years.

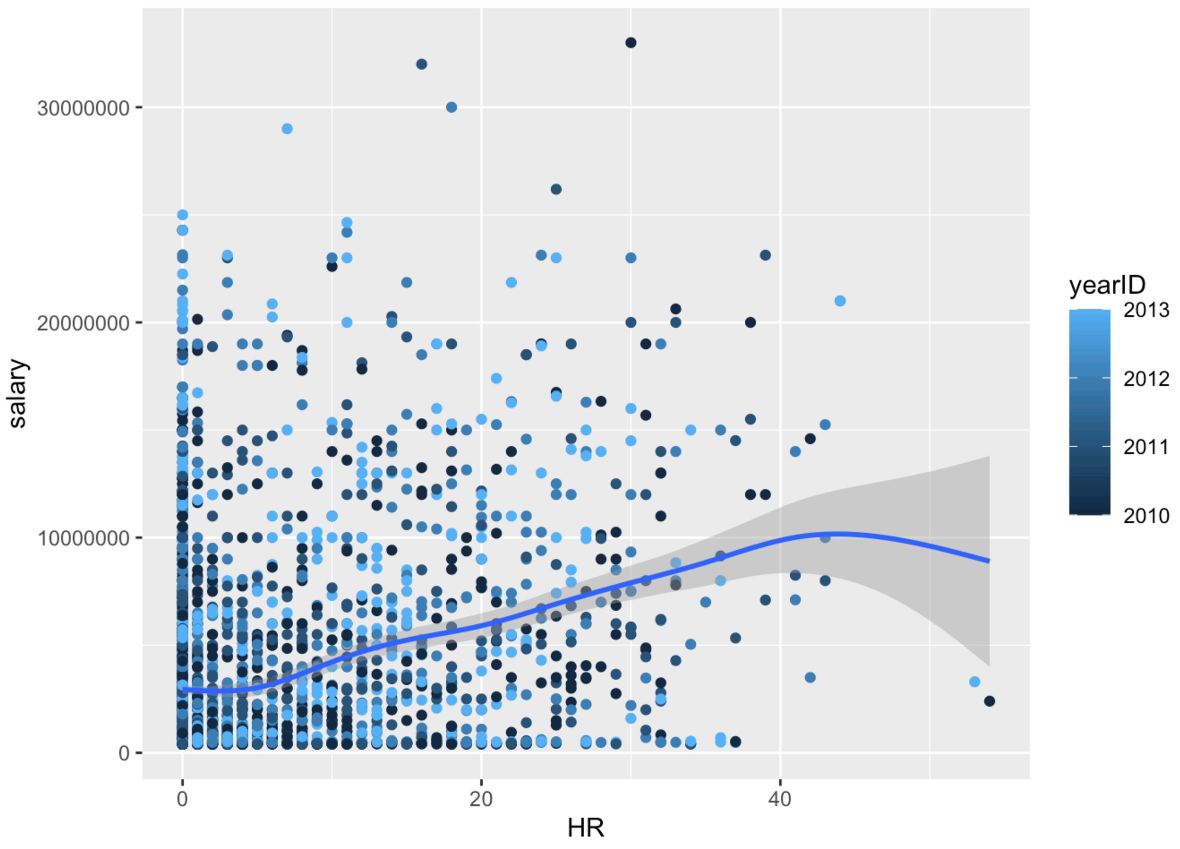


16. Do players who hit more home runs receive higher salaries?

The relationship between the number of home runs and salaries for each player may be confused by a general inflation of salaries. Therefore, I would like to only focus on the data BETWEEN 2010 AND 2013. We need information about HR (home runs) and salary from the Batting table JOINing with Salaries table. Of course, we will also include some columns, such as yearID and playerID. The way of JOIN is matching the same playerID and yearID in the two tables.

```
    sal_vs_run = dbGetQuery(db, "SELECT Batting.yearID, Batting.playerID, HR,
                            salary FROM Batting
                             INNER JOIN Salaries
                             ON Batting.playerID = Salaries.playerID AND
                            Batting.yearID = Salaries.yearID
                             WHERE Batting.yearID BETWEEN 2010 AND 2013
                             ORDER BY Batting.playerID")
```

Now we can represent the relationship between the number of home runs and salaries by ggplot. Here we use geom_point() to show the specific data and geom_smooth to fit a trend. We can tell that as we do more home runs, our salaries will also increase.



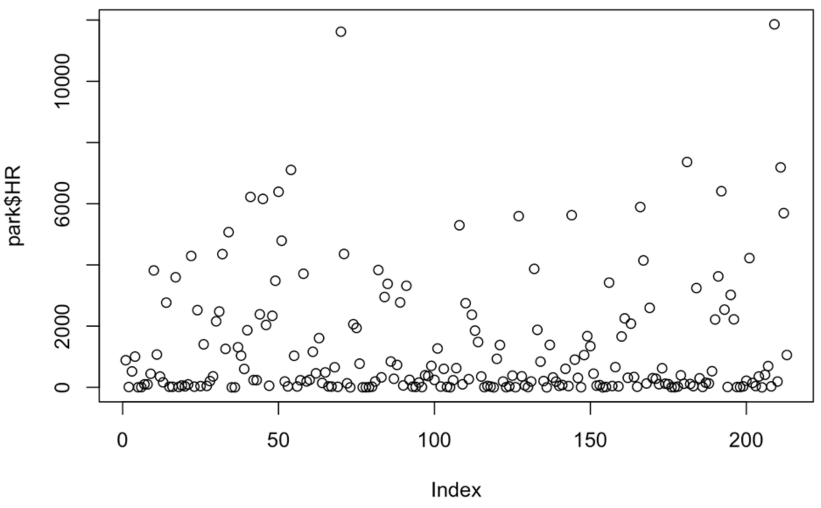
17. Are certain baseball parks better for hitting home runs?

We need to SELECT "park" column in Teams table to show the names of parks. Of course, we also need to include yearID and SUM(HR) to help us identify the park with highest number of home runs.

```
    park = dbGetQuery(db, "SELECT yearID, park, SUM(HR) AS HR FROM Teams
                                GROUP BY park")
```

I made a simple plot to show the sum of home runs in each park and found two numbers that are apparently higher than others.

```
    plot(park$HR)
```



Therefore, I choose the rows where the sum of HR is greater than 10000.

```
    park[park$HR > 10000,]
```

It shows me the two certain baseball parks better for hitting home runs.

```
##     yearID           park    HR
## 70    1934 Fenway Park II 11618
## 209   1914  Wrigley Field 11859
```

20. How many games do pitchers start in a season? Plot this against games finished in a season.

To know how many games pitchers start in a season, we can calculate the SUM of games played (G) WHERE player's stint is 1. Also, we need to include the yearID and GROUP the data by yearID.

```
    Start =  dbGetQuery(db, "SELECT yearID, SUM(G) AS StintGames
                        FROM Pitching WHERE stint = 1
                        GROUP BY yearID
                        ORDER BY yearID")
```

For the number of games played for each year, we can use the same method above but no longer set the stint to be 1.

```
    Game =  dbGetQuery(db, "SELECT yearID, SUM(G) AS Games
                        FROM Pitching GROUP BY yearID
                        ORDER BY yearID")
```

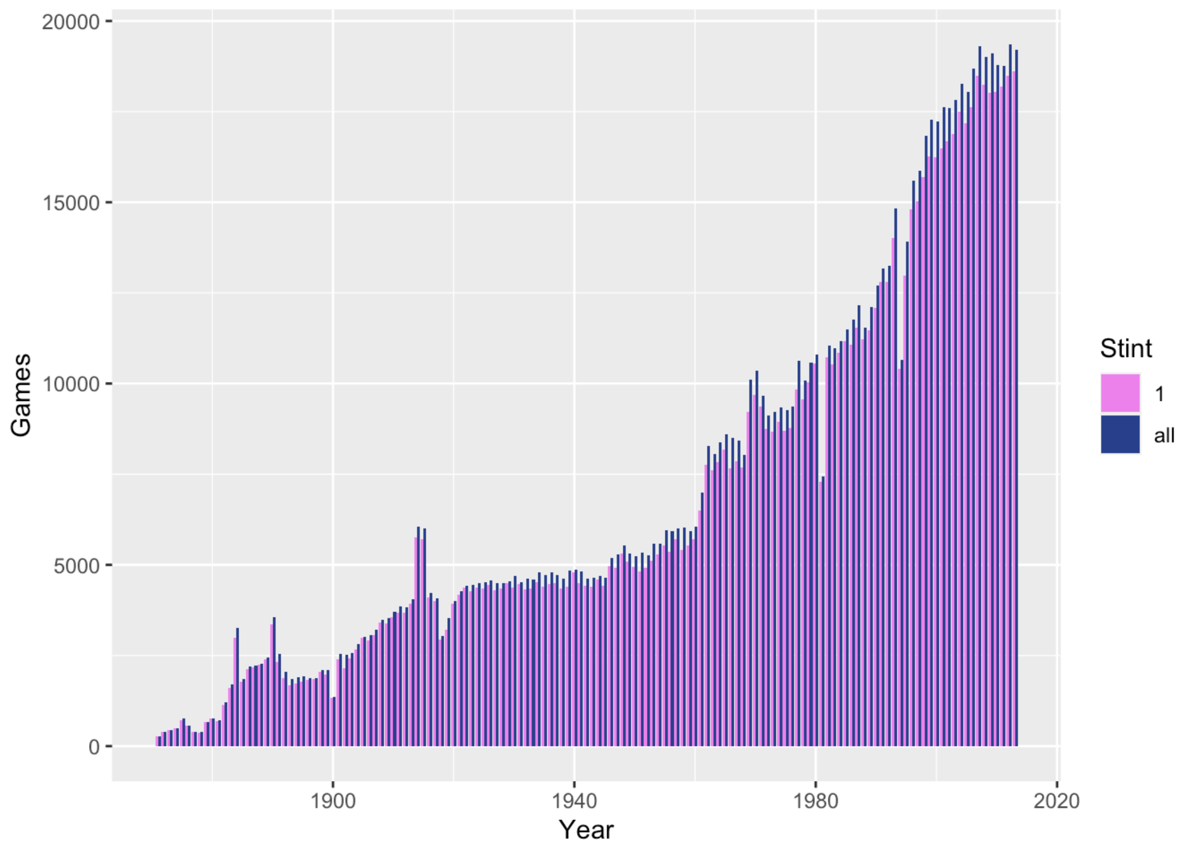
Then we combine the two data into a data frame and plot the number of games pitchers start in a season against number of games finished in a season.

```
    S  =  data.frame(Year  =  Start$yearID,  Games  =  Start$StintGames,  Stint  =
            rep("1", nrow(Start)))
    G = data.frame(Year = Game$yearID, Games = Game$Games, Stint = rep("all",
            nrow(Game)))

    ALL_DATA = rbind(S,G)

    ggplot(data=ALL_DATA, aes(x=Year, y=Games, fill = Stint))
    + geom_bar(stat="identity", position=position_dodge())
     + scale_fill_brewer(palette="Paired")
    + scale_fill_manual(values=c('violet','royalblue4'))
```

The result of ggplot is shown below. Note that the dark blue lines are total number of games each year and the pink lines are the number of games started by pitchers. We can tell that these two graphs are almost overlapped by each other, indicating that these two numbers are almost the same.


Games
Year
Stint
all