STAT428 Group6 Project Written Report

Simulating NBA Match Results and Predicting NBA Playoff Teams

Zepeng Xiao, zepengx2
Shuogong, shuog2
Xiaoping Hua, xh3
Dongfan Li, dongfan2
Sixin Ma, sixinma2

April 28, 2019

Abstract

The main purpose of our project is simulating match results and predicting NBA playoff teams for season 2017-18 based on this season's real data, and comparing our results with this season's real data to calculate our method's accuracy, since analyzing and predictiong NBA stats is popular currently. The method we used in our project includes **Random number generations**, **Permutation test** and **Ridge regression**.

We found that regarding the playoff teams for season 2017-18, the accuracy of our prediction is 87.5% (14/16) when comapring the actual result. The result can be regarded as accurate beacuse we used each team's former performance in this season to make predictions, which is reliable. But there are still some conditions that were not included in the model, like player's injury, team's stamina and so on.

Introduction

The National Basketball Association (NBA) is one of the most worldwide popular sporting events that is held every year in the world.

- It has two conferences, East and West, with 15 teams in each division.
- 30 teams fight in the league fight for a champion in a year-long season from October to June every year.
- Each season is split into regular season and playoffs.
- Each team has 82 matches to play in their regular season
- Teams in the same conference would play more often than those.
- Only the top 8 teams in each division are able to enter the playoffs of the season.
- Ranks are determined ascending by a special statistic called Games Behind:
  - When team a is the leading team of the conference.

$$TeamB'sgamesbehind = \frac{(TeamA'sWins - TeamA'sLosses) - (TeamB'sWins - TeamB'sLosses)}{2}$$

Thus which 16 teams would enter the playoffs every season is one of the biggest mystery in NBA every season.

Our goal, as introduced above, is simulating match results and predicting NBA playoff teams. In other words, we are going to predict which eight teams in each conference would get the lowest games behind by the end of the regular season.

The focus is on the past 17-18 season. We will try to predict based on prior part of this season's real data, and comparing our results with this season's real data in the rest season to calculate our method's accuracy. If we found our prediction is reliable in this season, we have prepared data in the past 10 years for test and verification.

Methods

Processing Data

Raw Data

Raw datasets we used in our project are "2017-18_standing.csv" and "2017-18_teamBoxScore.csv", whose details are showed below:

From 2017-18_standing.csv

```
##       stDate teamAbbr homeWin homeLoss awayWin awayLoss confWin confLoss
## 1 2017-10-17      ATL       0        0       0        0       0        0
## 2 2017-10-17      BKN       0        0       0        0       0        0
## 3 2017-10-17      BOS       0        0       0        1       0        1
##   lastTen gamePlay
## 1       0        0
## 2       0        0
## 3       0        1
```

From 2017-18_teamBoxScore.csv

```
##       gmDate teamAbbr teamConf teamLoc teamRslt teamPTS opptPTS
## 1 2017-10-17      BOS     East    Away     Loss      99     102
## 2 2017-10-17      CLE     East    Home      Win     102      99
## 3 2017-10-17      HOU     West    Away      Win     122     121
```

Processing data to fit models

In our project, we used actual match result from *2017-12-01* to *2018-03-11* to fit a model, and used this model to predict match result from *2018-03-12* to the end of NBA 2017-18 regular season since:

- At the beginning of the semester, each team's winning rate changed rapidly since the number of game played is small, which may be outliers for our fitted model.
- We can make more reliable prediction if we use the data in the same season.

The fitted data is showed below:

```
##       gmDate homeTeam awayTeam homeTeamRate awayTeamRate homeTeamlast10
## 1 2017-12-01       GS      ORL    0.7272727    0.5000000            0.7
## 2 2017-12-01      ORL       GS    0.5000000    0.7272727            0.1
## 3 2017-12-01      DET      WAS    0.8000000    0.5454545            0.6
##   awayTeamlast10  histRate Result
## 1            0.1 0.8333333    Win
## 2            0.7 0.1666667   Loss
## 3            0.5 0.4545455   Loss
```

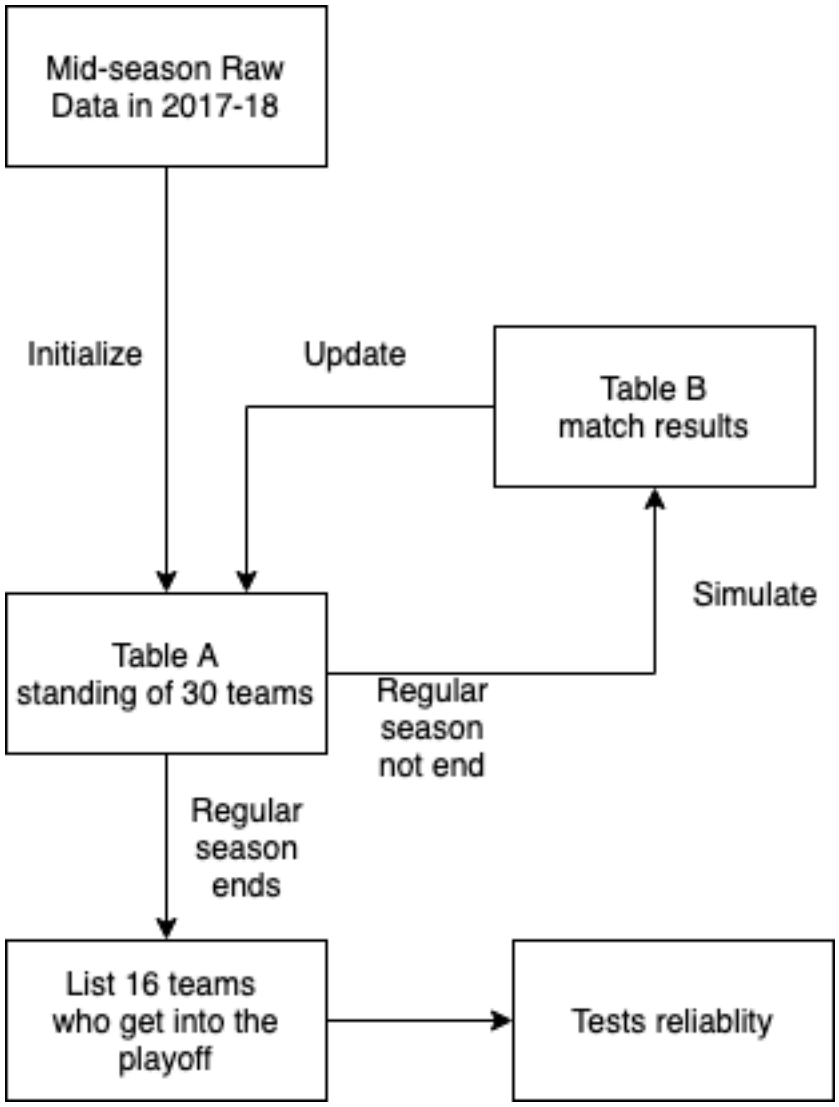
Some important columns' meaning:

- "gmDate": Game date
- "homeTeam" / "awayTeam": The Abbreviation of home team name/away team name
- "homeTeamRate": The winning rate when home team played at home
- "awayTeamRate": The winning rate when away team layed away
- "homeTeamlast10" / "awayTeamlast10": The winning rate of last 10 games for home team/away team
- "histRata": The historical rate when home team play against away team
- "Result": The result for the match

Processing table_a and table_b

Since for simulation part, we have to update the teams' standing after each simulation and use updated data to simulate next match result. We decided to made two tables A and B to store the standing data of all teams (in A) and the match data of each matches (in B). Then use the "current" standing data in A to simulate upcoming matches in B and use the following match data to regenerate new standing data in A.

The following chart about how we use table_a and table_b is showed below (detailed information will be shown in *Simulation* part):



Basic information about table_a and table_b are as follows:

Initial table_a:

```
str(tablea180311)
```

```
## 'data.frame':    30 obs. of  26 variables:
##  $ teamname     : Factor w/ 30 levels "ATL","BKN","BOS",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ confname     : Factor w/ 2 levels "East","West": 1 1 1 1 1 1 2 2 1 2 ...
##  $ homerate     : num  0.429 0.353 0.657 0.528 0.455 ...
##  $ awayrate     : num  0.156 0.273 0.719 0.323 0.242 ...
```

```
##  $ last10       : num  0.2 0.2 0.6 0.5 0.3 0.4 0.3 0.6 0.3 0.7 ...
##  $ homescorewin : num  105 105 104 108 103 ...
##  $ homescorelost: num  108.2 108.4 99.7 107.2 105.9 ...
##  $ awayscorewin : num  102 107 104 105 104 ...
##  $ awayscorelost: num  109 112 101 108 113 ...
##  $ confrate     : num  0.209 0.359 0.674 0.425 0.45 ...
##  $ numberdayoff : num  2 3 3 1 2 2 1 2 2 2 ...
##  $ lastgame     : Factor w/ 2 levels "Away","Home": 1 1 1 2 1 1 2 2 2 1 ...
##  $ totalmatch   : int  67 67 67 67 66 66 67 67 66 67 ...
##  $ confmatch    : int  43 39 43 40 40 41 45 45 45 41 ...
##  $ homematch    : int  35 34 35 36 33 33 36 36 35 33 ...
##  $ awaymatch    : int  32 33 32 31 33 33 31 31 31 34 ...
##  $ l1           : num  1 1 1 2 2 1 1 2 2 1 ...
##  $ l2           : num  1 2 2 1 1 1 2 2 1 1 ...
##  $ l3           : num  1 1 2 1 2 2 2 1 1 2 ...
##  $ l4           : num  2 1 1 1 1 2 1 1 1 2 ...
##  $ l5           : num  1 1 2 1 2 1 1 2 1 2 ...
##  $ l6           : num  2 1 2 1 1 1 1 2 2 2 ...
##  $ l7           : num  1 2 2 2 1 2 2 1 1 2 ...
##  $ l8           : num  1 1 2 2 1 1 1 1 1 2 ...
##  $ l9           : num  1 1 1 2 1 2 1 2 1 2 ...
##  $ l10          : num  1 1 1 2 1 1 1 2 2 1 ...
```

Columns' meaning:

- "Teamname": Name of team
- "Confname": East or West, which defines which conference the team belongs to.
- "Homerate": winrate at home
- "Awayrate": winrate away
- "Last10": winrate in the last 10 matches
- "Homescorewin": score won when at home
- "Homescorelost": score lost when at home
- "Awayscorewin": score win when playing away home
- "Awayscorelost": score lost when to play away home
- "Confrate": winrate when playing within the conference
- "Numberdayoff": number of days since the last match
- "Lastgame": is the last game played at home or away
- "totalmatch":number of matches played
- "confmatch":number of matches played within conference
- "Homematch": number of matches played at home
- "Awaymatch": number of matches played away
- "l1" ~ "l10": is the last 1~10 win or lose

Initial table_b:

```
str(data_B[,c(1,2,3,4,5,56)])
```

```
## 'data.frame':    231 obs. of  6 variables:
##  $ gmDate     : Factor w/ 168 levels "2017-10-17","2017-10-18",..: 139 139 139 139 140 140 140 140 14
##  $ HomeTeam(A): Factor w/ 30 levels "ATL","BKN","BOS",..: 11 15 21 25 23 30 1 2 20 5 ...
##  $ AwayTeam(B): Factor w/ 30 levels "ATL","BKN","BOS",..: 26 17 27 16 12 18 21 28 7 13 ...
##  $ SameConf   : logi  TRUE FALSE TRUE FALSE TRUE FALSE ...
##  $ HistData   : num  0.458 0.583 0.773 0.583 0.238 ...
##  $ result     : logi  NA NA NA NA NA NA ...
```

Columns' meaning:

- "Hometeam(A)": The Abbreviation of team A (played as home team)
- "Awayteam(B)": The Abbreviation of team B (played as away team)
- "SameConf": True if two teams are in the same conference, false if not
- "HistData": The historical rate for team A played with team B
- "result": The match result based on our simulation
- The rest of columns are the data that is contained in table_a describing conditions for both team A and team B

Generalized Linear Regression

NBA is the place "where amazing happens", so instead of directly predicting match results by our model, we decided to predict the win probability for home team in each team and make simulation to take "amazing things" into consideration.

Therefore, we treated the match result between home team A and away team B as a bernoulli distribution with probability $P(Y_{AB}|X_{AB})$, where the win probability of home team A is $P(Y_{AB}|X_{AB})$ and the win probability for away team B is $(1 - P(Y_{AB}|X_{AB}))$.

We use `glm()` function to predict our win probability, whose model is:

$$P(Y_{AB}|X_{AB}) = \frac{e^{\pi(X_{AB})}}{1 + e^{\pi(X_{AB})}} \text{ where } \pi(X_{AB}) = \beta_0 + \beta_1 x_{AB1} + \ldots + \beta_n x_{ABn}$$

$x_{AB1}, x_{AB2}, \ldots, x_{ABn}$ are predictor variables related to team A and team B.

When selecting predictors of our models, we think that the most significant factors that may affect match include the short-term performance and long-term performance for both teams in this season, and the historical performance between these two teams. Therefore, our results include predictors `homeTeamRate`, `awayTeamRate`, `homeTeamlast10`, `awayTeamlast10` and `histRate`. (The meaning of each predictors are defined preciously in **Processing Data** part).

The summary is showed below:

```
##
## Call:
## glm(formula = Result ~ homeTeamRate + awayTeamRate + homeTeamlast10 +
##     awayTeamlast10 + histRate, family = binomial, data = data_fit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5115  -0.8021   0.0000   0.8021   2.5115
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.0514     0.3999  -2.629  0.00856 **
## homeTeamRate    -0.5599     0.5245  -1.068  0.28571
## awayTeamRate     0.5599     0.5245   1.068  0.28571
## homeTeamlast10   4.7007     0.4448  10.568  < 2e-16 ***
## awayTeamlast10  -4.7007     0.4448 -10.568  < 2e-16 ***
## histRate         2.1027     0.4145   5.072 3.93e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1890.9  on 1363  degrees of freedom
```

```
## Residual deviance: 1359.4  on 1358  degrees of freedom
## AIC: 1371.4
##
## Number of Fisher Scoring iterations: 5
```

Hence our generalized model is:

$$p(Y_{AB}) = \frac{e^{-1.0513702+(-0.5599274X_{AB1})+(0.5599274X_{AB2})+(4.7006912X_{AB3})+(-4.7006912X_{AB4})+(2.1027405X_{AB5})}}{1+e^{-1.0513702+(-0.5599274X_{AB1})+(0.5599274X_{AB2})+(4.7006912X_{AB3})+(-4.7006912X_{AB4})+(2.1027405X_{AB5})}}$$

where X_{AB1} is homeTeamRate, X_{AB2} is awayTeamRate, X_{AB3} is homeTeamlast10, X_{AB4} is awayTeamlast10 and X_{AB5} is histRate.

Though all predictors seem significant(p-value < 0.05) in our generalized linear model, we still need to check if collinearity exists between some predictors. So we turn to permutation test for checking.

Permutation Test

The generalized linear model we used to predict the game result includes predictors that might be correlated to each other. For example, `homeTeamRate` and `homeTeamlast10` seem to be positively correlated in common sense. Therefore, to improve our regression fit, we want to thorougly examine the correlations among the variables and proceed to use ridge regression if the variables are confirmed to be correlated. The variables for testing are,

- homeTeamRate
- awayTeamRate
- homeTeamlast10
- awayTeamlast10
- histRate

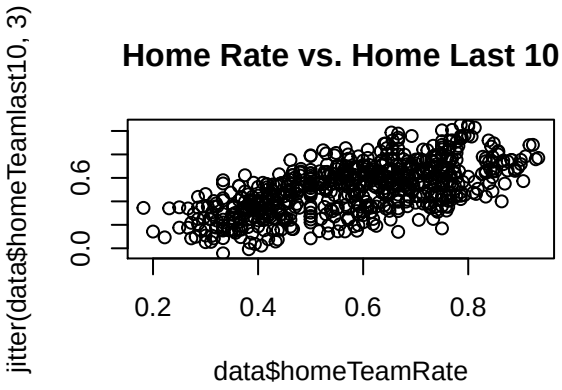
A permutation test essentially checks if X and Y have the same distribution by doing the following procedure,

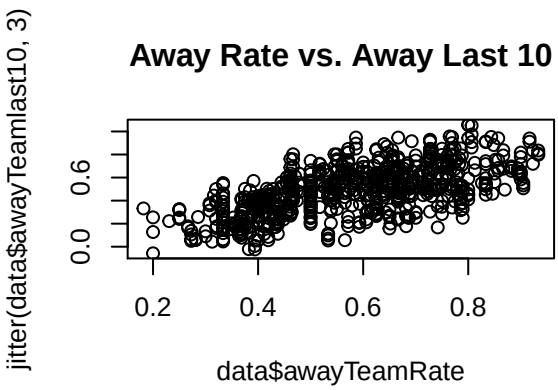
1. Observe a test statistic for the null hypothesis, H_0
2. For each replicated $b = 1, 2, ..., B$:
   - Generate a random permutation π
   - Generate a new test statistic from the random permutation
3. Get a Monte Carlo estimate of the p-value by calculating the probability of obtaining a new test statistic that is more extreme than the observed test statistic in the B replicates
4. Reject H_0 at a significance level α if the p-value is less than α.

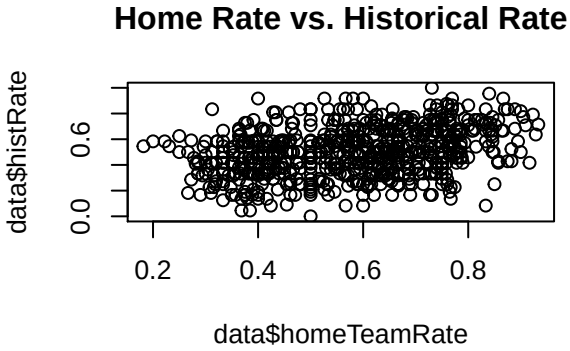
In our case, we can inherit the idea of permutation test and adapt it to paired data to check the correlation in between. If there is truly no association between X and Y, the distribution of $(X_i, Y_{\pi(i)})$ will be the same as that of (X_i, Y_i), where $\pi(i)$ is the i-th element of a permutation π of $1, 2, ..., 13$. We implement this idea by randomly permuting Y and pairing it with a fixed X and get a p-value for testing $H_0 : \rho = 0$ versus $H_1 : \rho \neq 0$, where ρ is either the Pearson correlation coefficient or the Spearman correlation coefficient:

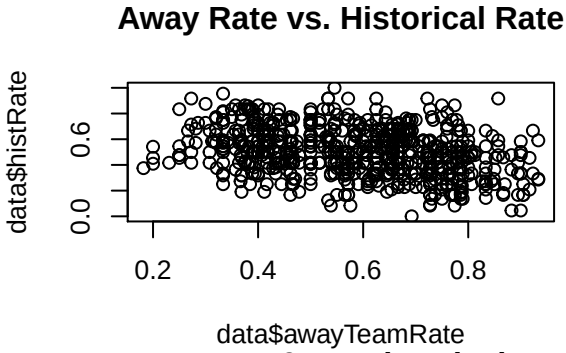
- The *Pearson Method* evaluates the **linear** relationship between two continuous variables, where a change in one variable is associated with a **proportional** change in the other variable.
- The *Spearman Method* is based on the ranked values for each variable rather than the raw data. It evaluates the **monotonic** relationship between two continuous or ordinal variables, where the variables tend to change together, but not necessarily at a constant rate. It is more general than the Pearson method.

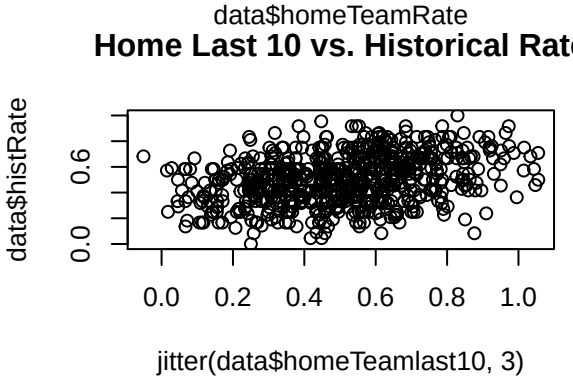
To determine which correlation coefficient to use for which pair of comparison, we plot variables against each another and added some noise using `jitter()` to visualize the trend of relationship and decide which method to use.

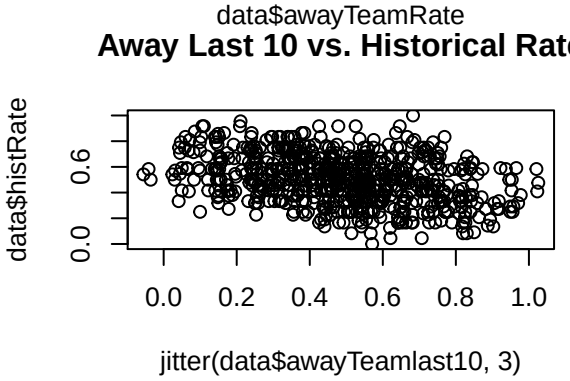

Home Rate vs. Home Last 10
jitter(data$homeTeamlast10, 3)
data$homeTeamRate
0.0
0.6
0.2
0.4
0.6
0.8

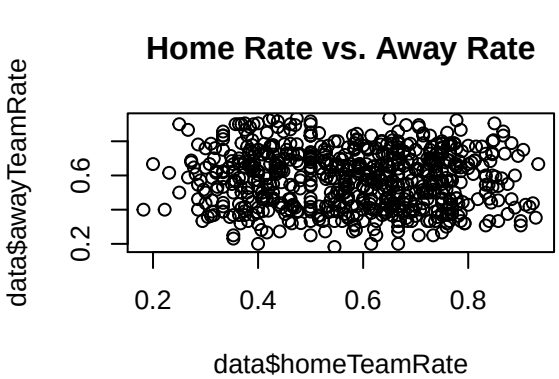

Away Rate vs. Away Last 10
jitter(data$awayTeamlast10, 3)
data$awayTeamRate
0.0
0.6
0.2
0.4
0.6
0.8

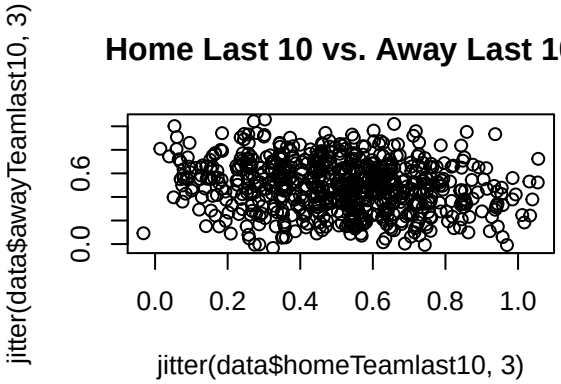

Home Rate vs. Historical Rate
data$histRate
data$homeTeamRate
0.0
0.6
0.2
0.4
0.6
0.8

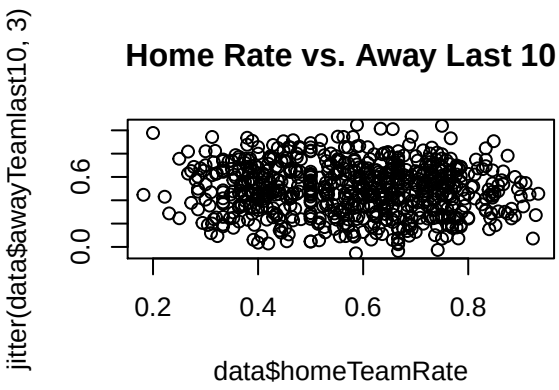

Away Rate vs. Historical Rate
data$histRate
data$awayTeamRate
0.0
0.6
0.2
0.4
0.6
0.8

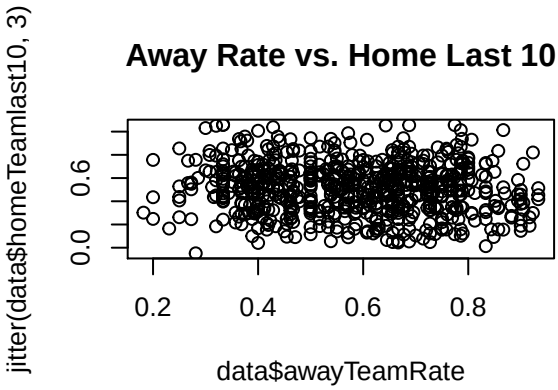

Home Last 10 vs. Historical Rate
data$histRate
jitter(data$homeTeamlast10, 3)
0.0
0.6
0.0
0.2
0.4
0.6
0.8
1.0


Away Last 10 vs. Historical Rate
data$histRate
jitter(data$awayTeamlast10, 3)
0.0
0.6
0.0
0.2
0.4
0.6
0.8
1.0


Home Rate vs. Away Rate
data$awayTeamRate
data$homeTeamRate
0.2
0.6
0.2
0.4
0.6
0.8


Home Last 10 vs. Away Last 10
jitter(data$awayTeamlast10, 3)
jitter(data$homeTeamlast10, 3)
0.0
0.6
0.0
0.2
0.4
0.6
0.8
1.0





According to the plots, *Home Rate vs. Home Last 10*, *Away Rate vs. Away Last 10*, *Home Last 10 vs. Historical Rate*, and *Away Last 10 vs. Historical Rate* appear to be linear. We use the Pearson method for these pairs. For the rest pairs, trends are not that obvious. Therefore, we use the more genearl Spearman method.

By performing permutation tests, we obtained the following table, which shows all the pairs we tested, the method we used, the p-value for the permutation test, and the decisions based on the p-values.

Pairs	Method	p-value	Decision
Home Rate vs. Home Last 10	Pearson	9.999e-05	Correlated
Away Rate vs. Away Last 10	Pearson	9.999e-05	Correlated
Home Rate vs. Historical Rate	Spearman	9.999e-05	Correlated
Away Rate vs. Historical Rate	Spearman	9.999e-05	Correlated
Home Last 10 vs. Historical Rate	Pearson	9.999e-05	Correlated
Away Last 10 vs. Historical Rate	Pearson	9.999e-05	Correlated
Home Rate vs. Away Rate	Spearman	0.4131587	Not Correlated
Home Last 10 vs. Away Last 10	Spearman	9.999e-05	Correlated
Home Rate vs. Away Last 10	Spearman	0.02689731	Correlated
Away Rate vs. Home Last 10	Spearman	0.02329767	Correlated

We found that all pairs of variables have correlation except *Home Rate vs. Away Rate*. This observation suggests that there exists problem of collinearity among the predictors. It leads us to further examine the VIF (Variance Inflation Factor) of the predictors and fit a Ridge Regression model to remedy the problem of collinearity.

Ridge Regression

Simulation Process

We repeatedly predicted playoff teams 1000 times by simulating match results with the ridge regression model above. Given table A and table B data, we could write our simulation process algorithm as below:

1. Create two 1000×8 empty data frame, `sim_record_East` and `sim_record_West` to store top 8 eastern team and western team names for 1000 simulations. Also, create a 30×10 data frame `sim_record_number`, whose column names are `"teamname", "confname", "1st", "2nd", "3rd", "4th", "5th", "6th", "7th", "8th"`, to accumulately record how many times each team enters the top 8 after 1000 simulations. And columns from `1st` to `8th` are all initialized with zeros.
2. For each game match (each row is one game match) in table B:
   - Update standing data (columns from `confname_A` to `l10_B` in game match row) from the corresponding rows in table A.

- Use updated game match data and our ridge regression model to predict the home team win probability. Using our ridge regression model, we could get `predictStat`, and `threshold = exp(predictStat) / (1 + exp(predictStat))`. Then randomly generate one number from `runif(1)` and compare it with `threshold`. Home team wins if the number generated is less than `threshold`, and vice versa.
- Once we get the predicted match result, we use it to update the corresponding data in table A, including `homerate`, `awayrate`, `last10`, etc.

3. After predicting all game matches' results, we push the eastern and western top 8 team names into `sim_record_East` and `sim_record_West`, also we add these 16 teams' counters by one in `sim_record_number`.
4. Repeat step 2, 3 for 1000 times.
5. Finally, we could get `sim_record_number` after 1000 times simulation. And for our final prediction, we could count total scores for all teams. Let

$$Total\ score = Times\ of\ 1^{st} \times 8 + \ times\ of\ 2^{nd} \times 7 + ... + \ times\ of\ 8^{th} \times 1$$

. We could get the rankings in order of total scores from largest to smallest. And the top 8 western and eastern teams are what we finally want.

Code for making simulations can be found below in **Appendix**.

Results

Linear Regrssion

Permutation Test

Ridge Regression

Simulation

Discussion

The test for correlation is not very accurate because the "Last 10" data (i.e. `homeTeamlast10` and `awayTeamlast10`) are highly categorical (discrete) because based on their calculation criteria: the number of times the team wins, which is an integer, divided by 10. When the "Last 10" data is paired with the other continuous variables, such as `homeTeamRate` and `histRate`, the correlation between a continuous variable and a somewhat discrete variable cannot be simply determined by correlation coefficients.

Appendix

Scatterplots for Correlation

```
data = read.csv("data_fit.csv")

par(mfrow = c(2,2))
plot(data$homeTeamRate, jitter(data$homeTeamlast10, 3), main = "Home Rate vs. Home Last 10")
plot(data$awayTeamRate, jitter(data$awayTeamlast10, 3), main = "Away Rate vs. Away Last 10")
plot(data$homeTeamRate, data$histRate, main = "Home Rate vs. Historical Rate")
plot(data$awayTeamRate, data$histRate, main = "Away Rate vs. Historical Rate")
par(mfrow = c(2,2))
plot(jitter(data$homeTeamlast10, 3), data$histRate, main = "Home Last 10 vs. Historical Rate")
plot(jitter(data$awayTeamlast10, 3), data$histRate, main = "Away Last 10 vs. Historical Rate")
plot(data$homeTeamRate, data$awayTeamRate, main = "Home Rate vs. Away Rate")
plot(jitter(data$homeTeamlast10, 3), jitter(data$awayTeamlast10, 3),
     main = "Home Last 10 vs. Away Last 10")
par(mfrow = c(2,2))
plot(data$homeTeamRate, jitter(data$awayTeamlast10, 3), main = "Home Rate vs. Away Last 10")
plot(data$awayTeamRate, jitter(data$homeTeamlast10, 3), main = "Away Rate vs. Home Last 10")
```

Permutation Test with either correlation coefficient

```
perm_test = function(X, Y, B = 10000, method = "pearson") {
  nu = seq_along(X)
  reps = numeric(B)

  if (method == "pearson") { # Pearson Method - default
    rho0 = abs(cor(X, Y))
    for ( i in 1:B ) {
      perm = sample(nu, size = length(X), replace = FALSE)
      X1 = X[perm]
      reps[i] = abs(cor(X1, Y))
    }
    pval = mean(c(rho0, reps) >= rho0)
  } else if (method == "spearman") { # Spearman Method
    rho0 = cor(X, Y, method = "spearman")
    t0 = abs(rho0*sqrt((length(X) - 2)/(1 - rho0^2)))
    for ( i in 1:B ) {
      perm = sample(nu, size = length(X), replace = FALSE)
      X1 = X[perm]
      rho = cor(X1, Y, method = "spearman")
      reps[i] = abs(rho*sqrt((length(X) - 2)/(1 - rho^2)))
    }
    pval = mean(c(t0, reps) >= t0)
  }

  return(pval)
}

# running the tests
perm_test(data$homeTeamRate, data$homeTeamlast10, 10000, "pearson")
```

```r
perm_test(data$awayTeamRate, data$awayTeamlast10, 10000, "pearson")
perm_test(data$homeTeamRate, data$histRate, 10000, "spearman")
perm_test(data$awayTeamRate, data$histRate, 10000, "spearman")
perm_test(data$homeTeamlast10, data$histRate, 10000, "spearman")
perm_test(data$awayTeamlast10, data$histRate, 10000, "spearman")
perm_test(data$homeTeamRate, data$awayTeamRate, 10000, "spearman")
perm_test(data$homeTeamlast10, data$awayTeamlast10, 10000, "spearman")
perm_test(data$homeTeamRate, data$awayTeamlast10, 10000, "spearman")
perm_test(data$awayTeamRate, data$homeTeamlast10, 10000, "spearman")
```

Simulations

```r
totalNRows = nrow(data_B)
set.seed(665267179)

sim_record_East = as.data.frame(matrix(NA, 1000, 8))
colnames(sim_record_East) = c("1st", "2nd", "3rd", "4th", "5th", "6th", "7th", "8th")
sim_record_West = as.data.frame(matrix(NA, 1000, 8))
colnames(sim_record_West) = c("1st", "2nd", "3rd", "4th", "5th", "6th", "7th", "8th")
sim_record_number = as.data.frame(matrix(0, 30, 10))
colnames(sim_record_number) = c("teamname", "confname",
                                "1st", "2nd", "3rd", "4th",
                                "5th", "6th", "7th", "8th")
sim_record_number$teamname = team_level
sim_record_number$confname = data_A$confname

# simulating
sim_time = 1000
for (i in 1:sim_time) {
  data_A_copy = data_A
  data_B_copy = data_B
  #one-time simulation
  for (nrowdata_B_copy in 1:totalNRows) {
    homeTeam = data_B_copy[nrowdata_B_copy, ]$`HomeTeam(A)`
    awayTeam = data_B_copy[nrowdata_B_copy, ]$`AwayTeam(B)`

    homeTeamRow = which(data_A_copy$teamname == homeTeam)
    awayTeamRow = which(data_A_copy$teamname == awayTeam)

    homeTeamdata_A_copy = data_A_copy[homeTeamRow, 2:26]
    awayTeamdata_A_copy = data_A_copy[awayTeamRow, 2:26]

    data_B_copy[nrowdata_B_copy, ][6:30] = data_A_copy[homeTeamRow, ][2:26]
    data_B_copy[nrowdata_B_copy, ][31:55] = data_A_copy[awayTeamRow, ][2:26]

    homeTeamRate = homeTeamdata_A_copy[2][[1]]
    awayTeamRate = awayTeamdata_A_copy[3][[1]]
    homeTeamlast10 = homeTeamdata_A_copy[4][[1]]
    awayTeamlast10 = awayTeamdata_A_copy[4][[1]]
    histRate = data_B_copy[nrowdata_B_copy, ][5][[1]]

    predictorDataFrame = data.frame(homeTeamRate = homeTeamRate,
```

```
                                    awayTeamRate = awayTeamRate,
                                    homeTeamlast10 = homeTeamlast10,
                                    awayTeamlast10 = awayTeamlast10,
                                    histRate = histRate)

  #predictStat = predict(glm1, predictorDataFrame)
  #ridge regression
  predictStat = predict.glmnet(rr, as.matrix(predictorDataFrame))
  [which(cv.out$lambda == cv.out$lambda.min)]

  threshold = exp(predictStat) / (1 + exp(predictStat))
  if (runif(1) < threshold) {
    result = 1
  } else {
    result = 0
  }

  data_B_copy[nrowdata_B_copy, ][56] = result

  homeRate = data_A_copy[homeTeamRow, ][3][[1]]
  homeMatch = data_A_copy[homeTeamRow, ][15][[1]]
  homeWinMatch = round(homeRate * homeMatch, digits=0)
  data_A_copy[homeTeamRow, ][15] = homeMatch + 1
  data_A_copy[homeTeamRow, ][3] = (homeWinMatch + result) / (homeMatch + 1)
  data_A_copy[homeTeamRow, ][18:26] = data_A_copy[homeTeamRow, ][17:25]
  data_A_copy[homeTeamRow, ][17] = data_B_copy[nrowdata_B_copy, ][56] + 1
  data_A_copy$last10[homeTeamRow] = sum(data_A_copy[homeTeamRow, 17:26] - 1) / 10


  awayRate = data_A_copy[awayTeamRow, ][4][[1]]
  awayMatch = data_A_copy[awayTeamRow, ][16][[1]]
  awayWinMatch = round(awayRate * awayMatch, digits=0)
  data_A_copy[awayTeamRow, ][16] = awayMatch + 1
  data_A_copy[awayTeamRow, ][4] = (awayWinMatch + (1 - result)) / (awayMatch + 1)
  data_A_copy[awayTeamRow, ][18:26] = data_A_copy[awayTeamRow, ][17:25]
  if (data_B_copy[nrowdata_B_copy, ][56] == 0) {
    data_A_copy[awayTeamRow, ][17] = 2
  } else {
    data_A_copy[awayTeamRow, ][17] = 1
  }
  data_A_copy$last10[awayTeamRow] = sum(data_A_copy[awayTeamRow, 17:26] - 1) / 10
  data_A_copy$totalmatch[homeTeamRow] = data_A_copy$totalmatch[homeTeamRow] + 1
  data_A_copy$totalmatch[awayTeamRow] = data_A_copy$totalmatch[awayTeamRow] + 1
}

# recording result
data_A_copy_East = data_A_copy[data_A_copy$confname == "East",]
data_A_copy_East = data_A_copy_East[order(
  -data_A_copy_East$homerate*data_A_copy_East$homematch
  -data_A_copy_East$awayrate*data_A_copy_East$awaymatch),]
sim_record_East[i, ] = data_A_copy_East$teamname[1:8]

data_A_copy_West = data_A_copy[data_A_copy$confname == "West",]
```

```
  data_A_copy_West = data_A_copy_West[order(
    -data_A_copy_West$homerate*data_A_copy_West$homematch
    -data_A_copy_West$awayrate*data_A_copy_West$awaymatch),]
  sim_record_West[i, ] = data_A_copy_West$teamname[1:8]

  for (rank in 1:8) {
    sim_record_number[which(sim_record_number$teamname ==
                              data_A_copy_East$teamname[rank]), rank + 2] =
      sim_record_number[which(sim_record_number$teamname ==
                                data_A_copy_East$teamname[rank]), rank + 2] + 1
    sim_record_number[which(sim_record_number$teamname ==
                              data_A_copy_West$teamname[rank]), rank + 2] =
      sim_record_number[which(sim_record_number$teamname ==
                                data_A_copy_West$teamname[rank]), rank + 2] + 1
  }
}

sim_record_number$total = rowSums(sim_record_number[,3:10])
sim_record_number$weightedtotal = sim_record_number$`1st`*8 +
  sim_record_number$`2nd`*7 + sim_record_number$`3rd`*6 +
  sim_record_number$`4th`*5 + sim_record_number$`5th`*4 +
  sim_record_number$`6th`*3 + sim_record_number$`7th`*2 +
  sim_record_number$`8th`
sim_record_number = sim_record_number[order(sim_record_number$confname,
                                            -sim_record_number$weightedtotal),]
```