Effectiveness of Tanking in the NBA: A 30-Year Study

Michael O’Donnell

City University of New York, School of Professional Studies

DATA621, Business Analytics and Data Mining

December 13, 2020

# Abstract

In 1966, the National Basketball Association (NBA) installed a reverse order draft; the order of draft picks was the reverse of teams’ regular season records. Inadvertently, this created an incentive for teams to lose (Becker and Huselid, 1992). In many cases, the incentive to lose was exposed by midseason teams without playoff potential. As an early example, the 1983-84 Houston Rockets decided to play more bench players after a disappointing 20-26 start to the season (Hallisey, 2016). They ended that season at 29-53 and drafted Hakeem Olajuwon 1st overall the next year.

Since 1966, the NBA draft was reformed but kept a structure that rewarded losing teams with higher draft picks. Thus, the term “tanking” was coined for teams that aimed for losing (Paxton, 2019). This paper acknowledges tanking exists, but it challenges its effectiveness. More specifically, it seeks to determine if tanking helps teams reach the NBA Finals.

# Problem Statement

Tanking in the NBA is prevalent. It is even sometimes transparent, like the 76ers “Trust the Process” years under General Manager Sam Hinkie (Choi, 2019). It is clear tanking guarantees better draft picks. But does it guarantee success?

To examine the relationship between tanking and success, this paper will analyze 30 consecutive NBA seasons from 1990-2020. In this analysis, success is defined as an NBA Finals appearance and tanking is defined as multiple losing seasons. The “length of a tanking” is defined as the number of consecutive losing seasons and “years since tanking” is defined as the number of seasons since a losing season. Overall, this paper will answer:

Does tanking help NBA Teams reach the Finals?

# Literature Review

## Incentive to Lose

Many different models are used in sports to determine success (Becker and Huselid, 1922). Some models reward only winning, while others reward winning and losing. In the NBA, both winning and losing are rewarded (Taylor and Trogdon, 2002); winning is rewarded with championship titles and losing is rewarded with better draft picks. Thus, NBA teams have both incentives to win or lose, but no incentive for a 50% winning percentage (Thomas, 2020).

## History of Tanking in the NBA

In 1966, the NBA installed a reverse order draft; the order of draft picks was the reverse of teams’ regular season records. Inadvertently, this created an incentive for teams to lose (Becker and Huselid, 1992). In many cases, the incentive to lose was exposed by midseason teams without playoff potential. As an early example, the 1983-84 Houston Rockets decided to play more bench players after a disappointing 20-26 start to the season (Hallisey, 2016). They ended that season at 29-53 and drafted Hakeem Olajuwon 1st overall the next year.

Since 1966, the NBA draft was reformed but kept a structure that rewarded losing teams with higher draft picks. Thus, the term “tanking” was coined for teams that aimed for losing (Paxton, 2019)

# Methodology

As stated above, this project’s research focused on answering the following question:

Does tanking help NBA Teams reach the Finals?

To answer this question, a five-step methodology was used:

1. **Data Acquisition**: The required data to answer this research question was NBA team regular season results and NBA team playoff results. Thus, all data was acquired from the NBA statistics website, basketball-reference.com. Since the data lived on multiple pages, a python web scraper was built to acquire the required 30 years of NBA data into one dataset.
2. **Data Preparation**: The data scraped from basketball-reference.com contained all NBA team regular season results and playoff results. But, the data did not contain information about tanking. To add this data, a python script was built to determine the “Years Since Tanking” and “Length of Tanking” for each row in the dataset.
3. **Data Exploration**: Each variable in the complete NBA dataset was explored for data type, correlation, and distribution. Then, the three most important variables were closely explored: “Consecutive Years in Playoffs”, “Years Since Tanking”, and “Length of Tanking”.
4. **Binary Logistic Regression Model**: After the data was prepared and explored, a Binary Logistic Regression model was created in R with a binary response variable, “NBA Finals Appearance”.
5. **Odds Ratio and Standardized Regression Coefficients**: After the Binary Logistic Regression Model was setup, the Odds Ratio and Standardized Regression Coefficient of each predictor variable were calculated in R to determine the most important predictors for an NBA Finals Appearance.

# Experimentation and Results

Question: Does tanking help NBA Teams reach the Finals?

## Data Acquisition

To acquire 30 seasons of NBA team data from basketball-reference.com, a python web scraper was built with the beautifulsoup library. The web scraper went to 30 web pages, which each contained 1 year of NBA Standings, and grabbed the NBA Teams’ regular season records. Then, all 30 years of standings were combined into one pandas dataframe and exported as a CSV. The details of the acquired data in the dataframe are below:

Table 1  
*NBA Seasons Collected by Team*

|  |  |  |
| --- | --- | --- |
| NBA Team | Seasons | Finals Appearances |
| Atlanta Hawks | 30 | 0 |
| Boston Celtics | 30 | 2 |
| Brooklyn Nets (NJ Nets) | 30 | 2 |
| Charlotte Bobcats | 10 | 0 |
| Charlotte Hornets | 20 | 0 |
| Chicago Bulls | 30 | 6 |
| Cleveland Cavaliers | 30 | 5 |
| Dallas Mavericks | 30 | 2 |
| Denver Nuggets | 30 | 0 |
| Detriot Pistons | 30 | 3 |
| Golden State Warriors | 30 | 5 |
| Houston Rockets | 30 | 2 |
| Indiana Pacers | 30 | 1 |
| Los Angeles Clippers | 30 | 0 |
| Los Angeles Lakers | 30 | 9 |
| Memphis Grizzlies | 19 | 0 |
| Miami Heat | 30 | 6 |
| Milwaukee Bucks | 30 | 0 |
| New Orleans Pelicans | 7 | 0 |
| New York Knicks | 30 | 2 |
| Oklahoma City Thunder (Seattle) | 30 | 2 |
| Orlando Magic | 30 | 2 |
| Philidelphia 76ers | 30 | 1 |
| Phoenix Suns | 30 | 1 |
| Portland Trail Blazers | 30 | 2 |
| Sacremento Kings | 30 | 0 |
| San Antonio Spurs | 30 | 6 |
| Toronto Raptors | 25 | 1 |
| Utah Jazz | 30 | 2 |
| Vancouver Grizzlies | 6 | 0 |
| Washington Bullets | 8 | 0 |
| Washington Wizards | 22 | 0 |

## Data Preparation

To isolate tanking as a predictor, two variables were created from the acquired data: “Years Since Tanking” and “Length of Tanking”. With these variables, the effect of tanking on NBA Finals appearance could be measured. To create these variables, a python script went through the dataset with a loop and calculated the “Years Since Tanking” and “Length of Tanking” for each row, which was 1 seasons for 1 team. An example of the results are shown below:

Table 2  
*Example of Dataset with Tanking Predictors*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Year | NBA Team | Finals Appearance | Consecutive Playoffs | | Years Since Tanking | Length of Tanking |
| 2020 | Atlanta Hawks | N | 0 | 0 | | 3 |
| 2020 | Boston Celtics | N | 6 | 6 | | 2 |
| 2020 | Brooklyn Nets | N | 2 | 0 | | 1 |
| 2020 | Charlotte Hornets | N | 0 | 0 | | 4 |
| 2020 | Chicago Bulls | N | 0 | 0 | | 3 |
| 2020 | Cleveland Cavaliers | N | 0 | 0 | | 2 |

## Data Exploration

Each variable in the complete NBA dataset was explored for data type, correlation, null variables, and distribution. Then, the three most important variables were closely explored: “Consecutive Years in Playoffs”, “Years Since Tanking”, and “Length of Tanking”.

Figure 1  
*Histogram of Important Predictors*

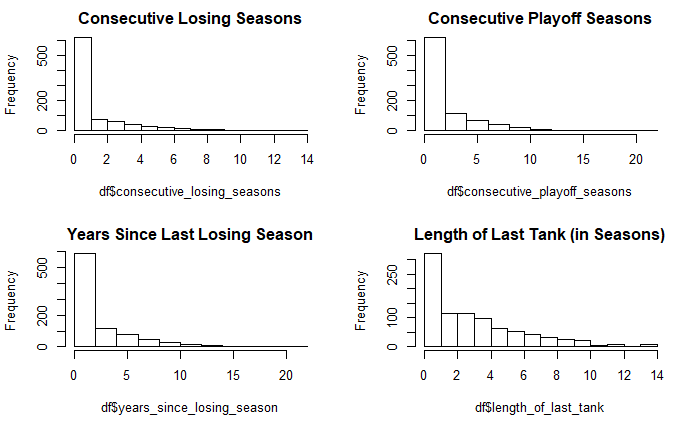
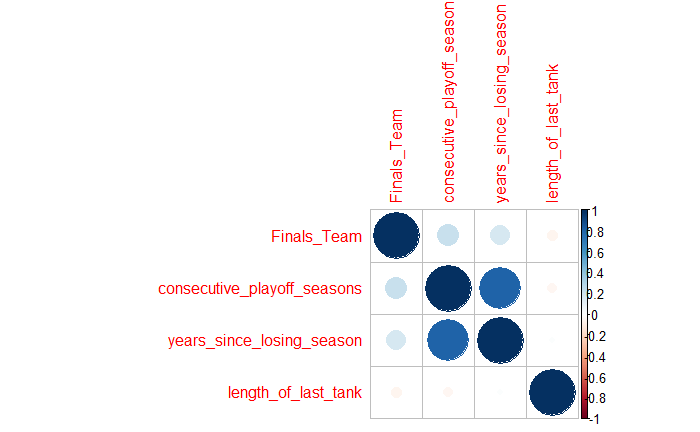


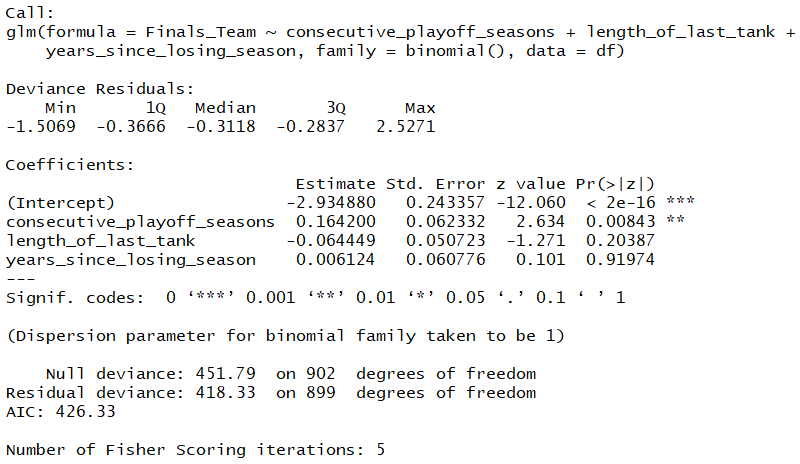
Figure 2  
*Correlation Plot of Important Predictors*



## Binary Logistic Regression Model

After the NBA data was prepared and explored a regression model was created. Since the data had a binary response and the effect of each predictor on the response was essential, a Binary Logistic Regression Model was appropriate. The Binary Logistic Regression Model was created in R with a logit link function (default). The summary of the model is below:

Figure 3  
*Summary of Binary Logistic Regression Model*



## Odds Ratio and Standardized Regression Coefficients

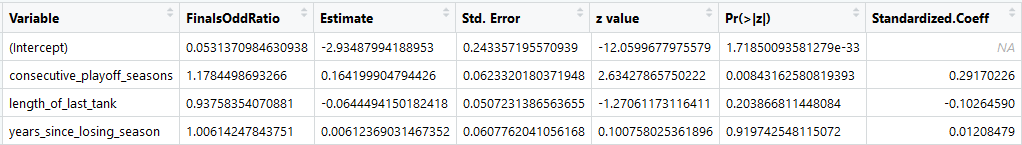
The Binary Logistic Regression Model was essential to the research, but not for prediction or accuracy. The model was important to assess the effectiveness of each predictor on the response. For example, it was important for the model to indicate if “Years Since Tanking” positively affected “Finals Appearance”. Thus, two variables were calculated for each predictor: Odds Ratio and Standardized Regression Coefficient.

The Odds Ratio is a measure of relationship between a predictor and binary response. If the Odds Ratio is positive, there is a positive relationship between the predictor and a positive response. For example, if a predictor has an Odds Ratio of 1.5, then for each unit the predictor increases the odds of the response being 1 is 50% greater.

The Standardized Regression Coefficient is a measure to rank the power of each predictor. The advantage of a Standardized Regression Coefficient over a Regression Coefficient is the units are standardized to best compare all predictors. For example, the predictors temperature in degrees and height in feet can be compared if the Regression Coefficients are Standardized.

With the Odds Ratio and Standardized Regression Coefficients calculated in R, the results were compared for each predictor in a dataframe:

Figure 4  
*Comparison of Results among Predictors*



# Conclusions

This research began with a question: Does tanking help NBA Teams reach the Finals? To answer the question, the Odds Ratio and Standardized Regression Coefficients were calculated from a Binary Logistic Regression Model for 2 variables for NBA teams over the last 30 seasons: “Years Since Tanking” and “Length of Tanking”.

For the results of the Odds Ratio, any number greater than 1.05 would indicate a statistically significant positive impact on the response (reaching the NBA Finals). Both “Years Since Tanking” and “Length of Tanking” had an Odds Ratio below 1.05.

For the results of the Standardized Regression Coefficient, any number greater than 0.05 would indicate a statistically significant positive impact on the response (reaching the NBA Finals). Both “Years Since Tanking” and “Length of Tanking” had a Standardized Regression Coefficient below 0.05.

Due to the results of the Odds Ratio and Standardized Regression Coefficient, neither predictors “Years Since Tanking” nor “Length of Tanking” had a positive effect on NBA Finals Appearance. Thus ultimately, Tanking does not help NBA Teams reach the Finals.

Table 3  
*Research Conclusions*

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Odds Ratio | Standardized Regression Coefficient | Conclusion |
| Years Since Tanking | 1.006 | 0.012 | No positive relationship on NBA Finals Appearance |
| Length of Tanking | 0.935 | -0.103 | No positive relationship on NBA Finals Appearance |

# References

Becker, B. E., &amp; Huselid, M. A. (1992). The Incentive Effects of Tournament Compensation Systems. Administrative Science Quarterly, 37(2), 336. doi:10.2307/2393228

Choi, M. S. (2019). Trust the Process: Tanking in the NBA. Wharton Research Scholars, (182).

Hallisey, R. P. (n.d.). Can NBA Teams Benefit from Losing? Honors Scholar Theses, (505).

Paxton, A. T. (2019). Trust the Process: How the NBA Can Combat Its "Tanking" Problem in Court. HeinOnline, 104.

Soebbing, B. P., &amp; Humphreys, B. R. (2011). Do Gamblers Think That Teams Tank? Evidence From The Nba. Contemporary Economic Policy, 31(2), 301-313. doi:10.1111/j.1465-7287.2011.00298.x

Taylor, B. A., &amp; Trogdon, J. G. (2002). Losing to Win: Tournament Incentives in the National Basketball Association. Journal of Labor Economics, 20(1), 23-41. doi:10.1086/323930

# Appendix with Code

## Data Acquisition

1. # -\*- coding: utf-8 -\*-
2. """
3. Created on Sun Nov 29 10:46:48 2020
5. @language: python
7. @author: Michael ODonnell
9. @title: scraping NBA team data
10. """
12. # import needed libraries
13. **from** urllib.request **import** urlopen
14. **from** bs4 **import** BeautifulSoup
15. **import** pandas as pd
17. # this function will scrape team performance by year for each specified year
18. **def** scrape\_NBA\_team\_data(years = [2017, 2018]):
20. # first, create empty dataframe with needed column headers
21. final\_df = pd.DataFrame(columns = ["Year", "Team", "W", "L",
22. "W/L%", "GB", "PS/G", "PA/G",
23. "SRS", "Playoffs",
24. "Losing\_season"])
26. # loop through each year, scraping team performance that year
27. **for** y **in** years:
28. # NBA season to scrape
29. year = y
31. # URL to scrape
32. url = f"https://www.basketball-reference.com/leagues/NBA\_{year}\_standings.html"
34. # HTML data collected
35. html = urlopen(url)
37. # create beautiful soup object from HTML
38. soup = BeautifulSoup(html, features="lxml")
40. # use getText()to extract the headers into a list
41. titles = [th.getText() **for** th **in** soup.findAll('tr', limit=2)[0].findAll('th')]
43. # first, find only column headers
44. headers = titles[1:titles.index("SRS")+1]
46. # then, update the titles list to exclude first set of column headers
47. titles = titles[titles.index("SRS")+1:]
49. # then, grab all row titles (ex: Boston Celtics, Toronto Raptors, etc)
50. **try**:
51. row\_titles = titles[0:titles.index("Eastern Conference")]
52. **except**: row\_titles = titles
53. # remove the non-teams from this list
54. **for** i **in** headers:
55. row\_titles.remove(i)
56. row\_titles.remove("Western Conference")
57. divisions = ["Atlantic Division", "Central Division",
58. "Southeast Division", "Northwest Division",
59. "Pacific Division", "Southwest Division",
60. "Midwest Division"]
61. **for** d **in** divisions:
62. **try**:
63. row\_titles.remove(d)
64. **except**:
65. **print**("no division:", d)
67. # next, grab all data from rows (avoid first row)
68. rows = soup.findAll('tr')[1:]
69. team\_stats = [[td.getText() **for** td **in** rows[i].findAll('td')]
70. **for** i **in** range(len(rows))]
71. # remove empty elements
72. team\_stats = [e **for** e **in** team\_stats **if** e != []]
73. # only keep needed rows
74. team\_stats = team\_stats[0:len(row\_titles)]
76. # add team name to each row in team\_stats
77. **for** i **in** range(0, len(team\_stats)):
78. team\_stats[i].insert(0, row\_titles[i])
79. team\_stats[i].insert(0, year)
81. # add team, year columns to headers
82. headers.insert(0, "Team")
83. headers.insert(0, "Year")
85. # create a dataframe with all aquired info
86. year\_standings = pd.DataFrame(team\_stats, columns = headers)
88. # add a column to dataframe to indicate playoff appearance
89. year\_standings["Playoffs"] = ["Y" **if** "\*" **in** ele **else** "N" **for** ele **in** year\_standings["Team"]]
90. # remove \* from team names
91. year\_standings["Team"] = [ele.replace('\*', '') **for** ele **in** year\_standings["Team"]]
92. # add a column to dataframe to indicate a losing season (win % < .5)
93. year\_standings["Losing\_season"] = ["Y" **if** float(ele) < .5 **else** "N" **for** ele **in** year\_standings["W/L%"]]
95. # append new dataframe to final\_df
96. final\_df = final\_df.append(year\_standings)
98. # print final\_df
99. **print**(final\_df.info)
100. # export to csv
101. final\_df.to\_csv("nba\_team\_data\_1990\_v2.csv", index=False)
103. # test on 2015 and 2016 because 2015 is old format and 2016 is new format
104. #scrape\_NBA\_team\_data(years = [2015,2016])
106. scrape\_NBA\_team\_data(years = [1990, 1991, 1992, 1993, 1994,
107. 1995, 1996, 1997, 1998, 1999,
108. 2000, 2001, 2002, 2003, 2004,
109. 2005, 2006, 2007, 2008, 2009,
110. 2010, 2011, 2012, 2013, 2014,
111. 2015, 2016, 2017, 2018, 2019,
112. 2020])
114. **def** NBA\_Final\_teams():
116. url = "https://www.basketball-reference.com/playoffs/"
118. # HTML data collected
119. html = urlopen(url)
121. # create beautiful soup object from HTML
122. soup = BeautifulSoup(html, features="lxml")
124. # use getText()to extract the headers into a list
125. finals\_titles = [th.getText() **for** th **in** soup.findAll('tr', limit=2)[1].findAll('th')]
127. # get rows from table
128. rows = soup.findAll('tr')[2:]
129. finals\_stats = [[td.getText() **for** td **in** rows[i].findAll('td')]
130. **for** i **in** range(len(rows))]
131. # pop the empty row
132. finals\_stats.pop(20)
133. finals\_stats = finals\_stats[0:38]
135. # add the years into finals\_stats
136. last\_year = 2020
137. **for** i **in** range(0, len(finals\_stats)):
138. finals\_stats[i].insert(0, last\_year)
139. last\_year -=1
141. # create the dataframe
142. nba\_finals = pd.DataFrame(finals\_stats, columns = finals\_titles)
144. nba\_finals.to\_csv("nba\_finals\_teams.csv", index=False)
146. #NBA\_Final\_teams()

## Data Preparation

1. # -\*- coding: utf-8 -\*-
2. """
3. Created on Sun Nov 29 15:41:27 2020
5. @author: ODsLaptop
7. @title: creating regression dataset
8. """
10. # import libraries
11. **import** pandas as pd
13. # loading nba team data
14. nba\_data = pd.read\_csv("https://raw.githubusercontent.com/odonnell31/NBA-Team-Strategies/main/data/nba\_team\_data\_1990\_v2.csv")
16. # create dataframe for only nba finals teams
17. nba\_finals\_teams = nba\_data[nba\_data['Finals\_Team'] == 'Y']
18. nba\_finals\_teams = nba\_finals\_teams[nba\_finals\_teams['Year'] > 1998]
19. nba\_finals\_teams = nba\_finals\_teams.reset\_index(drop=True)
21. # calc years since last losing season, and how many consecutibe losing season
22. years\_since\_losing = []
23. consecutive\_years\_losing = []
25. # iterate through all rows in dataframe
26. **for** i **in** range(len(nba\_finals\_teams)):
27. finals\_year = nba\_finals\_teams['Year'][i]
28. year\_prior = finals\_year - 1
29. team = nba\_finals\_teams['Team'][i]
30. last\_losing\_season = 0
31. consec\_losing\_seasons = 0
32. # flags
33. losing = 0
35. # if the last season was a losing season:
36. **if** nba\_data.loc[(nba\_data['Team'] == team) & (nba\_data['Year'] == year\_prior)]['Losing\_season'].values[0] == "Y":
37. last\_losing\_season = year\_prior
38. consec\_losing\_seasons += 1
39. losing = 1
40. year\_prior -= 1
42. # how many consecutive losing seasons were there?
43. **while** losing == 1:
44. **if** nba\_data.loc[((nba\_data['Team'] == team) & (nba\_data['Year'] == year\_prior))]['Losing\_season'].values[0] == 'Y':
45. consec\_losing\_seasons += 1
46. year\_prior -= 1
47. **else**:
48. losing = 0
50. # if the last season was a winning season
51. **elif** nba\_data.loc[((nba\_data['Team'] == team) & (nba\_data['Year'] == year\_prior))]['Losing\_season'].values[0] == 'N':
52. year\_prior -= 1
53. losing = 0
55. # work backwards to find the last losing season
56. **while** losing == 0:
58. # if it wasn't a losing season, keep looking
59. **if** nba\_data.loc[((nba\_data['Team'] == team) & (nba\_data['Year'] == year\_prior))]['Losing\_season'].values[0] == 'N':
60. year\_prior -= 1
61. losing = 0
63. # if it was a losing season, flag it
64. **else**:
65. last\_losing\_season = year\_prior
66. consec\_losing\_seasons += 1
67. year\_prior -= 1
68. losing = 1
70. # now, count the consecutive losing seasons
71. **while** losing == 1:
72. **if** nba\_data.loc[((nba\_data['Team'] == team) & (nba\_data['Year'] == year\_prior))]['Losing\_season'].values[0] == 'Y':
73. consec\_losing\_seasons += 1
74. year\_prior -= 1
75. losing = 1
77. # stop counting when they win again
78. **else**:
79. losing = 0

82. years\_since\_losing.append(finals\_year - last\_losing\_season)
83. consecutive\_years\_losing.append(consec\_losing\_seasons)
85. nba\_finals\_teams['years\_since\_losing\_season'] = years\_since\_losing
86. nba\_finals\_teams['consecutive\_years\_losing'] = consecutive\_years\_losing
88. # calc years of consecutive playoff appearances pre-finals appearance
89. consecutive\_playoff\_seasons = []
91. # iterate through all rows in dataframe
92. **for** i **in** range(len(nba\_finals\_teams)):
93. finals\_year = nba\_finals\_teams['Year'][i]
94. year\_prior = finals\_year - 1
95. team = nba\_finals\_teams['Team'][i]
96. consec\_playoff\_seasons = 0
97. # flags
98. playoffs = 0
100. # if the last season was a playoffs season:
101. **if** nba\_data.loc[(nba\_data['Team'] == team) & (nba\_data['Year'] == year\_prior)]['Playoffs'].values[0] == "Y":
102. consec\_playoff\_seasons += 1
103. year\_prior -= 1
104. playoffs = 1
106. # how many consecutive playoffs seasons were there?
107. **while** playoffs == 1:
108. **try**:
109. **if** nba\_data.loc[((nba\_data['Team'] == team) & (nba\_data['Year'] == year\_prior))]['Playoffs'].values[0] == 'Y':
110. consec\_playoff\_seasons += 1
111. year\_prior -= 1
112. **else**:
113. playoffs = 0
114. **except**:
115. **print**("ran out of seasons...")
116. playoffs = 0
118. **else**:
119. **print**("The", team, "did not make the playoffs prior to their",
120. finals\_year, "finals appearance")
122. consecutive\_playoff\_seasons.append(consec\_playoff\_seasons)
124. nba\_finals\_teams['consecutive\_playoff\_seasons'] = consecutive\_playoff\_seasons
126. nba\_finals\_teams.to\_csv("nba\_finals\_teams\_data\_1990\_v2.csv", index=False)

## Data Exploration

# language: R

# load required packages  
library(ggplot2)  
library(dplyr)

library(corrplot)

library(MASS)

library(caret)

library(haven)  
library(QuantPsyc)

# Loading the data  
git\_dir <- 'https://raw.githubusercontent.com/odonnell31/NBA-Team-Strategies/main/data'  
df = read.csv(paste(git\_dir, "/nba\_teams\_data\_1990\_2020.csv", sep=""))

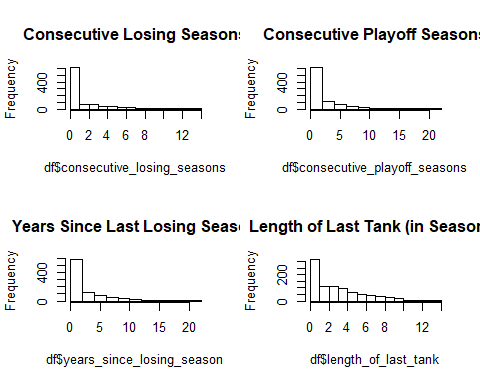
See a summary of each variable

summary(df)

## Year Team W L   
## Min. :1990 Atlanta Hawks : 31 Min. : 7.00 Min. : 9.00   
## 1st Qu.:1998 Boston Celtics : 31 1st Qu.:30.00 1st Qu.:30.00   
## Median :2005 Brooklyn Nets : 31 Median :41.00 Median :39.00   
## Mean :2005 Chicago Bulls : 31 Mean :40.03 Mean :40.03   
## 3rd Qu.:2013 Cleveland Cavaliers: 31 3rd Qu.:50.00 3rd Qu.:49.00   
## Max. :2020 Dallas Mavericks : 31 Max. :73.00 Max. :72.00   
## (Other) :717   
## W.L. PS.G PA.G SRS   
## Min. :0.1060 Min. : 81.9 Min. : 83.4 Min. :-14.680000   
## 1st Qu.:0.3780 1st Qu.: 95.8 1st Qu.: 95.9 1st Qu.: -3.175000   
## Median :0.5120 Median : 99.7 Median :100.2 Median : 0.170000   
## Mean :0.4998 Mean :100.4 Mean :100.4 Mean : -0.005637   
## 3rd Qu.:0.6200 3rd Qu.:104.3 3rd Qu.:104.7 3rd Qu.: 3.285000   
## Max. :0.8900 Max. :119.9 Max. :130.8 Max. : 11.800000   
##   
## Playoffs Losing\_season Finals\_Team   
## Min. :0.0000 Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000   
## Median :1.0000 Median :0.0000 Median :0.00000   
## Mean :0.5493 Mean :0.4374 Mean :0.06866   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.00000   
## Max. :1.0000 Max. :1.0000 Max. :1.00000   
##   
## consecutive\_losing\_seasons consecutive\_playoff\_seasons  
## Min. : 0.000 Min. : 0.00   
## 1st Qu.: 0.000 1st Qu.: 0.00   
## Median : 0.000 Median : 1.00   
## Mean : 1.493 Mean : 2.19   
## 3rd Qu.: 2.000 3rd Qu.: 3.00   
## Max. :14.000 Max. :22.00   
##   
## years\_since\_losing\_season length\_of\_last\_tank  
## Min. : 0.000 Min. : 0.000   
## 1st Qu.: 0.000 1st Qu.: 1.000   
## Median : 1.000 Median : 3.000   
## Mean : 2.533 Mean : 3.446   
## 3rd Qu.: 4.000 3rd Qu.: 5.000   
## Max. :22.000 Max. :14.000   
##

Look at histograms of important predictors

# setup 4 plots  
par(mfrow=c(2,2))  
  
# plot a histogram for each of the predictor variables  
hist(df$consecutive\_losing\_seasons, main = "Consecutive Losing Seasons")  
hist(df$consecutive\_playoff\_seasons, main = "Consecutive Playoff Seasons")  
hist(df$years\_since\_losing\_season, main = "Years Since Last Losing Season")  
hist(df$length\_of\_last\_tank, main = "Length of Last Tank (in Seasons)")



Subset the data for only possible predictors and response

keep\_vars <- c("Year", "Team", "Playoffs",  
 "Losing\_season", "Finals\_Team",  
 "consecutive\_losing\_seasons", "consecutive\_playoff\_seasons",  
 "years\_since\_losing\_season", "length\_of\_last\_tank")  
df <- df[keep\_vars]

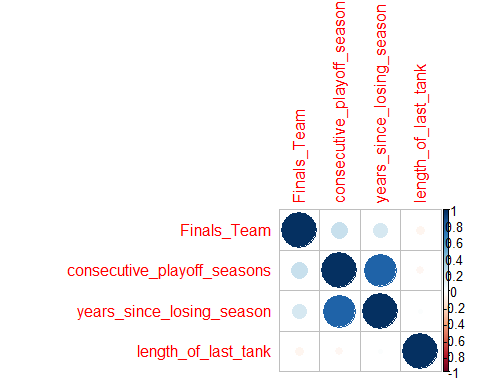
Check for NA’s

has\_NA = names(which(sapply(df, anyNA)))  
has\_NA

## character(0)

Explore correlations among important predictors

# look at correlations   
corr\_vars = c("Finals\_Team","consecutive\_playoff\_seasons",  
 "years\_since\_losing\_season", "length\_of\_last\_tank")  
cor\_train = cor(df[corr\_vars], use = "na.or.complete")  
corrplot(cor\_train)



## 

## Binary Logistic Regression Model

Create a binary logicstic regression model with Finals\_Team as the response

# create Binary Logistic Regression model  
finals\_logistic\_model <- glm(Finals\_Team ~ consecutive\_playoff\_seasons +  
 length\_of\_last\_tank +  
 years\_since\_losing\_season,  
 data = df, family = binomial())  
  
summary(finals\_logistic\_model)

##   
## Call:  
## glm(formula = Finals\_Team ~ consecutive\_playoff\_seasons + length\_of\_last\_tank +   
## years\_since\_losing\_season, family = binomial(), data = df)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.5069 -0.3666 -0.3118 -0.2837 2.5271   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.934880 0.243357 -12.060 < 2e-16 \*\*\*  
## consecutive\_playoff\_seasons 0.164200 0.062332 2.634 0.00843 \*\*   
## length\_of\_last\_tank -0.064449 0.050723 -1.271 0.20387   
## years\_since\_losing\_season 0.006124 0.060776 0.101 0.91974   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 451.79 on 902 degrees of freedom  
## Residual deviance: 418.33 on 899 degrees of freedom  
## AIC: 426.33  
##   
## Number of Fisher Scoring iterations: 5

## Odds Ratio and Standardized Regression Coefficients

Calculate Odd Ratios based on Regression Coefficients

#Logistic Regression Coefficient  
finals\_summary.coeff0 = summary(finals\_logistic\_model)$coefficient  
  
#Calculating Odd Ratios  
FinalsOddRatio = exp(coef(finals\_logistic\_model))  
finals\_summary.coeff = cbind(Variable = row.names(finals\_summary.coeff0), FinalsOddRatio, finals\_summary.coeff0)  
row.names(finals\_summary.coeff0) = NULL

Create a function to standardize the regression coefficients

# function to standardize regression coefficients  
standardize\_coefficients <- function (bl\_model)   
{ b <- summary(bl\_model)$coef[-1,1]  
 sx <- sapply(bl\_model$model[-1], sd)  
 beta <-(3^(1/2))/pi \* sx \* b  
 return(beta)  
}

Create standardized regression coefficients with new function

# use above function to standardize regression coefficients from model  
std\_Coeff = data.frame(Standardized.Coeff = standardize\_coefficients(finals\_logistic\_model))  
std\_Coeff = cbind(Variable = row.names(std\_Coeff), std\_Coeff)  
row.names(std\_Coeff) = NULL

Merge the Odds Ratios and Coefficients to see all results

#Final Summary Report  
final\_report = merge(finals\_summary.coeff, std\_Coeff, by = "Variable", all.x = TRUE)  
  
final\_report

## Variable FinalsOddRatio Estimate  
## 1 (Intercept) 0.0531370984630938 -2.93487994188953  
## 2 consecutive\_playoff\_seasons 1.1784498693266 0.164199904794426  
## 3 length\_of\_last\_tank 0.93758354070881 -0.0644494150182418  
## 4 years\_since\_losing\_season 1.00614247843751 0.00612369031467352  
## Std. Error z value Pr(>|z|)  
## 1 0.243357195570939 -12.0599677975579 1.71850093581279e-33  
## 2 0.0623320180371948 2.63427865750222 0.00843162580819393  
## 3 0.0507231386563655 -1.27061173116411 0.203866811448084  
## 4 0.0607762041056168 0.100758025361896 0.919742548115072  
## Standardized.Coeff  
## 1 NA  
## 2 0.29170226  
## 3 -0.10264590  
## 4 0.01208479