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 variabl es, 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

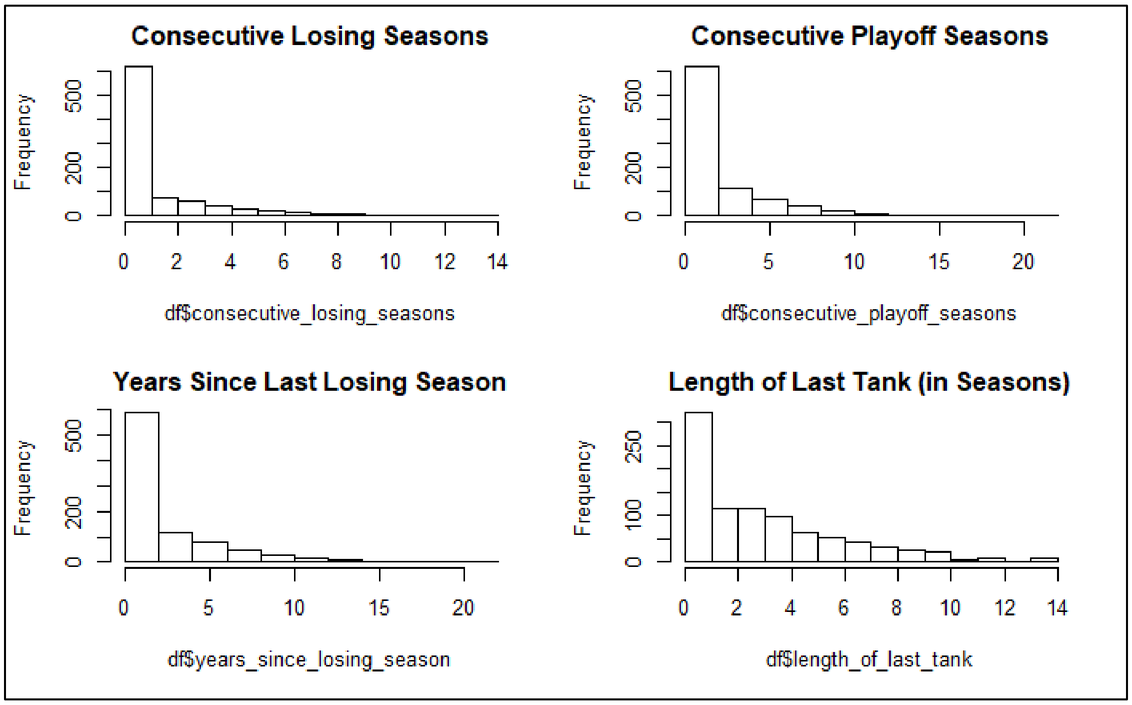
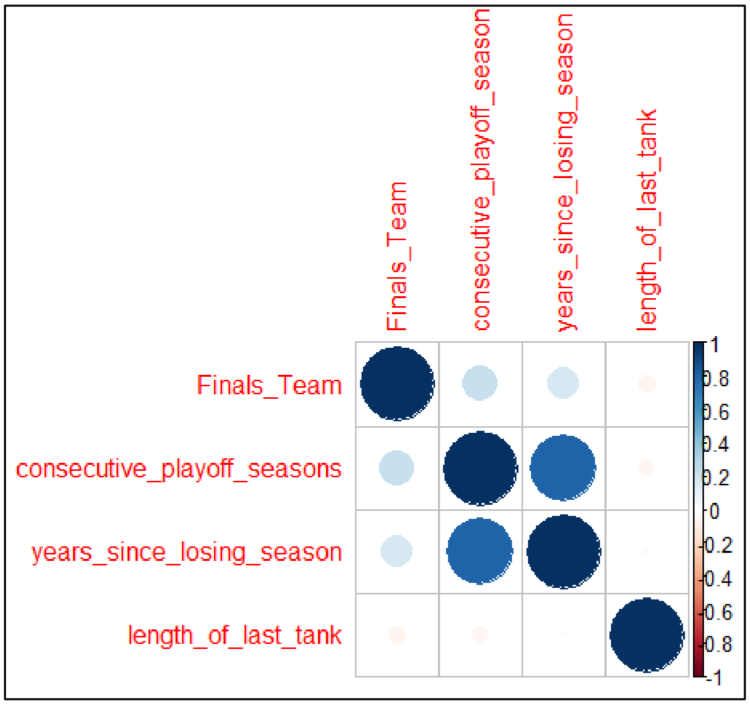


Figure 2

Correlation Plot of Important Predictors

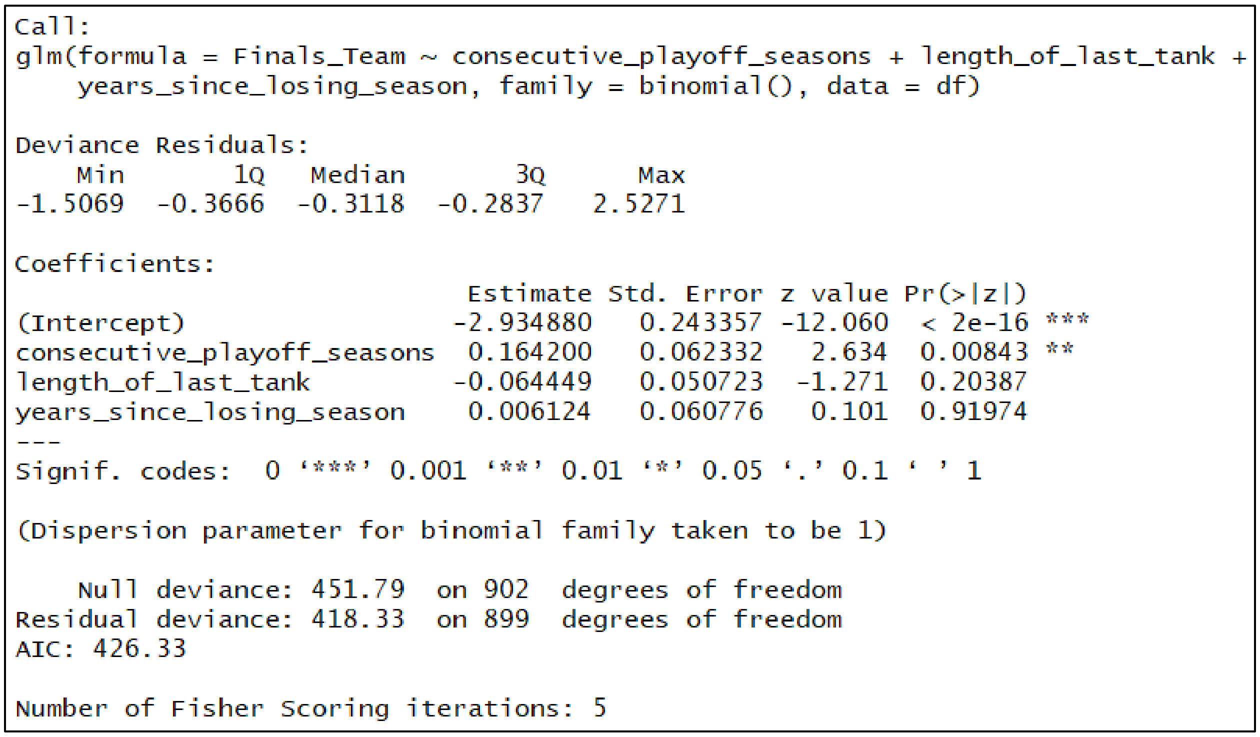


Binary Logistic Regression Model

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



```
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

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.

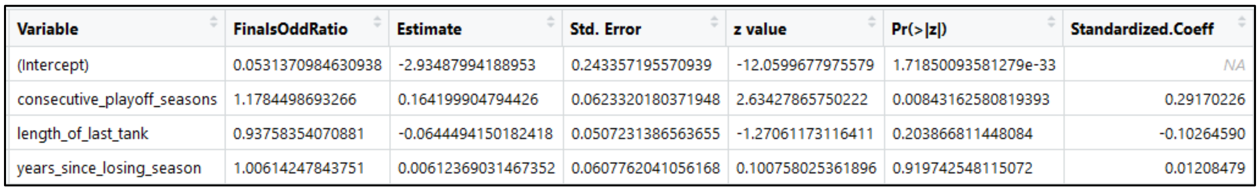
The Odds Ratio is a measure of relationship between a predictor and binary response. If the Odds Ratio is positive, there is a positive relationship between the predictor and a positive response. For example, if a predictor has an Odds Ratio of 1.5, then for each unit the predictor increases the odds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



Variable	FinalsOddRatio	Estimate	Std. Error	z value	Pr(>|z|)	Standardized.Coeff
(Intercept)	0.0531370984630938	-2.93487994188953	0.243357195570939	-12.0599677975579	1.71850093581279e-33	NA
consecutive_playoff_seasons	1.1784498693266	0.164199904794426	0.0623320180371948	2.63427865750222	0.00843162580819393	0.29170226
length_of_last_tank	0.93758354070881	-0.0644494150182418	0.0507231386563655	-1.27061173116411	0.203866811448084	-0.10264590
years_since_losing_season	1.00614247843751	0.00612369031467352	0.0607762041056168	0.100758025361896	0.919742548115072	0.01208479

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 <u>ultimately, Tanking does not help NBA Teams reach the Finals</u>.

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

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 29 10:46:48 2020

@language: python

@author: Michael ODonnell

@title: scraping NBA team data
"""

# import needed libraries
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd

# this function will scrape team performance by year for each specified year
def scrape_NBA_team_data(years = [2017, 2018]):

    # first, create empty dataframe with needed column headers
    final_df = pd.DataFrame(columns = ["Year", "Team", "W", "L",
                                       "W/L%", "GB", "PS/G", "PA/G",
                                       "SRS", "Playoffs",
                                       "Losing_season"])

    # loop through each year, scraping team performance that year
    for y in years:
        # NBA season to scrape
        year = y

        # URL to scrape
        url = f"https://www.basketball-reference.com/leagues/NBA_{year}_standings.html"

        # HTML data collected
        html = urlopen(url)

        # create beautiful soup object from HTML
        soup = BeautifulSoup(html, features="lxml")

        # use getText()to extract the headers into a list
        titles = [th.getText() for th in soup.findAll('tr', limit=2)[0].findAll('th')]

        # first, find only column headers
        headers = titles[1:titles.index("SRS")+1]

        # then, update the titles list to exclude first set of column headers
        titles = titles[titles.index("SRS")+1:]

        # then, grab all row titles (ex: Boston Celtics, Toronto Raptors, etc)
        try:
            row_titles = titles[0:titles.index("Eastern Conference")]
        except: row_titles = titles
        # remove the non-teams from this list
        for i in headers:
            row_titles.remove(i)
        row_titles.remove("Western Conference")
        divisions = ["Atlantic Division", "Central Division",
                     "Southeast Division", "Northwest Division",
                     "Pacific Division", "Southwest Division",
                     "Midwest Division"]
        for d in divisions:
            try:
                row_titles.remove(d)
            except:
                print("no division:", d)

        # next, grab all data from rows (avoid first row)
        rows = soup.findAll('tr')[1:]
        team_stats = [[td.getText() for td in rows[i].findAll('td')]
                    for i in range(len(rows))]
        # remove empty elements
        team_stats = [e for e in team_stats if e != []]
        # only keep needed rows
        team_stats = team_stats[0:len(row_titles)]

        # add team name to each row in team_stats
        for i in range(0, len(team_stats)):
            team_stats[i].insert(0, row_titles[i])
            team_stats[i].insert(0, year)

        # add team, year columns to headers
        headers.insert(0, "Team")
        headers.insert(0, "Year")

        # create a dataframe with all aquired info
        year_standings = pd.DataFrame(team_stats, columns = headers)

        # add a column to dataframe to indicate playoff appearance
        year_standings["Playoffs"] = ["Y" if "*" in ele else "N" for ele in year_standings["Team"]]
        # remove * from team names
        year_standings["Team"] = [ele.replace('*', '') for ele in year_standings["Team"]]
        # add a column to dataframe to indicate a losing season (win % < .5)
        year_standings["Losing_season"] = ["Y" if float(ele) < .5 else "N" for ele in year_standings["W/L%"]]

        # append new dataframe to final_df
        final_df = final_df.append(year_standings)

    # print final_df
    print(final_df.info)
    # export to csv
    final_df.to_csv("nba_team_data_1990_v2.csv", index=False)

# test on 2015 and 2016 because 2015 is old format and 2016 is new format
#scrape_NBA_team_data(years = [2015,2016])

scrape_NBA_team_data(years = [1990, 1991, 1992, 1993, 1994,
                              1995, 1996, 1997, 1998, 1999,
                              2000, 2001, 2002, 2003, 2004,
                              2005, 2006, 2007, 2008, 2009,
                              2010, 2011, 2012, 2013, 2014,
                              2015, 2016, 2017, 2018, 2019,
                              2020])

def NBA_Final_teams():

    url = "https://www.basketball-reference.com/playoffs/"

    # HTML data collected
    html = urlopen(url)

    # create beautiful soup object from HTML
    soup = BeautifulSoup(html, features="lxml")

    # use getText()to extract the headers into a list
    finals_titles = [th.getText() for th in soup.findAll('tr', limit=2)[1].findAll('th')]

    # get rows from table
    rows = soup.findAll('tr')[2:]
    finals_stats = [[td.getText() for td in rows[i].findAll('td')]
            for i in range(len(rows))]
    # pop the empty row
    finals_stats.pop(20)
    finals_stats = finals_stats[0:38]

    # add the years into finals_stats
    last_year = 2020
    for i in range(0, len(finals_stats)):
        finals_stats[i].insert(0, last_year)
        last_year -=1

    # create the dataframe
    nba_finals = pd.DataFrame(finals_stats, columns = finals_titles)

    nba_finals.to_csv("nba_finals_teams.csv", index=False)
```

```

        #NBA_Final_teams()
```

Data Preparation

```
# -*- coding: utf-8 -*-
"""
Created on Sun Nov 29 15:41:27 2020

@author: ODsLaptop

@title: creating regression dataset
"""

# import libraries
import pandas as pd

# loading nba team data
nba_data = pd.read_csv("https://raw.githubusercontent.com/odonnell31/NBA-Team-Strategies/main/data/nba_team_data_1990_v2.csv")

# create dataframe for only nba finals teams
nba_finals_teams = nba_data[nba_data['Finals_Team'] == 'Y']
nba_finals_teams = nba_finals_teams[nba_finals_teams['Year'] > 1998]
nba_finals_teams = nba_finals_teams.reset_index(drop=True)

# calc years since last losing season, and how many consecutibe losing season
years_since_losing = []
consecutive_years_losing = []

# iterate through all rows in dataframe
for i in range(len(nba_finals_teams)):
    finals_year = nba_finals_teams['Year'][i]
    year_prior = finals_year - 1
    team = nba_finals_teams['Team'][i]
    last_losing_season = 0
    consec_losing_seasons = 0
    # flags
    losing = 0

    # if the last season was a losing season:
    if nba_data.loc[(nba_data['Team'] == team) & (nba_data['Year'] == year_prior)]['Losing_season'].values[0] == "Y":
        last_losing_season = year_prior
        consec_losing_seasons += 1
        losing = 1
        year_prior -= 1

        # how many consecutive losing seasons were there?
        while losing == 1:
            if nba_data.loc[((nba_data['Team'] == team) & (nba_data['Year'] == year_prior))]['Losing_season'].values[0] == 'Y':
                consec_losing_seasons += 1
                year_prior -= 1
            else:
                losing = 0

    # if the last season was a winning season
    elif nba_data.loc[((nba_data['Team'] == team) & (nba_data['Year'] == year_prior))]['Losing_season'].values[0] == 'N':
        year_prior -= 1
        losing = 0

        # work backwards to find the last losing season
        while losing == 0:

            # if it wasn't a losing season, keep looking
            if nba_data.loc[((nba_data['Team'] == team) & (nba_data['Year'] == year_prior))]['Losing_season'].values[0] == 'N':
                year_prior -= 1
                losing = 0

            # if it was a losing season, flag it
            else:
                last_losing_season = year_prior
                consec_losing_seasons += 1
                year_prior -= 1
                losing = 1

        # now, count the consecutive losing seasons
        while losing == 1:
            if nba_data.loc[((nba_data['Team'] == team) & (nba_data['Year'] == year_prior))]['Losing_season'].values[0] == 'Y':
                consec_losing_seasons += 1
                year_prior -= 1
                losing = 1

            # stop counting when they win again
            else:
                losing = 0


    years_since_losing.append(finals_year - last_losing_season)
    consecutive_years_losing.append(consec_losing_seasons)

nba_finals_teams['years_since_losing_season'] = years_since_losing
nba_finals_teams['consecutive_years_losing'] = consecutive_years_losing

# calc years of consecutive playoff appearances pre-finals appearance
consecutive_playoff_seasons = []

# iterate through all rows in dataframe
for i in range(len(nba_finals_teams)):
    finals_year = nba_finals_teams['Year'][i]
    year_prior = finals_year - 1
    team = nba_finals_teams['Team'][i]
    consec_playoff_seasons = 0
    # flags
    playoffs = 0

        # if the last season was a playoffs season:
        if nba_data.loc[(nba_data['Team'] == team) & (nba_data['Year'] == year_prior)]['Playoffs'].values[0] == "Y":
            consec_playoff_seasons += 1
            year_prior -= 1
            playoffs = 1

            # how many consecutive playoffs seasons were there?
            while playoffs == 1:
                try:
                    if nba_data.loc[((nba_data['Team'] == team) & (nba_data['Year'] == year_prior))]['Playoffs'].values[0] == 'Y':
                        consec_playoff_seasons += 1
                        year_prior -= 1
                    else:
                        playoffs = 0
                except:
                    print("ran out of seasons...")
                    playoffs = 0

        else:
            print("The", team, "did not make the playoffs prior to their",
                  finals_year, "finals appearance")

        consecutive_playoff_seasons.append(consec_playoff_seasons)

    nba_finals_teams['consecutive_playoff_seasons'] = consecutive_playoff_seasons

    nba_finals_teams.to_csv("nba_finals_teams_data_1990_v2.csv", index=False)
```

Data Exploration

```r
# 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

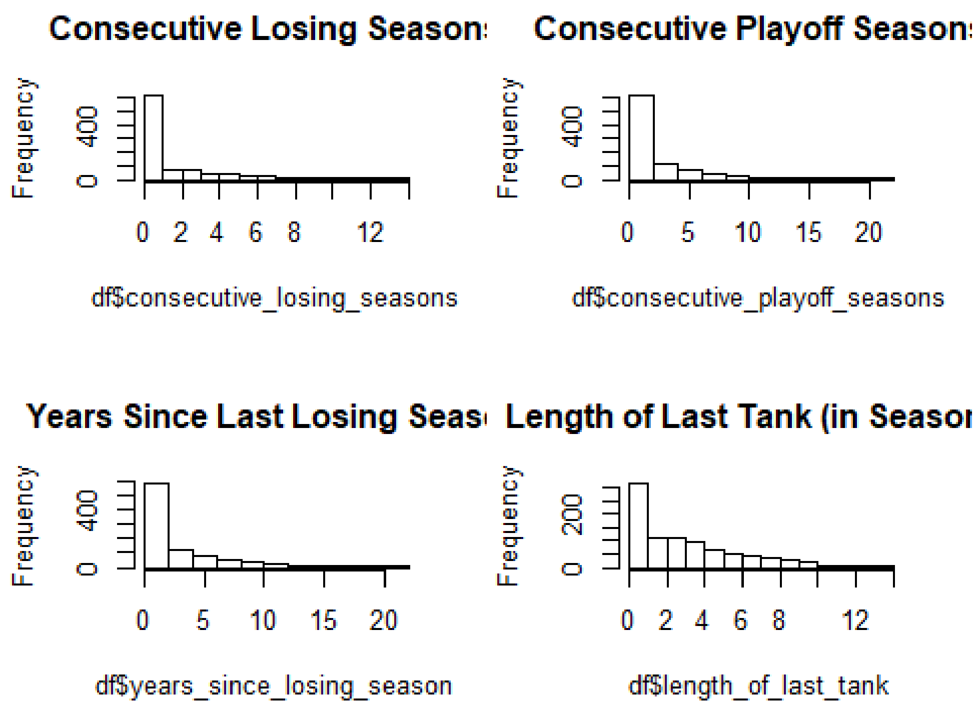
```r
summary(df)
```

```
##       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

```r
# 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]
```

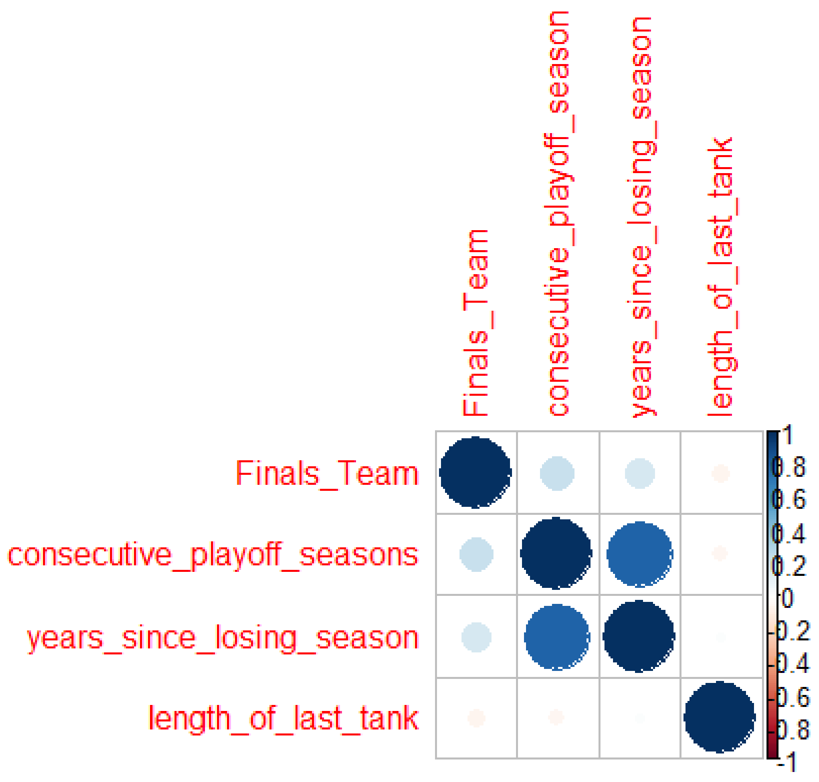
Check for NA’s

```
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_t
ank +
##     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), Fin
alsOddRatio, 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_l
ogistic_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
```