Joe QBS181 Project Code

Joe Gyorda

2022-11-03

Description of Code

This Markdown file contains the code for the analyses prepared by Joe Gyorda as part of a final project group project for QBS 181 at Dartmouth College, taught by Carly Bobak in the Fall of 2022. The project examined trends in NFL sports betting data over time, and in this Markdown, I attempted to develop visualizations and predictive models for the difference in the actual game scores of NFL games and the predicted score (called the spread). This file is outlined as follows:

Section 1. Visual analyses

Section 2. Linear mixed model

Section 3. Network model

Section 4. Scrap code for classification model

Section 1 - Visual analyses

```
setwd('/users/joegyorda/Desktop/wranglinghub')
football_data = read.csv('Merged_Stadium.csv')
```

```
# for now, we'll only focus on games where there was a spread

# what are the unique home teams?
sort(unique(football_data$team_home))
```

```
##  [1] "Arizona Cardinals"        "Atlanta Falcons"
##  [3] "Baltimore Colts"          "Baltimore Ravens"
##  [5] "Boston Patriots"          "Buffalo Bills"
##  [7] "Carolina Panthers"        "Chicago Bears"
##  [9] "Cincinnati Bengals"       "Cleveland Browns"
## [11] "Dallas Cowboys"           "Denver Broncos"
## [13] "Detroit Lions"            "Green Bay Packers"
## [15] "Houston Oilers"           "Houston Texans"
## [17] "Indianapolis Colts"       "Jacksonville Jaguars"
## [19] "Kansas City Chiefs"       "Las Vegas Raiders"
## [21] "Los Angeles Chargers"     "Los Angeles Raiders"
## [23] "Los Angeles Rams"         "Miami Dolphins"
## [25] "Minnesota Vikings"        "New England Patriots"
## [27] "New Orleans Saints"       "New York Giants"
## [29] "New York Jets"            "Oakland Raiders"
## [31] "Philadelphia Eagles"      "Phoenix Cardinals"
## [33] "Pittsburgh Steelers"      "San Diego Chargers"
## [35] "San Francisco 49ers"      "Seattle Seahawks"
## [37] "St. Louis Cardinals"      "St. Louis Rams"
## [39] "Tampa Bay Buccaneers"     "Tennessee Oilers"
## [41] "Tennessee Titans"         "Washington Commanders"
## [43] "Washington Football Team" "Washington Redskins"
```

```
# how many missing values
sum(is.na(football_data$spread_favorite))
```

```
## [1] 2735
```

```
# remove missing values! just remove all for now
# football_data_filter = football_data[complete.cases(football_data),]
football_data_filter = football_data %>% drop_na(spread_favorite)

# should be 0!
sum(is.na(football_data_filter$spread_favorite))
```

```
## [1] 0
```

This block uses dplyr to calculate, for each NFL team (when they were the team favored to win), how often the spread was correct, as well as how often the spread was larger/outperformed and smaller/underperformed. The resulting tibbles were combined into one, which was used to create a barplot with ggplot.

```
# how often is the spread correct (for each team)?
# comment out group_by for overall, otherwise gives each team's breakdown
football_data_filter %>%
  group_by(team_home) %>%
  summarise(Spread_Correct=sum(Actual.difference...spread==0)/
              length(Actual.difference...spread) * 100) %>%
  arrange(desc(Spread_Correct))
```

```
## # A tibble: 43 x 2
##    team_home                Spread_Correct
##    <chr>                             <dbl>
##  1 Washington Football Team           5.88
##  2 Baltimore Colts                    5.26
##  3 Baltimore Ravens                   4.61
##  4 Los Angeles Rams                   4.37
##  5 Tennessee Titans                   4.17
##  6 Cincinnati Bengals                 3.99
##  7 Tampa Bay Buccaneers               3.72
##  8 Buffalo Bills                      3.67
##  9 San Francisco 49ers                3.49
## 10 Denver Broncos                     3.30
## # ... with 33 more rows
```

```
# how often does favored team outperform spread (for each team)?
# comment out group_by for overall, otherwise gives each team's breakdown
```

```r
football_data_filter %>%
  group_by(team_home) %>%
  summarise(Over_Spread=sum(Actual.difference...spread>0)/
              length(Actual.difference...spread) * 100) %>%
  arrange(desc(Over_Spread))
```

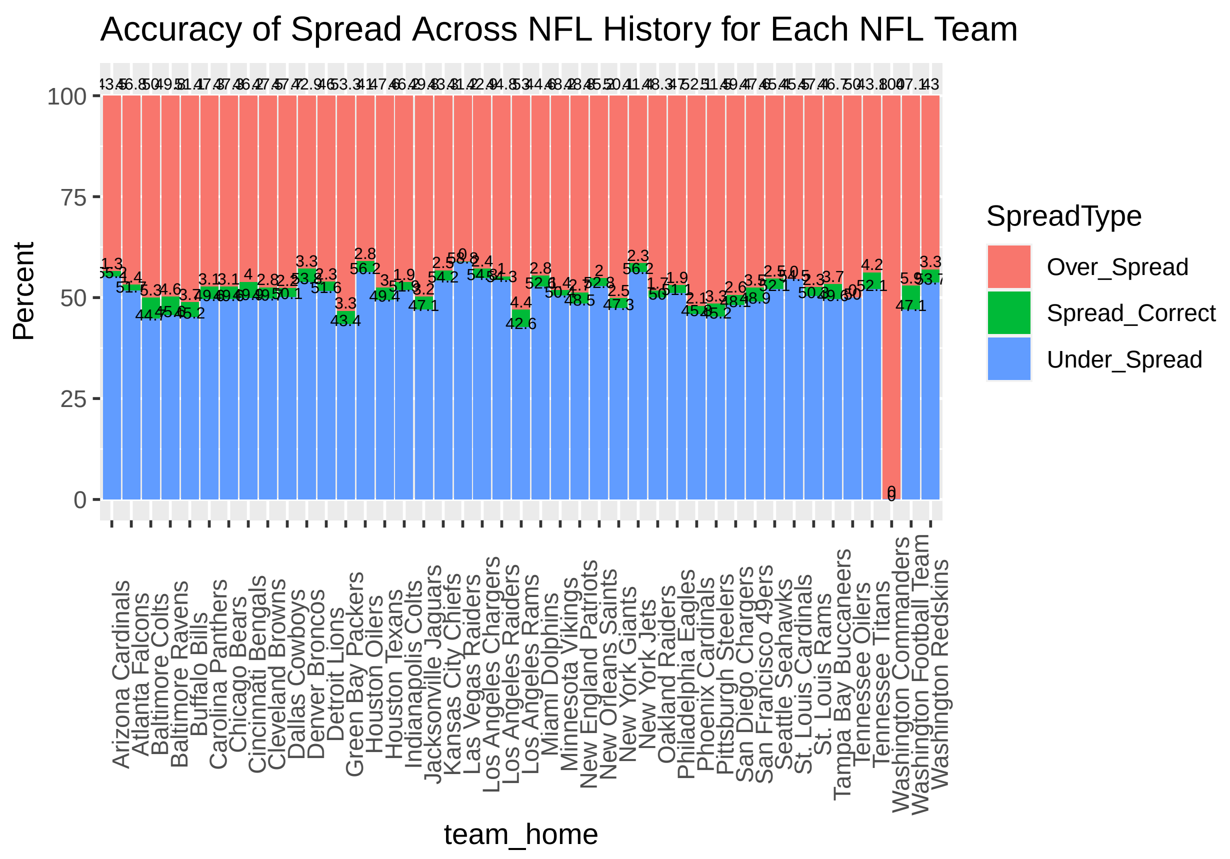
```
## # A tibble: 43 x 2
##    team_home             Over_Spread
##    <chr>                       <dbl>
##  1 Washington Commanders       100
##  2 Green Bay Packers            53.3
##  3 Los Angeles Rams             53.0
##  4 Phoenix Cardinals            52.1
##  5 Pittsburgh Steelers          51.5
##  6 Buffalo Bills                51.1
##  7 New York Giants              50.1
##  8 Baltimore Colts              50
##  9 Tennessee Oilers             50
## 10 Jacksonville Jaguars         49.8
## # ... with 33 more rows
```

```r
# how often does favored team underperform spread (for each team)?
# comment out group_by for overall, otherwise gives each team's breakdown
football_data_filter %>%
  group_by(team_home) %>%
  summarise(Under_Spread=sum(Actual.difference...spread<0)/
              length(Actual.difference...spread) * 100) %>%
  arrange(desc(Under_Spread))
```

```
## # A tibble: 43 x 2
##    team_home            Under_Spread
##    <chr>                       <dbl>
##  1 Las Vegas Raiders            58.8
##  2 Houston Oilers               56.2
##  3 New York Jets                56.2
##  4 Arizona Cardinals            55.2
##  5 Los Angeles Chargers         54.8
##  6 St. Louis Cardinals          54.5
##  7 Los Angeles Raiders          54.3
##  8 Kansas City Chiefs           54.2
##  9 Denver Broncos               53.8
## 10 Washington Redskins          53.7
## # ... with 33 more rows
```

```r
# combine all into 1
spread_breakdown = football_data_filter %>%
  group_by(team_home) %>%
  summarise(Over_Spread=sum(Actual.difference...spread>0)/
              length(Actual.difference...spread) * 100,
            Under_Spread=sum(Actual.difference...spread<0)/
              length(Actual.difference...spread) * 100,
            Spread_Correct=sum(Actual.difference...spread==0)/
```

```
              length(Actual.difference...spread) * 100) %>%
  gather(SpreadType, Percent, Over_Spread:Spread_Correct)

# uncomment png and dev.off() lines to save image to computer
# png(file="/users/joegyorda/Desktop/QBS181bars.png", width=4000, height=2000, res=350)
ggplot(data=spread_breakdown,aes(team_home,fill=SpreadType)) +
  geom_bar(aes(weight=Percent),position="stack") +
  theme(axis.text.x = element_text(angle = 90)) +
  ylab("Percent") +
  geom_text(position="stack",aes(team_home,Percent+1,label=round(Percent,1)),size=2) +
  ggtitle("Accuracy of Spread Across NFL History for Each NFL Team")
```



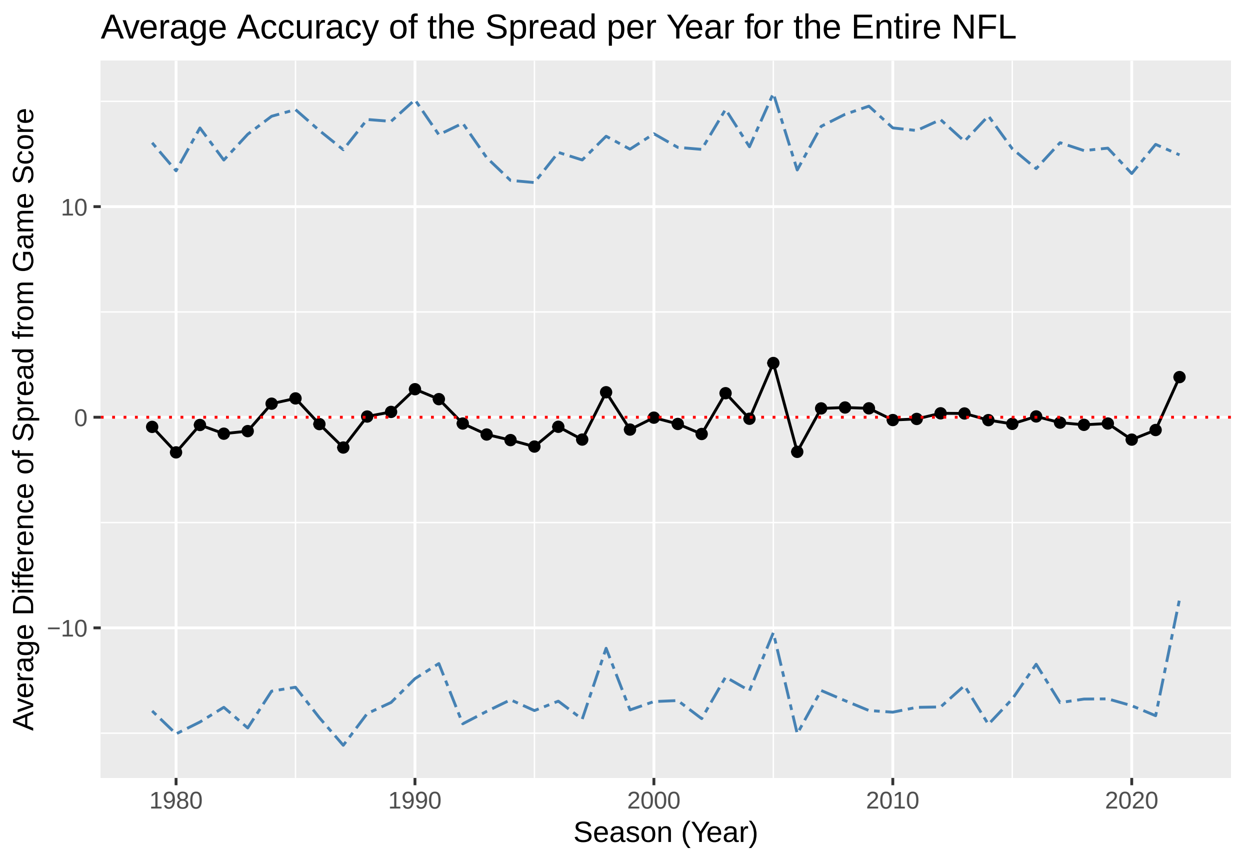
```
# dev.off()
```

Uses dplyr to create a new variable tracking the yearly average in difference between actual game score and predicted spread, and plots with ggplot along with standard deviation bars.

```
spread_score_diff_over_time = football_data_filter %>%
  filter(schedule_season > 1978) %>%  # only superbowl data before this
  group_by(schedule_season) %>%
  summarise(Avg_Diff=mean(Actual.difference...spread),SD_Diff=sd(Actual.difference...spread),
            Med_Diff=median(Actual.difference...spread))

spread_score_diff_over_time
```

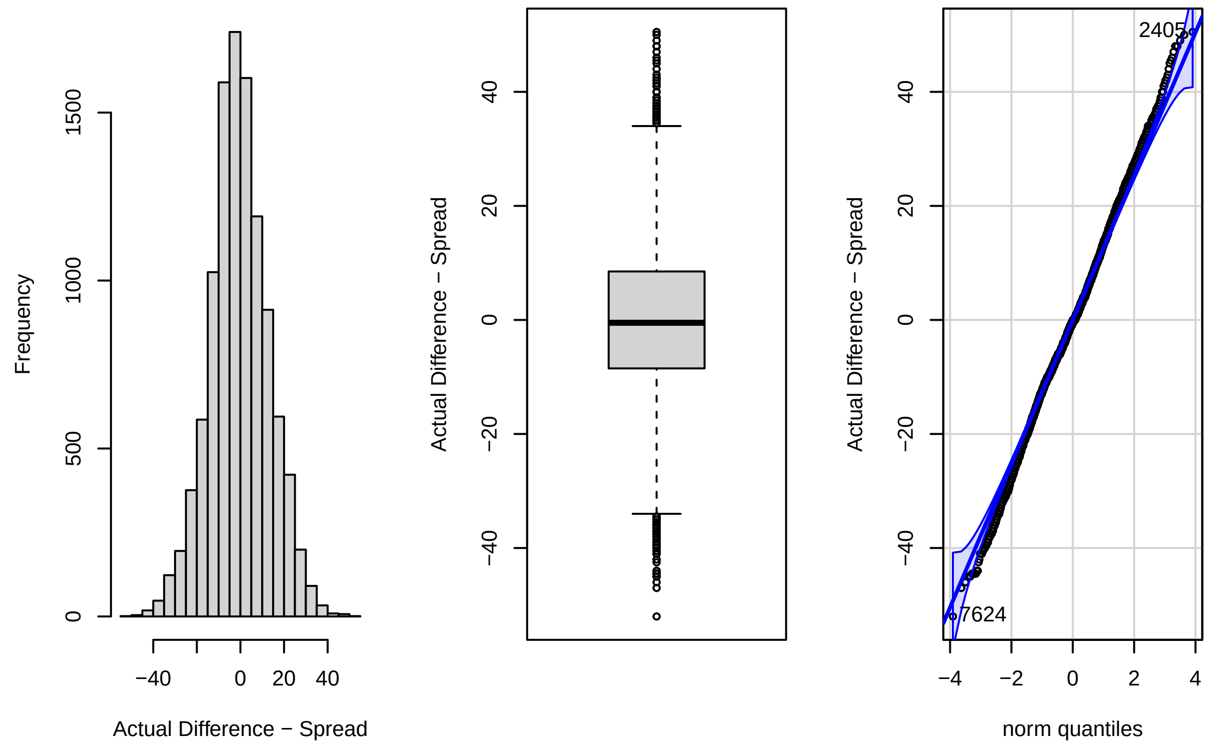
```
## # A tibble: 44 x 4
##    schedule_season Avg_Diff SD_Diff Med_Diff
##              <int>    <dbl>   <dbl>    <dbl>
##  1            1979  -0.457     13.5      0
##  2            1980  -1.67      13.4     -2
##  3            1981  -0.367     14.1     -1
##  4            1982  -0.780     13.0     -1.5
##  5            1983  -0.659     14.1     -1
##  6            1984   0.644     13.6      0
##  7            1985   0.895     13.7      1
##  8            1986  -0.326     13.9     -0.5
##  9            1987  -1.44      14.1     -1.5
## 10            1988   0.0343    14.1     -0.5
## # ... with 34 more rows
```

```
ggplot(data=spread_score_diff_over_time, aes(x=schedule_season, y=Avg_Diff)) +
  geom_line()+
  geom_point()+
  geom_hline(yintercept=0, linetype='dotted', col = 'red') +
  geom_line(aes(y = (Avg_Diff+SD_Diff)), color="steelblue", linetype="twodash") +
  geom_line(aes(y = (Avg_Diff-SD_Diff)), color="steelblue", linetype="twodash") +
  xlab("Season (Year)") + ylab("Average Difference of Spread from Game Score") +
  ggtitle("Average Accuracy of the Spread per Year for the Entire NFL")
```



Creates histogram, boxplot, and qqplot of the outcome (actual difference in game score minus predicted spread) to assess normality and check assumptions for regression.

```
# uncomment png and dev.off() lines to save image to computer
# png(file="/users/joegyorda/Desktop/QBS181plots.png", width=4000, height=2000, res=350)
par(mfrow=c(1,3))
hist(football_data_filter$Actual.difference...spread, xlab="Actual Difference - Spread", main="")
boxplot(football_data_filter$Actual.difference...spread, ylab="Actual Difference - Spread")
qqPlot(football_data_filter$Actual.difference...spread, ylab="Actual Difference - Spread")
```



```
## [1] 7624 2405
```

```
# dev.off()
# outcome looks normal!
```

Section 2 - Linear Mixed Model

Uses the lme4 R package to create a linear mixed model as outlined in the comments in the code block below. The model output and confidence intervals for the regression coefficients are shown.

```
# how important are weather, location, field type, etc to covering the spread? how do
# these predictors differ by team?

# the outcome variable is actual difference - spread
# this variable takes the difference b/w the real game score difference, and
# the predicted difference (spread)
```

```r
# positive value means favored team outperformed spread, negative means favored
# team underperformed the spread, and 0 means spread was correct

# only consider instances with full data in predictions
football_data_complete = football_data_filter[complete.cases(football_data_filter),]

# what's our sample size?

# create the model object using lmer from lme4 library in R
# includes random intercepts for schedule_season (the year) and team_favorite_id
#  (the favored team), as well as random slopes of time for each favored team
mod1 = lmer(`Actual.difference...spread`~ weather_temperature + weather_wind_mph +
              weather_humidity + weather_detail + stadium_type + stadium_weather_type
            + stadium_surface + as.numeric(ELEVATION)  +
              (1|schedule_season) + (schedule_season|team_favorite_id),
            data=football_data_complete)
```

```
## fixed-effect model matrix is rank deficient so dropping 1 column / coefficient

## boundary (singular) fit: see help('isSingular')
```

```r
# show model output and r squared
summary(mod1)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: Actual.difference...spread ~ weather_temperature + weather_wind_mph +
##     weather_humidity + weather_detail + stadium_type + stadium_weather_type +
##     stadium_surface + as.numeric(ELEVATION) + (1 | schedule_season) +
##     (schedule_season | team_favorite_id)
##    Data: football_data_complete
##
## REML criterion at convergence: 48733
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.9121 -0.6302 -0.0171  0.6345  3.8047
##
## Random effects:
##  Groups           Name            Variance  Std.Dev.  Corr
##  schedule_season  (Intercept)     2.430e-06  0.001559
##  team_favorite_id (Intercept)     1.987e+02 14.095487
##                   schedule_season 5.619e-05  0.007496 -1.00
##  Residual                         1.773e+02 13.317210
## Number of obs: 6080, groups:  schedule_season, 43; team_favorite_id, 33
##
## Fixed effects:
##                                     Estimate Std. Error t value
## (Intercept)                        1.7536307  1.2698812   1.381
## weather_temperature               -0.0141506  0.0127343  -1.111
## weather_wind_mph                  -0.0372379  0.0405288  -0.919
## weather_humidity                  -0.0172546  0.0116657  -1.479
## weather_detailDOME (Open Roof)    -5.6137183  4.9623026  -1.131
## weather_detailFog                      2.7236001  3.1642109   0.861
## weather_detailRain                     3.8775847  2.0833921   1.861
## weather_detailRain | Fog               5.4260126  2.9272936   1.854
## weather_detailSnow                     6.3069773  4.7361046   1.332
## weather_detailSnow | Fog              -2.5351765  6.6817324  -0.379
## weather_detailSnow | Freezing Rain -4.8610297 13.3488067  -0.364
## stadium_typeretractable               -0.9296774  1.5076016  -0.617
## stadium_weather_typemoderate           0.5327153  0.4993866   1.067
## stadium_weather_typewarm               0.4839171  0.6067082   0.798
## stadium_surfaceFieldTurf               0.9068016  0.6203781   1.462
## stadium_surfaceGrass                   0.3538343  0.4172298   0.848
## as.numeric(ELEVATION)                 -0.0003462  0.0006885  -0.503
```

```
##
## Correlation matrix not shown by default, as p = 17 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

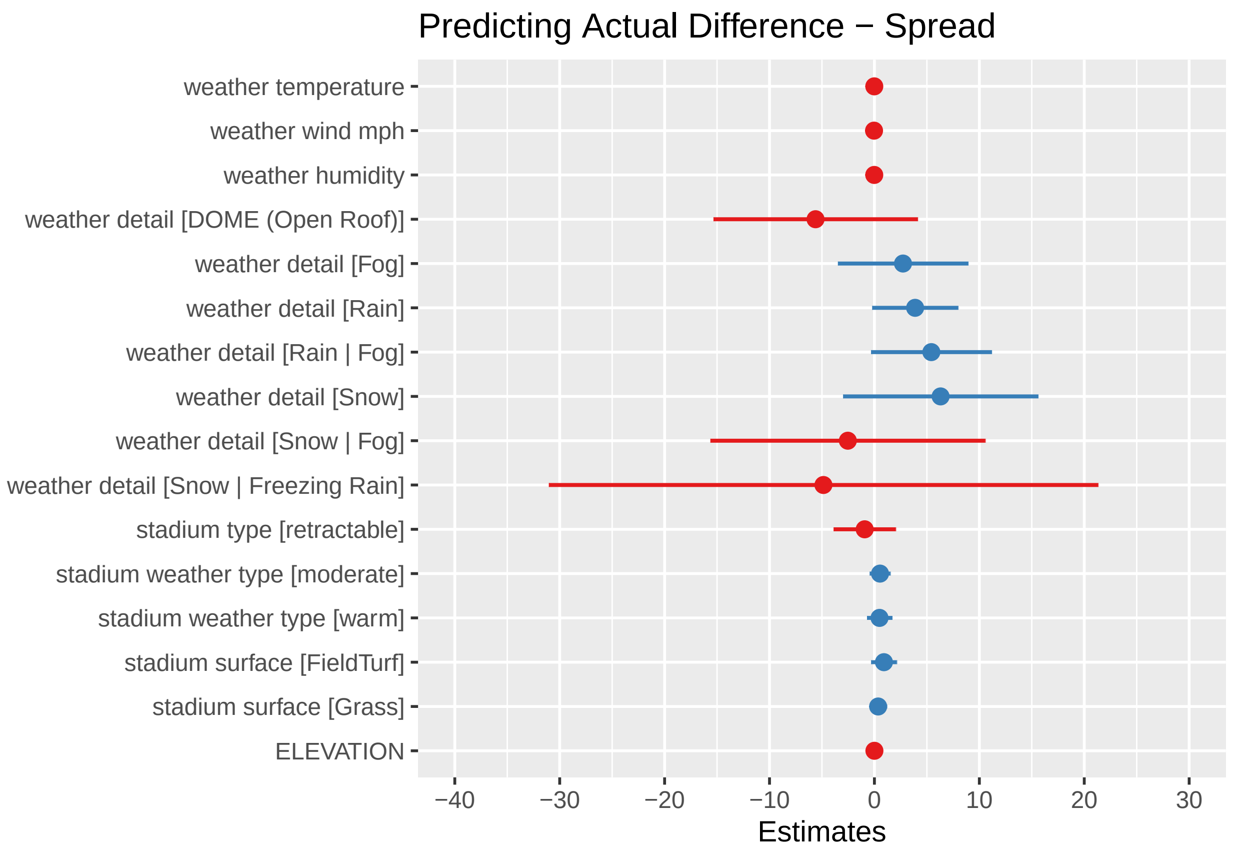
```
## fit warnings:
## fixed-effect model matrix is rank deficient so dropping 1 column / coefficient
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
r.squaredGLMM(mod1)
```

```
## Warning: 'r.squaredGLMM' now calculates a revised statistic. See the help page.
```

```
##              R2m         R2c
## [1,] 0.003640943 0.007933624
```

```
# confidence intervals for each predictor
sjPlot::plot_model(mod1, title="Predicting Actual Difference - Spread")
```



```
# sjPlot::tab_model(mod1)
# # sjPlot::plot_residuals(mod1)w
```

Section 3 - Network Analysis

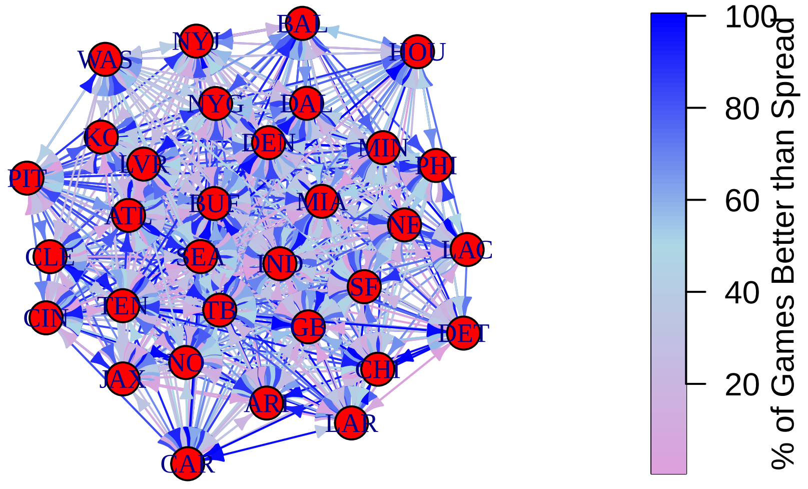
We use the igraph library in R to create a network of all 32 NFL teams to assess which teams are better at beating the spread against certain teams. This network may yield informative value by providing a cool visualization of which teams historically are the best to bet on (e.g., they are more likely to exceed the spread).

```
# create adjacency matrix, where rows are favored team, columns are unfavored team,
# entries are number of times the favored team i beat spread against unfavored team j
# this will be a non-symmetric matrix, as there will be matchups b/w 2 teams
# where one is favored and when the other is instead favored

# subset data we care about first
football_data_net = football_data_filter %>%
  select(schedule_season, Home.team.abbrev,
         Away.team.abbrev, team_favorite_id,
         Actual.difference...spread)

# add column for the nonfavored team too
football_data_net$team_notfavorite_id = ifelse(football_data_net$team_favorite_id !=
                                                  football_data_net$Home.team.abbrev,
                                                football_data_net$Home.team.abbrev,
```

```r
                                                    football_data_net$Away.team.abbrev)

# get list of unique teams - PICK when spread is 0, we drop these games
teams = sort(unique(football_data_net$team_favorite_id))
teams = teams[teams!="PICK"]

# initialize adjacency matric
teamMatrixSmall = matrix(0,nrow=32,ncol=32)
rownames(teamMatrixSmall) = colnames(teamMatrixSmall) = teams

# create the adjacency matrix, where the (i,j)th entry corresponds to the percent
# of the time where team i was favored against team j and outperformed the spread;
# the opposite interpretation is true for the (j,i)th entry
for (i in 1:length(teams)) {
  for (j in 1:length(teams)) {
    team1 = teams[i]; team2 = teams[j]
    if (team1 != team2) {
      favored_dat = football_data_net %>% filter(team_favorite_id==team1,
                                                   team_notfavorite_id==team2)
      percent_beat_or_equal = nrow(favored_dat[favored_dat$Actual.difference...spread>=0,]) /
        nrow(favored_dat) * 100
      teamMatrixSmall[i,j] = percent_beat_or_equal
    }
  }
}

# create graph object
teamgraph = graph_from_adjacency_matrix(teamMatrixSmall[1:32,1:32], mode='directed',
                                        weighted=TRUE,diag=F)

# update graph appearance
fun_color_range <- colorRampPalette(c("plum", "lightblue", "blue"))
my_colors <- fun_color_range(100)
E(teamgraph)$color = my_colors
```

```
## Warning in eattrs[[name]][index] <- value: number of items to replace is not a
## multiple of replacement length
```

```r
V(teamgraph)$color = "red"
V(teamgraph)$label.cex = .8
coords <- layout_with_gem(teamgraph)

# plot graph
# uncomment png and dev.off() lines to save image to computer
# png(file="/users/joegyorda/Desktop/QBS181net1.png", width=3600, height=3500, res=450)
plot.igraph(teamgraph,edge.arrow.size=.5, layout = coords)
image.plot(legend.only=T, zlim=range(1:100), col=my_colors,
           legend.lab=list('% of Games Better than Spread'), legend.cex=1)
```



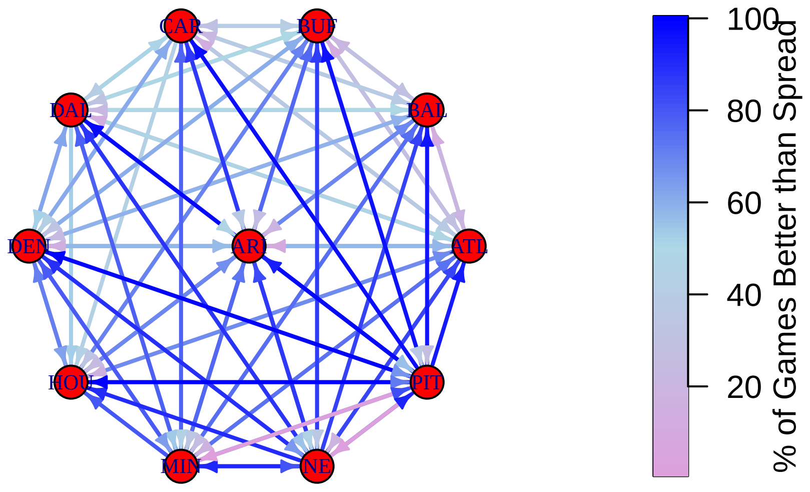
```r
# dev.off()
```

This chunk recreates the above network with fewer (11) teams handpicked to provide an easier visualization for interpretation purposes.

```r
# choose only a few teams and recreate graph
best_teams = sort(c("ARI","HOU","ATL","BAL","CAR","BUF","MIN","PIT","NE","DAL","DEN"))
mat = teamMatrixSmall[best_teams,best_teams]
teamgraph2 = graph_from_adjacency_matrix(mat, mode='directed',
                                         weighted=TRUE,diag=F)

# update edge and node colors
E(teamgraph2)$color = my_colors
```

```
## Warning in eattrs[[name]][index] <- value: number of items to replace is not a
## multiple of replacement length
```

```r
V(teamgraph2)$color = "red"
V(teamgraph2)$label.cex = .7
coords2 <- layout_as_star(teamgraph2) # pretty layout

# uncomment png and dev.off() lines to save image to computer
# png(file="/users/joegyorda/Desktop/QBS181net2.png", width=2100, height=2000, res=350)
plot.igraph(teamgraph2,edge.arrow.size=.5,layout =
                  coords2,edge.width=2)
image.plot(legend.only=T, zlim=range(1:100), col=my_colors,
              legend.lab=list('% of Games Better than Spread'), legend.cex=1)
```



```
# dev.off()



# SHOW SUMMARY TABLE OF TOP 5-10 BY OUTDEGREE AND BOTTOM 5-10
# average percent of the time each team is >= spread
sort(strength(teamgraph, mode='out')/degree(teamgraph,mode='out'))
```

```
##      ATL      LVR      NYJ      MIA      DET      LAC      DEN       TB
## 46.28030 47.10641 48.28700 48.38500 48.59644 48.80402 49.52766 49.87734
##      NYG      TEN      CIN      MIN      PIT      DAL       KC       NE
## 50.00541 50.43056 50.64769 50.89579 51.20007 51.22297 51.43506 51.76797
##      CLE      PHI      WAS      LAR      SEA      IND       NO      CAR
## 52.15476 52.21450 52.40886 52.45534 52.92716 54.02599 54.47278 54.90344
##      CHI       SF      BUF       GB      BAL
## 55.50673 55.54129 55.83079 59.51943 61.31077
```

```
# NYJ, MIA, DET are worst teams to bet on
# BAL, GB, BUF are best teams to bet on!
```

Section 4 - Scrap code for classification model

Below is classification code for predicting whether a given game was above/below the predicted spread using similar features as the linear mixed model. The results were poor (accuracy < 0.5), so the code is left here to show our work and implementation, but it should remain commented out.

```
# library(caret)
# library(randomForest)
# library(MLeval)
#
# # Create a new dataset with only the features we care about
# football_data_ml = football_data_filter
# football_data_ml = football_data_ml %>%
#   select(schedule_season, schedule_week, team_home, team_away, weather_temperature, weather_wind_mph,
#          weather_humidity, stadium_type, ELEVATION, over_under_line, Actual.difference...spread)
#
# # X1 is 1, X0 is 0, X.1 is -1
# football_data_ml$Actual.difference...spread = as.factor(ifelse(
#   football_data_ml$Actual.difference...spread > 0, 1, ifelse(
#     football_data_ml$Actual.difference...spread==0, 0,-1)))
#
# football_data_ml$Actual.difference.minus.spread = make.names(football_data_ml$Actual.difference...spr
# football_data_ml = football_data_ml %>% select(-Actual.difference...spread)
#
# # we'll just drop cases where spread exact
# football_data_ml = football_data_ml %>% filter(!Actual.difference.minus.spread=='X0')
# table(football_data_ml$Actual.difference.minus.spread)
#
# # need to do train-test split here!
#
# # set seeds
# set.seed(10111952)
#
# seeds_1 <- vector(mode="list", length=56)
# for(i in 1:55) seeds_1[[i]] <- sample.int(10000,144)
# seeds_1[[56]] <- sample.int(10000,1)
#
# fitControl_1 <- trainControl(method = "cv",
#                              number=5,
#                              classProbs = TRUE,
#                              savePredictions = TRUE,
#                              # sampling = "smote",
#                              seeds = seeds_1)
#
# # capture.output suppresses model output
# mod_1 <- caret::train(Actual.difference.minus.spread ~ ., data=football_data_ml,
#                         method = "rf",
#                         metric = 'Kappa',
#                         trControl = fitControl_1,
#                         na.action = na.omit)  # do we need this?
#
# # get model metrics
# (mod_1_results <- mod_1$results[rownames(mod_1$bestTune),])
#
```

```
# # perf_mod_1 <- MLeval::evalm(mod_1)
# sens = MLeval::evalm(mod_1,plots=c(),silent=TRUE)$optres$`Group 1`[1,1] # sensitivity
# spec = MLeval::evalm(mod_1,plots=c(),silent=TRUE)$optres$`Group 1`[2,1] # specificity
# auc = MLeval::evalm(mod_1,plots=c(),silent=TRUE)$optres$`Group 1`[13,1]
# ci = MLeval::evalm(mod_1,plots=c(),silent=TRUE)$optres$`Group 1`[13,2] # auc CI
#
# # return model performance metrics
# metrics = c(mod_1_results$Accuracy, sens, spec, auc, ci, mod_1_results$Kappa)
#
# caret::varImp(mod_1)
```

```
# ## let's just look at one season
# football_data_ml_2021 = football_data_ml %>% filter(schedule_season==2021)
# table(football_data_ml_2021$Actual.difference.minus.spread)
#
# football_data_ml_2021 = football_data_ml_2021 %>% select(-schedule_season)
#
# football_data_ml_2021$Actual.difference.minus.spread = as.factor(football_data_ml_2021$Actual.differe
#
# fitControl_1 <- trainControl(method = "repeatedcv",
#                              number=5,
#                              repeats=10,
#                              classProbs = TRUE,
#                              savePredictions = TRUE,
#                              # sampling = "smote",
#                              seeds = seeds_1)
#
# mod_1 <- caret::train(Actual.difference.minus.spread ~ schedule_week+team_home+team_away+weather_temp
#                           method = "rf",
#                           metric = 'Kappa',
#                           trControl = fitControl_1,
#                           na.action = na.omit)  # do we need this?
#
#  mod_1
```