Coding 2 - Assignment 1: Scraping ESPN for NFL Player Statistics

Bruno Helmeczy

23/11/2020

Function 2 Get game stats for best X Number of Players

```
# Scraping NFL Statistics from ESPN.com
library(readr)

GetTopNFLPlayersYTD <-  function(Playtype, top_x_players, year = 2020) {

# Check for required Packages
  if(!require(rvest)) install.packages("rvest", repos = "http://cran.us.r-project.org")
  if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
  if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
  if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")

# Load Required Packages
    library(rvest)
    library(data.table)
    library(dplyr)
    library(stringr)

  Stattype <- c("Passing","Rushing","Receiving")

# Check if Valid PlayType is inputed -> Return Error if inValid
  if (is.na(Stattype[grep(tolower(Playtype),tolower(Stattype))])) {
    print(paste0("ERROR: Choose from: '",
                 Stattype[1],"' or '",
                 Stattype[2],"' or '",
                 Stattype[3], "'" ))

  } else {
  myurl <- paste0('https://www.espn.com/nfl/stats/player/_/stat/',
                  tolower(Stattype[grep(tolower(Playtype),tolower(Stattype))])
                  ,'/season/',year,'/seasontype/2')

  print(paste0("Finding YTD ", Stattype[grep(tolower(Playtype),tolower(Stattype))]
               , " data from ", year ))

# Getting Table of top players year-to-date
  TableBox <- read_html(myurl) %>% html_nodes('.Table2__title--remove-capitalization')

# TableBox has 2 html_nodes, Table--align-right (Statistics)
```

```
                          #  & Table--fixed-left (Links)
  TL <- TableBox %>% html_nodes('.Table--align-right') %>% html_table()

# TL = Temporary list = PlayerNames, Rank, Stats
# TL[[1]] = Players Names & Rank / TL[[2]] = Statistics
  TL <- as.data.frame(cbind(TL[[1]],TL[[2]]))

# Getting Correct PlayerLinks
  PlayerList  <- TableBox %>% html_nodes('.Table--fixed-left') %>% html_table()
  PlayerLinks <- TableBox %>% html_nodes('.Table--fixed-left') %>%
    html_nodes('.AnchorLink') %>% html_attr('href')

# 1) Add "gamelog/_/" to access ALL games
  PlayerLinks <- unlist(lapply(PlayerLinks, function(x) {
    paste0( unlist(strsplit(x, '_/', fixed = T ))[1],"gamelog/_/",
            unlist(strsplit(x, '_/', fixed = T ))[2])
  }))

# 2) Remove player name from link - only 'id/ Nr.s' needed
  PlayerLinks <- unlist(lapply(PlayerLinks, function(x) {
    str_split(x,str_split_fixed(x,"/", n = 10)[,10])[1]
  }))

# 3) Remove empty strings from end of each link
  PlayerLinks <- PlayerLinks[ PlayerLinks != ""]

# 4) Add replace 'type/' with 'type/nfl/year/' + chosen year
  PlayerLinks <- unlist(lapply(PlayerLinks, function(x) {
    paste0( unlist(strsplit(x, 'type/', fixed = T ))[1],"type/nfl/year/", year)
  }))

# 5) Attach corrected Links to Players
  PlayerLinkList <- cbind(PlayerList[[1]],PlayerLinks)

# Join PlayerLinklist to TL w left_join - based on ranking & Name
  TL_w_Links <- TL %>% left_join(PlayerLinkList, by = c("RK", "Name"))
  TL_w_Links <- TL_w_Links[1:top_x_players,]

# Show for which players per game statistics is being loaded
  print(paste0("The top ",top_x_players, " ",
               Stattype[grep(tolower(Playtype),tolower(Stattype))],
               " players in ", year, " are:"))
  print(cbind(TL_w_Links$Name[1:top_x_players]))

  print(TL[1:top_x_players,])

# Create Empty dataframe to fill with per game statistics
  df_final <- data.frame()

for (i in 1:top_x_players)    {
  print(paste0("Getting game statistics data for ", TL_w_Links$Name[i] ))

  x <- read_html(TL_w_Links$PlayerLinks[i])
```

```
  listofTables <- x %>% html_nodes('.Table--align-right') %>% html_table()

# Getting game links
  ID_if_game_link <- x %>% html_nodes('.Table--align-right') %>%
    html_nodes('.AnchorLink') %>% html_attr('data-game-link')
  GameLink <- x %>% html_nodes('.Table--align-right') %>%
    html_nodes('.AnchorLink') %>% html_attr('href')

  GameLink <- as.data.frame(cbind(ID_if_game_link,GameLink)) %>%
    filter(!is.na(ID_if_game_link) ) %>% select(GameLink)

# Selecting Correct element from listofTables
    # In previous years, if player was in playoffs, listofTables has multiple elements
  # Has columns named as year + Regular Season
  # Loop through all list elements
  # Correct list has "Regular" in list element j's column names
  # If found, replace 1st element with correct list
  for (j in 1:(length(listofTables))) {
    if (length(grep("Regular",colnames(listofTables[[j]]))) > 0 ) {
      listofTables[[1]] <- listofTables[[j]]
      next()
    }
  }

# Select 1st elemnt from listofTables as Stats
  # Subset to Game ID variables & only chosen PlayType
  Stats <- listofTables[[1]]
  Stats <- cbind(Stats[colnames(Stats) == paste0(year ," Regular Season")],
                 Stats[colnames(Stats) == Stattype[grep(tolower(Playtype),tolower(Stattype))]])

# Re-Setting correct Column Names - Statistics' names in 1st Row
  colnames(Stats) <- Stats[1,]

# Removing incorrect Rows: 1st = Column Names ; Last = Year-To-Date Stats
  Stats <- Stats[2:(length(Stats[,1])-1),]

# Subsetting gamelinks to only Regular Season games - Playoff games are located on top
  GameLink <- GameLink[((length(GameLink[,1]))-(length(Stats[,1]))+1):(length(GameLink[,1])),]

# Creating a dataframe of Players' Name , Stats per game & Links to each game
  df <- data.frame(Player = TL_w_Links$Name[i], Stats, GameLink)

# Adding current players pergame stats to final dataframe
  df_final <- rbind(df_final,df)
}
print(paste0("Returning per game ",
             Stattype[grep(tolower(Playtype),tolower(Stattype))],
             " statistics for: "))
print(unique(df_final$Player))

write.csv(df_final, paste0("Top_",top_x_players
                           ,"_NFL_",Stattype[grep(tolower(Playtype),tolower(Stattype))],
                           "_Players_Game_Stats_",year,".csv"))
```

```
saveRDS(df_final, paste0("Top_", top_x_players,
                         "_NFL_",Stattype[grep(tolower(Playtype),tolower(Stattype))],
                         "_Players_Game_Stats_",year,".rds"))


return(df_final)
  }
}
```

```
# Scraping NFL Statistics from ESPN.com

# Game Stats
Passers   <- GetTopNFLPlayersYTD('Pass', 50)
Runners   <- GetTopNFLPlayersYTD('Rush', 50)
Recievers <- GetTopNFLPlayersYTD('Rec', 50)

Runners <- readRDS("Top_50_NFL_Rushing_Players_Game_Stats_2020.rds")

Passers$id <- paste0(Passers$Player,"_",Passers$Date)
Runners$id <- paste0(Runners$Player,"_",Runners$Date)
Recievers$id <- paste0(Recievers$Player,"_",Recievers$Date)

write_csv(Passers,   "Top_50_NFL_Passing_Players_Game_Stats_2020.csv")
write_csv(Runners,   "Top_50_NFL_Rushing_Players_Game_Stats_2020.csv")
write_csv(Recievers, "Top_50_NFL_Receiving_Players_Game_Stats_2020.csv")




Pass_YTD <- readRDS("Top_50_NFL_Passing_Players_YTD_Stats_2020.rds")
Rush_YTD <- readRDS("Top_50_NFL_Rushing_Players_YTD_Stats_2020.rds")
Rec_YTD  <- readRDS("Top_50_NFL_Receiving_Players_YTD_Stats_2020.rds")

Pass_YTD <- cbind(Pass_YTD,Stat = "Passing")
Rush_YTD <- cbind(Rush_YTD,Stat = "Rushing")
Rec_YTD  <- cbind(Rec_YTD, Stat = "Receiving")




# Get YTD Stats & Cleaned df 4 Prob Anal -----
source(paste0("https://raw.githubusercontent.com/BrunoHelmeczy/CEU_DA2_Assignment2/main/Codes/Get_Game_
source(paste0("https://raw.githubusercontent.com/BrunoHelmeczy/CEU_DA2_Assignment2/main/Codes/Get_Team_
source(paste0("https://raw.githubusercontent.com/BrunoHelmeczy/CEU_DA2_Assignment2/main/Codes/F_x_Clean_

Games <- GetGameScores(22,2020)
TeamStats <- GetNFLTeamStatsYTD(2020)
dfclean <- CleanedNFL_df(GetGameScores(16,2020),GetNFLTeamStatsYTD(2020))
dfclean$Team_Outcome <- ifelse(dfclean$Team_Outcome == "Win",1,0)
dfclean$Winner <- ifelse(dfclean$Winner == "Away", 1,0)
dfclean$Winner <- NULL
dfclean$Team_Outcome <- as.integer(dfclean$Team_Outcome)
```

```r
# Keep Only Team Abbreviations

TeamCities <- Games$Home_Team %>% unique()
x <- gsub("[^A-Z]"," ",TeamCities)
Abbr <- substring(x,nchar(x)-2,nchar(x)) %>% trimws()
Cities <- data.frame(TeamCities,Abbr)

# Teams for YTD Stats
YTD_Stats <- lapply(list(Pass_YTD,Rush_YTD,Rec_YTD), function(df) {
  x <- gsub("[^A-Z]"," ", df$Name)
  df$Team_Abbr <- substring(x, nchar(x)-2,nchar(x)) %>% trimws()
  df$Name <- substring(df$Name,1,nchar(df$Name)-nchar(df$Team_Abbr))
  df <- df %>% left_join(Cities, by = c("Team_Abbr" = "Abbr"))

  TeamsLong <- TeamStats$Team

  Cities2 <- lapply(1:32, function(x) {
    Xs <- TeamsLong[x] %>% str_split(" ") %>% unlist()
    City <- Xs[1:(length(Xs)-1)] %>% paste(collapse = " ")
  }) %>% unlist()

  Teams <- data.frame(TeamsLong,Cities2)

  MergeKeys <- NULL
  #i <- 2
  for (i in 1:32) {
    MergeKeys[i] <- grep(Teams$Cities2[i],df$TeamCities,value =T)[1]
  }

  Teams <- data.frame(Teams,MergeKeys)

  Teams$MergeKeys[Teams$TeamsLong == "New York Jets"] <- "New York NYJ"
  Teams$MergeKeys[Teams$TeamsLong == "Los Angeles Rams"] <- "Los Angeles LAR"
  Teams$MergeKeys[Teams$TeamsLong == "New York Giants"] <- "New York NYG"
  Teams$MergeKeys[Teams$TeamsLong == "Los Angeles Chargers"] <- "Los Angeles LAC"

  df <- df %>% left_join(Teams, by = c("TeamCities" = "MergeKeys")) %>%
    dplyr::select(-c(TeamCities,Cities2, Team_Abbr))


  return(df)
})

# Teams for Game Stats
Game_Stats <- lapply(list(Passers,Runners,Recievers), function(df) {
  x <- gsub("[^A-Z]"," ", df$Player)
  df$Team_Abbr <- substring(x, nchar(x)-2,nchar(x)) %>% trimws()
  df$Player <- substring(df$Player,1,nchar(df$Player)-nchar(df$Team_Abbr))
  df <- df %>% left_join(Cities, by = c("Team_Abbr" = "Abbr"))

  TeamsLong <- TeamStats$Team

  Cities2 <- lapply(1:32, function(x) {
```

```
    Xs <- TeamsLong[x] %>% str_split(" ") %>% unlist()
    City <- Xs[1:(length(Xs)-1)] %>% paste(collapse = " ")
  }) %>% unlist()

  Teams <- data.frame(TeamsLong,Cities2)

  MergeKeys <- NULL
  i <- 1
  for (i in 1:32) {
    MergeKeys[i] <- grep(Teams$Cities2[i],df$TeamCities,value =T)[1]
  }

  Teams <- data.frame(Teams,MergeKeys)

  Teams$MergeKeys[Teams$TeamsLong == "New York Jets"] <- "New York NYJ"
  Teams$MergeKeys[Teams$TeamsLong == "Los Angeles Rams"] <- "Los Angeles LAR"
  Teams$MergeKeys[Teams$TeamsLong == "New York Giants"] <- "New York NYG"
  Teams$MergeKeys[Teams$TeamsLong == "Los Angeles Chargers"] <- "Los Angeles LAC"

  df <- df %>% left_join(Teams, by = c("TeamCities" = "MergeKeys")) %>%
    dplyr::select(-c(TeamCities,Cities2, Team_Abbr))


  return(df)

})

Pass_YTD <- YTD_Stats[[1]]
Rush_YTD <- YTD_Stats[[2]]
Rec_YTD <-  YTD_Stats[[3]]
Rush_YTD <- Rush_YTD[duplicated(Rush_YTD[,2]) == F,]

Passers <- Game_Stats[[1]]
Runners <- Game_Stats[[2]]
Recievers <- Game_Stats[[3]]

checkup <- CleanedNFL_df(GetGameScores(22,2020),GetNFLTeamStatsYTD(2020))
checkup$Team_Outcome <- ifelse(checkup$Team_Outcome == "Win",1,0)
checkup$Winner <- ifelse(checkup$Winner == "Away", 1,0)
checkup$Winner <- NULL
checkup$Team_Outcome <- as.integer(checkup$Team_Outcome)

dfclean$OFF_pass_SACK_dummy <- ifelse(log(dfclean$OFF_pass_SACK) >= -0.2 &
                                        log(dfclean$OFF_pass_SACK) <= 1,1,0)

checkup$OFF_pass_SACK_dummy <- ifelse(log(checkup$OFF_pass_SACK) >= -0.2 &
                                        log(checkup$OFF_pass_SACK) <= 1,1,0)

modelformraw <- formula(Team_Outcome ~ log(OFF_pass_RTG) + log(DEF_pass_RTG) +
                          OFF_pass_SACK_dummy*log(OFF_pass_SACK) +
                          log(OFF_rush_AVG) +
                          lspline(OFF_Turnovers,c(0.8,3)) +
                          lspline(DEF_pass_SACK,1.6) )
```

```
modelformsimp <- formula(Team_Outcome ~ log(OFF_pass_RTG) + log(DEF_pass_RTG) +
          log(OFF_pass_SACK) +
          log(OFF_rush_AVG) +
#          OFF_Turnovers +
          log(DEF_pass_SACK) )

lpmfull <- lm( modelformraw , data=dfclean)
lpmsimp <- lm( modelformsimp , data=dfclean)


summary(lpmfull)
summary(lpmsimp)

checkup$pred_prob <- predict(lpmfull,type = "response", newdata = checkup)
checkup$pred_Team_win <- ifelse(checkup$pred_prob > 0.5,1,0)
checkup$correct <- ifelse(checkup$pred_Team_win == checkup$Team_Outcome, "Correct","Wrong")

table(checkup$correct[1:240]) %>% prop.table()
table(checkup$correct[241:269]) %>% prop.table()
table(checkup$correct) %>% prop.table()

table(checkup$pred_Team_win,checkup$Team_Outcome)
lpmfull$coefficients %>% cbind()

# LPM Simple -----
checkup$pred_prob_s <- predict(lpmsimp,type = "response", newdata = checkup)
checkup$pred_Team_win_s <- ifelse(checkup$pred_prob_s > 0.5,1,0)
checkup$correct_s <- ifelse(checkup$pred_Team_win_s == checkup$Team_Outcome, "Correct","Wrong")

table(checkup$correct_s[1:240]) %>% prop.table()
table(checkup$correct_s[241:269]) %>% prop.table()
table(checkup$correct_s) %>% prop.table()

table(checkup$pred_Team_win_s,checkup$Team_Outcome)
lpmsimp$coefficients %>% cbind()



checkup$pred_prob_s %>% hist()


# Transform Team Stats 2 per-game
TeamStats
colnames(TeamStats) <- gsub("/","_pr_",colnames(TeamStats))
colnames(TeamStats) <- gsub("%","_prc",colnames(TeamStats))

#### 2) Transform all variables 2 per Game ####
YTDTeams <- TeamStats %>%
  transmute(Team,
            off_pass_CMP       = round(off_pass_CMP/GP ,2)
            ,off_pass_ATT       = round(off_pass_ATT/GP ,2)
            ,off_pass_CMP_prc
            ,off_pass_AVG       = round(off_pass_AVG ,2)
```

```
            ,off_pass_YDS_pr_G  = round(off_pass_YDS_pr_G,2)
            ,off_pass_TD        = round(off_pass_TD/GP,2)
            ,off_pass_INT       = round(off_pass_INT/GP,2)
            ,off_pass_SACK      = round(off_pass_SACK/GP,2)
            ,off_pass_RTG
            ,off_rush_ATT       = round(off_rush_ATT/GP,2)
            ,off_rush_AVG       = round(off_rush_AVG,2)
            ,off_rush_YDS_pr_G  = round(off_rush_YDS_pr_G,2)
            ,off_rush_TD        = round(off_rush_TD/GP,2)
            ,off_rush_FUM       = round(off_rush_LST/GP,2)
            ,off_Turnovers      = round(off_rush_LST/GP,2) + round(off_pass_INT/GP,2)

            ,def_pass_CMP       = round(def_pass_CMP/GP ,2)
            ,def_pass_ATT       = round(def_pass_ATT/GP ,2)
            ,def_pass_CMP_prc
            ,def_pass_AVG       = round(def_pass_AVG,2)
            ,def_pass_YDS_pr_G  = round(def_pass_YDS_pr_G,2)
            ,def_pass_TD        = round(def_pass_TD/GP,2)
            ,def_pass_INT       = round(def_pass_INT/GP,2)
            ,def_pass_SACK      = round(def_pass_SACK/GP,2)
            ,def_pass_RTG
            ,def_rush_ATT       = round(def_rush_ATT/GP,2)
            ,def_rush_AVG       = round(def_rush_AVG,2)
            ,def_rush_YDS_pr_G  = round(def_rush_YDS_pr_G,2)
            ,def_rush_TD        = round(def_rush_TD/GP,2)
            ,def_rush_FUM       = round(def_rush_LST/GP,2)
            ,def_Turnovers      = round(def_rush_LST/GP,2) + round(def_pass_INT/GP,2))
```

```
# Double-Loop to map every team-opponent combination ----
Teams <- NULL
Opponents <- NULL
counter <- 1
for (i in TeamStats$Team) {
  for (j in TeamStats$Team) {
    if (i == j) {next()}
    Teams[counter] <- i
    Opponents[counter] <- j
    counter <- counter + 1
  }
}

AllMatchups <- data.frame(Teams,Opponents) %>% unique()


# To load in Tableau -----
checkup   # Includes Predictions
Pass_YTD  # Passing Leaders
Rush_YTD  # Rushing Leaders
```

```
Rec_YTD   # Receiving Leaders
YTDTeams  # Team Stats PER GAME
TeamStats # Team Stats AGGREG
AllMatchups # Iteration of all possible matchups

write.csv(checkup, "NFL_df_w_Predictions.csv")
write.csv(Pass_YTD,"NFL_YTD_Passing_Leaders.csv")
write.csv(Rush_YTD,"NFL_YTD_Rushing_Leaders.csv")
write.csv(Rec_YTD,"NFL_YTD_Receiving_Leaders.csv")
write.csv(YTDTeams,"Team_Stats_per_game.csv")
write.csv(TeamStats,"Team_Stats_Aggreg.csv")
write.csv(AllMatchups,"Matchups.csv")
```