MiniProject2NoahMitch

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   4.0.0     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(stringr)
library(rvest)
```

```
Attaching package: 'rvest'

The following object is masked from 'package:readr':

    guess_encoding
```

```
library(polite)
library(sf)
```

```
Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
library(maps)
```

```
Attaching package: 'maps'

The following object is masked from 'package:purrr':

    map
```

```
library(viridis)
```

```
Loading required package: viridisLite

Attaching package: 'viridis'

The following object is masked from 'package:maps':

    unemp
```

```
library(leaflet)
library(htmltools)
library(readr)
library(janitor)
```

```
Attaching package: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

Website Homepage: https://www.basketball-reference.com

Example Player Page: https://www.basketball-reference.com/players/e/edwaran01.html

Example Team Page: https://www.basketball-reference.com/teams/MIN/2025.html

Wiki NBA Team Abbreviation Page: https://en.wikipedia.org/wiki/Wikipedia:WikiProject_National_Basketba

Checking that Scrapping is Allowed

```
robotstxt::paths_allowed("https://www.basketball-reference.com/players/e/edwaran01.html")
```

```
 www.basketball-reference.com
```

```
[1] TRUE
```

```
robotstxt::paths_allowed("https://www.basketball-reference.com/teams/MIN/2025.html")
```

```
 www.basketball-reference.com
```

```
[1] TRUE
```

Player URL Setup

Since players have their own unique identifier for URL, values have to be hand taken from a player's page's URL. For the sake of time, we only took the identifiers of the top 5 players based on average Minutes Per Game from each team. If a player had already been taken from another team and appears again in the top 5 of another, we took the next best performing player from that team as well. This leads to 250 total players to be examined.

```
#top 5 players per team based on highest minutes per game average

player_url <- c("l/lavinza01", "f/foxde01", "d/derozde01", "s/sabondo01", "m/murrake02",  #k
                "b/bookede01", "d/duranke01", "b/bealbr01", "j/jonesty01", "o/onealro01", #s
                "h/hardeja01", "l/leonaka01", "z/zubaciv01", "p/powelno01", "b/bogdabo01", #
                "b/butleji01", "c/curryst01", "w/wiggian01", "g/greendr01", "p/podzibr01", #
                "d/doncilu01", "r/reaveau01", "j/jamesle01", "d/davisan02", "h/hachiru01", #
                "g/gilgesh01", "w/willija06", "d/dortlu01", "h/harteis01", "w/wallaca01", #t
                "j/jokicni01", "m/murraja01", "b/braunch01", "p/portemi01", "g/gordoaa01", #
                "c/camarto01", "s/simonan01", "g/grantje01", "s/sharpsh01", "a/aytonde01", #
                "e/edwaran01", "r/randlju01", "g/goberru01", "m/mcdanja02", "r/reidna01", #t
                "g/georgke01", "m/markkla01", "c/collijo01", "k/kesslwa01", "s/sextoco01", #
                "v/vanvlfr01", "g/greenja05", "t/thompam01", "b/brookdi01", "s/sengual01", #
                "w/wembavi01", "v/vassede01", "p/paulch01", "b/barneha02", "c/castlst01", #s
                "b/banede01", "m/moranja01", "j/jacksja02", "w/wellsja01", "a/aldamsa01", #g
                "m/murphtr02", "i/ingrabr01", "m/mccolcj01", "m/murrade01", "j/joneshe01", #
                "i/irvinky01", "w/washipj01", "c/chrisma02", "m/marshna01", "t/thompkl01", #
                "m/maxeyty01", "o/oubreke01", "g/grimequ01", "g/georgpa01", "m/martica02", #
```

```
                "h/hartjo01", "b/bridgmi01", "a/anunoog01", "b/brunsja01", "t/townska01", #kr
                "b/barnesc01", "b/barrerj01", "p/poeltja01", "d/dickgr01", "q/quickim01", #ra
                "t/tatumja01", "b/brownja02", "w/whitede01", "h/holidjr01", "p/porzikr01", #c
                "s/schrode01", "j/johnsca02", "t/thomaca02", "f/finnedo01", "t/timmedr01", #r
                "w/whiteco01", "v/vucevni01", "d/dosunay01", "g/giddejo01", "h/huertke01", #h
                "c/cunnica01", "h/harrito02", "i/iveyja01", "h/hardati02", "b/beaslma01", #pr
                "m/mitchdo01", "g/garlada01", "m/mobleev01", "a/allenja01", "s/strusma01", #c
                "l/lillada01", "k/kuzmaky01", "a/antetgi01", "l/lopezbr01", "p/princta02", #h
                "h/halibty01", "s/siakapa01", "t/turnemy01", "m/mathube01", "n/nembhan01", #p
                "h/herroty01", "a/adebaba01", "m/mitchda01", "r/roziete01", "j/jovicni01", #h
                "y/youngtr01", "j/johnsja05", "d/daniedy01", "h/huntede01", "o/okongon01", #h
                "m/millebr02", "b/ballla01", "b/bridgmi02", "w/willigr01", "g/greenjo02", #ho
                "b/banchpa01", "w/wagnefr01", "c/caldwke01", "s/suggsja01", "c/cartewe01", #r
                "c/coulibi01", "c/carrica01", "p/poolejo01", "s/sarral01", "j/johnsaj01") #wi
```

Scrapping using table and webpage based data (html_text and html_table)

```
seasonstat <- tibble() #fresh empty tibble that is ready to be added to

for(i in 1:150) { #loop for 150 players in player_url
  Sys.sleep(5) #Basketball reference has a limit on requests a minute so 5 seconds prevents i
  url <- str_c("https://www.basketball-reference.com/players/", player_url[i], ".html")
  bow(url, force = TRUE) #announce and ask for permission to scrape
  map <- read_html(url)
  player_nodes <- html_nodes(map, "span")
  player_names <- html_text(player_nodes) #html text containing player's name
  last5_check <- html_nodes(map, "h2")
  last5 <- html_text(last5_check) #html text containing if a player has a last 5 games table
  tables2 <- html_nodes(map, css = "table")
  if (last5[[1]] == "Last 5 Games") { #checks if player has last 5 games table
  table2 <- html_table(tables2, header = TRUE, fill = TRUE)[[3]] |> #table of season averages
  filter(Season == "2024-25") |>
  mutate(Player = player_names[[9]]) #player's name is 9th in vector
  seasonstat <- rbind(seasonstat, table2) #adds to out season stats table
  } else { #players with no last 5 games table
table2 <- html_table(tables2, header = TRUE, fill = TRUE)[[2]] |>
  filter(Season == "2024-25") |>
  mutate(Player = player_names[[9]])
  seasonstat <- rbind(seasonstat, table2)
  }
}
```

Backing Up File

```
write_csv(seasonstat, "~/sds264proj/seasonstat.csv")
```

Cleaning Our Data Set

```
season25 <- seasonstat |>
  select(-Lg, -Awards, -Season) |>
  clean_names() |>
  filter(team != "2TM") |>
  rename(position = pos, games_played = g, games_started = gs, minutes_played = mp, field_goa
```

Saving Data Set

```
write_csv(season25, "~/sds264proj/season25.csv")
```

CSV Description

Note: All of these statistics and figures are relative to the 2025 NBA season

age: Players age at start of 2025 season team: Accompanying team player was on for corresponding statistics position: Position labeled by team games_played: Number of games player entered a game games_started: Number of games a player was on the starting roster minutes_played: Average minutes played per game played field_goals: Average field goal makes per games played field_goal_attempts: Average field goal attempts per games played field_goal_percent: Average field goal percent per games played three_point_makes: Average three point makes per games played three_point_attempts: Average three point attempts per games played three_point_percent: Average three point percent per games played two_point_makes: Average two point makes per games played two_point_attempts: Average two point attempts per games played two_point_percent: Average two point percent per games played expected_field_goal_percent: Average expected field goal percent per games played free_throw_makes: Average free throw makes per games played free_throw_attempts: Average free throw attempts per games played free_throw_percent: Average free throw percent per games played offensive_rebounds: Average offensive rebounds per games played defensive_rebounds: Average defensive rebounds per games played total_rebounds: Average total rebounds per games played assists: Average assists per game played steals: Average steals per game played blocks: Average blocks per game played turnovers: Average turnovers per game played personal_fouls: Average personal fouls per game played points: Average points per game played player: Name of player

Team Data

Wiki NBA team abbreviation page: https://en.wikipedia.org/wiki/Wikipedia:WikiProject_National_Basketbal

```
robotstxt::paths_allowed("https://en.wikipedia.org/wiki/Wikipedia:WikiProject_National_Basket
```

```
 en.wikipedia.org
```

```
[1] TRUE
```

To loop through each url of team data, we only need to change the three letter team abbreviation to determine which team's page we are accessing. Instead of manually entering this list, I took it upon myself to scrape the names from a list from wikipedia.

```
# scrape list of team abbreviations from wikipedia and clean up list of team abbreviations
team_abbv_wiki <- read_html("https://en.wikipedia.org/wiki/Wikipedia:WikiProject_National_Bas

team_abbv_temp <- html_nodes(team_abbv_wiki, "td:nth-child(1)")

team_abbv <- html_text(team_abbv_temp)
```

```
# cleaning up abbreviation data
for (i in seq_along(team_abbv)) {
  team_abbv[i] <- str_replace(team_abbv[i], "^(.)(.)(.).*$", "\\1\\2\\3")
  team_abbv[i] <- str_remove(team_abbv[i], "\n")
}

# accounting for errors made by wikipedia site
team_abbv[3] = "BRK"
team_abbv[4] = "CHO"
team_abbv[24] = "PHO"
team_abbv
```

```
 [1] "ATL" "BOS" "BRK" "CHO" "CHI" "CLE" "DAL" "DEN" "DET" "GSW" "HOU" "IND"
[13] "LAC" "LAL" "MEM" "MIA" "MIL" "MIN" "NOP" "NYK" "OKC" "ORL" "PHI" "PHO"
[25] "POR" "SAC" "SAS" "TOR" "UTA" "WAS"
```

```
# function to scrape basketball data
get_text_from_page <- function(page, css_selector) {
  page |>
    html_nodes(css_selector) |>
    html_text()
}
```

```r
# function to turn basketball data into a tibble
basketball_tibble <- function(team, year = 2025) {
  url = str_c("https://www.basketball-reference.com/teams/", team, "/", year, ".html")
  session <- bow(url, force = T)
  page <- scrape(session)
  team_record <- get_text_from_page(page, ".prevnext+ p")
  points_per_g <- get_text_from_page(page, "p:nth-child(6)")
  ratings <- get_text_from_page(page, "p:nth-child(8)")
  tibble(team = team, team_record = team_record, points_per_g = points_per_g, ratings = ratin
}

team_stats_temp <- map2(team_abbv, 2025, basketball_tibble) # creates a list of one-row tibbl
team_stats <- list_rbind(team_stats_temp) # binds the tibbles together
```

```r
# cleaning up our tibble
team_stats <- team_stats |>
  mutate(team_record = str_extract(team_record, "\\d\\d-\\d\\d"), # extracts the xx-xx numbe
         wins = parse_number(str_extract(team_record, "^\\d\\d")), # extracts the first set
         losses = parse_number(str_extract(team_record, "\\d\\d$")), # extracts the last set
         points_per_g = str_remove_all(points_per_g, " "),
         points_per_g = str_remove_all(points_per_g, "\n"), # gets rid of enters and spaces
         opp_points_per_g = str_extract(points_per_g, "OppPTS/G:\\d\\d\\d\\.\\d"),# retrieves
         points_per_g = parse_number(str_extract(points_per_g, "\\d\\d\\d\\.\\d")), # retriev
         opp_points_per_g = parse_number(str_extract(opp_points_per_g, "\\d\\d\\d\\.\\d")),
         ratings = str_remove_all(ratings, " "),
         ratings = str_remove_all(ratings, "\n"), # gets rid of enters and spaces
         off_rating = parse_number(str_extract(ratings, "\\d\\d\\d\\.\\d")), # retrieves firs
         def_rating = str_extract(ratings, "DefRtg:\\d\\d\\d\\.\\d"), # retrieves rating afte
         def_rating = parse_number(str_remove(def_rating, "DefRtg:")), # removes "DefRtg" pa
         net_rating = off_rating - def_rating) |> # defines net_rating
  select(team, wins, losses, points_per_g, opp_points_per_g, off_rating, def_rating, ) # spe
```

```r
# saving data set as csv
write_csv(team_stats, "~/sds264proj/team_stats.csv")
```

CSV Description

team - the three-letter abbreviation of the team associated to the stats on a given row wins - total number of wins during the in-season losses - total number of losses during the in-season points_per_g - the average amount of points scored per game for a given team opp_points_per_g - the average the average amount of points scored per game for the any

particular opponent of a given team off_rating - offensive rating, measures points scored per 100 possessions def_rating - defensive rating, measures points allowed per 100 possessions net_rating - off_rating minus def_rating