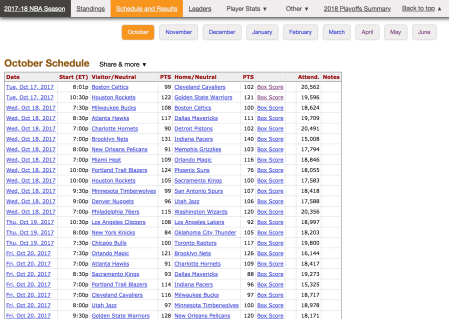
I’m a casual NBA fan: I don’t have time to watch the games but enjoy viewing the highlights on Instagram/Youtube (especially [Shaqtin’ A Fool](http://www.nba.com/fool)!); I sometimes read game articles and analyses (e.g. [Blogtable](http://www.nba.com/blogtable)). Apart from the game being an amazing visual spectacle, it’s fun to drink in the deluge of stats that each game brings. I’m not even talking about advanced stats and “[ABPRmetrics](https://en.wikipedia.org/wiki/APBRmetrics" \t "_blank)“: there’s something exciting about seeing how many different statistical records can be broken on a given night.

As a data/stats person, I’ve been wanting to get my hands on NBA data and play around with it on my own. However, in my internet searching I didn’t come across any free easy-to-use datasets. The website [Basketball-Reference.com](https://www.basketball-reference.com/) is an excellent compendium of all the data I would want, but it was embedded within the webpage, not made available in an analysis-ready format. (Or at least, I couldn’t find it, or it wasn’t free.)



A screenshot from [basketball-reference.com](http://basketball-reference.com).

I recently found some spare time on my hands and decided that it was time for me to learn how to scrape data from this website. And it was surprisingly easy! In this post, I will walk through the steps for scraping top-level game data for the 2017-2018 NBA season (i.e. data from the screenshot above).  Click [here](https://github.com/kjytay/misc/blob/master/blog/2018-12-11_nba_game_data.R) to view the full R code. If you only want the data, you can download it [here](https://github.com/kjytay/misc/blob/master/data/NBA-2018_game_data.rds) in RDS format.

**Scraping the data**

First, let’s load the packages we will use for the web scraping:

library(rvest)

library(lubridate)

From the screenshot above, you may notice that game data for the season is split over several pages, with one page for the games in a given month. As such we will need to loop over the months and scrape the webpage for each month. We do that in the full R script; the explanation below shows the code for scraping for the month of October.

We can get the webpage as an xml\_document object by using rvest‘s read\_html function:

year <- "2018"

month <- "october"

url <- paste0("<https://www.basketball-reference.com/leagues/NBA_>", year,

"\_games-", month, ".html")

webpage <- read\_html(url)

To get the data we want from this object, we need to look for the CSS selectors of the data we want. This involves inspecting the raw HTML of the webpage and finding the unique path that gets your data (and nothing else). To get the column names for this dataset, we extract the HTML nodes with CSS selector "table#schedule > thead > tr > th", and then pull out the value of the attribute "data-stat":

col\_names <- webpage %>%

html\_nodes("table#schedule > thead > tr > th") %>%

html\_attr("data-stat")

col\_names <- c("game\_id", col\_names)

Notice that pipes %>% work with rvest‘s functions. (The game\_id column cannot be pulled out in this way, and so I’ve added it in manually.)

Next, I will extract the dates and game IDs in a similar manner. The only snag here is that the table in the month of April is slightly different, since the playoffs start that month:



Part of the Apr 2018 table. The Playoffs row interferes with our web scraping.

We will need a bit of tinkering to remove the effects of that row:

dates <- webpage %>%

html\_nodes("table#schedule > tbody > tr > th") %>%

html\_text()

dates <- dates[dates != "Playoffs"]

game\_id <- webpage %>%

html\_nodes("table#schedule > tbody > tr > th") %>%

html\_attr("csk")

game\_id <- game\_id[![is.na](http://is.na)(game\_id)]

The rest of the data is fairly straight forward to pull out. We then combine this data along with dates and game\_id into a single data frame:

data <- webpage %>%

html\_nodes("table#schedule > tbody > tr > td") %>%

html\_text() %>%

matrix(ncol = length(col\_names) - 2, byrow = TRUE)

month\_df <- as.data.frame(cbind(game\_id, dates, data), stringsAsFactors = FALSE)

names(month\_df) <- col\_names

From here, assume that we did the above for all the months and combined them into one big data frame df. When web scraping, all the data is pulled out as character strings, so we need to do some typecasting to get the data into the correct type. I also added a column to indicate whether a game was a regular season game or a playoff game (this is where we need the lubridate package) and dropped the box score column.

# change columns to the correct types

df$visitor\_pts <- as.numeric(df$visitor\_pts)

df$home\_pts <- as.numeric(df$home\_pts)

df$attendance <- as.numeric(gsub(",", "", df$attendance))

df$date\_game <- mdy(df$date\_game)

# add column to indicate if regular season or playoff

playoff\_startDate <- ymd("2018-04-14")

df$game\_type <- with(df, ifelse(date\_game >= playoff\_startDate,

"Playoff", "Regular"))

# drop boxscore column

df$box\_score\_text <- NULL

**Sanity check: Table standings**

Let’s perform a sanity check by recreating the regular season table standings for each conference. The code in this section could be more elegant by using functions from the tidyverse, but I’ll demonstrate that we can do what we want using just base R functions.

First we create columns indicating the winner and loser of each game, then pull out just the regular season games:

df$winner <- with(df, ifelse(visitor\_pts > home\_pts,

visitor\_team\_name, home\_team\_name))

df$loser <- with(df, ifelse(visitor\_pts < home\_pts,

visitor\_team\_name, home\_team\_name))

regular\_df <- subset(df, game\_type == "Regular")

Next, we build up a new data frame where each row corresponds to one team. We manually input the conference and division for each team, since there are only 30 of them (getting them programmatically would probably take longer than manual data entry):

teams <- sort(unique(regular\_df$visitor\_team\_name))

standings <- data.frame(team = teams, stringsAsFactors = FALSE)

standings$conf <- c("East", "East", "East", "East", "East",

"East", "West", "West", "East", "West",

"West", "East", "West", "West", "West",

"East", "East", "West", "West", "East",

"West", "East", "East", "West", "West",

"West", "West", "East", "West", "East")

standings$div <- c("Southeast", "Atlantic", "Atlantic", "Southeast", "Central",

"Central", "Southwest", "Northwest", "Central", "Pacific",

"Southwest", "Central", "Pacific", "Pacific", "Southwest",

"Southeast", "Central", "Northwest", "Southwest", "Atlantic",

"Northwest", "Southeast", "Atlantic", "Pacific", "Northwest",

"Pacific", "Southwest", "Atlantic", "Northwest", "Southeast")

We populate the win loss columns in the following way: for each team, find the number of times it appears in each of the winner and loser columns in df. I use a for loop here, which is not a big problem here since there are only 30 teams, but the code could probably be improved to avoid the loop.

standings$win <- 0; standings$loss <- 0

for (i in 1:nrow(standings)) {

standings$win[i] <- sum(regular\_df$winner == standings$team[i])

standings$loss[i] <- sum(regular\_df$loser == standings$team[i])

}

The win-loss percentage can be calculated easily:

standings$wl\_pct <- with(standings, win / (win + loss))

Now that our standings table is complete, we can compare them with the actual standings table. There are slightly differences because when teams tie in W-L percentage, we just list them in alphabetical order. In real life tiebreaking is quite a bit more complicated (see the basis for tiebreaking near the bottom of [this page](https://stats.nba.com/standings/)).

# Eastern conference standings

east\_standings <- subset(standings, conf == "East") east\_standings[with(east\_standings, order(-wl\_pct, team)), c("team", "win", "loss")] #> team win loss

#> 28 Toronto Raptors 59 23

#> 2 Boston Celtics 55 27

#> 23 Philadelphia 76ers 52 30

#> 6 Cleveland Cavaliers 50 32

#> 12 Indiana Pacers 48 34

#> 16 Miami Heat 44 38

#> 17 Milwaukee Bucks 44 38

#> 30 Washington Wizards 43 39

#> 9 Detroit Pistons 39 43

#> 4 Charlotte Hornets 36 46

#> 20 New York Knicks 29 53

#> 3 Brooklyn Nets 28 54

#> 5 Chicago Bulls 27 55

#> 22 Orlando Magic 25 57

#> 1 Atlanta Hawks 24 58

# Western conference standings

west\_standings <- subset(standings, conf == "West") west\_standings[with(west\_standings, order(-wl\_pct, team)), c("team", "win", "loss")] #> team win loss

#> 11 Houston Rockets 65 17

#> 10 Golden State Warriors 58 24

#> 25 Portland Trail Blazers 49 33

#> 19 New Orleans Pelicans 48 34

#> 21 Oklahoma City Thunder 48 34

#> 29 Utah Jazz 48 34

#> 18 Minnesota Timberwolves 47 35

#> 27 San Antonio Spurs 47 35

#> 8 Denver Nuggets 46 36

#> 13 Los Angeles Clippers 42 40

#> 14 Los Angeles Lakers 35 47

#> 26 Sacramento Kings 27 55

#> 7 Dallas Mavericks 24 58

#> 15 Memphis Grizzlies 22 60

#> 24 Phoenix Suns 21 61



2017-18 regular season conference standings from [basketball-reference.com](http://basketball-reference.com).