STAT GR5206 HW2_mjs2364

Mathieu Sauterey - UNI: MJS2364

1 octobre 2017

i. Use the readLines() command we studied in class to load the NetsSchedule.html file into a character vector in R. Call the vector nets1718.

```
nets1718 <- readLines("NetsSchedule.html", warn = FALSE)
```

a. How many lines are in the NetsSchedule.html file?

```
length(nets1718) #computes the number of lines of the file
```

```
## [1] 828
```

There are 828 lines in the file.

b. What is the total number of characters in the file?

```
sum(nchar(nets1718)) #computes the sum of the number of characters in every line of the file
```

```
## [1] 129188
```

There is a total of 129,188 characters in the file.

c. What is the maximum number of characters in a single line of the file?

```
max(nchar(nets1718)) #finds the maximum number of characters among all lines of the file
```

```
## [1] 9736
```

The maximum number of characters in a single line of the file is 9,736 characters.

ii. Open NetsSchedule.html as a webpage. This should happen if you simply click on the file. You should see a table listing all the games scheduled for the 2017-2018 NBA season. There are a total of 82 regular season games scheduled. Who and when are they playing first? Who and when are they playing last?

They are playing first against Indiana on Wednesday, October 18 at 7:00pm. They are playing last against Boston on Wednesday, April 11 at 8:00pm.

iii. Now, open NetsSchedule.html using a text editor. To do this you may need to right-click on the file and tell your computer to use a text editor to open the file. What line in the file holds information about the first game of the regular season (date, time, opponent)? What line provides the date, time, and opponent for the final game?

Line 321 holds information about the first game of the regular season against Indiana. Line 402 provides information for the final game against Boston.

iv. Write a regular expression that will capture the date of the game. Then using the grep() function find the lines in the file that correspond to the games.

```
date_game <- "[A-Z][a-z]{2},\\s[A-Z][a-z]{2}\\s[0-9]{1,2}" #Regular expression to find dates
```

```
length((grep(nets1718,pattern = date_game))) #prints the number of lines containing a game date
```

```
## [1] 82
```

There are 82 lines containing game dates in the file. This figure is correct because that there are 82 Nets games scheduled.

```
head(grep(nets1718,pattern = date_game),1) #prints the first line number containaing a game date
```

```
## [1] 321
```

The statement above aims to return the number of the line holding information for the first game of the season, against Indiana. This prompts line 321 which is correct per (iii).

```
tail(grep(nets1718,pattern = date_game),1) #prints the last line number containaing a game date
```

```
## [1] 402
```

The statement above aims to return the number of the line holding information for the last game of the season, against Boston. This prompts line 402 which is correct per (iii).

v. Using the expression you wrote in (iv) along with the functions regexp() and regmatches(), extract the dates from the text file. Store this information in a vector called date to save to use below.

```
date_game <- "[A-Z][a-z]{2},\\s[A-Z][a-z]{2}\\s[0-9]{1,2}" #Regular expression to find dates
date_log <- grepl(nets1718, pattern = date_game) #stores a logical vector of lines with dates
matches_date <- gregexpr(date_game, nets1718[date_log]) # creates a list with dates str locations
date_list <- regmatches(nets1718[date_log], matches_date) # stores the dates in a list
date <- unlist(date_list) # converts the list containing game dates into a vector
```

vi. Use the same strategy as in (v) and (vi) to create a time vector that stores the time of the game.

```
time_game <- "[0-9]{1,2}:[0-9]{1,2}\\sPM" #Regular expression to find game times
time_log <- grepl(nets1718, pattern = time_game) #stores a logical vector of lines with times
matches_time <- gregexpr(time_game, nets1718[time_log]) # creates a list with times str locations
time_list <- regmatches(nets1718[time_log], matches_time) # stores the times in a list
time <- unlist(time_list) # converts the list containing game times into a vector
```

vii. Capture whether the game is home or away using a regular expression. You may want to use the HTML code around these values to guide your search. Then extract this information and use it to create a vector called home which takes the value 1 if the game is played at home or 0 if it is away.

```
home_game <- "class=\"game-status\">(vs|@)" #Regular expression to find game locations
home_log <- grepl(nets1718, pattern = home_game) #stores a logical vector of lines with location
matches_home <- gregexpr(home_game, nets1718[home_log]) # creates a list of locations' str locations
home_list <- regmatches(nets1718[home_log], matches_home) # stores the game locations in a list
home <- unlist(home_list) # converts the list containing game locations into a vector
home <- gsub("class=\"game-status\">","",home) # cleans up HTML codes preceding location strings
home <- as.numeric(home == "vs") #converts location vector into numerical 1-0 'logic' vector
```

viii. Finally we would like to find the opponent, again capture this information using a regular expression. Extract these values and save them to a vector called opponent. Again, to write your regular expression you may want to use the HTML code around the names to guide your search.

```
opponent_game <- "([A-Za-z]+)?\\s?[A-Za-z]+</a></li></ul></td><td>[0-9]{1,2}:[0-9]{1,2}\\sPM"
opponent_log <- grepl(nets1718, pattern = opponent_game)#logical vector of lines with opponent
matches_opponent <- gregexpr(opponent_game, nets1718[opponent_log]) #list of opponent str location
opponent_list <- regmatches(nets1718[home_log], matches_opponent) #stores opponent info in list
opponent <- unlist(opponent_list) #converts the list into a vector
opponent <- gsub("</a></li></ul></td><td>[0-9]{1,2}:[0-9]{1,2}\\sPM","",opponent) #cleans up
```

ix. Construct a data frame of the four variables in the following order: date, time, opponent, home. Print the frame from rows 1 to 10. Does the data match the first 10 games as seen from the web browser?

```
head(data.frame(date, time, opponent, home),10)
```

```
##           date     time    opponent home
## 1  Wed, Oct 18  7:00 PM     Indiana    0
## 2  Fri, Oct 20  7:30 PM     Orlando    1
## 3  Sun, Oct 22  3:30 PM     Atlanta    1
## 4  Tue, Oct 24  7:00 PM     Orlando    0
## 5  Wed, Oct 25  7:30 PM   Cleveland    1
## 6  Fri, Oct 27  7:30 PM   NY Knicks    0
## 7  Sun, Oct 29  6:00 PM      Denver    1
## 8  Tue, Oct 31  7:30 PM     Phoenix    1
## 9   Fri, Nov 3 10:30 PM Los Angeles    0
## 10  Mon, Nov 6  9:00 PM     Phoenix    0
```

The first 10 rows of our dataframe perfectly matches the information of the first 10 games as seen from the web browser.