Homework_3

Christine Chong cc4190

June 9, 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 nets1617.

```
nets1617 <- readLines("NetsSchedule.html" )
```

a. How many lines are in the NetsSchedule.html file? Here we can use the length function to find the number of lines

```
length(nets1617)
```

```
## [1] 811
```

b. What is the total number of characters in the file? Here we can add all the character counts from each line of the file using sum and nchar.

```
sum(nchar(nets1617))
```

```
## [1] 127835
```

c. What is the maximum number of characters in a single line of the file? Here we can use the max function on nchar(nets1617) to figure out the maximum number of characters in a single line of the file. (Especially because nchar(nets1617) prints out the number of character counts there are for each line of the file).

```
max(nchar(nets1617))
```

```
## [1] 7211
```

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 2016-2017 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 Boston on October 26 first. They are playing Chicago on April 12 last.

iii. Now, open NetsSchedule.html using a text editor. To do this you may need to rightclick 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? It may be helpful to use CTRL-F or COMMAND-F here and also work between the file in R and in the text editor. Using NetsSchedule.html we'd like to extract the following variables: the date, the game time (ET), the opponent, and whether the game is home or away. Looking at the file in the text editor, locate each of these variables. For the next part of the homework we use regular expressions to extract this information. 1

Line 315 starts the code for the Schedule.

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. Make sure that grep() finds 82 lines, and the first and last locations grep() finds match the first and last games you found in (ii). Here we can use grep along with regular expressions to find the date Capital Letter, 2 Lower Case Letters Space Number and Maybe a Number

```
findByDate<- "[A-Z][a-z]{2}, [A-Z][a-z]{2} [0-9][0-9]?"
grepDate <- grep(findByDate, nets1617)
length(grepDate)
```

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

v. Using the expression you wrote in (v) 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. HINT: We did something like this in class. Here we can use grepl, gregexpr and regmatches to get the specific strings that match the pattern we stated earlier. We must unlist the matches here so that the later data frame prints out as intended

```r
match_bool<- grepl(nets1617, pattern = findByDate)
match_dates <- gregexpr(pattern = findByDate, text = nets1617[match_bool])
matches <- regmatches(nets1617[match_bool], match_dates)
matches <- unlist(matches)
head(matches, 4)
```

```
## [1] "Wed, Oct 26" "Fri, Oct 28" "Sat, Oct 29" "Mon, Oct 31"
```

vi. Use the same strategy as in (v) and (vi) to create a time vector that stores the time of the game. Here we can use the same strategy as earlier except the regular expression will be different. Number, : ,2 Numbers, Space, 2 Capital Letter,

```r
findTime <- "[0-9]\\:[0-9]{2}\\s+[A-Z]{2}"
grepTime <- grep(findTime, nets1617)
length(grepTime)
```

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

```r
time_bool<-grepl(nets1617, pattern =findTime)
match_time <-gregexpr(pattern = findTime, text=nets1617[time_bool])
times <- regmatches(nets1617[time_bool], match_time)
times <- unlist(times)
head(times, 4)
```

```
## [1] "7:30 PM" "7:30 PM" "8:00 PM" "7:30 PM"
```

vii. We would now like to gather information about whether the game is home or away. This information is indicated in the schedule by either an '@' or a 'vs' in front of the opponent. If the Nets are playing '@' their opponent's court, the game is away. If the Nets are playing 'vs' the opponent, the game is at home. Capture this information 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. HINT: In my solution, I use the fact that in each line, the string
appears before this information. So my regular expression searches for that string followed by '@' or that string followed by 'vs'. After I've extracted these strings, I use substr() to finally extract just the '@' or the 'vs'. Here we have to use a slightly different strategy. (Writing the html code as the regular expression instead of the expression itself). And then we used an ifelse statement to create a vector of 0s and 1s.
, Maybe @, Maybe lower case letter, Maybe lower case letter

```r
findStatus <- "<li class=\"game-status\">[[:punct:]]?[a-z]?[a-z]?"

grepStatus <- grep(findStatus, nets1617)
length(grepStatus)
```

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

```r
status_bool <- grepl(nets1617, pattern=findStatus)
match_status <-gregexpr(pattern = findStatus, text=nets1617[status_bool])
status <- regmatches(nets1617[status_bool], match_status)
home <- ifelse(status ==  "<li class=\"game-status\">@", 0, 1)
home
```

```
##  [1] 0 1 0 1 1 1 1 0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0 1 0 1
## [36] 1 1 1 0 1 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 1 1 1
## [71] 1 0 0 1 0 1 1 0 0 1 0 0
```

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. Here we have to use the url to find the components that contained the team names. www.espn.com/nba/team/_/name/, Lowercase Letter at least once, / , Lower case letter at least once, Maybe -, Maybe lower case letter at least once, Maybe -, Lower case letter at least once. After we vectorized the url we can use string split by / and get the last index of this vector to get the team name

```r
findTeam <- "www.espn.com/nba/team/_/name/[a-z]{1,}/[a-z]{1,}-?[a-z]{1,}-?[a-z]{1,}"
grepTeam <- grep(findTeam, nets1617)
length(grepTeam)
```

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

```r
team_bool <-grepl(nets1617, pattern = findTeam)
match_team <- gregexpr(pattern =  findTeam, text=nets1617[team_bool])
team <-regmatches(nets1617[team_bool], match_team)
opponent <- c()
for(i in 1:length(team)){
 url_vec <- strsplit(team[[i]][1], split = "/")
 opponent[i] <- unlist(url_vec)[7]
}
opponent
```

```
##  [1] "boston-celtics"         "indiana-pacers"
##  [3] "milwaukee-bucks"        "chicago-bulls"
##  [5] "detroit-pistons"        "charlotte-hornets"
##  [7] "minnesota-timberwolves" "new-york-knicks"
##  [9] "phoenix-suns"           "la-clippers"
## [11] "los-angeles-lakers"     "oklahoma-city-thunder"
## [13] "portland-trail-blazers" "boston-celtics"
## [15] "indiana-pacers"         "sacramento-kings"
## [17] "la-clippers"            "milwaukee-bucks"
## [19] "milwaukee-bucks"        "washington-wizards"
## [21] "denver-nuggets"         "san-antonio-spurs"
## [23] "houston-rockets"        "los-angeles-lakers"
## [25] "orlando-magic"          "philadelphia"
## [27] "toronto-raptors"        "golden-state-warriors"
## [29] "cleveland-cavaliers"    "charlotte-hornets"
## [31] "chicago-bulls"          "washington-wizards"
## [33] "utah-jazz"              "indiana-pacers"
## [35] "cleveland-cavaliers"    "philadelphia"
## [37] "atlanta-hawks"          "new-orleans-pelicans"
## [39] "toronto-raptors"        "houston-rockets"
## [41] "toronto-raptors"        "new-orleans-pelicans"
## [43] "charlotte-hornets"      "san-antonio-spurs"
## [45] "miami-heat"             "cleveland-cavaliers"
## [47] "minnesota-timberwolves" "miami-heat"
## [49] "new-york-knicks"        "indiana-pacers"
## [51] "toronto-raptors"        "charlotte-hornets"
## [53] "washington-wizards"     "miami-heat"
## [55] "memphis-grizzlies"      "milwaukee-bucks"
## [57] "denver-nuggets"         "golden-state-warriors"
## [59] "sacramento-kings"       "utah-jazz"
## [61] "portland-trail-blazers" "memphis-grizzlies"
## [63] "atlanta-hawks"          "dallas-mavericks"
## [65] "new-york-knicks"        "oklahoma-city-thunder"
## [67] "new-york-knicks"        "boston-celtics"
## [69] "dallas-mavericks"       "detroit-pistons"
## [71] "phoenix-suns"           "washington-wizards"
## [73] "atlanta-hawks"          "philadelphia"
## [75] "detroit-pistons"        "orlando-magic"
## [77] "atlanta-hawks"          "philadelphia"
## [79] "orlando-magic"          "chicago-bulls"
## [81] "boston-celtics"         "chicago-bulls"
```

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? Here we can use the variables earlier to create the data frame combo. It does match the first 10 games as seen from the web browser.

```r
combo <- data.frame(matches,times,home,opponent, check.names = TRUE)
head(combo, 10)
```

```
##        matches   times home               opponent
## 1  Wed, Oct 26 7:30 PM    0         boston-celtics
## 2  Fri, Oct 28 7:30 PM    1         indiana-pacers
## 3  Sat, Oct 29 8:00 PM    0        milwaukee-bucks
## 4  Mon, Oct 31 7:30 PM    1          chicago-bulls
## 5   Wed, Nov 2 7:30 PM    1        detroit-pistons
## 6   Fri, Nov 4 7:30 PM    1      charlotte-hornets
## 7   Tue, Nov 8 7:30 PM    1 minnesota-timberwolves
## 8   Wed, Nov 9 7:00 PM    0        new-york-knicks
## 9  Sat, Nov 12 9:00 PM    0           phoenix-suns
## 10 Mon, Nov 14 0:30 PM    0            la-clippers
```