Homework 3 Solutions

Cynthia Rush (cgr2130)

October 10, 2016

i.

```
setwd("~/Desktop/Data")
nets1617 <- readLines("NetsSchedule.html")
```

The number of lines in the file corresponds to the length of the vector *nets1617*.

```
length(nets1617)
```

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

I can find the number of characters in each line of the file by running *nchar(nets1617)* since *nchar()* vectorizes. This will return a vector of length 9306 with each element telling the number of characters in the corresponding line of the file. Then we can take a sum of these values to give the total number of characters.

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

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

Finally, I can use the *max()* command, with *nchar(nets1617)* as its input, to find the maximum number of characters in any line of the code.

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

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

ii. In the first game of the regular season, the Nets are playing the Boston Celtics in Boston on Wednesday, October 26 at 7:30PM. In the last game of the season, the Nets are playing the Chicago Bulls in Chicago on Wednesday, April 12 at 8:00PM.

iii. The 315^{th} line corresponds to the first game of the regular season and the 396^{th} line corresponds to the last game of the regular season.

iv. I use a regular expression to search for a capital letter, followed by two lowercase letters, a comma, a space, a capital letter, two lowercase letters, a space, and then one or more digits. This regular expression is found in *date_exp*. Then I use *grep()* to search *nets1617* for lines with dates in them. These lines are stored in *game.lines*. Looking at the first and last values of *game.lines* I see information on the first and last games.

```
date_exp <- "[A-Z][a-z]{2},\\s[A-Z][a-z]{2}\\s[0-9]+"
game.lines <- grep(date_exp, nets1617)
nets1617[game.lines[1]]
```

```
## [1] "\t\t\t\t<div class=\"mod-page-tabs mod-thirdnav-tabs\" style=\"padding-top: 3px;\"><ul class=\"
```

```
nets1617[game.lines[length(game.lines)]]
```

```
## [1] "<td colspan=\"4\"><a href=\"http://www.stubhub.com/boston-celtics-tickets-boston-celtics-boston-
```

v. *gregexpr()* returns the starting locations and the lengths of each of the game dates, then we can actually extract the information using *regmatches()*. Since the output of *regmatches()* is a list, we use the *unlist()* command to turn it into a vector.

```
date.locations <- gregexpr(date_exp, nets1617[game.lines])
date <- regmatches(nets1617[game.lines], date.locations)
date <- unlist(date)
```

vi. Extracting the game times is similar to extracting the dates, but now my regular expression searches for one or more digits followed by a colon, 2 digits, a space, and then either AM or PM.

```
time_exp <- "[0-9]+:[0-9]{2} (PM|AM)"
time.locations <- gregexpr(time_exp, nets1617[game.lines])
time <- regmatches(nets1617[game.lines], time.locations)
time <- unlist(time)
```

vii. In my solution, I use the fact that in each line, the string `<li class= "game-status ">` appears before the home or away information. So my regular expression searches for `<li class= "game-status ">` followed by '@' or `<li class= "game-status ">` followed by 'vs'. As in part (v) and (vi) I use *gregexpr()* and *regmatches()* to actually extract the strings which match the regular expression. Since these strings include `<li class= "game-status ">` before '@' or 'vs', I then use the *substr()* command just the '@' or the 'vs'. Finally, I create the *home* vector from this information.

```
away_exp <- "<li class=\"game-status\">@|<li class=\"game-status\">vs"
away.locations <- gregexpr(away_exp, nets1617[game.lines])
away <- regmatches(nets1617[game.lines], away.locations)
away <- substr(away, 25, nchar(away))
home <- rep(1, length(away))
home[away == "@"] <- 0
```

viii. In my solution, I use the fact that in each line, the string `<li class=team-name>` appears before the opponent and `</a>` afterwards. So my regular expression searches for `<li class=team-name>` followed by anything inside '<' and '>', letters or space, and `</a>`. *gregexpr()* and *regmatches()* are used to actually extract the strings which match the regular expression. Since these strings include extra information, I use another regular expression to search for the opponent's name recognizing that this will take the form of letters or spaces coming after '>' and before '<'. Finally, we extract just the opponent names using *substr()*.

```
opponent_exp <- "<li class=\"team-name\"><.+>([a-zA-Z]|\\s)+</a>"
opponent.locations <- gregexpr(opponent_exp, nets1617[game.lines])
opponent <- regmatches(nets1617[game.lines], opponent.locations)
opponent <- unlist(opponent)

name_exp <- ">([a-zA-Z]|\\s)+<"
name.locations <- gregexpr(name_exp, opponent)
name <- regmatches(opponent, name.locations)
opponent <- substr(name, 2, nchar(name)-1)
```

ix.

```
schedule <- data.frame(date, time, opponent, home)
schedule[1:10,]
```

```
##           date     time  opponent home
## 1  Wed, Oct 26  7:30 PM    Boston    0
## 2  Fri, Oct 28  7:30 PM   Indiana    1
## 3  Sat, Oct 29  8:00 PM Milwaukee    0
## 4  Mon, Oct 31  7:30 PM   Chicago    1
## 5   Wed, Nov 2  7:30 PM   Detroit    1
## 6   Fri, Nov 4  7:30 PM Charlotte    1
## 7   Tue, Nov 8  7:30 PM Minnesota    1
## 8   Wed, Nov 9  7:00 PM NY Knicks    0
## 9  Sat, Nov 12  9:00 PM   Phoenix    0
## 10 Mon, Nov 14 10:30 PM        LA    0
```