Lab 3

Name: Yi Chen; UNI: yc3356

October 2, 2017

Instructions

Before you leave lab today make sure that you upload a .pdf file to the canvas page (this should have a .pdf extension). This should be the PDF output after you have knitted the file, we don't need the .Rmd file (don't upload the one with the .Rmd extension). The file you upload to the Canvas page should be updated with commands you provide to answer each of the questions below. You can edit this file directly to produce your final solutions. Note, however, in the file you upload you should the above header to have the date, your name, and your UNI. Similarly, when you save the file you should replace **UNI** with your actualy UNI.

Tasks

In this lab, we will use data from homicides in Baltimore City that are collected by the Baltimore Sun newspaper. The data is presented in a map that is publically available at the following website: . I've scraped the data from the website and saved it in a file **BaltimoreHomicides.txt**. Load the data with the following:

```
data <- readLines("BaltimoreHomicides.txt")
```

The data we have corresponds to all the homicides in 2010. There are 224 lines of HTML in the dataset (verify this using **length(data)**) and 224 homicides in 2010.

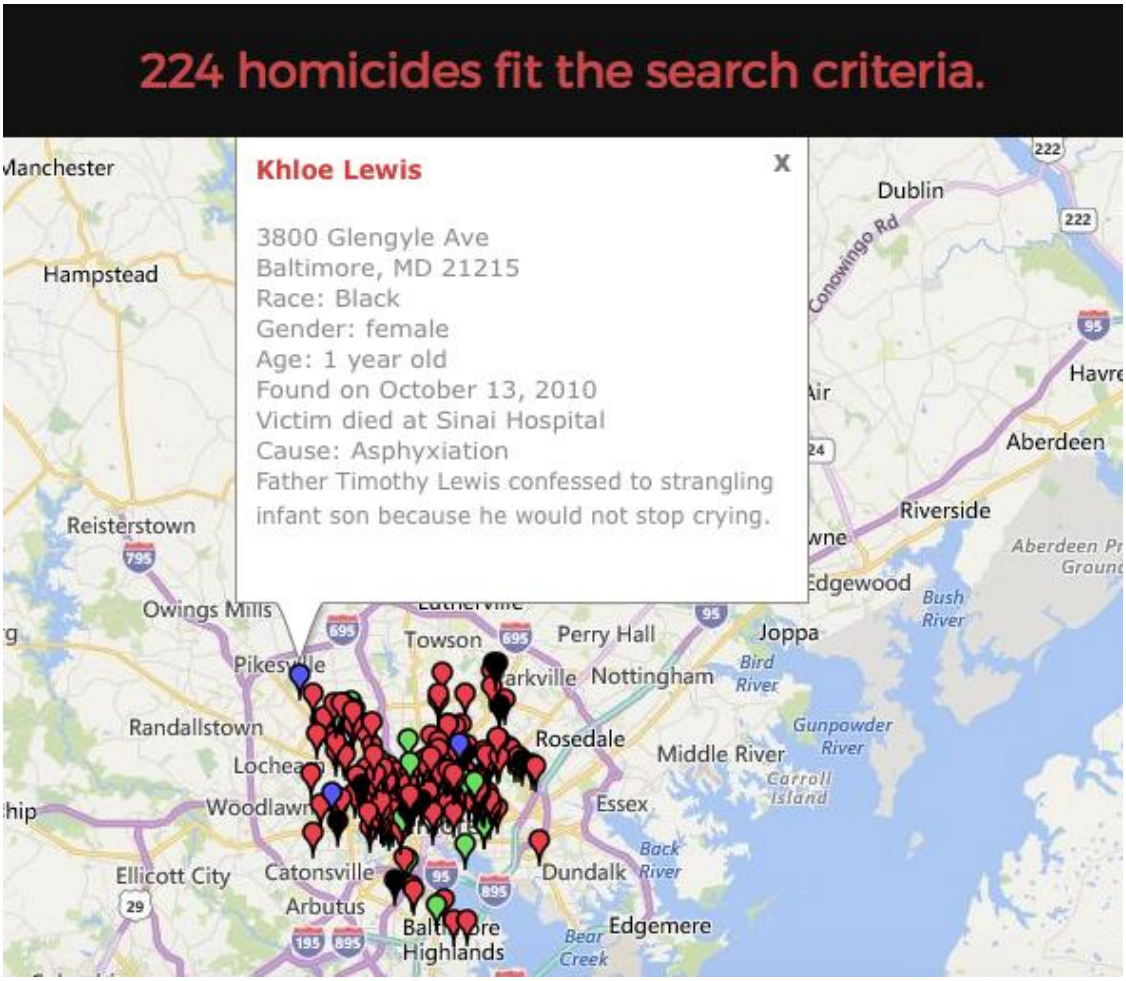
```
# there are two way to calculate the number of lines in this field.
number_of_lines_1 <- length(data)
number_of_lines_2 <- summary(data)
#test
number_of_lines_2[1] == number_of_lines_1

## Length
##   TRUE

cat('there are ', number_of_lines_1,'lines')

## there are  224 lines
```

(1) In the image below, you can see one of the homicides in our dataset. Use a **grep()** call to find which row in the dataset corresponds to the death of Khloe Lewis. (You could search, for example, for her name and the row you should return is 52).



image

```
grep('Khloe Lewis',data)
```

```
## [1] 52
```

(2) Suppose we wanted to identify the records for all the victims of shootings (as opposed to other causes). Using the following bit of code and some of your own exploration, how many of the homicides in our dataset are the result of shootings?

```
# search the 'shooting' information in the data(shooting is begin with a
Lower case letters)
deaths1 <- grep("shooting", data)
# search the 'Shooting' information in the data(shooting is begin with a
uppercase letter)
deaths2 <- grep("Shooting", data)
length(deaths1)
```

```
## [1] 172
```

```
length(deaths2)
```

```
## [1] 171
```

```r
setdiff(deaths1, deaths2) # there are one more element in death1.
```

```
## [1] 105
```

```r
cat('there are:',length(deaths1),'homicides are result of shootings.')
```

```
## there are: 172 homicides are result of shootings.
```

```r
# this problem can also be solved in this way
regular <- regexpr("[s,S]hooting", data)
result<- regmatches(data, regular)
length(result)
```

```
## [1] 172
```

```r
cat('there are:',length(result),'homicides are result of shootings.')
```

```
## there are: 172 homicides are result of shootings.
```

- **explain**

1. when the data is write in the file, sometimes people write 'Shooting' and 'shooting' for the same thing. Just one begin with **lower case letter** while one begin with **uppercase letter**.
   Thus, we need to find both these two kind of information all together.
2. *death1* and *death2* have a lot of information in common.use **setdiff** function, which only return the different element in both vectors. As we can see there are only one elements different in these two results. death1 have one more element.

(3) The following code creates a vector of the ages of the homicide victims in the dataset:

```r
# Hi teacher your code is wrong, you miss the data of the victims who age is
one year!!! otherwise the number of year is not equal to the number of names
r       <- regexpr(">Age:\\s.*year[s]* old<", data)
age_vec <- regmatches(data, r)
age_vec <- substring(age_vec, 2, nchar(age_vec) - 1)
head(age_vec)
```

```
## [1] "Age: 50 years old" "Age: 25 years old" "Age: 20 years old"
## [4] "Age: 23 years old" "Age: 14 years old" "Age: 43 years old"
```

(a) Explain what the regular expression ">Age:\s.*years old<" searches for.

1. Regular expression describes a pattern of specific string which can be used to match and check whether another string contains this particular substring.
2. ">Age:\s.*years old<" is this specific pattern, which means this string being with '>Age:' and end with 'years old<'.
3. "\s.*" have two many parts: \s means Blank character, asterisk means repeated zero or more times
4. in this way all the information about age in the patter of **>Age: (some blank space) years old<** can be matched.

(b) Explain in words what the output of the first line of code provides.

1. **regexpr** will returns the location of the rst match with attributes like the length of the match. **data** is the main string where it will match the regular expression. The result of this function can help to find the location of the substring of the age information.

(c) The second line of code.

1. use **regmatches** we can then extract the strings and the output of regexpr() and returns the actual matching strings.

(d) The third line of code.

1. **nchar()** is the function takes a character vector as an argument and returns a vector whose elements contain the sizes of the corresponding elements of x. In this way we can calculate the total number of characters in every string.
2. every element in the age_vec begin with ">" and end with "<", it is not what i need.
3. by using **substring** i can extract certain substring from the given string. In this way for every elements in age_vec, we substring from the second character to the second last character. Thus, "<" and ">" have been excluded.

(4) Use the same strategy as we used in question (3) to create a vector holding each victims' name. Hint: I had to use the fact that the first letter of the name is capitalized and the specific structure of the HTML code after the name in writing my regular expression.

```
# creat the regular expression begin with > and then a capitalized letter and
several low case letter and end with </a></dt>
r_of_name <- regexpr("[A-Z][a-z]*\\s.*</a></dt>", data)
name_vec  <- regmatches(data, r_of_name)
# exclud the useless information
name_vec  <- substring(name_vec,1,nchar(name_vec)-9)
head(name_vec)

## [1] "Bernard Clowney"   "David King"        "Raymond Woodland"
## [4] "Keith L. Robinson" "Issac Joyner"      "Mustafa Malik"
```