Homework 6 – Due Monday October 26, 2015

Cheryl Calhoun

10/19/2015

Assignment

The Gettysburg Address Text Analysis

This is the entirety of the Gettysburg Address. It is loaded here as a single string.

```
gettysburg <- "Four score and seven years ago our fathers brought forth on this
continent, a new nation, conceived in Liberty, and dedicated to the propositio
n that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any na
tion so conceived and so dedicated, can long endure. We are met on a great batt
le-field of that war. We have come to dedicate a portion of that field, as a fi
nal resting place for those who here gave their lives that that nation might li
ve. It is altogether fitting and proper that we should do this.

But, in a larger sense, we can not dedicate -- we can not consecrate -- we can
not hallow -- this ground. The brave men, living and dead, who struggled here,
have consecrated it, far above our poor power to add or detract. The world wil
l little note, nor long remember what we say here, but it can never forget wha
t they did here. It is for us the living, rather, to be dedicated here to the u
nfinished work which they who fought here have thus far so nobly advanced. It i
s rather for us to be here dedicated to the great task remaining before us -- t
hat from these honored dead we take increased devotion to that cause for which
they gave the last full measure of devotion -- that we here highly resolve tha
t these dead shall not have died in vain -- that this nation, under God, shall
have a new birth of freedom -- and that government of the people, by the peopl
e, for the people, shall not perish from the earth."
```

We will use `gregexpr()` and `regmatches()` to extract:

All capitalized words that are not at the beginning of sentences.

```
## Find all words starting with a capital letter that are not preceeded by a sp
ace and either a period or new line character.
capmatches <- gregexpr("(?<!\\.|\\n)\\s[A-Z][a-z]*", gettysburg, perl="TRUE")
regmatches (gettysburg, capmatches)
```

```
## [[1]]
## [1] " Liberty" " God"
```

All nine-letter words.

```
## Find all words that contain exactly 9 uppercase and or lowercase letters.
nineletterwords <- gregexpr("[^A-z][A-z]{9}[^A-z]", gettysburg)
regmatches (gettysburg, nineletterwords)
```

```
## [[1]]
##  [1] " continent," " conceived " " dedicated " " conceived " " dedicated,"
##  [6] " struggled " " dedicated " " dedicated " " remaining " " increased "
```

The number of times the word “nation” appears.

```
## Find all occurences of the word "nation".
nation <- gregexpr("nation", gettysburg)
nation <- regmatches (gettysburg, nation)
##length(nation)
```

The number of sentences.

```
## Find the number of sentances using the period "\." as the break between sent
ances.
sentances <- gregexpr("\\.", gettysburg)
regmatches (gettysburg, sentances)
```

```
## [[1]]
##  [1] "." "." "." "." "." "." "." "." "." "."
```

```
## length(sentances)
```

The number of paragraphs in the phrase.

```
## Find the number of paragraphs using the new line character "\n" as the indic
ator of a break between paragraphs.
capmatches <- gregexpr("\\n\\n", gettysburg)
regmatches (gettysburg, capmatches)
```

```
## [[1]]
## [1] "\n\n" "\n\n"
```

Results: R is using the end of line character `\n` to end the line and then `\n` again to create a blank line between paragraphs. Using the end of line character, we can locate two paragraph breaks, which indicates three total paragraphs for the Gettysburg Address.

The Emancipation Proclamation Text Analysis

Loading the Emancipation Proclamation from http://www.acthomas.ca/FSSS/data/emancipation.txt (http://www.acthomas.ca/FSSS/data/emancipation.txt).

```
emancipation <- readLines ("http://www.acthomas.ca/FSSS/data/emancipation.txt")

## Finding the number of blank lines in this file.
blanklines <- gregexpr("[^[:blank:]]*$", emancipation)
blanklines <- unlist (regmatches (emancipation, blanklines))
```

Results: There are 41 lines of text in the file, 21 of these lines are blank.

Use `gregexpr()` and `regmatches()` to extract:

All capitalized words that are not at the beginning of sentences.

```
## Find all words starting with a capital letter that are not preceeded by a sp
ace and either a period or new line character.
capmatches <- gregexpr("(?<!\\.|\\n)\\s[A-Z][a-z]+", emancipation, perl="TRUE")
capmatches <- unlist (regmatches (emancipation, capmatches))
capmatches
```

```
##   [1] " Text"          " Emancipation"  " Proclamation"  " President"
##   [5] " Abraham"       " Lincoln"       " Version"       " Emancipation"
##   [9] " Proclamation"  " Transcription" " President"     " United"
##  [13] " States"        " America"       " Proclamation"  " September"
##  [17] " Lord"          " President"     " United"        " States"
##  [21] " January"       " Lord"          " State"         " State"
##  [25] " United"        " States"        " Executive"     " Government"
##  [29] " United"        " States"        " Executive"     " January"
##  [33] " States"        " States"        " United"        " States"
##  [37] " State"         " Congress"      " United"        " States"
##  [41] " State"         " State"         " United"        " States"
##  [45] " Abraham"       " Lincoln"       " President"     " United"
##  [49] " States"        " Commander"     " Army"          " Navy"
##  [53] " United"        " States"        " United"        " States"
##  [57] " January"       " Lord"          " States"        " States"
##  [61] " United"        " States"        " Texas"         " Louisiana"
##  [65] " Parishes"      " St"            " Plaquemines"   " Jefferson"
##  [69] " St"            " St"            " St"            " Ascension"
##  [73] " Assumption"    " Terrebonne"    " Lafourche"     " St"
##  [77] " St"            " Orleans"       " City"          " New"
##  [81] " Orleans"       " Mississippi"   " Alabama"       " Florida"
##  [85] " Georgia"       " South"         " Carolina"      " North"
##  [89] " Carolina"      " Virginia"      " West"          " Virginia"
##  [93] " Berkley"       " Accomac"       " Northampton"   " Elizabeth"
##  [97] " City"          " York"          " Princess"      " Ann"
## [101] " Norfolk"       " Norfolk"       " Portsmouth"    " States"
## [105] " States"        " Executive"     " United"        " States"
## [109] " United"        " States"        " Constitution"  " Almighty"
## [113] " God"           " United"        " States"        " City"
## [117] " Washington"    " January"       " Lord"          " Independence"
## [121] " United"        " States"        " America"       " President"
## [125] " Secretary"     " State"
```

All fully capitalized names.

```
## Find all fully capitalized names.
capnames <- gregexpr("([A-Z][A-Z]+ ?[A-Z]?[\\.]? [A-Z][A-Z]+)+", emancipation)
capnames <- unlist (regmatches (emancipation, capnames))
capnames
```

```
## [1] "ABRAHAM LINCOLN"   "WILLIAM H. SEWARD"
```

All nine-letter words.

```
## Find all words containing exactly nine-letters.
nineletterwords <- gregexpr("[^A-z][A-z]{9}[^A-z]", emancipation)
nineletterwords <- unlist(regmatches (emancipation, nineletterwords))
nineletterwords
```

```
##  [1] " President " " President " " September," " President " " following,"
##  [6] " rebellion " " Executive " " including " " authority " " recognize "
## [11] " Executive " " aforesaid," " designate " " rebellion " " elections "
## [16] " qualified " " testimony," " rebellion " " therefore " " President "
## [21] " Commander-" " rebellion " " authority " " necessary " " rebellion,"
## [26] " mentioned," " designate " " rebellion " " following," " Louisiana,"
## [31] " Jefferson," " Ascension," " Lafourche," " including " " Elizabeth "
## [36] " including " " precisely " " aforesaid," " Executive " " including "
## [41] " recognize " " necessary " " recommend " " condition," " positions,"
## [46] " sincerely " " warranted " " necessity," " President:" " Secretary "
```

All Saints mentioned by name.

```
## Find all Saints mentioned by name. Search for St. at the beginning of the na
me.
saints <- gregexpr(" St\\. [A-z]*", emancipation)
saints <- unlist(regmatches (emancipation, saints))
saints
```

```
## [1] " St. Bernard" " St. John"    " St. Charles" " St. James"
## [5] " St. Mary"    " St. Martin"
```

The Top 100 Grossing Movies Analysis

We will use the following data set as a source file for the Top 100 grossing movies: [http://www.boxofficemojo.com/alltime/world/ (http://www.boxofficemojo.com/alltime/world/)]

Here is an extraction of the HTML code for the second-highest grossing movie:

```
<tr bgcolor="#f4f4ff"><td align="center"><font size="2">2</font></td>
<td><font size="2"><a href="/movies/?id=titanic.htm"><b>Titanic</b></a></font>
</td>
<td><font size="2">Par.</font></td>
<td align="right"><font size="2"><b>$2,186.8</b></font></td>
<td align="right"><font size="2">$658.7</font></td>
<td align="right"><font size="2">30.1%</font></td>
<td align="right"><font size="2">$1,528.1</font></td>
<td align="right"><font size="2">69.9%</font></td>
<td align="center"><font size="2">1997^</font></td>
</tr>
```

```
## Load the raw HTML in from http://www.acthomas.ca/FSSS/data/boxoffice.html:
boxoffice <- readLines ("http://www.acthomas.ca/FSSS/data/boxoffice.html")
```

Results: There are 1103 raw lines of text in this file.

First, we'll find the lines that have the movie titles in them. We will do this by selecting a piece of text that is unique to the title lines in each case (but not specific to one movie). In this data set, there is an “a href” statement with a URL starting in “/movies/” on the line corresponding to the movie title. We can use the “movies” string to isolate the title line.

```
## Finding the files that have "movies" in the URL.
movies <- grep ("movies", boxoffice, value="TRUE")
```

Results: There are 100 movies in this file.

With this new vector of 100 lines, we will devise a search term that cuts down to as much of the name of the movie as possible.

```
## movienames = gregexpr("(?<!<b>)([[:alnum:]]+.*[A-Z][[:alnum:]]+)+", movies,
perl="TRUE")
movienames = gregexpr("[A-z0-9:(&)'.,;\\- ]+(?=<)", movies, TRUE, perl="TRUE")
movielist <- unlist (regmatches (movies, movienames))[1:100]
movielist[1:10]
```

```
##  [1] "Avatar"
##  [2] "Titanic"
##  [3] "Jurassic World"
##  [4] "Marvel's The Avengers"
##  [5] "Furious 7"
##  [6] "Avengers: Age of Ultron"
##  [7] "Harry Potter and the Deathly Hallows Part 2"
##  [8] "Frozen"
##  [9] "Iron Man 3"
## [10] "Minions"
```

We still have a trailing “<” tag which we need to remove.

```
movielist <- gsub("<","", movielist)
movielist[1:10]
```

```
##  [1] "Avatar"
##  [2] "Titanic"
##  [3] "Jurassic World"
##  [4] "Marvel's The Avengers"
##  [5] "Furious 7"
##  [6] "Avengers: Age of Ultron"
##  [7] "Harry Potter and the Deathly Hallows Part 2"
##  [8] "Frozen"
##  [9] "Iron Man 3"
## [10] "Minions"
```

Results: There are 100 year records (rows) in the `movielist` file.

Now for the year. We’ll select a piece of text that is unique to the year line in each case. In this case we are going to look for `<font size = "2">`. This will help us to eliminate the lines where the movie title contains a year. We will then look for `[0-9]{4}` to represent the year.

```
years <- grep ("<font size=\"2\">[0-9]{4}", boxoffice, value="TRUE")
yearslist = gregexpr("[0-9]{4}", years, perl="TRUE")
yearlist <- unlist (regmatches (years, yearslist))
yearlist[1:10]
```

```
##  [1] "2009" "1997" "2015" "2012" "2015" "2015" "2011" "2013" "2013" "2015"
```

Results: There are 100 year records (rows) in the `yearslist` file.

Finally, we’ll isolate the worldwide gross, by selecting on the `<b>` tags.

```
gross <- grep ("<b>\\$", boxoffice, value="TRUE")
grosslist = gregexpr("\\$[0-9]?,?[0-9]{3}\\.[0-9]{1}", gross, perl="TRUE")
grosslist <- unlist (regmatches (gross, grosslist))
grosslist[1:10]
```

```
##  [1] "$2,788.0" "$2,186.8" "$1,665.5" "$1,519.6" "$1,511.7" "$1,402.8"
##  [7] "$1,341.5" "$1,274.2" "$1,215.4" "$1,153.1"
```

Results: There are 100 gross earnings records (rows) in the `grosslist` file.

The final data frame is creatied by combining the three lists. Below is the top 10, the middle 10 (46-55) and the bottom 10 (91-100) on the list.

```
## Creating a data frame from the extracted files.
full.movies <- data.frame (movies = movielist,
                           year = yearlist,
                           gross = grosslist,
                           stringsAsFactors=FALSE)
full.movies[1:10, ]
```

```
##                                        movies year    gross
## 1                                      Avatar 2009 $2,788.0
## 2                                     Titanic 1997 $2,186.8
## 3                              Jurassic World 2015 $1,665.5
## 4                       Marvel's The Avengers 2012 $1,519.6
## 5                                   Furious 7 2015 $1,511.7
## 6                     Avengers: Age of Ultron 2015 $1,402.8
## 7  Harry Potter and the Deathly Hallows Part 2 2011 $1,341.5
## 8                                      Frozen 2013 $1,274.2
## 9                                  Iron Man 3 2013 $1,215.4
## 10                                    Minions 2015 $1,153.1
```

```
full.movies[46:55, ]
```

```
##                                               movies year  gross
## 46           The Twilight Saga: Breaking Dawn Part 2 2012 $829.7
## 47                                         Inception 2010 $825.5
## 48                                        Spider-Man 2002 $821.7
## 49                                  Independence Day 1996 $817.4
## 50                                   Shrek the Third 2007 $799.0
## 51          Harry Potter and the Prisoner of Azkaban 2004 $796.7
## 52                        E.T.: The Extra-Terrestrial 1982 $792.9
## 53                               Fast &amp; Furious 6 2013 $788.7
## 54 Indiana Jones and the Kingdom of the Crystal Skull 2008 $786.6
## 55                                      Spider-Man 2 2004 $783.8
```

```
full.movies[91:100, ]
```

```
##                                movies year  gross
## 91                      Kung Fu Panda 2008 $631.7
## 92                    The Incredibles 2004 $631.4
## 93                          Fast Five 2011 $626.1
## 94                            Hancock 2008 $624.4
## 95                              MIB 3 2012 $624.0
## 96                         Iron Man 2 2010 $623.9
## 97                        Ratatouille 2007 $623.7
## 98     How to Train Your Dragon 2 2014 $618.9
## 99  The Lost World: Jurassic Park 1997 $618.6
## 100     The Passion of the Christ 2004 $611.9
```