STA 141A
Fundamentals of Statistical Data Science

Fall 2016

Instructor: Debashis Paul

Lecture 13

Information on Project

- Proposal due date (extended due to Veterans' Day holiday): **November 15 (Tuesday)**.
- Project Proposal will not be graded. Needs to be submitted electronically through smartsite (announcement will be sent).
- Project submission due date: **December 8 (Thursday).**
- Each project can be done by at most four students. Contribution of each group member must be clearly indicated in the final report.

Structure of Project Proposal

1. Name and UCD Email ID of all the group members.
2. A brief description of the project, including an overall goal.
3. A brief description of the data to be used in the project, including the sources.
4. A list of 2-3 key questions that are to be addressed during the project.
5. Methodologies to be used. This may include specific computational and/or statistical procedure, specific software packages (not necessarily in R), etc.
6. The whole proposal should be at most 1 page long.

Introduction to Text Processing

- Text processing tasks are ubiquitous. A few areas of applications: written document processing, extracting information from the genome, email filtering, online search engines.
- A typical task of text processing is through matching a *pattern* with a piece of text (character string, possibly very long and structured).
- The pattern could be a character string or, more generally, a **regular expression**, which is effectively a class or set of character strings defined through a set of rules.
- Regular expression can also be viewed as a language, with its own vocabulary and grammar.
- We shall use regular expression, together with *string splitting* and *substitution* features in base R package and the more specialized **stringr** package to perform text processing tasks.

Splitting and pasting strings

- We can use strsplit( ) function in R to split a string, and use paste( ) function to put together several character strings.

```
atxt = "Text processing is here to stay! A new lesson starts here!"
atxt.lines = strsplit(atxt,split="!")  # break atxt into two sentences using splitting by "!"
atxt.words = lapply(atxt.lines[[1]],strsplit,split=" ")  # extract individual words
#(including blank space in front of second sentence)
atxt.wordvec = unlist(atxt.words) # creates just a vector of words
atxt.wordvec = atxt.wordvec[atxt.wordvec !=""]  # remove the blank space
atxt.nopunc = paste(atxt.wordvec,collapse=" ") # put the words back together (without punctuation)
```

Functions used to find match

- Base R functions grep( ), grepl( ), regexpr( ), gregexpr( ) and regexec( ) search for matches to a pattern within each element of a vector of character strings. They differ in the format and in the amount of detail in the results.
- regmatches( ) can be used to extract or replace matched substrings from match data found by regexpr( ), gregexpr( ) or regexec( ).
- sub( ) and gsub( ) perform replacement of the first and all matches, respectively, with a given pattern.
- All these functions use regular expression as arguments, special cases being specific character strings. They also allow Perl-type features in the match search.

Using grep( ) : examples

- grep( ) can be used to find the indices of match (or non-match) within a vector of character strings.

```
## Usage:
# grep(pattern, x, ignore.case = FALSE, perl = FALSE, value = FALSE,
#                fixed = FALSE, useBytes = FALSE, invert = FALSE)
grep("new", atxt.wordvec)  # returns indices of all the words in atxt that match with “new”
 grep("new", atxt) # returns 1 since the word “new” appears in atxt
grep("new", atxt.lines[[1]])  # returns 2 since “new” only appears in second sentence
grep("new", atxt.lines[[1]], value=T) # returns the matched second sentence
grep("new", atxt.lines[[1]], invert=T, value=T) # returns the non-matched first sentence
```

Using sub( ) and gsub( )

- sub( ) replaces the first match with a pattern, while gsub( ) replaces all the matches.

```
sub("new","old",atxt) # replaces “old” with “new” in atxt
sub("new","old",atxt.lines[[1]]) # replaces in matched string; leaves alone the non-matched string
btxt.wordvec = c("Text", "is", "repeated", "here", ";", "text", "again",
                 ",", "and", "more", "text", "!", "3", "times","!")
btxt = paste(btxt.wordvec, collapse=" ")
sub("text","rain",btxt, ignore.case=T) # only replaces the first occurrence of "text" with "rain”
gsub("text","rain",btxt, ignore.case=T) # replaces all the occurrences of "text" with "rain”
sub("text","rain",btxt, ignore.case=T) # every single word “text” in the vector is replaced by “rain”
```

Regular expression

- For general rules on using regular expression, check the document "Intro to Regular Expression in R" on smartsite.
- For a thorough overview of the grammar, check Nick Ulle's notes for the discussion section on 11/04/16.
- Online resources include regexr.com. This site has an interactive text box that can be utilized as a test bed for checking the matches for a given regular expression. Also has a short tutorial.
- A brief overview of using regular expression in R versus RStudio:

  https://www.r-bloggers.com/regular-expressions-in-r-vs-rstudio/