Beginning Text Preparation

In order to perform text analysis, there are a few R commands you should have up your sleeve. Some of the commands help get you set up and locate all of the files in your corpora. Other commands can be used throughout the programming process to check on your algorithm and make sure everything looks the way you think it should. Learning the following commands will give you a brief introduction to R while also setting you up with a solid toolkit to begin programming.

Set Working Directory Your working directory is the folder from which you will be pulling your texts.

Set your working directory by pointing to the location on your computer where you have stored the files. The syntax for R is as follows: "setwd" followed by a set of parentheses () and double quotes "". Between the double quotes is the path to your directory.

Below, we have chosen to Save the folder "RAnalysis" on the Desktop on a Mac. It contains all the other R scripts, texts, notebooks, and results. If you have branched the IU Text Analysis Github, simply note where you have save the folder. If you are on a PC, you will need to use an absolute path such as "C:Users:XXX."

Hint: Can't figure out your syntax? Click Session > Set Working Directory > Choose Directory, then select the Text_Analysis directory in which you are working. This will set your working directory in the console below while you are working here, but make sure to copy the path into the "setwd" command below to keep the directory constant if you close this script and reopen later. ***

```
setwd("~/Desktop/R/Text_Analysis/RNotebooks")
```

Install Packages R's extensibility comes in large part from packages. Packages are groups of functions, data, and algorithms that allow users to easily carry out processes without recreating the wheel. Some packages are included in the basic installation of R, others created by R users are available for download.

To install any package, you can click Tools > Install Packages, begin typing the package you wish to install, select the package name, and click Install. To use the Console in RStudio, you can put your cursor at the carrot and type the following (with any package name substituting for "knitr":

install.packages("knitr")

Load Packages In order to access the packages you have installed within the environment in which you are currently working, you must load them at the beginning of your script. To load packages, use the library() command (see code below).

The three packages listed are used to render the RNotebook you are currently viewing:

knitr - Creates a formatted report from the script provided

markdown - a package used to render textual documents from plain text to others such as R and XML

rmarkdown - similar to markdown but specifically to render R documents

```
library(knitr)
library(markdown)
library(rmarkdown)
```

To perform topic modeling, you may load MALLET or tm. For data visualization, a popular package is ggplot2, or perhaps hclust to create a cluster dendrogram. You can peruse the various contributed packages here. The tutorials included in the R Toolkit will instruct you which packages to install and load.

Load data Now you are ready to start looking at your data! First, you must load it into your environment. The scan() function will do this for you. If you want to load just one text into your environment, here is the syntax:

The first argument is the filename (or path if the file resides in a different directory than your working directory). The second argument "what" specified as type "character" will read the text in as a character vector. The third argument "sep" specified as "backslash" + "n" which is the way to code line breaks in R.

So putting everything together, this line reads in JaneEyre.txt, and separates the text into a character vector by line.

```
text <- scan("~/Desktop/R/Text_Analysis/data/bronte/janeEyre.txt", what="character", sep="\n")
```

The final, crucial aspect of this line is the assignment. "<-" assigns whatever results on the right side of the arrow into the variable specified on the left side. Some programming languages use "=" instead of the arrow. R will also acknowledge this, but using the arrow is best practice.

Here, we have named that variable "text" since it holds the text with which we are working. However, you can name this variable whatever you would like. This line will give the exact same result, although it is best to name the variable in relation to what it holds:

```
potatoes <- scan("~/Desktop/R/Text_Analysis/data/bronte/janeEyre.txt", what="character", sep="\n")
```

Now that we have the text saved as a variable, we can reuse that variable simply by calling "text" instead of the entire scan line again. The next few commands will use "text" to explore the data.

R Objects R is distinct from other programming languages in that it handles objects a little differently. Throughout text analysis, you will need to massage your text and textual data by changing it into various kinds of objects which make things easier. Check out this tutorial by Neal Groothuis which simply explains the various types of data objects. Since some kinds of objects prohibit certain actions, a simple way to check the type of object you are currently working with is class(). For example, the code below shows that the class of the Jane Eyre "text" is a character vector, which we would assume since that is what we specified while loading it in.

```
class(text)
```

```
## [1] "character"
```

Data Inspection Just as you may want to verify the type of the object with which you are working, you may want to view it from other angles to make sure the data is formed as you expect.

The length function shows the number of elements within an object. If you find that the length is zero, you may have to go back and reload the data, or check to make sure your algorithm is working correctly.

```
length(text)
```

```
## [1] 16706
```

We can also look at individual elements - Lets see what the first line of our text is...

```
text[1]
```

```
## [1] "The Project Gutenberg eBook, Jane Eyre, by Charlotte Bronte, Illustrated"
```

Or perhaps you would like to see the first few elements:

```
head(text)
```

```
## [1] "The Project Gutenberg eBook, Jane Eyre, by Charlotte Bronte, Illustrated"
## [2] "by F. H. Townsend"
## [3] "This eBook is for the use of anyone anywhere at no cost and with"
## [4] "almost no restrictions whatsoever.  You may copy it, give it away or"
## [5] "re-use it under the terms of the Project Gutenberg License included"
## [6] "with this eBook or online at www.gutenberg.org"
```

Or last few elements:

```
tail(text)
```

```
## [1] "Most people start at our Web site which has the main PG search facility:"
## [2] "     http://www.gutenberg.org"
## [3] "This Web site includes information about Project Gutenberg-tm,"
## [4] "including how to make donations to the Project Gutenberg Literary"
## [5] "Archive Foundation, how to help produce our new eBooks, and how to"
## [6] "subscribe to our email newsletter to hear about new eBooks."
```

There are many more ways to inspect parts of data (check out the CRAN), but these quick checks are helpful while manipulating the data and debugging the inevitable issues you will encounter while developing your script.

Explore The above commands are a few tips and tricks to get you started with R. Similar to R's extensibility with packages, the R user community has great resouces for learners. The CRAN FAQ and the CRAN Manual answers quite a few questions about R and its uses.

Googling the issue, function, or object name with "r" will return extremely helpful resources. If a PDF from cran.r-project.org appears, there you will find extensive documentation and examples for that function, etc. and other related resources. Similarly, any result from r-bloggers.com will most likely be helpful. For any other issues, Stack Overflow is helpful to find answers to common questions as well as ask your own.

The rest of the IU tutorials explain some methods for textual analysis using R. If you are ready to dive in, click on one to begin!