First Semester: Text Mining/Digital Humanities in R

Ben Williams

Goals:

* Become comfortable with basic programming in R
* Be able to load data, especially texts, into R
* Be able to clean data, especially texts, in R
* Be able to calculate basic text mining statistics from texts, e.g. word counts, sentiments, etc.
* Be able to make various plots with these text mining outputs

Vision:

* This course is an introduction to basic text mining techniques using the power of computers. The goal is for students to be able to implement text mining on their own texts of interest. No programming experience is required, and the students will be walked through all aspects of programming for text mining. The programming language used here is the free and open source software **R**. There are other programming languages that can be used for text mining, but for now we will focus on **R**. At the end of this course, students will be able to find a text of their choice, load it into R, and derive knowledge from it using text mining techniques.

Tools:

* **R**
* **RStudio**
* Your own laptop, brought to class fully charged each day

Books:

* [Text Mining with R: A Tidy Approach](http://tidytextmining.com/), by *Julia Silge and David Robinson (TMR)*
* [R for Data Science](http://r4ds.had.co.nz/), by *Garrett Grolemund and Hadley Wickham* (helpful but not required) *(R4DS)*
* [The Hitchhikers Guide to ggplot2 in R](https://leanpub.com/hitchhikers_ggplot2), by Jodie Burchell and Mauricio Vargas (helpful but not required, it is available for free on leanpub) (*HGGPLOT2*)

Tentative Schedule

Week 1: Introduction

-What to expect in this class

-typing a lot of code

-being frustrated that the code doesn’t work

-being elated when the code does work

-pushing yourself to learn

-How to succeed in this class

-be organized (data, code)

-make thorough comments on your code so you can go back to it in the future or show it to

others and have them understand your code

-don’t copy/paste (unless told!) the code, write it out yourself

-figure out issues with your code on your own – google, stack overflow, built-in help files in **R**

Activities during class:

* Make a folder for this class, learn how to have an organized directory system
* Download and install R and R-Studio
* Write your first line of code (“Hello World!”)

Key commands/syntax to know:

* print()
* <-

HW:

* Make flashcards of the key commands/syntax above, so you have a working definition of them

Week 2: Preparing to code

-What is **R** and why is it so great?

-**R** is an open source programming language. Open sourcemeans that anyone can contribute to **R** to make it better/more specific (think Wikipedia).

-The way in which we access all this open source help is through packages. A *package* is something that the user must install into the **R** program they run on their personal machine. When you first download **R**, think of it as though you are buying (except it is free) the very basic model of a car. When you want to make upgrades to the car, like a stereo or SMU-themed seat cover, you have to install them after you obtain the base model. Similarly, each new **R** *package* you install is an upgrade to base **R**, just like the seat covers that you installed in your new car.

-The packages are downloaded from something called CRAN. CRAN stands for Comprehensive **R** Archive Network. You won’t actually interact with CRAN, but it is nice to know where these packages come from. Anyone can upload a package they have created to CRAN so that anyone can download it and use it. For example, we will download a package to do text-mining in **R**, since the specific text-mining functions we will use are not included in base **R**.

-The fact that **R** is open-source is one reason it is so great. It is free and there are so many packages to perform a myriad of tasks. It is mostly used for statistical or data science purposes, perfect for us in this class.

-What is **RStudio**?

-**RStudio** is an IDE for **R**. IDE stands for Integrated Development Environment. **RStudio** provides us with an intuitive, visually pleasing environment to use the **R** programming language. Think of how Microsoft Word is a nicer and more intuitive way to type documents than in a regular notepad document.

Activities during class:

* Explore the differences between **R** and **RStudio**.
* Explore the functionality of RStudio (what are these quadrants all about?)
* Install and load packages both with lines of code and by “point and click” method
* Example of a plot with base R and with a package to improve plotting

Key commands/syntax to know:

* install.packages()
* library()
* str()
* head()
* tail()
* $
* summary()

HW:

* Download and install the package *tidytext*. Read the preface and pages 1-11 of TMWR. Bring your own definition of what text mining is to the next class.
* Make flashcards of the key commands/syntax above, so you have a working definition of them

Week 3 An introduction to text mining and tidy data

-What is text mining?

-Text mining is the process of trying to understand a text via various metrics and generally uses computer technology. Text mining is thus different from close reading. In close reading, we might be interested in an ironic statement made by a character in reference to something that happened at the beginning of the book, while in text mining we might want to know what the most common word that character uses throughout the book is. Both text mining and close reading are valuable. Text mining allows us to think differently about texts and come up with novel conclusions. Additionally, we can use text mining techniques over a vast amount of text in a very small amount of time thanks to computational power. The computer can read much faster than even the quickest speed-reader and so we can make conclusions about a large body of text much faster than if we read it.

-What is tidy data?

-Tidy data is a term coined by Hadley Wickham (Wickham 2014), and refers to datasets in which each variable is a column, each observation is a row, and each type of observational unit is a table. Suppose you sat outside and wanted to know how many people passed you every minute and if they were walking their dog or not. Each minute would be an observation, and so each minute would correspond to each row in the dataset you would create. You have 2 variables, the first is the number of people that pass you, and the second is whether or not they were walking (described by 0 or 1 for no and yes, respectively).

-What is tidy text data?

-Similar to tidy data, tidy text data is a dataset with one-token-per-row. A token is some unit of interest, and can be a word, two words next to each other, a sentence, etc.

Activities during class:

* Explore [Google n-grams](https://books.google.com/ngrams) (What is an n-gram?)
  + Google n-grams tells us how word counts vary over time in the Google books corpus
  + There are various arithmetic tools to use within n-grams (multiplication, division) that will be shown in class
* Work through code on pages 1-3 in TMR

Key commands/syntax to know:

* c()
* data\_frame()
* %>%
* unnest\_tokens()
* tibble

Homework:

* Make flashcards of the key commands/syntax above, so you have a working definition of them
* Come up with some example of your choosing using google n-grams
* Rework the code from pages 1-3 with a different poem that you make up
* Write a new **R** script with the code from pages 4-10 written verbatim to go through next week

Week 4: Let’s do some Text Mining and include Sentiment Analysis

-How was the homework? Are there any questions. Let’s see 2 student examples.

-Recap: what is tidy data? What is tidy text data? What is a package?

-What is sentiment analysis?

-Sentiment analysis is the method of analyzing texts by analyzing the sentiment or emotion associated with the words used in the text. Usually we use a pre-specified lexicon where words are already tagged with emotion and maybe the magnitude of the emotion.

-What does word frequency mean?

- Word frequency is the number of times a specific word occurs in a piece of text, book, corpus, etc. Studying word frequencies is straightforward and easy to interpret, yet is a vital piece of text-mining.

-What does # mean in R?

-It works differently from Twitter! In an **R** script using # tells R to not “read” whatever follows the # on the rest of the line of code. We use # to make comments on our code within the script. This is a critical coding practice because you may do something in your code that makes perfect sense to you at the current time, but in 3 months when you open the script again, you may not remember. Additionally, if you are collaborating with others, it is important to make sure they understand your thinking. I like to make comments to remind myself of my next step, say what the last line of code I wrote does, and keep a record of issues I still need to solve.

Activities during class:

* Load code into R from last week
* Talk about joins
  + Joining is how we combine 2 datasets. There are a few different methods to join datasets, we can keep all observations from both datasets or just portions. Thinking about venn-diagrams will be helpful! See <http://r4ds.had.co.nz/relational-data.html> for a thorough examination, we will not need all this information for this class, but it is helpful to know!
* Work through code on pages 4-10 of TMR

Key commands/syntax to know:

* group\_by()
* mutate()
* #
* left\_join()
* inner\_join()
* anti\_join()

HW:

* Make flashcards of the key commands/syntax above, so you have a working definition of them
* Rewrite and rerun the code from pages 4-10 in TMR, but this time, make thorough comments on the code as you write it. Don’t just copy my comments, make your own.

Week 5: Plots (Examples borrowed from Roger Peng’s Coursera [course](https://www.coursera.org/learn/exploratory-data-analysis/lecture/qBzHS/ggplot2-part-1))

-What makes up a plot?

-A plot is a way to visualize your data. There are many different types of plots. They can be interactive, 2-dimensonal, 3-dimensional, overlapping, etc. Plots help us easily see patterns in the data and allow for fast identification of patterns. Instead of searching through numbers for patterns, seeing a graph is infinitely better. As they say, “a picture is worth a thousand words”. In this course, we will focus on 2-dimenstion plots. In a 2-dimensional plot, we will have 2 axes, a vertical axis and a horizontal axis. We typically refer to the horizontal axis as the “x-axis” and the vertical axis as the “y-axis”.

-What is a barplot?

-A barplot is a type of graph that lets us see count the number of occurrences of some specific variable or outcome in a dataset. For example, if I wanted to plot how many times males and females drink coffee, I might use a barplot. In this case, on the x-axis, I would have the variable gender, and on the y-axis, I would have the number of times coffee was drunk.

-What is a line-plot?

-A line-plot is a type of graph that lets us see how changes in the x-axis variable are related to changes in the y-axis variable. Perhaps the x-axis is what year in college a student is, and the y-axis is GPA. Then we can look at how GPA changes as a student progresses from being a frosh to a senior.

-What is the big deal with ggplot2?

-ggplot2 is an R package created by Hadley Wickham. It is a way to make very intuitive, elegant, and flexible plots with R. Here is a [gallery](http://www.r-graph-gallery.com/portfolio/ggplot2-package/) of plots made with ggplot2 for you to peruse.

-Elements of ggplot2

-If you want a quick and dirty graph, use the function qplot() which I will demonstrate in class. qplot() is not as flexible as other plotting methods in ggplot2, but it quickly allows you to look at your data. If you have an idea about a specific plot you want to make, use the function ggplot(), which has much more flexibility, but requires more input.

* data frame
* aesthetics: how the data is mapped to colors, shapes, sizes, etc.
* geoms: how the data is plotted – bar plot, line-plot, pie graph, etc.
* facets: conditional plotting

-High level view of ggplot2

-Roger Peng says that a good way to think about ggplot2 is though you are an artist with a blank pallet. An artist starts with a blank canvas and builds layers and aesthetics into the painting. We are trying to do the same exercise. We will start with a dataframe and tell **R** that we want to use it to make some sort of plot. We will tell **R** how to map variables to colors and what to put on the various axes. Then we can add on additional layers to the plot, like what kind of plot we want (bar, line, etc), what the titles will be, how many panels we want, etc.

Activities during class:

* Base R vs ggplot2 plot example
* qplot() examples
* ggplot() examples, starting from nothing and building up layers

Key commands/syntax to know:

* qplot()
* facets
* ggplot()
* aes
* fill
* shape
* +
* labs()
* xlab()
* ylab()

HW:

* Make flashcards of the key commands/syntax above, so you have a working definition of them
* Make 5 completely different plots with ggplot2. Use the same dataset we used in class or find a new one. They should have different colors, facets, shapes, etc. Look online for help! The gallery will give you great ideas as well. Try to make something completely new and cool.
* Read pages 13-22 of TMR, and write down the code verbatim from the text, we will run it next class

See the ggplot2 [website](http://ggplot2.tidyverse.org/index.html) for more help!

Week 6: Back to sentiment analysis

-What is a lexicon?

-A lexicon is a list of words with the word’s sentiment attached to it. Sentiment can be defined in several ways. A word could be “positive” or “negative”, could be scored on a scale from -5 to 5 where higher numbers mean more “positive”, or they could be sentiments like “trust”, “fear”, “sadness”, etc. See pages 13-15 of TMR.

-What are the pros and cons of using lexicons to analyze the sentiment of a text or corpus?

-Does a word always carry the same sentiment no matter how it is used? What about negation or sarcasm?

What are “stop words”?

-Stop words are words that we are not interested in mining from our text. Perhaps they are names, numbers, or common words like “the”, “why”, or “about”.

Activities during class

* Work through code from pages 13-22 in TMR

Key commands/syntax to know:

* filter()
* ()

HW:

* Make flashcards of the key commands/syntax above, so you have a working definition of them

Miscellania

-File Directory Organization

In general, keeping your files organized on your personal computer is a good idea. It helps to be able to easily navigate to your files, especially if you have many of them. In your forays into programming, it will be extremely important to know where all your scripts and data live. My advice is to make a new folder for this class either your main ‘users’ file directory. (I will go over this day 1).Perhaps you have a folder already titles ‘SMU’ and within this folder you have various classes. This is also a wonderful spot to add a new folder with the title of this class and the year, i.e. ‘Text\_Mining\_2017’. It is highly recommended to use an underscore (\_) instead of a space between words since you will be typing it into **R** and you do not want to miss a space. Within this ‘Text\_Mining\_2017’ folder, I recommend a folder titles ‘scripts’ for all of your R scripts to live and a folder titled ‘data’ for all your datasets to live. Be sure to title your

-Commenting your code

setwd()