Lab #6: Topic Modeling

Due in Sakai March 8, 2019 (by 10:00a)

Introduction

For this lab, you will generate and interpret your own topic model analysis using latent Dirichlet allocation (LDA).

Getting Started

First, set your working directory. Open up a fresh RStudio session (if you open RStudio and a previous work session is loaded, be sure you save any R and/or RData files before you close them out). Then set your working directory:

```
setwd("working_directory_here")
```

Then open up a fresh R script. Save it using your first initial, full last name, and then "_lab6." So my script would be titled **MTaylor_lab6.R**.

You should always strive to keep your scripts tidy. At the top of your R script, type this (substituting in your first initial and last name):

```
##############################
##  MTaylor_lab6.R
##  Note: Code for lab assignment #6
##  Author: Marshall A. Taylor
##############################

###BEGIN###
```

You are ready to begin your code. Be sure to include all the code necessary for me to check your work. Document your code thoroughly (using "#").

Remember that saving your R work is a two-step process. You save your R script using Cmd+Enter (Mac) or Cntrl+Enter (Windows). You save your R data objects like this:

```
save.image("MTaylor_lab6.RData")
```

Lastly, prep a Word, Pages, or LaTeX document that has the same title structure: e.g., **MTaylor_lab6.docx**. This is where you put your write-ups and visualizations.

You should turn in three documents to Sakai: your **R script** that shows the code you used, **RData file** that provides the data your scraped, and **Word document** (or whatever text processor you choose to use) with your write-ups.

Assignment

Note that some of these functions may take a while to run. Plan your time accordingly.

Let's generate a representation of the underlying latent topics in a corpus of academic journal articles. The data were collected by Deny A. Kwary (2018), and consist of 895 articles from a top academic publisher: Elsevier. The articles span across four general disciplines: health sciences, life sciences, physical sciences,

and social sciences. The publication dates range from 2011 to 2015. You can learn more about this corpus here and here. The data are in a RDS file called "article_corpus.RDS," which you can read in using the `readRDS()` function.

The texts are full academic articles. This introduces some interesting preprocessing issues, since you have page breaks, tables, figures, etc. As such, I have included the preprocessing script I used to clean the data in the Hints section at the end of this document. You do not *have* to follow those steps, but they worked for me.

Also, note that the RDS file is already a `tm` corpus object. So no need to convert the texts to a corpus before you begin preprocessing.

Once you have cleaned up the texts, do the following:

1. Generate the topics. Note that this will probably not be the very first topic model analysis you'll run. You'll probably want to try out different k (i.e., different numbers of topics) and compare across them. Walk me through the steps you took to decide on the topic model analysis you settled on, and why you settled on it. Show whatever graphs/plots/top terms you think are necessary to communicate your argument.
2. Create an interactive visualization that allows me to compare terms across various term relevance scores. When you submit this, you will want to upload the folder that the `LDAvis` package outputs. So, if you set `out.dir = "vis"` in the `serVis()` function, you will have a folder called "vis." You will want to submit that entire folder (not just the files contained therein). You may have to compress the folder in order to submit it via Sakai. (Note: you will probably want to use this visualization to complete step #1.)

Hints

The R scripts for the dictionary and sentiment analysis slides (pres71.R) might be very helpful. Like, *really really helpful*.

This is the preprocessing workflow I used to clean the articles. Feel free to customize it any way you want.

```
#Clean it up
removeAllPunct <- function(x) gsub("[[:punct:]]", " ", x)
removeSpecialChars <- function(x) gsub("[^a-zA-Z0-9 ]", " ", x)

article.tm <- tm_map(article.tm, content_transformer(removeAllPunct))
article.tm <- tm_map(article.tm, content_transformer(removeSpecialChars))
article.tm <- tm_map(article.tm, content_transformer(tolower))
article.tm <- tm_map(article.tm, removeNumbers)
article.tm <- tm_map(article.tm, removeWords, stopwords("english"))
article.tm <- tm_map(article.tm, stripWhitespace)

article.tm <- convert.tm.to.character(article.tm)
article.tm <- lemmatize_strings(article.tm,
                                dictionary = lexicon::hash_lemmas)

article.tm <- VCorpus(VectorSource(article.tm))
article.tm <- tm_map(article.tm, content_transformer(removeAllPunct))
article.tm <- tm_map(article.tm, content_transformer(removeSpecialChars))
article.tm <- tm_map(article.tm, stripWhitespace)

inspect(article.tm[[1]]) #Check to make sure everything looks good
```

The the `removeSpecial` bit of code is adapted from here, and the `removeAllPunct` code is adapted from here.

References

Kwary, Deny A. 2018. "A Corpus and Concordancer of Academic Journal Articles." *Data in Brief* 16:94-100.