Blog for 10-23 tidytext (not tiny text)

We’ll need to install & library the following packages:

tidy text

tm (don’t need to library) -tm stands for text mining

wordcloud

dplyr

Now we need some text. Google Born to Run album lyrics. We found some that formatted into a text file easily at <http://www.myths.com/pub/lyrics/Springsteen_3_btr.html>.

In R studio, get a New text file:

Copy the lytics from the website & put them in the R.text file. Scroll through & delete any of the song titles.

Save the file as “lyrics.txt”

Lets look at the lines:

scan('lyrics.txt',what=character(),sep='\n')

First step is a to set up the lyrics into a variable called lines:

lines<-scan('lyrics.txt',what=character(),sep='\n')

When we run this, we get the following in the Rstudio command prompt telling us how many lines were read:

> lines<-scan('lyrics.txt',what=character(),sep='\n')

Read 292 items

We are going to use the 292 number to define our data frame with the line numbers from our text:

data\_frame(line=1:78,text=lines)

lets make a new dataframe variable out of that called lines\_df:

lines\_df<-data\_frame(line=1:296,text=lines)

Now we’ll use tidytext:

We will use unnest: Unnest makes individual words from our data frame. The unnest needs (input, output). For our output (which we’ll call words) this will make a column called ‘word’. The input (which we’ll call text) will be the individual words. The following shows us the words & the count of the number of times used.

lines\_df%>%

unnest\_tokens(word,text)

We’ll save this into a variable words\_df:

words\_df<-lines\_df%>%

unnest\_tokens(word,text)

Now we aren’t interested in words like the, a, and, etc. These are called stop words in tidytext.

If we only wanted to see the stop words:

words\_df%>%

filter(word %in% stop\_words$word)

the %in% means belonging to

tidy text knows that the ‘$word’ in ‘stop\_words$word” is the word column.

BUT we DON’T want the stop words so we’ll use the ! (exclamation point) for NOT:

So the following shows us everything but the stop words

words\_df%>%

filter(!(word %in% stop\_words$word))

we’ll save this as words\_df again:

words\_df<-words\_df%>%

filter(!(word %in% stop\_words$word))

Now let’s look at our data. In the command line type in: head(words\_df)

We need to count the frequency of the words next. We use the count=n():

words\_df%>%

group\_by(word)%>%

summarize(count=n())

Now save to a variable:

word\_freq<-words\_df%>%

group\_by(word)%>%

summarize(count=n())

Columns of words & the frequency- this is what we need to make a word cloud:

So for our word cloud:

wordcloud(word\_freq$word,word\_freq$count))

B&W word cloud

Now let’s add some color. The number 7 tells the color pallet ‘Dark2’ how many colors to use:

Some of the color pallets don’t have much variety, but ‘Dark2’ seems pretty good:

wordcloud(word\_freq$word,word\_freq$count,colors=brewer.pal(7,'Dark2'))

end.