Suits(USA network): Text Mining with R

R. Mukamuri

July 8, 2016

Load the required libraries

```
library(rvest)
```

```
## Loading required package: xml2
```

Select tv show

```
tvshow <- "suits"
```

Create download directory and change to it

```
directory = paste("~/Data Analysis/files/", tvshow,
sep="")
dir.create(directory, recursive = TRUE, showWarnings = FALSE)
setwd(directory)
```

Setting base url and complete url

```
baseurl <- "http://www.springfieldspringfield.co.uk/"
url <- paste(baseurl, "episode_scripts.php?tv-show=",
             tvshow, sep="")
```

Read the HTML page

```
scrape_url <- read_html(url)
# node selector
s_selector <- ".season-episode-title"
# scrape href nodes in .season-episode-title
all_urls_season <- html_nodes(scrape_url, s_selector) %>%
  html_attr("href")
# Show some structure of the all_url_seasons.
str(all_urls_season)
```

```
##  chr [1:76] "view_episode_scripts.php?tv-show=suits&episode=s01e01" ...
```

```
# first 6 episodes
head(all_urls_season)
```

```
## [1] "view_episode_scripts.php?tv-show=suits&episode=s01e01"
## [2] "view_episode_scripts.php?tv-show=suits&episode=s01e02"
## [3] "view_episode_scripts.php?tv-show=suits&episode=s01e03"
## [4] "view_episode_scripts.php?tv-show=suits&episode=s01e04"
## [5] "view_episode_scripts.php?tv-show=suits&episode=s01e05"
## [6] "view_episode_scripts.php?tv-show=suits&episode=s01e06"
```

```
# last 6
tail(all_urls_season)
```

```
## [1] "view_episode_scripts.php?tv-show=suits&episode=s05e11"
## [2] "view_episode_scripts.php?tv-show=suits&episode=s05e12"
## [3] "view_episode_scripts.php?tv-show=suits&episode=s05e13"
## [4] "view_episode_scripts.php?tv-show=suits&episode=s05e14"
## [5] "view_episode_scripts.php?tv-show=suits&episode=s05e15"
## [6] "view_episode_scripts.php?tv-show=suits&episode=s05e16"
```

We have 76 url episodes, Now we have all the variables and season urls,lets harvest the scripts and save them to seperate text files for doing our text mining

```
# Loop through all the season urls
for (i in all_urls_season){
  uri <- read_html(paste(baseurl, i, sep = "/"))
  # same thing here first check which node we need to select, so first do an inspection of the site
script_selector <- ".scrolling-script-container"
# scrape all script text to a variable
text <- html_nodes(uri, script_selector) %>%
  html_text()
# Get last five characters of all_url_season as season for saving this to seperate text files
substrRight <- function(x, n) {
  substr(x, nchar(x)-n+1, nchar(x))
}
seasons <- substrRight(i, 5)
# Write each script to a seperate text file
write.csv(text, file = paste(directory, "/", tvshow,
                              "_", seasons, ".txt", sep = ""), row.names = FALSE)
}
```

Start the text mining

```
#load library
library(tm)
```

```
## Loading required package: NLP
```

```
# set filepath to scripts
cname <- file.path(directory)
# see if the filepath contains our scripts
(docname <- dir(cname))
```

```
##  [1] "Suits_01e01.txt" "Suits_01e02.txt" "Suits_01e03.txt"
##  [4] "Suits_01e04.txt" "Suits_01e05.txt" "Suits_01e06.txt"
##  [7] "Suits_01e07.txt" "Suits_01e08.txt" "Suits_01e09.txt"
## [10] "Suits_01e10.txt" "Suits_01e11.txt" "Suits_01e12.txt"
## [13] "Suits_02e01.txt" "Suits_02e02.txt" "Suits_02e03.txt"
## [16] "Suits_02e04.txt" "Suits_02e05.txt" "Suits_02e06.txt"
## [19] "Suits_02e07.txt" "Suits_02e08.txt" "Suits_02e09.txt"
## [22] "Suits_02e10.txt" "Suits_02e11.txt" "Suits_02e12.txt"
## [25] "Suits_02e13.txt" "Suits_02e14.txt" "Suits_02e15.txt"
## [28] "Suits_02e16.txt" "Suits_03e01.txt" "Suits_03e02.txt"
## [31] "Suits_03e03.txt" "Suits_03e04.txt" "Suits_03e05.txt"
## [34] "Suits_03e06.txt" "Suits_03e07.txt" "Suits_03e09.txt"
## [37] "Suits_03e10.txt" "Suits_03e11.txt" "Suits_03e12.txt"
## [40] "Suits_03e13.txt" "Suits_03e14.txt" "Suits_03e15.txt"
## [43] "Suits_03e16.txt" "Suits_03e80.txt" "Suits_04e01.txt"
## [46] "Suits_04e02.txt" "Suits_04e03.txt" "Suits_04e04.txt"
## [49] "Suits_04e05.txt" "Suits_04e06.txt" "Suits_04e07.txt"
## [52] "Suits_04e08.txt" "Suits_04e09.txt" "Suits_04e10.txt"
## [55] "Suits_04e11.txt" "Suits_04e12.txt" "Suits_04e13.txt"
## [58] "Suits_04e14.txt" "Suits_04e15.txt" "Suits_04e16.txt"
## [61] "Suits_05e01.txt" "Suits_05e02.txt" "Suits_05e03.txt"
## [64] "Suits_05e04.txt" "Suits_05e05.txt" "Suits_05e06.txt"
## [67] "Suits_05e07.txt" "Suits_05e08.txt" "Suits_05e09.txt"
## [70] "Suits_05e10.txt" "Suits_05e11.txt" "Suits_05e12.txt"
## [73] "Suits_05e13.txt" "Suits_05e14.txt" "Suits_05e15.txt"
## [76] "Suits_05e16.txt"
```

```
# Crete a Corpus of the text files so we can do some analysis
docs <- Corpus(DirSource(cname), readerControl = list(id=docname))
# Show summary of the Corpus, we have 40 document in our Corpus
summary(docs)
```

```
##                 Length Class            Mode
## Suits_01e01.txt 2      PlainTextDocument list
## Suits_01e02.txt 2      PlainTextDocument list
## Suits_01e03.txt 2      PlainTextDocument list
## Suits_01e04.txt 2      PlainTextDocument list
## Suits_01e05.txt 2      PlainTextDocument list
## Suits_01e06.txt 2      PlainTextDocument list
## Suits_01e07.txt 2      PlainTextDocument list
## Suits_01e08.txt 2      PlainTextDocument list
## Suits_01e09.txt 2      PlainTextDocument list
## Suits_01e10.txt 2      PlainTextDocument list
## Suits_01e11.txt 2      PlainTextDocument list
## Suits_01e12.txt 2      PlainTextDocument list
## Suits_02e01.txt 2      PlainTextDocument list
## Suits_02e02.txt 2      PlainTextDocument list
## Suits_02e03.txt 2      PlainTextDocument list
## Suits_02e04.txt 2      PlainTextDocument list
## Suits_02e05.txt 2      PlainTextDocument list
## Suits_02e06.txt 2      PlainTextDocument list
## Suits_02e07.txt 2      PlainTextDocument list
## Suits_02e08.txt 2      PlainTextDocument list
## Suits_02e09.txt 2      PlainTextDocument list
## Suits_02e10.txt 2      PlainTextDocument list
## Suits_02e11.txt 2      PlainTextDocument list
## Suits_02e12.txt 2      PlainTextDocument list
## Suits_02e13.txt 2      PlainTextDocument list
## Suits_02e14.txt 2      PlainTextDocument list
## Suits_02e15.txt 2      PlainTextDocument list
## Suits_02e16.txt 2      PlainTextDocument list
## Suits_03e01.txt 2      PlainTextDocument list
## Suits_03e02.txt 2      PlainTextDocument list
## Suits_03e03.txt 2      PlainTextDocument list
## Suits_03e04.txt 2      PlainTextDocument list
## Suits_03e05.txt 2      PlainTextDocument list
## Suits_03e06.txt 2      PlainTextDocument list
## Suits_03e07.txt 2      PlainTextDocument list
## Suits_03e09.txt 2      PlainTextDocument list
## Suits_03e10.txt 2      PlainTextDocument list
## Suits_03e11.txt 2      PlainTextDocument list
## Suits_03e12.txt 2      PlainTextDocument list
## Suits_03e13.txt 2      PlainTextDocument list
## Suits_03e14.txt 2      PlainTextDocument list
## Suits_03e15.txt 2      PlainTextDocument list
## Suits_03e16.txt 2      PlainTextDocument list
## Suits_03e80.txt 2      PlainTextDocument list
## Suits_04e01.txt 2      PlainTextDocument list
## Suits_04e02.txt 2      PlainTextDocument list
## Suits_04e03.txt 2      PlainTextDocument list
## Suits_04e04.txt 2      PlainTextDocument list
## Suits_04e05.txt 2      PlainTextDocument list
## Suits_04e06.txt 2      PlainTextDocument list
## Suits_04e07.txt 2      PlainTextDocument list
## Suits_04e08.txt 2      PlainTextDocument list
## Suits_04e09.txt 2      PlainTextDocument list
## Suits_04e10.txt 2      PlainTextDocument list
## Suits_04e11.txt 2      PlainTextDocument list
## Suits_04e12.txt 2      PlainTextDocument list
## Suits_04e13.txt 2      PlainTextDocument list
## Suits_04e14.txt 2      PlainTextDocument list
## Suits_04e15.txt 2      PlainTextDocument list
## Suits_04e16.txt 2      PlainTextDocument list
## Suits_05e01.txt 2      PlainTextDocument list
## Suits_05e02.txt 2      PlainTextDocument list
## Suits_05e03.txt 2      PlainTextDocument list
## Suits_05e04.txt 2      PlainTextDocument list
## Suits_05e05.txt 2      PlainTextDocument list
## Suits_05e06.txt 2      PlainTextDocument list
## Suits_05e07.txt 2      PlainTextDocument list
## Suits_05e08.txt 2      PlainTextDocument list
## Suits_05e09.txt 2      PlainTextDocument list
## Suits_05e10.txt 2      PlainTextDocument list
## Suits_05e11.txt 2      PlainTextDocument list
## Suits_05e12.txt 2      PlainTextDocument list
## Suits_05e13.txt 2      PlainTextDocument list
## Suits_05e14.txt 2      PlainTextDocument list
## Suits_05e15.txt 2      PlainTextDocument list
## Suits_05e16.txt 2      PlainTextDocument list
```

```r
# Inspect the first document, it has 12958 characters
inspect(docs[1])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1
```

```
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 48944
```

There is a lot of information in the script we do not need and is not useful for text mining. We need to clean it up. We remove all numbers, convert text to lowercase, remove punctuation and stopwords, in this case english.

```
docs <- tm_map(docs, tolower)
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removeWords, stopwords("english"))
```

Now we will perform stemming, a stem is a form to which affixes can be attached. An example of this is wait, waits, waited, waiting, all of them are common to wait.

```
library(SnowballC)
docs <- tm_map(docs, stemDocument)
```

We have removed a lot of characters which resulted in a lot of whitespaces, we remove this also.

```
docs <- tm_map(docs, stripWhitespace)
#Let's have a look to our first document.

inspect(docs[1])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1
##
## [[1]]
## [1] x
## [2]
## [3]
## [4]
## [5]
## [6]  gerald tates wants know whats happening deal go get harvey check raise im can pay later ive got
```

I have hash it because wordpress has problems with editing the post.We are ready with preprosessing the data and turn the document back as plain text documents.

```
docs <- tm_map(docs, PlainTextDocument)
```

Create a Term Document Matrix of our documents. Which reflects the number of times each term in the corpus is found in each of the documents. And add some readable columnnmes

```
# Create a tdm
tdm <- TermDocumentMatrix(docs)
# Add readable columnnames, in our case the document filename
```

```r
docname <- gsub("suits_", "",docname)
docname <- gsub(".txt", "",docname)
docname <- paste("s",docname, sep="")
colnames(tdm) <- docname
# Show and inspect the tdm
tdm
```

```
## <<TermDocumentMatrix (terms: 12798, documents: 76)>>
## Non-/sparse entries: 77160/895488
## Sparsity           : 92%
## Maximal term length: 24
## Weighting          : term frequency (tf)
```

```r
inspect(tdm[1:10,1:6])
```

```
## <<TermDocumentMatrix (terms: 10, documents: 6)>>
## Non-/sparse entries: 2/58
## Sparsity           : 97%
## Maximal term length: 11
## Weighting          : term frequency (tf)
##
##              Docs
## Terms          sSuits_01e01 sSuits_01e02 sSuits_01e03 sSuits_01e04
##   aah                     0            0            0            0
##   aaron                   0            0            1            0
##   aba                     0            0            0            0
##   abandon                 0            0            0            0
##   abandoned               0            0            0            0
##   abandoning              0            0            0            0
##   abandonment             0            0            0            0
##   abatement               0            0            0            2
##   abbey                   0            0            0            0
##   abcs                    0            0            0            0
##              Docs
## Terms          sSuits_01e05 sSuits_01e06
##   aah                     0            0
##   aaron                   0            0
##   aba                     0            0
##   abandon                 0            0
##   abandoned               0            0
##   abandoning              0            0
##   abandonment             0            0
##   abatement               0            0
##   abbey                   0            0
##   abcs                    0            0
```

Do the same for a Document Term Matrix (this is a transpose of a tdm)

```r
dtm <- DocumentTermMatrix(docs)
rownames(dtm) <- docname
dtm
```

```
## <<DocumentTermMatrix (documents: 76, terms: 12798)>>
## Non-/sparse entries: 77160/895488
## Sparsity           : 92%
## Maximal term length: 24
## Weighting          : term frequency (tf)
```

```r
inspect(dtm[1:10,1:6])
```

```
## <<DocumentTermMatrix (documents: 10, terms: 6)>>
## Non-/sparse entries: 2/58
## Sparsity           : 97%
## Maximal term length: 10
## Weighting          : term frequency (tf)
##
##                Terms
## Docs             aah aaron aba abandon abandoned abandoning
##   sSuits_01e01   0     0   0       0         0          0
##   sSuits_01e02   0     0   0       0         0          0
##   sSuits_01e03   0     1   0       0         0          0
##   sSuits_01e04   0     0   0       0         0          0
##   sSuits_01e05   0     0   0       0         0          0
##   sSuits_01e06   0     0   0       0         0          0
##   sSuits_01e07   0     0   0       0         0          0
##   sSuits_01e08   0     0   0       0         0          0
##   sSuits_01e09   0     0   0       0         0          0
##   sSuits_01e10   0     1   0       0         0          0
```

Term frequency Let have a look of the most frequently terms first and show the top 20.

```r
freq <- sort(colSums(as.matrix(dtm)), decreasing=TRUE)
head(freq,20)
```

```
##   know   dont  youre   just  going    get   want harvey   well  right
##   3921   3811   3254   2752   2484   2338   2335   2166   2022   1797
##  didnt  thats    can   like    now  think    one  louis   tell  gonna
##   1794   1744   1664   1616   1612   1593   1550   1545   1434   1340
```

Plotting the terms frequencies

Add is to a data frame so we can plot it and show the top 20.

```r
tf <- data.frame(term=names(freq), freq=freq)
head(tf,20)
```

```
##          term freq
## know     know 3921
## dont     dont 3811
## youre   youre 3254
## just     just 2752
## going   going 2484
## get       get 2338
## want     want 2335
## harvey harvey 2166
## well     well 2022
## right   right 1797
## didnt   didnt 1794
## thats   thats 1744
## can       can 1664
## like     like 1616
## now       now 1612
## think   think 1593
## one       one 1550
## louis   louis 1545
## tell     tell 1434
## gonna   gonna 1340
```

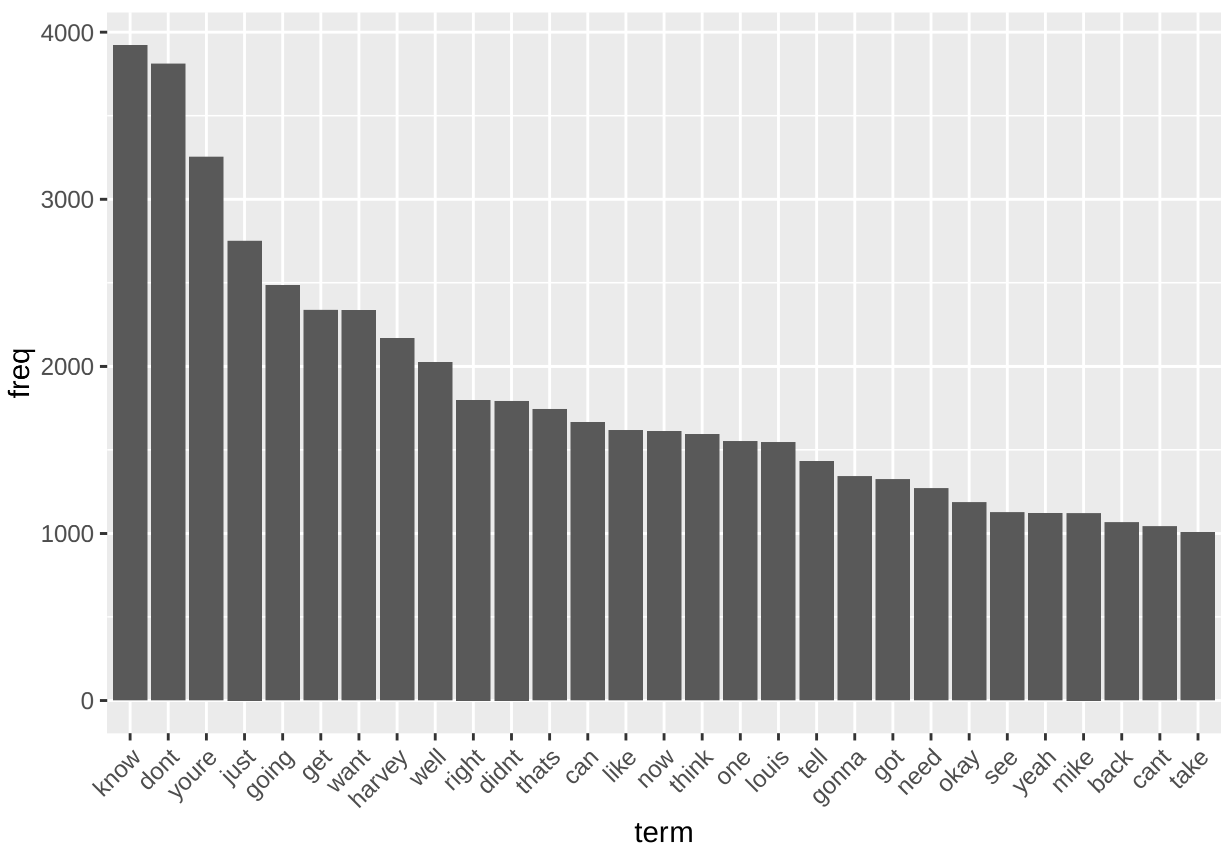
```
#Let's plot it.

# descending sort of teh tf by freq
tf$term <- factor(tf$term, levels = tf$term[order(-tf$freq)])
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate
```

```
p <- ggplot(subset(tf, freq>1000), aes(term, freq))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
p
```



The most frequent term in "know"" followe by "don't", Harvey, Lewis and Mike are the most mentioned.As we can see in our first look at the tdm, we have a lot op sparse terms in our documents (97%). That is a lot, lets remove these.

```
tdm.common = removeSparseTerms(tdm, sparse = 0.014)
tdm
```

```
## <<TermDocumentMatrix (terms: 12798, documents: 76)>>
## Non-/sparse entries: 77160/895488
## Sparsity           : 92%
## Maximal term length: 24
## Weighting          : term frequency (tf)
```

```
tdm.common
```

```
## <<TermDocumentMatrix (terms: 84, documents: 76)>>
## Non-/sparse entries: 6300/84
## Sparsity           : 1%
## Maximal term length: 9
## Weighting          : term frequency (tf)
```

That is a 90% less sparsity. See how many terms we had and now have

```
dim(tdm)
```

```
## [1] 12798    76
```

```r
dim(tdm.common)
```

```
## [1] 84 76
```

Hmm from 12798 terms to only 186 terms, we inspect the first 10 terms of the first 6 documents.

```r
inspect(tdm.common[1:10,1:6])
```

```
## <<TermDocumentMatrix (terms: 10, documents: 6)>>
## Non-/sparse entries: 60/0
## Sparsity           : 0%
## Maximal term length: 7
## Weighting          : term frequency (tf)
##
##          Docs
## Terms     sSuits_01e01 sSuits_01e02 sSuits_01e03 sSuits_01e04 sSuits_01e05
##   another            8            3            4            2            3
##   back              21           15           11            7           16
##   better             4            4            9            6            6
##   busy               1            4            3            2            2
##   call               2            8            5            3           11
##   can               42           25           20           20           19
##   cant              11            8           11           12           11
##   come              15           15            8            7           10
##   day                9            3            5            4            1
##   didnt             23           22           18           26           15
##          Docs
## Terms     sSuits_01e06
##   another            3
##   back               8
##   better             4
##   busy               6
##   call               2
##   can               15
##   cant              15
##   come              16
##   day                5
##   didnt             19
```

Let visualize these most common terms in a heatmap with ggplot. As ggplot works with a matrix we need to convert the tdm.comon to a matrix because the tdm is a spare matrix.

```r
tdm.dense <- as.matrix(tdm.common)
dim(tdm.dense)
```

```
## [1] 84 76
```

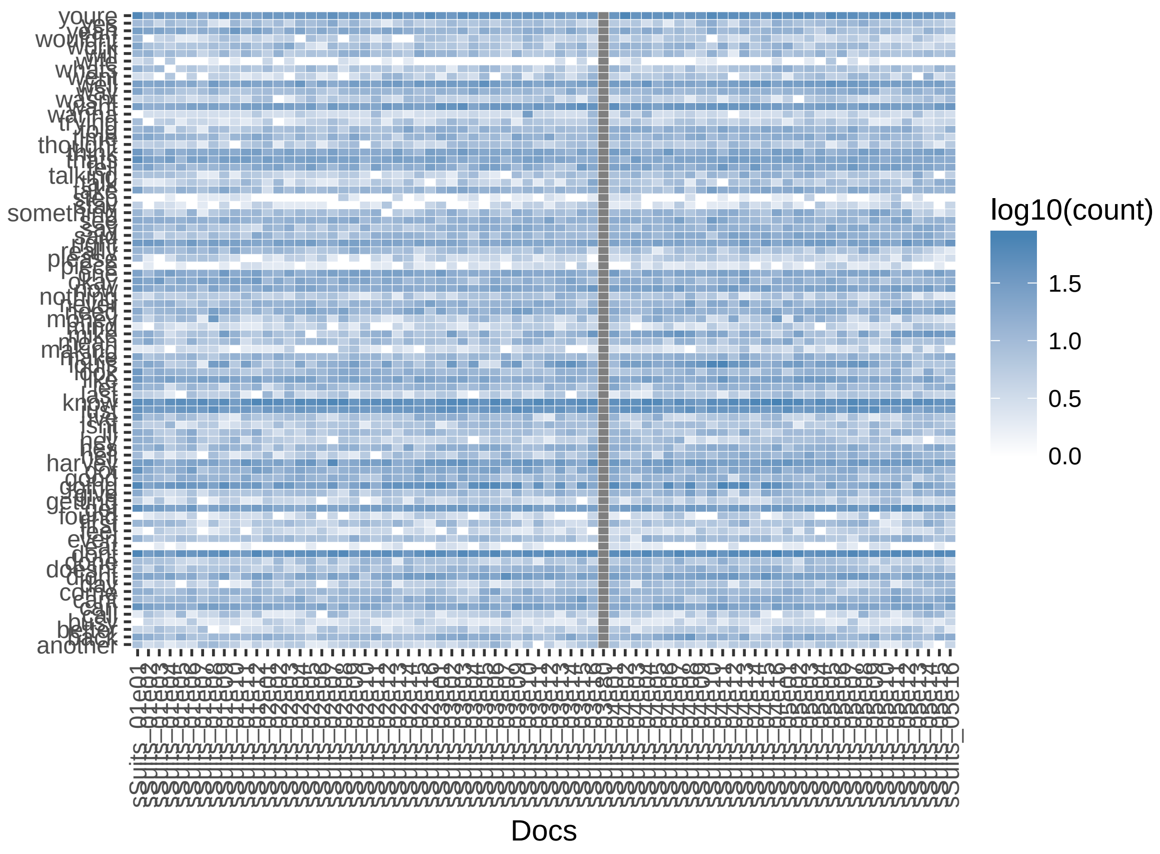
We need the data as a normal matrix in order to produce the visualisation.

```r
library(reshape2)
tdm.dense.m <- melt(tdm.dense, value.name = "count")
head(tdm.dense.m)
```

```
##     Terms         Docs count
## 1 another sSuits_01e01     8
## 2    back sSuits_01e01    21
## 3  better sSuits_01e01     4
## 4    busy sSuits_01e01     1
## 5    call sSuits_01e01     2
## 6     can sSuits_01e01    42
```

Make the heatmap visualization.

```r
library(ggplot2)
ggplot(tdm.dense.m, aes(x = Docs, y = Terms, fill = log10(count))) +
     geom_tile(colour = "white") +
     scale_fill_gradient(high="steelblue" , low="white")+
     ylab("") +
     theme(panel.background = element_blank()) +
     theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))
```

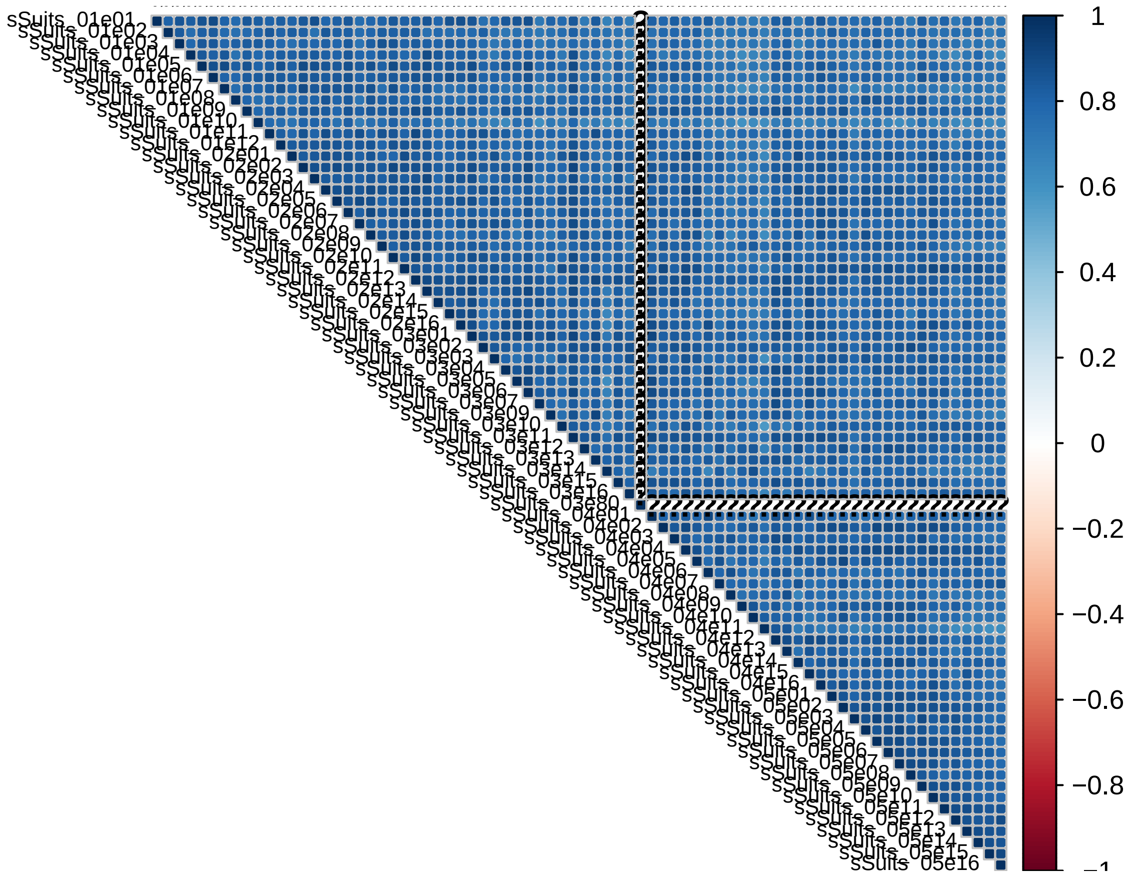


was expecting "bullshit" to be among the most common words haha. Now we plot a correlogram of the episodes.Note: Correlogram is a graph of correlation matrix. It is very useful to highlight the most correlated variables in a data table. In this plot, correlation coefficients is colored according to the value. Correlation matrix can be also reordered according to the degree of association between variables.

```r
corr <- cor(tdm.dense)
```

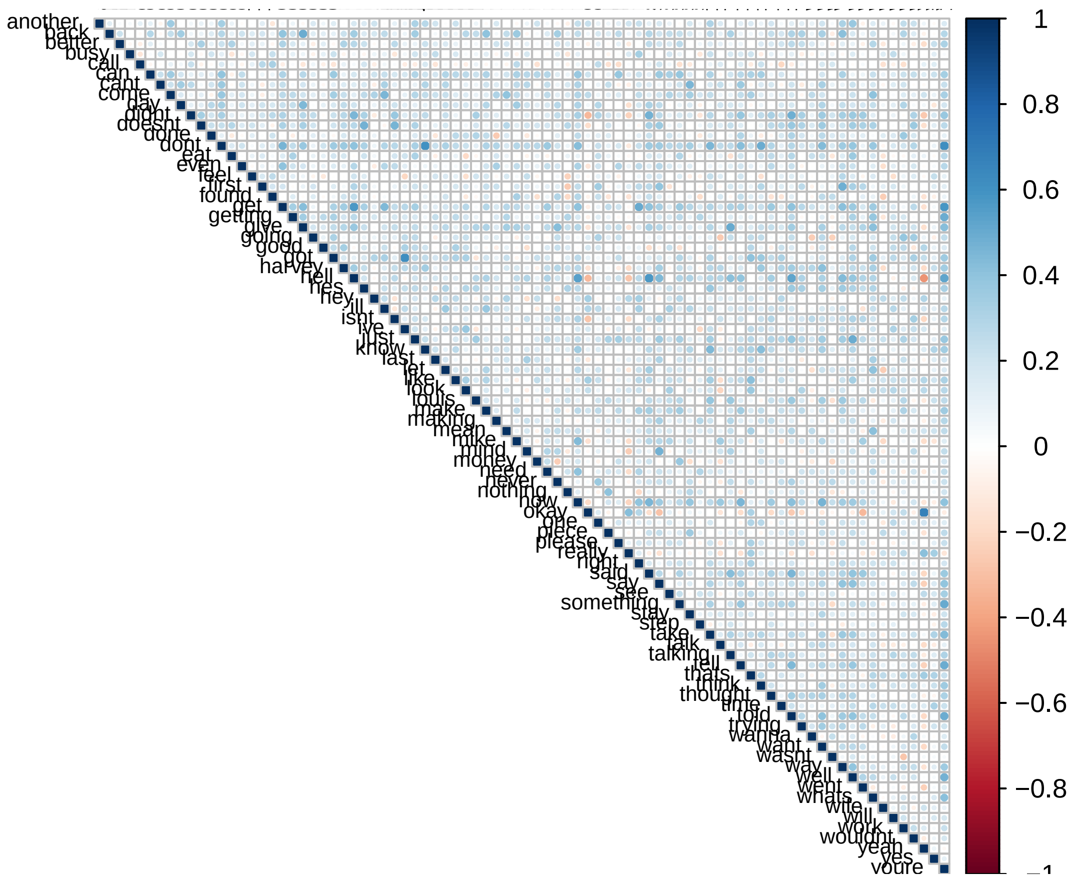
```
## Warning in cor(tdm.dense): the standard deviation is zero
```

```r
library(corrplot)
corrplot(corr, method = "circle", type = "upper", tl.col="black", tl.cex=0.7)
```



Transpose the tdm.dense so we can plot a correlogram of the terms.

```r
tdm.dense.t <- t(tdm.dense)
corr.t <- cor(tdm.dense.t)
corrplot(corr.t,method = "circle", type = "upper", tl.col="black", tl.cex=0.7)
```



To be continued. . .