Presentation Notes

Jill MacKay

7 May 2018

```
library(tidyverse)
library(tidytext)
library(gutenbergr)
library(tm)
library(wordcloud)
```

A Fundamental Assertion

All ‘data science’ does is condense *lots of data* into a *smaller bit of data*.

If I asked you to interpret that data, you would *not* say:

```
print(lots.of.data)
```

Dealing with Lots of Data

To interpret *lots of data* you would instead likely do something like:

```
mean(lots.of.data)
sd(lots.of.data)
```

The rest is just quibbling over methodology (which is extremely fun to do!)

```
median(lots.of.data)
```

Textual Data

In the same way, if you have a large chunk of text, you do not ‘analyse’ it by saying . . .

```
lotta.text <- gutenberg_download(31847)
lotta.text <- lotta.text %>%
  select (-gutenberg_id)
print(lotta.text)
```

Textual Data

There are three main ways to analyse textual data in R

- As corpora (`tm`)
- As tidytext (`tidytext`)
- Thematically (`RQDA`)

(Lets use a simpler example than `lottatext`)

```
data <-  c("",
           "word1",
           "word1 word2",
           "word1 word2 word3",
           "word1 word2 word3 word4",
           "word1 word2 word3 word4 word5")
```

The tm Package

The `tm` package is about 'text mining' and treats text as a 'corpus' (or many 'corpora').

The basic command is:

```
tm.text <- Corpus(VectorSource(data))
```

Where `Corpus` reads in the data.

`VectorSource` tells `Corpus` to expect the source file to be in a vector format.

The tm Package

A corpus can be thought of as a document of documents.

In this case, using the command `VectorSource` has each 'row' of the example data as a new document.

If you have multiple big text files in a folder you can use `DirSource` as opposed to `VectorSource`

`DirSource` will read all the documents in the directory it's pointed to.

Think of each document as an 'observation'.

Document Term Matrices vs Term Document Matrices

Tutorials often use these interchangeably, but there are times you want one, and times you want the other.

The key difference is that `DocumentTermMatrix` has documents listed in the first **column**

```
dtm.data <- DocumentTermMatrix(tm.text)
head(as.matrix(dtm.data))
```

Document Term Matrices vs Term Document Matrices

Versus the `TermDocumentMatrix` has documents listed in the first **row**

```
tdm.data <- TermDocumentMatrix(tm.text)
head(as.matrix(tdm.data))
```

Using Document Term Matrices for Frequency

You can explore datasets quickly using these matrices.

```
# TermDocumentMatrix
tdm.m <- as.matrix(tdm.data)
tdm.v <- sort(rowSums(tdm.m), decreasing = TRUE)
tdm.d <- data.frame(word = names(tdm.v), freq = tdm.v)
head(tdm.d, 10)
```

Exploring Frequency with tm Commands

There are a few interesting commands in the `tm` package which are frequently found in online tutorials (there are caveats to how they work though)

```
#Find the top 3 terms
findFreqTerms(tdm.data, 3)

#Find correlations above 0 with Word 1 (Pearson correlations, 0 is lower limit)
findAssocs(tdm.data, "word1", 0)
```

Word Clouds

People love to visualise text data with word clouds. `tm` works very well with `wordcloud` to do this (`wordcloud` expects data in Corpus format)

```
wordle <- wordcloud(tm.text, scale = c(5,0.5), max.words = 10, random.order = FALSE, random.color = FALS
wordle
```

tm versus tidytext

The `tm` package is very good at word frequency, and very little else.

My preferred alternative is `tidytext`.

Corpora are documents as you or I would read them. `tidytext` converts text into tidy data.

```
data_tb <- tibble(text = data)

#This command unnests all the words so every word is on a new row sequentially
data_tb_un <- data_tb %>%
    unnest_tokens(word, text)
```

Using tidytext for Frequency Analysis

```
data_freq <- data_tb_un %>%
  count (word, sort =TRUE)
data_freq
```

Like all tidyverse stuff, this can be built up in oniony layers

```
data_freq <- data_tb_un %>%
  count(word, sort = TRUE) %>%
  top_n (3)
data_freq
```

Using tidytext for Frequency Analysis

This can also be visualised like so:

```
data_freq <- data_tb_un %>%
  count(word, sort = TRUE) %>%
  top_n (3) %>%
  ungroup() %>%
  mutate(text_order = nrow(.):1)


ggplot (data = data_freq, aes(reorder(word, text_order), n)) +
  geom_bar (stat = "identity") +
  labs(x = "Word", y= "Frequency in 'data'") +
  coord_flip()+
  theme_classic()
```

Word Associations via n-grams

This time, instead of creating a tibble where each word is on a different row, we are going to create a tibble where each row is a sequential pair of words (bigrams).

This creates repetition, in the first two rows of the new tibble `data_eng` there is an overlap, row1-column2's 'word1' is the same as row2-column1's 'word1' (this will become more clear in a realistic dataset)

```
data_eng <- data_tb %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

data_eng
```

Word Associations via n-grams

We can count the bigrams in the same was as before

```
bigramcount <- data_eng %>%
  count (bigram, sort = TRUE)
bigramcount
```

Word Associations via n-grams

We can count the bigrams in the same was as before

```
ggplot (data = bigramcount, aes(x = (reorder(bigram,n)), y = n)) +
  geom_bar (stat = "identity") +
  labs(x = "Bigram", y= "Frequency in 'data'") +
  coord_flip()+
  theme_light()
```

RQDA

`RQDA` is a package which allows R to 'code' free-text data. 'Coding' in this context, means 'categorising'.

R cannot thematically analyse text **for** you. It can only organise your categories.

RQDA is available here - http://rqda.r-forge.r-project.org/. There are a few intermediary steps to installing RQDA, as it requires GTK+ to work.

RQDA

We're going to use a document which is freely available from the Gutenberg Project, (Dog Stories from the Spectator)[http://www.gutenberg.org/ebooks/31847].

```
#To open RQDA
RQDA()
```

A new window will open . . .

RQDA