Reading Set 5

Ziji Zhou

Due by 10pm ET on Monday

Reading Set Information

A more thorough reading and light practice of the textbook reading prior to class allows us to jump into things more quickly in class and dive deeper into topics. As you actively read the textbook, you will work through the Reading Sets to help you engage with the new concepts and skills, often by replicating on your own the examples covered in the book.

These should be completed on your own without help from your peers. While most of our work in this class will be collaborative, it is important each individual completes the active readings. The problems should be straightforward based on the textbook readings, but if you have any questions, feel free to ask me!

GitHub Workflow

1. Before editing this file, verify you are working on the copy saved in *your* repo for the course (check the filepath and the project name in the top right corner).
2. Before editing this file, make an initial commit of the file to your repo to add your copy of the problem set.
3. Change your name at the top of the file and get started!
4. You should *save, knit, and commit* the .Rmd file each time you've finished a question, if not more often.
5. You should occasionally *push* the updated version of the .Rmd file back onto GitHub. When you are ready to push, you can click on the Git pane and then click **Push**. You can also do this after each commit in RStudio by clicking **Push** in the top right of the *Commit* pop-up window.
6. When you think you are done with the assignment, save the pdf as "*Name_thisfilename_date*.pdf" (it's okay to leave out the date if you don't need it) before committing and pushing (this is generally good practice but also helps me in those times where I need to download all student homework files).

Gradescope Upload

For each question (e.g., 3.1), allocate all pages associated with the specific question. If your work for a question runs onto a page that you did not select, you may not get credit for the work. If you do not allocate *any* pages when you upload your pdf, you may get a zero for the assignment.

You can resubmit your work as many times as you want before the deadline, so you should not wait until the last minute to submit some version of your work. Unexpected delays/crises that occur on the day the assignment is due do not warrant extensions (please submit whatever you have done to receive partial credit).

Problem 1 **Ethics in practice** Last week we emphasized the importance of verifying with the website that scraping is allowed by checking the website's "robots.txt" file using the `paths_allowed()` function from the **robotstxt** package.

1.1 The first url in section 19.1.1 is copied below. Are robots allowed to scrape that page?

```
macbeth_url <- "http://www.gutenberg.org/cache/epub/1129/pg1129.txt"
paths_allowed(macbeth_url)
```

```
## [1] FALSE
```

Since `paths_allowed()` returned `FALSE` we cannot automatically scrape data from this site.

1.2 Read this page on web scraping from the Gender Novels Project (some links are broken). Then read Project Gutenberg's policy about robot access to their pages. Reflect on the situation we are in. What do you think are the appropriate steps to proceed if you are interested in scraping texts from Project Gutenberg? Is it okay to ignore the "robots.txt" file in this situation?

It isn't ok to ignore the robots.txt file as it is unethical and could lead to my IP address being blocked from accessing as well as possibly overloading the site. Since Gutenberg offers mirrors we could scrape from one of the mirror site (assuming that its respective robots.txt allows for our script).

1.3 The **gutenburgr** package allows us to be more responsible in our datascraping by incorporating the site's recommendations for scraping (that is, the site allows us to download texts from a mirror site rather than scraping gutenberg.org directly). I have provided the code below to demonstrate how we can use this package to get a text-analysis-friendly version of the text. Run the code below just to explore how this package works.

```
# Find unique ID for Macbeth on Gutenberg.org
gutenberg_works(title == "Macbeth")

# Use the ID to download Macbeth using a site mirror
Macbeth <- gutenberg_download(1533)
```

1.4 While the **gutenbergr** package is a great package, we want to be able to follow along with the code in the text for this chapter! Thankfully, the **mdsr** package already contains the Macbeth dataset. Run the code below to make the dataset appear in your environment pain. The next code chunk in the text introduces `str_split()`. Follow along with that block of code in the chunk below. Verify that `Macbeth_raw` is a vector of length 1. What is the length of `macbeth` after using `str_split()`? Why are the lengths different?

```r
data(Macbeth_raw)
#length(Macbeth_raw)
macbeth <- Macbeth_raw %>%
  str_split("\r\n") %>%
  pluck(1)
#length(macbeth)
```

`Macbeth_raw` stored the entirety of Macbeth on a single character vector, leading to the table containing a 1x1 table containing all of the text. By splitting the string up by the line breaks each line is then broken into a seperate row, showing that Macbeth has a total of 3194 rows.

Problem 2 **Text as data** In Section 19.1.1, the `str_subset()`, `str_detect()`, and `str_which()` functions are introduced for detecting a pattern in a character vector (like finding a needle in a haystack). This section also introduces *regular expressions*.

2.1 Explain what the 6 returned records tell us about what each function does below:

`str_subset()` returns the vectors in which the str " MACBETH" was detected. `str_which()` returns the a vector of the row numbers in which the string is detected. `str_detect` returns a 1xn table listing a boolean of whether the line contains the string.

```
str_subset(macbeth, "  MACBETH") %>% head()
```

```
## [1] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
## [2] "  MACBETH. So foul and fair a day I have not seen."
## [3] "  MACBETH. Speak, if you can. What are you?"
## [4] "  MACBETH. Stay, you imperfect speakers, tell me more."
## [5] "  MACBETH. Into the air, and what seem'd corporal melted"
## [6] "  MACBETH. Your children shall be kings."
```

```
str_which(macbeth, "  MACBETH") %>% head()
```

```
## [1] 228 433 443 466 478 483
```

```
str_detect(macbeth, "  MACBETH") %>% head()
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

2.2 Why do the two lines below differ in their results?

`str_subset(macbeth, "MACBETH\\.")` The double brackets inside the second argument imply a literal period in string search. `str_subset(macbeth, "MACBETH.")` Here the period implies that any character could follow "MACBETH".

```
str_subset(macbeth, "MACBETH\\.") %>% head()
```

```
## [1] "  MACBETH. So foul and fair a day I have not seen."
## [2] "  MACBETH. Speak, if you can. What are you?"
## [3] "  MACBETH. Stay, you imperfect speakers, tell me more."
## [4] "  MACBETH. Into the air, and what seem'd corporal melted"
## [5] "  MACBETH. Your children shall be kings."
## [6] "  MACBETH. And Thane of Cawdor too. Went it not so?"
```

```
str_subset(macbeth, "MACBETH.") %>% head()
```

```
## [1] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
## [2] "  LADY MACBETH, his wife"
## [3] "  MACBETH. So foul and fair a day I have not seen."
## [4] "  MACBETH. Speak, if you can. What are you?"
## [5] "  MACBETH. Stay, you imperfect speakers, tell me more."
## [6] "  MACBETH. Into the air, and what seem'd corporal melted"
```

2.3 The three commands below look similar, but return different results. In words, explain what overall pattern is being searched for in each of the three cases (i.e., what do the patterns `MAC[B-Z]`, `MAC[B|Z]`, and `^MAC[B-Z]` indicate?)?

All three searches for "MAC" with additional factors to the search. The first one searches for "MAC" followed by any character between "B-Z" (inclusive). The second can be only followed by "B" or "Z". The last one anchors the search term "MAC[B-Z]" to only the beginning of the text.

```r
str_subset(macbeth, "MAC[B-Z]") %>% head()
```

```
[1] "MACHINE READABLE COPIES MAY BE DISTRIBUTED SO LONG AS SUCH COPIES"
[2] "MACHINE READABLE COPIES OF THIS ETEXT, SO LONG AS SUCH COPIES"
[3] "WITH PERMISSION.  ELECTRONIC AND MACHINE READABLE COPIES MAY BE"
[4] "THE TRAGEDY OF MACBETH"
[5] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
[6] "  LADY MACBETH, his wife"
```

```r
str_subset(macbeth,"MAC[B|Z]") %>% head()
```

```
[1] "THE TRAGEDY OF MACBETH"
[2] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
[3] "  LADY MACBETH, his wife"
[4] "  MACBETH. So foul and fair a day I have not seen."
[5] "  MACBETH. Speak, if you can. What are you?"
[6] "  MACBETH. Stay, you imperfect speakers, tell me more."
```

```r
str_subset(macbeth, "^MAC[B-Z]") %>% head()
```

```
[1] "MACHINE READABLE COPIES MAY BE DISTRIBUTED SO LONG AS SUCH COPIES"
[2] "MACHINE READABLE COPIES OF THIS ETEXT, SO LONG AS SUCH COPIES"
```

2.4 **Optional** Explore these other patterns to figure out what they do.

- I have no idea what the asterisk does and I'm having trouble undertsanding it.
- ".MAC[B-Z]" searches for a pattern that can have anything precede "MAC[B-Z]"
- "more$" searches for the pattern "more" at the end of the text
- "^MAC[B|Z]" returns a zero length string because it searches for the pattern of either "MACB" or "MACZ" at the beginning of the text, since most texts have two spaces at the head, there are no returns.

```r
str_subset(macbeth, ".*MAC[B-Z]") %>% head()
```

```
[1] "MACHINE READABLE COPIES MAY BE DISTRIBUTED SO LONG AS SUCH COPIES"
[2] "MACHINE READABLE COPIES OF THIS ETEXT, SO LONG AS SUCH COPIES"
[3] "WITH PERMISSION.  ELECTRONIC AND MACHINE READABLE COPIES MAY BE"
[4] "THE TRAGEDY OF MACBETH"
[5] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
[6] "  LADY MACBETH, his wife"
```

```r
str_subset(macbeth, ".MAC[B-Z]") %>% head()
```

```
    [1] "WITH PERMISSION.  ELECTRONIC AND MACHINE READABLE COPIES MAY BE"
    [2] "THE TRAGEDY OF MACBETH"
    [3] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
    [4] "  LADY MACBETH, his wife"
    [5] "  MACDUFF, Thane of Fife, a nobleman of Scotland"
    [6] "  LADY MACDUFF, his wife"
```

```r
str_subset(macbeth, "more$") %>% head()
```

```
    [1] "    Who, almost dead for breath, had scarcely more"
    [2] "    Hath left you unattended. [Knocking within.] Hark, more"
    [3] "more"
```

```r
# Code below should return character(0) (i.e., nothing)
str_subset(macbeth, "^MAC[B|Z]") %>% head()
```

```
    character(0)
```

Problem 3 **Optional** In section 19.2.2, the **wordcloud** package is used to create a word cloud based on text in abstracts from Data Science articles in arXiv, "a fast-growing electronic repository of preprints of scientific papers from many disciplines" with corresponding package **aRxiv**. I've provided some code below to get you started coding along with the extended example. What words are included in **tidytext** `stop_words` dataset? Do you think all of these words should be considered stop words (i.e. excluded from analysis) in all scenarios? Are there any that might be useful in some contexts? What does `get_stopwords()` do?

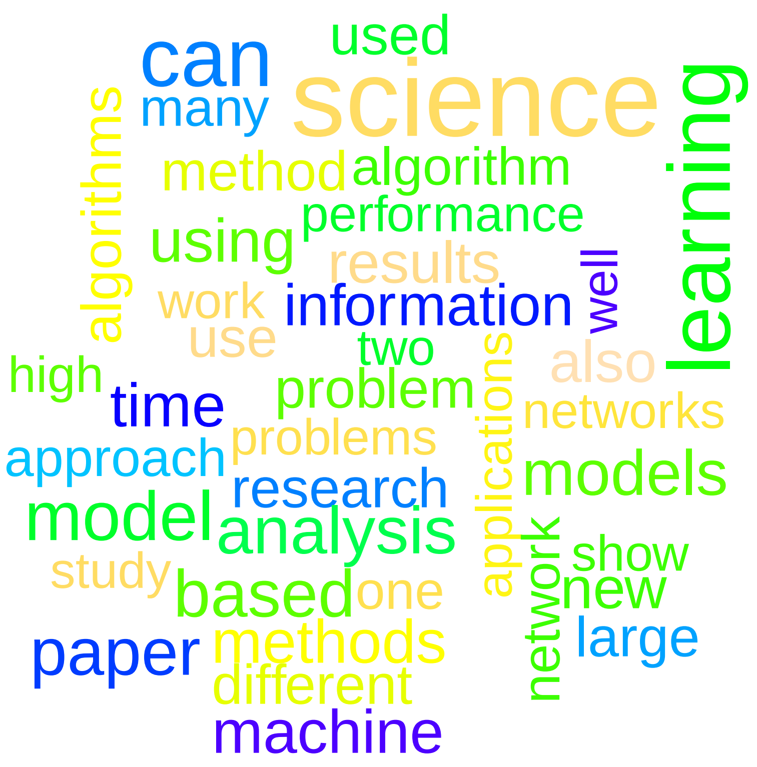
`stop_words` includes words in the english language that are used grammarically and doesn't indicate the subject of the text. `get_stopwords()` gets the library in a tidy format.

```r
glimpse(DataSciencePapers)
```

```
Rows: 1,089
Columns: 15
$ id               <chr> "astro-ph/0701361v1", "0901.2805v1", "0901.3118v2", "~
$ submitted        <chr> "2007-01-12 03:28:11", "2009-01-19 10:38:33", "2009-0~
$ updated          <chr> "2007-01-12 03:28:11", "2009-01-19 10:38:33", "2009-0~
$ title            <chr> "How to Make the Dream Come True: The Astronomers' Da~
$ abstract         <chr> "  Astronomy is one of the most data-intensive of the~
$ authors          <chr> "Ray P Norris", "Heinz Andernach", "O. V. Verkhodanov~
$ affiliations     <chr> "", "", "Special Astrophysical Observatory, Nizhnij A~
$ link_abstract    <chr> "http://arxiv.org/abs/astro-ph/0701361v1", "http://ar~
$ link_pdf         <chr> "http://arxiv.org/pdf/astro-ph/0701361v1", "http://ar~
$ link_doi         <chr> "", "http://dx.doi.org/10.2481/dsj.8.41", "http://dx.~
$ comment          <chr> "Submitted to Data Science Journal Presented at CODAT~
$ journal_ref      <chr> "", "", "", "", "EPJ Data Science, 1:9, 2012", "", "E~
$ doi              <chr> "", "10.2481/dsj.8.41", "10.2481/dsj.8.34", "", "10.1~
$ primary_category <chr> "astro-ph", "astro-ph.IM", "astro-ph.IM", "astro-ph.I~
$ categories       <chr> "astro-ph", "astro-ph.IM|astro-ph.CO", "astro-ph.IM|a~
```

```r
DataSciencePapers <- DataSciencePapers %>%
  mutate(submitted = lubridate::ymd_hms(submitted),
         updated = lubridate::ymd_hms(updated),
         field = str_extract(primary_category, "^[a-z,-]+"),
         compsci = ifelse(field == "cs",
                          "Computer Science",
                          "Other discipline"))

data(stop_words)

arxiv_words <- DataSciencePapers %>%
  unnest_tokens(output = word, input = abstract, token = "words") %>%
  anti_join(get_stopwords(), by = "word") %>%
  select(word, id)

arxiv_word_freqs <- arxiv_words %>%
  count(id, word, sort = TRUE) %>%
  select(word, n, id)

arxiv_abstracts <- arxiv_words %>%
  group_by(id) %>%
  summarize(abstract_clean = paste(word, collapse = " "))

arxiv_papers <- DataSciencePapers %>%
  left_join(arxiv_abstracts, by = "id")
```

```
# May also need to install the "tm" package in order to use the function
set.seed(1966)
arxiv_papers %>%
  pull(abstract_clean) %>%
  wordcloud(max.words = 40,
            scale = c(8, 1),
            colors = topo.colors(n = 30),
            random.color = TRUE)
```


used
can
science
many
algorithms
learning
method
algorithm
performance
using
results
well
work
information
use
two
also
high
time
problem
applications
networks
problems
approach
research
models
model
analysis
network
show
study
based
one
new
paper
methods
large
different
machine