Chapter 19, Part 1: Text as Data

- The data we have studied so far have been (1) numerical; (2) categorical; or (3) character data where each value is a relatively short character string.
- Computers can store large amounts of *text*, for example, documents, books, or even collections of documents and books (called *corpora*).
- Since large amounts of text can be stored as data, recently packages have been developed to wrangle and analyze such text data.

Regular Expressions and Retrieving Text Files

- Special commands called *regular expressions* can be used to parse large quantities of text and find strings of interest.
- We will see a number of these in our example analysis of William Shakespeare's famous play, *Macbeth*.
- The complete text of *Macbeth* is available as a `.txt` file, preloaded in the `mdsr` package.
- We can also import a text file from a website using the `getURL` function in the `RCurl` package.
- Many older literary works are freely available as text files on the *Project Gutenberg* website, so we can retrieve full-text versions of many works in the public domain, not only *Macbeth*.

Splitting Character Strings

- Initially, the object `Macbeth_raw` is a single massive string of text containing the entire work (a character vector of length 1).
- We can split this into a character vector where each element of the vector is one line of text.
- We use the `str_split` function to split the string at the end-of-line characters (`"\r\n"`).
- Now we have a character vector called `macbeth` of length 3194, each element being a line of the document.

Finding Patterns in the Text

- Conveniently, each speaking line begins with two spaces followed by the speaking character's name in all capital letters.
- We can search for all lines when the character Macbeth speaks using

```
macbeth_lines <- macbeth %>%
str_subset("  MACBETH")
```

- The first argument of the `str_subset` function (here, `macbeth`) is the object through which we want to search.
- The second argument of the `str_subset` function (here, `"  MACBETH"`) is the expression we want to find.
- It returns all the lines of text in which the specified expression is found.

Other Functions for Finding Patterns

- The `str_detect` function works similarly, but it returns a logical vector of the same length as the number of lines in the document.
- The outputted logical vector will be `TRUE` for every element where the pattern is detected and `FALSE` for every element where the pattern is not detected.
- The `str_extract` function from the `stringr` package will extract the piece of each matching line that actually matched.
- See the examples with the *Macbeth* text data.

Regular Expression Syntax

- In the previous example, we searched for a specific string, but using *regular expressions* can make our searches much more flexible.
- Regular expressions are very commonly used for text mining in many programming languages, not just R.
- The period symbol (`.`) is a *metacharacter*, i.e, a wild card that can stand for any character, in searches.
- If the string you want to search for contains an actual period, you must precede the period with a backslash (or in R, actually two backslashes).
- See the examples with the *Macbeth* text analysis.

Searching for Any of a Set of Characters

- Putting sets of characters in brackets (like `[A-D]` or `[1-5]`) indicates you are searching for any character in that set.
- The specification `(A|D)` indicates you are searching for `A` or `D` (and not the letters in between).
- Placing the symbol `^` before the character string you are searching for indicates you are looking only for instances when the string is at the beginning of the line of text.
- Conversely, the symbol `$` indicates you are looking only for instances when the specified string is at the end of the line of text.
- See the *Macbeth* examples.

Searching for Repeated Patterns

- Certain special characters allow you to search for instances when characters are repeated a specific number of times:
  - `?` indicates zero or one time
  - `*` indicates zero or more times
  - `+` indicates one or more times
- This quantification is applied to the element in the pattern just before the `?`, `*`, or `+` symbol.
- Since the speaking lines in the `macbeth` object always start with two spaces, we can use these tools to distinguish between occurrences in speaking lines and in other lines (see examples).

Example of Analyzing Speaking Frequency of Four Characters in *Macbeth*

- Section 19.1.2 has a detailed example of putting these tools together to get a picture of how often the characters Macbeth, Lady Macbeth, Banquo, and Duncan speak during the five acts of the play.
- By converting logical vectors returned by `str_detect` to numeric vectors of 1's and 0's, we can actually plot the speaking frequencies as a function of line number over the duration of the play.
- Spoiler warning: This example indicates the fates of these characters in the storyline!
- Later, we will use these and other tools to do a brief analysis of a work written by your favorite professor.