Introduction to Text Analysis

POS6933: Computational Social Science

Jake S. Truscott, Ph.D

University of Florida

Spring 2026

Overview

- Non-Traditional Data Structures
- Using Text in R
- Retrieving Text Data
- Creating a Corpus

Motivation

- Many important political and social phenomena are expressed in language, not numbers
- Votes, surveys, and roll calls capture outcomes, but not always reasoning, framing, or strategy
- Text (in particular) allows us to observe (among other things):
  - Preferences before choices
  - Strategical signaling
  - Agenda setting & framing
- Operative Goal of Social Science: **Represent sophisticated human behaviors quantitatively** – Viewing text as data is a similar vein.

Defining *Non-Traditional Data*

- **Unstructured or Semi-Structured Data** – i.e., data that do not come in a fixed or rigid (numeric) format. Instead consist of free-form content where structure must be inferred rather than assumed.
- Examples:
  - Speeches, Debates, and Oral Arguments
  - Judicial Opinions (and Separate Opinions)
  - News Articles & Editorials
  - Social Media Posts
  - Legislative Text & Statutes
  - Manifestos & Party Platforms
  - Books, Academic Articles, and Manuscripts

What Can We Derive from Text?

- Can be Used to Measure:
  - Ideology
  - Sentiment, Tone, or Feeling
  - Issue or Topic Attention
  - Similarity or Coordination
- **Key Assumption**: Language reflects latent traits – i.e., it represents either unseen or undefined characteristics, much in the same way that we can prescribe a series of votes in Congress as reflecting conservative or liberal tendencies.

Advantages and Shortcomings

Advantages

- Captures nuance and context very well
- Methodologies are often very flexible
- Provides for qualitative inferences and ex ante assessment
- Scales to large corpora and across time, actors, institutions, etc.
- Can be abundant where numerical data is sparse

Shortcomings

- Measurement depends on modeling choices
- Technical complexity ranges considerably – tradeoff motivated by a priori expectations and ability to capture high dimensional relationships
- Often introduces high dimensionality – risks of overfitting, drawing inferrential value from spurious relationships, etc.
- Generally requires validation against known qualities when used for measurement

Key Steps

- Retrieving Text (**Today**)
- Pre-processing & Reducing Feature Complexity (**Today**)
- Creating a Corpus (**Today**)
- Modeling Text
- Analysis

R Data Types

- `numeric` – (1, 2, 3, 4, 5)
- `integer` – (1L, 50L, 100L)
- `complex` – (9+3i)
- `character` – (’text strings’)
- `logical` – (TRUE or FALSE)

Using Text in R

- `R` (and `Python`) are both very flexible for handling `character` (`string`) data
- Countless sources of text data – from a single haiku to bounded volumes providing an expansive anthology of human knowledge, we can use text analysis tools to bridge an entire domain of qualitative and quantitative inquiry.

Text Object in R

```r
sample_text <- "This is Sample Text"
print(sample_text)
```

```
[1] "This is Sample Text"
```

```r
sample_vector <- c("Sample 1", "Sample 2", "Sample 3")
print(sample_vector)
```

```
[1] "Sample 1" "Sample 2" "Sample 3"
```

Regular Expressions

- A regular expression (**regex**) is a pattern used to search, match, or manipulate text
- In essence, it is compact language for telling `R` what text should look like, not what it should be exactly.
- We will use these in conjunction with functions like `grep`, `grepl`, and `gsub` to retrieve and manipulate text.

Regex Examples

text < − ‘This is a sample string with a date 2026-02-06, a time 14:30, an email test.user@example.com, and the number 42.’

```r
library(stringr)

unlist(stringr::str_extract_all(text, "\\b[a-zA-Z]+\\b"))  # All Text
```

```
 [1] "This"    "is"      "a"       "sample"  "string"  "with"    "a"       "date"    "a"       "time"   
[12] "email"   "test"    "user"    "example" "com"     "and"     "the"     "number" 
```

```r
unlist(stringr::str_extract_all(text, "\\b\\d+\\b"))  # All Numbers
```

```
[1] "2026" "02"   "06"   "14"   "30"   "42"  
```

Regex Examples (Cont.)

```r
unlist(stringr::str_extract(text, "\\b\\d{4}-\\d{2}-\\d{2}\\b"))  # Grab Date
```

```
[1] "2026-02-06"
```

```r
unlist(stringr::str_extract(text, "\\b\\d{2}:\\d{2}\\b"))  # Grab Time HH:MM
```

```
[1] "14:30"
```

```r
unlist(stringr::str_extract(text, "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}"))  # Email Address
```

```
[1] "test.user@example.com"
```

```r
unlist(stringr::str_extract(text, "^[^,]+"))  # All Before 1st Comma
```

```
[1] "This is a sample string with a date 2026-02-06"
```

```r
unlist(stringr::str_replace_all(text, "[[:punct:]]",
    ""))  # Remove Punctuation
```

```
[1] "This is a sample string with a date 20260206 a time 1430 an email testuserexamplecom and the number
```

Important Functions w/ Regular Expressions

```
set.seed(1234)

string <- "The quick brown fox jumps over the lazy dog"

gsub("quick", "wild", string)  # Replace Quick
```

```
[1] "The wild brown fox jumps over the lazy dog"
```

```
grepl("quick brown", string, ignore.case = F)  # Check String
```

```
[1] TRUE
```

Partitioning Text (Lorem Ipsum)

lorem_ipsum $< -$ "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas scelerisque eros nec libero luctus, a gravida augue dictum. Integer sem est, malesuada nec mi ut, venenatis pellentesque massa. Mauris ac ex odio. Integer eget est lacus. Ut varius, sapien nec efficitur malesuada, sem lectus aliquet lacus, ac efficitur ipsum mauris vitae augue. Aliquam ornare faucibus nibh, a varius mauris mattis id. Nullam eu nibh aliquam, vestibulum neque sed, sagittis mi. Etiam blandit facilisis sagittis. Duis ut dolor sed nibh egestas porta. Aenean quis lorem nec augue semper convallis lacinia eget orci. Nam eget dolor tortor."

```r
unlist(stringr::str_split(lorem_ipsum, pattern = '\\.[[:space:]]'))[1:5]
```

[1] "Lorem ipsum dolor sit amet, consectetur adipiscing elit"
[2] "Maecenas scelerisque eros nec libero luctus, a gravida augue dictum" [3] "Integer sem est, malesuada nec mi ut, venenatis pellentesque massa" [4] "Mauris ac ex odio"

[5] "Integer eget est lacus"

Additional Considerations Re: Regular Expressions

- Punctuation need to double-backslashes
- They are very literal – make sure you're considering what you're asking it to retrieve/search.
- Some tools/functions for applying regular expressions are different in application (e.g., dplyr, stringr, stringi, tm, etc.) – make sure you're considering how they approach splits at punctuation!
- A lot of this is trial & error

Regex Practice

Practice Sentence

- **Sentence 1**: The cautious archivist indexed seven obscure manuscripts before dawn.
- **Sentence 2**: Tomorrow a reckless cyclist shattered records while racing before sunset.

- Write a single regex to recover the time of day mentioned in each sentence – and only the time of day (*Hint*: Make sure you remove punctuation...)

Regex Practice (Cont.)

```r
s1 <- "The cautious archivist indexed seven obscure manuscripts before dawn."
s2 <- "Tomorrow a reckless cyclist shattered records while racing before sunset."

sentences <- c(s1, s2)

gsub(".*before ", "", gsub("\\.", "", sentences))
```

```
[1] "dawn"   "sunset"
```

Lies, Damn Lies, and Statistics (2000)

Lies, Damn Lies, and Statistics (2000)

- Aaron Sorkin's *The West Wing* ran from 1999-2007 – easily the best political drama ever created.
- *Data for Progress* +23% Favorable Rating (higher among older, more educated, and higher-income Americans)
- 100 Awards from 289 Nominations (27 Emmys, 2 Peabody Awards, 6 SAG Awards, among others) – it's incredible.

Lies, Damn Lies, and Statistics (2000)

- *Lies, Damn Lies, and Statistics* premiered May 10, 2000
- Title sourced from Mark Twain (or Benjamin Disraeli): *There are lies, damned lies, and statistics* – used to describe instances where people are given credibility for often weak or disagreeable positions by using statistics to sound empirically rigid.
- **Synopsis**: The Bartlet administration anxiously awaits crucial polling data following a shift in strategy, while managing political crises. Despite internal pessimism and a personal scandal involving Sam, the poll shows a surprising nine-point increase in approval.
- Opening Scene

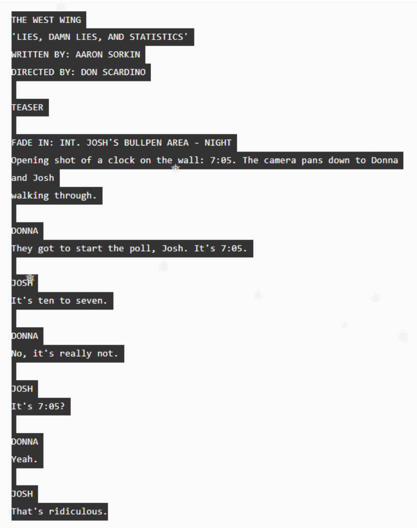
Lies, Damn Lies, and Statistics (2000)

- Let's imagine I wanted to know which character spoke the most lines.
- **How could I do that?**

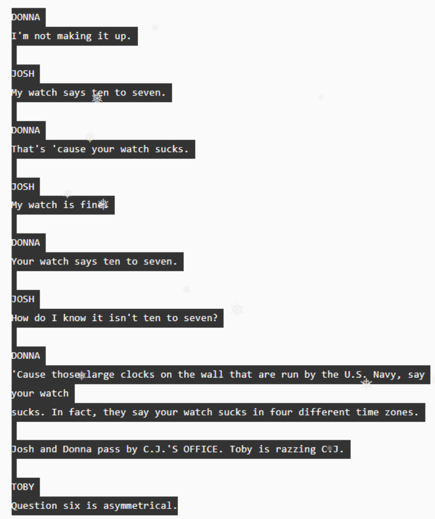
Lies, Damn Lies, and Statistics (2000)

- Let's imagine I wanted to know which character spoke the most lines.
- **How could I do that?**
- Options:
  - Watch the episode and count utterances
  - Read the closed captioning transcription and (again) count utterances
  - Recover the close captioning transcript and use our computing resources to accurately (and quickly) analyze

First Look at Transcript



```
THE WEST WING
'LIES, DAMN LIES, AND STATISTICS'
WRITTEN BY: AARON SORKIN
DIRECTED BY: DON SCARDINO

TEASER

FADE IN: INT. JOSH'S BULLPEN AREA - NIGHT
Opening shot of a clock on the wall: 7:05. The camera pans down to Donna
and Josh
walking through.

DONNA
They got to start the poll, Josh. It's 7:05.

JOSH
It's ten to seven.

DONNA
No, it's really not.

JOSH
It's 7:05?

DONNA
Yeah.

JOSH
That's ridiculous.
```



```
DONNA
I'm not making it up.

JOSH
My watch says ten to seven.

DONNA
That's 'cause your watch sucks.

JOSH
My watch is fine.

DONNA
Your watch says ten to seven.

JOSH
How do I know it isn't ten to seven?

DONNA
'Cause those large clocks on the wall that are run by the U.S. Navy, say
your watch
sucks. In fact, they say your watch sucks in four different time zones.

Josh and Donna pass by C.J.'S OFFICE. Toby is razzing C.J.

TOBY
Question six is asymmetrical.
```

Goal

- Clearly, the text is not as organized as I'd like it to be.
- **Remember**: Organizing text data is like deciphering the wording of variables – you can write generalizable coding routines to parse text but a lot of this work is going to be application-specific
- My next steps will be to consciously try to develop a 4-column dataframe that identifies:
  - **Character**: Which character is currently speaking.
  - **Dialogue**: The text (string)
  - **Word Count**: How many words are found in the text
  - **Line Number**: The current number of dialogue entries to that point
- I'm only interested in Josh, Toby, C.J., Donna, Sam, Leo, and President Bartlet

Plan of Attack

- Recover .txt file of episode script
- Convert to dataframe
- Use a regular expression (regex) to identify & partition text for each character
- Count number of utterances and words in each utterance

Recover Script

```r
west_wing <- readLines(west_wing_script_location, warn = FALSE)  # Read Txt from GitHub Repo

head(west_wing)  # Print Head
```

```
[1] "THE WEST WING"                "’LIES, DAMN LIES, AND STATISTICS’" "WRITTEN BY: AARON SORKIN"
[4] "DIRECTED BY: DON SCARDINO"    ""                                  "TEASER"
```

Process Script

```
# A tibble: 10 x 4
   character dialogue                                  id word_count
   <chr>     <chr>                                  <int>      <int>
 1 DONNA     They got to start the poll, Josh. It’s 7:05.     1          9
 2 JOSH      It’s ten to seven.                         2          4
 3 DONNA     No, it’s really not.                       3          4
 4 JOSH      It’s 7:05?                                 4          2
 5 DONNA     Yeah.                                      5          1
 6 JOSH      That’s ridiculous.                         6          2
 7 DONNA     I’m not making it up.                      7          5
 8 JOSH      My watch says ten to seven.                8          6
 9 DONNA     That’s ’cause your watch sucks.            9          5
10 JOSH      My watch is fine.                         10          4
```

Measure Speaker Variance

```r
damn_lies %>%
    group_by(character) %>%
    summarise(total_words = sum(word_count), average_words = round(mean(word_count)),
        total_lines = n()) %>%
    arrange(desc(total_words)) %>%
    rename(Character = character, `Total Words` = total_words,
        `Average Words` = average_words, `Total Lines` = total_lines)
```

```
# A tibble: 7 x 4
  Character ‘Total Words‘ ‘Average Words‘ ‘Total Lines‘
  <chr>             <int>           <dbl>         <int>
1 BARTLET            1359               9           145
2 C.J.                985              11            91
3 JOSH                757              11            67
4 LEO                 679               8            90
5 TOBY                617               8            78
6 SAM                 454               6            75
7 DONNA               162               8            20
```

```r
# For Each Character -- Summarize Total Words,
# Avg. Per Utterance, and Total Utterances
```

Measuring Length of Supreme Court Opinions (Black & Spriggs 2008)

- Main reading: Black & Spriggs 2008
- What is the methodology?
- What is the main finding?

Measuring Words in Supreme Court Oral Arguments (Dobbs v. Jackson)

Practice

- Using the Supreme Court's argument in *Dobbs v. Jackson* (2021), filter to `role = justice` and recover the total utterances of each `speaker`, as well as the total words for each.

```
dobbs <- get(load("data/class_4/dobbs_19-1392.rdata"))  # Load Dobbs
```

Dobbs – Utterances

```
dobbs %>%
    filter(role == "Justice") %>%
    group_by(speaker) %>%
    summarise(utterances = n()) %>%
    arrange(desc(utterances))  # Utterances
```

```
# A tibble: 9 x 2
  speaker                utterances
  <chr>                       <int>
1 John G. Roberts, Jr.           39
2 Sonia Sotomayor                37
3 Samuel A. Alito, Jr.           29
4 Clarence Thomas                20
5 Stephen G. Breyer              14
6 Amy Coney Barrett              12
7 Brett M. Kavanaugh             10
8 Neil Gorsuch                   10
9 Elena Kagan                     6
```

Dobbs – Total Words

```r
dobbs %>%
    filter(role == "Justice") %>%
    group_by(speaker) %>%
    summarize(text = paste(text, collapse = " "), .groups = "drop") %>%
    tidytext::unnest_tokens(word, text) %>%
    group_by(speaker) %>%
    summarise(word_count = n(), .groups = "drop") %>%
    arrange(desc(word_count))  # Words Spoken
```

```
# A tibble: 9 x 2
  speaker              word_count
  <chr>                     <int>
1 Sonia Sotomayor            1467
2 Stephen G. Breyer          1310
3 John G. Roberts, Jr.       1222
4 Brett M. Kavanaugh         1041
5 Amy Coney Barrett          1025
6 Samuel A. Alito, Jr.        844
7 Elena Kagan                 685
8 Clarence Thomas             556
9 Neil Gorsuch                524
```

gutenbergr Repository

- Interfaces with Project Gutenberg to download public-domain texts directly into R
- Returns texts in tidy data frames, making them easy to merge, filter, and analyze
- Supports metadata queries (author, title, language, subject, ID)
- Ideal for large-scale text analysis and reproducible workflows

gutenbergr Repository (Cont.)

```r
library(gutenbergr)
gutenberg_metadata %>%
    filter(title == "Oliver Twist")
```

```
# A tibble: 4 x 8
  gutenberg_id title        author           gutenberg_author_id language gutenberg_bookshelf
         <int> <chr>        <chr>                          <int> <fct>    <chr>
1          730 Oliver Twist Dickens, Charles                  37 en       "Category: Novels/Category: C~
2         9727 Oliver Twist Dickens, Charles                  37 en       ""
3        16023 Oliver Twist Dickens, Charles                  37 fr       "FR Littérature/Category: Nov~
4        56586 Oliver Twist Dickens, Charles                  37 de       "Category: Novels/Category: C~
```

```r
oliver_twist <- gutenberg_download(730)  # Download Oliver Twist
```

Practice – *Oliver Twist* by Charles Dickens

Practice Task – Work w/ Classmate

- Using `gutenbergr` – Recover the text from *Olive Twist* by Charles Dickens
- Construct a regular expression to identify chapters and breaks (*Hint*: Use `regex` cheat sheet!)
- Partition the text to two columns – `Chapter` & `Text`
- Return table using `stargazer` to identify the total and unique volume of words for each chapter.

Practice – *War and Peace* by Leo Tolstoy

Practice Task – Work w/ Classmate

- Using `gutenbergr` – Recover the text from *War & Peace* by Leo Tolstoy
- Construct a regular expression to identify chapters and breaks (*Hint*: Use `regex` cheat sheet!)
- Partition the text to two columns – `Chapter` & `Text`
- Return table using `stargazer` to identify the total and unique volume of words for each chapter.

Practice – Pick Another Book!

Complete the same task (again) but with a book or document of your choice (`gutenbergr` – You and a classmate will present it to the class.

The Pipeline

Raw Text → Corpus (**You're HERE**) → Tokens → Features (DFM) → Models / Analysis

Corpus

- **Corpus** A structured & systematic collection of texts that you treat as data rather than as individual documents.
- **Main Idea**: Once text is in a corpus, you stop reading it line-by-line and start analyzing patterns across many texts.
- Allows for multiple texts to assume consistent (comparable) structure and includes associated metadata
- We're going to practice today constructing a corpus with `quanteda`

Sample Corpus Using Quanteda

```r
library(quanteda)  # Load Quanteda
```

```
Package version: 4.0.2
Unicode version: 15.1
ICU version: 74.1

Parallel computing: 14 of 14 threads used.

See https://quanteda.io for tutorials and examples.
```

```r
texts <- c("The quick brown fox jumps over the lazy dog.",
    "Data science is revolutionizing the way we analyze information.",
    "Text analysis in R is fun and informative!")  # Sample Texts (as vector)

texts_with_meta <- tibble(doc_id = c("sentence_1",
    "sentence_2", "sentence_3"), text = texts, author = c("Josh",
    "Leo", "Toby"), date = as.Date(c("2025-01-01",
    "2025-01-02", "2025-01-03")))  # Create Metadata for Texts (Same as tm example!)

quanteda_corpus <- corpus(texts_with_meta, text_field = "text")
```

Sample Corpus Using Quanteda

```
Corpus consisting of 3 documents, showing 3 documents:

       Text Types Tokens Sentences author       date
 sentence_1    10     10         1   Josh 2025-01-01
 sentence_2    10     10         1    Leo 2025-01-02
 sentence_3     9      9         1   Toby 2025-01-03
```

Lies, Damn Lies, and Statistics (Again!)

```
damn_lies_corpus <- quanteda::corpus(damn_lies, text_field = "dialogue")  # Create Corpus (Text = 'dialog

summary(damn_lies_corpus[1:10])  # Inspect (Just First Couple of Rows)
```

```
Corpus consisting of 10 documents, showing 10 documents:

   Text Types Tokens Sentences character id word_count
  text1    13     14         2     DONNA  1          9
  text2     5      5         1      JOSH  2          4
  text3     6      6         1     DONNA  3          4
  text4     5      5         1      JOSH  4          2
  text5     2      2         1     DONNA  5          1
  text6     3      3         1      JOSH  6          2
  text7     6      6         1     DONNA  7          5
  text8     7      7         1      JOSH  8          6
  text9     7      7         1     DONNA  9          5
 text10     5      5         1      JOSH 10          4
```

Lies, Damn Lies, and Statistics (Again! Cont.)

```
damn_lies_corpus[1]
```

```
Corpus consisting of 1 document and 3 docvars.
text1 :
"They got to start the poll, Josh. It’s 7:05."
```

```
quanteda::docvars(damn_lies_corpus[1])
```

```
  character id word_count
1     DONNA  1          9
```

Corpus Practice

Using one of the `gutenbergr` texts from today – Construct a corpus, including both the text and the chapter metadata

Looking Forward (Next Class)

- The Bag of Words – Or, how we can use words for more than just descriptive statistics.