Censorship and Narrative Control: A Textual and Comparative Analysis of Florida's and Iowa's Banned Books

Grace O'Malley

2025-04-27

Table of contents

1 Introduction

All code chunks that are part of preprocessing have been marked as 'eval=FALSE' for rendering purposes. Any data that is included will be read in with readr::read_csv for rendering. I have noted the save points where data is read into the document for your convenience. If you wish to

test any of the data preprocessing code, simply remove the 'eval=FALSE' tag and run the chunk. This is very linear in terms of how it's set up, but it allows for a lot of flexibility in terms of what can be done with the data, but bear that in mind if you are trying to test code that relies on previous steps being completed. To view any visuals not included in my final paper, run all chunks linearly from

1.1 Required Packages

This is the listing of required packages to run the code below. There is a for loop that can be run to install and library any that are needed.

```
required_packages <- c("jsonlite", "httr", "stringdist", "quanteda", "gutenbergr",
                       "tokenizers", "tidytext", "wordcloud", "RColorBrewer", "gg
                       "openNLP", "ggsci", "glmnet", "xgboost", "yardstick", "pR

# Check if all required packages are installed and install them if not
for (pkg in required_packages) {
    if (!requireNamespace(pkg, quietly = TRUE)) {
        renv::install(pkg)
    }
        library(pkg, character.only = TRUE)
}
data("gutenberg_metadata", package = "gutenbergr")
```

1.2 Data Collection

The banned books list was downloaded from PEN America's website (link).

```
# assign file path to variable
banned_books_file_path <- "/Users/coniecakes/Library/CloudStorage/OneDrive-Persona

banned_books_list <- readxl::read_excel(banned_books_file_path, sheet = "Sorted by

utils::head(banned_books_list, 10)
```

1.3 Exploratory Data Analysis

I reviewed the data set and removed unnecessary data and focused on my intended target regions of Florida and Maryland. After viewing the number of available banned books from Maryland, I selected Iowa as an alternative region because Maryland had a limited amount of texts available.

```
colnames(banned_books_list) <- c("Title", "Author", "Secondary_Author", "Illustrat
                                 "Series_Name", "State", "District", "Date_Removed"

banned_books_cleaned <- banned_books_list %>%
    dplyr::select(Title, Author, State, District, Date_Removed, Ban_Status, Initia
    dplyr::mutate(Date_Removed = as.Date(lubridate::my(Date_Removed)))

head(banned_books_cleaned,10)

banned_books_cleaned %>%
    dplyr::summarise(dplyr::across(dplyr::everything(), ~ sum(is.na(.)), .names =

fl_banned_books_list <- banned_books_cleaned %>%
    dplyr::filter(State == "Florida") %>%
    dplyr::distinct(Title, .keep_all = TRUE)

md_banned_books_list <- banned_books_cleaned %>%
    dplyr::filter(State == "Maryland") %>%
    dplyr::distinct(Title, .keep_all = TRUE)

ia_banned_books_list <- banned_books_cleaned %>%
    dplyr::filter(State == "Iowa") %>%
    dplyr::distinct(Title, .keep_all = TRUE)
```

1.4 Data Sampling

I sampled 10 book titles from the state of Florida's banned books list. I originally wanted to use Maryland for comparison, but they only had 6 titles, so I selected Maryland as my subgroup comparison.

```r
# select random seed
seed <- 245
set.seed(seed)

# sample books from Florida & New York
fl_books_sample <- fl_banned_books_list %>%
    dplyr::sample_n(10)

md_books_sample <- md_banned_books_list %>%
    dplyr::sample_n(10)

# print sampled books
print(fl_books_sample)
print(md_books_sample)
```

1.5 Data Collection

1.5.1 Gutenberg Search

I began to search for data online. I cleaned the book titles from the Florida sample and cleaned the titles from the Gutenberg library (`gutenbergr`) to ensure they matched. I then compared the two lists and found that none of the Florida titles were also in the Gutenberg library. This caused me to pivot to search for any ISBNs from the Florida banned books list sample to attempt to search them in other libraries.

```r
# search for titles in gutenberg library
fl_sample_books_titles_cleaned <- fl_books_sample %>%
    dplyr::select(Title) %>%
    dplyr::pull() %>%
    tolower() %>%
    stringr::str_trim()

gutenberg_titles_cleaned <- gutenbergr::gutenberg_works() %>%
    dplyr::select(title) %>%
    dplyr::pull() %>%
```

```
    tolower() %>%
    stringr::str_trim()

in_gutenberg <- c()

for(i in fl_banned_books_list_cleaned ) {
  if (i %in% gutenberg_titles_cleaned) {
    in_gutenberg <- c(in_gutenberg, i)
  }
}

in_gutenberg
```

ISBN Search

I attempted to search by ISBN, but could not write a function that would effectively locate the ISBNs of the texts I needed to find (see appendices for unused code). Since the ISBN search was unsuccessful, I needed to pivot my search techniques.

Title Search

The next course of action leveraged the book titles in classic text searches. I cleaned all book titles from the Florida banned books list and compared them to the Gutenberg library. My goal is to find a suitable sample of texts here that I can use before having to find alternative methods to obtain these texts. I wrote a function to return the matched books based on the gutenberg library titles.

```
# create a function to search for titles in a library
full_gutenberg_search <- function(banned_books, gutenberg_titles) {
    matched_books <- banned_books[banned_books %in% gutenberg_titles]
    return(matched_books)
}
```

I identified titles that I can pull from the Gutenberg library for my project - however I will have to pivot since there are less than 10 books available from the Maryland banned books list. The intended sample size is 10 texts, which are readily available from Florida (53) and a sufficient number from

Iowa (16). After matching the titles, I will see how many I can downloand uncorrupted - if there is any data corruption, I may need to revise my sample numbers or find alternative data sources.

```
fl_books_titles_cleaned <- fl_banned_books_list %>% # cleaned fl titles list
    dplyr::select(Title) %>%
    dplyr::pull() %>%
    tolower() %>%
    stringr::str_trim()

md_books_titles_cleaned <- md_banned_books_list %>% # cleaned md titles list
    dplyr::select(Title) %>%
    dplyr::pull() %>%
    tolower() %>%
    stringr::str_trim()

ia_books_titles_cleaned <- ia_banned_books_list %>% # cleaned ia titles list
    dplyr::select(Title) %>%
    dplyr::pull() %>%
    tolower() %>%
    stringr::str_trim()

fl_match_list <- full_gutenberg_search(fl_books_titles_cleaned, gutenberg_titles_c
fl_match_list

md_match_list <- full_gutenberg_search(md_books_titles_cleaned, gutenberg_titles_c
md_match_list

ia_match_list <- full_gutenberg_search(ia_books_titles_cleaned, gutenberg_titles_c
ia_match_list
```

To maximize my search returns, I created a fuzzy title search function to allow proper title matching and minimizing the effects of grammar differences.

```
fuzzy_title_search <- function(book_list) {
    gutenberg_metadata <- gutenbergr::gutenberg_works()
    matched_books <- book_list %>%
```

```
        sapply(function(book) {
            distances <- stringdist::stringdist(book, gutenberg_metadata$title, me
            closest_match <- gutenberg_metadata$title[which.min(distances)]
            return(closest_match)
        })
        return(matched_books)
}
```

I pulled the random sample of book titles from the complete pool of available book titles from Florida and Iowa, respectively.

```
ia_sample <- sample(ia_match_list, 10)
fl_sample <- sample(fl_match_list, 10)

cat("IA Sample:\n",{ia_sample}, sep="\n")
cat("\nFL Sample:\n",{fl_sample}, sep="\n")
```

The fuzzy matching title search yielded matches for all but one of the sampled titles, which I've identified and will add in manually by matching the gutenberg id number.

```
# Iowa fuzzy matching
ia_fuzzy_matches <- fuzzy_title_search(ia_sample) # incorrectly selected 1 title
ia_fuzzy_matches

ia_fuzzy_matches_full <- fuzzy_title_search(ia_match_list)

# Florida fuzzy matching
fl_fuzzy_matches <- fuzzy_title_search(fl_sample)
fl_fuzzy_matches

fl_fuzzy_matches_full <- fuzzy_title_search(fl_match_list)
```

I ran into a lot of problems downloading books. After my first pass, I was only able to get 6 / 20. I adjusted to search for the 10 books from Florida and all 16 books from Iowa to see if that produced any additional downloads,

but there was only a marginal improvement on my second pass - I was only able to retrieve 10/26. The next step is to download them directly from Project Gutenberg's website, which should be more reliable. If I cannot obtain enough texts, I may need to adjust my sample sizes for subgroup comparison.

```
# retrieve Iowa text ids
ia_gutenberg_ids <- gutenbergr::gutenberg_works() %>%
  dplyr::filter(title %in% ia_fuzzy_matches) %>%
  dplyr::select(gutenberg_id, title)
# correct "Dead End" entry
correct_entry <- gutenbergr::gutenberg_works() %>%
    dplyr::filter(tolower(title) == "dead end") %>%
    dplyr::select(gutenberg_id, title)
correct_entry
ia_gutenberg_ids$gutenberg_id[10] <- correct_entry$gutenberg_id
ia_gutenberg_ids$title[10] <- correct_entry$title

ia_gutenberg_ids_full <- gutenbergr::gutenberg_works() %>%
  dplyr::filter(title %in% ia_fuzzy_matches_full) %>%
  dplyr::select(gutenberg_id, title)

# retrieve Florida text ids
fl_gutenberg_ids <- gutenbergr::gutenberg_works() %>%
  dplyr::filter(title %in% fl_fuzzy_matches) %>%
  dplyr::select(gutenberg_id, title)

fl_gutenberg_ids_full <- gutenbergr::gutenberg_works() %>%
  dplyr::filter(title %in% fl_fuzzy_matches_full) %>%
  dplyr::select(gutenberg_id, title)
```

To pull texts directly from Project Gutenberg's website, I wrote a short function to download texts and identify any texts that were not downloaded in the process.

```
load_gutenberg_text <- function(book_id) {
    url <- paste0("https://www.gutenberg.org/files/", book_id, "/", book_id, "-0.t
```

```
    response <- httr::GET(url)
    if (httr::status_code(response) == 200) {
        text <- httr::content(response, "text", encoding = "UTF-8")
        return(text)
    } else {
        message(paste("Book ID", book_id, "could not be loaded."))
        return(NULL)
    }
}
```

I was able to pull in all of the texts necessary for the Florida banned books list - both through dowload from `gutenbergr` and the Project Gutenberg website. I've assigned it to a data frame `fl_book_texts_df`.

```
# download Florida texts and set up corpus
fl_book_texts <- gutenbergr::gutenberg_download(fl_gutenberg_ids$gutenberg_id)
fl_book_texts <- fl_book_texts %>%
    dplyr::group_by(gutenberg_id) %>%
    dplyr::summarise(text = paste0(text, collapse = " "))
# identify missing texts
fl_missing_books <- fl_gutenberg_ids %>%
  dplyr::filter(!(gutenberg_id %in% fl_book_texts$gutenberg_id))

fl_missing_book_texts_list <- list()
for (i in seq_along(fl_missing_books$gutenberg_id)) {
    book_id <- fl_missing_books$gutenberg_id[i]
    book_title <- fl_missing_books$title[i]
    book_text <- load_gutenberg_text(book_id)
    if (!is.null(book_text)) {
        fl_missing_book_texts_list[[as.character(book_id)]] <- data.frame(
            gutenberg_id = book_id,
            text = book_text,
            stringsAsFactors = FALSE
        )
    }
}
fl_missing_book_texts_df <- dplyr::bind_rows(fl_missing_book_texts_list)
```

```
fl_book_texts_df <- dplyr::bind_rows(fl_book_texts, fl_missing_book_texts_df) # fl
```

I pulled the book title, id, and text for the Iowa sample and added in the missing book title from the fuzzy matching. I had a number of issues downloading book texts so I had to do some exta work obtaining the data. Ultimately, I ended up with `ia_book_texts_df`.

```
ia_books_text_list <- list()
for (i in seq_along(ia_gutenberg_ids$gutenberg_id)) {
    book_id <- ia_gutenberg_ids$gutenberg_id[i]
    book_title <- ia_gutenberg_ids$title[i]
    book_text <- load_gutenberg_text(book_id)
    if (!is.null(book_text)) {
        ia_books_text_list[[as.character(book_id)]] <- data.frame(
            gutenberg_id = book_id,
            title = book_title,
            text = book_text,
            stringsAsFactors = FALSE
        )
    }
}

ia_book_texts_df <- dplyr::bind_rows(ia_books_text_list)

# replace 1 book that could not be downloaded
missing_books <- ia_gutenberg_ids_full %>%
  dplyr::filter(!(title %in% ia_book_texts_df$title))

if (nrow(missing_books) > 0) {
    ia_additional_sample <- missing_books %>%
        dplyr::slice_sample(n = 1)
    print(ia_additional_sample)
} else {
    print("No additional books available for sampling.")
}

ia_additional_sample_text <- load_gutenberg_text(ia_additional_sample$gutenberg_id
```

```r
ia_additional_sample_text_df <- data.frame(gutenberg_id = book_id,
            title = book_title,
            text = book_text,
            stringsAsFactors = FALSE)
ia_book_texts_df <- dplyr::bind_rows(ia_book_texts_df, ia_additional_sample_text_d

dreams_end <- data.frame(gutenberg_id = as.integer(68179), text = load_gutenberg_t

ia_book_texts_df <- dplyr::bind_rows(ia_book_texts_df, dreams_end)

ia_book_texts_df <- ia_book_texts_df %>%
    dplyr::select(-title) %>%
    dplyr::bind_rows(dreams_end) # Iowa banned books list

# had issues downloading books again - had to run a separate download to retrieve

redeemed <- gutenbergr::gutenberg_download(59277)
extra_book <- gutenbergr::gutenberg_download(16348)

redeemed <- redeemed %>%
    dplyr::summarise(text = paste0(text, collapse = " ")) %>%
    dplyr::mutate(gutenberg_id = as.integer(59277)) %>%
    dplyr::select(gutenberg_id, text)

extra_book <- extra_book %>%
    dplyr::summarise(text = paste0(text, collapse = " ")) %>%
    dplyr::mutate(gutenberg_id = as.integer(16348)) %>%
    dplyr::select(gutenberg_id, text)

ia_book_texts_df <- ia_book_texts_df %>%
    dplyr::bind_rows(redeemed, extra_book)

# remove duplicates
ia_book_texts_df <- ia_book_texts_df[-c(9,10),]

ia_book_texts_df <- ia_book_texts_df %>%
  filter(gutenberg_id != 68179) # iowa banned books list
```

1.5.2 Title Additions

I needed to correct the title and author names for further processing. This is an ugly way to do it, but it was easier than trying to loop through the previous code to pull the data. On a larger data set, the time spent generating a loop or function to do this would be worth it, but it was not a beneficial use of time in this case. I added in the book titles and authors and rearragned the columns.

```
fl_book_titles <- c("Wuthering Heights", "Leonardo Da Vinci", "The Pirate", "The I
                    "The Road", "Chain Reaction", "The Heir")
fl_book_authors <- c("Emily Brontë", "Maurice W. Brockwell", "Captain Frederick Ma
                    "William Shakespeare", "Jack London", "Boyd Ellanby", "Sydney

ia_book_titles <- c("The Picture of Dorian Gray", "The Talisman", "Christine", "Th
                    "Redeemed", "Dreamland")

ia_book_authors <- c("Oscar Wilde", "Sir Walter Scott", "Elizabeth Von Arnim", "Ar
                    "G. G. Revelle", "Wallace Macfarlane", "George Sheldon Downs",
ia_book_texts_df$title <- ia_book_titles
ia_book_texts_df$author <- ia_book_authors
fl_book_texts_df$title <- fl_book_titles
fl_book_texts_df$author <- fl_book_authors

fl_book_texts_df <- fl_book_texts_df %>%
    dplyr::select(gutenberg_id, title, author, text)

ia_book_texts_df <- ia_book_texts_df %>%
    dplyr::select(gutenberg_id, title, author, text)
```

1.5.3 Google Books Metadata Collection

This was an interesting issue - I ran into troubles trying to write functions to return the metadata from google books, even trying to leverage the google books API did not work. I ended up using ChatGPT to query and return a csv of the titles, subjects, and descriptions, which I was able to read into a variable and left join to my texts data frames to have the complete data

set required to begin data preprocessing. I selected 3 texts at random from each list and validated that they were correct, by direct comparison with the google books website for each text - a measure to assure hallucination had not entered the chat. I now have my final data sets ready for preproccessing and analysis:

- `fl_book_texts_df`
- `ia_book_texts_df`

Note - After this point, I will be saving (readr::write_csv) and reading in (readr::read_csv) my data so that I am not constantly re-preprocessing data.

```
fl_book_metadata <- readr::read_csv("Project/data/fl_book_metadata.csv")
ia_book_metadata <- readr::read_csv("Project/data/ia_book_metadata.csv")

fl_book_texts_df %>% dplyr::left_join(fl_book_metadata, by = "title") -> fl_book_t
ia_book_texts_df %>% dplyr::left_join(ia_book_metadata, by = "title") -> ia_book_t
```

1.5.4 Data Collection for Books Not Banned

To build a successful classification model, I needed to obtain text of books that were not banned. I selected 20 books at random from the Project Gutenberg website to use for my model training. The texts will be cleaned and preprocessed in the same manner as the Florida and Iowa texts before being used in the model training process (section 5).

```
books_not_banned <- data.frame(
  gutenberg_id = c(13996, 56415, 145, 5200, 1342, 2554, 4300, 219, 730, 161, 55, 1
  title = c("The Divine Fire", "Making the Nine", "Middlemarch", "Metamorphosis",
  "Crime and Punishment", "Ulysses", "Heart of Darkness", "Oliver Twist", "Sense a
  "A Modest Proposal", "The Adventures of Sherlock Holmes", "Little Women", "Narra
  "The Devil is an Ass", "The Souls of Black Folk", "Simple Sabotage Field Manual"
  author = c("May Sinclair", "Albertus T. Dudley", "George Eliot", "Franz Kafka",
  "Charles Dickens", "Jane Austen", "L. Frank Baum", "Leon Tolstoy", "Jonathan Swi
  "Ben Johnson", "W. E. B. Du Bois", "United States Office of Strategic Services",
  stringsAsFactors = FALSE
```

```
)

books_not_banned$text <- sapply(books_not_banned$gutenberg_id, load_gutenberg_text

# adding metadata from google books
books_not_banned_metadata <- readr::read_csv("Project/data/books_not_banned_metada

books_not_banned_metadata <- books_not_banned_metadata %>%
  dplyr::rename(
    title = Title,
    subject = Subject,
    description = Description
  )

books_not_banned_df <- books_not_banned %>%
  dplyr::left_join(., books_not_banned_metadata, by = "title")
```

1.5.5 Test Data

After selecting the training data, we will select a subset of books for testing our model.

```
labeled_test_data <- data.frame(
  gutenberg_id = c(2759, 67138, 20, 40539, 23005, 42914, 48258, 49745, 4497, 25060
  title = c("The Man in the Iron Mask", "The Sun Also Rises", "Paradise Lost", "A
  author = c("Alexandre Dumas", "Ernest Hemingway", "John Milton", "Guy Thorne", "
  "Elijah Kellogg", "George Meredith", "Louis Becke"),
  label = c("banned", "banned", "banned", "not banned", "not banned", "not banned"
  "not banned")
)

labeled_test_data$text <- sapply(labeled_test_data$gutenberg_id, load_gutenberg_te
```

```
test_data_metadata <- readr::read_csv(here::here("Project/data/test_data_metadata.

labeled_test_data <- labeled_test_data %>%
```

```
  dplyr::left_join(., test_data_metadata, by = "title")

labeled_test_data <- labeled_test_data %>%
  dplyr::select(gutenberg_id, title, author, text, subject, description, label)
```

1.6 Data Preprocessing

During the data preprocessing phase I will take all the text data and pass it through several functions to:

- Assure UTF-8 enconding of text data
- Make all letters lowercase
- Remove Gutenberg boilerplate text
- Normalize quotation marks for accuracy
- Expand contractions
- Remove punctuation
- Remove special characters
- Remove extra white space and formatting
- Remove stopwords
- Remove numbers
- Tokenize sentences

1.6.1 Custom Data Pre-processing Functions

Normally, I would structure this as a package and these functions would be in a separate directory where you could load them, but since I am submitting this as a stand alone QMD, I am including them in the body.

```
clean_text <- function(text_input) {
  text_vector <- as.character(text_input)
  text_vector <- stringr::str_remove(text_vector, "^\\*\\*\\* START OF THE PROJECT
  text_vector <- stringr::str_replace_all(text_vector, "\\r\\n", " ")
  text_vector <- tolower(text_vector)
  text_vector <- stringr::str_replace_all(text_vector, "[“”]", "\"")
  text_vector <- stringr::str_replace_all(text_vector, "[‘’]", "'")
```

```
  text_vector <- textclean::replace_contraction(text_vector)
  text_vector <- stringr::str_remove_all(text_vector, "[[:punct:]]")
  text_vector <- stringr::str_remove_all(text_vector, "\\d+")
  text_vector <- stringr::str_replace_all(text_vector, "[^a-zA-Z\\s]", "")
  text_vector <- stringr::str_squish(text_vector)
  text_vector <- tm::removeWords(text_vector, tm::stopwords("en"))
  text_vector <- stringr::str_remove_all(text_vector, "\\b[a-zA-Z]\\b")
  text_vector <- tokenizers::tokenize_sentences(text_vector)
  text_vector <- sapply(text_vector, function(x) paste(x, collapse = " "))

  return(text_vector)
}
```

```
# create a lightweight function to clean google books descriptions
clean_descriptions <- function(text_vector) {
  text_vector <- iconv(text_vector, to = "UTF-8")
  text_vector <- tolower(text_vector)
  text_vector <- stringr::str_remove_all(text_vector, "[[:punct:]]")
  text_vector <- stringr::str_remove_all(text_vector, "\\d+")
  text_vector <- stringr::str_replace_all(text_vector, "[^a-zA-Z\\s]", "")
  text_vector <- stringr::str_squish(text_vector)
  text_vector <- tm::removeWords(text_vector, tm::stopwords("en"))
  text_vector <- stringr::str_remove_all(text_vector, "\\b[a-zA-Z]\\b")
  text_vector <- tokenizers::tokenize_sentences(text_vector)
  text_vector <- sapply(text_vector, function(x) paste(x, collapse = " "))

  return(text_vector)
}
```

```
clean_text_headers <- function(text) {
  text_vector <- unlist(strsplit(text, "\r\n|\n"))
  complete_text <- paste(text_vector, collapse = " ")
  header_start_pattern <- "(?i)\\*{3,}\\s*start of the project gutenberg ebook\\s*
  header_match <- stringr::str_locate(complete_text, header_start_pattern)
  if (!is.na(header_match[1])) {
    header_end_pos <- header_match[2]
    trimmed_text <- substring(complete_text, header_end_pos + 1)
```

```
  } else {
    trimmed_text <- complete_text
  }

  return(trimmed_text)
}
```

```
# custom punctuation removal
remove_punctuation_custom <- function(text) {
  # remove punctuation including single and double quotes
  return(gsub("[[:punct:]\"']", "", text))
}
```

1.6.2 Function Application

First Preprocessing Run

```
# apply preprocessing function to each data set
fl_book_texts_df$cleaned_text <- sapply(fl_book_texts_df$text, clean_text)
ia_book_texts_df$cleaned_text <- sapply(ia_book_texts_df$text, clean_text)

fl_book_texts_df$cleaned_description <- sapply(fl_book_texts_df$description, clean
ia_book_texts_df$cleaned_description <- sapply(ia_book_texts_df$description, clean

fl_book_texts_df <- fl_book_texts_df %>%
    dplyr::filter(!is.na(cleaned_description)) %>%
    dplyr::filter(!is.na(cleaned_text))

ia_book_texts_df <- ia_book_texts_df %>%
    dplyr::filter(!is.na(cleaned_description)) %>%
    dplyr::filter(!is.na(cleaned_text))
```

Second Preprocessing Run

```r
# this is for working on data that has already been saved
fl_book_texts_df <- readr::read_csv("Project/data/fl_book_texts.csv")
fl_book_texts_df <- fl_book_texts_df %>%
  dplyr::select(-...1) %>%
  dplyr::mutate(gutenberg_id = as.character(gutenberg_id)) %>%
  dplyr::mutate(text = lapply(text, clean_text_headers)) %>%
  dplyr::mutate(text = lapply(text, function(t) tm::removeWords(t, enhanced_stopw
  dplyr::mutate(text = lapply(text, function(t) tm::removePunctuation(t))) %>%
  dplyr::mutate(text = lapply(text, function(t) tm::removeNumbers(t))) %>%
  dplyr::mutate(text = lapply(text, remove_punctuation_custom))

# redeemed text did not properly preprocess
redeemed_text_uncleaned <- readr::read_csv("Project/data/fl_book_texts.csv")
redeemed_text_uncleaned <- redeemed_text_uncleaned %>%
  dplyr::filter(title == "Redeemed")

redeemed_text_cleaned <- redeemed_text_uncleaned %>%
  dplyr::select(-...1) %>%
  dplyr::mutate(text = as.character(text)) %>%
  dplyr::mutate(gutenberg_id = as.character(gutenberg_id)) %>%
  dplyr::mutate(text = iconv(text, to = "UTF-8", sub = "byte")) %>%
  dplyr::mutate(text_cleaned = tm::removeWords(text, enhanced_stopwords)) %>%
  dplyr::mutate(text_cleaned = tm::removePunctuation(text_cleaned)) %>%
  dplyr::mutate(text_cleaned = tm::removeNumbers(text_cleaned)) %>%
  dplyr::mutate(text_cleaned = remove_punctuation_custom(text_cleaned))

start_phrase <- "Two lives that once part are as ships that divide, When, moment c

# remove everything before the start_phrase
redeemed_text_cleaned$text <- sub(paste0(".*(?=", start_phrase, ")"), "", redeemed

# Ensure `fl_book_text_df` is updated with the new text
fl_book_texts_df <- fl_book_texts_df %>%
  dplyr::mutate(text = ifelse(title == "Redeemed", redeemed_text_cleaned$text_clea

# flatten lists to save csv
fl_book_texts_df <- fl_book_texts_df %>%
```

```
  dplyr::mutate(text = purrr::map_chr(text, paste, collapse = " "))

ia_book_texts_df <- readr::read_csv("Project/data/ia_book_texts.csv")
ia_book_texts_df <- ia_book_texts_df %>%
  dplyr::select(-...1) %>%
  dplyr::mutate(gutenberg_id = as.character(gutenberg_id)) %>%
  dplyr::mutate(text = lapply(text, clean_text_headers)) %>%
  dplyr::mutate(text = lapply(text, function(t) tm::removeWords(t, enhanced_stopw
  dplyr::mutate(text = lapply(text, function(t) tm::removePunctuation(t))) %>%
  dplyr::mutate(text = lapply(text, function(t) tm::removeNumbers(t))) %>%
  dplyr::mutate(text = lapply(text, remove_punctuation_custom))

ia_book_texts_df <- ia_book_texts_df %>%
  dplyr::mutate(text = ifelse(title == "Redeemed", redeemed_text_cleaned$text_clea

# flatten lists to save csv
ia_book_texts_df <- ia_book_texts_df %>%
  dplyr::mutate(text = purrr::map_chr(text, paste, collapse = " "))
```

Google Books Description Preprocessing

```
# preprocess book descriptions
fl_book_texts_df <- fl_book_texts_df %>%
  dplyr::mutate(description = clean_descriptions(description))

ia_book_texts_df <- ia_book_texts_df %>%
  dplyr::mutate(description = clean_descriptions(description))
```

Note - This is another save point. After preprocessing, I save the text to csv's for easy access later on - after this point they will be read in

```
# load cleaned text
fl_book_texts_cleaned_df <- readr::read_csv(here::here("Project/data/fl_book_texts
fl_book_texts_cleaned_df <- fl_book_texts_cleaned_df %>%
  dplyr::mutate(text = purrr::map(text, ~ .x)) %>%
  dplyr::mutate(gutenberg_id = as.character(gutenberg_id))
```

```
ia_book_texts_cleaned_df <- readr::read_csv(here::here("Project/data/ia_book_texts
ia_book_texts_cleaned_df <- ia_book_texts_cleaned_df %>%
  dplyr::mutate(text = purrr::map(text, ~ .x)) %>%
  dplyr::mutate(gutenberg_id = as.character(gutenberg_id))
```

1.6.3 Name Entity Recognition

After running my initial analysis, I realized that names were heavily skewing my data. I returned to this point in my code and completed Name Entity Recognition filtering to remove names from my data. This made the outputs much cleaner, meaningful, and productive. I added the names to my stopwords list for future use.

```
cleanNLP::cnlp_init_udpipe(model_path = here::here("Project/english-ud-2.1-201801

extract_character_names <- function(text) {
  annotations <- cleanNLP::cnlp_annotate(text)

  name_candidates <- annotations$token %>%
    dplyr::filter(xpos == "NNP") %>%
    dplyr::filter(relation == "nsubj") %>%
    dplyr::filter(stringr::str_detect(token, "^[A-Z]")) %>%
    dplyr::filter(stringr::str_length(token) >= 3) %>%
    dplyr::filter(!token %in% tolower(c("God", "Satan", "Angel", "Saint", "Miss",
    dplyr::pull(lemma) %>%
    tolower() %>%
    unique()

  return(name_candidates)
}
```

```
fl_name_candidates <- purrr::map(fl_book_texts_cleaned_df$text, extract_character_
ia_name_candidates <- purrr::map(ia_book_texts_cleaned_df$text, extract_character_

name_stopwords <- tolower(unique(unlist(c(fl_name_candidates, ia_name_candidates)
```

Note - This is another save point. The ***name_stopwords*** *list was expansive so I wrote it to a csv. Here it is read back in for convenience.*

```
name_stopwords <- readr::read_csv(here::here("Project/data/name_stopwords.csv"))
name_stopwords <- name_stopwords$name_stopwords

# removing additional stopwords from texts
enhanced_stopwords <- c(
  stopwords::stopwords("en", source = "stopwords-iso"),
  quanteda::stopwords("en"),
  name_stopwords,
  tidytext::stop_words$word,
  "the", "and", "but", "gutenberg"
) %>% unique()
```

1.6.4 Non Banned Books

Note - This is another point where previously saved data is read in. The ***books_not_banned_df*** *is the data frame of books not banned for training. Here it is read back in for convenience. This cell is still tagged as eval=FALSE because the final completed and cleaned data sets will be read in later.*

```
books_not_banned_df <- readr::read_csv(here::here("Project/data/books_not_banned_c

#  clean text
books_not_banned_df <- books_not_banned_df %>%
  dplyr::mutate(gutenberg_id = as.character(gutenberg_id)) %>%
  dplyr::mutate(text = lapply(text, clean_text)) %>%
  dplyr::mutate(text = lapply(text, clean_text_headers)) %>%
  dplyr::mutate(text = lapply(text, remove_punctuation_custom)) %>%
  dplyr::mutate(text = lapply(text, function(t) tm::removeWords(t, enhanced_stopw
  dplyr::mutate(text = lapply(text, function(t) tm::removePunctuation(t))) %>%
  dplyr::mutate(text = lapply(text, function(t) tm::removeNumbers(t)))

# clean descriptions
books_not_banned_df <- books_not_banned_df %>%
  dplyr::mutate(description = lapply(description, clean_descriptions)) %>%
```

```
  dplyr::mutate(description = lapply(description, remove_punctuation_custom)) %>%
  dplyr::mutate(description = lapply(description, function(t) tm::removeWords(t, 
  dplyr::mutate(description = lapply(description, function(t) tm::removePunctuati
  dplyr::mutate(description = lapply(description, function(t) tm::removeNumbers(t

# NER
new_name_candidates <- purrr::map(books_not_banned_df$text, extract_character_name

name_stopwords <- tolower(unique(unlist(c(fl_name_candidates, ia_name_candidates,
#readr::write_csv(data.frame(name_stopwords = name_stopwords), "Project/data/name_
name_stopwords <- readr::read_csv("Project/data/name_stopwords.csv")$name_stopwor

# removing additional stopwords from texts
enhanced_stopwords <- c(
  stopwords::stopwords("en", source = "stopwords-iso"),
  quanteda::stopwords("en"),
  name_stopwords,
  tidytext::stop_words$word,
  "the", "and", "but", "gutenberg"
) %>% unique()

# final cleaning to remove NER entities

books_not_banned_text_cleaned_df <- books_not_banned_df %>%
  tidytext::unnest_tokens(word, text) %>%
  dplyr::filter(!word %in% enhanced_stopwords) %>%
  dplyr::group_by(gutenberg_id) %>%
  dplyr::summarise(text = paste(word, collapse = " "))

books_not_banned_text_cleaned_df <- books_not_banned_text_cleaned_df %>%
  mutate(
    text = sapply(text, toString),
    description = sapply(description, toString)
  )
```

Note - this is the final point where I will read in data directly to variables. From here on out, code can be run without having to change any eval=FALSE

flags.

```
# load cleaned text
fl_book_texts_cleaned_df <- readr::read_csv(here::here("Project/data/fl_book_texts
fl_book_texts_cleaned_df <- fl_book_texts_cleaned_df %>%
  dplyr::mutate(text = purrr::map(text, ~ .x)) %>%
  dplyr::mutate(gutenberg_id = as.character(gutenberg_id))

ia_book_texts_cleaned_df <- readr::read_csv(here::here("Project/data/ia_book_texts
ia_book_texts_cleaned_df <- ia_book_texts_cleaned_df %>%
  dplyr::mutate(text = purrr::map(text, ~ .x)) %>%
  dplyr::mutate(gutenberg_id = as.character(gutenberg_id))

books_not_banned_text_cleaned_df <- readr::read_csv(here::here("Project/data/books
books_not_banned_text_cleaned_df <- books_not_banned_text_cleaned_df %>%
  dplyr::mutate(text = purrr::map(text, ~ .x)) %>%
  dplyr::mutate(gutenberg_id = as.character(gutenberg_id))
```

This is the point where I have completed my data preprocessing and I join all data together to create my final training data set `complete_book_texts_cleaned_df`. After this point, all

```
# Label and combine datasets
fl_books_labeled <- fl_book_texts_cleaned_df %>%
  dplyr::mutate(region = "Florida", banned = 1)

ia_books_labeled <- ia_book_texts_cleaned_df %>%
  dplyr::mutate(region = "Iowa", banned = 1)

nonbanned_books_labeled <- books_not_banned_text_cleaned_df %>%
  dplyr::mutate(region = "Other", banned = 0)

# Merge into one master dataset
complete_book_texts_cleaned_df <- dplyr::bind_rows(fl_books_labeled, ia_books_labe
```

1.7 Data Analysis and Visualization

```
complete_book_texts_cleaned_df <- readr::read_csv(here::here("Project/data/complet
```

1.7.1 Term Frequency and Term Frequency Inverse Document Frequency Analysis

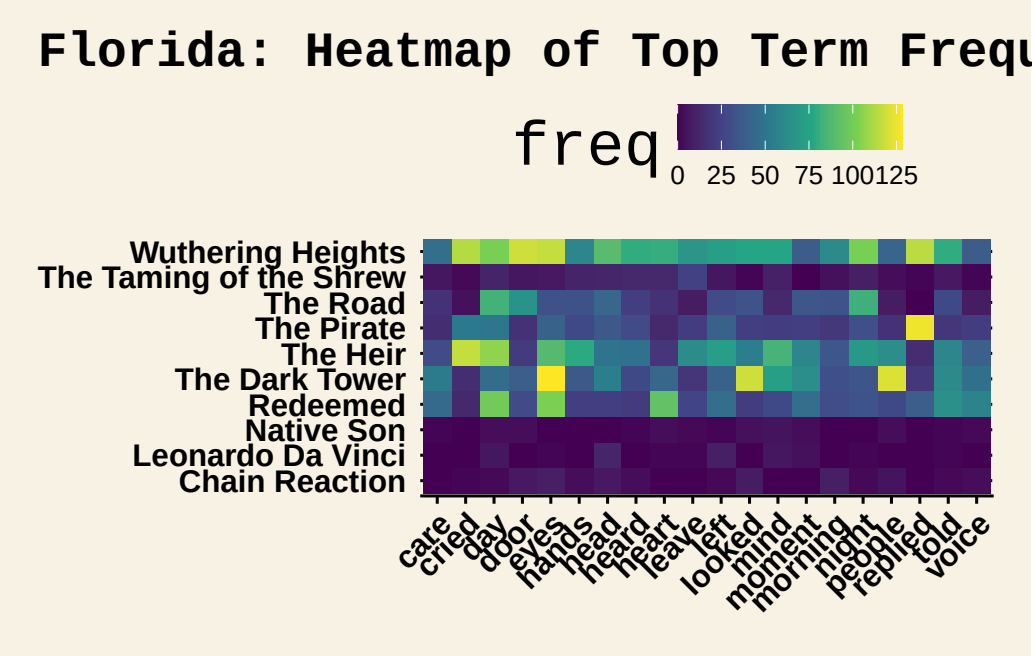
1.7.1.1 Florida

```
# calculate fl tf
fl_tf <- fl_book_texts_cleaned_df %>%
  tidytext::unnest_tokens(word, text) %>%
  dplyr::filter(!word %in% enhanced_stopwords) %>%
  dplyr::count(gutenberg_id, word, sort = TRUE)

# calculate fl tfidf
fl_tfidf <- fl_tf %>%
  tidytext::bind_tf_idf(term = word, document = gutenberg_id, n = n) %>%
  dplyr::left_join(
    fl_book_texts_cleaned_df %>% dplyr::select(gutenberg_id, title),
    by = "gutenberg_id"
  )

# calculate top fl tfidf
top_fl_tfidf <- fl_tfidf %>%
  dplyr::group_by(gutenberg_id) %>%
  dplyr::arrange(dplyr::desc(tf_idf)) %>%
  dplyr::slice_head(n = 20)
```

Florida Heatmap of Term Frequencies

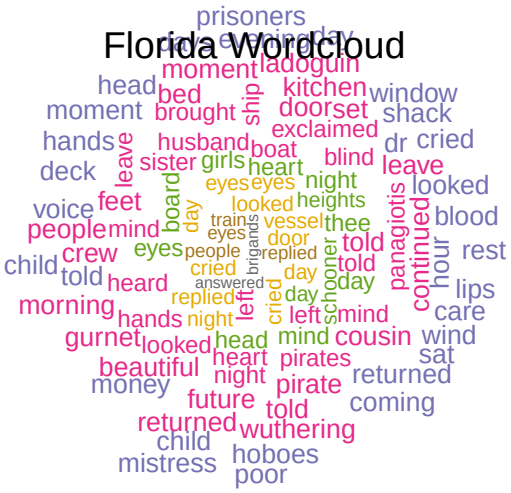
```
# calculate top fl tf
top_fl_tf <- fl_tf %>%
  dplyr::group_by(word) %>%
  dplyr::summarise(total = sum(n)) %>%
```

```
  dplyr::slice_max(order_by = total, n = 20)

# heatmap
fl_tf_heatmap <- fl_tfidf %>%
  dplyr::filter(word %in% top_fl_tf$word) %>%
  dplyr::select(gutenberg_id, title, word, n) %>%
  tidyr::pivot_wider(names_from = word, values_from = n, values_fill = 0) %>%
  tidyr::pivot_longer(cols = -c(gutenberg_id, title), names_to = "word", values_to

# Plot
ggplot2::ggplot(fl_tf_heatmap, ggplot2::aes(x = word, y = title, fill = freq)) +
  ggthemes::theme_wsj() +
  ggplot2::geom_tile() +
  ggplot2::scale_fill_viridis_c() +
  ggplot2::labs(title = "Florida: Heatmap of Top Term Frequencies",
                x = "Term", y = "Book Title") +
  ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1)) +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0, , size = rel(0.8))
  ) +
  ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```



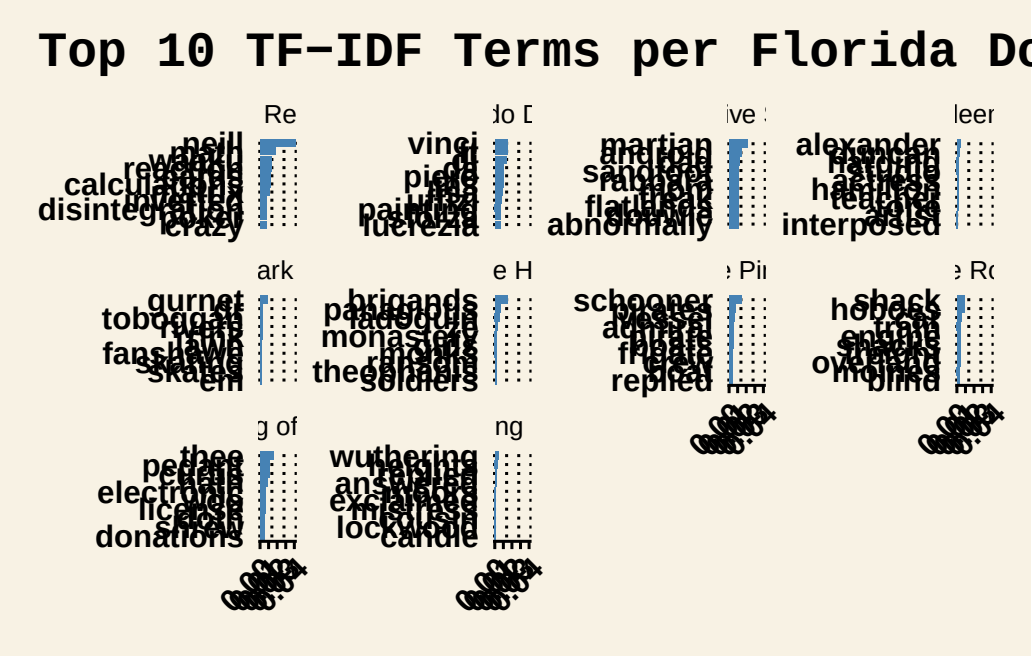
Florida Wordcloud of Term Frequencies

```
# build wordcloud
wordcloud::wordcloud(
  words = fl_tfidf$word,
  freq = fl_tfidf$n,
  min.freq = 5,
  max.words = 100,
  scale = c(0.5, 1),
  colors = RColorBrewer::brewer.pal(8, "Dark2"),
  random.order = FALSE,
)
graphics::mtext("Florida Wordcloud", side = 3, cex = 1.2)
```



Florida Top 10 TF-IDF Terms

```
# calculate top tf-idf
fl_top_tfidf <- fl_tfidf %>%
  dplyr::group_by(gutenberg_id) %>%
  dplyr::slice_max(order_by = tf_idf, n = 10) %>%
  dplyr::ungroup()

# plot tf-idf
ggplot2::ggplot(fl_top_tfidf, ggplot2::aes(x = reorder(word, tf_idf), y = tf_idf))
  ggthemes::theme_wsj() +
  ggplot2::geom_col(fill = "steelblue") +
  ggplot2::facet_wrap(~ title, scales = "free_y") +
  ggplot2::coord_flip() +
  ggplot2::labs(title = "Top 10 TF-IDF Terms per Florida Document",
                x = "Term", y = "TF-IDF Score") +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0, , size = rel(0.8))
  ) +
  ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```



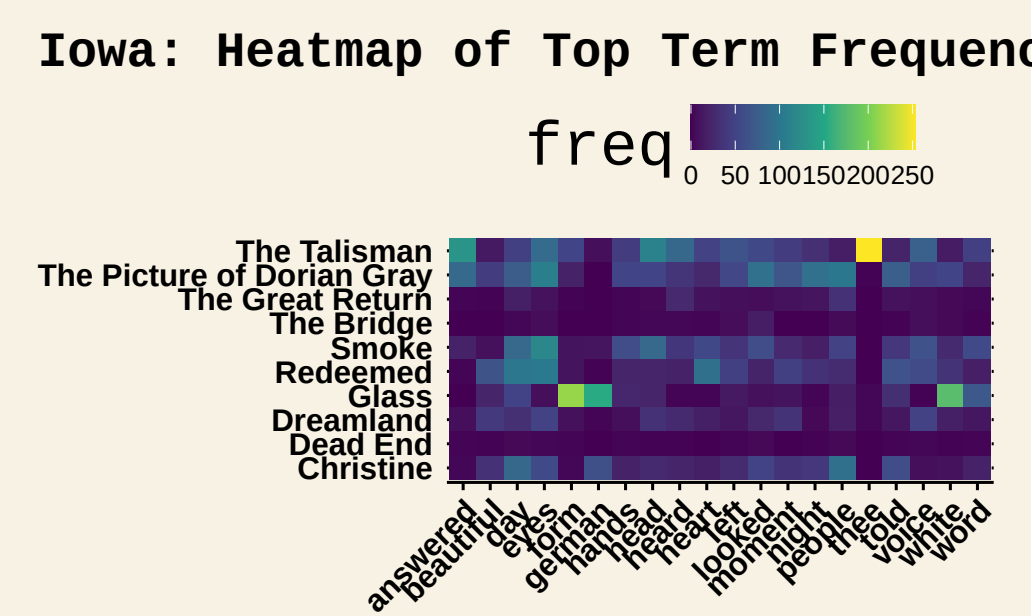
1.7.1.2 Iowa

```
# calculate ia tf
ia_tf <- ia_book_texts_cleaned_df %>%
  tidytext::unnest_tokens(word, text) %>%
  dplyr::filter(!word %in% enhanced_stopwords) %>%
  dplyr::count(gutenberg_id, word, sort = TRUE)

# calculate ia tfidf
ia_tfidf <- ia_tf %>%
  tidytext::bind_tf_idf(term = word, document = gutenberg_id, n = n) %>%
  dplyr::left_join(
    ia_book_texts_cleaned_df %>% dplyr::select(gutenberg_id, title),
    by = "gutenberg_id"
  )

# calculate top ia tfidf
top_ia_tfidf <- ia_tfidf %>%
  dplyr::group_by(gutenberg_id) %>%
```

```
  dplyr::arrange(dplyr::desc(tf_idf)) %>%
  dplyr::slice_head(n = 20)
```

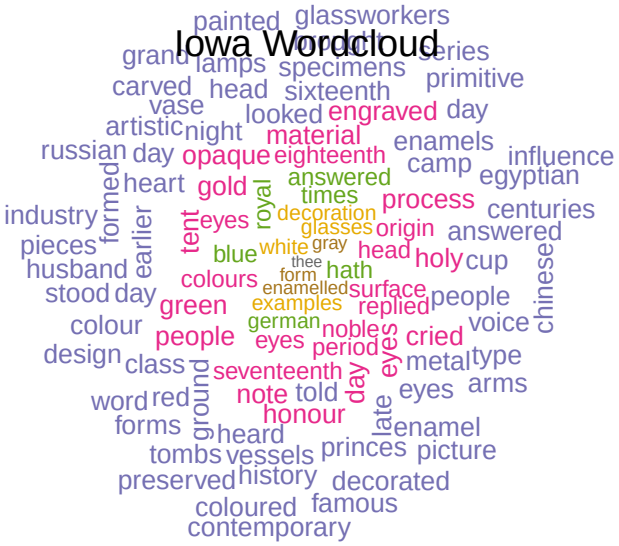
Iowa Heatmap of Term Frequencies

```
# calculate top ia tf
top_ia_tf <- ia_tf %>%
  dplyr::group_by(word) %>%
  dplyr::summarise(total = sum(n)) %>%
  dplyr::slice_max(order_by = total, n = 20)

# heatmap
ia_tf_heatmap <- ia_tfidf %>%
  dplyr::filter(word %in% top_ia_tf$word) %>%
  dplyr::select(gutenberg_id, title, word, n) %>%
  tidyr::pivot_wider(names_from = word, values_from = n, values_fill = 0) %>%
  tidyr::pivot_longer(cols = -c(gutenberg_id, title), names_to = "word", values_to

# plot
ggplot2::ggplot(ia_tf_heatmap, ggplot2::aes(x = word, y = title, fill = freq)) +
  ggthemes::theme_wsj() +
  ggplot2::geom_tile() +
  ggplot2::scale_fill_viridis_c() +
  ggplot2::labs(title = "Iowa: Heatmap of Top Term Frequencies",
                x = "Term", y = "Book Title") +
  ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))+
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0, , size = rel(0.8))
  )
```



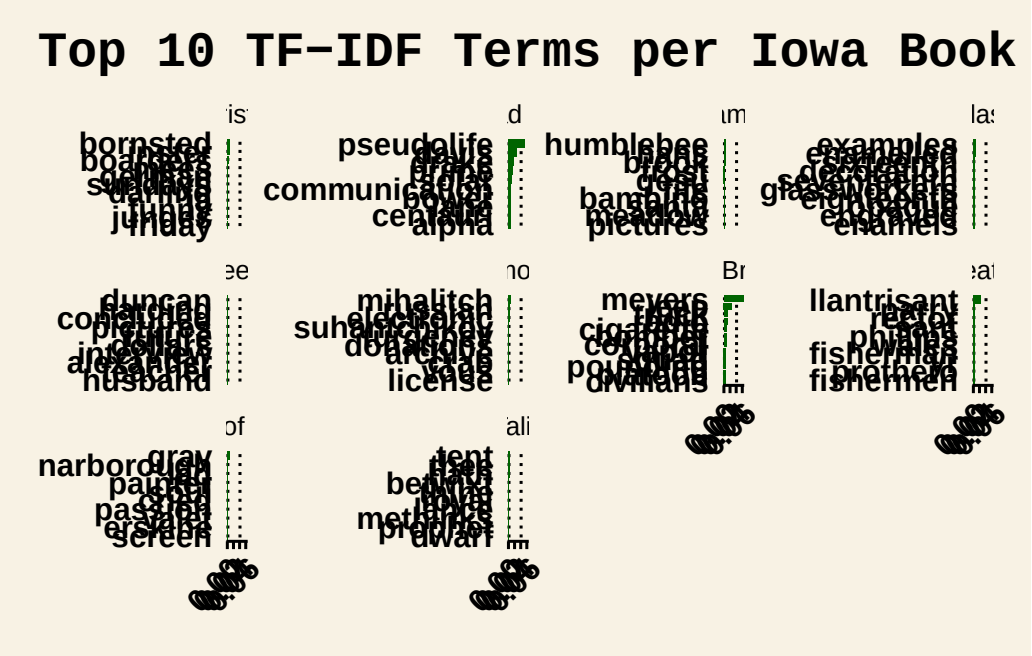
Iowa of Term Frequencies

```
wordcloud::wordcloud(
  words = ia_tfidf$word,
  freq = ia_tfidf$n,
  min.freq = 5,
  max.words = 100,
  scale = c(0.5, 1),
  colors = RColorBrewer::brewer.pal(8, "Dark2"),
  random.order = FALSE
)
graphics::mtext("Iowa Wordcloud", side = 3, line = .5, cex = 1.2)
```



Iowa Top 10 TF-IDF Terms

```
# calculate top ia tf
top_ia_tfidf <- ia_tfidf %>%
  dplyr::group_by(gutenberg_id) %>%
  dplyr::slice_max(order_by = tf_idf, n = 10) %>%
  dplyr::ungroup()

# Plot
ggplot2::ggplot(top_ia_tfidf, ggplot2::aes(x = reorder(word, tf_idf), y = tf_idf))
  ggthemes::theme_wsj() +
  ggplot2::geom_col(fill = "darkgreen") +
  ggplot2::facet_wrap(~ title, scales = "free_y") +
  ggplot2::coord_flip() +
  ggplot2::labs(
    title = "Top 10 TF-IDF Terms per Iowa Book",
    x = "Term", y = "TF-IDF Score"
  ) +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0, , size = rel(0.8))
  ) +
  ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```



After reviewing the TF heatmaps and wordclouds, we can already begin to see a divergence in patterns between Florida and Iowa. The texts of the Florida banned books list appear to be more emotionally focused, see high frequency words like heart, cried, and care - while the texts of the Iowa banned books list appears to be more historically / technically focused, seeing frequent words like platoon, engraved, and enamelled.

The TF-IDF charts confirm this divergence. Florida's High Signal words include more identity, family, and conflict markers where Iowa still has a concentration of historical and technical markers.

1.7.2 Phrase Frequency Analysis

Next point of analysis is phrase frequency analysis - specifically bigrams.

1.7.2.1 Florida

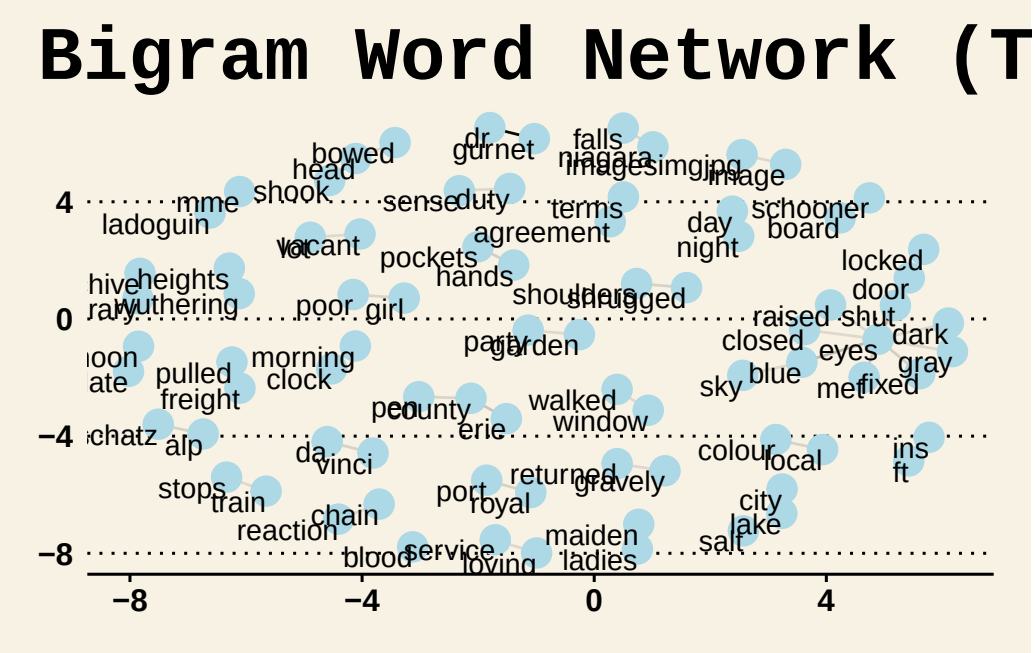
```
# tokenize fl bigrams
fl_bigrams <- fl_book_texts_cleaned_df %>%
```

```
  tidytext::unnest_tokens(bigram, text, token = "ngrams", n = 2)

# split and filter
fl_bigrams_filtered <- fl_bigrams %>%
  tidyr::separate(bigram, into = c("word1", "word2"), sep = " ") %>%
  dplyr::filter(!word1 %in% enhanced_stopwords, !word2 %in% enhanced_stopwords) %>
  tidyr::unite(bigram, word1, word2, sep = " ")

# calculate fl tf
fl_bigram_tf <- fl_bigrams_filtered %>%
  dplyr::count(gutenberg_id, bigram, sort = TRUE)

# calculate fl tfidf
fl_bigram_tfidf <- fl_bigram_tf %>%
  tidytext::bind_tf_idf(term = bigram, document = gutenberg_id, n = n) %>%
  dplyr::left_join(
    fl_book_texts_cleaned_df %>% dplyr::select(gutenberg_id, title),
    by = "gutenberg_id"
  )

# top bigrams per doc
top_fl_bigram_tfidf <- fl_bigram_tfidf %>%
  dplyr::group_by(gutenberg_id) %>%
  dplyr::slice_max(order_by = tf_idf, n = 10) %>%
  dplyr::ungroup()
```

Florida Bigram Network

```
# split bigrams into related adjacent columns
fl_bigram_pairs <- fl_bigrams_filtered %>%
  tidyr::separate(bigram, into = c("word1", "word2"), sep = " ") %>%
  dplyr::count(word1, word2, sort = TRUE) %>%
  dplyr::filter(n >= 7)

fl_bigram_graph <- igraph::graph_from_data_frame(fl_bigram_pairs)

# plot bigram network
```

```
ggraph::ggraph(fl_bigram_graph, layout = "fr") +
  ggthemes::theme_wsj() +
  ggraph::geom_edge_link(ggplot2::aes(edge_alpha = n), show.legend = FALSE) +
  ggraph::geom_node_point(color = "lightblue", size = 5) +
  ggraph::geom_node_text(ggplot2::aes(label = name), vjust = 1, hjust = 1) +
  ggplot2::labs(title = "Bigram Word Network (TF   7)") +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0)
  )
```



1.7.2.2 Iowa

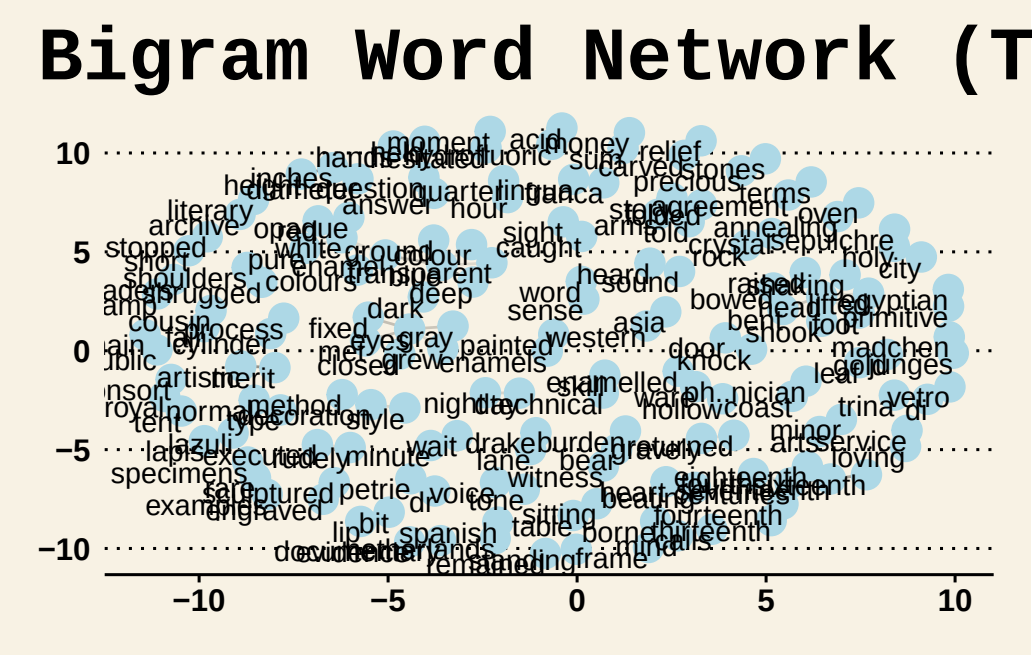
```
# tokenize ia bigrams
ia_bigrams <- ia_book_texts_cleaned_df %>%
  tidytext::unnest_tokens(bigram, text, token = "ngrams", n = 2)

# split and filter
```

```r
ia_bigrams_filtered <- ia_bigrams %>%
  tidyr::separate(bigram, into = c("word1", "word2"), sep = " ") %>%
  dplyr::filter(!word1 %in% enhanced_stopwords, !word2 %in% enhanced_stopwords) %>
  tidyr::unite(bigram, word1, word2, sep = " ")

# calculate ia tf
ia_bigram_tf <- ia_bigrams_filtered %>%
  dplyr::count(gutenberg_id, bigram, sort = TRUE)

# calculate ia tfidf
ia_bigram_tfidf <- ia_bigram_tf %>%
  tidytext::bind_tf_idf(term = bigram, document = gutenberg_id, n = n) %>%
  dplyr::left_join(
    fl_book_texts_cleaned_df %>% dplyr::select(gutenberg_id, title),
    by = "gutenberg_id"
  )

# top bigrams per doc
top_ia_bigram_tfidf <- ia_bigram_tfidf %>%
  dplyr::group_by(gutenberg_id) %>%
  dplyr::slice_max(order_by = tf_idf, n = 10) %>%
  dplyr::ungroup()
```

Iowa Bigram Network

```r
# split bigrams into related adjacent columns
ia_bigram_pairs <- ia_bigrams_filtered %>%
  tidyr::separate(bigram, into = c("word1", "word2"), sep = " ") %>%
  dplyr::count(word1, word2, sort = TRUE) %>%
  dplyr::filter(n >= 7)

ia_bigram_graph <- igraph::graph_from_data_frame(ia_bigram_pairs)

# plot bigram network
ggraph::ggraph(ia_bigram_graph, layout = "fr") +
  ggthemes::theme_wsj() +
  ggraph::geom_edge_link(ggplot2::aes(edge_alpha = n), show.legend = FALSE) +
```

```
  ggraph::geom_node_point(color = "lightblue", size = 5) +
  ggraph::geom_node_text(ggplot2::aes(label = name), vjust = 1, hjust = 1) +
  ggplot2::labs(title = "Bigram Word Network (TF   7)") +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0)
  )
```



After reviewing the Bigram Co-occurence Networks, we continue to see this lexical divergence between Florida and Iowa. Florida continues to exhibit phrasing pairs consistent with emotional and interpersonal issues - shrugged + shoulders, eyes + raised, eyes + closed, loving + service. Iowa shows further technical and historical tones - rare + specimens, rare + examples, Phoenician + Coast, Spanish + Netherlands.

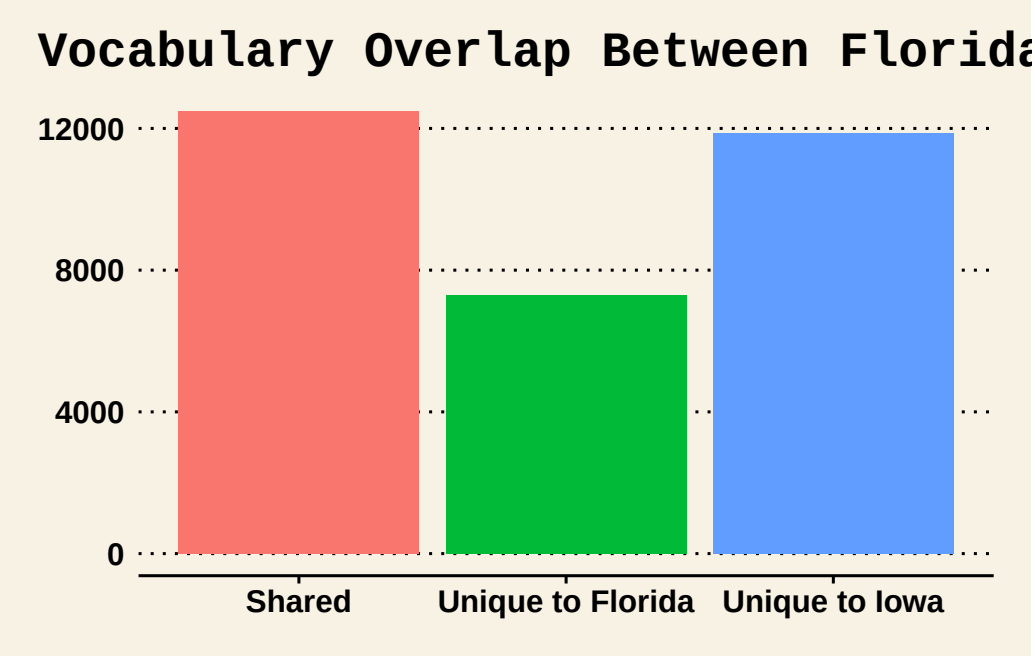
1.7.3 Subgroup Comparisons

Now the data is brought back together for subgroup comparisons. This will also introduce thematic analysis using the google books themes and

descriptions.

Common Terms Bar Plot

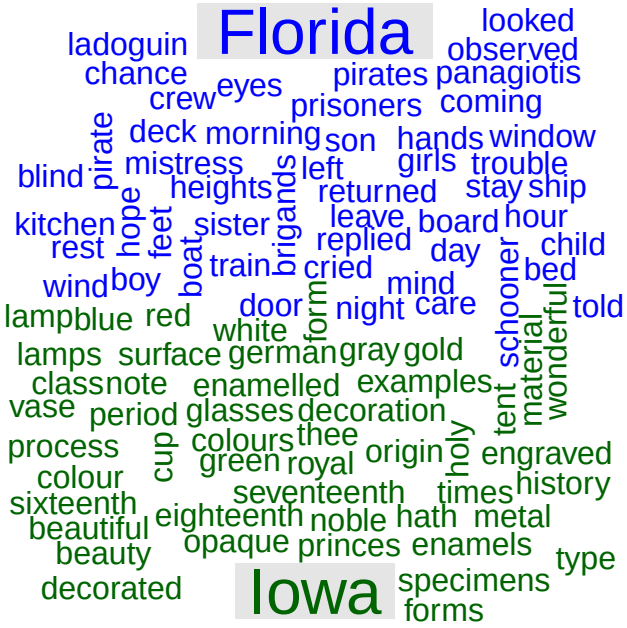
```
# create a plot of shared versus unique terms
shared_terms <- generics::intersect(fl_tfidf$word, ia_tfidf$word)
fl_unique <- generics::setdiff(fl_tfidf$word, ia_tfidf$word)
ia_unique <- generics::setdiff(ia_tfidf$word, fl_tfidf$word)

summary_df <- tibble::tibble(
  Category = c("Shared", "Unique to Florida", "Unique to Iowa"),
  Count = c(length(shared_terms), length(fl_unique), length(ia_unique))
)

ggplot2::ggplot(summary_df, ggplot2::aes(x = Category, y = Count, fill = Category)
  ggthemes::theme_wsj() +
  ggplot2::geom_col(show.legend = FALSE) +
  ggplot2::labs(title = "Vocabulary Overlap Between Florida and Iowa Books") +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0, , size = rel(0.8))
  )
```



Comparison Cloud

```
# use region label and count tf
fl_tf_region <- fl_tfidf %>%
  dplyr::select(word, n) %>%
  dplyr::mutate(region = "Florida")

ia_tf_region <- ia_tfidf %>%
  dplyr::select(word, n) %>%
  dplyr::mutate(region = "Iowa")

# combine and group
fl_ia_tf <- dplyr::bind_rows(fl_tf_region, ia_tf_region) %>%
  dplyr::group_by(region, word) %>%
  dplyr::summarise(freq = sum(n), .groups = "drop") %>%
  tidyr::pivot_wider(names_from = region, values_from = freq, values_fill = 0)

# set rownames
fl_ia_matrix <- fl_ia_tf %>%
  tibble::column_to_rownames("word") %>%
```

```
  as.matrix()

graphics::par(mar = c(1, 2, 2, 1))
wordcloud::comparison.cloud(
  fl_ia_matrix,
  colors = c("blue", "darkgreen"),
  title.size = 2,
  max.words = 100,
  scale = c(1,1),
  random.order = FALSE,
  title.colors = c("blue", "darkgreen")
)
```



The bar chart shows there are a large number of shared words between the two states banned books texts ~40% of the words are shared - however the shared words are not necessarily significant words or we would see more overlap in the TF and TF-IDF analysis. The comparison cloud confirms there is marginal overlap in important words between the two sides. Florida still squarely in the emotional, interpersonal, identity corner while Iowa resides in the technical and historical corner.

```
# combine subject fields from both datasets
all_subjects <- c(fl_book_texts_cleaned_df$subject, ia_book_texts_cleaned_df$subje

# tokenize and flatten
subject_terms <- all_subjects %>%
  stringr::str_split(",|;|\\|") %>%
  unlist() %>%
  stringr::str_trim() %>%
  tolower() %>%
  unique()

# review subjects
print(subject_terms)
```

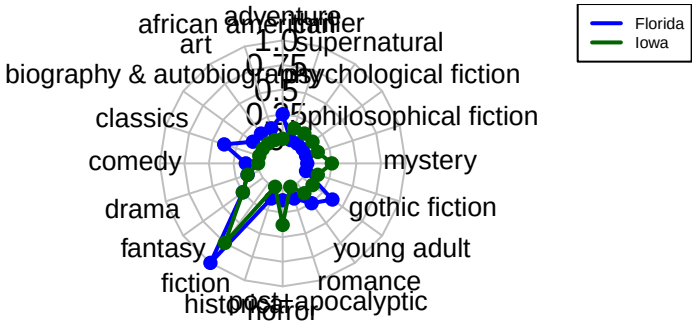
```
 [1] "fiction"                    "classics"
 [3] "biography & autobiography" "art"
 [5] "adventure"                  "historical"
 [7] "fantasy"                    "horror"
 [9] "african american"           "young adult"
[11] "drama"                      "comedy"
[13] "post-apocalyptic"           "romance"
[15] "gothic fiction"             "philosophical fiction"
[17] "thriller"                   "supernatural"
[19] "mystery"                    "psychological fiction"
```

Thematic Emphasis Radar Plot (Google Books Themes)

```
# create subjects list
fl_subjects <- fl_book_texts_cleaned_df %>%
  dplyr::select(-c(text, description, gutenberg_id)) %>%
  dplyr::mutate(region = "Florida")

ia_subjects <- ia_book_texts_cleaned_df %>%
  dplyr::select(-c(text, description, gutenberg_id)) %>%
  dplyr::mutate(region = "Iowa")
```

```r
combined_subjects <- dplyr::bind_rows(fl_subjects, ia_subjects)

# clean subjects
combined_subjects_long <- combined_subjects %>%
  dplyr::select(region, subject) %>%
  tidyr::separate_rows(subject, sep = ",|;|\\|") %>%
  dplyr::mutate(subject = stringr::str_trim(subject)) %>%
  dplyr::mutate(subject = stringr::str_to_lower(subject)) %>%
  dplyr::filter(!is.na(subject) & subject != "")

theme_summary <- combined_subjects_long %>%
  dplyr::group_by(region, subject) %>%
  dplyr::summarise(count = dplyr::n(), .groups = "drop")

theme_props <- theme_summary %>%
  tidyr::pivot_wider(names_from = subject, values_from = count, values_fill = 0) %
  tibble::column_to_rownames("region")

max_value <- max(theme_props, na.rm = TRUE)

theme_radar <- rbind(
  rep(max_value, ncol(theme_props)),
  rep(0, ncol(theme_props)),
  theme_props
)

fmsb::radarchart(
  theme_radar,
  axistype = 1,
  pcol = c("blue", "darkgreen"),
  plwd = 2,
  plty = 1,
  cglcol = "gray",
  cglty = 1,
  axislabcol = "black",
  caxislabels = c("0", "0.25", "0.5", "0.75", "1.0"),
  vlcex = 0.8,
```

```
  title = "Thematic Emphasis by Region (from Subject Metadata)"
)
graphics::legend("topright",
                 legend = rownames(theme_props),
                 col = c("blue", "darkgreen"),
                 lty = 1,
                 lwd = 2,
                 cex = 0.5,
                 xjust = 5,
                 yjust = 5)
```



This radar chart pulls from the Google Books metadata themes (subjects). We can see here that there is a heavy skew for both states towards fiction. While this chart does not further an analytical divergence beyond what we have seen, the importance of this chart will be seen later as we analyze the stark differences between the Google Book themes and the deduced themes from the book texts.

1.7.4 Dictionary-Based Analysis

Next is dictionary based analysis. I spent an extensive amount of time researching terms and fine tuning this dictionary (all references are in my paper references section). This part was tricky because I could not include certain terms that appeared in books for benign purposes. For example, if I tried to use "race" as a term in the dictionary as a keyword indicator of race being

involved in the book as a theme, then a book about horseracing may feature prominently in the "race" category, when, in fact, it was not at all about race.

```
theme_dictionary_long <- list(

  gender = c(
    "transgender", "gender non-conforming", "genderqueer", "agender", "bigender",
    "two-spirit", "pronouns", "misgender", "deadname", "transition", "gender dysph
    "gender expression", "gender roles", "gender norms", "gender variance",
    "gender identity", "gender fluidity", "gender nonconformity", "gender inclusiv
    "gender diversity", "gender equality", "gender equity", "gender justice",
    "gender mainstreaming", "gender sensitivity", "gender responsiveness",
    "gender ideology", "biological sex", "trans rights", "women's rights",
    "gender-critical", "TERF", "gender self-identification",
    "gender recognition certificate", "single-sex spaces", "bathroom bills",
    "sports eligibility", "parental rights in education", "puberty blockers",
    "gender transition in minors", "detransition", "trans", "cis", "cisgender", "a
    "transpbobia", "transphobic", "homophobia", "homophobic"
  ),

  race = c(
    "systemic racism", "racial profiling", "racial discrimination",
    "racial inequality", "racial justice", "racial equity", "racial bias",
    "racial prejudice", "racial stereotyping", "racial microaggressions",
    "racial disparities", "racial oppression", "racial segregation",
    "racial integration", "racial reconciliation", "critical race theory",
    "white privilege", "black lives matter", "affirmative action", "reparations",
    "colorism", "intersectionality", "racial identity", "racial consciousness",
    "racial solidarity", "racial justice movement", "racial equity initiatives",
    "racial healing", "racial justice advocacy", "racial justice education",
    "DEI", "anti-racism training", "wokeism", "cancel culture", "CRT bans",
    "anti-CRT legislation", "racial sensitivity training", "equity audits",
    "racial equity policies", "race-based admissions", "race-conscious policies",
    "racial equity mandates", "racial equity impact assessments",
    "racial equity frameworks", "racial", "racist", "anti racist", "reverse racism
    "african american", "black people", "white people", "latinx", "asian american
    "ethnicity", "colonialism", "slavery"
```

```
),

sexuality = c(
  "pansexual", "demisexual", "aromantic", "intersex", "non-heteronormative",
  "sexual orientation", "sexual identity", "sexual fluidity", "sexual diversity"
  "sexual rights", "sexual liberation", "sexual health", "sexual education",
  "sexual minorities", "sexual stigma", "LGBTQIA+", "pride", "rainbow flag",
  "queer culture", "queer theory", "queer activism", "queer visibility",
  "queer representation", "queer inclusion", "queer rights", "queer liberation",
  "queer community", "queer identity", "queer expression", "queer resilience",
  "Don't Say Gay", "parental rights in education", "LGBTQ+ curriculum",
  "drag bans", "trans youth healthcare", "conversion therapy bans",
  "religious freedom vs. LGBTQ+ rights", "bathroom access debates",
  "same-sex marriage debates", "LGBTQ+ adoption rights",
  "LGBTQ+ discrimination laws", "LGBTQ+ hate crimes legislation",
  "LGBTQ+ military service policies", "LGBTQ+ workplace protections", "same sex"
  "bisesxual", "asexual", "lgbt", "queer", "gay", "lesbian", "bisexual", "transg
  "sexuality", "coming out", "gender expression"
),

violence = c(
  "mass shooting", "school shooting", "police brutality", "gang violence",
  "hate crime", "sexual assault", "intimate partner violence", "child abuse",
  "elder abuse", "human trafficking", "torture", "genocide", "lynching",
  "mob violence", "vigilante justice", "state violence", "structural violence",
  "symbolic violence", "cultural violence", "economic violence", "political viol
  "institutional violence", "systemic violence", "interpersonal violence",
  "collective violence", "direct violence", "indirect violence",
  "psychological violence", "emotional violence", "verbal violence",
  "Second Amendment rights", "gun control legislation", "red flag laws",
  "stand your ground laws", "police reform", "defund the police",
  "law and order rhetoric", "crime wave narratives", "border security measures"
  "anti-terrorism policies", "domestic terrorism designations",
  "political violence discourse", "protest-related violence",
  "riot control measures", "public safety funding", "gun violence", "gun control
  "physical abuse", "physical violence", "murder", "suicide", "violent assault"
  "domestic violence", "relationship violence", "sexual violence"
```

```
),

politics = c(
  "governance", "political system", "political ideology", "political party",
  "political campaign", "political debate", "political discourse",
  "political participation", "political engagement", "political representation",
  "political accountability", "political transparency", "political corruption",
  "political reform", "political polarization", "liberalism", "conservatism",
  "socialism", "communism", "capitalism", "nationalism", "populism",
  "progressivism", "libertarianism", "centrism", "authoritarianism",
  "totalitarianism", "democracy", "theocracy", "technocracy",
  "cancel culture", "wokeism", "culture wars", "identity politics",
  "political correctness", "deep state", "election fraud claims",
  "voter suppression", "gerrymandering", "mail-in voting debates",
  "misinformation", "disinformation", "fake news", "media bias",
  "political censorship", "political protest", "political unrest", "civil disobe
  "legislation", "legislative", "political activism", "political activist", "civ
  "authoritarian"
),

history = c(
  "Jim Crow laws", "emancipation", "civil rights movement", "suffrage movement",
  "abolitionism", "manifest destiny", "trail of tears", "internment camps",
  "apartheid", "decolonization", "historical memory", "historical revisionism",
  "historical preservation", "historical reenactment", "historical narratives",
  "founding fathers", "colonialism", "reconstruction", "slavery", "civil war",
  "holocaust", "slave trade", "confederacy", "whitewashing history",
  "curriculum censorship", "monuments debate", "CRT history bans", "segregation"
  "reconstruction", "founding fathers", "historical trauma", "revisionism"
),

religion = c(
  "religion", "christianity", "islam", "judaism", "atheism", "agnosticism",
  "buddhism", "hinduism", "spirituality", "interfaith", "evangelical",
  "religious freedom", "freedom of worship", "religious pluralism",
  "religious identity", "faith-based", "bible literacy", "church-state separatio
  "public prayer", "religious exemption", "religious indoctrination",
```

```
  "religious curriculum", "proselytizing", "religious liberty", "blasphemy laws"
),

education = c(
  "curriculum", "textbook", "critical thinking", "pedagogy", "literacy",
  "academic freedom", "standardized testing", "education equity", "school choice
  "public education", "private education", "charter schools", "homeschooling",
  "banned books", "reading lists", "civics education", "parental rights in educa
  "education reform", "education censorship", "divisive concepts", "teacher unio
  "educational standards", "learning loss", "educational indoctrination", "curri
),

identity = c(
  "cultural identity", "gender identity", "intersectionality", "individuality",
  "self expression", "social identity", "collective identity", "personal identit
  "lived experience", "marginalized identity", "oppressed identity",
  "labeling", "identity politics", "identity suppression", "identity conflict",
  "identity negotiation", "identity-based censorship"
),

anti_intellectualism = c(
  "censorship", "banned books", "book ban", "indoctrination", "misinformation",
  "propaganda", "thought control", "anti-science", "truth suppression",
  "curriculum sanitization", "knowledge control", "epistemic injustice",
  "disinformation", "anti-expert sentiment", "deplatforming", "cancel culture",
  "ideological indoctrination", "educational gag orders", "viewpoint discriminat
),

family = c(
  "nuclear family", "chosen family", "foster family", "adoptive family",
  "parental rights", "parenthood", "domestic life", "family values", "family str
  "family unit", "intergenerational family", "nontraditional family",
  "same-sex parenting", "single parent household", "broken home",
  "child-rearing", "upbringing", "home environment", "parental involvement",
  "family-friendly curriculum", "family identity", "familial support"
)
```

```
)
```

```
detect_theme <- function(text, keywords) {
  pattern <- paste0("\\b(", paste(keywords, collapse = "|"), ")\\b")
  stringr::str_detect(tolower(text), pattern)
}
```

After writing a function to detect themes, I applied it to all of my data (text and google books descriptions) to begin my thematic analysis.

```
# use dictionary to compare themes
fl_theme_flags <- fl_book_texts_cleaned_df %>%
  dplyr::mutate(
    across(
      .cols = tidyselect::everything(),
      .fns = ~ as.character(.)
    )
  ) %>%
  dplyr::mutate(
    dplyr::across(
      .cols = dplyr::everything(),
      .fns = ~ stringr::str_to_lower(.)
    )
  ) %>%
  dplyr::mutate(
    has_gender = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$ge
    has_race = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$race
    has_sexuality = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long
    has_violence = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$v
    has_politics = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$p
    has_history = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$hi
    has_religion = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$r
    has_identity = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$i
    has_anti_intellectualism = purrr::map_lgl(text, ~ detect_theme(.x, theme_dicti
    has_family = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$fam
    has_education = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long
  )
```

```r
# use dictionary to compare themes
ia_theme_flags <- ia_book_texts_cleaned_df %>%
  dplyr::mutate(
    across(
      .cols = tidyselect::everything(),
      .fns = ~ as.character(.)
    )
  ) %>%
  dplyr::mutate(
    dplyr::across(
      .cols = dplyr::everything(),
      .fns = ~ stringr::str_to_lower(.)
    )
  ) %>%
  dplyr::mutate(
    has_gender = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$ge
    has_race = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$race
    has_sexuality = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long
    has_violence = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$v
    has_politics = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$p
    has_history = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$hi
    has_religion = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$r
    has_identity = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$i
    has_anti_intellectualism = purrr::map_lgl(text, ~ detect_theme(.x, theme_dicti
    has_family = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long$fam
    has_education = purrr::map_lgl(text, ~ detect_theme(.x, theme_dictionary_long
  )
```

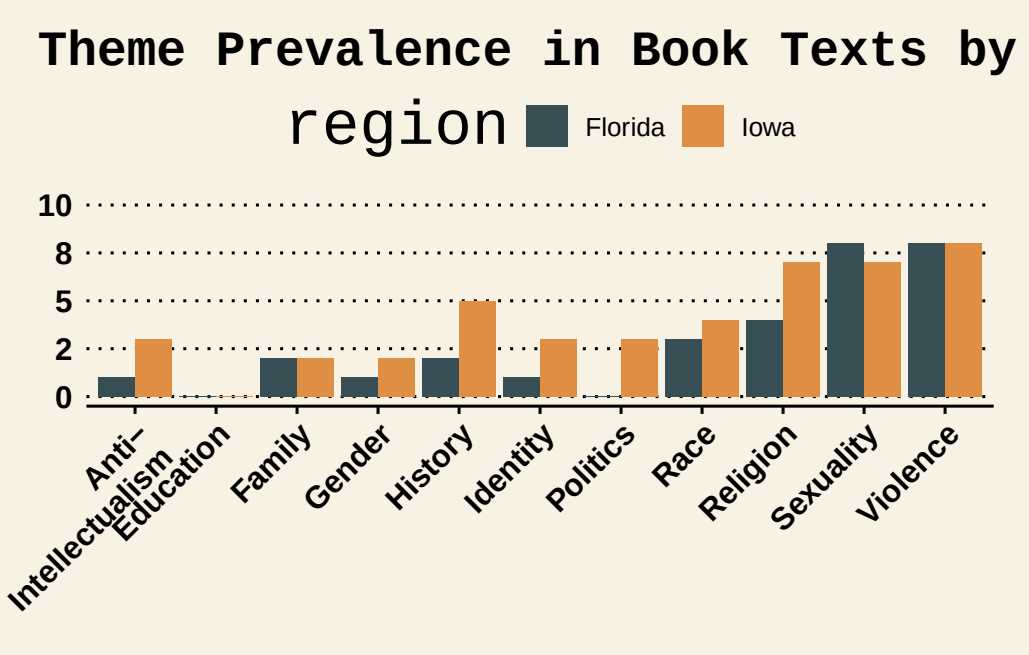
```r
# created mapping labels for visualizations
theme_label_map <- c(
  has_gender = "Gender",
  has_race = "Race",
  has_sexuality = "Sexuality",
  has_violence = "Violence",
  has_politics = "Politics",
  has_history = "History",
  has_religion = "Religion",
```

```
  has_identity = "Identity",
  has_anti_intellectualism = "Anti-\nIntellectualism",
  has_family = "Family",
  has_education = "Education"
)
```

Theme Prevalance in Book Texts

```
# standardize column type
fl_theme_flags <- fl_theme_flags %>%
  dplyr::mutate(text = as.character(text))

# standardize column type
ia_theme_flags <- ia_theme_flags %>%
  dplyr::mutate(text = as.character(text))

# combine data sets
fl_ia_theme_counts <- dplyr::bind_rows(
  fl_theme_flags %>% dplyr::mutate(region = "Florida"),
  ia_theme_flags %>% dplyr::mutate(region = "Iowa")
)

# create a summary table
theme_summary <- fl_ia_theme_counts %>%
  tidyr::pivot_longer(cols = dplyr::starts_with("has_"), names_to = "theme", value
  dplyr::group_by(region, theme) %>%
  dplyr::summarise(count = sum(present), .groups = "drop")%>%
  dplyr::mutate(theme = theme_label_map[theme])

# plot
ggplot2::ggplot(theme_summary, ggplot2::aes(x = theme, y = count, fill = region))
  ggthemes::theme_wsj() +
  ggplot2::geom_col(position = "dodge") +
  ggsci::scale_fill_jama() +
  ggplot2::scale_y_continuous(labels = scales::number_format(accuracy = 1), limits
  ggplot2::labs(title = "Theme Prevalence in Book Texts by Region",
                x = "Theme", y = "Number of Books") +
```

```
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0, , size = rel(0.8))
  ) +
  ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```




```
# comparing themes from the descriptions
fl_description_theme_flags <- fl_book_texts_cleaned_df %>%
  dplyr::mutate(
    across(dplyr::everything(), as.character)
  ) %>%
  dplyr::mutate(
    across(dplyr::everything(), stringr::str_to_lower)
  ) %>%
  dplyr::mutate(
    has_gender = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary_l
    has_race = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary_lon
    has_sexuality = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionar
    has_violence = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary
```

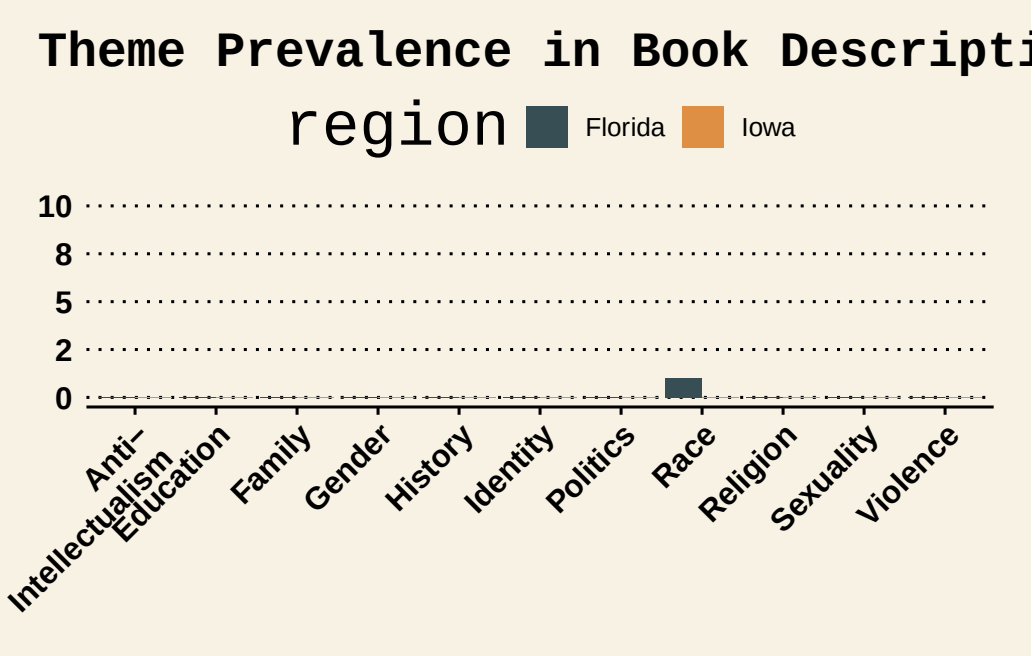
```
    has_politics = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary
    has_history = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary_
    has_religion = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary
    has_identity = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary
    has_anti_intellectualism = purrr::map_lgl(description, ~ detect_theme(.x, ther
    has_family = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary_l
    has_education = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionar
  )
```

```
# comparing themes from the descriptions
ia_description_theme_flags <- ia_book_texts_cleaned_df %>%
  dplyr::mutate(
    across(dplyr::everything(), as.character)
  ) %>%
  dplyr::mutate(
    across(dplyr::everything(), stringr::str_to_lower)
  ) %>%
  dplyr::mutate(
    has_gender = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary_l
    has_race = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary_lon
    has_sexuality = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionar
    has_violence = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary
    has_politics = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary
    has_history = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary_
    has_religion = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary
    has_identity = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary
    has_anti_intellectualism = purrr::map_lgl(description, ~ detect_theme(.x, ther
    has_family = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionary_l
    has_education = purrr::map_lgl(description, ~ detect_theme(.x, theme_dictionar
  )
```

Theme Prevalance in Book Descriptions

```
# standardize column type
fl_description_theme_flags <- fl_description_theme_flags %>%
  dplyr::mutate(region = "Florida")
```

```r
ia_description_theme_flags <- ia_description_theme_flags %>%
  dplyr::mutate(region = "Iowa")

# combine data sets
description_theme_flags <- dplyr::bind_rows(
  fl_description_theme_flags,
  ia_description_theme_flags
)

# create a summary table
description_theme_summary <- description_theme_flags %>%
  tidyr::pivot_longer(
    cols = dplyr::starts_with("has_"),
    names_to = "theme",
    values_to = "present"
  ) %>%
  dplyr::group_by(region, theme) %>%
  dplyr::summarise(count = sum(present), .groups = "drop") %>%
  dplyr::mutate(theme = theme_label_map[theme])

# plot
ggplot2::ggplot(description_theme_summary, ggplot2::aes(x = theme, y = count, fil
  ggthemes::theme_wsj() +
  ggplot2::geom_col(position = "dodge") +
  ggplot2::scale_y_continuous(labels = scales::number_format(accuracy = 1), limits
  ggsci::scale_fill_jama() +
  ggplot2::labs(
    title = "Theme Prevalence in Book Descriptions by Region",
    x = "Theme", y = "Number of Books"
  ) +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0, size = rel(0.8))
  ) +
  ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```



When we compare the themes side by side, we see a startling revalation - the Google Book themes cover almost none of the themes that come from textual analysis. Through careful dictionary based analysis, we can see a number of these books include topics that are controversial and likely cause them to be banned, however these themes do not appear in any of the Google Books themes. This leads us to two possible conclusions:

1. Publishers are actively hiding controversial themes to shield books from potential harm or backlash
2. Publishers are actively hiding controversial themes to suppress opposing or underrepresented view points on those issues.

Regardless of the motive, there is a clear incongruence between what is on the digital cover and what is between the digital book covers.

1.7.5 Classification Model

Finally we move on to our classification model. Here we train two separate models - one Ridge Regression and one XGMBoost model. Ridge was chosen

to make a more robust logistic regression model - this allows for reducing the impact of ceratin variables while not eliminating them (as is what happens if we used a LASSO model). The XGMBoost model serves as the opposing model - a nonlinear model that will hopefully capture underlying trends that would otherwise be lost to the introduced bias that results from the simplicity of a linear model.

```
# function to apply theme detection per source
create_theme_flags <- function(df, field, prefix) {
  for (theme_name in names(theme_dictionary_long)) {
    column_name <- paste0(prefix, "_has_", theme_name)
    df[[column_name]] <- purrr::map_lgl(df[[field]], ~ detect_theme(.x, theme_dict
  }
  return(df)
}
```

```
# apply new theme detection function
complete_book_texts_cleaned_df <- create_theme_flags(complete_book_texts_cleaned_d
complete_book_texts_cleaned_df <- create_theme_flags(complete_book_texts_cleaned_d
complete_book_texts_cleaned_df <- create_theme_flags(complete_book_texts_cleaned_d

theme_cols <- grep("^text_has_|^desc_has_|^pub_has_", names(complete_book_texts_cl

feature_matrix <- complete_book_texts_cleaned_df %>%
  dplyr::select(gutenberg_id, title, author, region, banned, dplyr::all_of(theme_c
```

Note - here is a save point for the feature matrix created above for convenience

```
feature_matrix <- readr::read_csv(here::here("Project/data/feature_matrix.csv"))
```

1.7.5.1 Model Functions

```
# lightweight preprocessing function
preprocess_text <- function(x) {
  x <- stringr::str_to_lower(x)
  x <- stringr::str_replace_all(x, "[^a-z\\s]", " ")
```

```
  x <- stringr::str_squish(x)
  return(x)
}
```

```
# core prediction function
predict_ban_risk <- function(title, author, text, description, published_themes) {
  # clean text
  text_clean <- preprocess_text(text)
  desc_clean <- preprocess_text(description)
  pub_clean <- preprocess_text(published_themes)
  # detect themes
  feature_vector <- tibble::tibble(title = title, author = author)

  for (theme_name in names(theme_dictionary_long)) {
    feature_vector[[paste0("text_has_", theme_name)]] <- detect_theme(text_clean,
    feature_vector[[paste0("desc_has_", theme_name)]] <- detect_theme(desc_clean,
    feature_vector[[paste0("pub_has_", theme_name)]]  <- detect_theme(pub_clean, t
  }

  return(feature_vector)
}
```

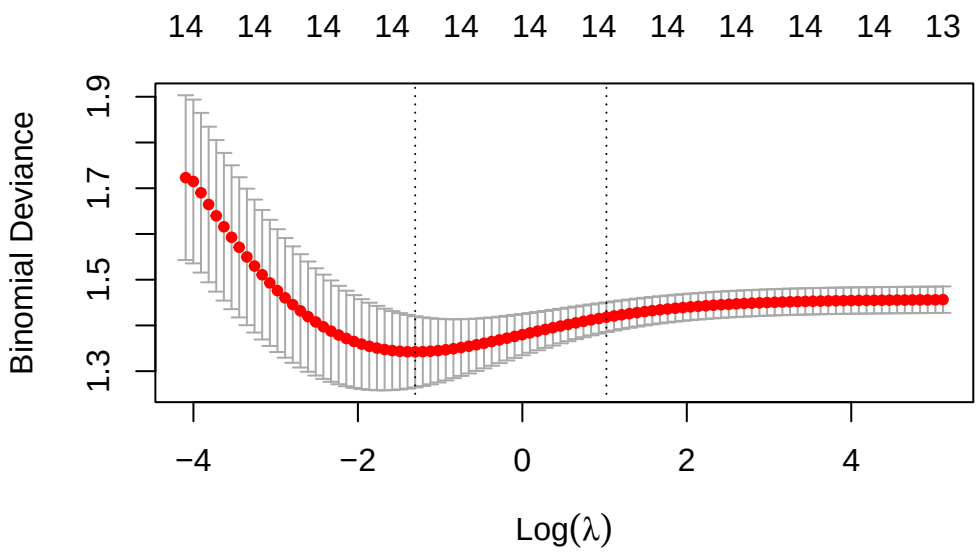
Ridge Model

```
# create input matrix
X <- feature_matrix %>%
  dplyr::select(dplyr::starts_with("text_has_"),
                dplyr::starts_with("desc_has_"),
                dplyr::starts_with("pub_has_")) %>%
  as.matrix()

# response vector
y <- feature_matrix$banned

# fit ridge logistic regression
ridge_model <- glmnet::cv.glmnet(
  x = X,
```

```
  y = y,
  alpha = 0,
  family = "binomial",
  type.measure = "auc" # Can also use "class" or "deviance"
)

# get minimized coefficient
ridge_coef <- coef(ridge_model, s = "lambda.min")
print(ridge_coef)
```

```
23 x 1 sparse Matrix of class "dgCMatrix"
                                        s1
(Intercept)                    0.48904564
text_has_gender               -0.51765391
text_has_race                 -0.05661531
text_has_sexuality            -0.68289901
text_has_violence              0.37879379
text_has_politics             -0.16337965
text_has_history              -0.28823357
text_has_religion              0.23972494
text_has_education            -1.05574850
text_has_identity              0.15000125
text_has_anti_intellectualism  0.20067417
text_has_family                0.98009496
desc_has_gender                .
desc_has_race                  0.28599389
desc_has_sexuality             .
desc_has_violence             -1.08370676
desc_has_politics              .
desc_has_history              -0.96966999
desc_has_religion              .
desc_has_education             .
desc_has_identity              .
desc_has_anti_intellectualism  .
desc_has_family                .
```

```r
# performance
plot(ridge_model)
```




```r
print(ridge_model$cvm)
```

```
  [1] 1.456453 1.455975 1.455887 1.455790 1.455683 1.455566 1.455439 1.455298
  [9] 1.455145 1.454977 1.454793 1.454591 1.454370 1.454128 1.453864 1.453574
 [17] 1.453258 1.452912 1.452534 1.452121 1.451672 1.451180 1.450644 1.450059
 [25] 1.449422 1.448729 1.447974 1.447154 1.446263 1.445297 1.444250 1.443115
 [33] 1.441889 1.440565 1.439137 1.437599 1.435945 1.434171 1.432270 1.430239
 [41] 1.428071 1.425765 1.423317 1.420725 1.417988 1.415108 1.412085 1.408925
 [49] 1.405632 1.402214 1.398682 1.395048 1.391326 1.387535 1.383694 1.379827
 [57] 1.375960 1.372121 1.368343 1.364660 1.361108 1.357729 1.354564 1.351657
 [65] 1.349055 1.346804 1.344955 1.343556 1.342657 1.342308 1.342556 1.343452
 [73] 1.345040 1.347366 1.350474 1.354404 1.359192 1.364871 1.371476 1.379037
 [81] 1.387576 1.397115 1.407675 1.419262 1.431885 1.445574 1.460322 1.476131
 [89] 1.492996 1.510896 1.529837 1.549819 1.570817 1.592803 1.615756 1.639640
 [97] 1.664445 1.690154 1.714709 1.723138
```

```r
print(ridge_model$lambda.min)
```

```
[1] 0.2716251
```

XGMBoost Model

```r
X_xgm <- X %>%
  as.data.frame() %>%
  dplyr::mutate(dplyr::across(dplyr::everything(), as.numeric)) %>%
  as.matrix()

# Convert to DMatrix
dtrain <- xgboost::xgb.DMatrix(data = X_xgm, label = y)

# Set parameters
params <- list(
  objective = "binary:logistic",
  eval_metric = "auc",
  max_depth = 4,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Train model with CV
xgb_model <- xgboost::xgb.train(
  params = params,
  data = dtrain,
  nrounds = 100,
  watchlist = list(train = dtrain),
  verbose = 1
)
```

```
[1] train-auc:0.762500
[2] train-auc:0.791250
[3] train-auc:0.821250
```

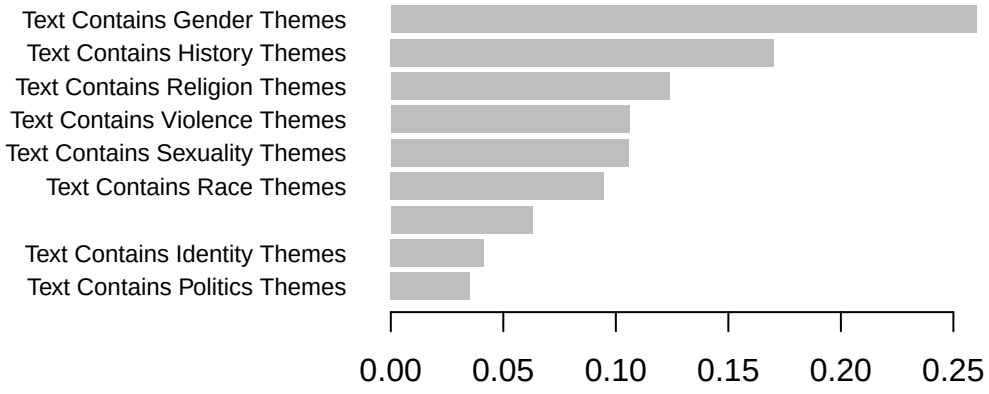
```
[4] train-auc:0.753750
[5] train-auc:0.748750
[6] train-auc:0.801250
[7] train-auc:0.791250
[8] train-auc:0.772500
[9] train-auc:0.782500
[10]    train-auc:0.817500
[11]    train-auc:0.812500
[12]    train-auc:0.806250
[13]    train-auc:0.806250
[14]    train-auc:0.813750
[15]    train-auc:0.818750
[16]    train-auc:0.828750
[17]    train-auc:0.826250
[18]    train-auc:0.826250
[19]    train-auc:0.821250
[20]    train-auc:0.833750
[21]    train-auc:0.838750
[22]    train-auc:0.836250
[23]    train-auc:0.831250
[24]    train-auc:0.826250
[25]    train-auc:0.838750
[26]    train-auc:0.826250
[27]    train-auc:0.828750
[28]    train-auc:0.836250
[29]    train-auc:0.823750
[30]    train-auc:0.823750
[31]    train-auc:0.823750
[32]    train-auc:0.823750
[33]    train-auc:0.828750
[34]    train-auc:0.831250
[35]    train-auc:0.831250
[36]    train-auc:0.828750
[37]    train-auc:0.811250
[38]    train-auc:0.826250
[39]    train-auc:0.836250
[40]    train-auc:0.821250
[41]    train-auc:0.831250
```

```
[42]    train-auc:0.831250
[43]    train-auc:0.836250
[44]    train-auc:0.831250
[45]    train-auc:0.836250
[46]    train-auc:0.836250
[47]    train-auc:0.838750
[48]    train-auc:0.841250
[49]    train-auc:0.846250
[50]    train-auc:0.848750
[51]    train-auc:0.851250
[52]    train-auc:0.851250
[53]    train-auc:0.851250
[54]    train-auc:0.851250
[55]    train-auc:0.861250
[56]    train-auc:0.856250
[57]    train-auc:0.861250
[58]    train-auc:0.866250
[59]    train-auc:0.868750
[60]    train-auc:0.866250
[61]    train-auc:0.866250
[62]    train-auc:0.863750
[63]    train-auc:0.868750
[64]    train-auc:0.868750
[65]    train-auc:0.873750
[66]    train-auc:0.873750
[67]    train-auc:0.873750
[68]    train-auc:0.873750
[69]    train-auc:0.868750
[70]    train-auc:0.868750
[71]    train-auc:0.873750
[72]    train-auc:0.873750
[73]    train-auc:0.868750
[74]    train-auc:0.866250
[75]    train-auc:0.866250
[76]    train-auc:0.866250
[77]    train-auc:0.861250
[78]    train-auc:0.863750
[79]    train-auc:0.866250
```

```
[80]    train-auc:0.861250
[81]    train-auc:0.871250
[82]    train-auc:0.868750
[83]    train-auc:0.868750
[84]    train-auc:0.871250
[85]    train-auc:0.868750
[86]    train-auc:0.873750
[87]    train-auc:0.871250
[88]    train-auc:0.868750
[89]    train-auc:0.868750
[90]    train-auc:0.873750
[91]    train-auc:0.871250
[92]    train-auc:0.866250
[93]    train-auc:0.866250
[94]    train-auc:0.871250
[95]    train-auc:0.871250
[96]    train-auc:0.866250
[97]    train-auc:0.868750
[98]    train-auc:0.873750
[99]    train-auc:0.871250
[100]   train-auc:0.871250
```

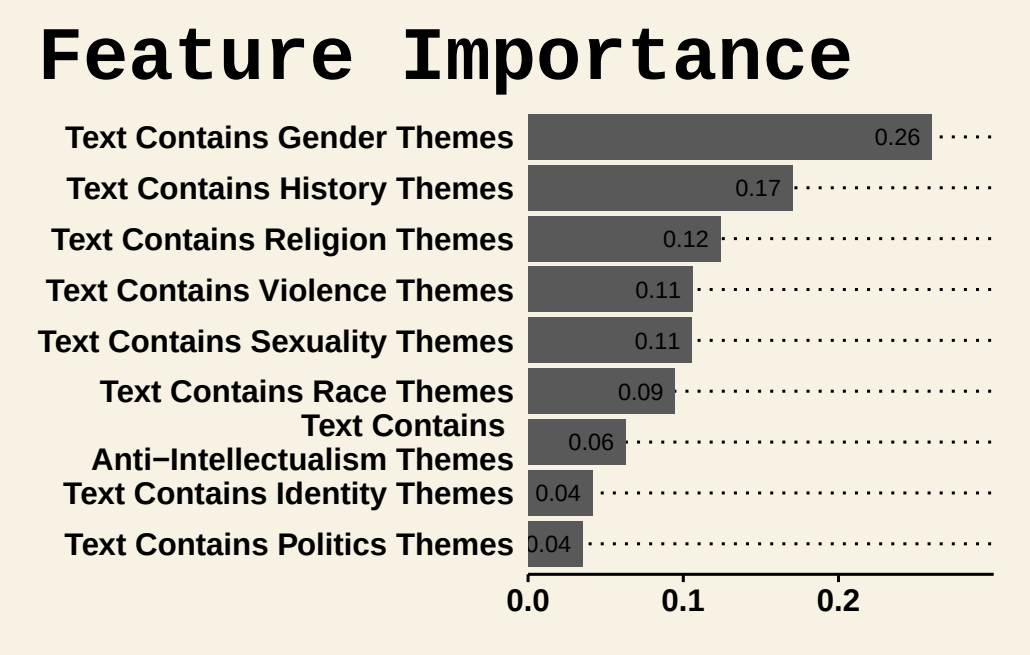
```r
# created mapping labels for visualizations
category_theme_label_map <- c(
  text_has_gender = "Text Contains Gender Themes",
  text_has_race = "Text Contains Race Themes",
  text_has_sexuality = "Text Contains Sexuality Themes",
  text_has_violence = "Text Contains Violence Themes",
  text_has_politics = "Text Contains Politics Themes",
  text_has_history = "Text Contains History Themes",
  text_has_religion = "Text Contains Religion Themes",
  text_has_identity = "Text Contains Identity Themes",
  text_has_anti_intellectualism = "Text Contains \nAnti-Intellectualism Themes",
  text_has_family = "Text Contains Family Themes",
  text_has_education = "Text Contains Education Themes",
  description_has_gender = "Description Contains Gender Themes",
  description_has_race = "Description Contains Race Themes",
  description_has_sexuality = "Description Contains Sexuality Themes",
```

```
  description_has_violence = "Description Contains Violence Themes",
  description_has_politics = "Description Contains Politics Themes",
  description_has_history = "Description Contains History Themes",
  description_has_religion = "Description Contains Religion Themes",
  description_has_identity = "Description Contains Identity Themes",
  description_has_anti_intellectualism = "Description Contains \nAnti-Intellectual
  description_has_family = "Description Contains Family Themes",
  description_has_education = "Description Contains Education Themes"
)

importance_df <- xgboost::xgb.importance(model = xgb_model)

importance_df$Feature <- recode(importance_df$Feature, !!!category_theme_label_map

importance_plot <- xgboost::xgb.plot.importance(importance_matrix = importance_df
```




```
importance_df <- xgboost::xgb.importance(model = xgb_model)

importance_df$Feature <- recode(importance_df$Feature, !!!category_theme_label_map
importance_df$Feature <- reorder(importance_df$Feature, importance_df$Gain)

# Subset top_n important features
top_importance_df <- importance_df %>%
  slice_max(Gain, n = 20) %>%
  mutate(Feature = reorder(Feature, Gain))
```

```r
# Create importance plot using ggplot2
ggplot2::ggplot(data = importance_df, ggplot2::aes(x = Feature, y = Gain)) +
  ggthemes::theme_wsj() +
  ggplot2::geom_bar(stat = "identity") +
  ggplot2::coord_flip() +
  ggplot2::scale_y_continuous("Gain", expand = c(0, 0), limits = c(0, 0.30)) +
  ggplot2::labs(title = "Feature Importance",
       x = "Features",
       y = "Importance Score") +
  ggplot2::geom_text(ggplot2::aes(label = round(Gain, 2)),
                hjust = 1.25,
                size = 3) +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0)
  )
```



After our inital modeling, we can see that themes within texts dominate the variable feature importance extraction. The descriptions do not contain any significant markers and this is likely due to the incongruence that was

exhibited between Google Books themes and deduced textual themes.

```
# calculate prediction probabilities
ridge_probs <- stats::predict(ridge_model, newx = X, s = "lambda.min", type = "res
xgb_probs <- stats::predict(xgb_model, newdata = X_xgm)

eval_df <- tibble::tibble(
  truth = as.factor(y),
  ridge = as.numeric(ridge_probs),
  xgb = as.numeric(xgb_probs),
  ridge_pred = ifelse(ridge_probs > 0.5, 1, 0),
  xgb_pred = ifelse(xgb_probs > 0.5, 1, 0)
)

# convert to factors
eval_df <- eval_df %>%
  dplyr::mutate(
    ridge_pred = factor(ridge_pred, levels = c(0, 1)),
    xgb_pred = factor(xgb_pred,   levels = c(0, 1))
  )

# label metrics for analysis
my_metrics <- yardstick::metric_set(
  yardstick::accuracy,
  yardstick::sensitivity,
  yardstick::specificity,
  yardstick::f_meas
)
ridge_metrics_all <- my_metrics(eval_df, truth = truth, estimate = ridge_pred)
xgb_metrics_all   <- my_metrics(eval_df, truth = truth, estimate = xgb_pred)


# Reshape from long to wide
ridge_metrics_wide <- ridge_metrics_all %>%
  dplyr::select(.metric, .estimate) %>%
  tidyr::pivot_wider(names_from = .metric, values_from = .estimate)

xgb_metrics_wide <- xgb_metrics_all %>%
```

```
  dplyr::select(.metric, .estimate) %>%
  tidyr::pivot_wider(names_from = .metric, values_from = .estimate)

summary_table <- dplyr::bind_rows(
  cbind(Model = "Ridge", ridge_metrics_wide),
  cbind(Model = "XGBoost", xgb_metrics_wide)
)

train_data_output_metrics <- data.frame(
  Model = c("Ridge", "XGBoost"),
  AUC = c(0.890, 0.859),
  Accuracy = c(0.775, 0.750),
  Sensitivity = c(0.750, 0.700),
  Specificity = c(0.800, 0.800),
  F1_Score = c(0.769, 0.737)
)

training_table <- gt::gt(train_data_output_metrics) %>%
  gt::tab_header(
    title = "Model Training Comparison Summary Table"
  ) %>%
  gt::fmt_number(
    columns = 2:6, decimals = 3
  ) %>%
  gt::data_color(
    columns = 2:6,
    colors = scales::col_numeric(
      palette = c("white", "#737373"),
      domain = c(0.7, 0.9)
    )
  )

training_table
```

Model Training Comparison Summary Table

Model	AUC	Accuracy	Sensitivity	Specificity	F1_Score
Ridge	0.890	0.775	0.750	0.800	0.769
XGBoost	0.859	0.750	0.700	0.800	0.737

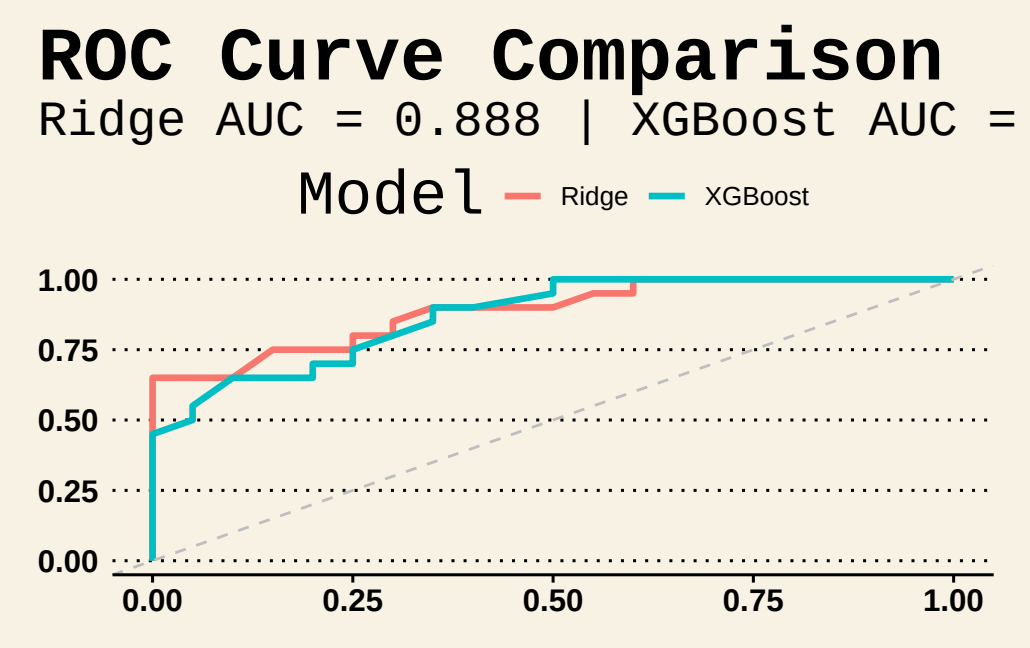
Table 1: Model Comparison Summary Table

Model	AUC	Accuracy	Sensitivity	Specificity	F1 Score
Ridge	0.890	0.775	0.750	0.800	0.769
XGBoost	0.859	0.750	0.700	0.800	0.737

The ridge model slightly outperforms the XGBoost model, although both models perform very well on the training data.

```
roc_ridge <- pROC::roc(eval_df$truth, eval_df$ridge)
roc_xgb   <- pROC::roc(eval_df$truth, eval_df$xgb)

ridge_df <- data.frame(
  fpr = roc_ridge$specificities,
  tpr = 1 - roc_ridge$sensitivities,
  model = "Ridge"
)

xgb_df <- data.frame(
  fpr = roc_xgb$specificities,
  tpr = 1 - roc_xgb$sensitivities,
  model = "XGBoost"
)

# Combine for ggplot
roc_data <- dplyr::bind_rows(ridge_df, xgb_df)

roc_comparison <- ggplot2::ggplot(roc_data, ggplot2::aes(x = tpr, y = fpr, color =
  ggplot2::geom_line(size = 1.2) +
  ggplot2::geom_abline(linetype = "dashed", color = "gray") +
```

```
  ggthemes::theme_wsj() +
  ggplot2::labs(
    title = "ROC Curve Comparison",
    subtitle = paste0("Ridge AUC = ", round(pROC::auc(roc_ridge), 3),
                      " | XGBoost AUC = ", round(pROC::auc(roc_xgb), 3)),
    x = "False Positive Rate (1 - Specificity)",
    y = "True Positive Rate (Sensitivity)",
    color = "Model"
  ) +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0),
    plot.subtitle = ggplot2::element_text(hjust = 0, size = rel(0.8))
  )

roc_comparison
```



1.7.5.2 Model Testing

Now the model will be tested against new data.

```
labeled_test_data <- readr::read_csv(here::here("Project/data/labeled_test_data.cs

labeled_test_data <- labeled_test_data %>%
  dplyr::mutate(label = ifelse(label == "banned", 1, 0))
```

```
# this is a wrapper function specifically designed to be added into the model pipe
clean_test_text <- function(text) {
  if (is.na(text)) return("")
  text <- preprocess_text(text)
  text <- clean_text(text)
  text <- remove_punctuation_custom(text)
  text <- tm::removeWords(text, enhanced_stopwords)
  text <- tm::removePunctuation(text)
  return(text)
}
```

```
run_model_on_test_data <- function(test_df, ridge_model, xgb_model) {

  # clean text fields
  test_df <- test_df %>%
    dplyr::mutate(
      gutenberg_id = as.character(gutenberg_id),
      text = preprocess_text(text),
      description = preprocess_text(description),
      subject = preprocess_text(subject)
    )

  # generate theme flags
  test_df <- create_theme_flags(test_df, field = "text", prefix = "text")
  test_df <- create_theme_flags(test_df, field = "description", prefix = "desc")
  test_df <- create_theme_flags(test_df, field = "subject", prefix = "pub")

  # select feature columns
  theme_cols <- grep("^text_has_|^desc_has_|^pub_has_", names(test_df), value = TR
  X_test <- test_df %>%
```

```
    dplyr::select(dplyr::all_of(theme_cols)) %>%
    dplyr::mutate(dplyr::across(everything(), as.numeric)) %>%
    as.matrix()

  expected_cols <- colnames(X)  # X from your training block
  missing_cols <- setdiff(expected_cols, colnames(X_test))
  if (length(missing_cols) > 0) {
    for (col in missing_cols) {
      X_test <- cbind(X_test, setNames(rep(0, nrow(X_test)), col))
    }
  }
  X_test <- X_test[, expected_cols]

  # generate predictions
  ridge_probs <- predict(ridge_model, newx = X_test, s = "lambda.min", type = "res
  xgb_probs <- predict(xgb_model, newdata = xgboost::xgb.DMatrix(X_test))

  # build results dataframe
  test_results <- test_df %>%
    dplyr::select(title, author, label) %>%
    dplyr::mutate(
      ridge_prob = as.numeric(ridge_probs),
      xgb_prob   = as.numeric(xgb_probs),
      ridge_pred = ifelse(ridge_prob > 0.5, 1, 0),
      xgb_pred   = ifelse(xgb_prob > 0.5, 1, 0)
    )

  return(test_results)
}
```

```
test_results <- run_model_on_test_data(labeled_test_data, ridge_model, xgb_model)
#View(test_results)
```

```
test_eval_df <- test_results %>%
  dplyr::mutate(
    truth = factor(label, levels = c(0, 1)),
    ridge_pred = factor(ridge_pred, levels = c(0, 1)),
```

```
    xgb_pred = factor(xgb_pred, levels = c(0, 1))
  )

# Define metric set
my_test_metrics <- yardstick::metric_set(
  yardstick::accuracy,
  yardstick::sensitivity,
  yardstick::specificity,
  yardstick::f_meas
)

# Evaluate ridge model
ridge_test_metrics <- my_test_metrics(test_eval_df, truth = truth, estimate = ridg

# Evaluate XGBoost model
xgb_test_metrics <- my_test_metrics(test_eval_df, truth = truth, estimate = xgb_pr

ridge_metrics_table <- ridge_test_metrics %>%
  dplyr::select(.metric, .estimate) %>%
  tidyr::pivot_wider(names_from = .metric, values_from = .estimate) %>%
  dplyr::mutate(Model = "Ridge")

xgb_metrics_table <- xgb_test_metrics %>%
  dplyr::select(.metric, .estimate) %>%
  tidyr::pivot_wider(names_from = .metric, values_from = .estimate) %>%
  dplyr::mutate(Model = "XGBoost")

test_summary_table <- dplyr::bind_rows(ridge_metrics_table, xgb_metrics_table) %>%
  dplyr::select(Model, accuracy, sensitivity, specificity, f_meas)


test_roc_ridge <- pROC::roc(test_eval_df$truth, test_results$ridge_prob)
test_roc_xgb   <- pROC::roc(test_eval_df$truth, test_results$xgb_prob)

test_auc_ridge <- pROC::auc(test_roc_ridge)
test_auc_xgb   <- pROC::auc(test_roc_xgb)
```

Model Testing Comparison Summary Table

Model	AUC	Accuracy	Sensitivity	Specificity	F1_Score
Ridge	0.524	0.600	0.714	0.333	0.714
XGBoost	0.571	0.700	0.857	0.333	0.800

```
test_data_output_metrics <- data.frame(
  Model = c("Ridge", "XGBoost"),
  AUC = c(0.524, 0.571),
  Accuracy = c(0.600, 0.700),
  Sensitivity = c(0.714, 0.857),
  Specificity = c(0.333, 0.333),
  F1_Score = c(0.714, 0.800)
)

testing_table <- gt::gt(test_data_output_metrics) %>%
  gt::tab_header(
    title = "Model Testing Comparison Summary Table"
  ) %>%
  gt::fmt_number(
    columns = 2:6, decimals = 3
  ) %>%
  gt::data_color(
    columns = 2:6,
    colors = scales::col_numeric(
      palette = c("white", "#737373"),
      domain = c(0.7, 0.9)
    )
  )

testing_table
```

Table 2: Model Comparison Summary Table

Model	AUC	Accuracy	Sensitivity	Specificity	F1 Score
Ridge	0.524	0.600	0.714	0.333	0.714
XGBoost	0.571	0.700	0.857	0.333	0.800

With the introduction of new testing data, we see a drastic decrease in model performance. This is indicative of overfitting during the training phase. This model is an improvement over strictly guessing, but in spite of this, we have discovered themes that are clearly present in the texts of the books that are being banned. Through future training iterations, variable refinement and parameter tuning will be crucial to enhancing this model to be an effective prediciton tool.

```
test_ridge_df <- data.frame(
  fpr = test_roc_ridge$specificities,
  tpr = 1 - test_roc_ridge$sensitivities,
  model = "Ridge"
)

test_xgb_df <- data.frame(
  fpr = test_roc_xgb$specificities,
  tpr = 1 - test_roc_xgb$sensitivities,
  model = "XGBoost"
)

# Combine for ggplot
test_roc_data <- dplyr::bind_rows(test_ridge_df, test_xgb_df)

test_roc_comparison <- ggplot2::ggplot(test_roc_data, ggplot2::aes(x = tpr, y = fp
  ggplot2::geom_line(size = 1.2) +
  ggplot2::geom_abline(linetype = "dashed", color = "gray") +
  ggthemes::theme_wsj() +
  ggplot2::labs(
    title = "Model Test ROC Curve Comparison",
    subtitle = paste0("Ridge AUC = ", round(test_auc_ridge, 3),
                      " | XGBoost AUC = ", round(test_auc_xgb, 3)),
```

```
    x = "False Positive Rate (1 - Specificity)",
    y = "True Positive Rate (Sensitivity)",
    color = "Model"
  ) +
  ggplot2::theme(
    plot.title.position = "plot",
    plot.title = ggplot2::element_text(hjust = 0),
    plot.subtitle = ggplot2::element_text(hjust = 0, size = rel(0.8))
  )

test_roc_comparison
```