CSCI-89B Final Project

The Computational Prometheus: Tracing the Evolution of Sci-Fi through Structural Topic Modeling

Liam Yardley

Contents

Setup

```
required_packages = c(
  # Data manipulation and plotting
  "tidyverse",
  # Downloading books from Project Gutenberg
  "gutenbergr",
  # Tidy text mining principles
```

```
  "tidytext",
  # Lemmatization tools
  "textstem",
  # Structural Topic Modeling
  "stm",
  # Text Mining infrastructure
  "tm",
  # Stemming algorithms
  "SnowballC",
  # Topic modeling evaluation
  "textmineR",
  # Word cloud visualization
  "wordcloud",
  # String processing
  "stringi",
  # Graph plotting
  "igraph",
  "gridExtra"
)

# Install missing packages automatically
new_packages = required_packages[!(required_packages %in% installed.packages()[,"Package"])]
if(length(new_packages)) install.packages(new_packages)

# --- Core Data Science Stack ---
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.5.2
```

```
## Warning: package 'tidyr' was built under R version 4.5.2
```

```
## Warning: package 'readr' was built under R version 4.5.2
```

```
## Warning: package 'forcats' was built under R version 4.5.2
```

```
## Warning: package 'lubridate' was built under R version 4.5.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.6
## v forcats   1.0.1     v stringr   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(stringi)

# --- NLP & Text Mining Stack ---
library(gutenbergr)
```

```
## Warning: package 'gutenbergr' was built under R version 4.5.2
```

```
library(tidytext)
library(textstem)
```

```
## Warning: package 'textstem' was built under R version 4.5.2
## Loading required package: koRpus.lang.en
## Warning: package 'koRpus.lang.en' was built under R version 4.5.2
## Loading required package: koRpus
## Warning: package 'koRpus' was built under R version 4.5.2
## Loading required package: sylly
## Warning: package 'sylly' was built under R version 4.5.2
## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()
##
##
## Attaching package: 'koRpus'
##
## The following object is masked from 'package:readr':
##
##     tokenize
```

```
library(tm)
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
##
##
## Attaching package: 'tm'
##
## The following object is masked from 'package:koRpus':
##
##     readTagged
```

```
library(SnowballC)

# --- Modeling & Visualization Stack ---
library(stm)
```

```
## stm v1.3.8 successfully loaded. See ?stm for help.
##  Papers, resources, and other materials at structuraltopicmodel.com
```

```
library(textmineR)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
```

```
##     expand, pack, unpack
##
##
## Attaching package: 'textmineR'
##
## The following object is masked from 'package:Matrix':
##
##     update
##
## The following object is masked from 'package:stats':
##
##     update
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.5.2
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
##
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
##
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
##
## The following object is masked from 'package:tidyr':
##
##     crossing
##
## The following object is masked from 'package:tibble':
##
##     as_data_frame
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.5.2
```

```
##
```

```
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
knitr::opts_chunk$set(echo = TRUE)
```

Useful Functions

```
# --- Pretty Tables ---
library(knitr)
create_table = function(data_list,
                                row_labels = NULL,
                                col_labels = NULL,
                                transpose = FALSE,
                                digits = 3,
                                caption = "Summary Table") {

  # Combine inputs into a data frame
  df = as.data.frame(data_list, stringsAsFactors = FALSE)

  # Assign row and column names
  if (!is.null(row_labels)) rownames(df) = row_labels
  if (!is.null(col_labels)) colnames(df) = col_labels

  # Optional transpose
  if (transpose) {
    df = as.data.frame(t(df))
  }

  # Apply numeric formatting
  df[] = lapply(df, function(x) {
      if (is.numeric(x)) {
        return(formatC(x, digits = digits, format = "fg", flag = "#"))
      } else {
        return(x)
      }
    })

  # Return nicely formatted knitr::kable
  knitr::kable(df,

               booktabs = TRUE,
               longtable = TRUE,
               caption = caption,
               position = "h!",
               row.names = FALSE)
}

# --- Printing ---
# These were removed after images were generated

# For ggplots
```

```
# ggsave("filename.pdf",
#        plot = plot,
#        width = 36, # Adjust as needed
#        height = 12, # Adjust as needed
#        units = "cm")
#
# # For base R
# pdf("filename.pdf", width = 12, height = 8) # Adjust as needed (in inches)
#
# # Plotting code here
#
# dev.off()
```

Introduction

This workflow demonstrates the application of Structural Topic Modeling (STM) for literary genre classification. Using Mary Shelley's Frankenstein and The Last Man as test cases, we explore how probabilistic topic models can quantify the thematic distance between ambiguous texts and established genre categories.

Shelley's work occupies a unique position in literary history, often debated as either Romantic, Gothic Horror, or the progenitor of Science Fiction. To resolve this classification quantitatively, we construct a comparative corpus comprising:

- Realism/Romance: Representing the stylistic baseline of the Romantic/Victorian era.
- Gothic Horror: Representing themes of fear, the supernatural, and atmospheric dread.
- Science Fiction: Representing themes of technology, exploration, and future speculation.

Objectives

This analysis proceeds in three stages:

Stylometric Comparison (PCA): Mapping the syntactic similarity between authors.

Topic Modeling (STM): Identifying latent thematic clusters (e.g., "Naval Exploration" vs. "Space Travel").

Genre Classification: Using topic prevalence to determine if Shelley's narrative structure aligns more closely with the static horror of the Gothic or the dynamic exploration of Science Fiction.

Note: While modern discussions often link Frankenstein to Artificial Intelligence (see Botting, 2017), this analysis focuses strictly on the historical genre evolution from 1800 to 1950.

Data Handling

Corpus Selection

The dataset consists of 44 novels selected from Project Gutenberg. To ensure a robust classification model, we define three "Control Groups" (Romance, Gothic, Sci-Fi) to act as the training context for our "Target Works" (Shelley).

The selection includes canonical texts chosen for their genre-defining characteristics. For instance, War of the Worlds and The Plague Ship are included specifically to provide a thematic baseline for the apocalyptic narratives found in Shelley's The Last Man.

The code below defines the Project Gutenberg IDs for the corpus and downloads the raw text data. These could be changed to adapt this code or genre classification of any works.

This data is available at Project Gutenberg (2025), https://www.gutenberg.org/.

```
# --- Download Data ---
shelley_ids = c(
  # Target Works
  84, # Frankenstein by Mary Shelley
  18247 # The Last Man by Mary Shelley
)

romance_ids = c(
  # Romance
  1342, # Pride and Prejudice by Jane Austen
  161, #    Sense and Sensibility by    Jane Austen
  1260, # Jane Eyre by Charlotte Bronte
  768, # Wuthering Heights by Emily Brontë
  1400, # Great Expectations by Charles Dickens
  158, # Emma by Jane Austen
  730, # Oliver Twist by Charles Dickens
  98, # A Tale of Two Cities by Charles Dickens
  110, # Tesse of the d'Urbervilles by Thomas Hardy
  107, # Far from the Madding Crowd by Thomas Hardy
  514, # Little Women by Louisa May Alcott
  1399 # Anna Karenina by Leo Tolstoy
)

gothic_ids = c(
  # Gothic Horror
  345, # Dracula by Bram Stoker
  1188, #   The Lair of the White Worm by Bram Stoker
  3781, #   The Jewel of Seven Stars by Bram Stoker
  43, # Dr. Jekyll and Mr. Hyde by Robert Louis Stevenson
  174, # The Picture of Dorian Gray by Oscar Wilde
  10007, # Carmilla by J. Sheridan Le Fanu
  175, # Phantom of the Opera by Gaston Leroux
  2610, # The Hunchback of Notre-Dame by Victor Hugo
  6087, # The Vampyre by John Polidori
  696, #    The Castyle of Otranto  by Horace Walpole
  3268, # The Mysteries of Udolpho by Ann Radcliffe
  41,   # The Legend of Sleepy Hollow byWashington Irving
  583 # The Woman in White by Wilkie Collins
)

scifi_ids = c(
  35, # The Time Machine by HG Wells
  36, # The War of the Worlds by HG Wells
  159, # The Island of Doctor Moreau by HG Wells
  1013, # The First Men in the Moon by H.G. Wells
  5230, # The Invisible Man by H.G. Wells
  164, # 20,000 Leagues Under the Sea by Jules Verne
  18857, # Journey to the Center of the Earth by Jules Verne
  139, # The Lost World by Arthur Conan Doyle
  126, # The Poison Belt by Arthur Conan Doyle
  32032, # Second Variety by Philip K.Dick
  32154, # The Variable Man by Philip K.Dick
```

```
  16921, # Plague Ship by Andre Norton
  20782, # Triplanetary by E.E. "Doc" Smith
  32, # Herland by Charlotte Perkins Gilman
  21970, # The Scarlet Plague by Jack London
  32530 # Armageddon 2419 A.D. by Philip Francis Nowlan
)

book_ids = c(shelley_ids, romance_ids, gothic_ids, scifi_ids)

# --- Get Books ---
mirror = "http://mirrors.xmission.com/gutenberg/"

raw_books = gutenberg_download(book_ids, meta_fields = c("title", "author"), mirror = mirror)

# --- Clean and Label ---
books_clean = raw_books %>%
  # Remove Gutenberg Licence info
  filter(!is.na(text), text != "") %>%
  mutate(
    # Assign GENRE
    genre = case_when(
      gutenberg_id %in% shelley_ids ~ "Shelley",
      gutenberg_id %in% romance_ids ~ "Realism/Romance",
      gutenberg_id %in% gothic_ids  ~ "Gothic Horror",
      gutenberg_id %in% scifi_ids   ~ "Sci-Fi",
      TRUE ~ "Other"
    ),

    # Assign ERA (Approximate historical groups)
    era = case_when(
      # Modern Sci-Fi (1930s-1950s)
      gutenberg_id %in% c(32032, 32154, 16921, 20782) ~ "Modern (Atomic Age)",

      # Late Victorian(1880s-1920s)
      gutenberg_id %in% c(35, 36, 159, 139, 8748, 32, 345, 1188, 3781, 174, 43, 175, 110, 107) ~ "Late V

      # Default to Romantic / Early-Mid Victorian (1800s-1870s)
      TRUE ~ "Romantic/Mid-Victorian (Pre-1880)"
    ),
  )

# --- Book Summary ---
book_summary = books_clean %>%
  select(genre, title, author, era, gutenberg_id) %>%
  distinct(gutenberg_id, .keep_all = TRUE) %>%
  arrange(genre, author)

create_table(book_summary,
             caption = "Corpus of Selected Texts",
             col_labels=c("Genre", "Book Title", "Author", "Era", "ID"))
```

Table 1: Corpus of Selected Texts

Genre	Book Title	Author	Era	ID
Gothic Horror	The Woman in White	Collins, Wilkie	Romantic/Mid-Victorian (Pre-1880)	583.
Gothic Horror	Notre-Dame de Paris	Hugo, Victor	Romantic/Mid-Victorian (Pre-1880)	2610.
Gothic Horror	The Legend of Sleepy Hollow	Irving, Washington	Romantic/Mid-Victorian (Pre-1880)	41.0
Gothic Horror	Carmilla	Le Fanu, Joseph Sheridan	Romantic/Mid-Victorian (Pre-1880)	10007.
Gothic Horror	The Phantom of the Opera	Leroux, Gaston	Late Victorian (1880-1920)	175.
Gothic Horror	The Vampyre; a Tale	Polidori, John William	Romantic/Mid-Victorian (Pre-1880)	6087.
Gothic Horror	The Mysteries of Udolpho	Radcliffe, Ann Ward	Romantic/Mid-Victorian (Pre-1880)	3268.
Gothic Horror	The Strange Case of Dr. Jekyll and Mr. Hyde	Stevenson, Robert Louis	Late Victorian (1880-1920)	43.0
Gothic Horror	Dracula	Stoker, Bram	Late Victorian (1880-1920)	345.
Gothic Horror	The Lair of the White Worm	Stoker, Bram	Late Victorian (1880-1920)	1188.
Gothic Horror	The Jewel of Seven Stars	Stoker, Bram	Late Victorian (1880-1920)	3781.
Gothic Horror	The Castle of Otranto	Walpole, Horace	Romantic/Mid-Victorian (Pre-1880)	696.
Gothic Horror	The Picture of Dorian Gray	Wilde, Oscar	Late Victorian (1880-1920)	174.
Realism/Romance	Little Women	Alcott, Louisa May	Romantic/Mid-Victorian (Pre-1880)	514.
Realism/Romance	Pride and Prejudice	Austen, Jane	Romantic/Mid-Victorian (Pre-1880)	1342.
Realism/Romance	Sense and Sensibility	Austen, Jane	Romantic/Mid-Victorian (Pre-1880)	161.
Realism/Romance	Emma	Austen, Jane	Romantic/Mid-Victorian (Pre-1880)	158.
Realism/Romance	Jane Eyre: An Autobiography	Brontë, Charlotte	Romantic/Mid-Victorian (Pre-1880)	1260.
Realism/Romance	Wuthering Heights	Brontë, Emily	Romantic/Mid-Victorian (Pre-1880)	768.
Realism/Romance	Great Expectations	Dickens, Charles	Romantic/Mid-Victorian (Pre-1880)	1400.
Realism/Romance	Oliver Twist	Dickens, Charles	Romantic/Mid-Victorian (Pre-1880)	730.
Realism/Romance	A Tale of Two Cities	Dickens, Charles	Romantic/Mid-Victorian (Pre-1880)	98.0
Realism/Romance	Tess of the d'Urbervilles: A Pure Woman	Hardy, Thomas	Late Victorian (1880-1920)	110.
Realism/Romance	Far from the Madding Crowd	Hardy, Thomas	Late Victorian (1880-1920)	107.
Realism/Romance	Anna Karenina	Tolstoy, Leo, graf	Romantic/Mid-Victorian (Pre-1880)	1399.

Genre	Book Title	Author	Era	ID
Sci-Fi	Second Variety	Dick, Philip K.	Modern (Atomic Age)	32032.
Sci-Fi	The Variable Man	Dick, Philip K.	Modern (Atomic Age)	32154.
Sci-Fi	The Lost World	Doyle, Arthur Conan	Late Victorian (1880-1920)	139.
Sci-Fi	The Poison Belt	Doyle, Arthur Conan	Romantic/Mid-Victorian (Pre-1880)	126.
Sci-Fi	Herland	Gilman, Charlotte Perkins	Late Victorian (1880-1920)	32.0
Sci-Fi	The Scarlet Plague	London, Jack	Romantic/Mid-Victorian (Pre-1880)	21970.
Sci-Fi	Plague Ship	Norton, Andre	Modern (Atomic Age)	16921.
Sci-Fi	Armageddon—2419 A.D.	Nowlan, Philip Francis	Romantic/Mid-Victorian (Pre-1880)	32530.
Sci-Fi	Triplanetary	Smith, E. E. (Edward Elmer)	Modern (Atomic Age)	20782.
Sci-Fi	Twenty Thousand Leagues under the Sea	Verne, Jules	Romantic/Mid-Victorian (Pre-1880)	164.
Sci-Fi	A Journey to the Centre of the Earth	Verne, Jules	Romantic/Mid-Victorian (Pre-1880)	18857.
Sci-Fi	The Time Machine	Wells, H. G. (Herbert George)	Late Victorian (1880-1920)	35.0
Sci-Fi	The War of the Worlds	Wells, H. G. (Herbert George)	Late Victorian (1880-1920)	36.0
Sci-Fi	The island of Doctor Moreau	Wells, H. G. (Herbert George)	Late Victorian (1880-1920)	159.
Sci-Fi	The First Men in the Moon	Wells, H. G. (Herbert George)	Romantic/Mid-Victorian (Pre-1880)	1013.
Sci-Fi	The Invisible Man: A Grotesque Romance	Wells, H. G. (Herbert George)	Romantic/Mid-Victorian (Pre-1880)	5230.
Shelley	Frankenstein; Or, The Modern Prometheus	Shelley, Mary Wollstonecraft	Romantic/Mid-Victorian (Pre-1880)	84.0
Shelley	The Last Man	Shelley, Mary Wollstonecraft	Romantic/Mid-Victorian (Pre-1880)	18247.

Structural Topic Models require a high volume of discrete data points to detect patterns effectively. Since a novel constitutes a single, lengthy document, we must segment the text to increase our sample size. These works will be chunked into 100 words, to allow for sufficient data for Structural Topic Modelling. This has created a total of 42,204 documents as my corpus. Below is a sample of what these chunks would look like.

```
# --- Chunking ---
documents_df = books_clean %>%
  unnest_tokens(word, text) %>%
  # Lemmatise (instead of Stemming)
  mutate(word = lemmatize_words(word)) %>%
  group_by(title) %>%
  # Create chunks of 100 words
  mutate(chunk_id = row_number() %/% 100) %>%
  # Ungroup first to ensure metadata carries over correctly
  ungroup() %>%
  group_by(title, genre, era, chunk_id) %>%
  summarize(text_segment = paste(word, collapse = " "), .groups = "drop") %>%
  # Create Unique Doc IDs
```

```
  mutate(doc_id = paste(str_sub(title, 1, 10), chunk_id, sep = "_"))

# --- Check Results ---
print(paste("Total Documents Created:", nrow(documents_df)))
```

```
## [1] "Total Documents Created: 44168"
```

```
create_table(table(documents_df$genre), caption = "Number of Documents per Genre", col_labels=c("Genre"
```

Table 2: Number of Documents per Genre

Genre	Document Count
Gothic Horror	13178.
Realism/Romance	20303.
Sci-Fi	8167.
Shelley	2520.

```
create_table(head(documents_df,10), caption = "Sample of Documents")
```

Table 3: Sample of Documents

title	genre	era	chunk_id	text_id	segment	doc_id
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Victorian (Pre-1880)	0	1	A journey to the centre of the earth by jules verne redactor's note journey to the centre of the earth be numb v002 in the taves and michaluk number of the work of jules verne first publish in england by griffith and farran 1871 this edition be not a translation at all but a complete re write of the novel with portion add and omit and name change the much reprint version it be enter into project gutenberg for reference purpose only a good translation be *a journey into the interior of the earth* translate by rev f a	A Journey _0
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Victorian (Pre-1880)	1	1	Middleson also available on project gutenberg table of content chapter 1 my uncle make a great discovery chapter 2 the mysterious parchment chapter 3 a astound discovery chapter 4 we start on the journey chapter 5 first lesson in climb chapter 6 our voyage to iceland chapter 7 conversation and discovery chapter 8 the eider down hunter off at last chapter 9 our start we meet with adventure by the way chapter 10 travel in iceland chapter 11 we reach mount sneffels the reykir chapter 12 the ascent of mount sneffels chapter 13 the shadow of scartaris chapter 14 the	A Journey _1
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Victorian (Pre-1880)	2	1	Middle journey commence chapter 15 we continue our descent chapter 16 the eastern tunnel chapter 17 deep and deep the coal mine chapter 18 the wrong road chapter 19 the western gallery a new route chapter 20 water where be it a bitter disappointment chapter 21 under the ocean chapter 22 sunday below grind chapter 23 alone chapter 24 lose chapter 25 the whisper gallery chapter 26 a rapid recovery chapter 27 the central sea chapter 28 launch the raft chapter 29 on the water a raft voyage chapter 30 terrific saurian combat chapter 31 the sea monster chapter 32	A Journey _2

title	genre	era	chunk_id	text_segment	doc_id
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Mid-Victorian (Pre-1880)	3/00	the battle of the element chapter 33 our route reverse chapter 34 a voyage of discovery chapter 35 discovery upon discovery chapter 36 what be it chapter 37 the mysterious dagger chapter 38 no outlet blast the rock chapter 39 the explosion and its result chapter 40 the ape gigans chapter 41 hunger chapter 42 the volcanic shaft chapter 43 daylight at last chapter 44 the journey end chapter 1 my uncle make a great discovery look back to all that have occur to me since that eventful day i be scarcely able to believe in the reality of my	A Journey _3
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Mid-Victorian (Pre-1880)	4/00	adventure they be truly so wonderful that even now i be bewilder when i think of them my uncle be a german have marry my mother's sister a englishwoman be very much attach to his fatherless nephew he invite me to study under him in his home in the fatherland this home be in a large town and my uncle a professor of philosophy chemistry geology mineralogy and many other ology one day after pass some hour in the laboratory my uncle be absent at the time i suddenly feel the necessity of renovate the tissue i i.e i i	A Journey _4
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Mid-Victorian (Pre-1880)	5/00	hungry and be about to rouse up our old french cook when my uncle professor von hardwigg suddenly open the street door and come rush upstairs now professor hardwigg my worthy uncle be by no mean a bad sort of man he be however choleric and original to bear with him mean to obey and scarcely have his heavy foot resound within our joint domicile than he shout for me to attend upon him harry harry harry i hasten to obey but before i can reach his room jump three step at a time he be stamp his right	A Journey _5
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Mid-Victorian (Pre-1880)	6/00	upon the land harry he cry in a frantic tone be you come up now to tell the truth at that moment i be far much interest in the question as to what be to constitute our dinner than in any problem of science to me soup be much interest than soda a omelette much tempt than arithmetic and a artichoke of ten time much value than any amount of asbestos but my uncle be not a man to be keep wait so adjourn therefore all minor question i present myself before him he be a very learn man	A Journey _6
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Mid-Victorian (Pre-1880)	7/00	how much person in this category supply themselves with information as peddler do with good for the benefit of other and lie up store in order to diffuse them abroad for the benefit of society in general not so my excellent uncle professor hardwigg he study he consume the midnight oil he pore over heavy tome and digest huge quarto and folio in order to keep the knowledge acquire to himself there be a reason and it may be regard as a good one why my uncle object to display his learn much than be absolutely necessary he stammer and	A Journey _7

title	genre	era	chunk_id	segment	doc_id
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Victorian (Pre-1880)	8/0	when intent upon explain the phenomenon of the heaven be apt to find himself at fault and allude in such a vague way to sun moon and star that few be able to comprehend his mean to tell the honest truth when the right word would not come it be generally replace by a very powerful adjective in connection with the science there be many almost unpronounceable name name very much resemble that of welsh village and my uncle be very fond of use them his habit of stammer be not thereby improve in fact there be period in his	A Journey_8
A Journey to the Centre of the Earth	Sci-Fi	Romantic/Victorian (Pre-1880)	9/0	discourse when he would finally give up and swallow his discomfiture in a glass of water as i say my uncle professor hardwigg be a very learn man and i now add a much kind relative i be bind to him by the double tie of affection and interest i take deep interest in all his doing and hope some day to be almost as learn myself it be a rare thing for me to be absent from his lecture like him i prefer mineralogy to all the other science my anxiety be to gain real i knowledge of the	A Journey_9

A preliminary check of the document counts reveals a significant class imbalance. The Romantic novels in this corpus are substantially longer than the Gothic or Sci-Fi works, resulting in a disproportionate number of text chunks. To prevent the model from overfitting to the dominant class (Realism/Romance), we implement a random down-sampling strategy to cap each genre at 8,000 documents.

Note: This step is specific to the distribution of this particular corpus. If replicating this analysis with a smaller dataset, this block should be modified or omitted to avoid discarding data.

```
# --- Balance Corpus by Sampling Genres ---
set.seed(42)
target = 8000

balanced_df = documents_df %>%
  group_by(genre) %>%
  slice_sample(n = target) %>%
  ungroup()

df_original = as.data.frame(table(documents_df$genre))
colnames(df_original) = c("Genre", "Original_Count")

df_balanced = as.data.frame(table(balanced_df$genre))
colnames(df_balanced) = c("Genre", "Sample_Count")

sampling_summary = merge(df_original, df_balanced, by = "Genre")

# Add a column showing the Percentage Kept
sampling_summary$Percent_Kept = paste0(
  round((sampling_summary$Sample_Count / sampling_summary$Original_Count) * 100, 1),
  "%"
)

create_table(sampling_summary, caption = "Number of Documents per Genre After Sampling", col_labels=c("
```

Table 4: Number of Documents per Genre After Sampling

Genre	Original Count	Sample Count	Percent Kept
Gothic Horror	13178.	8000.	60.7%
Realism/Romance	20303.	8000.	39.4%
Sci-Fi	8167.	8000.	98%
Shelley	2520.	2520.	100%

Data Preparation

Before conducting STM, we need to preprocess the data I will be using the in-built `textProcessor`, and I will be removing stopwords, numbers, punctuation and stemming the words, as well as restricting to stems of a length 3 or more.

Initially used stemming, this did not yield nice, interpretable words, which is especially important when analysing literature and considering key themes and topics.

Consideration of Stop Words

One issue that may arise (and did on my first attempt) is very common words dominating topics, but are ultimately meaningless. For example, *chapter*, yes, this could have some meaning, like the 'chapter of my life', but that would be a rare occurrence, instead this word will exist dozens or more times in each novel, but serve no purpose. Therefore, I will create a custom set of stop words to eliminate these.

Another issue I came across, was the STM model being lazy and creating topics that were just books. As we have chunked each novel into 100 word chunks, there is a lot of data for each book, and therefore the model can just group these chunks into topics based on the book they came from, rather than any thematic or topical similarity. To counteract this, I created a custom stop word list that included book titles, author names, and other common words that were not useful for topic modelling. One major think that could impact that would be names, therefore I have added the main character names from each book to the stop word list to try and prevent this issue (names generated with help from AI: Gemini)

```
# --- Custom Stopwords ---
# Generation of names to be omitted supported by Gemini
my_stopwords = c(
  # --- Shelley ---
  # Frankenstein
  "victor", "frankenstein", "elizabeth", "lavenza", "clerval", "henry", "walton",
  "robert", "justine", "moritz", "felix", "agatha", "safie", "delacey", "creature", "monster",
  # The Last Man
  "lionel", "verney", "adrian", "raymond", "perdita", "idris", "evadne", "windsor",

  # --- Realism & Romance ---
  # Austen (P&P, S&S, Emma)
  "darcy", "bennet", "bingley", "wickham", "collins", "lydia", "kitty", "gardiner",
  "elinor", "marianne", "dashwood", "edward", "ferrars", "willoughby", "brandon",
  "emma", "woodhouse", "knightley", "churchill", "fairfax", "bates", "elton",
  # Bronte Sisters (Jane Eyre, Wuthering Heights)
  "jane", "eyre", "rochester", "adele", "varens", "bertha", "mason", "rivers", "temple",
  "heathcliff", "catherine", "cathy", "earnshaw", "linton", "lockwood", "nelly", "dean",
  # Dickens (Great Expectations, Oliver Twist, Tale of Two Cities)
  "pip", "pirrip", "estella", "havisham", "gargery", "magwitch", "jaggers", "drummle",
  "oliver", "twist", "fagin", "sikes", "nancy", "bumble", "brownlow", "dodger",
```

```
  "carton", "darnay", "manette", "lorry", "defarge", "pross", "stryver",
  # Hardy (Tess, Far From Madding Crowd)
  "tess", "durbeyfield", "angel", "clare", "alec", "durberville",
  "bathsheba", "everdene", "gabriel", "oak", "troy", "boldwood",
  # Other Classics (Little Women, Anna Karenina)
  "jo", "meg", "beth", "amy", "march", "laurie", "laurence", "bhaer",
  "anna", "karenina", "vronsky", "levin", "kitty", "stepan", "stiva", "oblonsky", "dolly", "alexey",

  # --- Gothic Horror ---
  # Stoker (Dracula, Lair, Jewel)
  "dracula", "helsing", "harker", "jonathan", "mina", "seward", "holmwood", "godalming", "renfield",
  "adam", "salton", "arabella", "caswall", "oolanga", "mim",
  "malcolm", "ross", "trelawny", "tera", "corbeck", "winchester",
  # Wilde, Stevenson, Le Fanu
  "dorian", "gray", "basil", "hallward", "wotton", "sybil", "vane",
  "jekyll", "hyde", "utterson", "lanyon", "enfield", "poole",
  "carmilla", "laura", "spielsdorf", "karnstein",
  # Leroux, Hugo, Polidori
  "phantom", "erik", "christine", "daae", "raoul", "chagny", "giry",
  "quasimodo", "esmeralda", "frollo", "phoebus", "gringoire",
  "ruthven", "aubrey", "ianthe",
  # Walpole, Radcliffe, Irving, Collins
  "manfred", "conrad", "isabella", "theodore", "matilda", "jerome", "otranto",
  "emily", "aubert", "valancourt", "montoni", "udolpho", "laurentini",
  "ichabod", "crane", "katrina", "tassel", "brom", "bones",
  "walter", "hartright", "marian", "halcombe", "fosco", "percival", "glyde", "catherick",

  # --- Sci-Fi ---
  # Wells (Time Machine, War of Worlds, Moreau)
  "weena", "morlock", "eloi", "traveller", "ogilvy", "henderson",
  "prendick", "moreau", "montgomery",
  # Verne (20k Leagues, Journey)
  "nemo", "nautilus", "aronnax", "conseil", "ned", "land",
  "lidenbrock", "axel", "hans", "grauben",
  # Doyle, Burroughs (Lost World, Mars)
  "challenger", "malone", "roxton", "summerlee",
  "john", "carter", "dejah", "thoris", "tars", "tarkas", "sola", "woola", "barsoom",
  # Dick, Norton, Smith, Gilman (Modern Era)
  "hendricks", "tasso", "klaus", "rudi", "david",
  "cole", "reinhart", "kaplan", "sherikov",
  "dane", "thorson", "rip", "ali", "kamil", "jellico", "tau", "sargol",
  "costigan", "clio", "bradley", "rog", "nerado", "samms", "kinnison",
  "van", "vandyck", "terry", "jeff", "ellador", "celia", "alima", "herland",

  # --- Procedural Junk ---
  "chapter", "vol", "volume", "book", "part", "gutenberg", "project", "edition",
  "contents", "title", "author", "prologue", "epilogue", "illustration"
)
```

```
# --- Prepare Data for STM ---
processed = textProcessor(
  documents = balanced_df$text_segment,
  metadata = balanced_df[, c("title", "genre", "era")],
  lowercase = TRUE,
```

```
  removestopwords = TRUE,
  removenumbers = TRUE,
  removepunctuation = TRUE,
  stem = FALSE,
  customstopwords = my_stopwords,
  wordLengths = c(3, Inf),
  verbose = TRUE
)
```

```
## Building corpus...
## Converting to Lower Case...
## Removing punctuation...
## Removing stopwords...
## Remove Custom Stopwords...
## Removing numbers...
## Creating Output...
```

```
# Build documents, vocabulary, and aligned metadata
out = prepDocuments(processed$documents, processed$vocab, processed$meta, lower.thresh = 10)
```

```
## Removing 22428 of 30616 terms (60013 of 1118854 tokens) due to frequency
## Your corpus now has 26520 documents, 8188 terms and 1058841 tokens.
```

```
docs = out$documents
vocab = out$vocab
meta = out$meta

# --- Summaries ---
num_docs = length(docs)
vocab_size = length(vocab)
meta_dim = dim(meta)

cat("Number of documents:", num_docs, "\n\n")
```

```
## Number of documents: 26520
```

```
cat("Vocabulary size:", vocab_size, "\n\n")
```

```
## Vocabulary size: 8188
```

```
cat("Metadata dims (rows, cols):", meta_dim[1], meta_dim[2], "\n\n")
```

```
## Metadata dims (rows, cols): 26520 3
```

```
# Average document length (in tokens) after preprocessing:
avg_len = mean(sapply(docs, length))
cat("Avg tokens per document (post-preprocessing):", round(avg_len, 2), "\n\n")
```

```
## Avg tokens per document (post-preprocessing): 79.85
```

```
# Check
head(vocab, 20)
```

```
##  [1] "'ll"         "'re"         "'ve"         "abandon"     "abandonment"
##  [6] "abate"       "abbess"      "abbey"       "abhor"       "abhorrence"
## [11] "abide"       "ability"     "abject"      "able"        "abnormal"
## [16] "aboard"      "abominable"  "abound"      "abraham"     "abreast"
```

```
head(meta)
```

```
##                         title         genre                                era
## 1         The Woman in White Gothic Horror Romantic/Mid-Victorian (Pre-1880)
## 2         The Woman in White Gothic Horror Romantic/Mid-Victorian (Pre-1880)
## 3        Notre-Dame de Paris Gothic Horror Romantic/Mid-Victorian (Pre-1880)
## 4 The Lair of the White Worm Gothic Horror        Late Victorian (1880-1920)
## 5   The Phantom of the Opera Gothic Horror        Late Victorian (1880-1920)
## 6                    Dracula Gothic Horror        Late Victorian (1880-1920)
```

```
# Example preprocessed document
cat("\nExample preprocessed document (first 20 tokens):\n")
```

```
##
## Example preprocessed document (first 20 tokens):
```

```
print(docs[[1]][1:20])
```

```
##  [1]   43    1  224    1  342    1  345    1  496    1  654    1  670    1 1262
## [16]    1 1342    1 1537    1
```

Style Comparison

Beyond thematic content, we can analyze the *stylistic* fingerprints of the authors. Style is often defined not by unique nouns (content), but by the distribution of high-frequency function words (the, and, of, but).

The following code block prepares the data for Principal Component Analysis (PCA) by: 1. Isolating the top 150 most frequent words in the entire corpus. 2. Calculating the relative frequency of these words for each book. 3. Projecting this high-dimensional data onto a 2D plane to visualize stylistic clusters.

```
TOP_N = 150

# --- Prepare Style ---
# We go back to the raw books
style_dtm = books_clean %>%
  unnest_tokens(word, text) %>%

  # Count word usage per book
  count(title, word) %>%
  group_by(title) %>%
  mutate(total_words = sum(n)) %>%

  # Calculate Relative Frequency (Frequency per 1000 words)
  # Normalise for book length
  mutate(freq = (n / total_words) * 1000) %>%
  ungroup() %>%

  # Filter for the Top N Most Frequent Words
  group_by(word) %>%
  mutate(word_total = sum(n)) %>%
  ungroup() %>%
  filter(word %in% slice_max(unique(select(., word, word_total)), order_by = word_total, n = TOP_N)$word

  # Reshape to Matrix (Rows = Books, Cols = Words)
  select(title, word, freq) %>%
  pivot_wider(names_from = word, values_from = freq, values_fill = 0)
```

```
# --- PCA ---
# Convert to matrix
pca_matrix = as.matrix(style_dtm[, -1])
rownames(pca_matrix) = style_dtm$title

# Run PCA
pca_result = prcomp(pca_matrix, scale. = TRUE)

# --- Plot the Literary Map ---
# Extract coordinates
pca_coords = as.data.frame(pca_result$x)
pca_coords$title = rownames(pca_coords)

# Add Genre back in for coloring
pca_plot_data = pca_coords %>%
  left_join(unique(books_clean[, c("title", "genre")]), by = c("title" = "title"))

# Visualize
my_plot = ggplot(pca_plot_data, aes(x = PC1, y = PC2, color = genre, label = title)) +
  geom_point(size = 4, alpha = 0.8) +
  geom_text(vjust = 1.5, size = 3, check_overlap = FALSE) +
  labs(title = "Stylometric Map Corpus",
       subtitle = "Based on usage of top 150 function words (PCA)",
       x = "PC1 (Primary Stylistic Dimension)",
       y = "PC2 (Secondary Stylistic Dimension)") +
  theme_minimal()

ggsave("Stylometric Map.pdf",
       plot = my_plot,
       width = 36,
       height = 24,
       units = "cm")
```

The PCA plot reveals that Gothic Horror is a stylistically diverse genre. Some works cluster near Realism/Romance, while others approach Science Fiction. Notably, early Gothic works like The Castle of Otranto and The Mysteries of Udolpho appear distinct from later Victorian Gothic novels like Dracula.

Mary Shelley's works lie in a unique neighborhood, stylistically adjacent to early Science Fiction works by Jules Verne. This suggests a potential stylistic lineage, though definitive influence is difficult to prove from clustering alone.

Cosine Similarity Analysis PCA provides a useful 2D approximation, but it inherently loses information by compressing the dimensions. To get a more precise measure of similarity, we calculate the Cosine Similarity between the full PCA vectors of Shelley's works and every other book in the corpus.

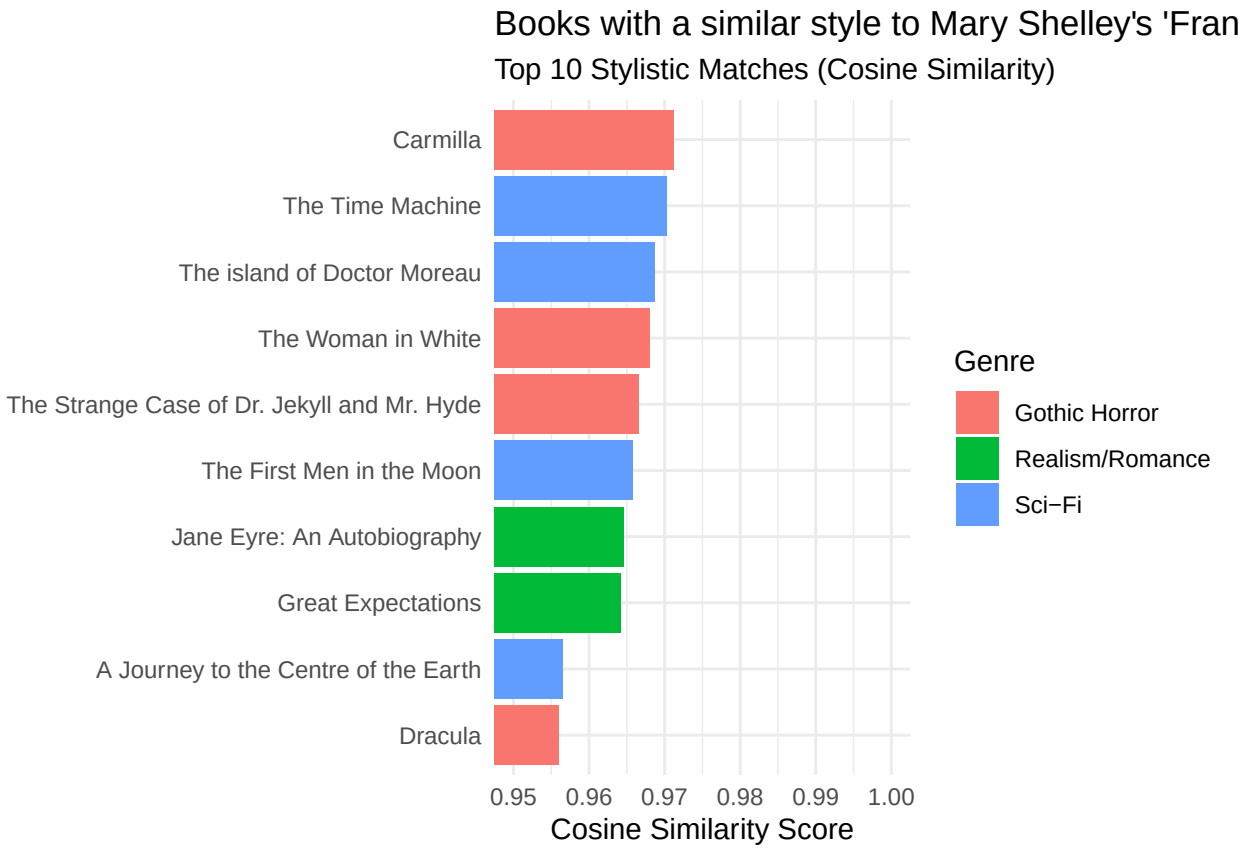
```
# --- Find Closest Books in Style ---
# Cosine Similarity Function
cosine_sim = function(a, b) {
  sum(a * b) / (sqrt(sum(a^2)) * sqrt(sum(b^2)))
}

# --- Frankenstein ---

target_vec = pca_matrix["Frankenstein; Or, The Modern Prometheus", ]
```

```r
similarities = data.frame(
  title = rownames(pca_matrix),
  similarity = apply(pca_matrix, 1, function(x) cosine_sim(target_vec, x))
)

# Get Top 10 Most Similar Books
top_similar = similarities %>%
  filter(title != "Frankenstein; Or, The Modern Prometheus") %>%
  arrange(desc(similarity)) %>%
  slice_head(n = 10) %>%
  left_join(unique(books_clean[, c("title", "genre")]), by = c("title" = "title"))

create_table(top_similar,
             caption = "Top 10 Books with a similar style to Mary Shelley's Frankenstein",
             col_labels=c("Book Title", "Cosine Similarity"))
```

Table 5: Top 10 Books with a similar style to Mary Shelley's Frankenstein

Book Title	Cosine Similarity	NA
Carmilla	0.971	Gothic Horror
The Time Machine	0.970	Sci-Fi
The island of Doctor Moreau	0.969	Sci-Fi
The Woman in White	0.968	Gothic Horror
The Strange Case of Dr. Jekyll and Mr. Hyde	0.967	Gothic Horror
The First Men in the Moon	0.966	Sci-Fi
Jane Eyre: An Autobiography	0.965	Realism/Romance
Great Expectations	0.964	Realism/Romance
A Journey to the Centre of the Earth	0.957	Sci-Fi
Dracula	0.956	Gothic Horror

```r
top_similar %>%
  ggplot(aes(x = reorder(title, similarity), y = similarity, fill = genre)) +
  geom_col() +
  coord_flip() +
  labs(
    title = "Books with a similar style to Mary Shelley's 'Frankenstein'",
    subtitle = "Top 10 Stylistic Matches (Cosine Similarity)",
    y = "Cosine Similarity Score",
    x = NULL,
    fill = "Genre"
  ) +
  theme_minimal() +
  scale_y_continuous(limits = c(0.95, 1.0), oob = scales::rescale_none)
```



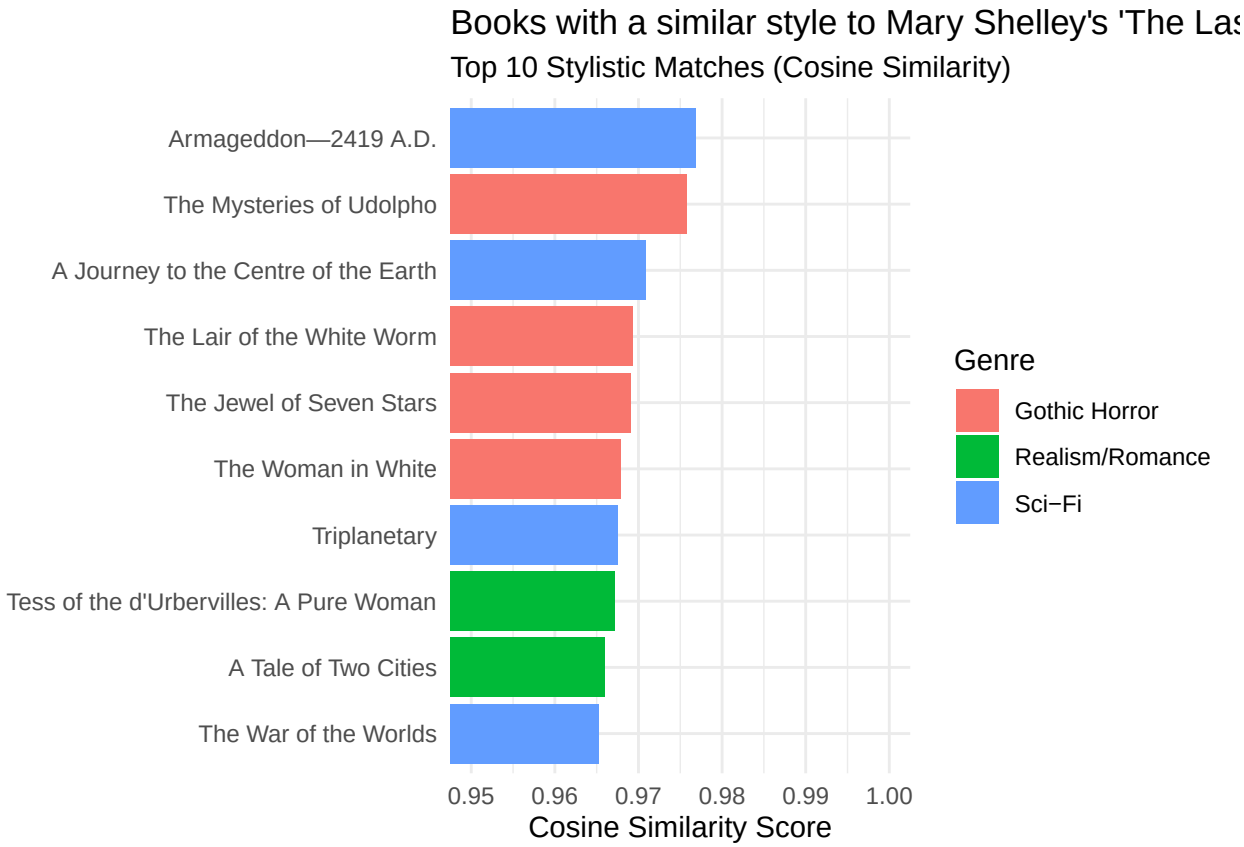

```
# --- The Last Man ---

target_vec = pca_matrix["The Last Man", ]

similarities = data.frame(
  title = rownames(pca_matrix),
  similarity = apply(pca_matrix, 1, function(x) cosine_sim(target_vec, x))
)

# Get Top 10 Most Similar Books
top_similar = similarities %>%
  filter(title != "The Last Man") %>%
  arrange(desc(similarity)) %>%
  slice_head(n = 10) %>%
  left_join(unique(books_clean[, c("title", "genre")]), by = c("title" = "title"))

create_table(top_similar,
             caption = "Top 10 Books with a similar style to Mary Shelley's 'The Last Man'",
             col_labels=c("Book Title", "Cosine Similarity"))
```

Table 6: Top 10 Books with a similar style to Mary Shelley's 'The Last Man'

Book Title	Cosine Similarity	NA
Armageddon—2419 A.D.	0.977	Sci-Fi
The Mysteries of Udolpho	0.976	Gothic Horror

Book Title	Cosine Similarity	NA
A Journey to the Centre of the Earth	0.971	Sci-Fi
The Lair of the White Worm	0.969	Gothic Horror
The Jewel of Seven Stars	0.969	Gothic Horror
The Woman in White	0.968	Gothic Horror
Triplanetary	0.967	Sci-Fi
Tess of the d'Urbervilles: A Pure Woman	0.967	Realism/Romance
A Tale of Two Cities	0.966	Realism/Romance
The War of the Worlds	0.965	Sci-Fi

```
top_similar %>%
  ggplot(aes(x = reorder(title, similarity), y = similarity, fill = genre)) +
  geom_col() +
  coord_flip() +
  labs(
    title = "Books with a similar style to Mary Shelley's 'The Last Man'",
    subtitle = "Top 10 Stylistic Matches (Cosine Similarity)",
    y = "Cosine Similarity Score",
    x = NULL,
    fill = "Genre"
  ) +
  theme_minimal() +
  scale_y_continuous(limits = c(0.95, 1.0), oob = scales::rescale_none)
```



This analysis indicates that Frankenstein sits on the stylistic border between Gothic Horror and Science

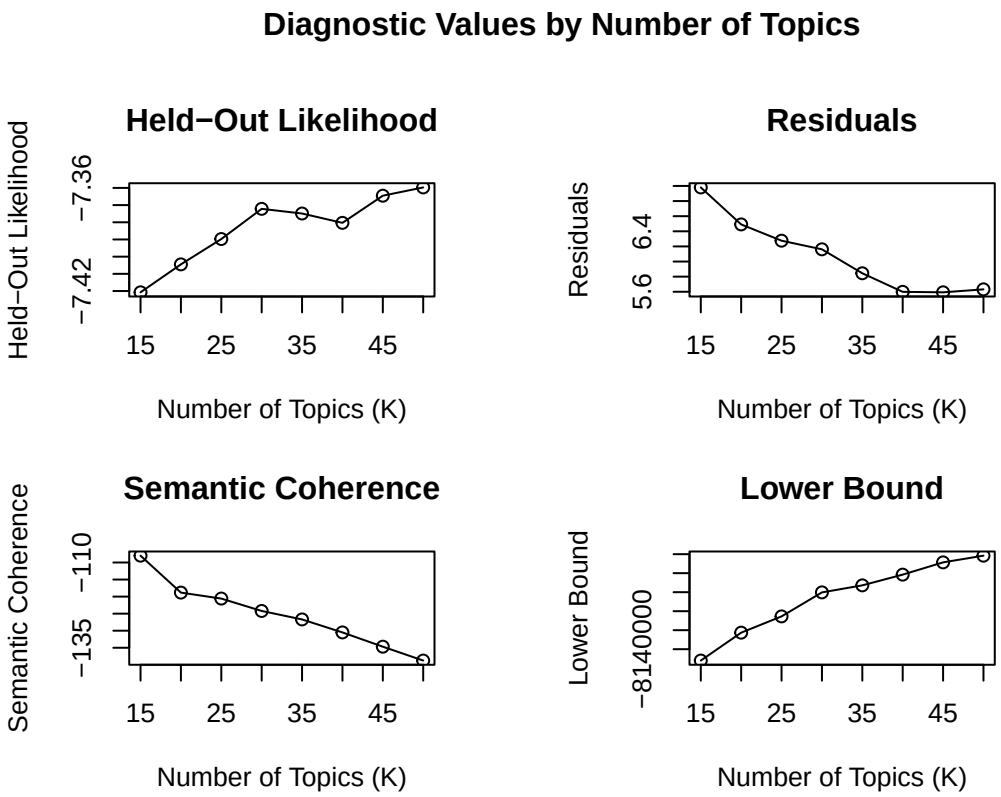
Fiction. If we were to use a K-Nearest Neighbors approach ($k = 5$), the classification would oscillate between the two genres depending on the specific value of k.Interestingly, The Last Man shares strong stylistic similarities with much later Science Fiction works, despite its early publication date. This supports the hypothesis that Shelley was not merely writing within the Romantic tradition, but was actively pioneering a new stylistic register that would later define speculative fiction.Having established the stylistic context, the next section applies Structural Topic Modeling to analyze the thematic content of the works.

Structural Topic Modelling

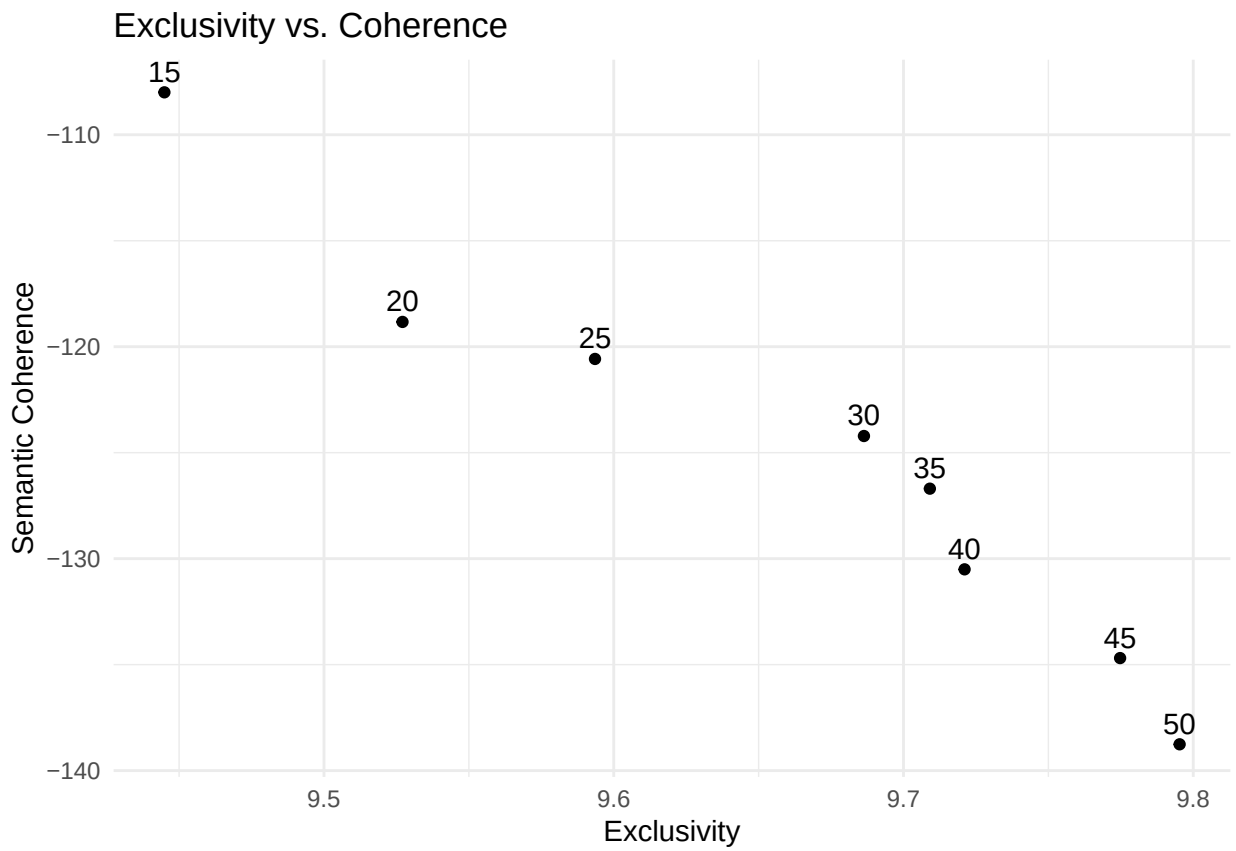
Number of Topics

As the corpus of text is very large, we would expect for there to be a large number of topics. Therefore, using `searchK` will aim to find the optimal number of topics for my analysis. First, we will do a wide search of $K \in \{15, 20, 25, 30, 35, 40, 45, 50\}$.

```
# --- Finding optimal number of topics --
# Set seed for reproducibility
set.seed(42)

# Use searchK to evaluate models with different numbers of topics
k_result = searchK(
  documents = docs,
  vocab = vocab,
  K = c(15,20,25,30,35,40,45,50),
  prevalence = ~ genre + era,
  data = meta,
  max.em.its = 150,
  init.type = "Spectral",
  verbose = FALSE)

plot(k_result)
```




```
# ---Plotting Coherence vs. Exclusivity ---
k_values = unlist(k_result$results$K)

coherence = sapply(k_result$results$semcoh, function(x) x[1])
exclusivity = sapply(k_result$results$exclus, function(x) x[1])

plot_data = data.frame(Topics = k_values, Coherence = coherence, Exclusivity = exclusivity)

ggplot(plot_data, aes(x = Exclusivity, y = Coherence, label = Topics)) +
  geom_point() +
  geom_text(vjust = -0.5) +
  labs(title = "Exclusivity vs. Coherence", x = "Exclusivity", y = "Semantic Coherence") +
  theme_minimal()
```



In order to get a balance of coherence and exclusivity, I will scale the these values, and construct a composite score for easier identification of the optimal number of topics. I will do this using the following

$$\text{Composite Score} = 0.5 \times \text{Coherence} + 0.5 \times \text{Exclusivity}$$

```
# --- Graphing Composite Score ---
rescale = function(x) {
  (x - min(x)) / (max(x) - min(x))
}

rescaled_coh = rescale(coherence)
rescaled_exc = rescale(exclusivity)
rescaled_data = data.frame(Topics = k_values, Rescaled_Coherence = rescaled_coh, Rescaled_Exclusivity =

create_table(rescaled_data, caption = "Rescaled Coherence and Exclusivity Scores")
```

Table 7: Rescaled Coherence and Exclusivity Scores

Topics	Rescaled_Coherence	Rescaled_Exclusivity
15.0	1.00	0
20.0	0.648	0.235
25.0	0.591	0.424
30.0	0.473	0.689
35.0	0.392	0.754
40.0	0.268	0.788

Topics	Rescaled_Coherence	Rescaled_Exclusivity
45.0	0.132	0.941
50.0	0	1.00

```
# Calculate Composite Score
composite_score = 0.5 * rescaled_coh + 0.5 * rescaled_exc
composite_data = data.frame(Topics = k_values, Composite_Score = composite_score)

create_table(composite_data, caption = "Composite Scores for Different Numbers of Topics")
```

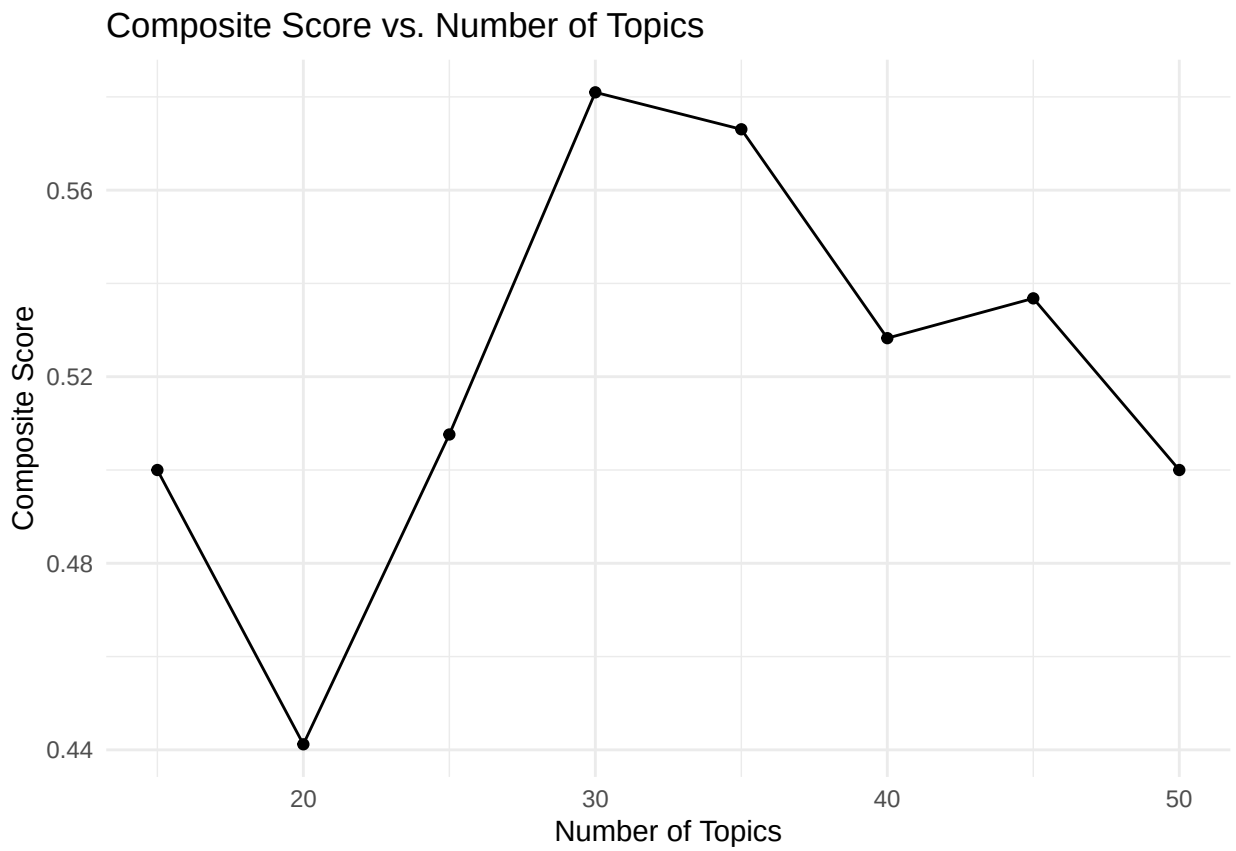
Table 8: Composite Scores for Different Numbers of Topics

Topics	Composite_Score
15.0	0.500
20.0	0.441
25.0	0.508
30.0	0.581
35.0	0.573
40.0	0.528
45.0	0.537
50.0	0.500

```
optimal_k = composite_data$Topics[which.max(composite_data$Composite_Score)]

cat("\nMaximum Composite Score at K =", optimal_k, "\n")
```

```
##
## Maximum Composite Score at K = 30
```

```
# Plot Composite Score vs. Number of Topics
ggplot(composite_data, aes(x = Topics, y = Composite_Score)) +
  geom_line() +
  geom_point() +
  labs(title = "Composite Score vs. Number of Topics", x = "Number of Topics", y = "Composite Score") +
  theme_minimal()
```



This gives an optimal $K \approx 35$. I initially wanted to hone in on the values around 35 to find a better value of K, however, I am not sure of the impact this would have, especially with the time taken to run these searches. Also, in this exploration, a mathematically optimal value of K may not be what is best for classifying these works, and we may need more topics to see the nuances between key themes.

STM Model

From the previous section, I will fit an STM model with $K = 35$ topics, and using `genre` and `era` as prevalence covariates. We will now run our model to find the 35 topics for our corpus.

The top words generated for each topic will assist in classifying the topics into specific genres or themes for analysis.

```
# --- Fit STM ---
set.seed(42)
stm_model = stm(
  documents = docs,
  vocab = vocab,
  K = optimal_k, # Number of topics
  prevalence = ~ genre + era,
  max.em.its = 150,
  data = meta,
  init.type = "Spectral",
  verbose = FALSE
)

# Number of Successful Runs
```

```r
length(stm_model$runout)
```

```
## [1] 0
```

```r
# Top 7 words for each topic
labelTopics(stm_model, n=7)
```

```
## Topic 1 Top Words:
##       Highest Prob: queen, make, now, one, week, trade, can
##       FREX: salariki, rycke, trade, trader, cargo, queen, cabin
##       Lift: salariki, clansman, groft, kallee, rycke, ryckes, sargolian
##       Score: rycke, emphatic, salariki, queen, cargo, medic, trade
## Topic 2 Top Words:
##       Highest Prob: much, mrs, can, miss, good, every, give
##       FREX: harriet, mrs, weston, acquaintance, miss, colonel, frank
##       Lift: chuse, dashwoods, eltons, jennings's, netherfield, palmer, randalls
##       Score: mrs, mann, miss, weston, harriet, sister, colonel
## Topic 3 Top Words:
##       Highest Prob: love, heart, word, say, dear, can, tear
##       FREX: god, thou, pity, thy, weep, forgive, alas
##       Lift: alfonso, bianca, dost, hast, frederic, hippolita, highness
##       Score: love, thou, thy, dost, dear, god, heart
## Topic 4 Top Words:
##       Highest Prob: foot, one, side, upon, top, can, stone
##       FREX: top, tunnel, narrow, edge, rope, plateau, cave
##       Lift: tunnel, mooncalves, plateau, basalt, hatchet, bale, protrude
##       Score: tunnel, cavor, rock, top, foot, huge, enormous
## Topic 5 Top Words:
##       Highest Prob: one, church, much, paris, king, saint, master
##       FREX: archdeacon, notre, jehan, gypsy, jacques, saint, dame
##       Lift: aux, châtelet, germain, mahiette, notre, pillory, trouillefou
##       Score: porte, saint, phœbus, archdeacon, notre, gypsy, jehan
## Topic 6 Top Words:
##       Highest Prob: work, eat, little, country, much, day, spend
##       FREX: pleasant, milk, enjoy, tea, spend, farm, farmer
##       Lift: jail, crick, dairy, ewe, talbothays, fare, hay
##       Score: jail, pleasant, tea, enjoy, eat, spend, milk
## Topic 7 Top Words:
##       Highest Prob: thing, much, time, seem, can, think, man
##       FREX: possibility, brain, scientific, impression, puzzle, thing, experience
##       Lift: filby, psychologist, sequence, lympne, critic, cavorite, journalist
##       Score: filby, thing, cavor, scientific, world, mind, moon
## Topic 8 Top Words:
##       Highest Prob: will, may, must, can, shall, good, know
##       FREX: will, may, shall, morrow, must, trouble, safe
##       Lift: later, phonograph, westenra, morris, diary, quincey, whitby
##       Score: will, shall, may, must, phonograph, lucy, morrow
## Topic 9 Top Words:
##       Highest Prob: life, feel, much, power, nature, become, mind
##       FREX: self, nature, passion, evil, sympathy, possess, feeling
##       Lift: positive, aspire, attainment, deceitful, degradation, antipathy, ardently
##       Score: positive, power, life, love, nature, happiness, feel
## Topic 10 Top Words:
##       Highest Prob: much, now, madame, appear, can, perceive, count
```

```
##       FREX: madame, annette, endeavour, castle, countenance, conduct, ludovico
##       Lift: cheron, annette's, bonnac, clairval, orsino, verezzi, villefort
##       Score: annette, madame, clairval, ludovico, château, ma'amselle, cheron
## Topic 11 Top Words:
##       Highest Prob: uncle, professor, earth, can, much, great, now
##       FREX: globe, uncle, raft, professor, interior, phenomenon, volcano
##       Lift: globe, arne, gretchen, saknussemm, sneffels, raft, icelander
##       Score: globe, uncle, professor, raft, earth, lava, sneffels
## Topic 12 Top Words:
##       Highest Prob: door, open, room, window, house, light, step
##       FREX: door, open, window, step, lamp, room, lock
##       Lift: rector, storey, stair, door, latch, lamp, verandah
##       Score: door, room, window, open, rector, house, street
## Topic 13 Top Words:
##       Highest Prob: get, old, horse, take, gentleman, boy, drink
##       FREX: joe, herbert, horse, coach, drink, gentleman, biddy
##       Lift: joe's, orlick, wopsle, jerry, biddy, joe, pumblechook
##       Score: joe, orlick, horse, gentleman, herbert, get, old
## Topic 14 Top Words:
##       Highest Prob: say, man, get, head, bite, pocket, stare
##       FREX: kemp, invisible, marvel, wemmick, bite, jew, bottle
##       Lift: jaffers, kemp, mariner, marvel's, adye, whiskey, henfrey
##       Score: mariner, kemp, say, 've, invisible, 're, marvel
## Topic 15 Top Words:
##       Highest Prob: paper, box, note, read, ghost, sing, play
##       FREX: opus, richard, ghost, paper, manager, moncharmin, box
##       Lift: poligny, buquet, commissary, eriks, foyer, mercier, mifroid
##       Score: box, opus, paper, tier, ghost, richard, moncharmin
## Topic 16 Top Words:
##       Highest Prob: dress, white, wear, hair, clothe, black, fine
##       FREX: wear, gold, dress, fashion, clothe, paint, colour
##       Lift: geneviève, satin, silk, flanders, plait, velvet, braid
##       Score: dress, geneviève, white, hair, wear, colour, gold
## Topic 17 Top Words:
##       Highest Prob: old, year, man, live, child, young, mother
##       FREX: year, live, age, ago, child, month, school
##       Lift: santa, edwin, magnate, hoo, granser, vesta, year
##       Score: year, child, old, mother, live, santa, father
## Topic 18 Top Words:
##       Highest Prob: look, hand, eye, face, see, turn, head
##       FREX: face, eye, look, lip, hand, pale, kiss
##       Lift: skate, forehead, eyelid, cheek, absently, face, flush
##       Score: face, eye, look, skate, hand, lip, kiss
## Topic 19 Top Words:
##       Highest Prob: wind, sun, air, wood, light, scene, mountain
##       FREX: wind, sun, breeze, mountain, wood, shade, snow
##       Lift: xxvi, cypress, verdant, o'er, garonne, mont, glen
##       Score: mountain, sun, xxvi, wind, tree, cloud, wood
## Topic 20 Top Words:
##       Highest Prob: country, england, among, first, find, history, every
##       FREX: history, england, labour, abide, number, protector, greece
##       Lift: publication, ryland, versailles, turk, visitation, famine, candidate
##       Score: publication, england, plague, country, history, protector, native
## Topic 21 Top Words:
```

```
##       Highest Prob: fall, upon, dead, cry, moment, arm, strike
##       FREX: blood, dead, horrible, horror, seize, burst, beat
##       Lift: cringe, parch, horrible, spasm, wrench, blood, moan
##       Score: cringe, blood, cry, dead, fall, horror, arm
## Topic 22 Top Words:
##       Highest Prob: sea, captain, water, two, boat, hundred, mile
##       FREX: island, boat, canadian, coast, sea, fish, depth
##       Lift: billion, cetacean, narwhal, poulps, suez, zoophytes, harpoon
##       Score: sea, captain, billion, island, ocean, water, fish
## Topic 23 Top Words:
##       Highest Prob: day, night, time, hour, return, come, morning
##       FREX: night, morning, sleep, bed, hour, carriage, next
##       Lift: mlle, soundly, morning, asleep, carriage, bed, night
##       Score: night, sleep, morning, hour, day, mlle, carriage
## Topic 24 Top Words:
##       Highest Prob: like, always, good, much, woman, one, people
##       FREX: alexandrovitch, always, sergey, arkadyevitch, ivanovitch, baby, talk
##       Lift: mihalovna, moadine, agafea, koznishev, lidia, ivanovna, sergey
##       Score: mihalovna, love, alexandrovitch, arkadyevitch, woman, sergey, always
## Topic 25 Top Words:
##       Highest Prob: come, see, run, fire, across, black, like
##       FREX: smoke, martian, bush, darkness, across, blaze, glare
##       Lift: putney, snipe, horsell, chobham, londonward, leatherhead, woking
##       Score: snipe, martian, tree, smoke, black, sky, bush
## Topic 26 Top Words:
##       Highest Prob: get, can, dont, come, back, around, right
##       FREX: cant, claw, hes, well, dont, ive, thats
##       Lift: lab, bunker, duffe, reinharts, vidscreen, centaurus, icarus
##       Score: dont, vidsender, cant, thats, bunker, youre, get
## Topic 27 Top Words:
##       Highest Prob: sir, letter, miss, write, lady, house, count
##       FREX: sir, fairlie, letter, doctor, anne, limmeridge, inquiry
##       Lift: kyrle, dawson, gilmore, gilmore's, merriman, michelson, fairlie's
##       Score: sir, letter, merriman, fairlie, miss, anne, write
## Topic 28 Top Words:
##       Highest Prob: say, good, know, tell, think, come, ask
##       FREX: yes, don't, tell, ask, answer, say, afraid
##       Lift: laban, ellen, liddy, shan't, tonight, ma', yes
##       Score: say, don't, 'll, yes, tell, ask, know
## Topic 29 Top Words:
##       Highest Prob: ship, space, beam, can, power, upon, force
##       FREX: nevian, wilma, han, rocket, boss, ultra, rodebush
##       Lift: acceleration, disintegrator, inertron, nevia, planetoid, radio, repellor
##       Score: ship, projectoscope, nevian, rodebush, nevians, rocket, space
## Topic 30 Top Words:
##       Highest Prob: one, much, make, little, take, see, can
##       FREX: make, last, one, end, little, voyage, long
##       Lift: voyage, homeward, duration, bold, haste, apparently, relapse
##       Score: voyage, one, much, make, little, last, take
```

Topic Summary

Considering both the most common words,and the most exlusive can assist in our interpertion of each topic. If some words only appear in a topic, they much be significant to its interpretation.

```r
label_output = labelTopics(stm_model, n = 5)

topic_labels_df = data.frame(
  Topic_ID = 1:nrow(label_output$prob),
  Top_Words_Prob = apply(label_output$prob, 1, paste, collapse = ", "),
  Unique_Words_FREX = apply(label_output$frex, 1, paste, collapse = ", ")
)

knitr::kable(topic_labels_df,
      # format = "latex",
      booktabs = TRUE,
      longtable = TRUE,
      caption = "Topic Definitions: Probability vs. Exclusivity",
      col.names = c("Topic ID", "Highest Probability Words", "Most Exclusive (FREX) Words"),
      row.names = FALSE)
```

Table 9: Topic Definitions: Probability vs. Exclusivity

Topic ID	Highest Probability Words	Most Exclusive (FREX) Words
1	queen, make, now, one, week	salariki, rycke, trade, trader, cargo
2	much, mrs, can, miss, good	harriet, mrs, weston, acquaintance, miss
3	love, heart, word, say, dear	god, thou, pity, thy, weep
4	foot, one, side, upon, top	top, tunnel, narrow, edge, rope
5	one, church, much, paris, king	archdeacon, notre, jehan, gypsy, jacques
6	work, eat, little, country, much	pleasant, milk, enjoy, tea, spend
7	thing, much, time, seem, can	possibility, brain, scientific, impression, puzzle
8	will, may, must, can, shall	will, may, shall, morrow, must
9	life, feel, much, power, nature	self, nature, passion, evil, sympathy
10	much, now, madame, appear, can	madame, annette, endeavour, castle, countenance
11	uncle, professor, earth, can, much	globe, uncle, raft, professor, interior
12	door, open, room, window, house	door, open, window, step, lamp
13	get, old, horse, take, gentleman	joe, herbert, horse, coach, drink
14	say, man, get, head, bite	kemp, invisible, marvel, wemmick, bite
15	paper, box, note, read, ghost	opus, richard, ghost, paper, manager
16	dress, white, wear, hair, clothe	wear, gold, dress, fashion, clothe
17	old, year, man, live, child	year, live, age, ago, child
18	look, hand, eye, face, see	face, eye, look, lip, hand
19	wind, sun, air, wood, light	wind, sun, breeze, mountain, wood
20	country, england, among, first, find	history, england, labour, abide, number
21	fall, upon, dead, cry, moment	blood, dead, horrible, horror, seize
22	sea, captain, water, two, boat	island, boat, canadian, coast, sea
23	day, night, time, hour, return	night, morning, sleep, bed, hour
24	like, always, good, much, woman	alexandrovitch, always, sergey, arkadyevitch, ivanovitch
25	come, see, run, fire, across	smoke, martian, bush, darkness, across
26	get, can, dont, come, back	cant, claw, hes, well, dont
27	sir, letter, miss, write, lady	sir, fairlie, letter, doctor, anne
28	say, good, know, tell, think	yes, don't, tell, ask, answer
29	ship, space, beam, can, power	nevian, wilma, han, rocket, boss
30	one, much, make, little, take	make, last, one, end, little

This considers the most common topics across the whole corpus.

```r
# Top words for each topic
td_beta = tidy(stm_model, matrix = "beta")

# For each document, proportion of each topic
td_gamma = tidy(stm_model, matrix = "gamma")

# Get top terms for each topic
top_terms = td_beta %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%  # Get the top 10 words per topic
  select(topic, term) %>%
  summarise(terms = list(term)) %>% # Collapse them into a list
  mutate(terms = map_chr(terms, paste, collapse = ", ")) # Paste them: "ship, sea, captain"

# Calculate average topic prevalence across all documents
gamma_terms = td_gamma %>%
  group_by(topic) %>%
  summarise(gamma = mean(gamma)) %>% # Calculate average prevalence
  arrange(desc(gamma)) %>%
  left_join(top_terms, by = "topic") %>% # Attach the word labels
  mutate(topic = paste0("Topic ", topic),
         topic = reorder(topic, gamma)) # Sort bars by size

# Plot the Top Topics by Prevalence
top_topics = gamma_terms %>%
  slice_max(gamma, n = 35) %>%
  ggplot(aes(topic, gamma, label = terms, fill = topic)) +
  geom_col(show.legend = FALSE) +
  geom_text(hjust = 0, nudge_y = 0.0005, size = 3) + # Print the words next to bars
  coord_flip() + # Make it horizontal
  scale_y_continuous(expand = c(0,0), limits = c(0, 0.1)) + # Fix axis size
  labs(x = NULL, y = "Topic Prevalence",
       title = "Topics by Prevalence",
       subtitle = "With top words contributing to each topic")

ggsave("Top Topics.pdf",
       plot = top_topics,
       width = 18, # Adjust as needed
       height = 12, # Adjust as needed
       units = "cm")
```

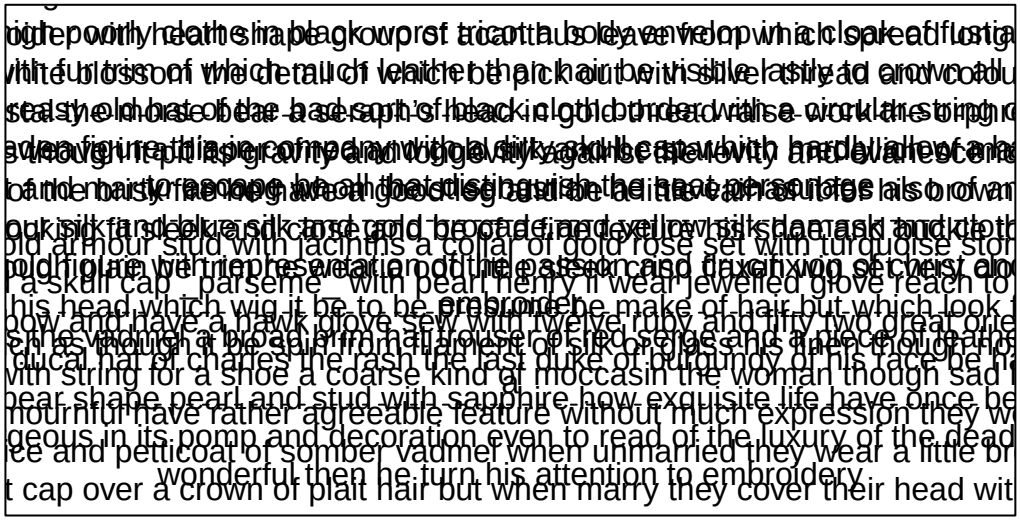
Topic Examples

In order to allow us to get a greater insight into each topic, we will consider the top 5 examples of chunks of text that relate to these topics. This should help us determine what themes these topics are portraying, and to check our interpretation of them.

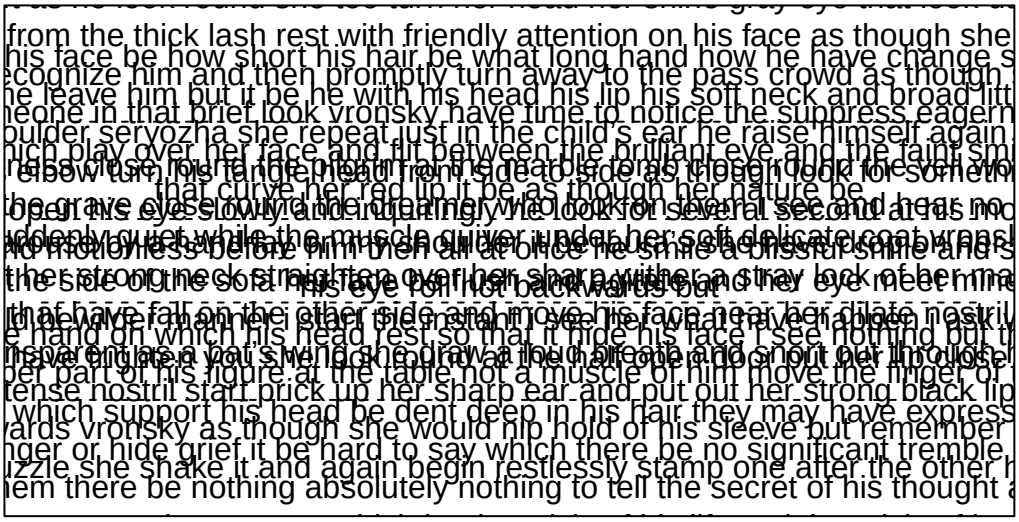
```r
# --- Qualitative Audit ---
for (t in c(16,18)) {
  thoughts = findThoughts(
    stm_model,
    texts = balanced_df$text_segment,
    topics = t,
    n = 5
```

```
  )$docs[[1]]

  title = paste("Topic", t, "Examples")
  plotQuote(thoughts, width = 80, main = title)
}
```

Topic 16 Examples



Topic 18 Examples



Topic Visualisation and Analysis

Now, the issue we come to now, is that we need to make some sense of these 'Topics' created by my STM model. The most frequent words, and the FREX (most unique) words will give some guide towards what these topics are, and the examples from each topic will also allow us to see the context of these topics. However, with 35 topics, and with the possible nuances of each of them, this alone is not enough to fully understand their meaning. Therefore I need a wide range of different tools and visualisations to be able to interpret them for comparison. Therefore my final analysis of these topics will be at the end of this section, having looked through a range of visualisation. I will be considering

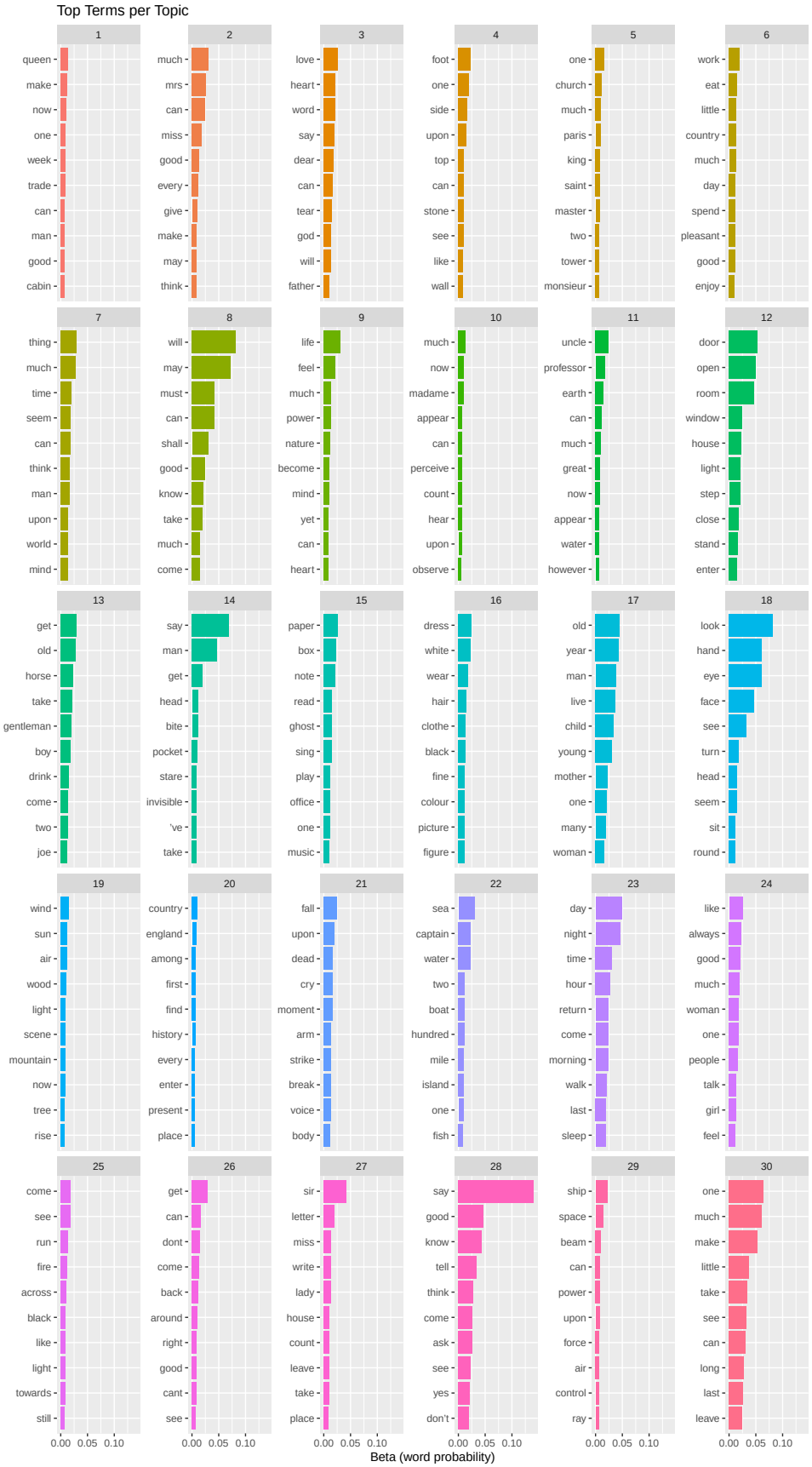
- **Top Terms per Topic**: To help visualise the most frequent words for each topic
- **Word Clouds**: Another way to visualise the most frequent words for each topic
- **Control Checks**: To see which topics align with specific texts to help compare topics, and also check that our topics, and our generalisation of them, do actually make sense.
- **Topic Prevalence by Genre**: To see which topics are more prevalent in which genres, and see if this aligns with our interpretations of the topics.
- **Topic Correlations**: To see which topics are related to each other, and see if this aligns with our interpretations of the topics, and to help group similar topics together.

Only all of this together will give us a full understanding and help to truly classify these topics and Shelley's works.

Once we have fully dissected our topics, we can then do **Topic Comparison** to compare specific topics between Shelley and different genres to see which she most aligns with.

Top Terms per Topic

```
# Top terms per topic for visualization
# (From Julia Stilge's example)
beta_td = tidy(stm_model)

top_terms = beta_td %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)


top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free_y") +
  scale_y_reordered() +
  labs(title = "Top Terms per Topic",
       x = "Beta (word probability)",
       y = NULL)
```


Top Terms per Topic
1
queen
make
now
one
week
trade
can
man
good
cabin
2
much
mrs
can
miss
good
every
give
make
may
think
3
love
heart
word
say
dear
can
tear
god
will
father
4
foot
one
side
upon
top
can
stone
see
like
wall
5
one
church
much
paris
king
saint
master
two
tower
monsieur
6
work
eat
little
country
much
day
spend
pleasant
good
enjoy
7
thing
much
time
seem
can
think
man
upon
world
mind
8
will
may
must
can
shall
good
know
take
much
come
9
life
feel
much
power
nature
become
mind
yet
can
heart
10
much
now
madame
appear
can
perceive
count
hear
upon
observe
11
uncle
professor
earth
can
much
great
now
appear
water
however
12
door
open
room
window
house
light
step
close
stand
enter
13
get
old
horse
take
gentleman
boy
drink
come
two
joe
14
say
man
get
head
bite
pocket
stare
invisible
've
take
15
paper
box
note
read
ghost
sing
play
office
one
music
16
dress
white
wear
hair
clothe
black
fine
colour
picture
figure
17
old
year
man
live
child
young
mother
one
many
woman
18
look
hand
eye
face
see
turn
head
seem
sit
round
19
wind
sun
air
wood
light
scene
mountain
now
tree
rise
20
country
england
among
first
find
history
every
enter
present
place
21
fall
upon
dead
cry
moment
arm
strike
break
voice
body
22
sea
captain
water
two
boat
hundred
mile
island
one
fish
23
day
night
time
hour
return
come
morning
walk
last
sleep
24
like
always
good
much
woman
one
people
talk
girl
feel
25
come
see
run
fire
across
black
like
light
towards
still
26
get
can
dont
come
back
around
right
good
cant
see
27
sir
letter
miss
write
lady
house
count
leave
take
place
28
say
good
know
tell
think
come
ask
see
yes
don't
29
ship
space
beam
can
power
upon
force
air
control
ray
30
one
much
make
little
take
see
can
long
last
leave
0.00 0.05 0.10
Beta (word probability)

Word Clouds

```r
library(RColorBrewer)

# choose a base palette with 3 distinct hues (for 3 topics)
base_cols = brewer.pal(3, "Dark2")

# Word clouds for each topic
par(mfrow = c(9, 4), mar = c(0,0,1,0))
for (k in 1:optimal_k) {
  cloud(stm_model,
        topic = k,
        max.words = 25,
        scale = c(1.8, 0.4),
        colors = brewer.pal(8, "Dark2"))

  # add title after the cloud plot
  mtext(paste("Topic", k), line = -0.5, cex = 1.2, font = 2)
}
par(mfrow = c(1, 1))
```

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Topic 6

Topic 7

Topic 8

Topic 9

Topic 10

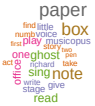
Topic 11

Topic 12

Topic 13

Topic 14

Topic 15



Topic 16

Topic 17

Topic 18

Topic 19

Topic 20

Topic 21

Topic 22

Topic 23

Topic 24

Topic 25

Topic 26

Topic 27

Topic 28

Topic 29

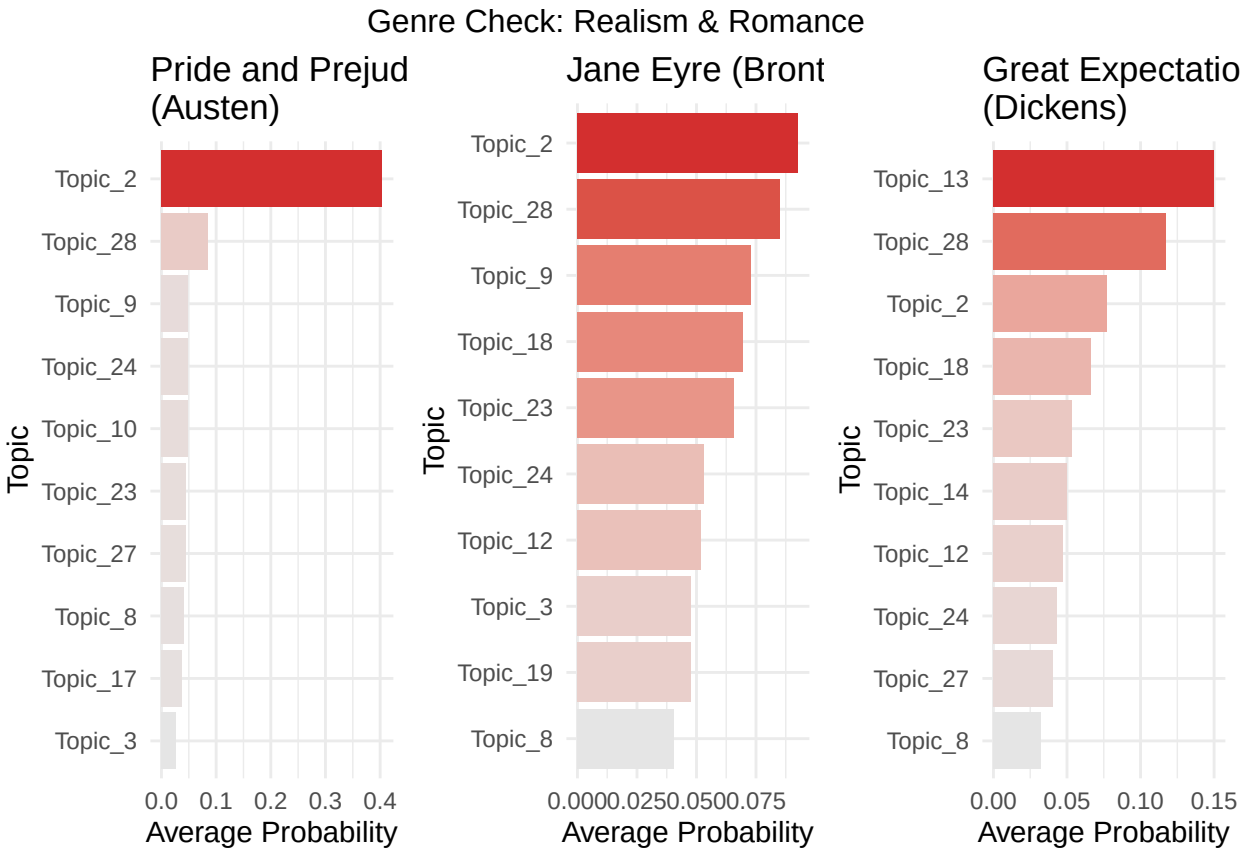
Topic 30

Control Checks

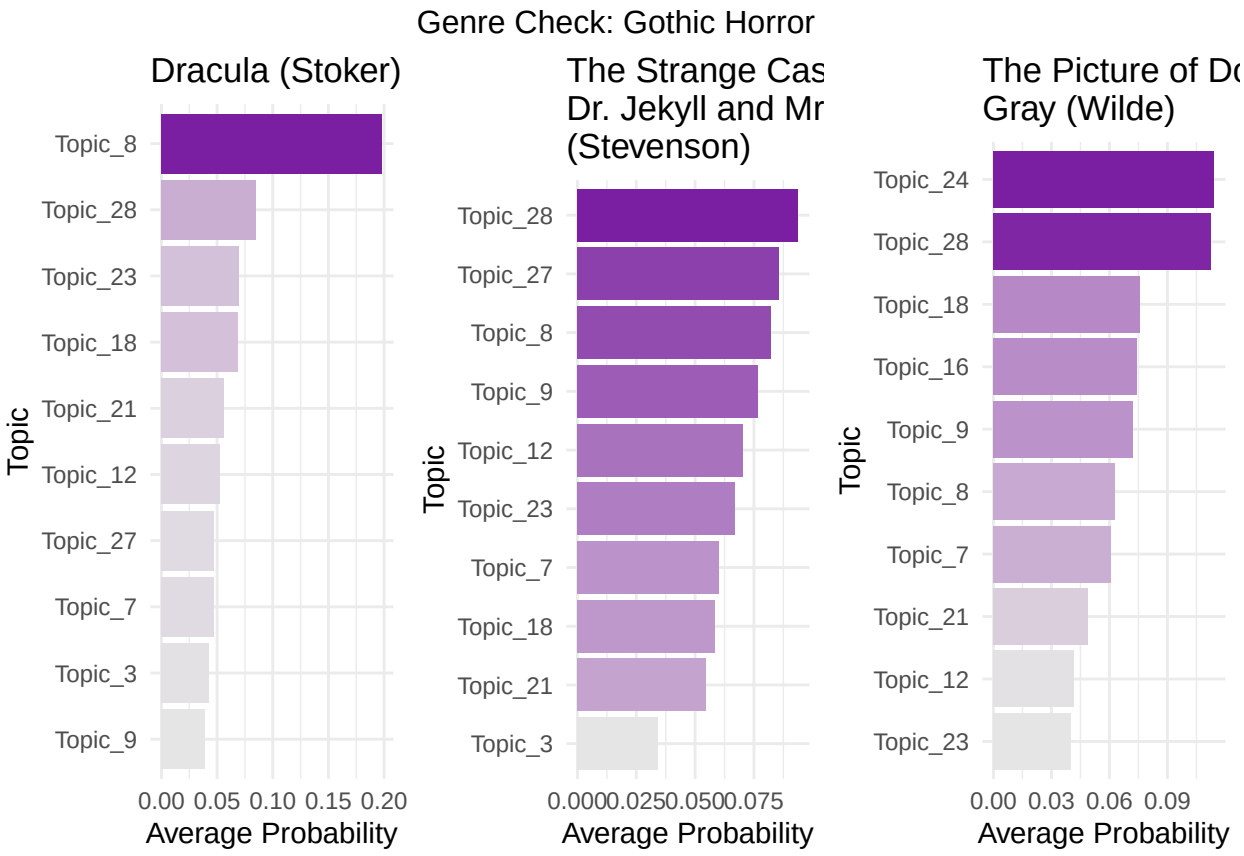
To ensure that our topics actually make sense in our context, as well as gain a deeper insight into each topic, we will check back to our novels, and make sure see the topics that are prevalent in some example of books. We expect to see to *Romance* themed topics (such as domestic life, romance, family) in *Pride and Prejudice*, *Gothic Horror* topics (such as supernatural, fear, death) in *Dracula*, and *Science Fiction* topics (such as technology, exploration, future) in *The Time Machine*. We will check this with a bar chart of prevalence of each topic, and see if it matches our previous interpretation of these topics.

```
# --- Topic Prevalence Check ---
topic_props = as.data.frame(stm_model$theta)
colnames(topic_props) = paste0("Topic_", 1:ncol(topic_props))

# Add metadata
meta_aligned = balanced_df
if(length(out$docs.removed) > 0) {
  meta_aligned = meta_aligned[-out$docs.removed, ]
}

# Combine them
doc_topic_df = cbind(meta_aligned, topic_props)

# Define colors for genres
genre_colors = c(
  "Realism/Romance" = "#D32F2F",  # Deep Red
  "Gothic Horror"   = "#7B1FA2",  # Deep Purple
  "Sci-Fi"          = "#1976D2",  # Strong Blue
  "Shelley"         = "#FBC02D"  # Gold/Dark Yellow (Readable)
)

# Control Check Function
plot_book_topics = function(book_title_pattern, color_tone = "steelblue") {

  # Filter for the specific book
  book_data = doc_topic_df %>%
    filter(str_detect(title, book_title_pattern)) %>%
    select(starts_with("Topic_")) %>%
    colMeans() %>%
    as.data.frame() %>%
    rename(Prevalence = ".") %>%
    mutate(Topic = rownames(.)) %>%
    arrange(desc(Prevalence)) %>%
    slice_head(n = 10)

  # Plot
  ggplot(book_data, aes(x = reorder(Topic, Prevalence), y = Prevalence, fill = Prevalence)) +
    geom_col(show.legend = FALSE) +
    coord_flip() +
    scale_fill_gradient(low = "grey90", high = color_tone) +
    labs(title = paste("Top Topics in:", book_title_pattern),
         y = "Average Probability", x = "Topic") +
    theme_minimal()
}


wrap_title = function(text) {
```

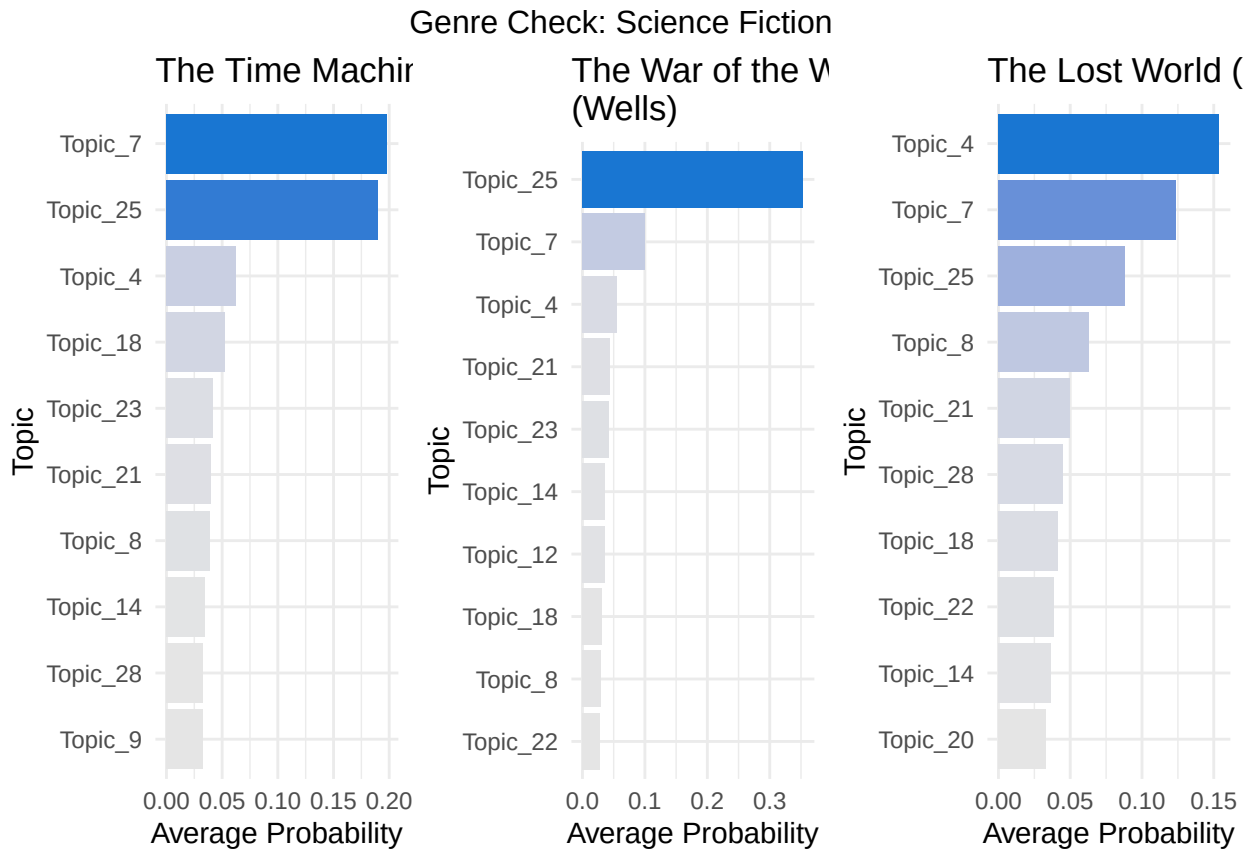
```
  str_wrap(text, width = 25) # Break line every 25 chars
}

# --- Romance ---
c_rom = genre_colors["Realism/Romance"]
p1 = plot_book_topics("Pride and Prejudice", c_rom) + labs(title=wrap_title("Pride and Prejudice (Auste
p2 = plot_book_topics("Jane Eyre", c_rom) + labs(title=wrap_title("Jane Eyre (Bronte)"))
p3 = plot_book_topics("Great Expectations", c_rom) + labs(title=wrap_title("Great Expectations (Dickens

romance_grid = grid.arrange(p1, p2, p3, ncol = 3, top = "Genre Check: Realism & Romance")
```



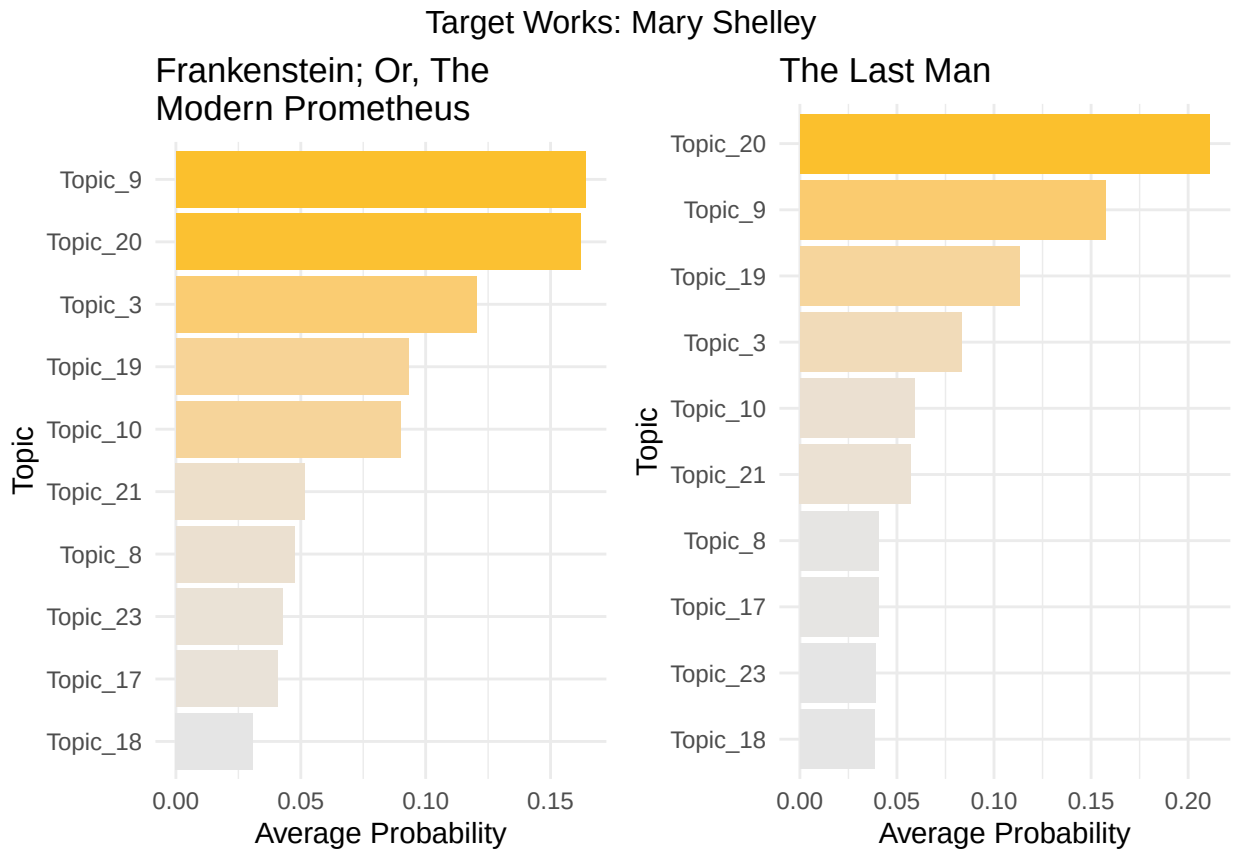

```
ggsave("Romance Check.pdf",
       plot = romance_grid,
       width = 36,
       height = 12,
       units = "cm")

# --- Gothic Horror ---
c_goth = genre_colors["Gothic Horror"]
p1 = plot_book_topics("Dracula", c_goth) + labs(title=wrap_title("Dracula (Stoker)"))
p2 = plot_book_topics("Dr. Jekyll", c_goth) + labs(title=wrap_title("The Strange Case of Dr. Jekyll and
p3 = plot_book_topics("Dorian Gray", c_goth) + labs(title=wrap_title("The Picture of Dorian Gray (Wilde

gothic_grid = grid.arrange(p1, p2, p3, ncol = 3, top = "Genre Check: Gothic Horror")
```




```
ggsave("Gothic Check.pdf",
       plot = gothic_grid,
       width = 36,
       height = 12,
       units = "cm")

# --- Science Fiction ---
c_sci = genre_colors["Sci-Fi"]
p1 = plot_book_topics("Time Machine", c_sci) + labs(title=wrap_title("The Time Machine (Wells)"))
p2 = plot_book_topics("War of the Worlds", c_sci) + labs(title=wrap_title("The War of the Worlds (Wells)
p3 = plot_book_topics("Lost World",c_sci) + labs(title=wrap_title("The Lost World (Doyle)"))

scifi_grid = grid.arrange(p1, p2, p3, ncol = 3, top = "Genre Check: Science Fiction")
```




```
ggsave("SciFi Check.pdf",
       plot = scifi_grid,
       width = 36,
       height = 12,
       units = "cm")

# --- Shelley ---
c_shel = genre_colors["Shelley"]
p1 = plot_book_topics("Frankenstein", c_shel) + labs(title=wrap_title("Frankenstein; Or, The Modern Pro
p2 = plot_book_topics("Last Man", c_shel) + labs(title=wrap_title("The Last Man"))

shelley_grid = grid.arrange(p1, p2, ncol = 2, top = "Target Works: Mary Shelley")
```




```
ggsave("Shelley Check.pdf",
       plot = shelley_grid,
       width = 36,
       height = 12,
       units = "cm")
```

Based on all of these representations, and the most frequent and exclusive words in each topic, we can highlight what idea each topic is protraying, as well as assosiate them with a genre. However, to further aid this, it may be useful to see if these topics are related to each other, therefore we will next consider topic correlation.

Topic Correlation

Themes that are quite common in a lot of these books, may be connected, to mulitple other topics. Therefore, we will consider the correlation between topics to see if any of them stand out as unique, or to see if we can cluster some topics together based on their correlations.

The Networks constructed here can be hard to read an interpret, so the following was done to improve this - Change cutoff for correlation to 0.1 to reduce number of edges - Topics were colour coded to easily see clusters - Shelley's topics have a yellow boarder, to see overlap with other genres - Any topics in multiple genres were colour blended to show this crossover - A table was created to show only correlations over 0.2 to check for the strongest relationships.

For adapting this model, it is key to change these topics based on earlier analysis of the topics. These have been changed from the report as that was generated on an earlier iteration of the model. The remaining code has been updated to fit within this data, and from interpretations of the previous prevalence bar charts.

This could be automatically linked by outputting the above code as dataframe instead and defining topics.

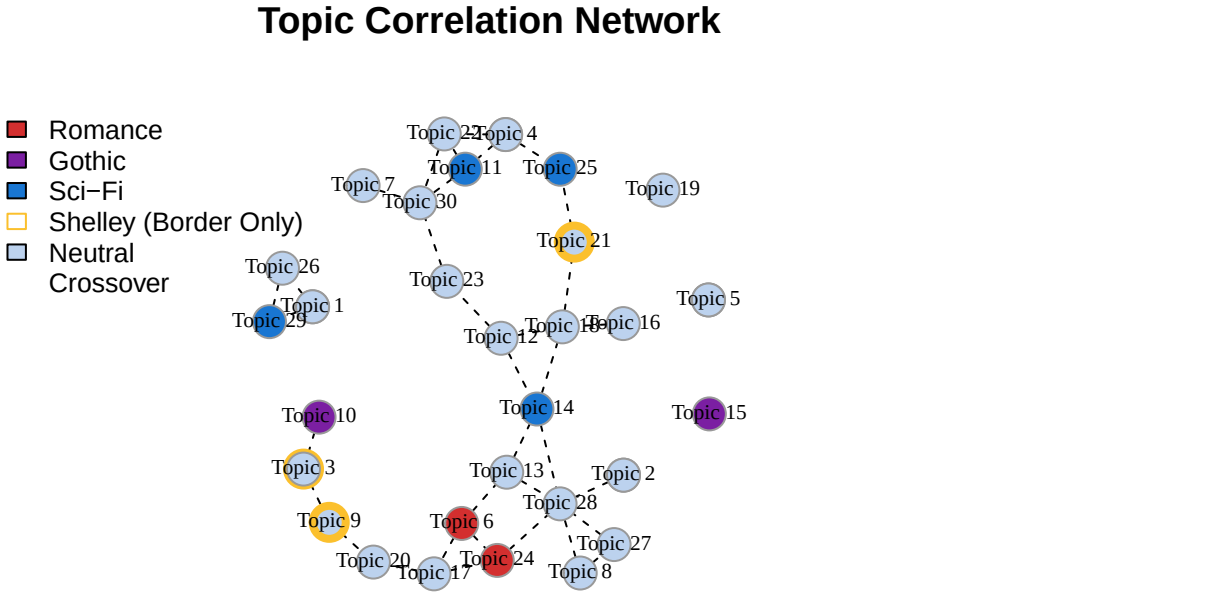
However, as we are only using selected texts, not the whole corpus, and considering qualitatively what each topic would relate to, we have stuck with manual updating.

```
# --- Topic Colour coding ---
# Romance (Red)
romance_topics = c(2,6,24)
# Gothic (Purple)
gothic_topics = c(10,15,5)
# Sci-Fi (Blue)
scifi_topics = c(25,29,11,14)
# Shelley (Yellow)
shelley_topics = c(9,21,3)

# Base colours
cols = c(
  Romance = "#D32F2F",
  Gothic  = "#7B1FA2",
  SciFi   = "#1976D2",
  Shelley = "#FBC02D",
  Neutral = "lightsteelblue2"
)

# Create vectors of length 35 (one for each topic)
v_fill   = rep(cols["Neutral"], 35)
v_border = rep("grey60", 35)
v_width  = rep(1, 35)

# Loop through 1:35 to handle crossovers
for (i in 1:35) {
  my_genres = c()

  if (i %in% romance_topics) my_genres = c(my_genres, cols["Romance"])
  if (i %in% gothic_topics)  my_genres = c(my_genres, cols["Gothic"])
  if (i %in% scifi_topics)   my_genres = c(my_genres, cols["SciFi"])

  # Logic:
  if (length(my_genres) == 1) {
    # Single Genre, use that colour
    v_fill[i] = my_genres[1]
  } else if (length(my_genres) > 1) {
    # Crossover, Blend the colors
    # We pick the midpoint
    v_fill[i] = colorRampPalette(my_genres)(3)[2]
  }
}

# Change border for Shelley Topics
for (i in shelley_topics) {
  v_border[i] = cols["Shelley"]
  v_width[i]  = 4
  }

# --- Topic Correlation Plot ---
topic_corr = topicCorr(stm_model, cutoff = 0.1)
```

```
plot(topic_corr,
     vertex.color = v_fill,
     vertex.frame.color = v_border,
     vertex.frame.width = v_width,
     vertex.label.cex = 0.7,
     vertex.label.color = "black",
     main = "Topic Correlation Network")
```

```
## Warning in mapply(coords[, 1], coords[, 2], vertex.color, vertex.frame.color, :
## longer argument not a multiple of length of shorter
## Warning in mapply(coords[, 1], coords[, 2], vertex.color, vertex.frame.color, :
## longer argument not a multiple of length of shorter
## Warning in mapply(coords[, 1], coords[, 2], vertex.color, vertex.frame.color, :
## longer argument not a multiple of length of shorter
```

```
legend("topleft",
       legend = c("Romance", "Gothic", "Sci-Fi", "Shelley (Border Only)", "Neutral", "Crossover"),
       fill = c(cols["Romance"], cols["Gothic"], cols["SciFi"], "white", cols["Neutral"], "white"),
       border = c("black", "black", "black", cols["Shelley"], "black", "white"),
       pt.lwd = c(1,1,1,3,1),
       bty = "n", cex = 0.8)
```




```
# --- Strong correlations ---
cor_matrix = topic_corr$cor

# Remove self-correlations
```

```r
diag(cor_matrix) = NA

# Convert to a dataframe (Topic A | Topic B | Correlation)
cor_df = as.data.frame(cor_matrix) %>%
  mutate(Topic_A = row_number()) %>%
  pivot_longer(
    cols = -Topic_A,
    names_to = "Topic_B_Label",
    values_to = "Correlation"
  ) %>%
  # Extract the number from "V11" -> 11
  mutate(Topic_B = as.integer(str_extract(Topic_B_Label, "\\d+"))) %>%
  filter(!is.na(Correlation)) %>%
  filter(Topic_A < Topic_B) %>%
  arrange(desc(Correlation))

# Filter and Clean up columns
strong_links = cor_df %>%
  filter(Correlation > 0.2) %>%
  mutate(
    Topic_A = as.integer(Topic_A),
    Topic_B = as.integer(Topic_B),
    Correlation = round(Correlation, 3)
  ) %>%
  # --- THE FIX: Select only what you want, in the order you want ---
  select(Topic_A, Topic_B, Correlation)

# Print the clean table
knitr::kable(strong_links, caption="Strong Topic Correlations (r > 0.2)")
```

Table 10: Strong Topic Correlations (r > 0.2)

Topic_A	Topic_B	Correlation
24	28	0.342
9	20	0.307
4	25	0.289
18	21	0.276
11	22	0.218
3	9	0.215
1	26	0.212
16	18	0.205

From here we can see some clear groupings and clusters between different genres.

Topic Prevalence by Genre

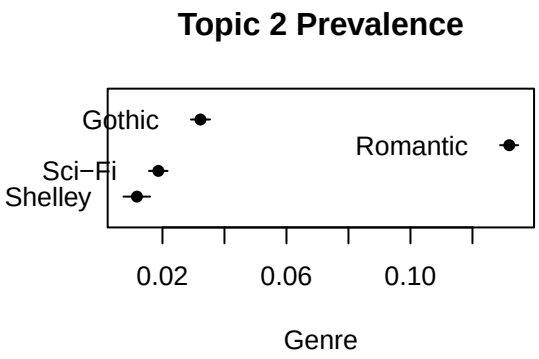
Key graph to show the range of prevalence of each topic in the four genres. This will help us see which topics are more prevalent in which genres, and see if this aligns with our interpretations of the topics.This allows us to see where Shelley's works sit in comparison to the other genres, and where she works may have influenced or been influenced by other genres.

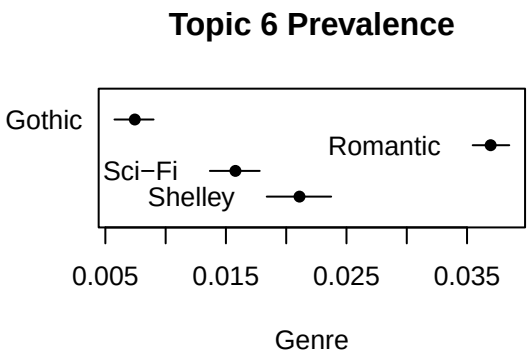
To be able to interpret this well, we have seperated topics by the genre they assosiate with, from out previous methods. Therefore, we can see which topics from each genre are more prevalent in Shelley's works, and

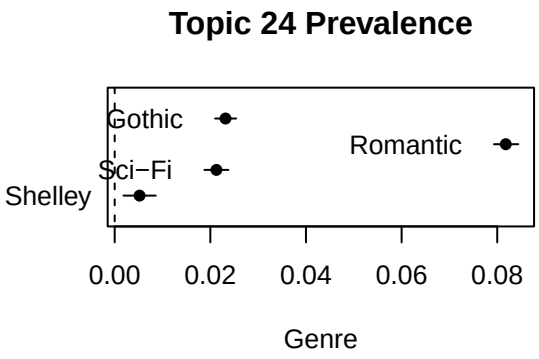
which topics from other genres are less prevalent. This will help us see which genre Shelley's works align with the most.

```
# --- Topic Prevalence by Genre ---
topic_effect = estimateEffect(
  1:optimal_k ~ genre + era,
  stm_model,
  meta = out$meta)
```

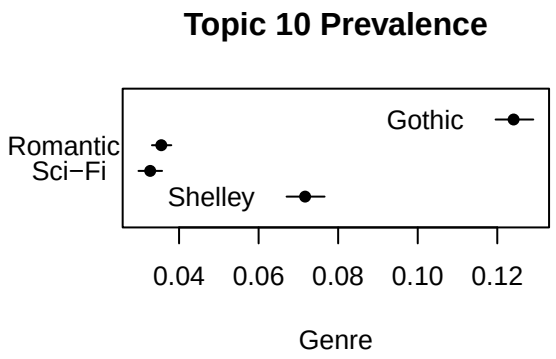
```
# Romance (Red)
romance_topics = c(2,6,24)
# Gothic (Purple)
gothic_topics = c(10,15,5)
# Sci-Fi (Blue)
scifi_topics = c(25,29,11,14)
# Shelley (Yellow)
shelley_topics = c(9,21,3)

# --- Romance Prevalence ---
par(mfrow = c(2,2))
for (topic in romance_topics) {

  plot(
    topic_effect,
    covariate = "genre",
    topics = topic,
    model = stm_model,
    method = "pointestimate",
    xlab = "Genre",

    main = paste("Topic", topic, "Prevalence"),

    labeltype = "custom",
    custom.labels = c("Gothic", "Romantic", "Sci-Fi", "Shelley")
  )
}


# --- Gothic Prevalence ---
par(mfrow = c(2,2))
```

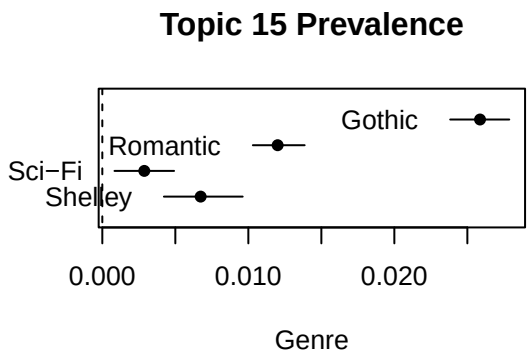


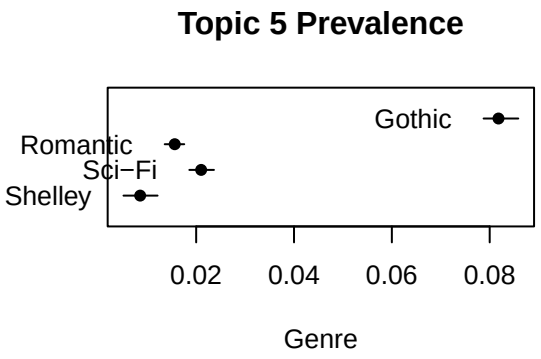




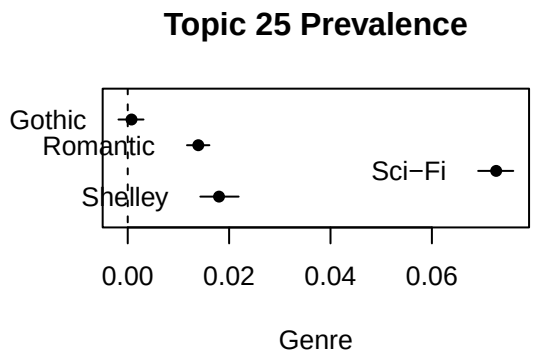

```r
for (topic in gothic_topics) {

  plot(
    topic_effect,
    covariate = "genre",
    topics = topic,
    model = stm_model,
    method = "pointestimate",
    xlab = "Genre",

    main = paste("Topic", topic, "Prevalence"),

    labeltype = "custom",
    custom.labels = c("Gothic", "Romantic", "Sci-Fi", "Shelley")
  )
}


# --- Science Fiction Prevalence ---
par(mfrow = c(2,2))
```

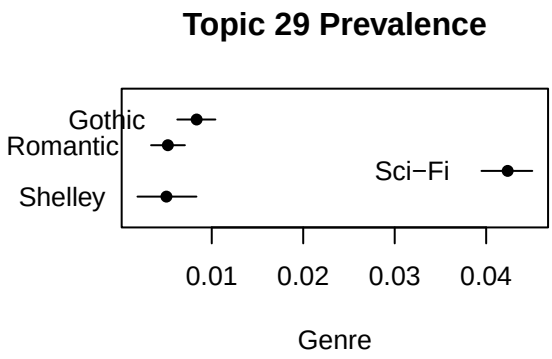


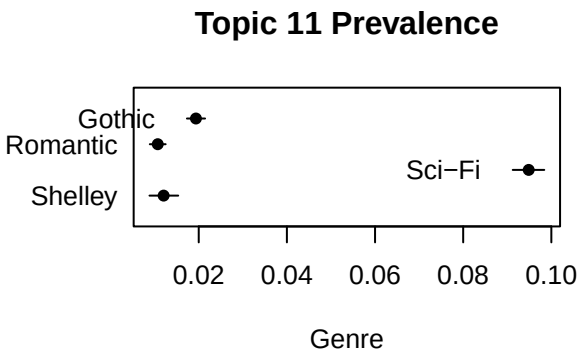


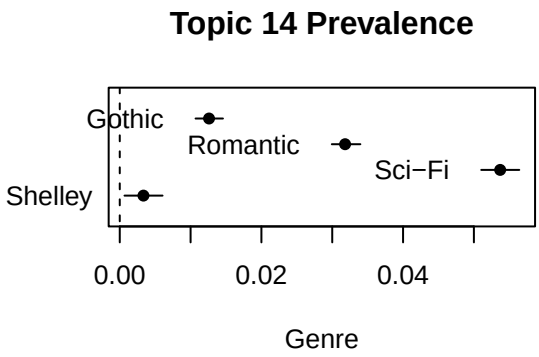



```
for (topic in scifi_topics) {

  plot(
    topic_effect,
    covariate = "genre",
    topics = topic,
    model = stm_model,
    method = "pointestimate",
    xlab = "Genre",

    main = paste("Topic", topic, "Prevalence"),

    labeltype = "custom",
    custom.labels = c("Gothic", "Romantic", "Sci-Fi", "Shelley")
  )
}
```

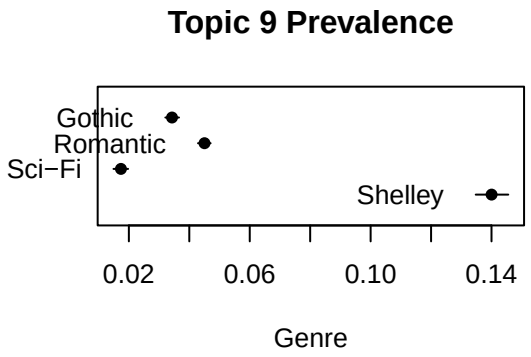


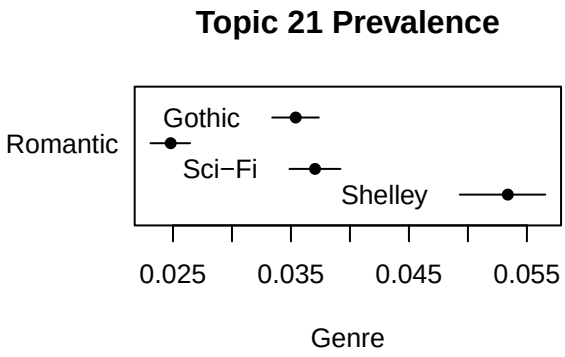


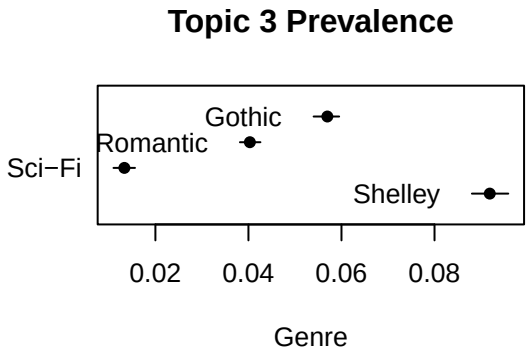





```
# --- Shelley Prevalence ---
par(mfrow = c(2,2))

for (topic in shelley_topics) {

  plot(
    topic_effect,
    covariate = "genre",
    topics = topic,
    model = stm_model,
    method = "pointestimate",
    xlab = "Genre",

    main = paste("Topic", topic, "Prevalence"),

    labeltype = "custom",
    custom.labels = c("Gothic", "Romantic", "Sci-Fi", "Shelley")
  )
}

# Reset layout
par(mfrow = c(1, 1))
```







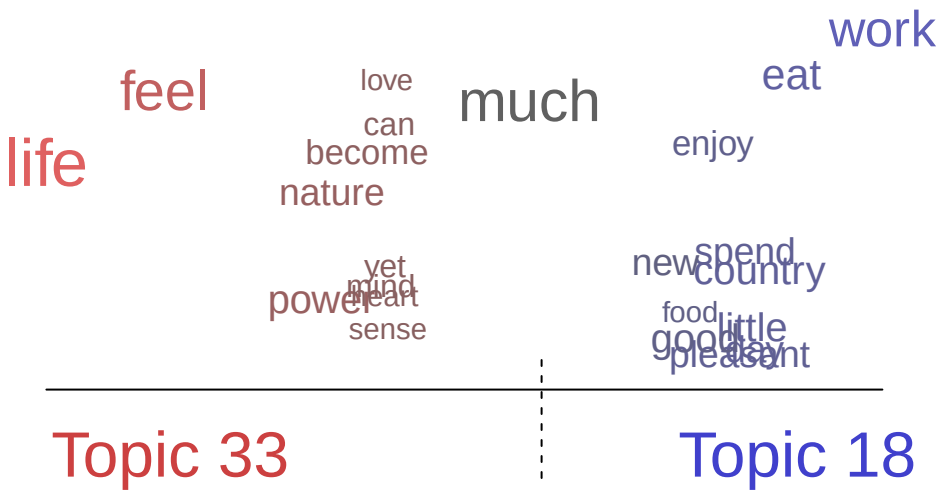
Topic Comparison

This visualisation allows us to see the comparison between two topics, and see which words are more associated with each topic. Therefore, we can compare Shelley's topics to those of other genres, and see which words are more associated with each genre. This will help us see which genre Shelley's works align with the most.

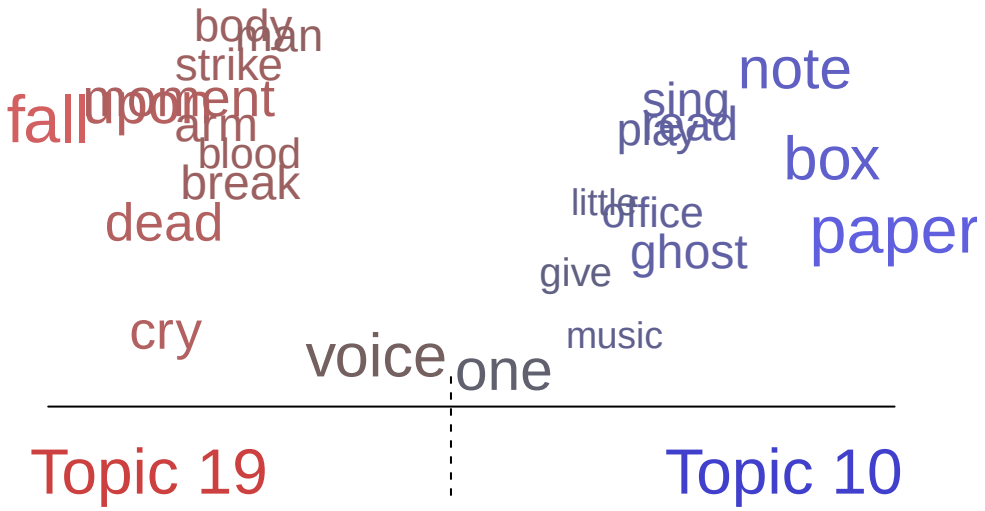
These topics need to be picked based on previous analysis. The best way to utilise this is to consider two topics that are related between two genres, but are possibly dealt with differently. For example, we considered topics related to explaining scenery using different topics, or love in order to compare their different treatment in each genre.

```
# --- Genre Visualisation ---
# "Shelley" vs. "Realism/Romance"
plot(
  stm_model,
  type = "perspectives",
  topics = c(9,6),
  model = stm_model,
  cov.value1 = "Shelley",
  cov.value2 = "Realism/Romance",
  main = "Comparing Vocabulary: Shelley vs. Romance",
  plabels = c("Topic 33", "Topic 18"),
  n = 25
)
```

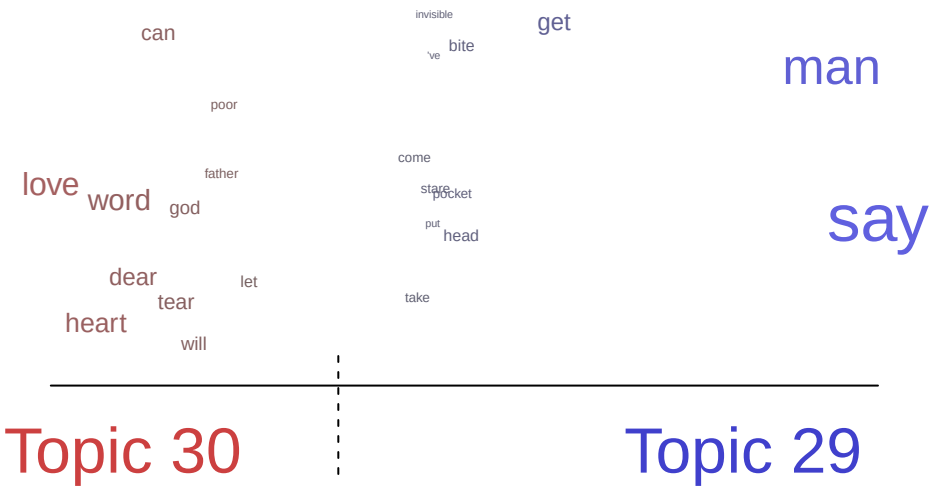
Comparing Vocabulary: Shelley vs. Romance




```
# "Shelley" vs. "Gothic Horror"
# pdf("Shelley vs Gothic.pdf", width = 12, height = 6)

plot(
  stm_model,
  type = "perspectives",
  topics = c(21,15),
  model = stm_model,
  cov.value1 = "Shelley",
  cov.value2 = "Gothic Horror",
  main = "Comparing Vocabulary: Shelley vs. Gothic",
  plabels = c("Topic 19", "Topic 10"),
  n = 25
)
```




```
#dev.off()

# "Shelley" vs. "Sci-Fi"
#pdf("Shelley vs SciFi.pdf", width = 12, height = 6)

plot(
  stm_model,
  type = "perspectives",
  topics = c(3,14),
  model = stm_model,
  cov.value1 = "Shelley",
  cov.value2 = "Sci-Fi",
  main = "Comparing Vocabulary: Shelley vs. Science Fiction",
  plabels = c("Topic 30", "Topic 29"),
  n = 25
)
```




```
#dev.off()
```

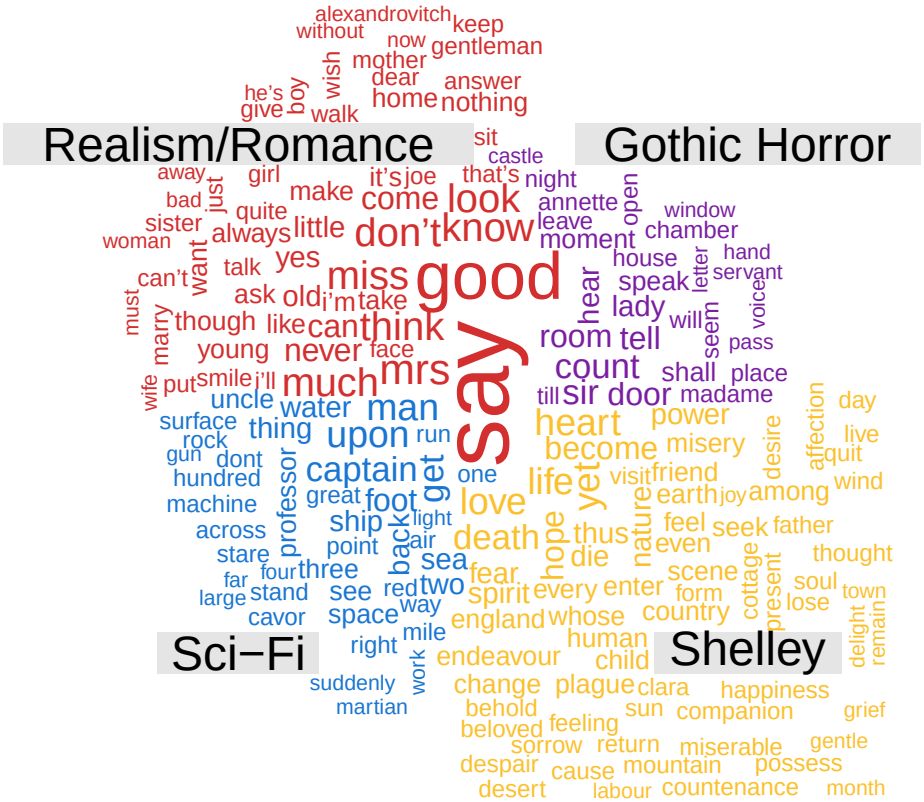
This next visualisation was more for a nice simple summary, it is a word cloud separated into each genre (by colour), highlighting the difference in vocabulary between the 3 genres and Shelley. This does look interesting, an serves as a useful summary image, or to get an idea of the difference in genres to begin analysis.

```
# Aggregate Text by Genre
genre_text = balanced_df %>%
  group_by(genre) %>%
  summarize(full_text = paste(text_segment, collapse = " "))

# Create a Term-Document Matrix
genre_corpus = VCorpus(VectorSource(genre_text$full_text))

# Clean data
genre_tdm = TermDocumentMatrix(genre_corpus, control = list(
  removePunctuation = TRUE,
  stopwords = c(stopwords("english"), my_stopwords),
  removeNumbers = TRUE,
  tolower = TRUE
))

# Convert to Matrix and Name the Columns
genre_mat = as.matrix(genre_tdm)
colnames(genre_mat) = genre_text$genre

# --- Generate the Comparison Cloud ---
```

```
cloud_colors = c(
  genre_colors["Gothic Horror"],
  genre_colors["Realism/Romance"],
  genre_colors["Sci-Fi"],
  genre_colors["Shelley"]
)


comparison.cloud(
  genre_mat,
  max.words = 200,         # Total words to show
  random.order = FALSE,    # Group them by genre
  colors = cloud_colors,   # Use your Red/Purple/Blue/Yellow palette
  title.size = 1.5,        # Size of the genre labels
  scale = c(3, 0.5)        # Size of biggest vs smallest words
)
```

```
## Warning in comparison.cloud(genre_mat, max.words = 200, random.order = FALSE, :
## towards could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(genre_mat, max.words = 200, random.order = FALSE, :
## many could not be fit on page. It will not be plotted.
```



Unused Code

If analysis of era was required, the results from this were very much as expected, and therefore was ommitted from analysis.

```
# # --- Era Visualisation ---
# era_effect = estimateEffect(
#   1:optimal_k ~ era,
#   stm_model,
#   meta = out$meta)
#
# # "Romantic" vs. "Victorian"
# plot(
#   era_effect,
#   covariate = "era",
#   topics = c(1:optimal_k),
#   model = stm_model,
#   method = "difference",
#   cov.value1 = "Late Victorian (1880-1920)",
#   cov.value2 = "Romantic/Mid-Victorian (Pre-1880)",
#   xlab = "More Romantic <--> More Victorian",
#   main = "Romantic vs Victorian"
# )
#
# # "Victorian" vs "Modern"
# plot(
#   era_effect,
#   covariate = "era",
#   topics = c(1:optimal_k),
#   model = stm_model,
#   method = "difference",
#   cov.value1 = "Modern (Atomic Age)",
#   cov.value2 = "Late Victorian (1880-1920)",
#   xlab = "More Victorian <--> More Modern",
#   main = "Victorian vs Modern"
# )
#
# # "Romantic" vs. "Modern"
# plot(
#   era_effect,
#   covariate = "era",
#   topics = c(1:optimal_k),
#   model = stm_model,
#   method = "difference",
#   cov.value1 = "Modern (Atomic Age)",
#   cov.value2 = "Romantic/Mid-Victorian (Pre-1880)",
#   xlab = "More Romantic <--> More Modern",
#   main = "Romantic vs Modern"
# )
```

This was to consider a topic and its prevalence across different genres, however, the previous visualisations was more effective.

```
# # New model comparing genre
# stm_genre_model = stm(
```

```
#   documents = out$documents,
#   vocab = out$vocab,
#   K = 20,
#   prevalence = ~ genre + era,
#   content = ~ genre,
#   data = out$meta,
#   max.em.its = 150,
#   init.type = "Spectral"
# )
#
# # --- Genre Visualisation ---
# # "Shelley" vs. "Realism/Romance"
# plot(
#   stm_model,
#   type = "perspectives",
#   topics = 7,
#   covar = "genre",
#   model = stm_model,
#   cov.value1 = "Shelley",
#   cov.value2 = "Realism/Romance",
#   main = "Word Usage: Shelley (Left) vs. Sci-Fi (Right)",
#   plabels = c("Shelley", "Realism/Romance"),
#   n = 25
# )
#
# # "Shelley" vs. "Gothic Horror"
# plot(
#   stm_model,
#   type = "perspectives",
#   topics = 7,
#   covar = "genre",
#   model = stm_model,
#   cov.value1 = "Shelley",
#   cov.value2 = "Gothic Horror",
#   main = "Word Usage: Shelley (Left) vs. Sci-Fi (Right)",
#   plabels = c("Shelley", "Gothic Horror"),
#   n = 25
# )
#
# # "Shelley" vs. "Sci-Fi"
# plot(
#   stm_model,
#   type = "perspectives",
#   topics = 7,
#   covar = "genre",
#   model = stm_model,
#   cov.value1 = "Shelley",
#   cov.value2 = "Sci-Fi",
#   main = "Word Usage: Shelley (Left) vs. Sci-Fi (Right)",
#   plabels = c("Shelley", "Sci-Fi"),
#   n = 25
# )
```

References

[Dealban, 2025] Dealban, D. (2025). Learning stm. GitHub Repository.

[Haldane, 2015] Haldane, M. (2015). What Frankenstein's creature can really tell us about AI.

[Mitra, 2011] Mitra, Z. (2011). A science fiction in a gothic scaffold: A reading of mary shelley's Frankenstein. Rupkatha Journal on Interdisciplinary Studies in Humanities, 3(1):52–59.

[Paley, 1993] Paley, M. D. (1993). Mary shelley's The Last Man: Apocalypse without millennium. Keats-Shelley Review, 4(1):1–25.

[Project Gutenberg, 2025] Project Gutenberg (2025). Project gutenberg. Digital Library of Free eBooks.

[Roberts et al., 2019] Roberts, M. E., Stewart, B. M., and Tingley, D. (2019). stm: An R package for structural topic models. Journal of Statistical Software, 91(2):1–40.

[Robinson, 2025] Robinson, D. (2025). gutenbergr: Download and Process Public Domain Works from Project Gutenberg. R package version 0.2.4.

[Silge, 2018a] Silge, J. (2018a). Evaluating STM.

[Silge, 2018b] Silge, J. (2018b). Sherlock holmes & STM.

[Silge and Robinson, 2017] Silge, J. and Robinson, D. (2017). Text Mining with R: A Tidy Approach. O'Reilly Media, Sebastopol, CA.

[Stableford, 1995] Stableford, B. (1995). Frankenstein and the origins of science fiction. In Seed, D., editor, Anticipations: Essays on Early Science Fiction and its Precursors. Syracuse University Press, Syracuse, NY.

[Wikipedia Contributors, 2025a] Wikipedia Contributors (2025a). Armageddon 2419 a.d. Accessed: 2025-12-07.

[Wikipedia Contributors, 2025b] Wikipedia Contributors (2025b). Gothic fiction. Accessed: 2025-12-07.

AI Usage Report

AI Usage Reporting: Transparency Statement

This project utilised Large Language Models (LLMs) to assist in the technical implementation and editorial refinement of the research. The specific contributions of the AI are detailed below:

1. Code Development and Debugging The R code used for data acquisition, preprocessing, and modelling was co-developed with AI assistance. Specifically, the

AI was used to: - Troubleshoot errors in the gutenbergr package mirror configuration. - Generate the syntax for the stm package, specifically regarding the estimateEffect and topicCorr functions. - Draft the ggplot2 code for the custom visualisation of topic prevalence and vocabulary comparison clouds. - Verification: All code was executed, tested, and validated by the author on a local machine to ensure reproducibility and accuracy.

2. Data Structuring

The AI assisted in compiling the list of Project Gutenberg IDs for the comparative corpus (Romantic, Gothic, and Science

Fiction novels) to ensure a balanced dataset.

It generated the LaTeX table code to format the raw results into academic-standard tables (e.g., Table 11: Thematic Super-Clusters).

3. Editorial and Stylistic Refinement

Draft sections of the report were submitted to the AI for proofreading. The AI provided suggestions for correcting spelling, grammar, and academic tone.

The AI assisted in refining the "Problem Statement" and "Abstract" to ensure conciseness and clarity.

Note: The core literary arguments, the interpretation of the "Bridge" hypothesis, and the final conclusions regarding Shelley's technological limitations are the original intellectual work of the author. The AI functioned solely as an editor and technical assistant.

4. Bibliographic Formatting

The AI was used to convert raw URLs and citations into formatted BibTeX entries to ensure compatibility with the LaTeX document structure.