visualizations_knit.rmd

2025-03-06

Topic Frequency Histogram

```
#Frequency By Topic test


# Load required libraries
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

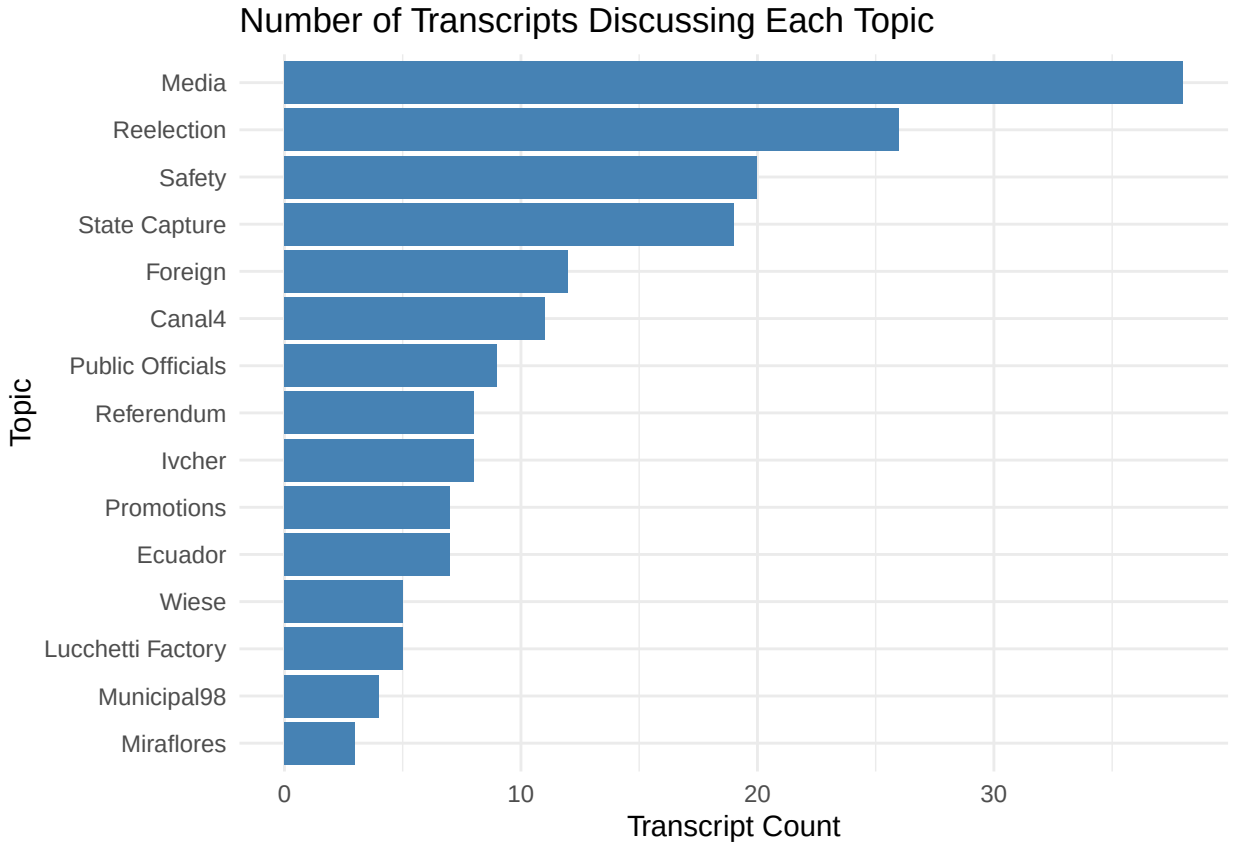
```
library(ggplot2)
library(tidyr)
library(stringr)

# Step 1: Read the inventory file
file <- "../data/Updated Inventory & Descriptions/Descriptions.csv"
inventory <- read_csv(file)
```

```
## Rows: 104 Columns: 24
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (23): date, speakers, original_n, Missing Topic, in_book, in_online_arch...
## dbl  (1): n
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Step 2: Identify topic columns
topic_columns <- inventory %>%
  select(starts_with("topic_")) %>%
  colnames()

# Step 3: Count how many transcripts mention each topic (non-NA values)
topic_counts <- inventory %>%
  select(all_of(topic_columns)) %>%
  summarise(across(everything(), ~ sum(!is.na(.)))) %>%
```

```
  pivot_longer(cols = everything(), names_to = "Topic", values_to = "Transcript_Count")

# Step 4: Clean up topic names for readability
topic_counts <- topic_counts %>%
  mutate(Topic = str_replace_all(Topic, "topic_", ""),
         Topic = str_replace_all(Topic, "_", " "),
         Topic = str_to_title(Topic)) %>%
  arrange(desc(Transcript_Count))

# Step 5: Plot histogram
ggplot(topic_counts, aes(x = reorder(Topic, Transcript_Count), y = Transcript_Count)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Number of Transcripts Discussing Each Topic",
       x = "Topic",
       y = "Transcript Count") +
  theme_minimal()
```



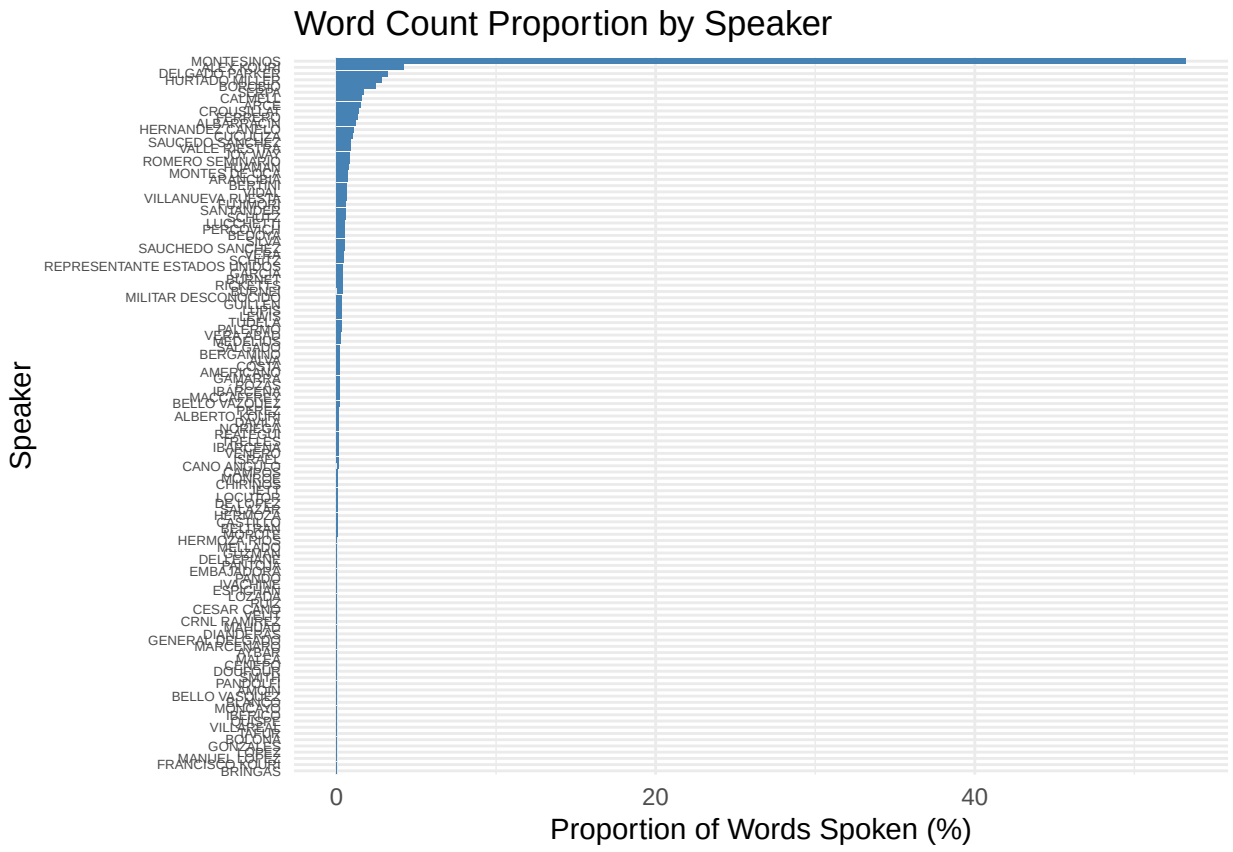
Word Count Proportion (excluding 'desconoocido')

```
#install.packages("readr")
#install.packages("dplyr")
#install.packages("ggplot2")

library(readr)
```

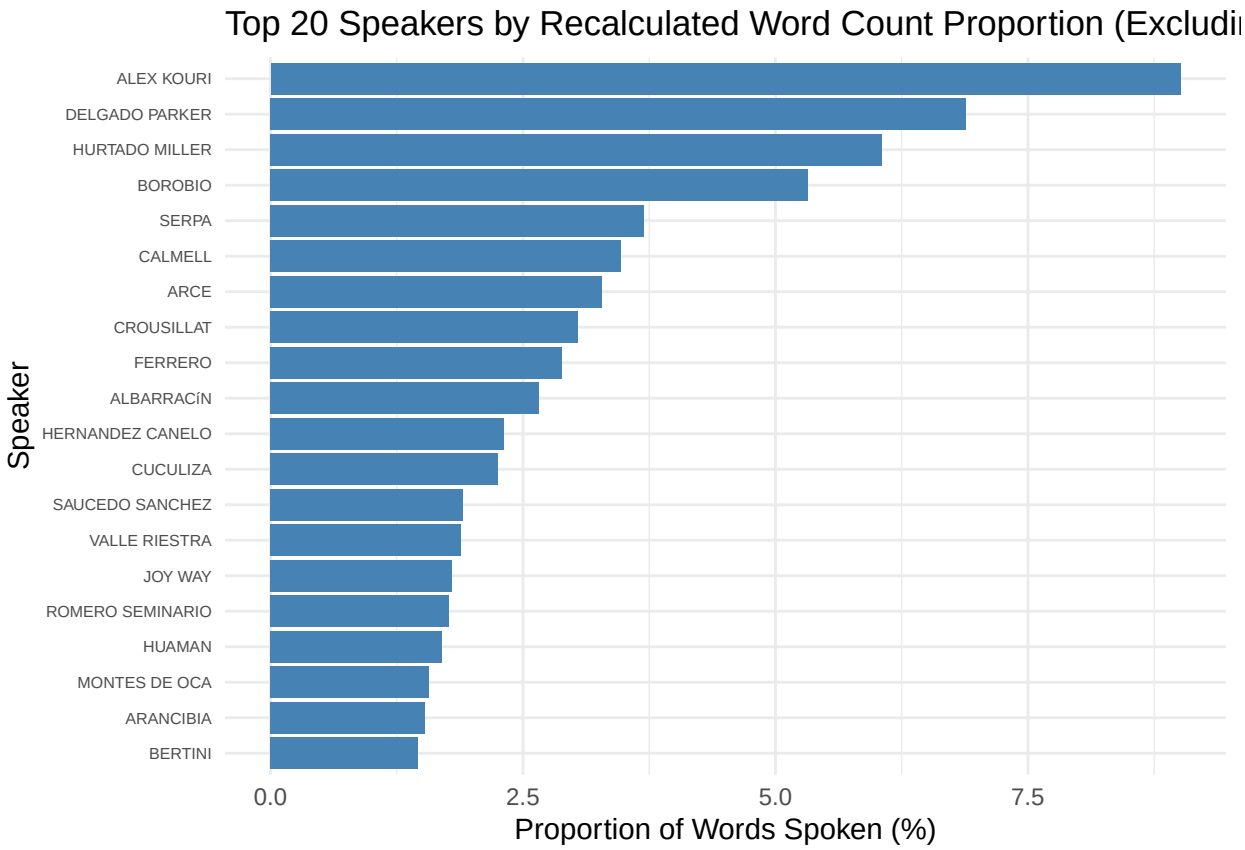
```r
library(dplyr)
library(ggplot2)

# Read in File
file_path <- "../data/count_results_all.tsv"
word_data <- read_tsv(file_path)
```

```
## Rows: 116 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (2): Speaker, Proportion of Word Count
## dbl (1): Total Word Count
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Exclude 'DESCONOCIDO'
word_data <- word_data %>% filter(Speaker != "DESCONOCIDO")

# Compute proportions (Exluding the use of existing Propoetion Column)
word_data <- word_data %>%
  mutate(Proportion = (`Total Word Count` / sum(`Total Word Count`)) * 100)

# Plot bar chart
ggplot(word_data, aes(x = reorder(Speaker, Proportion), y = Proportion)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +  # Flip coordinates for better readability
  labs(title = "Word Count Proportion by Speaker",
       x = "Speaker",
       y = "Proportion of Words Spoken (%)") +
  theme_minimal()+
  theme(axis.text.y = element_text(size = 5))
```




```r
# Filter out 'MONTESINOS' and 'DESCONOCIDO'
filtered_data <- word_data %>%
  filter(!(Speaker %in% c("MONTESINOS", "DESCONOCIDO")))

# Recalculate proportions based on filtered total word counts
filtered_data <- filtered_data %>%
  mutate(Proportion = (`Total Word Count` / sum(`Total Word Count`)) * 100)

# Select top 20 speakers by recalculated proportion
top20_data <- filtered_data %>%
  arrange(desc(Proportion)) %>%
  slice_head(n = 20)

# Plot bar chart
ggplot(top20_data, aes(x = reorder(Speaker, Proportion), y = Proportion)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 20 Speakers by Recalculated Word Count Proportion (Excluding MONTESINOS & DESCONOCI
       x = "Speaker",
       y = "Proportion of Words Spoken (%)") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 6))
```



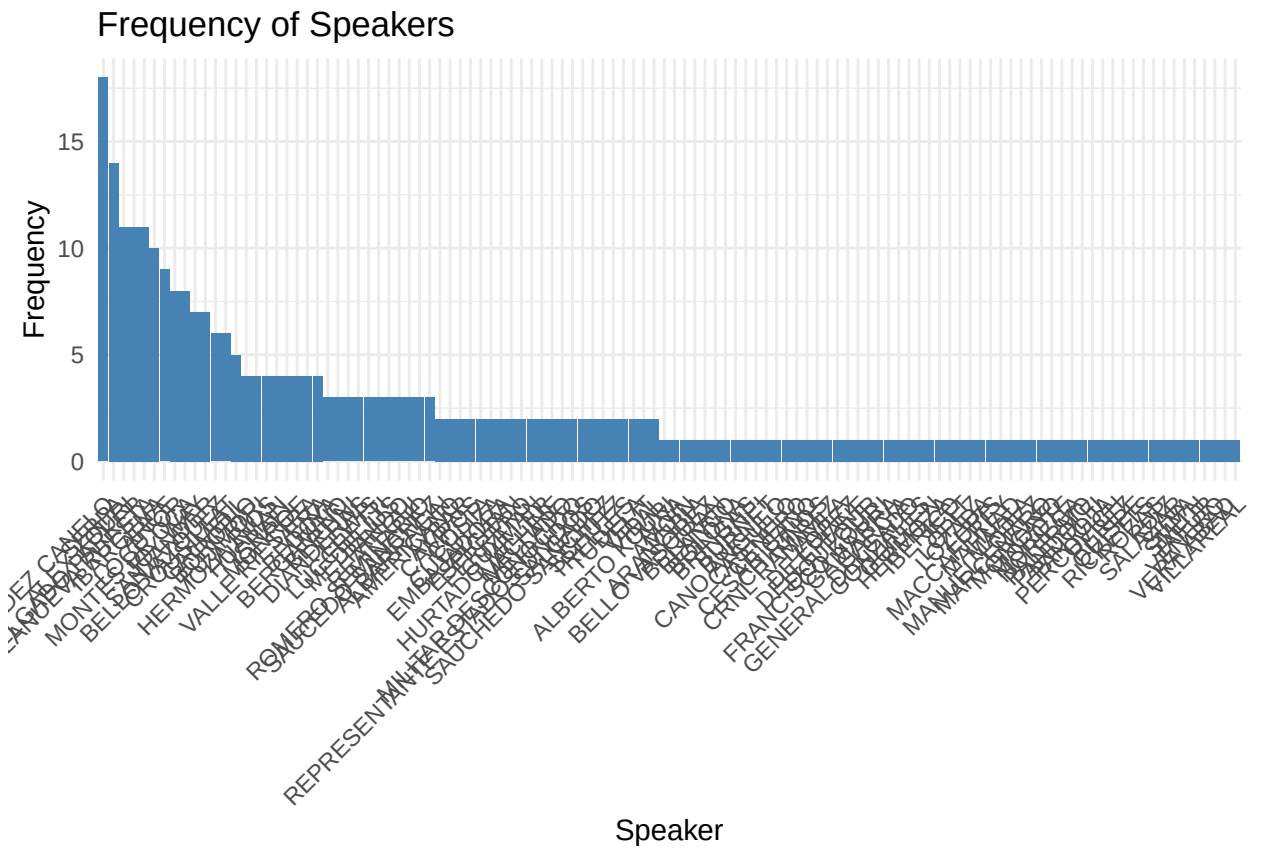
Speaker Frequency

```
# Load necessary libraries
library(ggplot2)
library(dplyr)
library(readr)

# Load the dataset
file<-"../output/csv outputs/speaker_frequency_results(all).csv"
df <- read_csv(file)
```

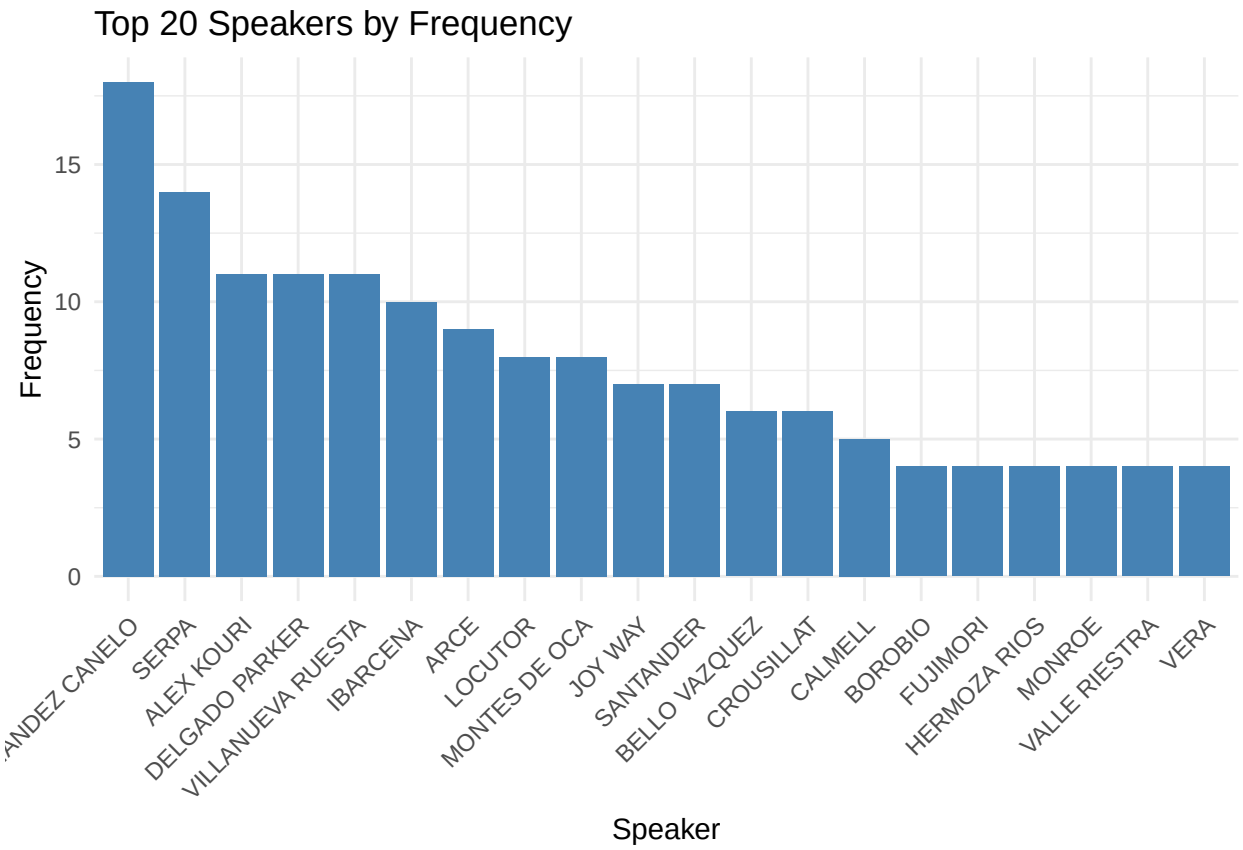
```
## Rows: 114 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Speaker
## dbl (1): Frequency
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Filter out 'Montesinos' and 'Desconocido'
df_filtered <- df %>%
  filter(!(Speaker %in% c("MONTESINOS", "DESCONOCIDO")))

# Create the bar graph
ggplot(df_filtered, aes(x = reorder(Speaker, -Frequency), y = Frequency)) +
```

```
  geom_bar(stat = "identity", fill = "steelblue") +
  theme_minimal() +
  labs(title = "Frequency of Speakers",
       x = "Speaker",
       y = "Frequency") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Keep only the top 20 speakers based on frequency
df_top20 <- df_filtered %>%
  arrange(desc(Frequency)) %>%
  slice_head(n = 20)

# Plot top 20
ggplot(df_top20, aes(x = reorder(Speaker, -Frequency), y = Frequency)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  theme_minimal() +
  labs(title = "Top 20 Speakers by Frequency",
       x = "Speaker",
       y = "Frequency") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



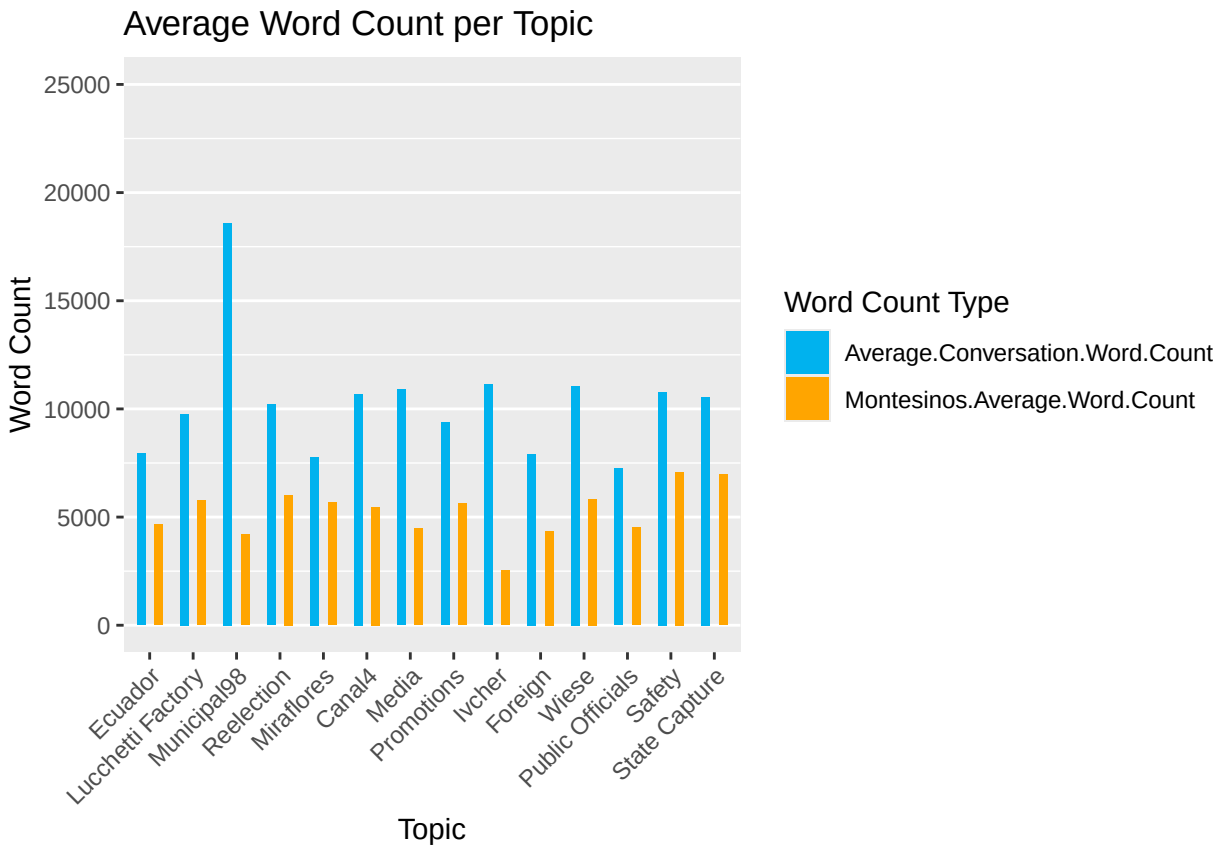
Word Count Per Topic

```
# Install & Library necessary packages
#install.packages("tidyr")
#install.packages("tidyverse")
#install.packages("dplyr")
#install.packages("ggplot2")
library(ggplot2)
library(dplyr)
library(tidyr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v purrr     1.0.4
## v lubridate 1.9.4     v tibble    3.2.1
## -- 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
```

```
# Read in 'topic_vmt_avg_count.csv' file
file<-"../output/csv outputs/topic_vmt_avg_count.csv"
df <- read.csv(file)

# Remove 'topic_' prefix from 'Topic' column
df$Topic <- gsub("topic_", "", df$Topic)
```

```r
# Replace underscores with spaces in 'Topic' column
df$Topic <- gsub("_", " ", df$Topic)

# Capitalize first letter of each item in 'Topic' column
df$Topic <- str_to_title(df$Topic)

# Convert 'Topic' to factor to maintain order
df$Topic <- factor(df$Topic, levels = df$Topic)

# Reshape data using pivot_longer()
df_long <- df %>%
  pivot_longer(cols = c(Average.Conversation.Word.Count, Montesinos.Average.Word.Count),
               names_to = "Word_Count_Type",
               values_to = "Word_Count")

# Create a grouped bar plot with y-axis limit set to 25,000 and removed x-axis lines
ggplot(df_long, aes(x = Topic, y = Word_Count, fill = Word_Count_Type)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.4) +
  labs(title = "Average Word Count per Topic",
       x = "Topic",
       y = "Word Count",
       fill = "Word Count Type") +
  scale_fill_manual(values = c("Average.Conversation.Word.Count" = "deepskyblue2",
                               "Montesinos.Average.Word.Count" = "orange")) +
  scale_y_continuous(limits = c(0, 25000)) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank()
  )
```



Histograms

Average Length of Conversations

```r
# Define the path to the finalized_data folder
data_path <- "../data/modified_data/finalized_data"

# Get a list of CSV and TSV files from finalized_data without setting it as the working directory
csv_files <- list.files(path = data_path, pattern = "\\.csv$", full.names = TRUE)
tsv_files <- list.files(path = data_path, pattern = "\\.tsv$", full.names = TRUE)

# Combine file lists
all_files <- c(csv_files, tsv_files)

# Load necessary libraries
#install.packages("readr")
#install.packages("stringr")
#install.packages("ggplot2")
library(ggplot2)
library(readr)  # For reading CSV & TSV files
library(stringr)  # For text processing

# Initialize total word count and file count
total_word_count <- 0
file_count <- 0
```

```r
# Function to count words in the 'speech' column safely
count_words <- function(text) {
  if (is.null(text) || all(is.na(text))) {
    return(0)  # Return 0 if text is NULL or all NA
  }
  text <- na.omit(text)  # Remove NA values
  sum(str_count(text, "\\S+"))  # Count words in non-NA text
}

# Loop through each file
for (file in all_files) {
  # Read the file and handle errors
  df <- tryCatch({
    if (grepl("\\.csv$", file)) {
      read_csv(file, show_col_types = FALSE)  # Read CSV
    } else if (grepl("\\.tsv$", file)) {
      read_tsv(file, show_col_types = FALSE)  # Read TSV
    }
  }, error = function(e) {
    cat("Error reading file:", file, "\n")
    return(NULL)
  })

  # Check if file was successfully read and 'speech' column exists
  if (!is.null(df) && "speech" %in% colnames(df)) {
    # Calculate total words in 'speech' column
    file_word_count <- count_words(df$speech)

    # Debugging: Print word count for each file
    cat("File:", basename(file), "- Word Count:", file_word_count, "\n")

    # Update total word count and file count
    total_word_count <- total_word_count + file_word_count
    file_count <- file_count + 1
  } else {
    cat("Skipping file (missing 'speech' column):", basename(file), "\n")
  }
}
```

```
## File: 1.csv - Word Count: 10375
## File: 100.csv - Word Count: 3040
## File: 101.csv - Word Count: 2496
## File: 102.csv - Word Count: 8694
## File: 103.csv - Word Count: 9206
## File: 104.csv - Word Count: 9289
## File: 13.csv - Word Count: 3447
## File: 19.csv - Word Count: 5693
## File: 2.csv - Word Count: 7120
## File: 24.csv - Word Count: 6405
## File: 25.csv - Word Count: 15867
## File: 3.csv - Word Count: 7006
## File: 37.csv - Word Count: 3317
## File: 38.csv - Word Count: 8263
## File: 39.csv - Word Count: 5594
```

```
## File: 4.csv - Word Count: 175
## File: 41.csv - Word Count: 4818
## File: 47.csv - Word Count: 15941
## File: 55.csv - Word Count: 9759
## File: 6.csv - Word Count: 13035
## File: 63.csv - Word Count: 9793
## File: 64.csv - Word Count: 4544
## File: 67.csv - Word Count: 11435
## File: 69.csv - Word Count: 8547
## File: 74.csv - Word Count: 11721
## File: 75.csv - Word Count: 5565
## File: 79.csv - Word Count: 1309
## File: 8.csv - Word Count: 4895
## File: 80.csv - Word Count: 12492
## File: 81.csv - Word Count: 4633
## File: 84.csv - Word Count: 3688
## File: 87.csv - Word Count: 3911
## File: 88.csv - Word Count: 5321
## File: 89.csv - Word Count: 515
## File: 90.csv - Word Count: 10433
## File: 91.csv - Word Count: 2693
## File: 94.csv - Word Count: 6655
## File: 95.csv - Word Count: 1809
## File: 96.csv - Word Count: 4625
## File: 97.csv - Word Count: 10615
## File: 98.csv - Word Count: 5824
## File: 99.csv - Word Count: 3609
## File: 10.tsv - Word Count: 15704

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 11.tsv - Word Count: 9420

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 12.tsv - Word Count: 1283
## File: 14.tsv - Word Count: 955

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 15.tsv - Word Count: 12545

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 16.tsv - Word Count: 13621
## File: 17.tsv - Word Count: 11547
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 20.tsv - Word Count: 16815

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 21.tsv - Word Count: 17012
## File: 22.tsv - Word Count: 2219
## File: 23.tsv - Word Count: 7675
## File: 26.tsv - Word Count: 11370
## File: 27.tsv - Word Count: 17701

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 28.tsv - Word Count: 10281
## File: 29.tsv - Word Count: 7865

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 30.tsv - Word Count: 5266

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 31.tsv - Word Count: 5557

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 32.tsv - Word Count: 11794

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 33.tsv - Word Count: 20785

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 34.tsv - Word Count: 29120
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 35.tsv - Word Count: 18154

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 36.tsv - Word Count: 10996

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 40.tsv - Word Count: 17326
## File: 42.tsv - Word Count: 3673

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 43.tsv - Word Count: 1946
## File: 44.tsv - Word Count: 8323
## File: 45.tsv - Word Count: 4270

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 46.tsv - Word Count: 7861

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 48.tsv - Word Count: 3508

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 49.tsv - Word Count: 11693

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## File: 5.tsv - Word Count: 9384

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

```
##   problems(dat)
```

```
## File: 50.tsv - Word Count: 16449
## File: 51.tsv - Word Count: 5050
## File: 52.tsv - Word Count: 17176
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 53.tsv - Word Count: 10802
## File: 54.tsv - Word Count: 28813
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 56.tsv - Word Count: 10720
## File: 57.tsv - Word Count: 14216
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 58.tsv - Word Count: 4750
## File: 59.tsv - Word Count: 8638
## File: 60.tsv - Word Count: 12628
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 61.tsv - Word Count: 3346
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 62.tsv - Word Count: 11803
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 65.tsv - Word Count: 2771
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 66.tsv - Word Count: 7588
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

```
##   problems(dat)
```

```
## File: 68.tsv - Word Count: 9840
## File: 7.tsv - Word Count: 11146
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 70.tsv - Word Count: 19642
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 71.tsv - Word Count: 11277
## File: 72.tsv - Word Count: 4484
## File: 73.tsv - Word Count: 3875
## File: 76.tsv - Word Count: 9166
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 77.tsv - Word Count: 11611
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 78.tsv - Word Count: 10508
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 82.tsv - Word Count: 13637
## File: 83.tsv - Word Count: 4086
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## File: 85.tsv - Word Count: 1806
## File: 86.tsv - Word Count: 5690
## File: 9.tsv - Word Count: 16843
```

```
# Calculate the average word count per file
average_word_count <- ifelse(file_count > 0, total_word_count / file_count, NA)

# Print result
cat("Total Word Count:", total_word_count, "\n")
```

```
## Total Word Count: 898202
```

```r
cat("Number of Files Processed:", file_count, "\n")
```

```
## Number of Files Processed: 101
```

```r
cat("Average Word Count per File:", average_word_count, "\n")
```

```
## Average Word Count per File: 8893.089
```

```r
# Initialize a data frame to store file names and word counts
word_counts <- data.frame(File = character(), Word_Count = numeric(), stringsAsFactors = FALSE)

# Loop through each file again to store word counts
for (file in all_files) {
  df <- tryCatch({
    if (grepl("\\.csv$", file)) {
      read_csv(file, show_col_types = FALSE)
    } else if (grepl("\\.tsv$", file)) {
      read_tsv(file, show_col_types = FALSE)
    }
  }, error = function(e) {
    cat("Error reading file:", file, "\n")
    return(NULL)
  })

  if (!is.null(df) && "speech" %in% colnames(df)) {
    file_word_count <- count_words(df$speech)

    # Store results in the data frame
    word_counts <- rbind(word_counts, data.frame(File = basename(file), Word_Count = file_word_count))
  }
}
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

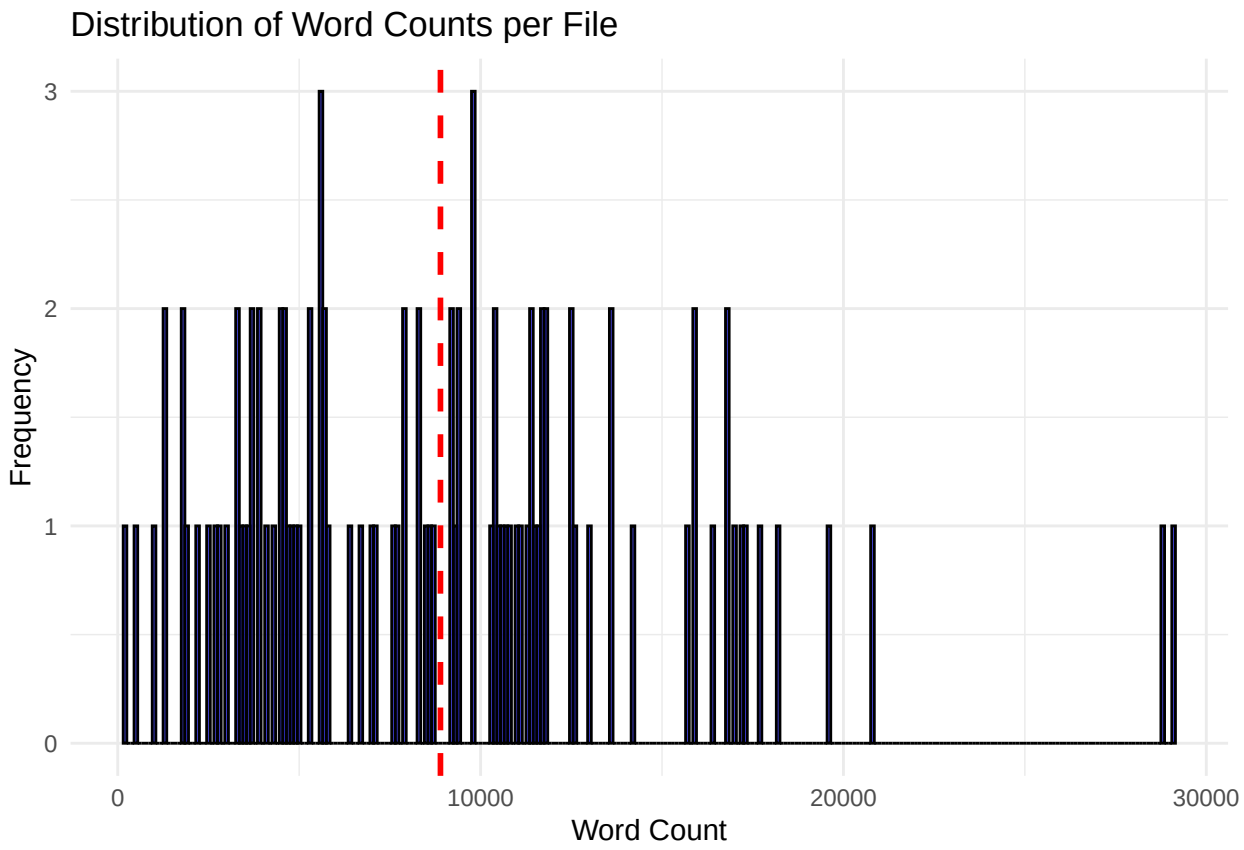
```
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
```

```
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

```
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```r
# Calculate average word count
average_word_count <- mean(word_counts$Word_Count, na.rm = TRUE)

# Create histogram
ggplot(word_counts, aes(x = Word_Count)) +
  geom_histogram(binwidth = 100, fill = "blue", alpha = 0.7, color = "black") +
  geom_vline(aes(xintercept = average_word_count), color = "red", linetype = "dashed", size = 1) +
  labs(title = "Distribution of Word Counts per File",
       x = "Word Count",
       y = "Frequency") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Average Conversation Length by Topic

```r
# Load required libraries
library(dplyr)
library(readr)
library(stringr)
library(ggplot2)

# Define file paths
inventory_file <- "../data/Updated Inventory & Descriptions/Descriptions.csv"
directory_path <- "../data/modified_data/finalized_data"

# Read the inventory data
inventory_data <- read_csv(inventory_file, col_types = cols())

# List of topic columns
topic_columns <- c("topic_referendum", "topic_ecuador", "topic_lucchetti_factory", "topic_municipal98",
                   "topic_reelection", "topic_miraflores", "topic_canal4", "topic_media", "topic_promot
                   "topic_ivcher", "topic_foreign", "topic_wiese", "topic_public_officials", "topic_saf

# Initialize word count storage
topic_word_count <- setNames(rep(0, length(topic_columns)), topic_columns)

# Function to count words in a transcript file (from the 'speech' column)
count_words_in_transcript <- function(file_path) {
  if (!file.exists(file_path)) return(0)

  # Read the transcript file
  transcript_data <- read_tsv(file_path, col_types = cols(), na = c("", "NA"))

  # Check if the 'speech' column exists
  if (!"speech" %in% colnames(transcript_data)) return(0)

  # Extract valid speeches (ignore missing values)
  valid_speeches <- transcript_data %>%
    filter(!is.na(speech)) %>%
    pull(speech)

  # Calculate total word count from the 'speech' column
  total_words <- sum(str_count(valid_speeches, "\\S+"))
  return(total_words)
}

# Iterate over transcript files and compute word counts
for (i in 1:nrow(inventory_data)) {
  transcript_id <- inventory_data$n[i]

  # Construct file path (assuming files are named as "n.tsv")
  file_path <- file.path(directory_path, paste0(transcript_id, ".tsv"))

  # Compute word count for the transcript
  transcript_word_count <- count_words_in_transcript(file_path)

  # Assign word count to relevant topics
```

```r
  for (topic in topic_columns) {
    if (!is.na(inventory_data[[topic]][i]) && inventory_data[[topic]][i] == "x") {
      topic_word_count[topic] <- topic_word_count[topic] + transcript_word_count
    }
  }
}
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

```
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
```

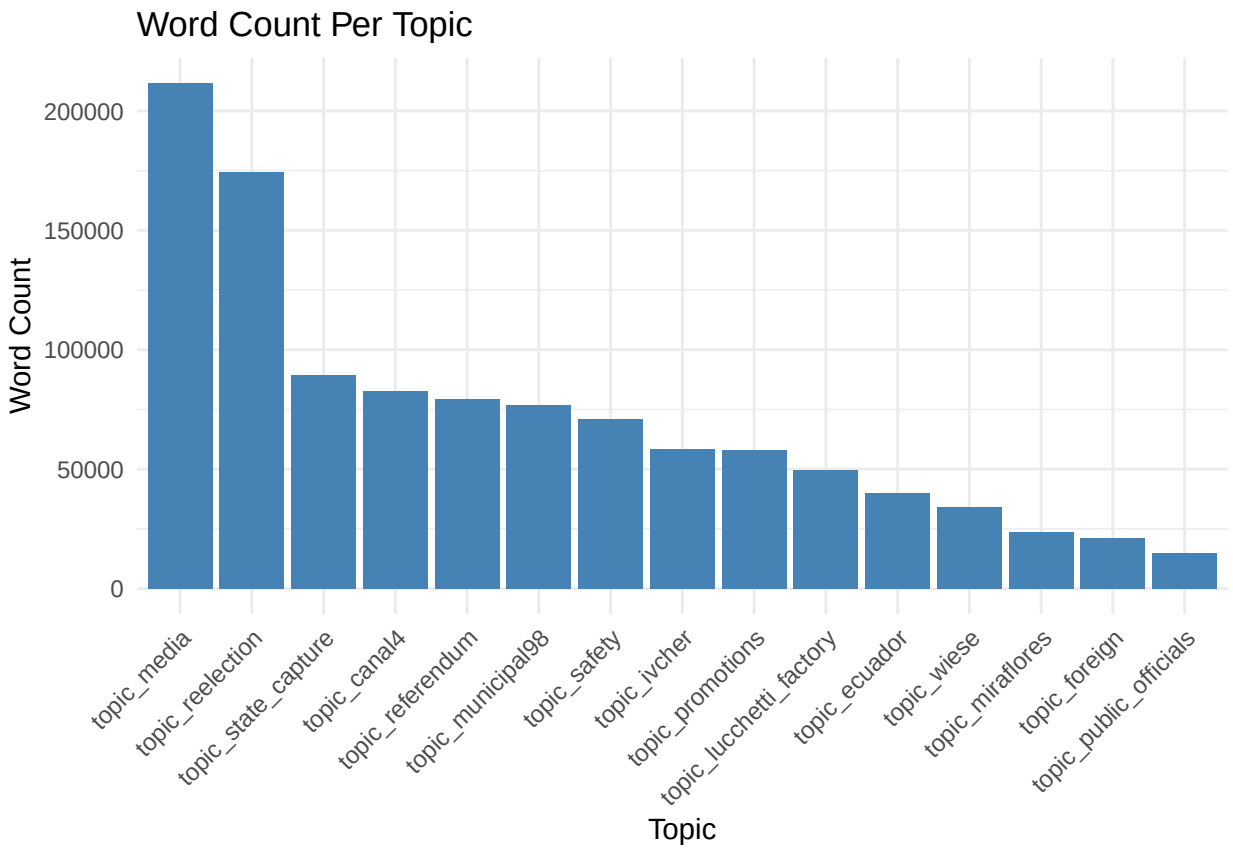
```
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```r
# Convert results to a data frame
word_count_df <- data.frame(
  Topic = names(topic_word_count),
  Word_Count = unlist(topic_word_count),
  stringsAsFactors = FALSE
)

# Save results to CSV
output_file <- "word_count_by_topic.csv"
write_csv(word_count_df, output_file)

# Print summary
print(word_count_df)
```

```
##                                            Topic Word_Count
```

```
## topic_referendum               topic_referendum      79413
## topic_ecuador                     topic_ecuador      40053
## topic_lucchetti_factory topic_lucchetti_factory      49447
## topic_municipal98             topic_municipal98      76755
## topic_reelection               topic_reelection     174304
## topic_miraflores               topic_miraflores      23839
## topic_canal4                       topic_canal4      82543
## topic_media                         topic_media     211588
## topic_promotions               topic_promotions      58156
## topic_ivcher                       topic_ivcher      58581
## topic_foreign                     topic_foreign      21249
## topic_wiese                         topic_wiese      33930
## topic_public_officials   topic_public_officials      14890
## topic_safety                       topic_safety      71054
## topic_state_capture         topic_state_capture      89594
```

```r
# Histogram of Word Count Per Topic
ggplot(word_count_df, aes(x = reorder(Topic, -Word_Count), y = Word_Count)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Word Count Per Topic", x = "Topic", y = "Word Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



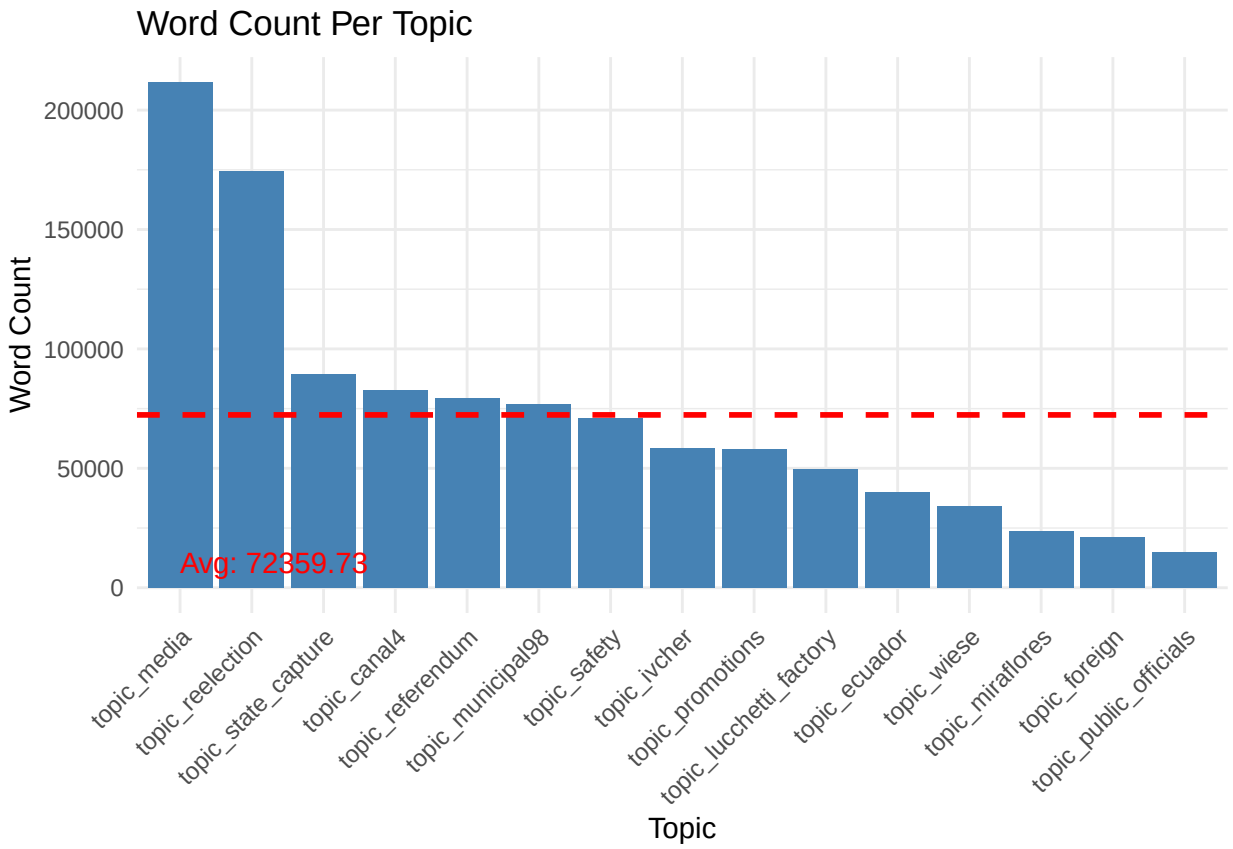

```r
# Calculate the average word count across all transcripts
average_word_count_all <- sum(word_count_df$Word_Count) / nrow(inventory_data)

# Print average word count
print(paste("Average Word Count Across All Transcripts:", round(average_word_count_all, 2)))
```

```
## [1] "Average Word Count Across All Transcripts: 10436.5"
```

```
# Calculate the average word count across all topics
average_word_count_topics <- sum(word_count_df$Word_Count) / length(topic_columns)

# Print average word count per topic
print(paste("Average Word Count Across All Topics:", round(average_word_count_topics, 2)))
```

```
## [1] "Average Word Count Across All Topics: 72359.73"
```

```
ggplot(word_count_df, aes(x = reorder(Topic, -Word_Count), y = Word_Count)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_hline(yintercept = average_word_count_topics, linetype = "dashed", color = "red", size = 1) +
  labs(title = "Word Count Per Topic", x = "Topic", y = "Word Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  annotate("text", x = 1, y = average_word_count_all + 100, label = paste("Avg:", round(average_word_cou
```




```
print(paste("Total Word Count Across All Transcripts:", sum(word_count_df$Word_Count)))
```

```
## [1] "Total Word Count Across All Transcripts: 1085396"
```

Bar Plot

```
library(readr)
library(dplyr)
library(stringr)
library(fs)
```

```r
library(purrr)

# Define the directory
finalized_data_path <- "../data/modified_data/finalized_data"

# Get CSV and TSV files
csv_files <- dir_ls(finalized_data_path, regexp = "\\.csv$")
tsv_files <- dir_ls(finalized_data_path, regexp = "\\.tsv$")
all_files <- c(csv_files, tsv_files)

# Helper to read file based on extension
read_transcript <- function(file) {
  if (str_detect(file, "\\.csv$")) {
    read_csv(file, show_col_types = FALSE)
  } else {
    read_tsv(file, show_col_types = FALSE)
  }
}

# Initialize lists to track total word count and file appearances
word_counts_list <- list()
speaker_file_map <- list()

# Process each file
for (file in all_files) {
  df <- read_transcript(file)

  if (!("speaker_std" %in% names(df)) || !("speech" %in% names(df))) next

  # Count words per speaker in this file
  file_word_counts <- df %>%
    filter(!is.na(speaker_std), !is.na(speech)) %>%
    group_by(speaker_std) %>%
    summarise(words = sum(str_count(speech, "\\S+")), .groups = "drop")

  # Update total word counts
  for (i in seq_len(nrow(file_word_counts))) {
    speaker <- file_word_counts$speaker_std[i]
    words <- file_word_counts$words[i]

    word_counts_list[[speaker]] <- (word_counts_list[[speaker]] %||% 0) + words
    speaker_file_map[[speaker]] <- union(speaker_file_map[[speaker]], file)
  }
}
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
```

```
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

```
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
```

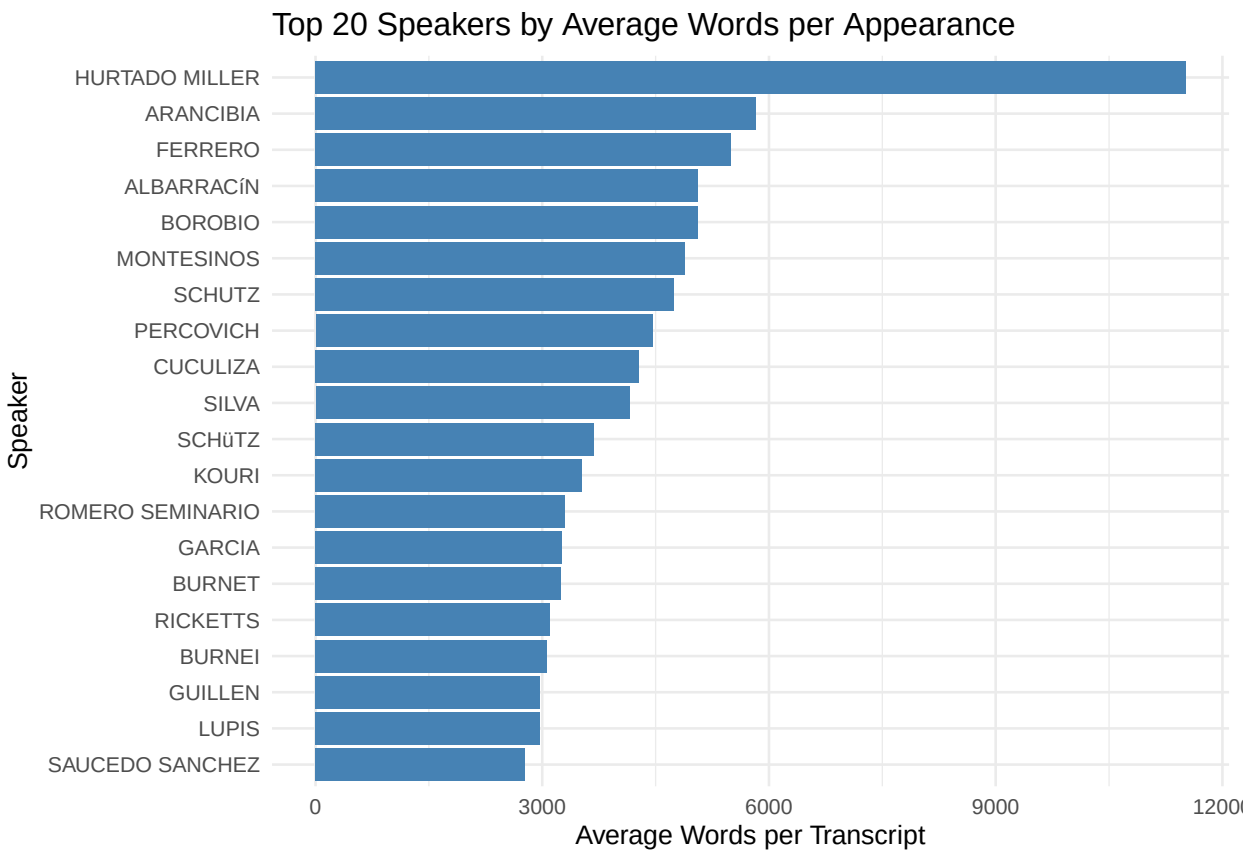
```
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
# Final summary
summary_df <- tibble(
  speaker_std = names(word_counts_list),
  total_words = unlist(word_counts_list),
  files_appeared = lengths(speaker_file_map),
  avg_words_per_file = total_words / files_appeared
)

# Sort by most talkative speakers
summary_df <- summary_df %>% arrange(desc(avg_words_per_file))

# Show result
print(summary_df)
```

```
## # A tibble: 471 x 4
##    speaker_std    total_words files_appeared avg_words_per_file
##    <chr>                <dbl>          <int>              <dbl>
##  1 HURTADO MILLER       23024              2             11512
##  2 ARANCIBIA             5826              1              5826
##  3 FERRERO              10987              2              5494.
##  4 ALBARRACíN           10118              2              5059
##  5 BOROBIO              20221              4              5055.
##  6 MONTESINOS          430344             88              4890.
##  7 SCHUTZ                4744              1              4744
##  8 PERCOVICH             4459              1              4459
##  9 CUCULIZA              8555              2              4278.
## 10 SILVA                 4154              1              4154
## # i 461 more rows
```

```
library(ggplot2)
```

```r
# Top 20 speakers by average words per transcript
top20 <- summary_df %>%
  slice_max(avg_words_per_file, n = 20) %>%
  mutate(speaker_std = fct_reorder(speaker_std, avg_words_per_file))

# Create bar plot
ggplot(top20, aes(x = speaker_std, y = avg_words_per_file)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(
    title = "Top 20 Speakers by Average Words per Appearance",
    x = "Speaker",
    y = "Average Words per Transcript"
  ) +
  theme_minimal(base_size = 10)
```



Average Conversation Length by Topic

```r
# Load required packages
library(readr)
library(dplyr)
library(tidyr)
library(ggplot2)
library(vistime)
library(lubridate)
```

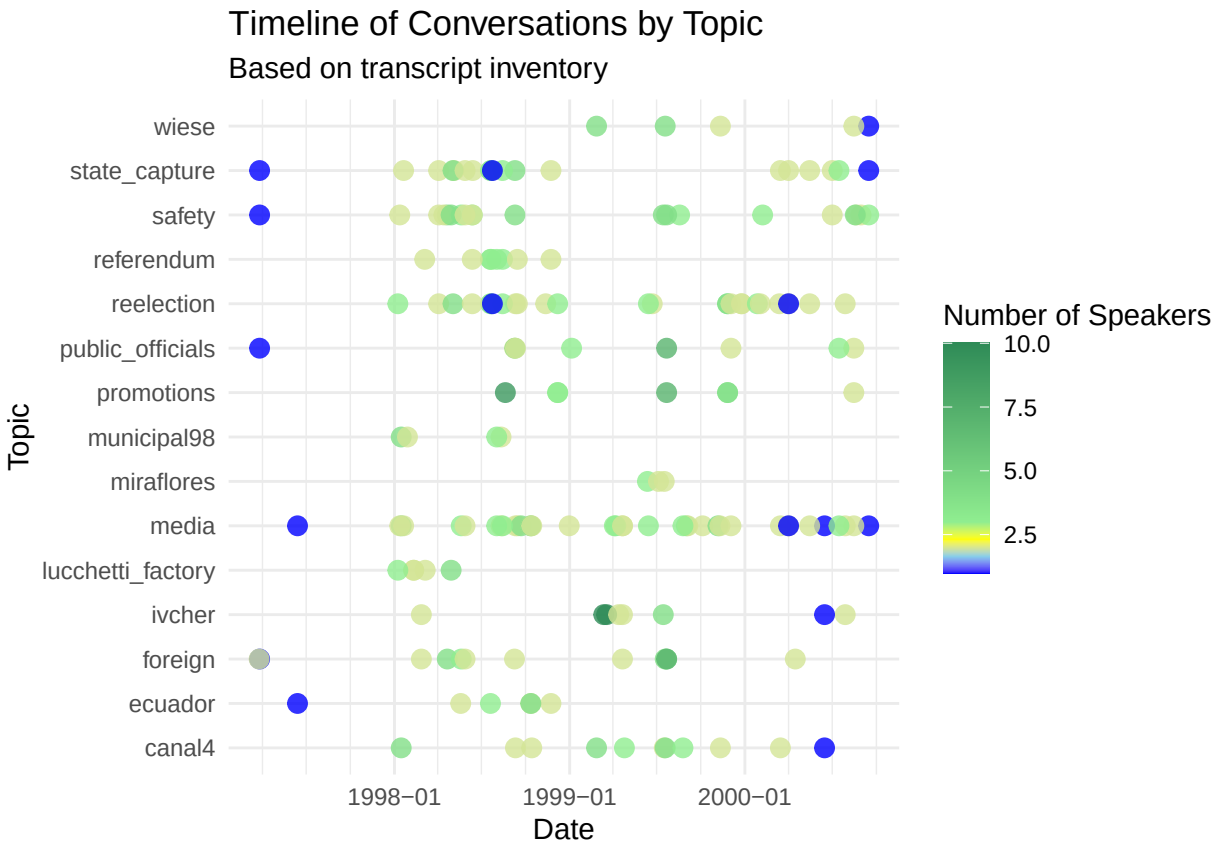
```r
library(stringr)  # For counting speakers

# Read the TSV file
file_path <- "../data/Inventory.csv"
data <- read_csv(file_path)
```

```
## Rows: 104 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (22): date, speakers, original_n, Missing Topic, in_book, in_online_arch...
## dbl  (1): n
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Convert 'date' column to Date format and filter out dates before 1990
data <- data %>%
  mutate(date = mdy(date)) %>%
  filter(!is.na(date) & date >= as.Date("1990-01-01"))

# Count number of speakers
data <- data %>%
  mutate(num_speakers = ifelse(is.na(speakers), 0, str_count(speakers, ",") + 1))

# Select topic columns and reshape into long format
topic_columns <- names(data)[grepl("^topic_", names(data))]

long_data <- data %>%
  select(n, date, speakers, num_speakers, all_of(topic_columns)) %>%
  pivot_longer(cols = all_of(topic_columns), names_to = "topic", values_to = "present") %>%
  filter(!is.na(present)) %>%
  mutate(topic = gsub("topic_", "", topic))  # Remove "topic_" prefix for clarity

# Define a custom gradient with multiple breakpoints
ggplot(long_data, aes(x = date, y = topic, color = num_speakers)) +
  geom_point(size = 3, alpha = 0.8) +
  scale_color_gradientn(colors = c("blue", "skyblue", "yellow", "lightgreen", "seagreen"),
                        values = scales::rescale(c(min(long_data$num_speakers, na.rm = TRUE),
                                                   quantile(long_data$num_speakers, 0.25, na.rm = TRUE)
                                                   median(long_data$num_speakers, na.rm = TRUE),
                                                   quantile(long_data$num_speakers, 0.75, na.rm = TRUE)
                                                   max(long_data$num_speakers, na.rm = TRUE)))) +
  scale_x_date(date_breaks = "1 year", date_minor_breaks = "3 months", date_labels = "%Y-%m") +
  labs(title = "Timeline of Conversations by Topic",
       subtitle = "Based on transcript inventory",
       x = "Date",
       y = "Topic",
       color = "Number of Speakers") +
  theme_minimal()
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



Summary Table

```
# Create a summary table counting occurrences of each unique value in "Type_Merged"
file <- "../data/Updated Inventory & Descriptions/Actors.csv"
df <- read_csv(file)
```

```
## New names:
## Rows: 153 Columns: 9
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (8): ...1, Position, Type, Montesinos' inner circle, speaker_std, comple... lgl
## (1): ...7
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
## * `` -> `...7`
## * `` -> `...9`
```

```
#Verify column names
colnames(df)
```

```
## [1] "...1"                      "Position"
## [3] "Type"                      "Montesinos' inner circle"
## [5] "speaker_std"               "completed_bio"
```

```
## [7] "...7"                    "notes"
## [9] "...9"
```

```r
# Ensure column names are clean
colnames(df) <- gsub(" ", "_", colnames(df))    # Replace spaces with underscores
colnames(df) <- gsub("'", "", colnames(df))     # Remove apostrophes

# Fix NA values: Ensure we replace only when necessary
df$Type_Merged <- ifelse(!is.na(df$Montesinos_inner_circle) & df$Montesinos_inner_circle == "X",
                         "Inner Circle",
                         ifelse(!is.na(df$Type), df$Type, NA))

# Verify the result
head(df[, c("Type", "Montesinos_inner_circle", "Type_Merged")])
```

```
## # A tibble: 6 x 3
##   Type             Montesinos_inner_circle Type_Merged
##   <chr>            <chr>                   <chr>
## 1 Security         <NA>                    Security
## 2 <NA>             <NA>                    <NA>
## 3 Congress         X                       Inner Circle
## 4 Businessperson   <NA>                    Businessperson
## 5 Congress         X                       Inner Circle
## 6 Elected Official X                       Inner Circle
```

```r
# Define output file path
output_file<-"../data/transcript_notes_cleaned.tsv"

# Write the cleaned dataset to a new CSV file
write_csv(df, output_file)

# Confirm the file was saved
message("Cleaned dataset saved as: ", output_file)
```

```
## Cleaned dataset saved as: ../data/transcript_notes_cleaned.tsv
```

```r
type_summary <- df %>%
  group_by(Type_Merged) %>%
  summarise(Count = n(), .groups = "drop") %>%
  arrange(desc(Count))

# Print summary table
print(type_summary)
```

```
## # A tibble: 12 x 2
##    Type_Merged      Count
##    <chr>            <int>
##  1 Inner Circle        33
##  2 <NA>                29
##  3 Security            25
##  4 Foreign             13
##  5 Bureaucrat          12
##  6 Media               12
##  7 Congress             9
##  8 Illicit              6
##  9 Judiciary            5
```

```
## 10 Businessperson       4
## 11 Elected Official     3
## 12 Illict               2
```

```r
library(readr)
library(dplyr)

# 1. Read and clean names as before
# read
df <- readr::read_csv("../data/Updated Inventory & Descriptions/Actors.csv")
```

```
## New names:
## Rows: 153 Columns: 9
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (8): ...1, Position, Type, Montesinos' inner circle, speaker_std, comple... lgl
## (1): ...7
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
## * `` -> `...7`
## * `` -> `...9`
```

```r
# clean column names
colnames(df) <- gsub(" ", "_", colnames(df))
colnames(df) <- gsub("'",   "",  colnames(df))


# 2. Build a new "Type_All" that always preserves Type
df <- df %>%
  mutate(
    Type_All = case_when(
      !is.na(Montesinos_inner_circle) & Montesinos_inner_circle == "X"
        ~ paste0(Type, " (Inner Circle)"),
      !is.na(Type)
        ~ Type,
      TRUE
        ~ NA_character_
    )
  )

# 3. (Optional) Inspect
head(df[, c("Type", "Montesinos_inner_circle", "Type_All")])
```

```
## # A tibble: 6 x 3
##   Type             Montesinos_inner_circle Type_All
##   <chr>            <chr>                   <chr>
## 1 Security         <NA>                    Security
## 2 <NA>             <NA>                    <NA>
## 3 Congress         X                       Congress (Inner Circle)
## 4 Businessperson   <NA>                    Businessperson
## 5 Congress         X                       Congress (Inner Circle)
## 6 Elected Official X                       Elected Official (Inner Circle)
```

```r
# 4. Write out cleaned data
write_csv(df, "../data/transcript_notes_inner_circle_cleaned.tsv")
```

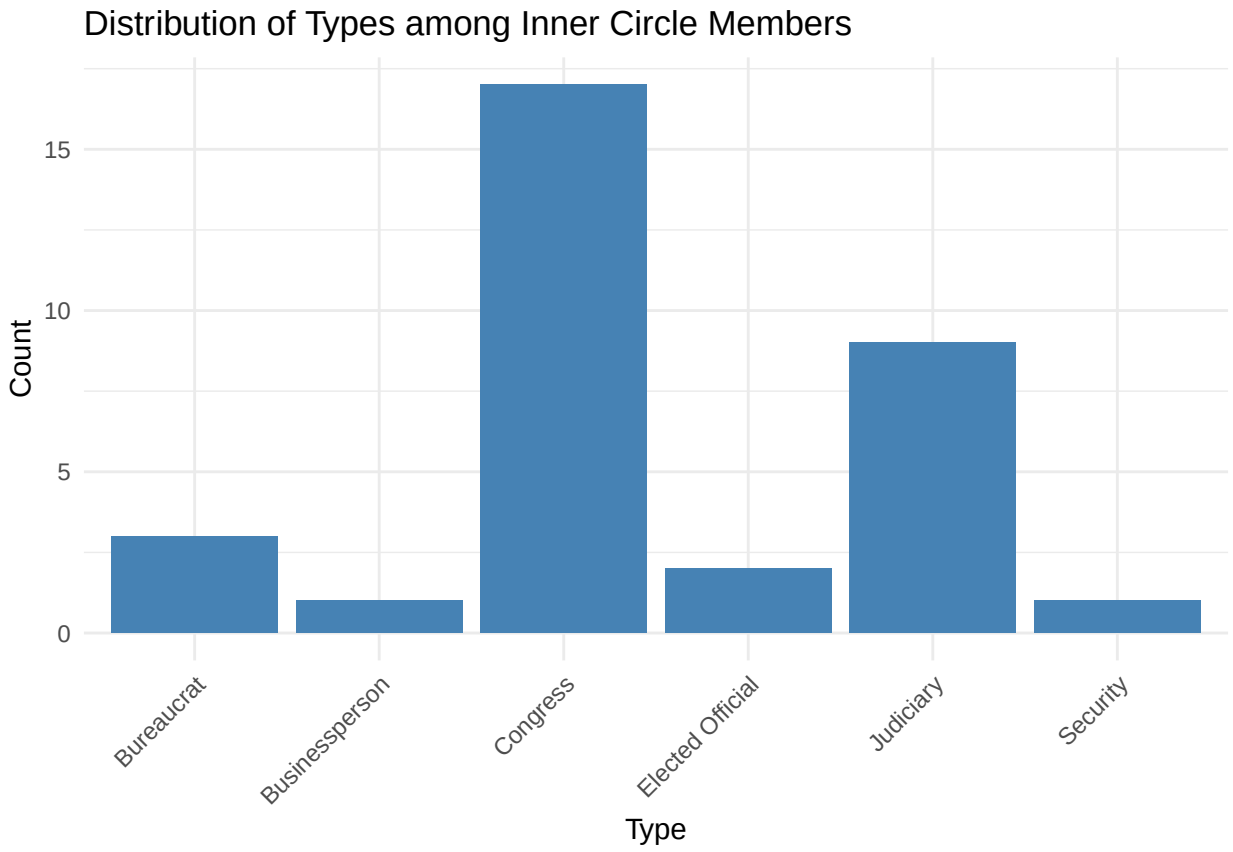
```r
message("Cleaned dataset saved as: ../data/transcript_notes_inner_circle_cleaned.tsv")
```

```
## Cleaned dataset saved as: ../data/transcript_notes_inner_circle_cleaned.tsv
```

```r
# 5. Summarize on the new column
type_summary <- df %>%
  group_by(Type_All) %>%
  summarise(Count = n(), .groups = "drop") %>%
  arrange(desc(Count))

print(type_summary)
```

```
## # A tibble: 17 x 2
##    Type_All                          Count
##    <chr>                             <int>
##  1 <NA>                                 29
##  2 Security                             25
##  3 Congress (Inner Circle)              17
##  4 Foreign                              13
##  5 Bureaucrat                           12
##  6 Media                                12
##  7 Congress                              9
##  8 Judiciary (Inner Circle)              9
##  9 Illicit                               6
## 10 Judiciary                             5
## 11 Businessperson                        4
## 12 Bureaucrat (Inner Circle)             3
## 13 Elected Official                      3
## 14 Elected Official (Inner Circle)       2
## 15 Illict                                2
## 16 Businessperson (Inner Circle)         1
## 17 Security (Inner Circle)               1
```

```r
library(dplyr)
library(ggplot2)

# 1. Subset to Inner Circle only
inner_df <- df %>%
  filter(!is.na(Montesinos_inner_circle) & Montesinos_inner_circle == "X")

# 2. Plot count of Types
ggplot(inner_df, aes(x = Type)) +
  geom_bar(fill = "steelblue") +                # bar plot of counts
  labs(
    title = "Distribution of Types among Inner Circle Members",
    x     = "Type",
    y     = "Count"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1)  # tilt labels if they overlap
  )
```




```
# 1) Read & clean actors data
actors_in  <- "../data/Updated Inventory & Descriptions/Actors.csv"
actors_out <- "../data/transcript_notes_cleaned.tsv"

actors_data <- read_csv(actors_in, col_types = cols()) %>%
  # normalize colnames
  rename_with(~ str_replace_all(.x, " ", "_")) %>%
  rename_with(~ str_replace_all(.x, "'", "")) %>%
  # create Type_Merged exactly as before
  mutate(
    Type_Merged = if_else(
      !is.na(Montesinos_inner_circle) & Montesinos_inner_circle == "X",
      "Inner Circle",
      Type
    )
  )
```

```
## New names:
## * `` -> `...1`
## * `` -> `...7`
## * `` -> `...9`
```

```
# save cleaned actors
write_csv(actors_data, actors_out)
message("Cleaned actors data saved to: ", actors_out)
```

```
## Cleaned actors data saved to: ../data/transcript_notes_cleaned.tsv
```

```r
# 2) Read all transcripts and count unique speaker appearances
transcript_dir   <- "../data/modified_data/finalized_data"
transcript_files <- list.files(transcript_dir, pattern="\\.(csv|tsv)$", full.names=TRUE)

read_transcript <- function(fp) {
  if (str_detect(fp, "\\.csv$")) read_csv(fp, col_types = cols())
  else                            read_tsv(fp, col_types = cols())
}

speaker_visits <- map_df(transcript_files, ~ {
  df <- read_transcript(.x)
  if ("speaker_std" %in% names(df)) {
    df %>% select(speaker_std) %>% distinct() %>%
      mutate(file = basename(.x))
  } else {
    tibble()  # skip if no speaker_std
  }
})
```

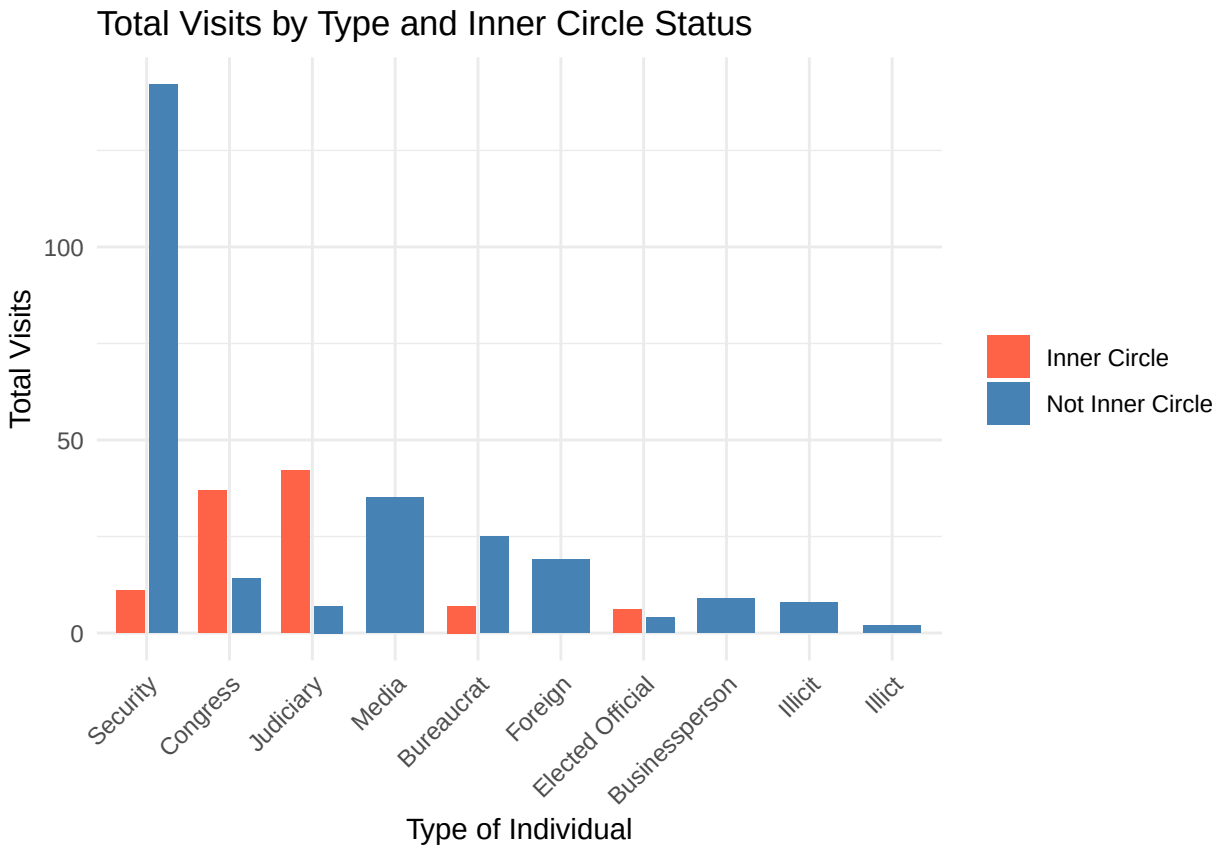
```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

```
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
```

```
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
speaker_counts <- speaker_visits %>%
  count(speaker_std, name = "visits")
```

```r
# 3) Merge counts with actors, flag status
merged_data <- speaker_counts %>%
  left_join(
    actors_data %>% select(speaker_std, Type, Montesinos_inner_circle),
    by = "speaker_std"
  ) %>%
  mutate(
    status = if_else(
      !is.na(Montesinos_inner_circle) & Montesinos_inner_circle == "X",
      "Inner Circle",
      "Not Inner Circle"
    )
  )

# 4) Summarize total visits by Type × status
visit_summary <- merged_data %>%
  filter(!is.na(Type)) %>%
  group_by(Type, status) %>%
  summarise(
    total_visits = sum(visits, na.rm = TRUE),
    .groups = "drop"
  )

# 5) Order Types by overall volume
type_order <- visit_summary %>%
  group_by(Type) %>%
  summarise(overall = sum(total_visits)) %>%
  arrange(desc(overall)) %>%
  pull(Type)

visit_summary <- visit_summary %>%
  mutate(Type = factor(Type, levels = type_order))

# 6) Plot: side-by-side bars, red = Inner Circle, blue = Not Inner Circle
ggplot(visit_summary, aes(x = Type, y = total_visits, fill = status)) +
  geom_col(position = position_dodge(width = 0.8), width = 0.7) +
  scale_fill_manual(
    values = c(
      "Inner Circle"     = "tomato",
      "Not Inner Circle" = "steelblue"
    )
  ) +
  labs(
    title = "Total Visits by Type and Inner Circle Status",
    x     = "Type of Individual",
    y     = "Total Visits",
    fill  = ""
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



#Visits By Type

```
library(tidyverse)
library(readr)
library(stringr)

# Load Actors.csv
actors <- read_csv("../data/Updated Inventory & Descriptions/Actors.csv") %>%
  mutate(Type = ifelse(Type == "Illict", "Illicit", Type)) %>%
  select(speaker_std, Type)
```

```
## New names:
## Rows: 153 Columns: 9
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (8): ...1, Position, Type, Montesinos' inner circle, speaker_std, comple... lgl
## (1): ...7
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
## * `` -> `...7`
## * `` -> `...9`
```

```
# Define transcript directory
transcript_dir <- "../data/modified_data/finalized_data"
transcript_files <- list.files(transcript_dir, pattern = "\\.(csv|tsv)$", full.names = TRUE)

# Function to read transcript
```

```r
read_transcript <- function(fp) {
  if (str_detect(fp, "\\.csv$")) read_csv(fp, col_types = cols()) else read_tsv(fp, col_types = cols())
}

# Count unique speaker_std appearances across files
speaker_visits <- map_df(transcript_files, ~ {
  df <- read_transcript(.x)
  if ("speaker_std" %in% names(df)) {
    df %>% select(speaker_std) %>% distinct() %>%
      mutate(file = basename(.x))
  } else {
    tibble()
  }
})
```

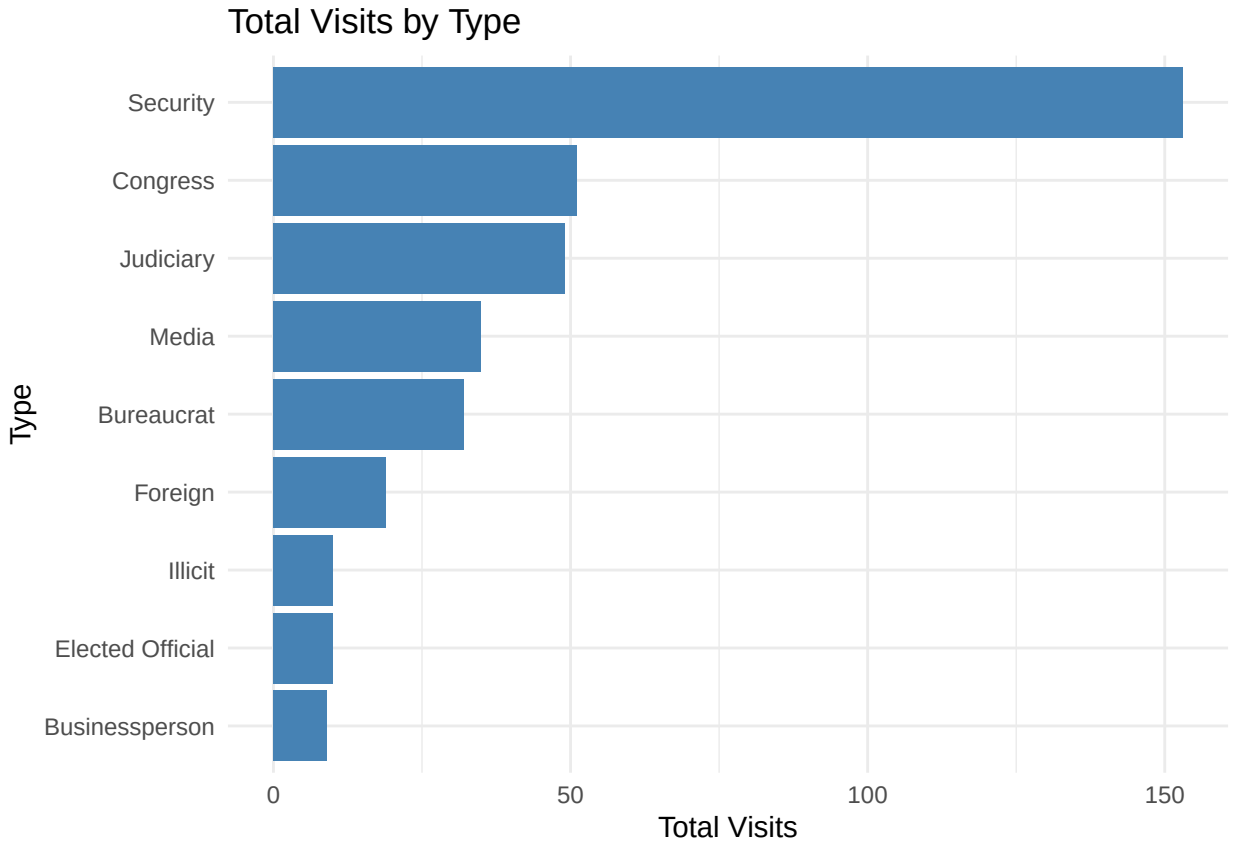
```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

```
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
```

```
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```r
# Count visits per speaker_std
speaker_counts <- speaker_visits %>%
  count(speaker_std, name = "visits")

# Join with Actors to get Type
type_visits <- speaker_counts %>%
```

```r
  left_join(actors, by = "speaker_std") %>%
  filter(!is.na(Type)) %>%
  group_by(Type) %>%
  summarise(total_visits = sum(visits), .groups = "drop")

# Plot: Total Visits by Type
ggplot(type_visits, aes(x = fct_reorder(Type, total_visits), y = total_visits)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(
    title = "Total Visits by Type",
    x = "Type",
    y = "Total Visits"
  ) +
  theme_minimal()
```




```r
# Reload actors with typo fixed again (if needed downstream)
actors <- read_csv("../data/Updated Inventory & Descriptions/Actors.csv") %>%
  mutate(Type = ifelse(Type == "Illict", "Illicit", Type))
```

```
## New names:
## Rows: 153 Columns: 9
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (8): ...1, Position, Type, Montesinos' inner circle, speaker_std, comple... lgl
## (1): ...7
## i Use `spec()` to retrieve the full column specification for this data. i
```

```
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
## * `` -> `...7`
## * `` -> `...9`
```

```r
# Count unique individuals per Type
type_counts <- actors %>%
  filter(!is.na(Type), !is.na(speaker_std)) %>%
  distinct(speaker_std, Type) %>%
  count(Type, name = "num_individuals")

# Plot: Number of Individuals per Type
ggplot(type_counts, aes(x = fct_reorder(Type, num_individuals), y = num_individuals)) +
  geom_col(fill = "darkgreen") +
  coord_flip() +
  labs(
    title = "Number of Individuals per Type",
    x = "Type",
    y = "Number of Unique Individuals"
  ) +
  theme_minimal()
```