visualizations_knit.rmd

2025-03-06

Topic Frequency Histogram

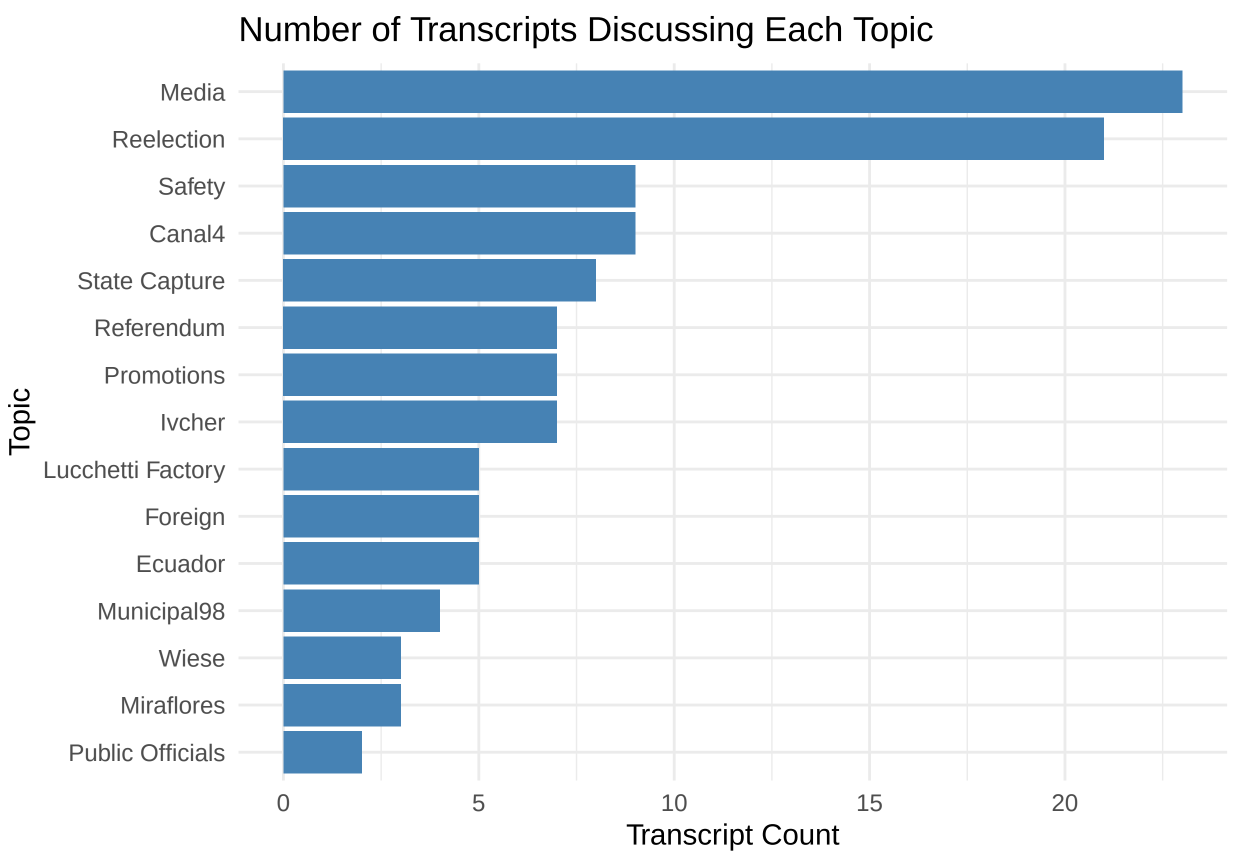
```r
#Frequency By Topic test


# 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
```

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

# Step 1: Read the inventory file
file <- "../data/Copy of inventory - Transcript inventory.tsv"
inventory <- read_tsv(file)
```

```
## Rows: 104 Columns: 22

## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (21): date, speakers, original_n, in_book, in_online_archive, type, topi...
## 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
# 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()
```

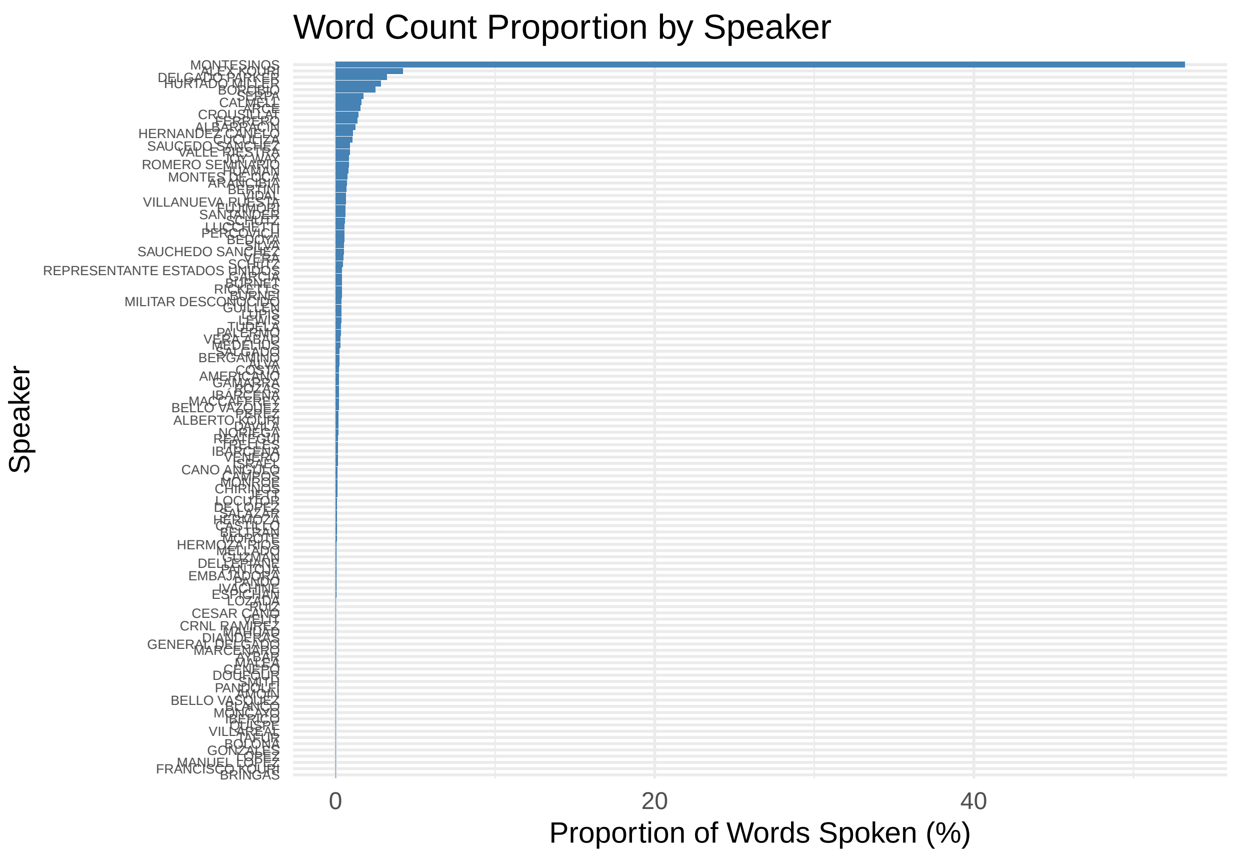


Word Count Proportion (excluding 'desconoocido')

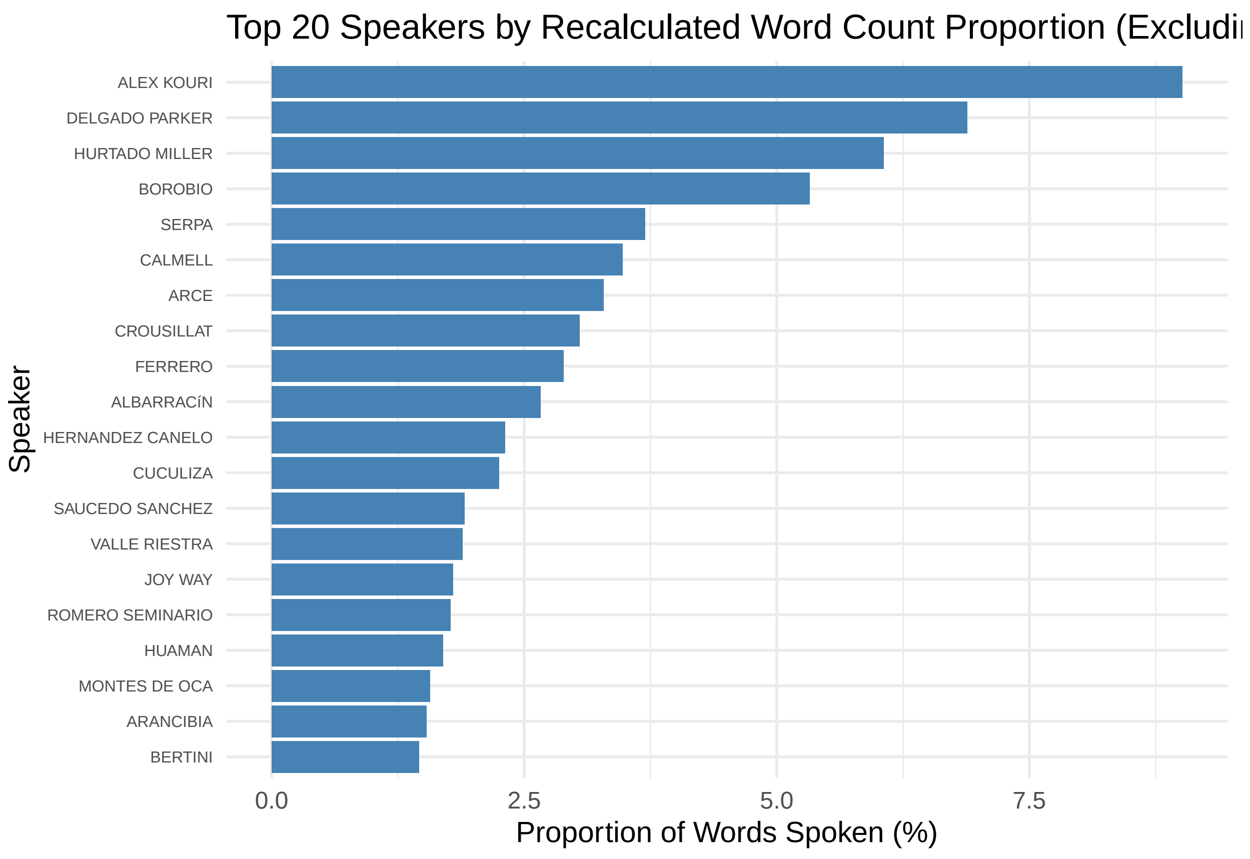
```r
#install.packages("readr")
#install.packages("dplyr")
#install.packages("ggplot2")

library(readr)
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))
```



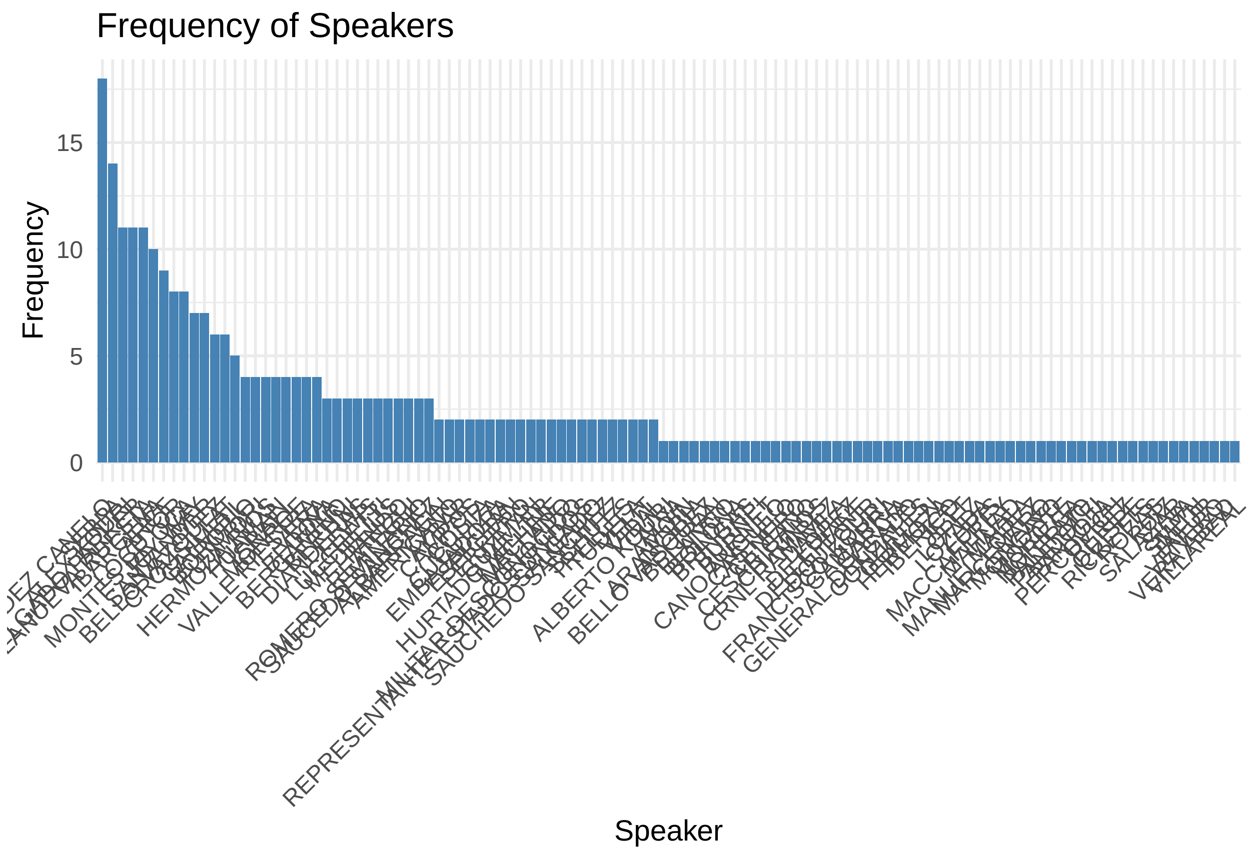
Speaker Frequency

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

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

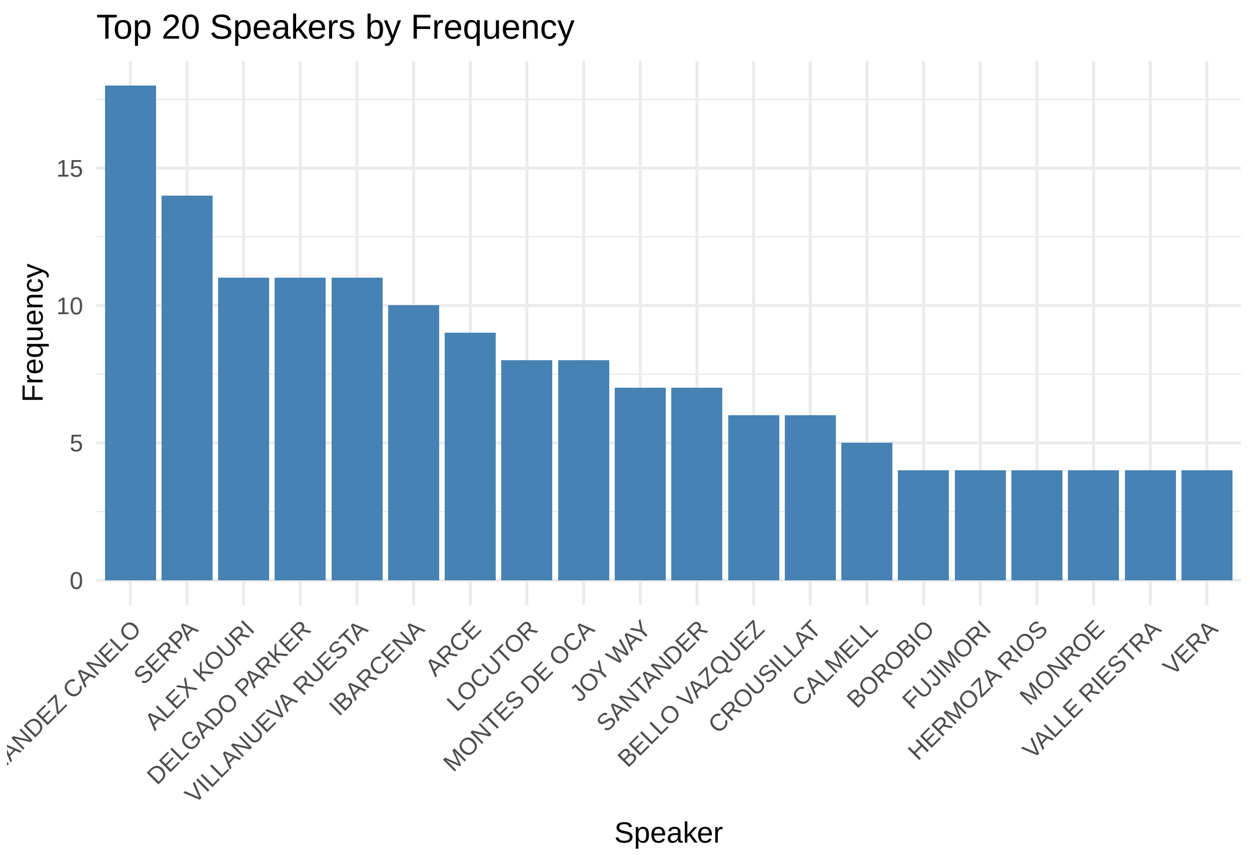
```
## Rows: 114 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Speaker
## dbl (1): Frequency
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types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

```r
  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))
```



```r
# 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))
```



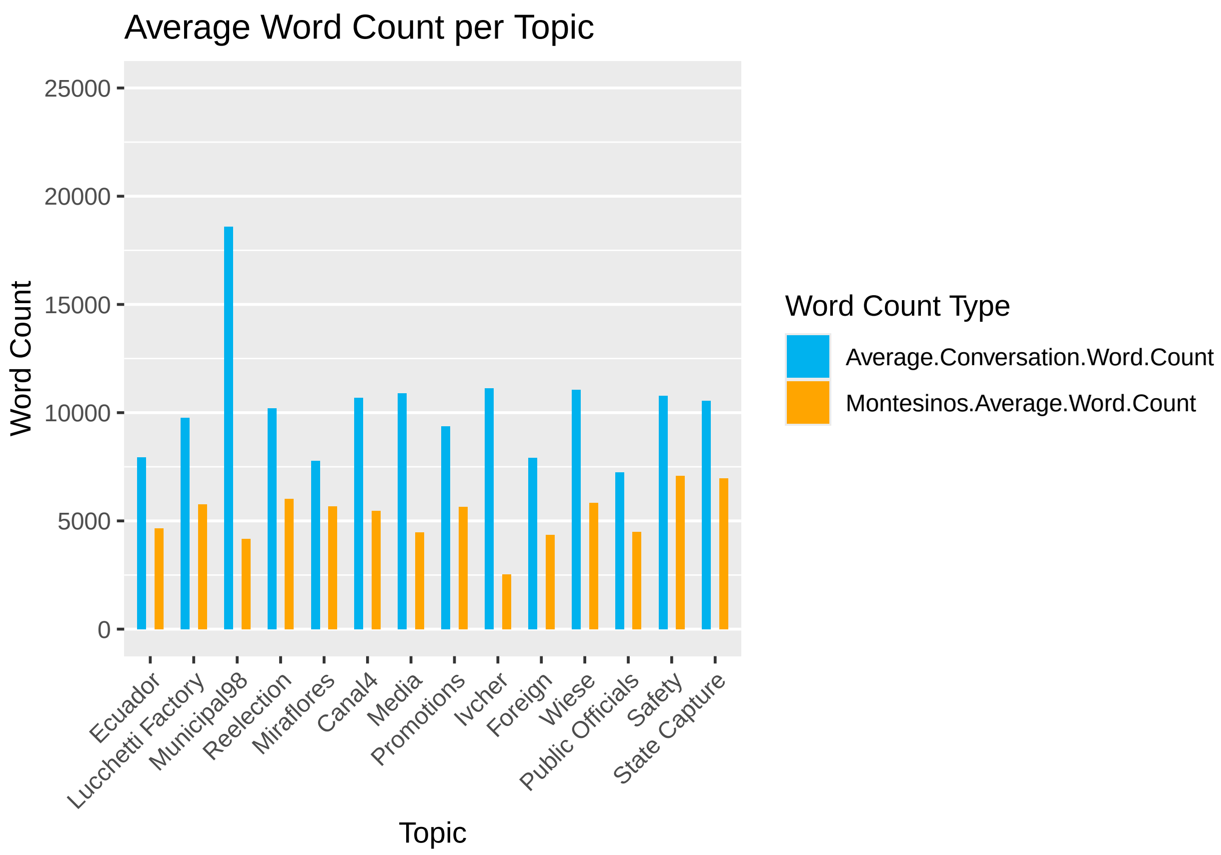
Word Count Per Topic

```r
# 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 error
```

```r
# Read in 'topic_vmt_avg_count.csv' file
file<-"../output/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

```
# 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
## File: 11.tsv - Word Count: 9600
## File: 12.tsv - Word Count: 1298
## File: 14.tsv - Word Count: 955
## File: 15.tsv - Word Count: 12607
## File: 16.tsv - Word Count: 13624
## File: 17.tsv - Word Count: 11547
## File: 20.tsv - Word Count: 16825
## File: 21.tsv - Word Count: 17199
## File: 22.tsv - Word Count: 2219
## File: 23.tsv - Word Count: 7675
## File: 26.tsv - Word Count: 11370
## File: 27.tsv - Word Count: 17701
## File: 28.tsv - Word Count: 10296
## File: 29.tsv - Word Count: 7865
## File: 30.tsv - Word Count: 6109
## File: 31.tsv - Word Count: 6473
## File: 32.tsv - Word Count: 11795
## File: 33.tsv - Word Count: 20880
## File: 34.tsv - Word Count: 29157
## File: 35.tsv - Word Count: 18275
## File: 36.tsv - Word Count: 11582
## File: 40.tsv - Word Count: 17333
## File: 42.tsv - Word Count: 3673
## File: 43.tsv - Word Count: 1948
## File: 44.tsv - Word Count: 8323
## File: 45.tsv - Word Count: 4270
## File: 46.tsv - Word Count: 8012
## File: 48.tsv - Word Count: 3903
## File: 49.tsv - Word Count: 12028
## File: 5.tsv - Word Count: 9392
## File: 50.tsv - Word Count: 18049
## File: 51.tsv - Word Count: 5050
## File: 52.tsv - Word Count: 17176
## File: 53.tsv - Word Count: 9266
## File: 54.tsv - Word Count: 28813
## File: 56.tsv - Word Count: 10738
## File: 57.tsv - Word Count: 14216
## File: 58.tsv - Word Count: 4774
## File: 59.tsv - Word Count: 8638
## File: 60.tsv - Word Count: 12628
## File: 61.tsv - Word Count: 3378
## File: 62.tsv - Word Count: 12784
## File: 65.tsv - Word Count: 2824
## File: 66.tsv - Word Count: 7629
## File: 68.tsv - Word Count: 9854
## File: 7.tsv - Word Count: 11146
## File: 70.tsv - Word Count: 19659
## File: 71.tsv - Word Count: 11440
## File: 72.tsv - Word Count: 4484
## File: 73.tsv - Word Count: 3875
## File: 76.tsv - Word Count: 9166
## File: 77.tsv - Word Count: 11649
## File: 78.tsv - Word Count: 10546
## File: 82.tsv - Word Count: 13670
## File: 83.tsv - Word Count: 4086
## File: 85.tsv - Word Count: 1853
## File: 86.tsv - Word Count: 5690
## File: 9.tsv - Word Count: 16843
```

```r
# 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: 903734
```

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

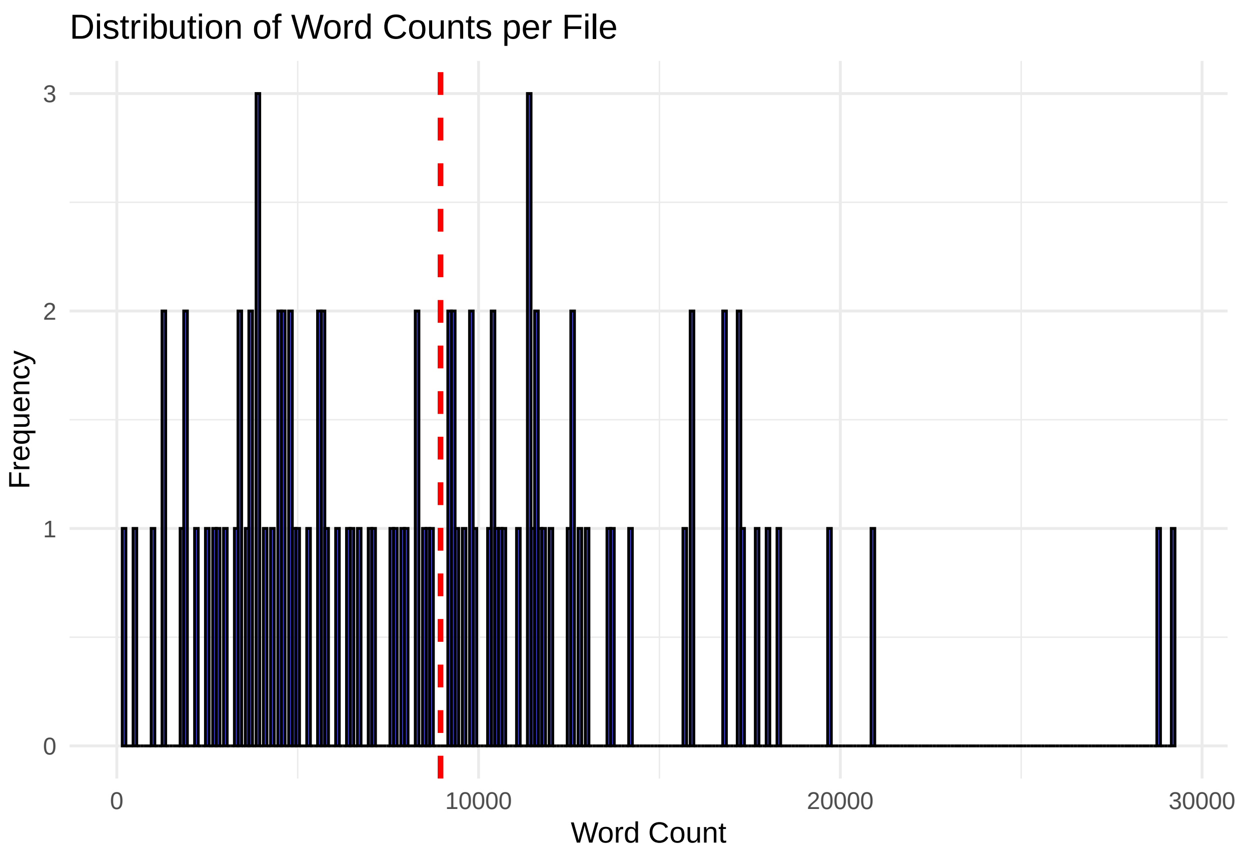
```
## Number of Files Processed: 101
```

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

```
## Average Word Count per File: 8947.861
```

```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))
  }
}

# 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.
```

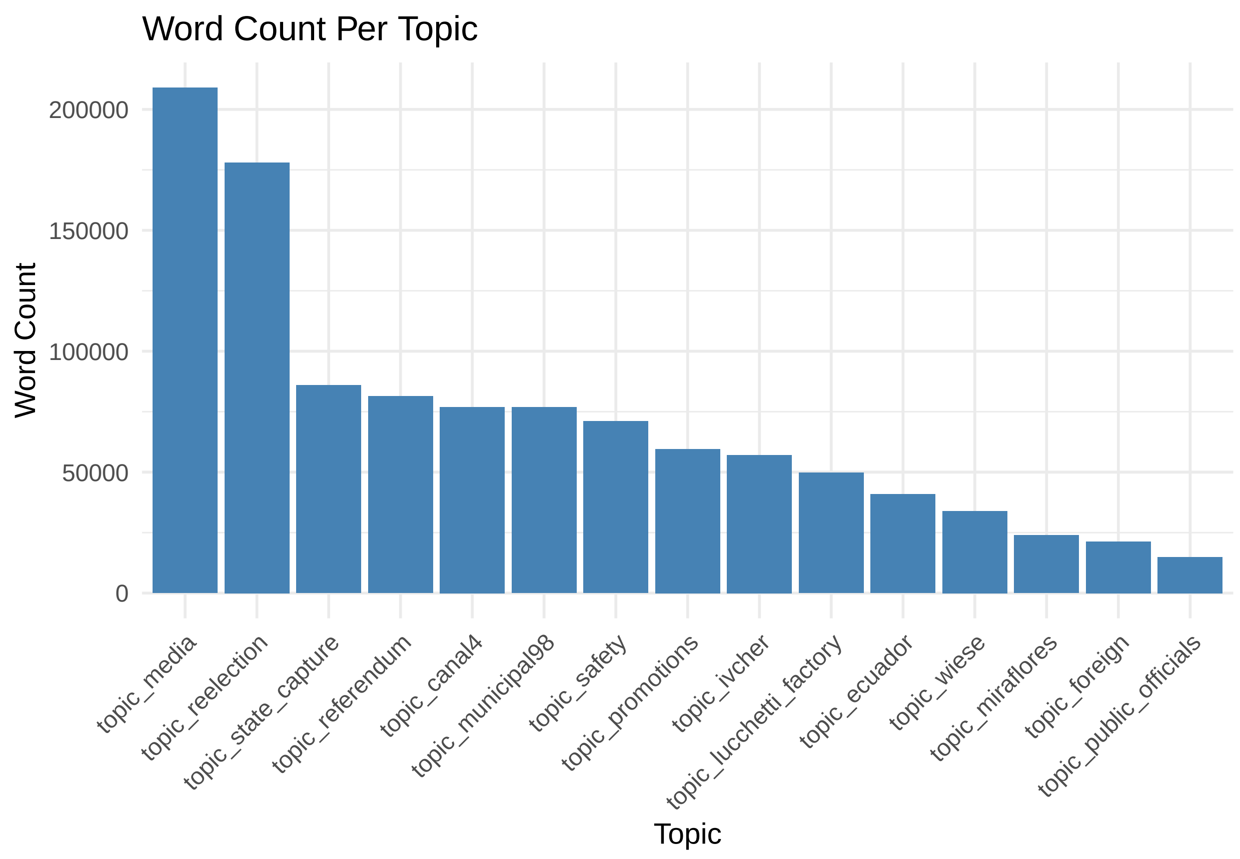


Average Conversation Length by Topic

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

# Define file paths
inventory_file <- "../data/Copy of inventory - Transcript inventory.tsv"
directory_path <- "../data/modified_data/finalized_data"

# Read the inventory data
inventory_data <- read_tsv(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
  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
    }
  }
}

# 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      81395
## topic_ecuador                     topic_ecuador      40898
## topic_lucchetti_factory topic_lucchetti_factory      49722
## topic_municipal98               topic_municipal98      76887
## topic_reelection                    topic_reelection     178073
## topic_miraflores                    topic_miraflores      23871
## topic_canal4                            topic_canal4      76911
## topic_media                              topic_media     208956
## topic_promotions                    topic_promotions      59562
## topic_ivcher                            topic_ivcher      57069
## topic_foreign                          topic_foreign      21316
## topic_wiese                              topic_wiese      33971
## topic_public_officials        topic_public_officials      14904
## topic_safety                            topic_safety      71067
## topic_state_capture              topic_state_capture      85990
```

```
# 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))
```



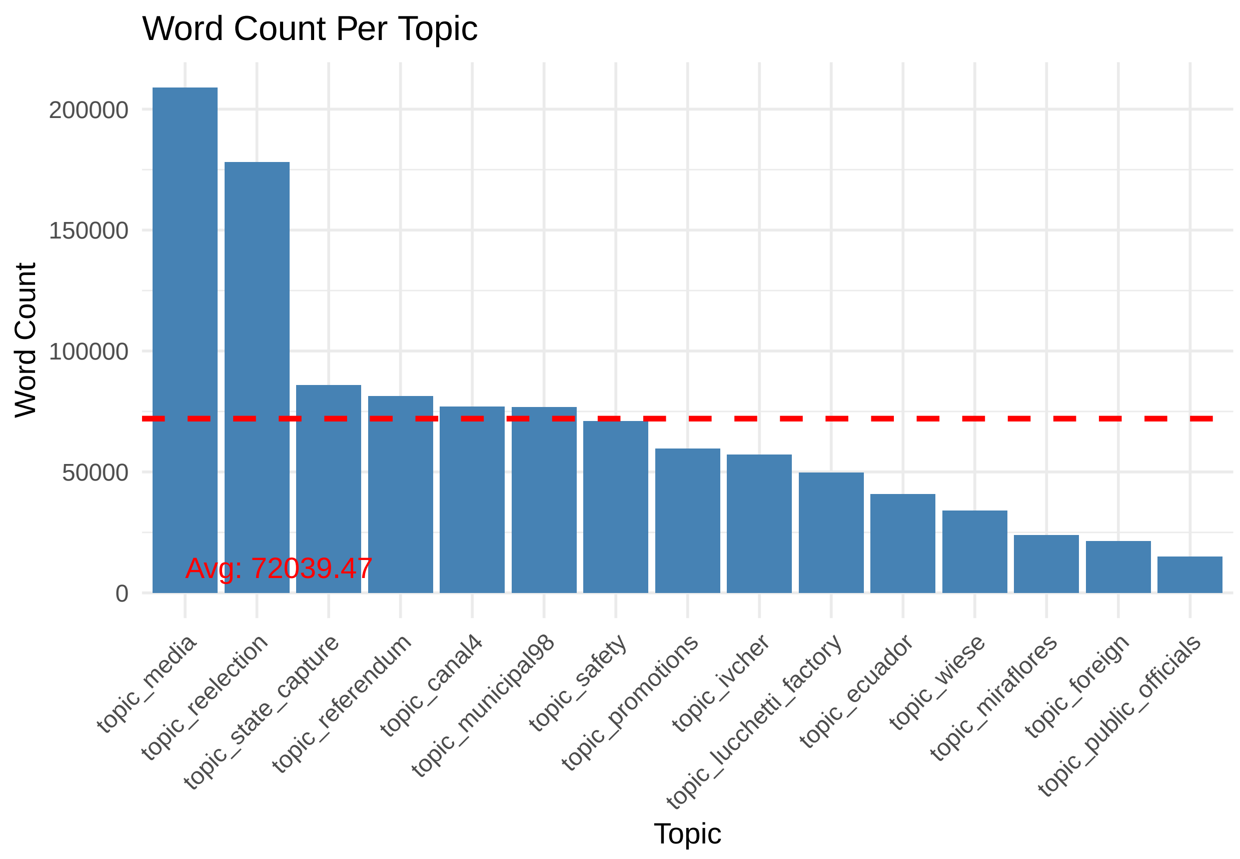
```
# Calculate the average word count across all transcripts
average_word_count_all <- sum(word_count_df$Word_Count) / nrow(inventory_data)

# 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: 10390.31"
```

```r
# 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: 72039.47"
```

```r
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_co
```



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

```
## [1] "Total Word Count Across All Transcripts: 1080592"
```

Bar Plot

```r
# Define the path to the finalized_data folder (relative to current working directory)
finalized_data_path <- "../data/modified_data/finalized_data"

# Load necessary libraries
#install.packages("readr")
```

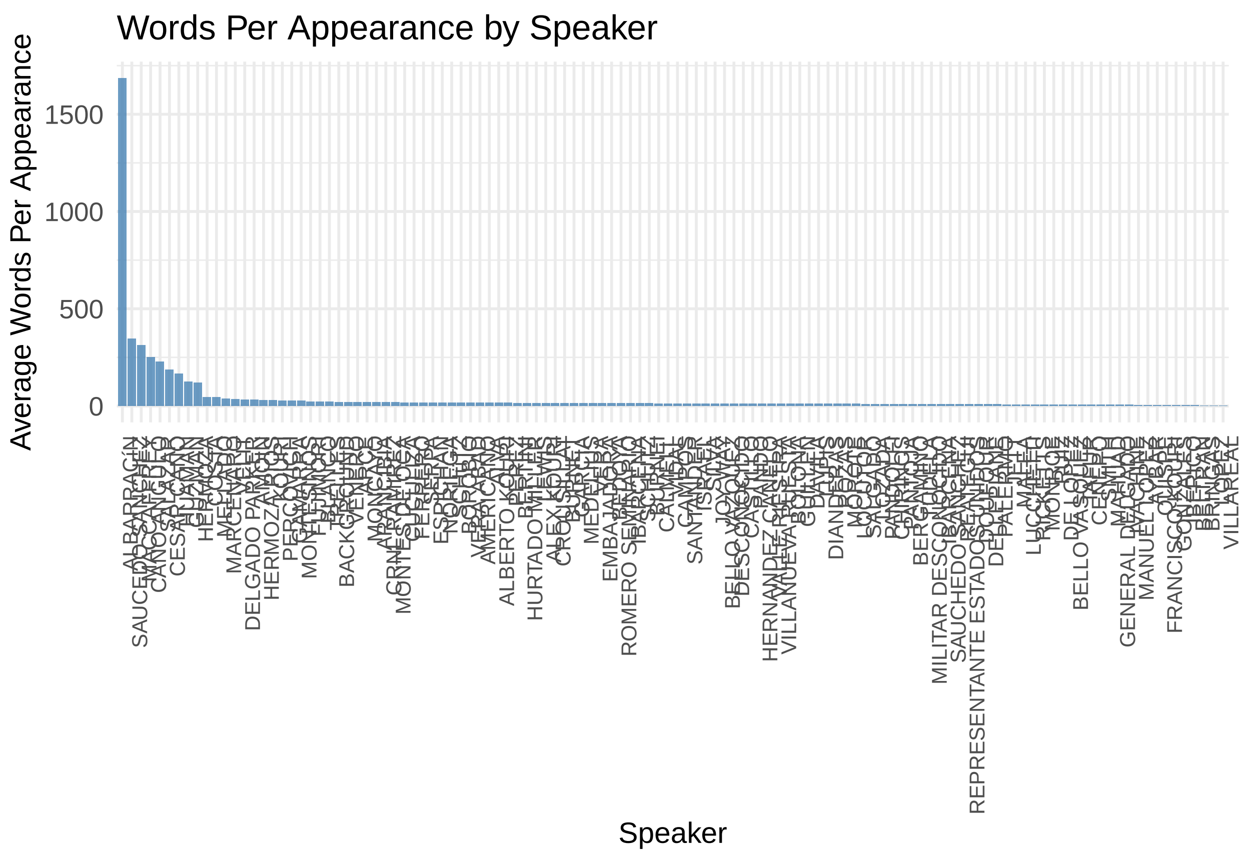
```r
#install.packages("stringr")
#install.packages("dplyr")
#install.packages("ggplot2")

library(readr)   # For reading CSV & TSV files
library(stringr) # For text processing
library(dplyr)   # For data manipulation
library(ggplot2) # For visualization

# Get a list of CSV and TSV files in the finalized_data directory
csv_files <- list.files(path = finalized_data_path, pattern = "\\.csv$", full.names = TRUE)
tsv_files <- list.files(path = finalized_data_path, pattern = "\\.tsv$", full.names = TRUE)

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

# 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
}

# Initialize an empty list to store data frames
all_speaker_data <- list()

# Loop through each file and accumulate results
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)
  })

  # Proceed only if the file was successfully read and contains required columns
  if (!is.null(df) && all(c("speech", "speaker_std") %in% colnames(df))) {

    # Process data to count words per speaker
    df <- df %>%
      filter(!is.na(speech) & !is.na(speaker_std)) %>%
      group_by(speaker_std) %>%
      summarise(
        Total_Words = sum(count_words(speech)),
        Appearances = n(),
        .groups = "drop"
      )
    # Store the processed data in a list
    all_speaker_data[[basename(file)]] <- df  # Use basename(file) for readability
  } else {
    cat("Skipping file (missing required columns):", basename(file), "\n")
  }
}

# Combine all accumulated data into a single data frame
if (length(all_speaker_data) > 0) {
  speaker_word_counts <- bind_rows(all_speaker_data) %>%
    group_by(Speaker = speaker_std) %>%
    summarise(
      Total_Words = sum(Total_Words),
      Appearances = sum(Appearances),
      .groups = "drop"
    ) %>%
    mutate(Average_Words_Per_Appearance = Total_Words / Appearances)
} else {
  speaker_word_counts <- data.frame()
  cat("No valid data found in any files.\n")
}

# Print summary
cat("Total Unique Speakers:", nrow(speaker_word_counts), "\n")
```

```
## Total Unique Speakers: 118
```

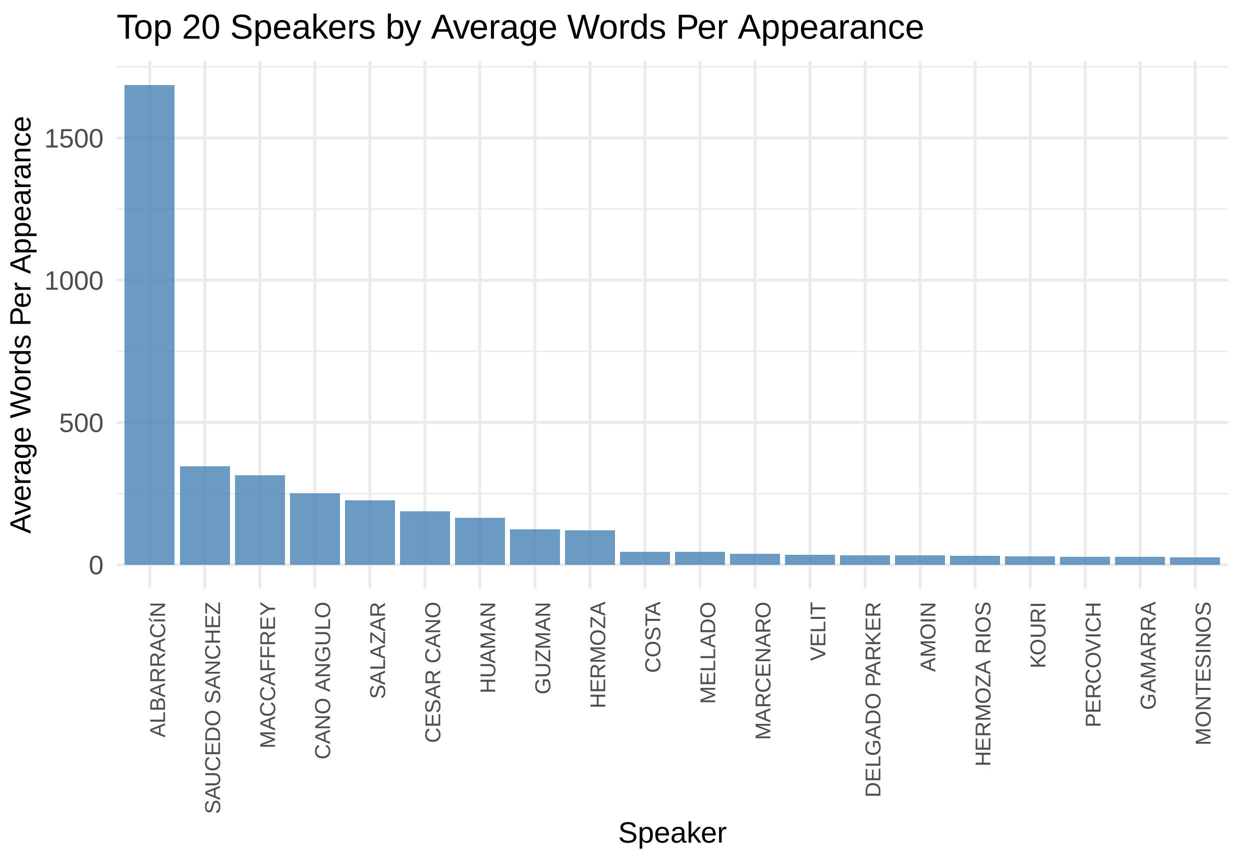
```r
# Display the final aggregated results
print(speaker_word_counts)
```

```
## # A tibble: 118 x 4
##    Speaker       Total_Words Appearances Average_Words_Per_Appearance
##    <chr>               <int>       <int>                        <dbl>
##  1 ALBARRACíN          10118           6                       1686.
##  2 ALBERTO KOURI        1406          88                         16.0
##  3 ALEX KOURI           9550         634                         15.1
##  4 ALVA                 1892         117                         16.2
##  5 AMERICANO            1708         105                         16.3
##  6 AMOIN                  32           1                         32
##  7 ARANCIBIA            5826         297                         19.6
##  8 ARCE                12493         618                         20.2
##  9 AYBAR                  74          15                          4.93
## 10 BACKGROUND          31521        1546                         20.4
## # i 108 more rows
```

```r
# Sort data by Average Words Per Appearance (Descending)
speaker_word_counts <- speaker_word_counts %>%
  arrange(desc(Average_Words_Per_Appearance))

# Create the bar plot with all speakers on the x-axis
ggplot(speaker_word_counts, aes(x = reorder(Speaker, -Average_Words_Per_Appearance), y = Average_Words_
  geom_bar(stat = "identity", fill = "steelblue", alpha = 0.8) +
  labs(title = "Words Per Appearance by Speaker",
       x = "Speaker",
       y = "Average Words Per Appearance") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 90, hjust = 1, size = 8),  # Rotate x-axis labels for readabilit
    axis.text.y = element_text(size = 10)  # Keep y-axis labels readable
  )
```



```
# Create the bar plot showing only the top 20 speakers by average words per appearance
top20_speakers <- speaker_word_counts %>%
  slice_max(Average_Words_Per_Appearance, n = 20)

ggplot(top20_speakers, aes(x = reorder(Speaker, -Average_Words_Per_Appearance), y = Average_Words_Per_A
  geom_bar(stat = "identity", fill = "steelblue", alpha = 0.8) +
  labs(title = "Top 20 Speakers by Average Words Per Appearance",
       x = "Speaker",
       y = "Average Words Per Appearance") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 90, hjust = 1, size = 8),  # Rotate x-axis labels for readabilit
    axis.text.y = element_text(size = 10)  # Keep y-axis labels readable
  )
```



Average Conversation Length by Topic

```r
# Load required packages
library(readr)
library(dplyr)
library(tidyr)
library(ggplot2)
library(vistime)
library(lubridate)
library(stringr)  # For counting speakers

# Read the TSV file
file_path <- "../data/Copy of inventory - Transcript inventory.tsv"
data <- read_tsv(file_path)
```

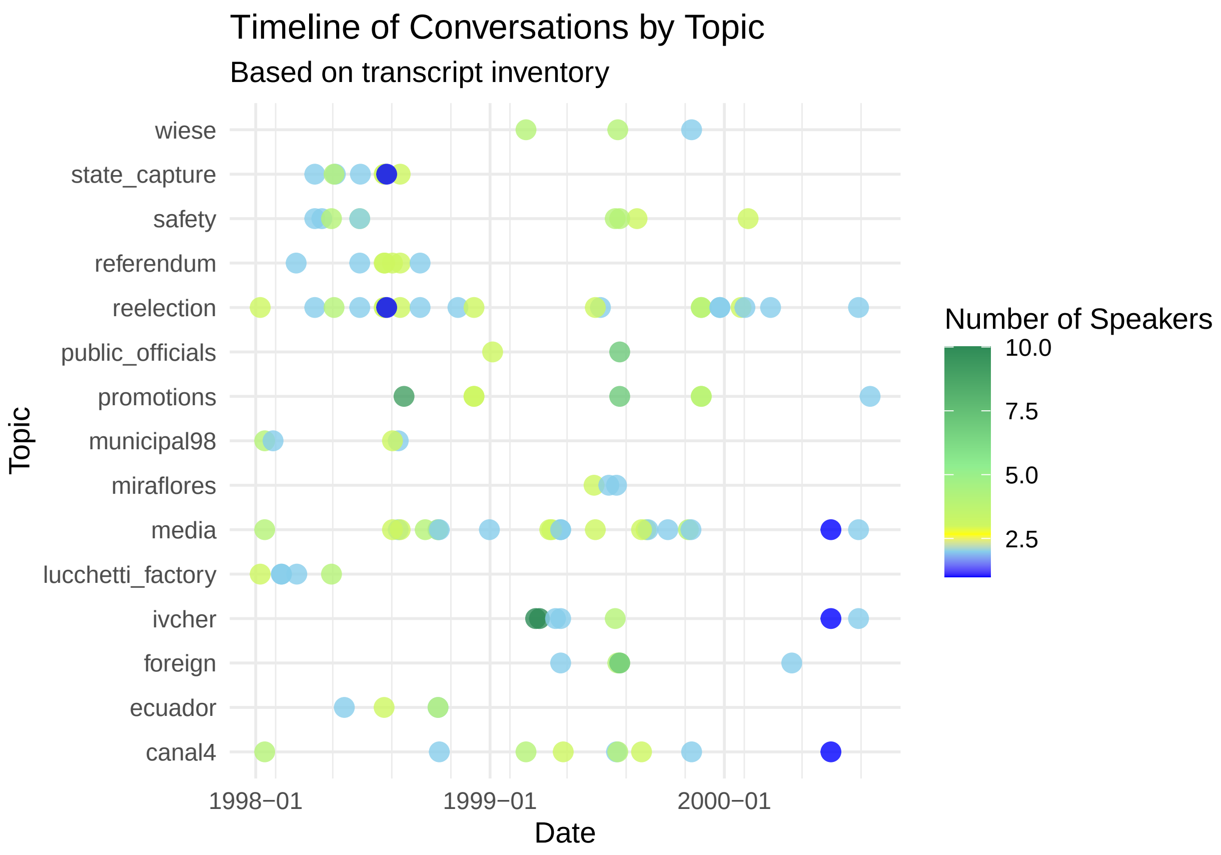
```
## Rows: 104 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (21): date, speakers, original_n, in_book, in_online_archive, type, topi...
## dbl  (1): n
##
## i Use `spec()` to retrieve the full column specification for this 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"))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `date = mdy(date)`.
## Caused by warning:
## !  1 failed to parse.
```

```
# 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/Copy of Master - transcript notes - Actors.tsv"
df <- read_tsv(file)
```

```
## New names:
## Rows: 93 Columns: 9
## -- Column specification
## -------------------------------------------------------- Delimiter: "\t" 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 Government     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: 14 x 2
##    Type_Merged                     Count
##    <chr>                           <int>
##  1 Inner Circle                       33
##  2 <NA>                               16
##  3 Military or Police                 11
##  4 Media                               8
##  5 Government                          6
##  6 Congress                            4
##  7 Foreign                             4
##  8 Businessperson                      3
##  9 Security                            2
## 10 Security rival                      2
## 11 Ally                                1
## 12 Front man                           1
## 13 Government, Judiciary               1
## 14 Military or Police, Government      1
```

```
library(readr)
library(dplyr)

# 1. Read and clean names as before
# read
df <- readr::read_tsv("../data/Copy of Master - transcript notes - Actors.tsv")
```

```
## New names:
## Rows: 93 Columns: 9
## -- Column specification
## -------------------------------------------------------- Delimiter: "\t" 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`
```

```
# 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 Government     X                       Government (Inner Circle)
```

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

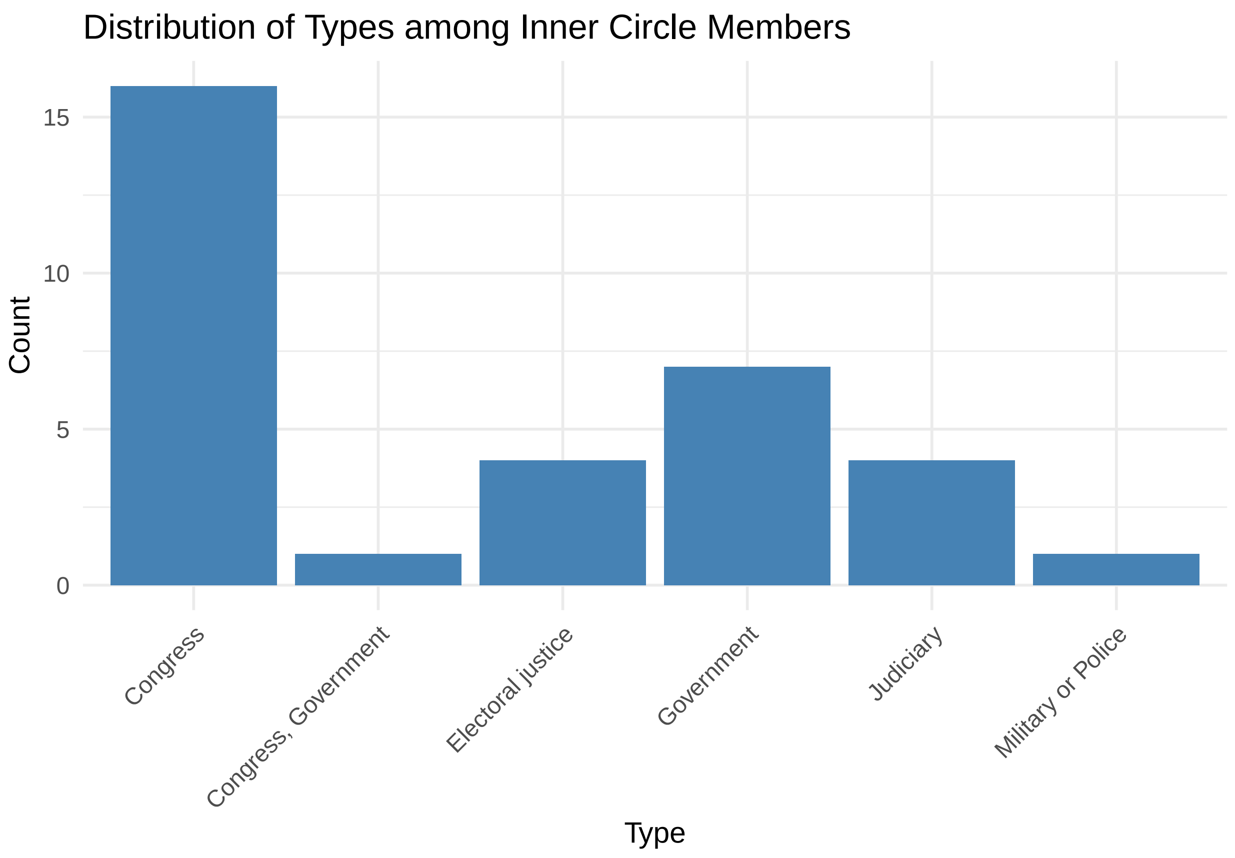
```r
message("Cleaned dataset saved as: ../data/transcript_notes_inner_circle_cleaned.tsv")
```

```
## 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: 19 x 2
##    Type_All                            Count
##    <chr>                               <int>
##  1 Congress (Inner Circle)                16
##  2 <NA>                                   16
##  3 Military or Police                     11
##  4 Media                                   8
##  5 Government (Inner Circle)               7
##  6 Government                              6
##  7 Congress                                4
##  8 Electoral justice (Inner Circle)        4
##  9 Foreign                                 4
## 10 Judiciary (Inner Circle)                4
## 11 Businessperson                          3
## 12 Security                                2
## 13 Security rival                          2
## 14 Ally                                    1
## 15 Congress, Government (Inner Circle)     1
## 16 Front man                               1
## 17 Government, Judiciary                   1
## 18 Military or Police (Inner Circle)       1
## 19 Military or Police, Government          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
  )
```



Histograms (Vists by type & Inner Circle V Outer Circle)

```r
# Load required libraries
install.packages("tidyverse")
```

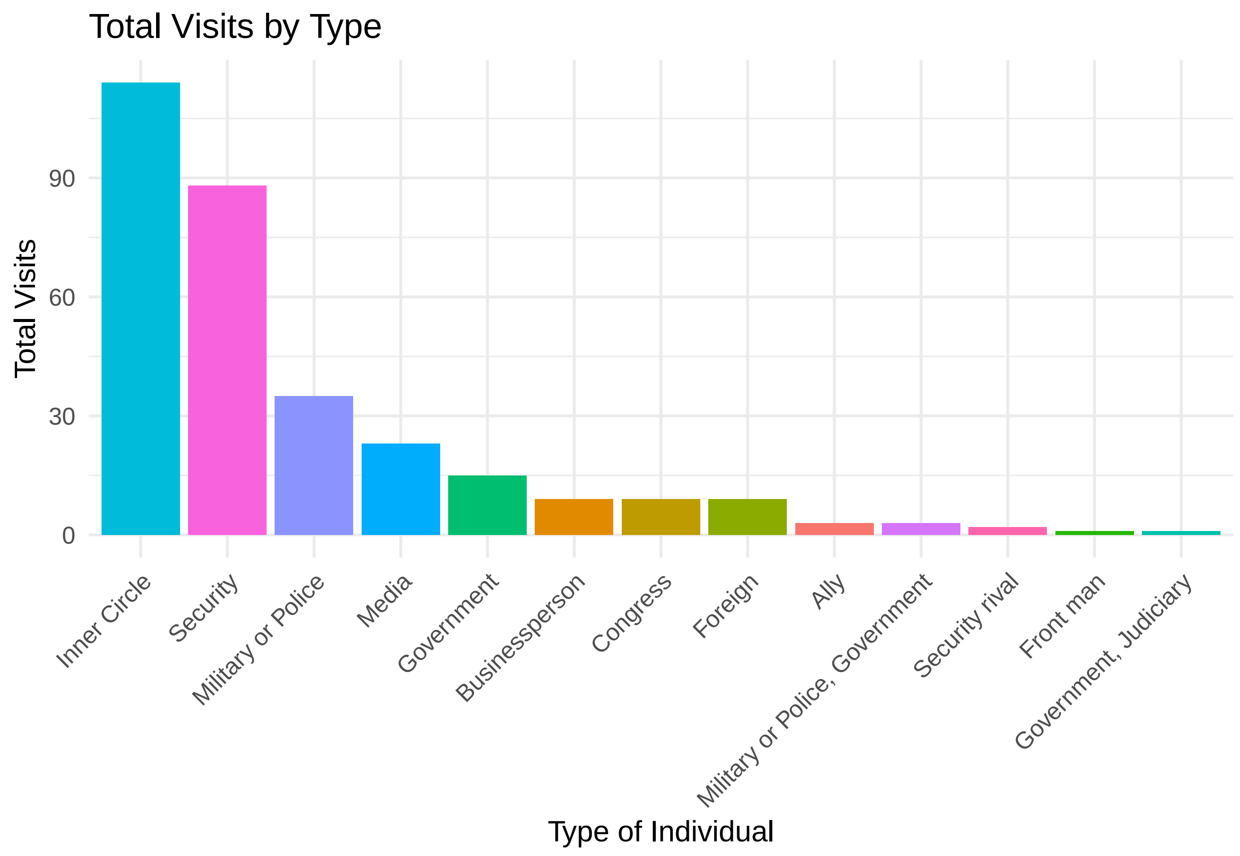
```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
library(tidyverse)

# Set the directory where transcript files are stored
transcript_dir <- "../data/modified_data/finalized_data"

# Get list of all CSV and TSV files
transcript_files <- list.files(path = transcript_dir, pattern = "\\.(csv|tsv)$", full.names = TRUE)

# Function to detect delimiter and read file correctly
read_transcript_file <- function(file_path) {
  if (grepl("\\.csv$", file_path)) {
    return(read_csv(file_path, col_types = cols()))
  } else if (grepl("\\.tsv$", file_path)) {
    return(read_tsv(file_path, col_types = cols()))
  } else {
    return(NULL)  # In case of an unexpected format
  }
}
```

```r
# Initialize an empty dataframe to store speaker visits
speaker_visits <- tibble(speaker_std = character(), file_name = character())

# Loop through each transcript file and extract unique speaker_std values
for (file in transcript_files) {
  # Read file using the appropriate function
  df <- read_transcript_file(file)

  # Check if 'speaker_std' column exists in the file
  if (!is.null(df) && "speaker_std" %in% colnames(df)) {
    # Store unique speakers per file
    unique_speakers <- df %>%
      select(speaker_std) %>%
      distinct() %>%
      mutate(file_name = basename(file))

    # Append to the master visits dataframe
    speaker_visits <- bind_rows(speaker_visits, unique_speakers)
  }
}

# Count unique appearances per speaker (number of transcript files they appear in)
speaker_counts <- speaker_visits %>%
  group_by(speaker_std) %>%
  summarize(visits = n(), .groups = "drop")

# Load the master actors dataset
actors_file <- "../data/transcript_notes_cleaned.tsv"
actors_data <- read_csv(actors_file, col_types = cols())

# Clean column names to match modifications
colnames(actors_data) <- colnames(actors_data) %>% str_replace_all(" ", "_") %>% str_replace_all("'", "

# Merge visits data with actor classifications
merged_data <- speaker_counts %>%
  left_join(actors_data %>% select(speaker_std, Type_Merged), by = "speaker_std")

# Count total visits by actor classification
visit_summary <- merged_data %>%
  group_by(Type_Merged) %>%
  summarize(total_visits = sum(visits, na.rm = TRUE), unique_speakers = n(), .groups = "drop")

# Display results
print(visit_summary)
```

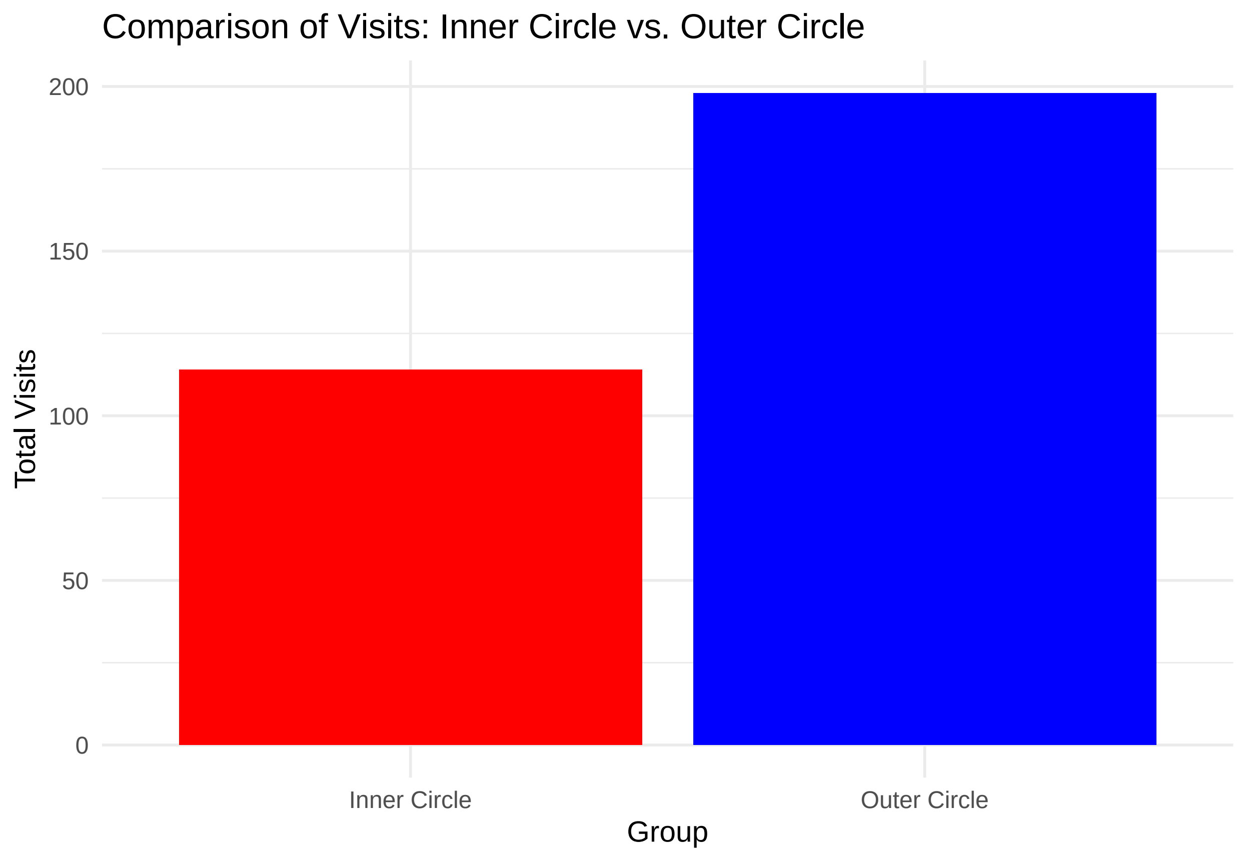
```
## # A tibble: 14 x 3
##    Type_Merged                     total_visits unique_speakers
##    <chr>                                  <int>           <int>
##  1 Ally                                       3               1
##  2 Businessperson                             9               3
##  3 Congress                                   9               4
##  4 Foreign                                    9               4
##  5 Front man                                  1               1
##  6 Government                            15             4
##  7 Government, Judiciary                  1             1
##  8 Inner Circle                         114            29
##  9 Media                                 23             5
## 10 Military or Police                    35            10
## 11 Military or Police, Government         3             1
## 12 Security                              88             1
## 13 Security rival                         2             1
## 14 <NA>                                 234            54
```

```
# Optionally, save to CSV
write_csv(visit_summary, "../data/visit_summary_by_type.csv")

# Load the summarized visit data
visit_summary <- read_csv("../data/visit_summary_by_type.csv", col_types = cols())
visit_summary <- visit_summary %>% filter(!is.na(Type_Merged))

# Create a bar plot to visualize the total visits for each type
ggplot(visit_summary, aes(x = reorder(Type_Merged, -total_visits), y = total_visits, fill = Type_Merged)
  geom_bar(stat = "identity", show.legend = FALSE) +  # Bar plot with total_visits as height
  labs(title = "Total Visits by Type",
       x = "Type of Individual",
       y = "Total Visits") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```



```
# Create a new column categorizing Inner Circle vs. Outer Circle
visit_summary <- visit_summary %>%
```

```r
  filter(!is.na(Type_Merged)) %>%
  mutate(circle_group = ifelse(Type_Merged == "Inner Circle", "Inner Circle", "Outer Circle"))

# Summarize total visits for each category
circle_summary <- visit_summary %>%
  group_by(circle_group) %>%
  summarize(total_visits = sum(total_visits, na.rm = TRUE), .groups = "drop")

# Create a bar plot to compare Inner Circle vs. Outer Circle
ggplot(circle_summary, aes(x = circle_group, y = total_visits, fill = circle_group)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  labs(title = "Comparison of Visits: Inner Circle vs. Outer Circle",
       x = "Group",
       y = "Total Visits") +
  theme_minimal() +
  scale_fill_manual(values = c("Inner Circle" = "red", "Outer Circle" = "blue"))  # Custom colors
```



```r
# 1) Read & clean actors data
actors_in  <- "../data/Copy of Master - transcript notes - Actors.tsv"
actors_out <- "../data/transcript_notes_cleaned.tsv"

actors_data <- read_tsv(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
    )
  )
```

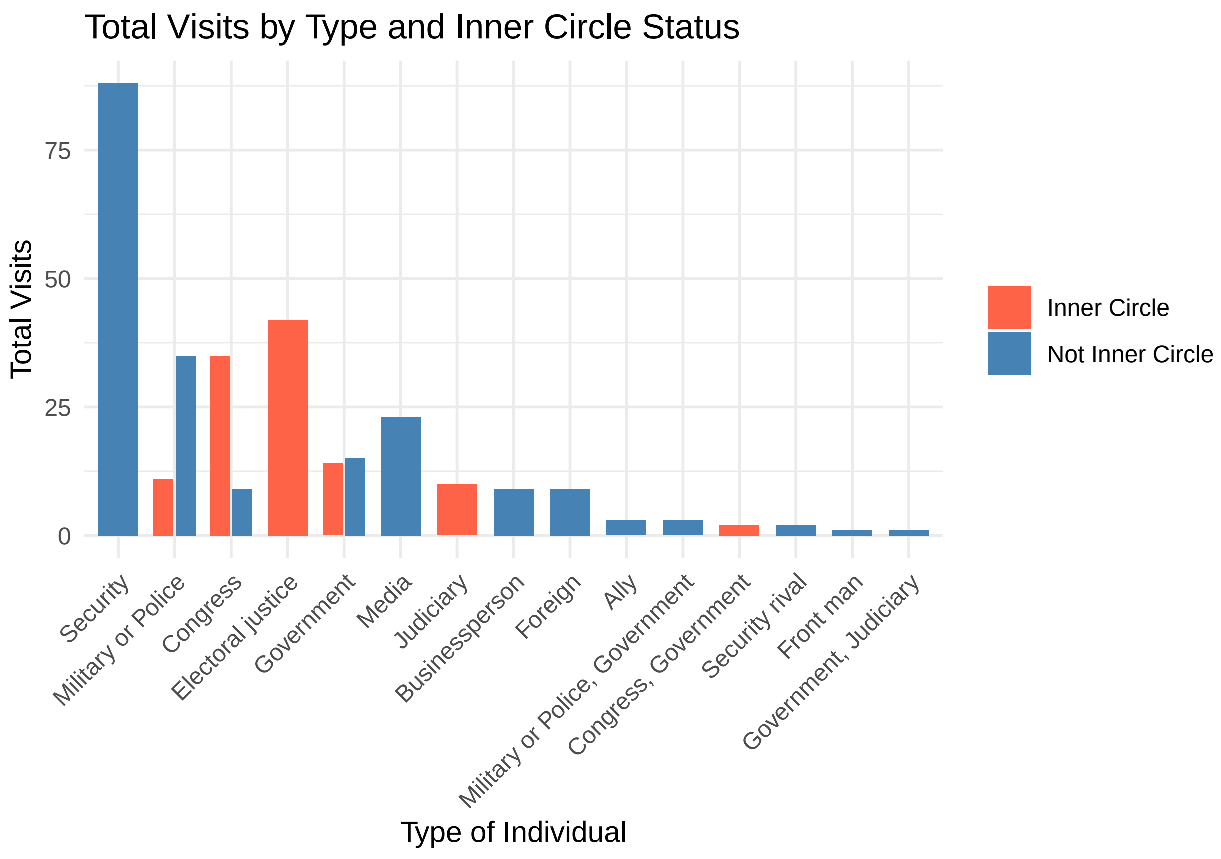
```
## New names:
## * `` -> `...1`
## * `` -> `...7`
## * `` -> `...9`
```

```
# 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
```

```
# 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
  }
})

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

# 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
```

```r
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))
```


Total Visits by Type and Inner Circle Status
Total Visits
0
25
50
75
Security
Military or Police
Congress
Electoral justice
Government
Media
Judiciary
Businessperson
Foreign
Ally
Military or Police, Government
Congress, Government
Security rival
Front man
Government, Judiciary
Type of Individual
Inner Circle
Not Inner Circle