Tidy Tuesday

Week 38

AUTHOR
Cristian T

PUBLISHED
September 29, 2024

This week we are exploring dialogue in Shakespeare play!

The dataset this week comes from shakespeare.mit.edu (via github.com/nrennie/shakespeare) which is the Web's first edition of the Complete Works of William Shakespeare. The site has offered Shakespeare's plays and poetry to the internet community since 1993.

Dialogue from Hamlet, Macbeth, and Romeo and Juliet are provided for this week. Which play has the most stage directions compared to dialogue? Which play has the longest lines of dialogue? Which character speaks the most?

`hamlet.csv`

variable	class	description
act	character	Act number.
scene	character	Scene number.
character	character	Name of character speaking or whether it's a stage direction.
dialogue	character	Text of dialogue or stage direction.
line_number	double	Dialogue line number.

`macbeth.csv`

variable	class	description
act	character	Act number.
scene	character	Scene number.
character	character	Name of character speaking or whether it's a stage direction.
dialogue	character	Text of dialogue or stage direction.
line_number	double	Dialogue line number.

`romeo_juliet.csv`

variable	class	description
act	character	Act number.
scene	character	Scene number.

variable	class	description
character	character	Name of character speaking or whether it's a stage direction.
dialogue	character	Text of dialogue or stage direction.
line_number	double	Dialogue line number.

Load the data

```
# Load the tidytuesday package
suppressMessages(library(tidytuesdayR)) # For accessing TidyTuesday datasets
suppressMessages(library(skimr)) # For summary and descriptive statistics
suppressMessages(library(tidyverse)) # For data manipulation and visualization
suppressMessages(library(dplyr)) # For data manipulation and transformation
suppressMessages(library(ggplot2)) # For data visualization
suppressMessages(library(RColorBrewer)) # For color palettes in visualizations
suppressMessages(library(ggimage)) # For adding images to plots
suppressMessages(library(tidytext))
suppressMessages(library(sentimentr))
suppressMessages(library(ggpubr))


# Load the current week's dataset
tuesdata <- tidytuesdayR::tt_load('2024-09-17')
```

```
Downloading file 1 of 3: `hamlet.csv`
Downloading file 2 of 3: `macbeth.csv`
Downloading file 3 of 3: `romeo_juliet.csv`
```

```
# Extract datasets from the TidyTuesday dataset
hamlet <- tuesdata$hamlet
macbeth <- tuesdata$macbeth
romeo_juliet <- tuesdata$romeo_juliet

# Rename datasets
#ca <- college_admissions


# Explore the structure of the dataset
str(hamlet) # Display the structure of 'hamlet'
```

```
spc_tbl_ [4,217 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ act        : chr [1:4217] "Act I" "Act I" "Act I" "Act I" ...
 $ scene      : chr [1:4217] "Scene I" "Scene I" "Scene I" "Scene I" ...
 $ character  : chr [1:4217] "[stage direction]" "Bernardo" "Francisco" "Bernardo" ...
 $ dialogue   : chr [1:4217] "FRANCISCO at his post. Enter to him BERNARDO" "Who's there?"
"Nay, answer me: stand, and unfold yourself." "Long live the king!" ...
 $ line_number: num [1:4217] NA 1 2 3 4 5 6 7 8 9 ...
 - attr(*, "spec")=
```

```
  .. cols(
  ..   act = col_character(),
  ..   scene = col_character(),
  ..   character = col_character(),
  ..   dialogue = col_character(),
  ..   line_number = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
str(macbeth) # Display the structure of 'macbeth'
```

```
spc_tbl_ [2,553 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ act        : chr [1:2553] "Act I" "Act I" "Act I" "Act I" ...
 $ scene      : chr [1:2553] "Scene I" "Scene I" "Scene I" "Scene I" ...
 $ character  : chr [1:2553] "[stage direction]" "First Witch" "First Witch" "Second Witch"
...
 $ dialogue   : chr [1:2553] "Thunder and lightning. Enter three Witches" "When shall we
three meet again" "In thunder, lightning, or in rain?" "When the hurlyburly's done," ...
 $ line_number: num [1:2553] NA 1 2 3 4 5 6 7 8 9 ...
 - attr(*, "spec")=
  .. cols(
  ..   act = col_character(),
  ..   scene = col_character(),
  ..   character = col_character(),
  ..   dialogue = col_character(),
  ..   line_number = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
str(romeo_juliet) # Display the structure of 'romeo_juliet'
```

```
spc_tbl_ [3,282 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ act        : chr [1:3282] "Act I" "Act I" "Act I" "Act I" ...
 $ scene      : chr [1:3282] "Prologue" "Prologue" "Prologue" "Prologue" ...
 $ character  : chr [1:3282] "Chorus" "Chorus" "Chorus" "Chorus" ...
 $ dialogue   : chr [1:3282] "Two households, both alike in dignity," "In fair Verona, where
we lay our scene," "From ancient grudge break to new mutiny," "Where civil blood makes civil
hands unclean." ...
 $ line_number: num [1:3282] 1 2 3 4 5 6 7 8 9 10 ...
 - attr(*, "spec")=
  .. cols(
  ..   act = col_character(),
  ..   scene = col_character(),
  ..   character = col_character(),
  ..   dialogue = col_character(),
  ..   line_number = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
skim(hamlet) # Provide detailed summary statistics for 'hamlet' (missing values, summ
```

Name	hamlet

Number of rows	4217
Number of columns	5
_____	
Column type frequency:	
character	4
numeric	1
_____	
Group variables	None

Data summary

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
act	0	1	5	7	0	5	0
scene	0	1	7	9	0	7	0
character	0	1	3	17	0	36	0
dialogue	0	1	3	671	0	4118	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
line_number	206	0.95	2006	1158.02	1	1003.5	2006	3008.5	4011	▇▇▇▇▇

```
skim(macbeth) # Provide detailed summary statistics for 'hamlet' (missing values, sum
```

Name	macbeth
Number of rows	2553
Number of columns	5
_____	
Column type frequency:	
character	4
numeric	1
_____	
Group variables	None

Data summary

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
act	0	1	5	7	0	5	0
scene	0	1	7	10	0	8	0
character	0	1	3	17	0	42	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
dialogue	0	1	3	132	0	2484	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
line_number	169	0.93	1192.5	688.35	1	596.75	1192.5	1788.25	2384	▇▇▇▇▇

```
skim(romeo_juliet) # Provide detailed summary statistics for 'hamlet' (missing values
```

Name	romeo_juliet
Number of rows	3282
Number of columns	5
_____	
Column type frequency:	
character	4
numeric	1
_____	
Group variables	None

Data summary

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
act	0	1	5	7	0	5	0
scene	0	1	7	9	0	7	0
character	0	1	4	17	0	35	0
dialogue	0	1	3	90	0	3205	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
line_number	189	0.94	1547	893.02	1	774	1547	2320	3093	▇▇▇▇▇

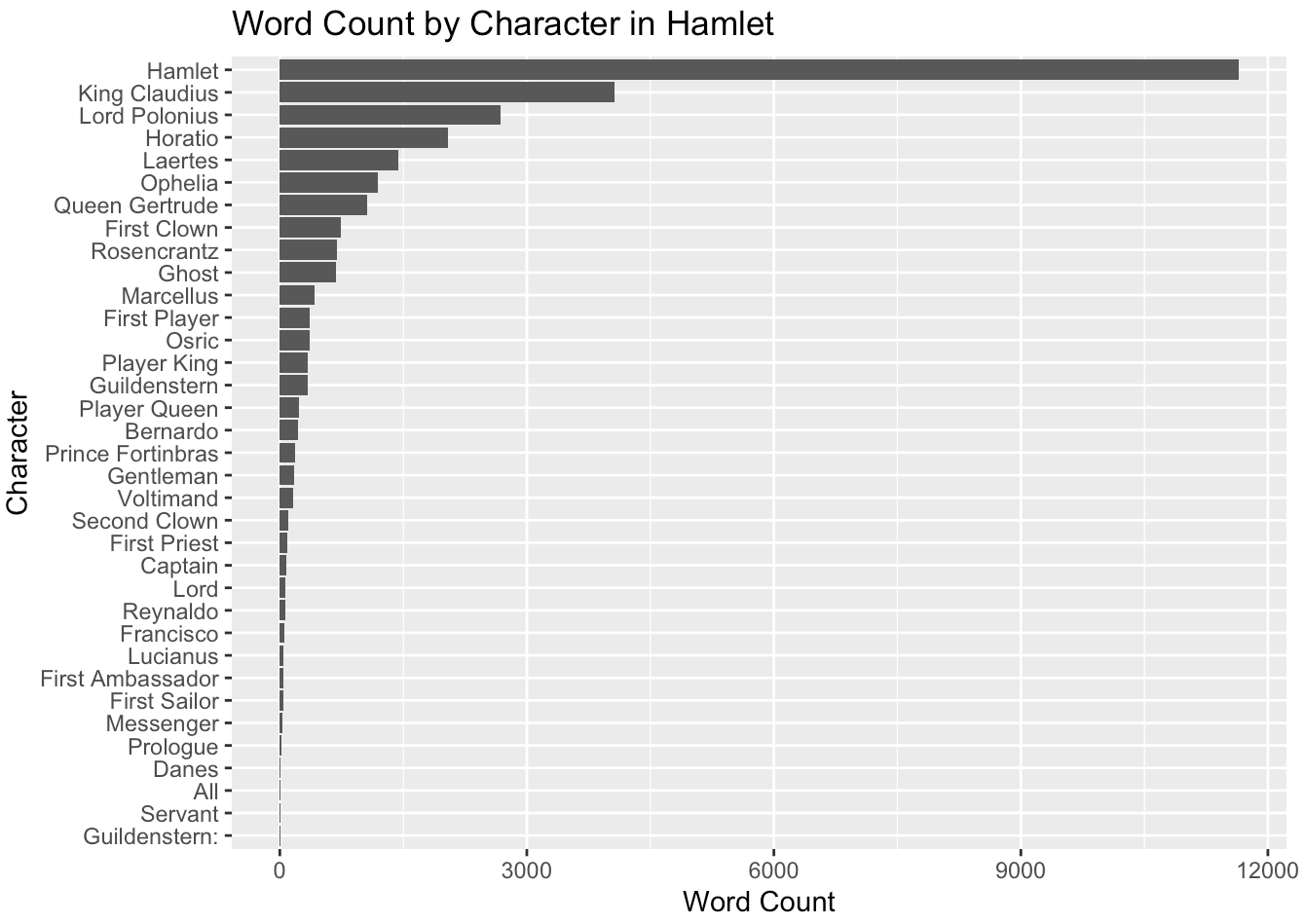
```
# Export data
# write.csv(hamlet, "hamlet.csv", row.names = FALSE)
# write.csv(macbeth, "macbeth.csv", row.names = FALSE)
# write.csv(romeo_juliet, "romeo_juliet.csv", row.names = FALSE)

# Combine datasets
combined_plays<- bind_rows(
  mutate(hamlet, play = "Hamlet"),
  mutate(macbeth, play = "Macbeth"),
  mutate(romeo_juliet, play = "Romeo and Juliet"))
```

```
#write.csv(combined_plays, "combined_plays.csv", row.names = FALSE)

#tidytuesdayR::use_tidytemplate()
```

```
#### Clean the data

# Missing values are associated with line numbers and stage direction
```

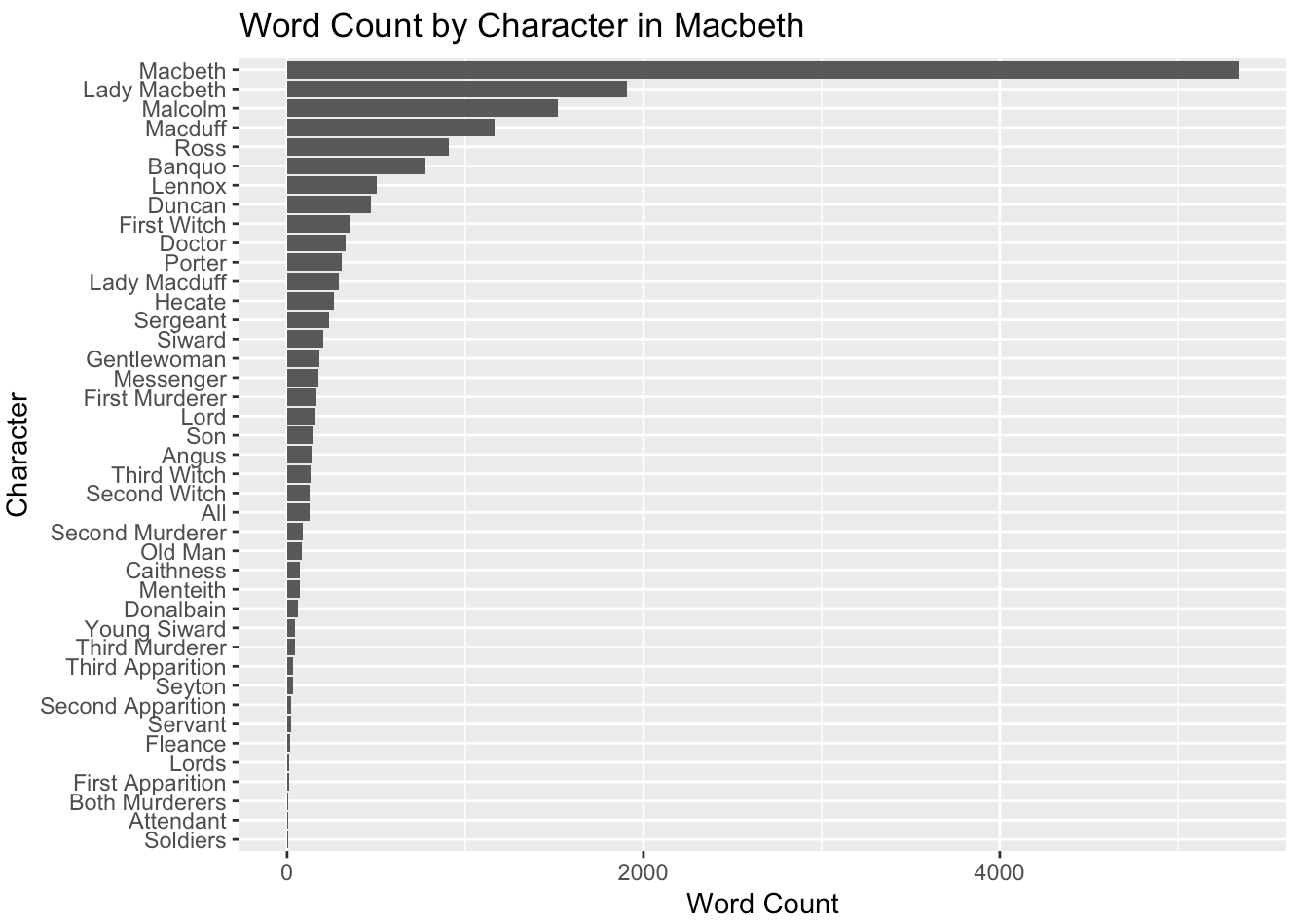
EDA

```
############ Wordcount by Character ############
hamlet_word_count <- hamlet %>%
  filter(!str_detect(character, "\\[stage direction\\]")) %>%
  unnest_tokens(word, dialogue) %>%
  count(character, sort = TRUE)

ggplot(hamlet_word_count, aes(x = reorder(character, n), y = n)) +
  geom_col() +
  coord_flip() +
  labs(title = "Word Count by Character in Hamlet", x = "Character", y = "Word Count"
```

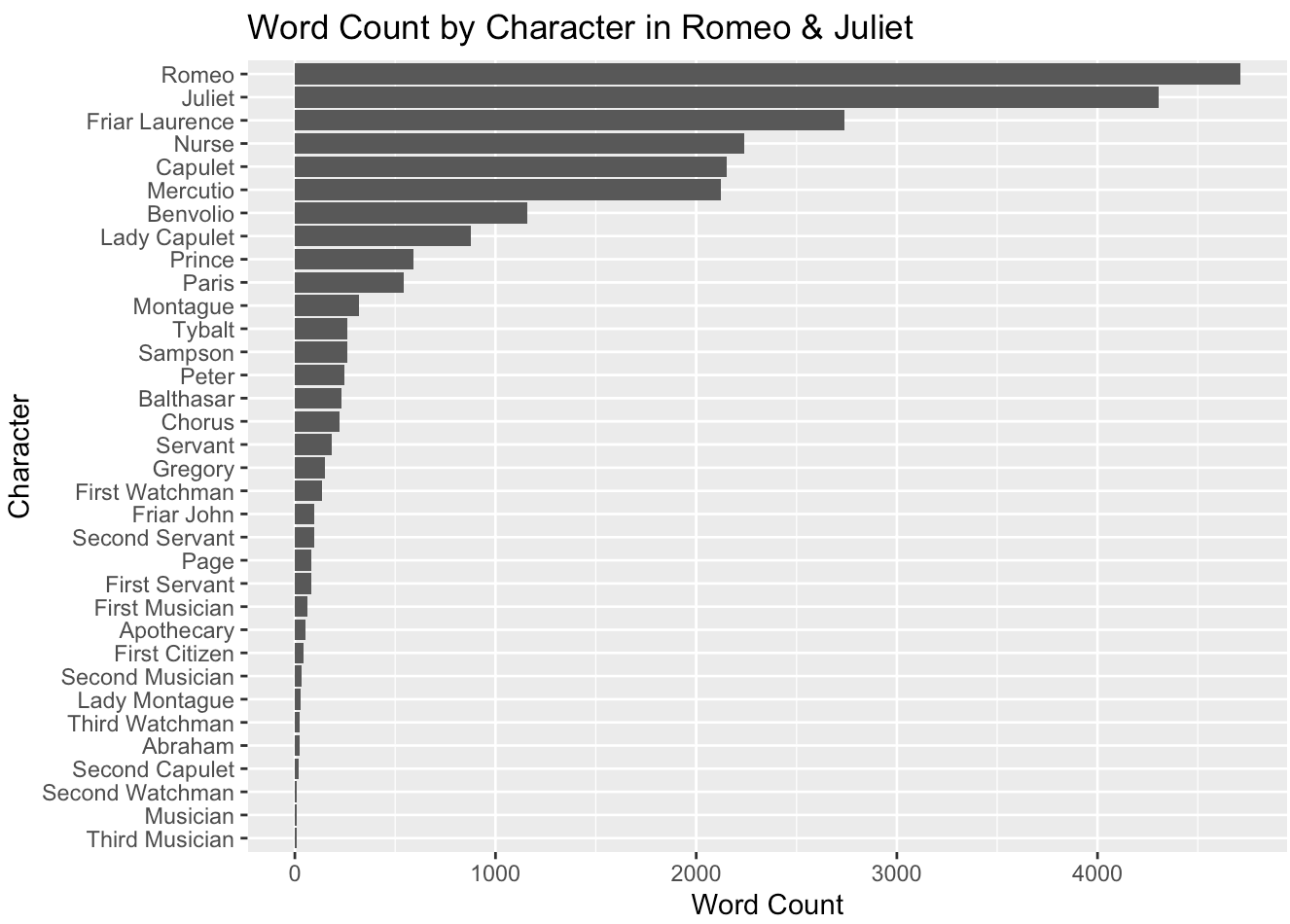



```
macbeth_word_count <- macbeth %>%
  filter(!str_detect(character, "\\[stage direction\\]")) %>%
  unnest_tokens(word, dialogue) %>%
  count(character, sort = TRUE)
```

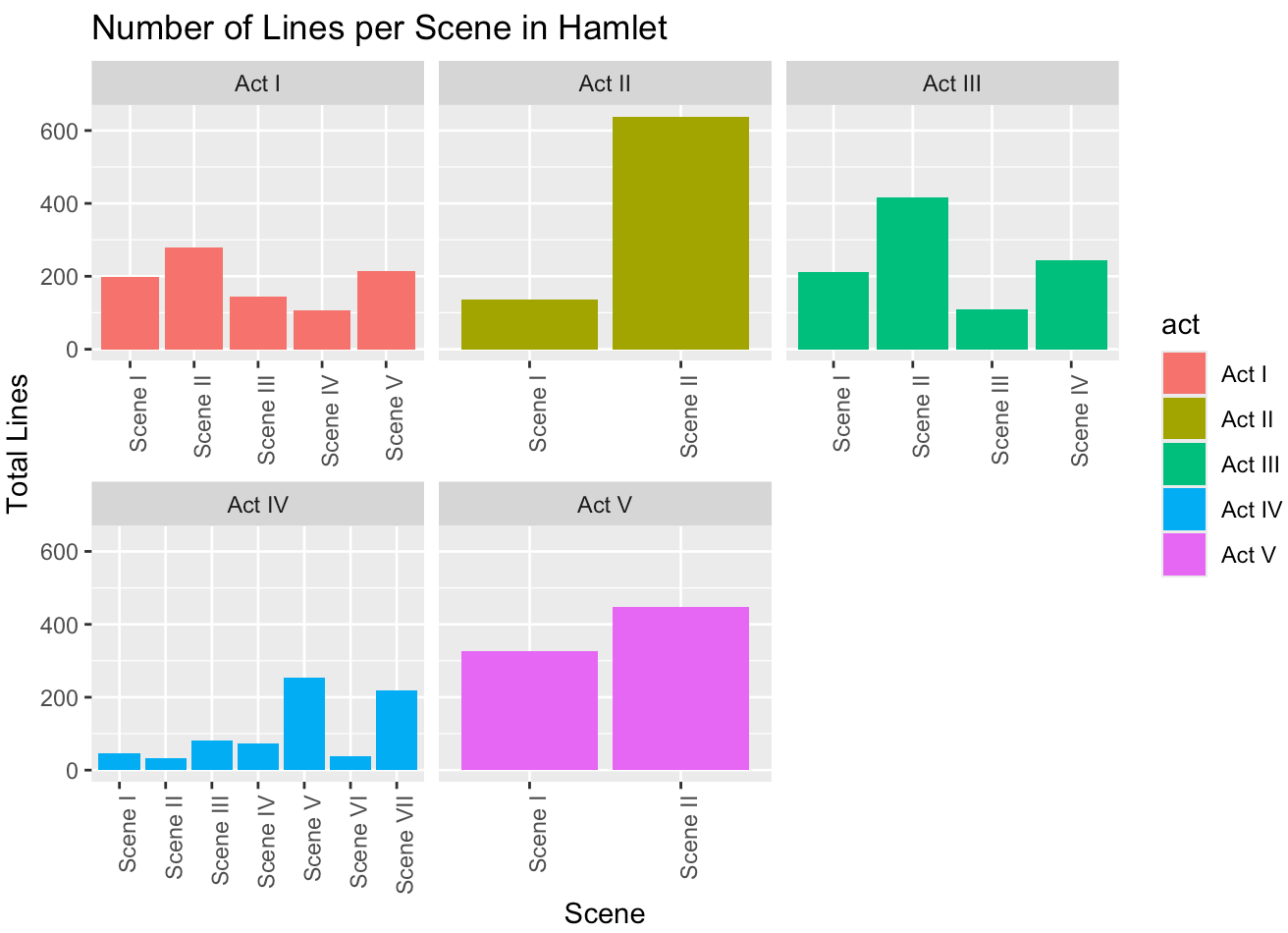
```
ggplot(macbeth_word_count, aes(x = reorder(character, n), y = n)) +
  geom_col() +
  coord_flip() +
  labs(title = "Word Count by Character in Macbeth", x = "Character", y = "Word Count
```



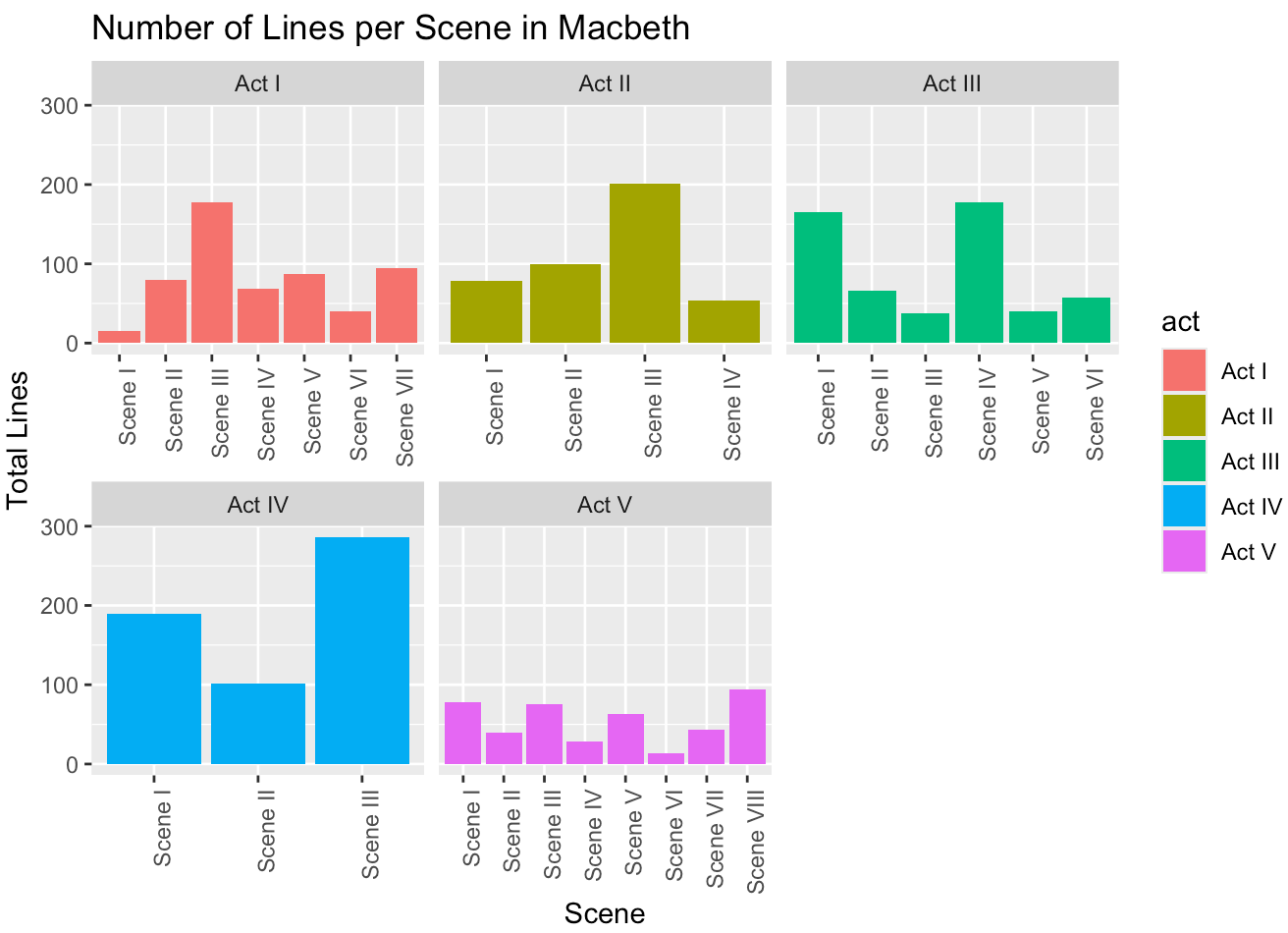

```
romeo_juliet_word_count <- romeo_juliet %>%
  filter(!str_detect(character, "\\[stage direction\\]")) %>%
  unnest_tokens(word, dialogue) %>%
  count(character, sort = TRUE)

ggplot(romeo_juliet_word_count, aes(x = reorder(character, n), y = n)) +
  geom_col() +
  coord_flip() +
  labs(title = "Word Count by Character in Romeo & Juliet", x = "Character", y = "Wor
```



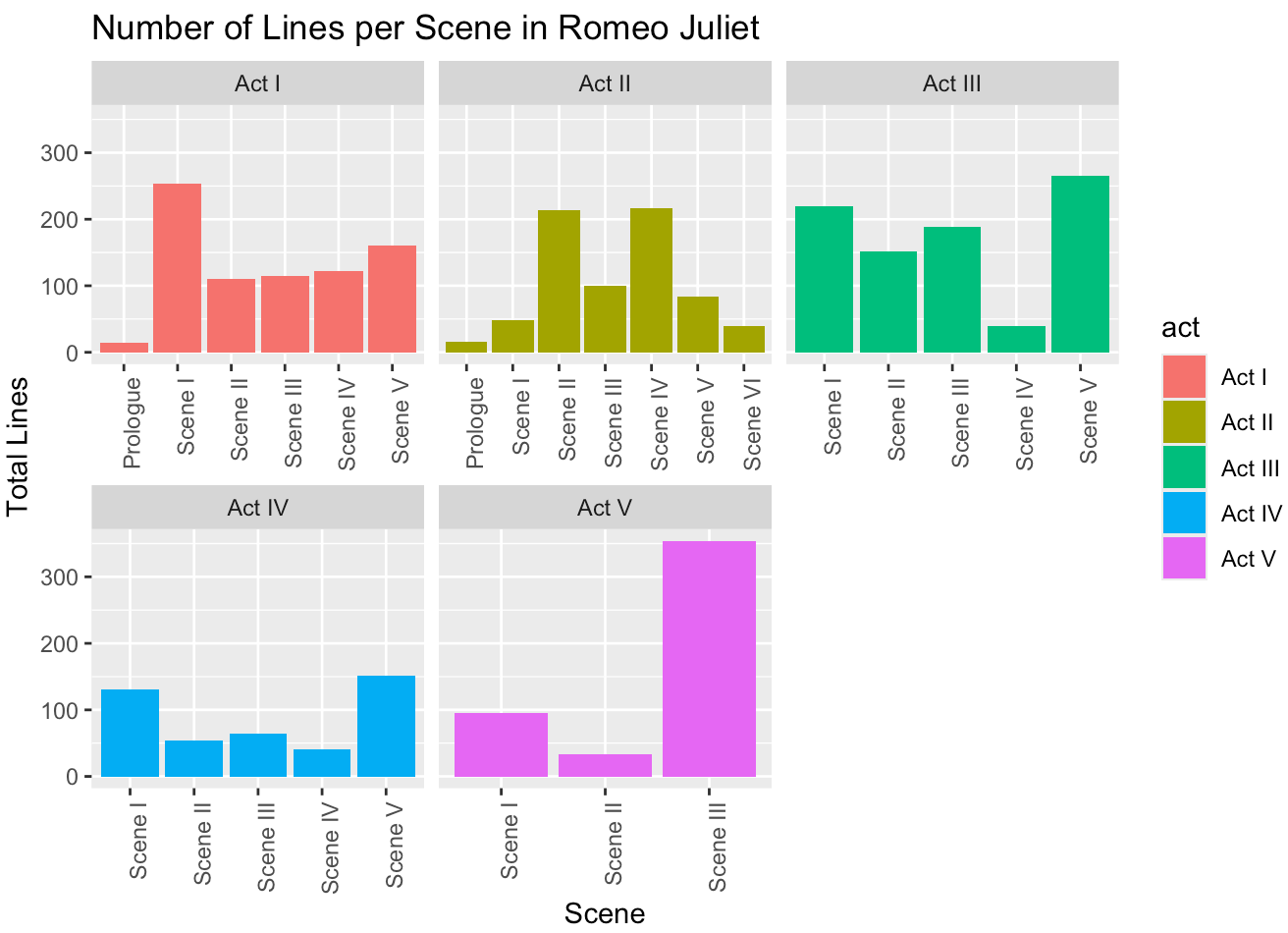

```
############ Dialogue by Scene ############
hamlet_lines_per_scene <- hamlet %>%
  group_by(act, scene) %>%
  summarise(total_lines = n())

ggplot(hamlet_lines_per_scene, aes(x = scene, y = total_lines, fill = act)) +
  geom_col() +
  facet_wrap(~act, scales = "free_x") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Number of Lines per Scene in Hamlet", x = "Scene", y = "Total Lines")
```



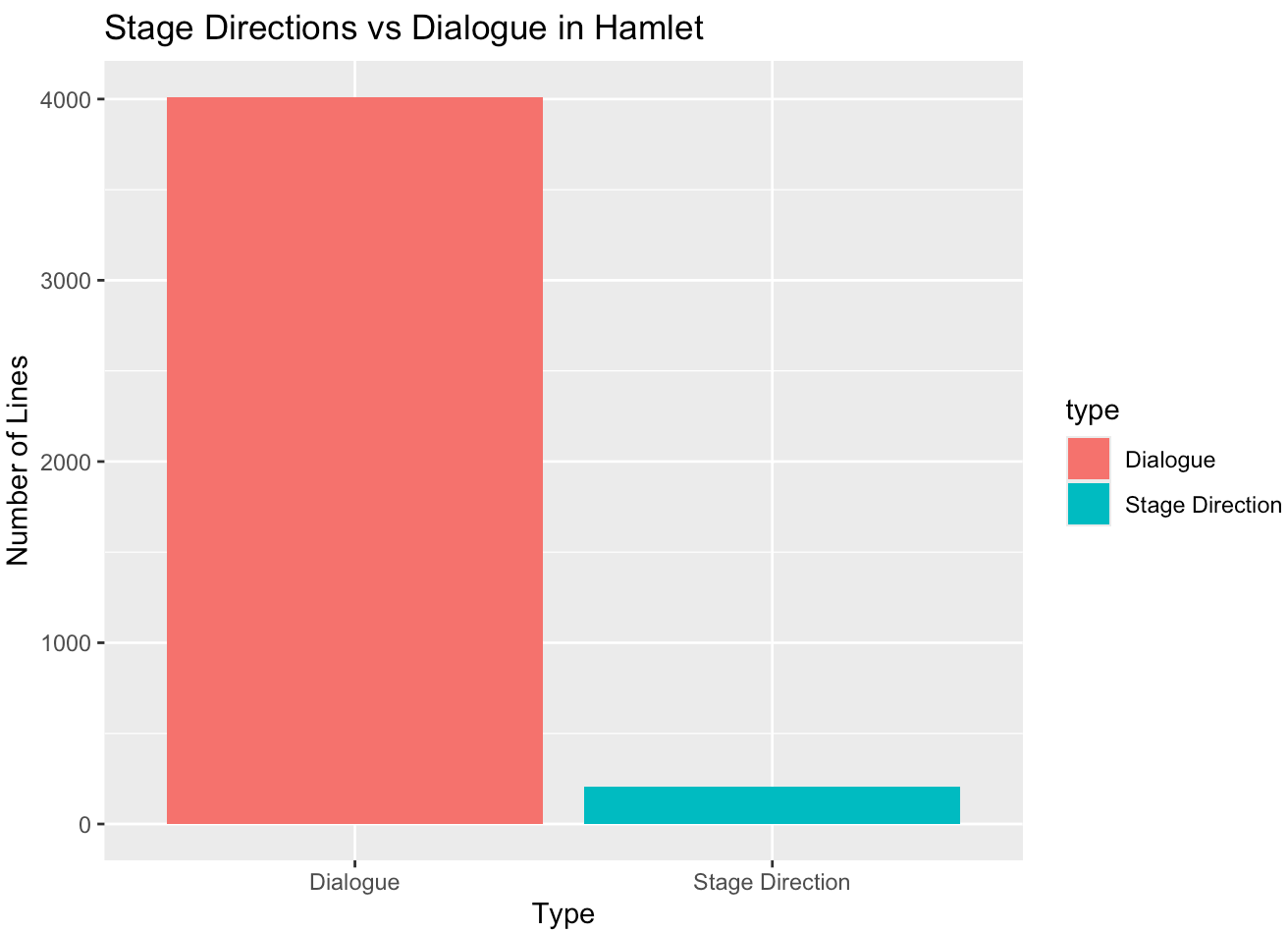

```r
macbeth_lines_per_scene <- macbeth %>%
  group_by(act, scene) %>%
  summarise(total_lines = n())

ggplot(macbeth_lines_per_scene, aes(x = scene, y = total_lines, fill = act)) +
  geom_col() +
  facet_wrap(~act, scales = "free_x") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Number of Lines per Scene in Macbeth", x = "Scene", y = "Total Lines"
```



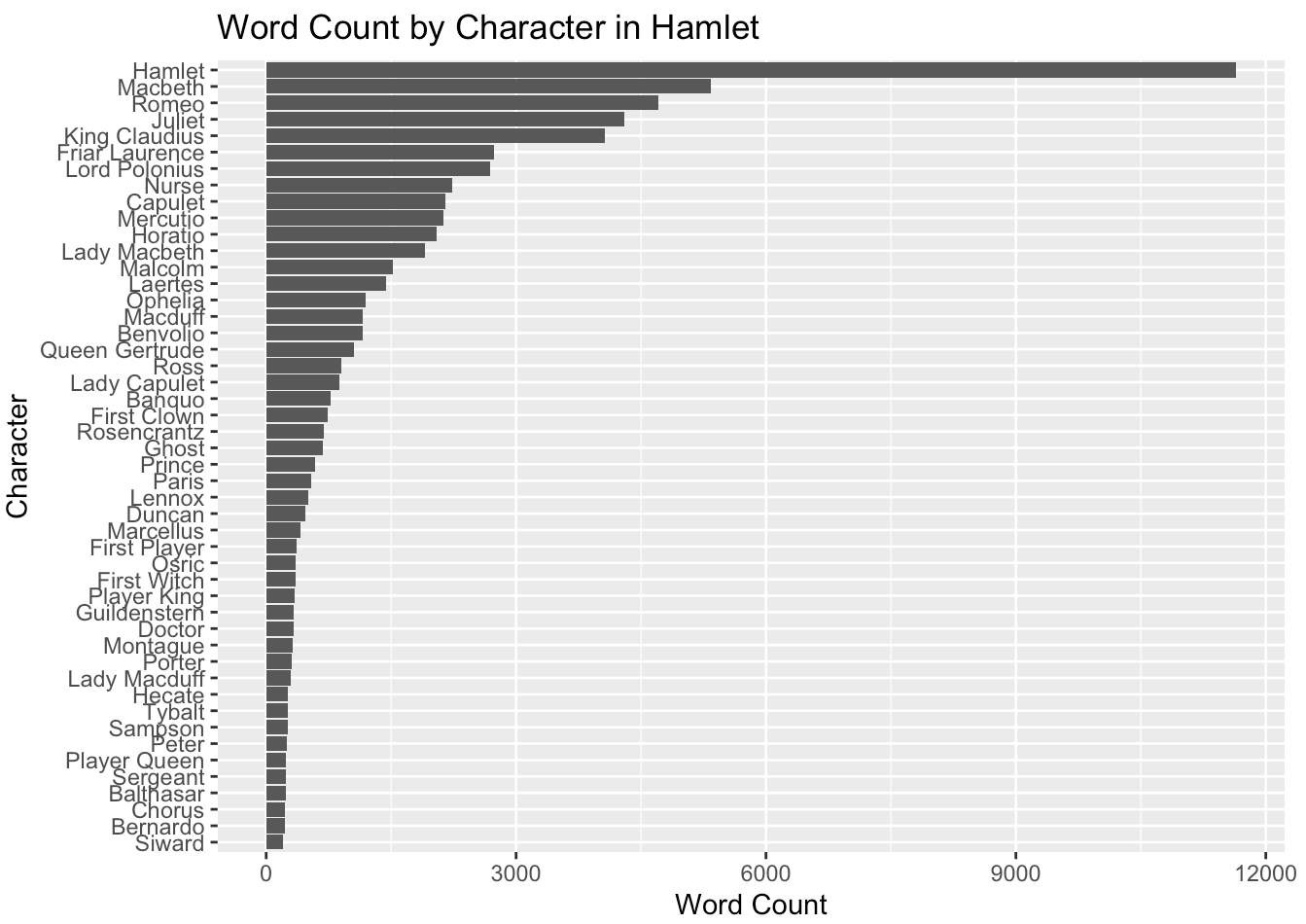

```
romeo_juliet_lines_per_scene <- romeo_juliet %>%
  group_by(act, scene) %>%
  summarise(total_lines = n())

ggplot(romeo_juliet_lines_per_scene, aes(x = scene, y = total_lines, fill = act)) +
  geom_col() +
  facet_wrap(~act, scales = "free_x") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Number of Lines per Scene in Romeo Juliet", x = "Scene", y = "Total L
```



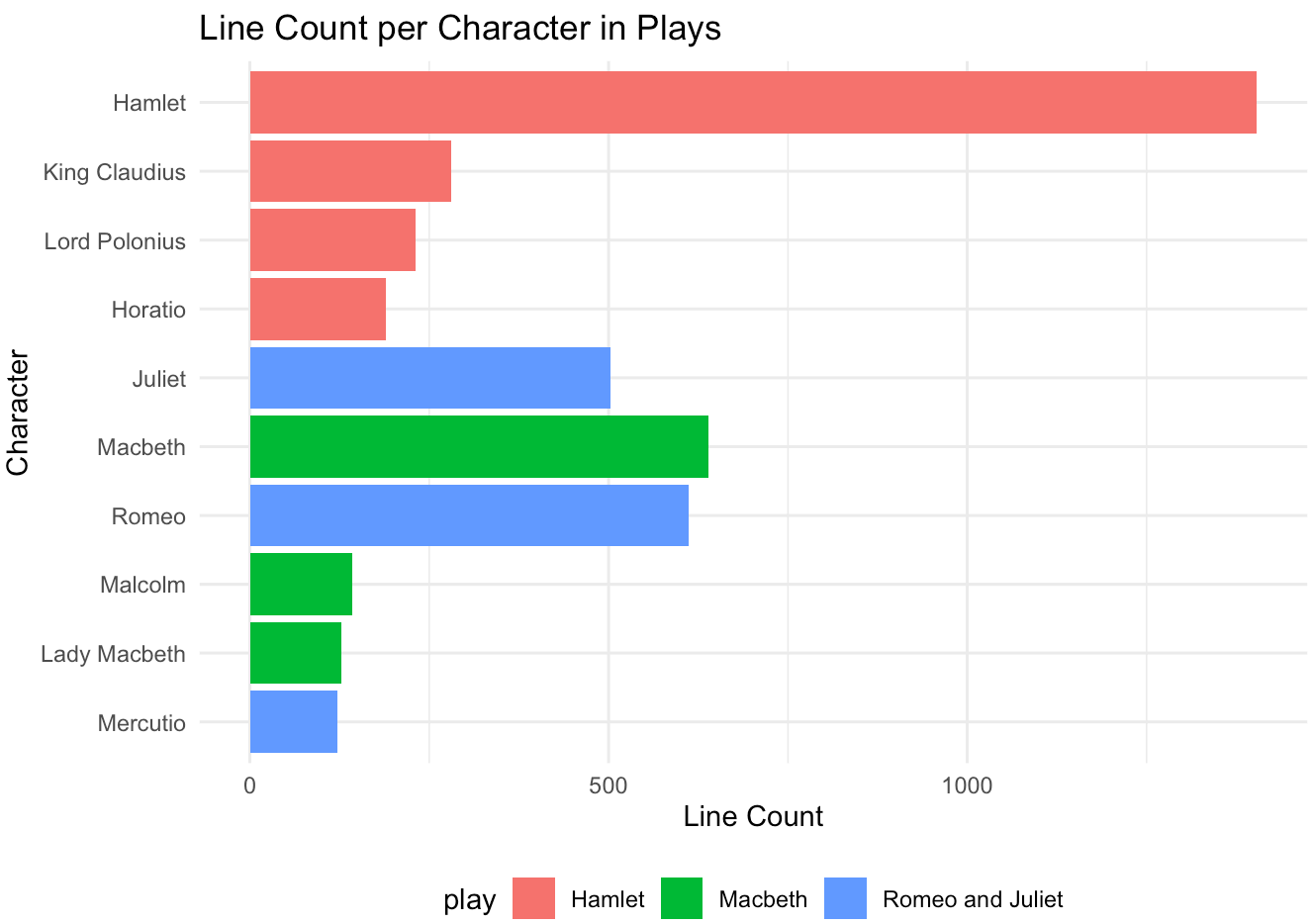

```
############## Stage Direction vs Dialogue ##############
hamlet_stage_vs_dialogue <- hamlet %>%
  mutate(type = ifelse(str_detect(character, "\\[stage direction\\]"), "Stage Directi
  group_by(type) %>%
  summarise(total_lines = n())

ggplot(hamlet_stage_vs_dialogue, aes(x = type, y = total_lines, fill = type)) +
  geom_col() +
  labs(title = "Stage Directions vs Dialogue in Hamlet", x = "Type", y = "Number of l
```




```
############## Combined counts ##############
play_word_count <- combined_plays %>%
  filter(!str_detect(character, "\\[stage direction\\]")) %>%
  unnest_tokens(word, dialogue) %>%
  count(play, character, sort = TRUE)

play_word_count_filter <- play_word_count %>% filter(n > 200)

ggplot(play_word_count_filter, aes(x = reorder(character, n), y = n)) +
  geom_col() +
  coord_flip() +
  labs(title = "Word Count by Character in Hamlet", x = "Character", y = "Word Count"
```

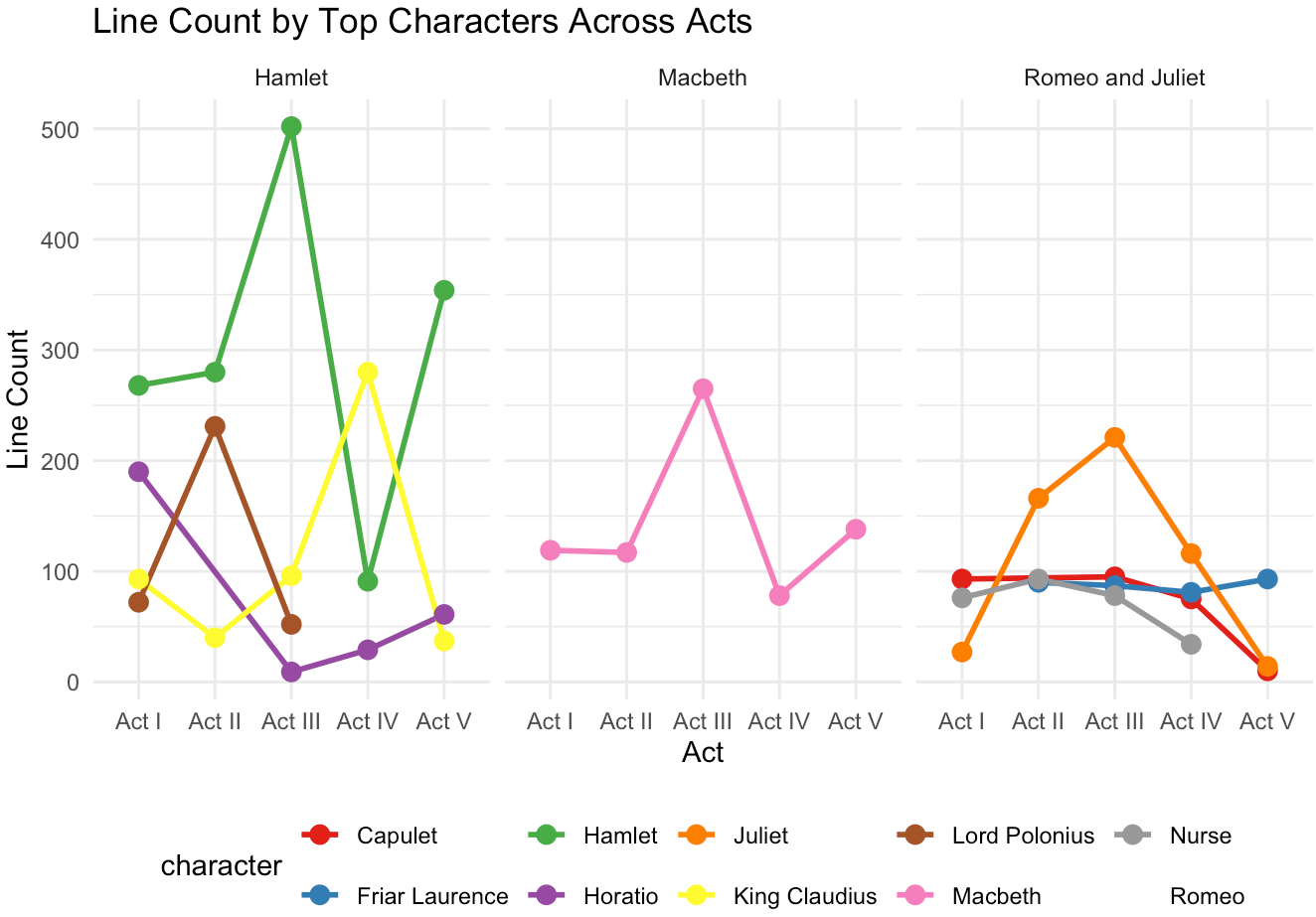



```r
play_line_count <- combined_plays %>%
  filter(!str_detect(character, "\\[stage direction\\]")) %>%
  group_by(play, character, act) %>%
  summarise(line_count = n()) %>%
  filter(line_count > 100)

ggplot(play_line_count, aes(x = reorder(character, line_count), y = line_count, fill
  geom_col() +
  coord_flip() +  # Flips the axes for better readability
  labs(title = "Line Count per Character in Plays",
       x = "Character",
       y = "Line Count") +
  theme_minimal() +
  theme(legend.position = "bottom")  # Adjust legend position if needed
```




```
###



play_line_count2 <- combined_plays %>%
  filter(!str_detect(character, "\\[stage direction\\]")) %>%
  group_by(play, character, act) %>%
  summarise(line_count = n()) %>%
  filter(line_count > 1)

# Define the number of top characters you want to display
top_n_characters <- 10

# Create a summary of line counts by character across all acts
top_characters <- play_line_count2 %>%
  group_by(character) %>%
  summarise(total_lines = sum(line_count)) %>%
  arrange(desc(total_lines)) %>%
  slice_head(n = top_n_characters) %>%
  pull(character)

# Filter the original data for these top characters
filtered_play_line_count <- play_line_count2 %>%
  filter(character %in% top_characters)

# Create line plots for each play with the filtered characters
ggplot(filtered_play_line_count, aes(x = act, y = line_count, color = character, grou
```

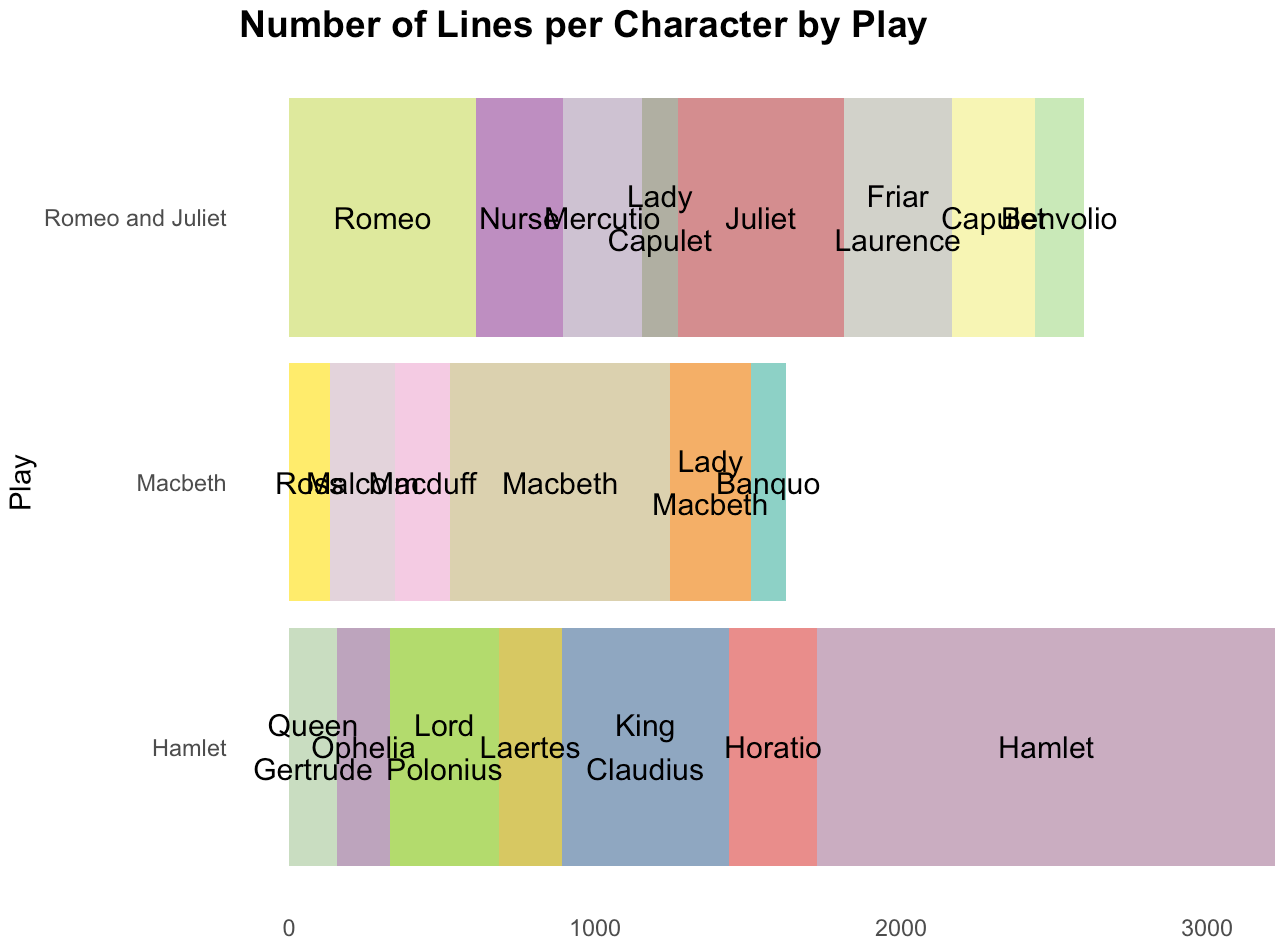
```
  geom_line(size = 1) +
  geom_point(size = 3) +
  labs(title = 'Line Count by Top Characters Across Acts',
       x = 'Act',
       y = 'Line Count') +
  facet_wrap(~ play) +
  scale_color_brewer(palette = "Set1") +  # Choose a color palette
  theme_minimal() +
  theme(legend.position = "bottom")
```



Plot 1 Number of Lines per Character by Play

```
play_line_count_char <- combined_plays %>%
  filter(!str_detect(character, "\\[stage direction\\]")) %>%
  group_by(play, character) %>%
  summarise(line_count = n(), .groups = "drop") %>%
  filter(line_count > 100)%>%
  arrange(desc(line_count))

# Random colors function

# Define a function to generate colors based on a data frame and column
generate_colors <- function(data, column) {
  num_items <- length(unique(data[[column]]))

  if (num_items <= 8) {
    return(brewer.pal(num_items, "Set3"))
```

```
  } else {
    return(colorRampPalette(brewer.pal(12, "Set3"))(num_items))
  }
}


random_colors <- generate_colors(play_line_count_char, "character")

ggplot(play_line_count_char, aes(x = play, y = line_count, fill = character)) +
  geom_bar(stat = "identity", position = "stack") +
    geom_text(aes(label = str_wrap(character, width = 10)), position = position_stack
  scale_fill_manual(values = random_colors) +
  labs(title = "Number of Lines per Character by Play",
       x = "Play",
       y = "Number of Lines",
       fill = "Character") +
  theme_minimal() +
  theme(axis.title.x = element_blank(),
        plot.title = element_text(size = 14, face = "bold"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = "none")+
  coord_flip()
```



In Hamlet, the character Hamlet dominates the play with a total of 1,495 lines. Following him is King Claudius with 546 lines, and Lord Polonius with 355 lines. Notably, Ophelia has 173 lines. In Macbeth, Macbeth himself has 717 lines, establishing him as a central figure. Lady Macbeth follows closely with 265 lines, showcasing her significant presence in the dialogue. Other key characters include Malcolm with 212 lines

and Macduff with 180 lines. In Romeo and Juliet, Romeo leads with 612 lines, while Juliet is also prominent with 544 lines. Additional important characters include Friar Laurence with 351 lines, Nurse with 281 lines, and Mercutio with 261 lines.

Plot 2 Average Sentiment by Character Type

```
############# Sentiment Analysis #############

# Convert text to lowercase
combined_plays$dialogue <- tolower(combined_plays$dialogue)

suppressMessages(library(sentimentr))
# https://github.com/trinker/sentimentr

# Filter out stage direction lines from the combined plays
filtered_plays <- combined_plays %>%
  filter(!str_detect(character, "\\[stage direction\\]"))

# Use the sentimentr package to compute sentiment scores
sentiment_results <- sentiment(filtered_plays$dialogue)

# Add `element_id` from `sentiment_results` to `filtered_plays` as a unique identifie
filtered_plays <- filtered_plays %>%
  mutate(element_id = row_number())

# Merge on `element_id` to combine the sentiment results with the filtered plays data
filtered_plays <- left_join(filtered_plays, sentiment_results, by = "element_id")

# Rename the merged sentiment column to a meaningful name and drop unnecessary columr
filtered_plays <- filtered_plays %>%
  select(-element_id, -sentence_id, -word_count) %>%
  rename(sentiment = sentiment)

# View the final structure
#str(filtered_plays)

# Preview the first few rows of the cleaned data with sentiment scores
#head(filtered_plays)

# Calculate average sentiment by character
avg_sentiment_by_character <- filtered_plays %>%
  group_by(character, play, act) %>%
  summarise(avg_sentiment = mean(sentiment, na.rm = TRUE)) %>%
  arrange(desc(avg_sentiment))

# View top characters by average sentiment
#head(avg_sentiment_by_character)



############# Character Categorization #############

# Define character categories for each play using these characteristics
# Protagonists: Main characters driving the plot.
```

```r
# Antagonists: Characters opposing the protagonists.
# Supporting Characters: Key secondary characters that assist the protagonists.
# Minor Characters: Less significant characters that contribute to the story.

# Hamlet
protagonists_hamlet <- c("Hamlet")
antagonists_hamlet <- c("King Claudius", "Lord Polonius")
supporting_characters_hamlet <- c("Ophelia", "Horatio", "Laertes", "Queen Gertrude")

#filtered_plays %>%
#  filter(character %in% c("Ophelia"))

# Macbeth
protagonists_macbeth <- c("Macbeth")
antagonists_macbeth <- c("Lady Macbeth")
supporting_characters_macbeth <- c("Banquo", "Duncan", "Macduff", "Malcolm", "Ross")

# Romeo and Juliet
protagonists_romeo_juliet <- c("Romeo", "Juliet")
antagonists_romeo_juliet <- c("Tybalt", "Paris")
supporting_characters_romeo_juliet <- c("Benvolio", "Mercutio", "Nurse", "Friar Laure

# Combine all character categories into a list for comparison
protagonists <- c(protagonists_hamlet, protagonists_macbeth, protagonists_romeo_julie
antagonists <- c(antagonists_hamlet, antagonists_macbeth, antagonists_romeo_juliet)
supporting_characters <- c(supporting_characters_hamlet, supporting_characters_macbet

# Categorize characters in filtered_plays
filtered_plays <- filtered_plays %>%
  mutate(character_type = case_when(
    character %in% protagonists ~ "Protagonist",
    character %in% antagonists ~ "Antagonist",
    character %in% supporting_characters ~ "Supporting Character",
    TRUE ~ "Minor Character"  # All others are minor characters
  ))

# Calculate average sentiment by character type and play
# avg_sentiment_by_character_type <- filtered_plays %>%
#   group_by(character_type, play) %>%
#   summarize(avg_sentiment = mean(sentiment, na.rm = TRUE), .groups = "drop")
#
# # Create a bar plot
# ggplot(avg_sentiment_by_character_type, aes(x = character_type, y = avg_sentiment,
#   geom_bar(stat = "identity", position = position_dodge()) +
#   theme_minimal() +
#   labs(title = "Average Sentiment by Character Type",
#        x = "Character Type",
#        y = "Average Sentiment") +
#   scale_fill_brewer(palette = "Set3") +
#   coord_cartesian(ylim = c(-0.15, 0.1)) +
#   theme(axis.title.x = element_blank(),  # Remove x-axis text
#         #axis.text.y = element_blank(), # Remove y-axis text
#         plot.title = element_text(size = 14, face = "bold"),
#         panel.grid.major = element_blank(),
```

```
#         panel.grid.minor = element_blank(),
#         legend.position = "bottom")

# Box plot with facets for a different view for all characters
# ggplot(filtered_plays, aes(x = character_type, y = sentiment, fill = character_type
#   geom_boxplot(outlier.shape = NA, position = position_dodge(width = 0.8)) +
#   geom_jitter(color = "black", alpha = 0.5, size = 0.5, position = position_jitter(
#   theme_minimal() +
#   labs(title = "Distribution of Sentiment by Character Type",
#        subtitle = "with distribution of all characters within each play",
#        x = "Character Type",
#        y = "Sentiment Score") +
#   scale_fill_brewer(palette = "Set3") +
#   coord_cartesian(ylim = c(-1.5, 1.5)) +
#   theme(axis.title.x = element_blank(),
#         plot.title = element_text(size = 14, face = "bold"),
#         panel.grid.major = element_blank(),
#         panel.grid.minor = element_blank(),
#         legend.position = "bottom") +
#   facet_wrap(~ play)

filtered_plays %>% arrange(desc(sentiment))
```

act	scene	character
<chr>	<chr>	<chr>
Act III	Scene V	Juliet
Act II	Scene IV	Romeo
Act II	Scene II	Hamlet
Act V	Scene I	Doctor
Act III	Scene I	Rosencrantz
Act III	Scene I	Hamlet
Act II	Scene II	Juliet
Act III	Scene II	Hamlet
Act IV	Scene V	King Claudius
Act IV	Scene III	Malcolm

1-10 of 10,000 rows | 1-3 of 8 columns

Previous 1 2 3 4 5 6 ... 1000 Next

```
filtered_plays %>% arrange(sentiment)
```

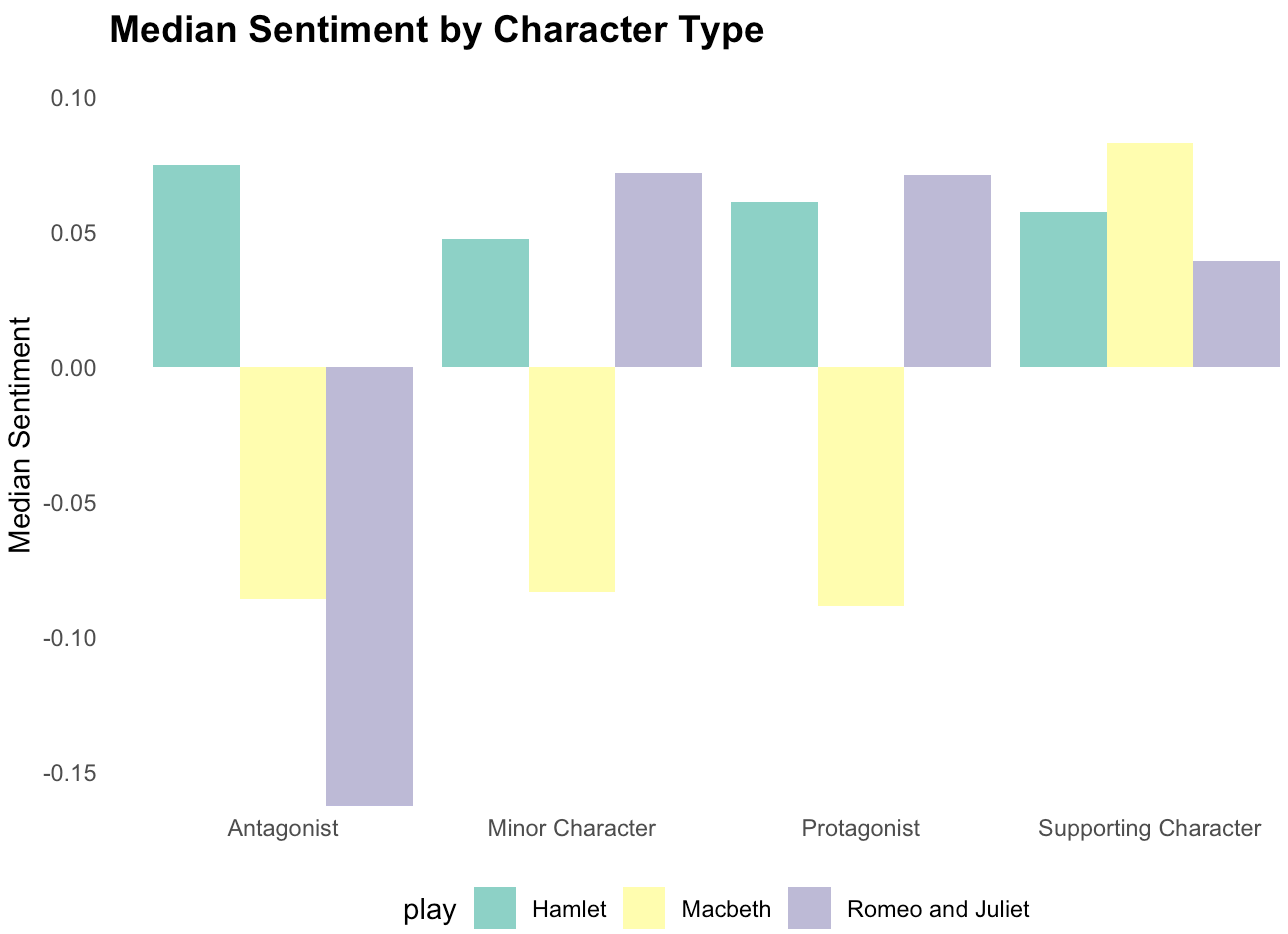
act	scene	character
<chr>	<chr>	<chr>
Act I	Scene V	Ghost
Act I	Scene I	Romeo
Act II	Scene II	Hamlet
Act II	Scene III	Macduff
Act II	Scene III	Porter
Act II	Scene III	Porter
Act III	Scene III	Romeo

act	scene	character
<chr>	<chr>	<chr>
Act I	Scene V	Ghost
Act I	Scene V	Hamlet
Act III	Scene IV	Hamlet

1-10 of 10,000 rows | 1-3 of 8 columns

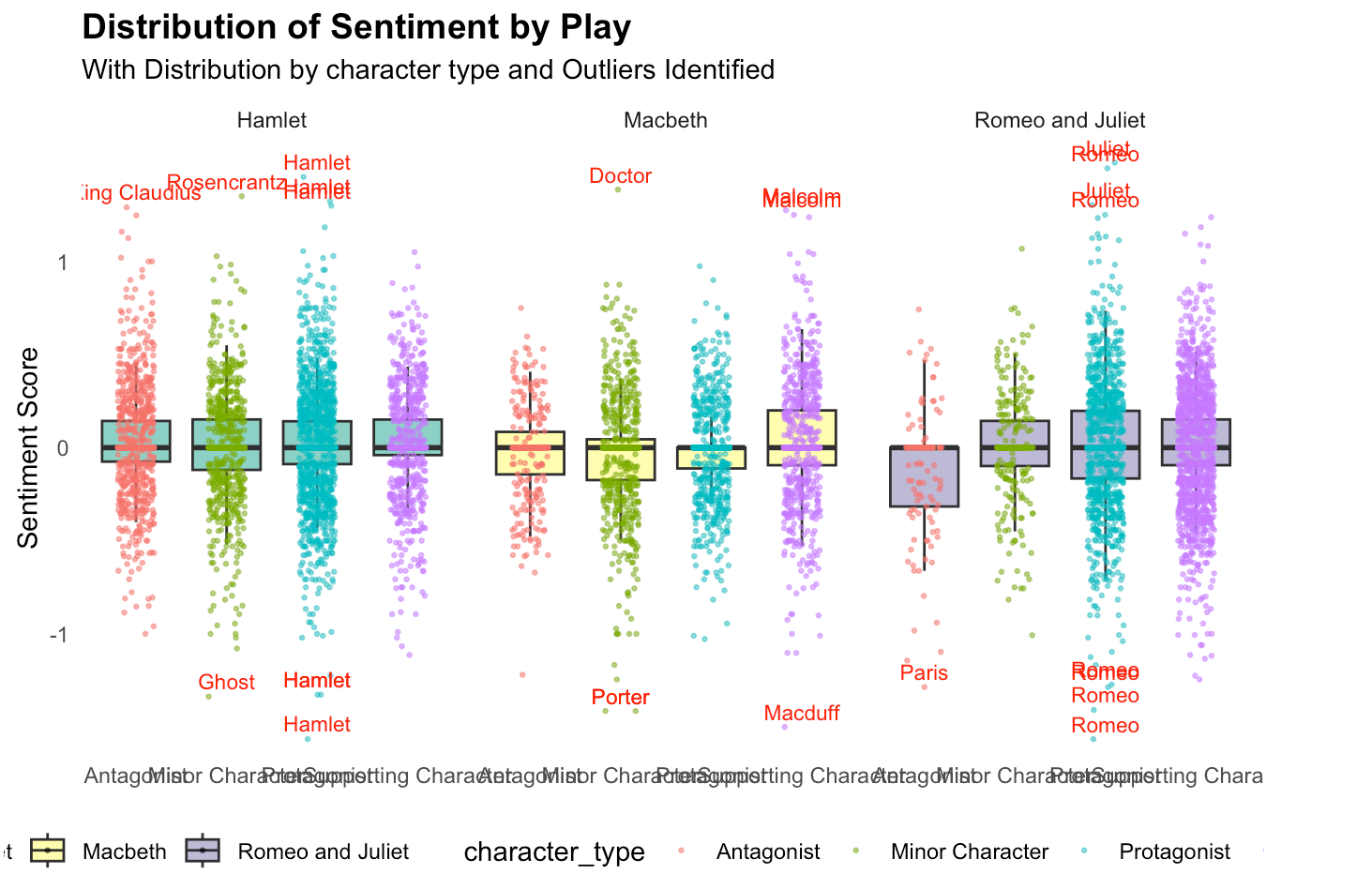
Previous 1 2 3 4 5 6 ... 1000 Next

```
############## Calculate for median
lower_threshold <- -0.00
upper_threshold <- 0.00
# Calculate average sentiment by character type and play
median_sentiment_filtered <- filtered_plays %>%
  filter(sentiment < lower_threshold | sentiment > upper_threshold) %>%  # Filter out
  group_by(character_type, play) %>%
  summarize(median_sentiment = median(sentiment, na.rm = TRUE))

ggplot(median_sentiment_filtered, aes(x = character_type, y = median_sentiment, fill
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_minimal() +
  labs(title = "Median Sentiment by Character Type",
       x = "Character Type",
       y = "Median Sentiment") +
  scale_fill_brewer(palette = "Set3") +
  coord_cartesian(ylim = c(-0.15, 0.1)) +
  theme(axis.title.x = element_blank(),  # Remove x-axis text
        #axis.text.y = element_blank(), # Remove y-axis text
        plot.title = element_text(size = 14, face = "bold"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = "bottom")
```




```
# Box plot with facets for a different view for all characters
# ggplot(filtered_plays, aes(x = character_type, y = sentiment, fill = play)) +
#   geom_boxplot(outlier.shape = NA, position = position_dodge(width = 0.8)) +
#     geom_jitter(aes(color = character_type), alpha = 0.5, size = 0.5, position = po
#   theme_minimal() +
#   labs(title = "Distribution of Sentiment by Play",
#        subtitle = "With Distribution of character type",
#        x = "Character Type",
#        y = "Sentiment Score") +
#   scale_fill_brewer(palette = "Set3") +
#   coord_cartesian(ylim = c(-1.5, 1.5)) +
#   theme(axis.title.x = element_blank(),
#         plot.title = element_text(size = 14, face = "bold"),
#         panel.grid.major = element_blank(),
#         panel.grid.minor = element_blank(),
#         legend.position = "bottom") +
#   facet_wrap(~ play)


############# alternate view
# Define a threshold for outliers
lower_threshold <- -1.25  # Lower bound for outliers
upper_threshold <- 1.25   # Upper bound for outliers

# Identify outliers in your data
filtered_plays <- filtered_plays %>%
  mutate(is_outlier = sentiment <= lower_threshold | sentiment >= upper_threshold)
```

```r
# Create the plot
ggplot(filtered_plays, aes(x = character_type, y = sentiment, fill = play)) +
  geom_boxplot(outlier.shape = NA, position = position_dodge(width = 0.8)) +
    geom_jitter(aes(color = character_type), alpha = 0.5, size = 0.5, position = posi
  theme_minimal() +
  labs(title = "Distribution of Sentiment by Play",
       subtitle = "With Distribution by character type and Outliers Identified",
       x = "Character Type",
       y = "Sentiment Score") +
  scale_fill_brewer(palette = "Set3") +
  coord_cartesian(ylim = c(-1.5, 1.5)) +
  theme(axis.title.x = element_blank(),
        plot.title = element_text(size = 14, face = "bold"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = "bottom") +
  facet_wrap(~ play) +
  geom_text(data = filtered_plays %>% filter(is_outlier),
            aes(label = character),  # Use character names as labels
            vjust = -0.5,  # Adjust vertical position of text
            color = "red",  # Color for the outlier text
            size = 3,
            )  # Adjust text size as needed
```



The sentiment scores show that the antagonists in Romeo and Juliet display the most negative sentiments, while protagonists across Hamlet and Macbeth are slightly positivity or neutral. The Supporting characters generally show more positive sentiment in Hamlet and Macbeth than the minor characters and antagonists.

In Hamlet, the protagonist shows slight positivity, indicating moments of introspection and depth amid his struggles with existential questions and moral dilemmas reflecting a complex psychological landscape. The antagonists, King Claudius and Lord Polonius display a negative sentiment, implying a morally ambiguous portrayal where their manipulative and deceitful actions create tension against Hamlet. The supporting characters, such as Ophelia, Horatio, Laertes, and Queen Gertrude, generally have favorable sentiments that contribute positively to the narrative, showcasing their roles as emotional anchors in Hamlet's turbulent journey.

Macbeth has a slightly negative average sentiment score, illustrating his tragic descent from a noble warrior to a tyrannical ruler consumed by ambition and guilt. This character arc highlights the play's central themes of ambition and moral decay. The primary antagonist, Lady Macbeth, reflects a similarly dark sentiment, indicating her crucial role in driving Macbeth's ambition and the ensuing chaos. The supporting characters, such as Banquo, Duncan, Macduff, Malcolm, and Ross, are portrayed with average sentiment scores that contribute positively to the narrative. Notably, Banquo's loyalty and moral integrity contrast sharply with Macbeth's deteriorating character, creating a compelling dynamic that underscores the tragedy of ambition.

In Romeo and Juliet, we see a slightly positive average sentiment score that reflects their passionate love story. Their relationship, however, is set against a backdrop of familial conflict and societal expectations, contributing to the overall tragic tone of the play. The antagonists, notably Tybalt and Paris, display a more negative sentiment. Tybalt's aggressive nature and Paris's socially enforced pursuit of Juliet create obstacles for the young lovers. Their average sentiment score underscores the intense familial and societal conflicts that frame the narrative. Supporting characters, including Benvolio, Mercutio, Nurse, and the Capulet and Montague families, have average sentiment scores that fluctuate, with some providing comic relief while others deepen the tragedy through their actions and responses to the central conflict.

Plot 3 Language Use by Gender

```
############ Gendered Language Analysis ############
# Create a list of known female characters
female_characters <- c("Queen Gertrude", "Ophelia", "Player Queen", "Lady Macbeth", '

# Assign gender
combined_plays <- filtered_plays %>%
  mutate(gender = ifelse(character %in% female_characters, "Female", "Male"))

gender_conts <- combined_plays %>%
  group_by(gender) %>%  # Group by the existing gender column
  summarise(count = n()) %>%  # Count occurrences
  mutate(proportion = count / sum(count))


# Unnest the dialogue into tokens
tidy_text <- combined_plays %>% unnest_tokens(word, dialogue)

# Join with Bing
sentiment_data <- tidy_text %>% inner_join(get_sentiments("bing"), by = "word")

# Group sentiments by gender
gender_sentiment <- sentiment_data %>%
  group_by(gender, sentiment.y) %>%
  summarise(word_count = n()) %>%
```

```r
  ungroup()

# Create a bar plot
gender1 <- ggplot(gender_sentiment, aes(x = gender, y = word_count, fill = sentiment.
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2") +
  labs(title = "Sentiment by Gender",
       x = NULL,
       y = "Word Count")+
  theme(laxis.title.x = element_blank(),
        plot.title = element_text(size = 14, face = "bold"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = "None",
        axis.text.x = element_blank(),  # Remove x-axis text
        axis.ticks.x = element_blank()) +  # Remove x-axis ticks) +
  coord_flip() +
  geom_text(aes(label = word_count),
            position = position_dodge(width = 0.9),
            hjust = 1.3,
            color = "black",
            size = 4)
# Male characters express a higher count of both positive and negative words compared


# Type-Token Ratio (TTR) evaluates the richness or diversity of vocabulary in a text.

# Calculate lexical richness by grouping by gender
lexical_richness <- tidy_text %>%
  group_by(gender) %>%
  summarise(unique_words = n_distinct(word), total_words = n()) %>%
  mutate(ttr = unique_words / total_words)  # Type-Token Ratio (TTR)

# Create a bar plot for Type-Token Ratio (TTR) by gender
gender2 <- ggplot(lexical_richness, aes(x = gender, y = ttr, fill = gender)) +
  geom_bar(stat = "identity") +
  scale_fill_brewer(palette = "Set2") +
  theme_minimal() +
  labs(title = "Language Use by Gender",
       x = NULL,
       y = "Lexical Richness")+
  theme(#axis.title.x = element_blank(),
        plot.title = element_text(size = 14, face = "bold"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = "None",
        axis.text.x = element_blank(),  # Remove x-axis text
        axis.ticks.x = element_blank()) +  # Remove x-axis ticks
  coord_flip() +
  geom_text(aes(label = round(ttr, 2)),
            position = position_dodge(width = 0.9),
            hjust = 1.3,
            color = "black",
```

```
            size = 5)

# We see that female characters use a relatively diverse range of vocabulary in their

# With male characters, we see that their vocabulary is less diverse than that of fem

# Even though women may be portrayed as more emotionally expressive or complex, male
```