Untitled

Import Required Libraries

```
library(dplyr)      # For data manipulation
```

```
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(readr)      # For reading data
library(ggplot2)    # For plotting
library(patchwork)  # For combining multiple plots
```

```
Warning: package 'patchwork' was built under R version 4.4.2
```

In this section, we import the necessary libraries for data manipulation (`dplyr`) and reading CSV files (`readr`). These libraries will be essential for our data analysis tasks.

Read Data

First, we read the CSV file containing COVID-19 data.

```r
covid_data <- read_csv("covid_data_filtered.csv")
```

```
Rows: 12850 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): week_ending_date, state, data_as_of
dbl (7): COVID_deaths_weekly, COVID_deaths_total, crude_COVID_rate_weekly, a...

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.
```

Here, we load the COVID-19 data from a CSV file into a data frame called `covid_data`. This dataset will be used for further analysis and computation of mortality rates.

Data Preprocessing

We convert the `week_ending_date` column to date format and define some dashed dates.

```r
# Convert week_ending_date to date format
covid_data$week_ending_date <- as.Date(covid_data$week_ending_date, format = "%Y/%m/%d")

# Dashed dates
dashed_dates <- as.Date(c("2020-07-01", "2021-02-01", "2021-09-01", "2022-06-01", "2023-03-0
```

In this part, we convert the `week_ending_date` column from a character string to a date format for proper date manipulation. We also define specific dashed dates that will be used to categorize the data into different periods.

Calculate Average Mortality Rate

Next, we calculate the average mortality rate for each state in each time period.

```r
average_mortality_by_state <- covid_data %>%
  # Add period column
  mutate(period = case_when(
    week_ending_date < dashed_dates[1] ~ "Before 2020-07-01",
    week_ending_date >= dashed_dates[1] & week_ending_date < dashed_dates[2] ~ "2020-07-01 t
    week_ending_date >= dashed_dates[2] & week_ending_date < dashed_dates[3] ~ "2021-02-01 t
    week_ending_date >= dashed_dates[3] & week_ending_date < dashed_dates[4] ~ "2021-09-01 t
```

```
    week_ending_date >= dashed_dates[4] & week_ending_date < dashed_dates[5] ~ "2022-06-01 to
    week_ending_date >= dashed_dates[5] & week_ending_date < dashed_dates[6] ~ "2023-03-01 to
    TRUE ~ "After 2024-12-01"
  )) %>%
  # Calculate average mortality rate for each state and period
  group_by(state, period) %>%
  summarise(average_mortality_rate = mean(crude_COVID_rate_weekly, na.rm = TRUE), .groups =
```

Next, we create a new column called `period` to categorize the data based on the predefined dashed dates. Afterward, we calculate the average mortality rate for each state and period, aggregating the crude COVID rate while ignoring any missing values.

Output Results

We output the results and save them as a CSV file.

```
# Output results
print(average_mortality_by_state)
```

```
# A tibble: 300 x 3
   state period                    average_mortality_rate
   <chr> <chr>                                      <dbl>
 1 AK    2020-07-01 to 2021-02-01                   3.36
 2 AK    2021-02-01 to 2021-09-01                   2.46
 3 AK    2021-09-01 to 2022-06-01                   5
 4 AK    2022-06-01 to 2023-03-01                   0
 5 AK    2023-03-01 to 2024-12-01                   0
 6 AK    Before 2020-07-01                          0
 7 AL    2020-07-01 to 2021-02-01                   5.29
 8 AL    2021-02-01 to 2021-09-01                   2.56
 9 AL    2021-09-01 to 2022-06-01                   4.11
10 AL    2022-06-01 to 2023-03-01                   0.878
# i 290 more rows
```

```
# Write the data frame to a CSV file
write_csv(average_mortality_by_state, "average_mortality_by_state.csv")
```

Get the List of States and Calculate the Number of Plots Needed

We assume that `average_mortality_by_state` is a data frame containing states and their corresponding mortality rates. We will extract a list of all states and calculate how many plots we need to create:

```
state_list <- unique(average_mortality_by_state$state)  # Get a list of unique states
num_states <- length(state_list)                          # Calculate the number of states

# Plot 10 states per plot
states_per_plot <- 10
num_plots <- ceiling(num_states / states_per_plot)       # Calculate the total number of plot
```

Grouping and Plotting

Next, we will group the data by state and create plots for each group:

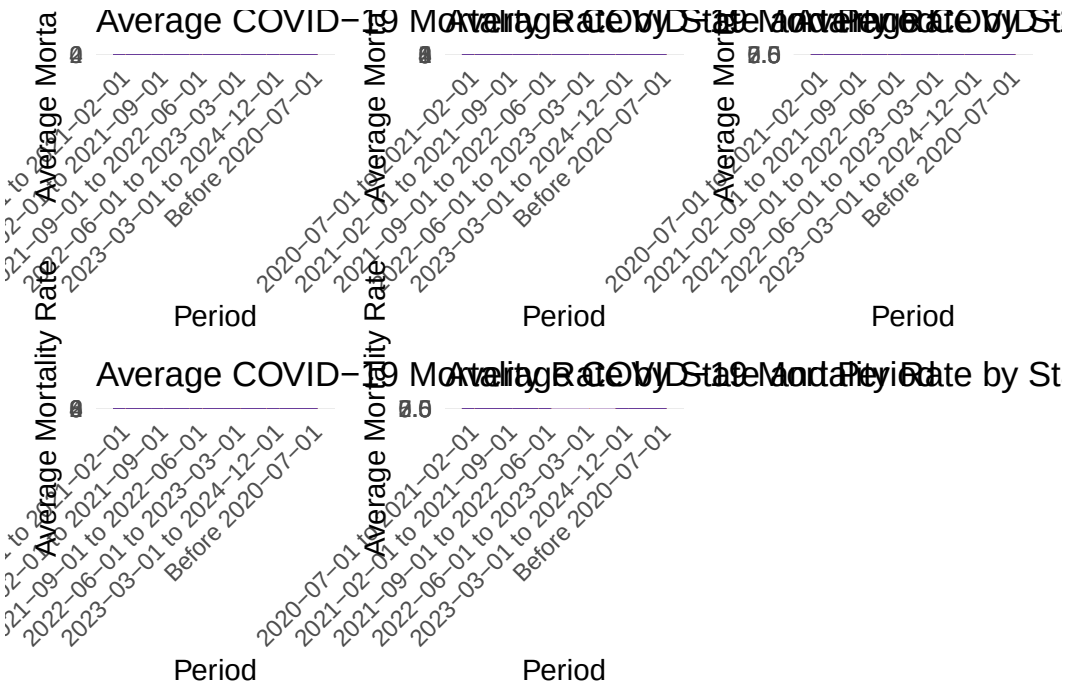
```
plots <- list()  # Create a list to store each ggplot object
for (i in 1:num_plots) {
  start_index <- (i - 1) * states_per_plot + 1  # Starting index for the current plot
  end_index <- min(i * states_per_plot, num_states)  # Ending index for the current plot
  states_to_plot <- state_list[start_index:end_index]  # States to be plotted

  # Filter the data for the current plot
  data_to_plot <- average_mortality_by_state %>% filter(state %in% states_to_plot)

  # Loop to draw the line plot
  p <- ggplot(data_to_plot, aes(x = period, y = average_mortality_rate, group = state, color
    geom_line() +  # Add line
    geom_point() +  # Add data points
    labs(title = paste("Average COVID-19 Mortality Rate by State and Period", sep = ""),
         x = "Period", y = "Average Mortality Rate") +  # Add labels
    theme_minimal() +  # Use a clean and simple theme
    theme(axis.text.x = element_text(angle = 45, hjust = 1), legend.position = "none") +  # S
    scale_color_brewer(palette = "Paired")  # Set color palette

  plots[[i]] <- p  # Store the ggplot object
}
```

Combining Plots Using Patchwork

Finally, we use the `patchwork` package to combine all individual plots into a single large plot and make necessary layout adjustments:

```
combined_plot <- wrap_plots(plots, ncol = 3) +  # Set the number of plots per row
  plot_layout(nrow = ceiling(num_plots / 3), ncol = 3) +  # Adjust rows and columns based on
  plot_spacer()  # Reduce space between plots

# Display the combined large plot
print(combined_plot)
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).

Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).

Warning: Removed 3 rows containing missing values or values outside the scale range
(`geom_point()`).
```



The other plots are likewise:

read the COVID-19 data from CSV file.

```
# Read the CSV file
covid_data <- read_csv("covid_data_filtered.csv")
```

```
Rows: 12850 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): week_ending_date, state, data_as_of
dbl (7): COVID_deaths_weekly, COVID_deaths_total, crude_COVID_rate_weekly, a...

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.
```

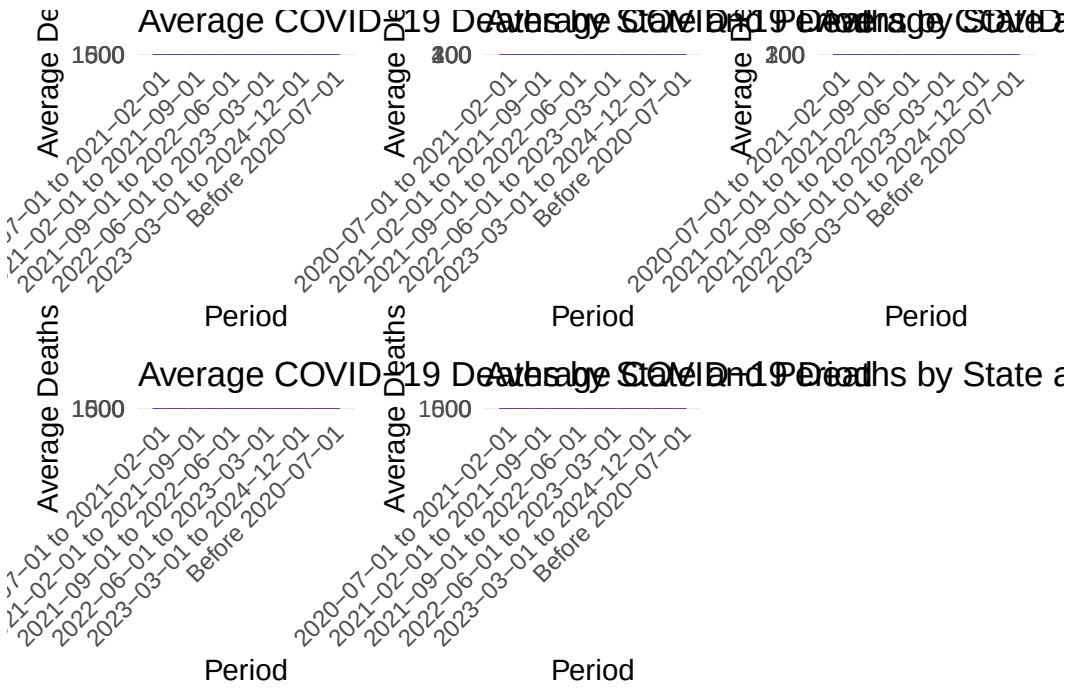
```
# Convert week_ending_date to date format
covid_data$week_ending_date <- as.Date(covid_data$week_ending_date, format = "%Y/%m/%d")
```

```
# Dates represented by dashed lines
dashed_dates <- as.Date(c("2020-07-01", "2021-02-01", "2021-09-01", "2022-06-01", "2023-03-01

# Calculate the average number of deaths per state in each time period
average_deaths_by_state <- covid_data %>%
  mutate(period = case_when(
    week_ending_date < dashed_dates[1] ~ "Before 2020-07-01",
    week_ending_date >= dashed_dates[1] & week_ending_date < dashed_dates[2] ~ "2020-07-01 to
    week_ending_date >= dashed_dates[2] & week_ending_date < dashed_dates[3] ~ "2021-02-01 to
    week_ending_date >= dashed_dates[3] & week_ending_date < dashed_dates[4] ~ "2021-09-01 to
    week_ending_date >= dashed_dates[4] & week_ending_date < dashed_dates[5] ~ "2022-06-01 to
    week_ending_date >= dashed_dates[5] & week_ending_date < dashed_dates[6] ~ "2023-03-01 to
    TRUE ~ "After 2024-12-01"
  )) %>%
  group_by(state, period) %>%
  summarise(average_deaths = mean(COVID_deaths_weekly, na.rm = TRUE), .groups = 'drop')
```

plot line charts based on the number of states, with each chart displaying 10 states.

```
# Get the list of states
state_list <- unique(average_deaths_by_state$state)
num_states <- length(state_list)
```

```r
# Plot 10 states per graph
states_per_plot <- 10
num_plots <- ceiling(num_states / states_per_plot)

# Plotting by groups
plots <- list()  # Create a list to store each ggplot object
for (i in 1:num_plots) {
  # Calculate the range of states for the current graph
  start_index <- (i - 1) * states_per_plot + 1
  end_index <- min(i * states_per_plot, num_states)
  states_to_plot <- state_list[start_index:end_index]

  # Filter data for the current graph
  data_to_plot <- average_deaths_by_state %>% filter(state %in% states_to_plot)

  # Plot the line graph
  p <- ggplot(data_to_plot, aes(x = period, y = average_deaths, group = state, color = state)
    geom_line() +
    geom_point() +
    labs(title = "Average COVID-19 Deaths by State and Period",
         x = "Period",
         y = "Average Deaths") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          legend.position = "none") +  # Hide legend
    scale_color_brewer(palette = "Paired")

  plots[[i]] <- p  # Store each ggplot object in the list
}

# Combine charts using patchwork
combined_plot <- wrap_plots(plots, ncol = 3) +
  plot_layout(nrow = ceiling(num_plots / 3), ncol = 3) +
  plot_spacer()  # Reduce whitespace between charts

# Display the combined large graph
print(combined_plot)
```




```
# Get the list of states
state_list <- unique(average_mortality_by_state$state)
num_states <- length(state_list)

# Plot 10 states per graph
states_per_plot <- 10
num_plots <- ceiling(num_states / states_per_plot)
```

```
# Create a list to store each ggplot object
plots <- list()

for (i in 1:num_plots) {
  # Calculate the range of states for the current plot
  start_index <- (i - 1) * states_per_plot + 1
  end_index <- min(i * states_per_plot, num_states)
  states_to_plot <- state_list[start_index:end_index]

  # Filter data for the current plot
  data_to_plot <- average_mortality_by_state %>% filter(state %in% states_to_plot)

  # Create a line plot
  p <- ggplot(data_to_plot, aes(x = period, y = average_mortality_rate, group = state, color
    geom_line() +
```

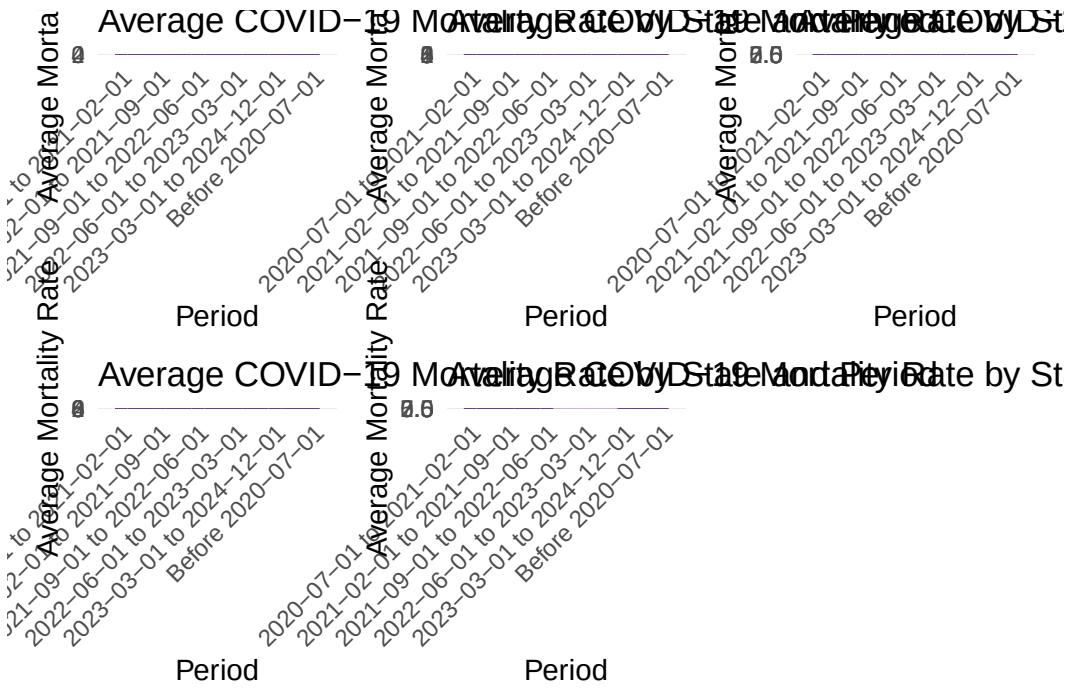
```
    geom_point() +
    labs(title = "Average COVID-19 Mortality Rate by State and Period",
         x = "Period",
         y = "Average Mortality Rate") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          legend.position = "none") +  # Hide legend
    scale_color_brewer(palette = "Paired")

  plots[[i]] <- p
}

# Combine plots using patchwork
combined_plot <- wrap_plots(plots, ncol = 3) +
  plot_layout(nrow = ceiling(num_plots / 3), ncol = 3) +
  plot_spacer()  # Reduce whitespace between plots

# Display the combined large plot
print(combined_plot)
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_point()`).
```

```
Warning: Removed 3 rows containing missing values or values outside the scale range
(`geom_point()`).
```



Conclusion

This analysis demonstrates the variation in COVID-19 mortality rates across states. By visualizing the data, we can better observe the differences between states.