data visualization (lab-2)

Pradip Basnet

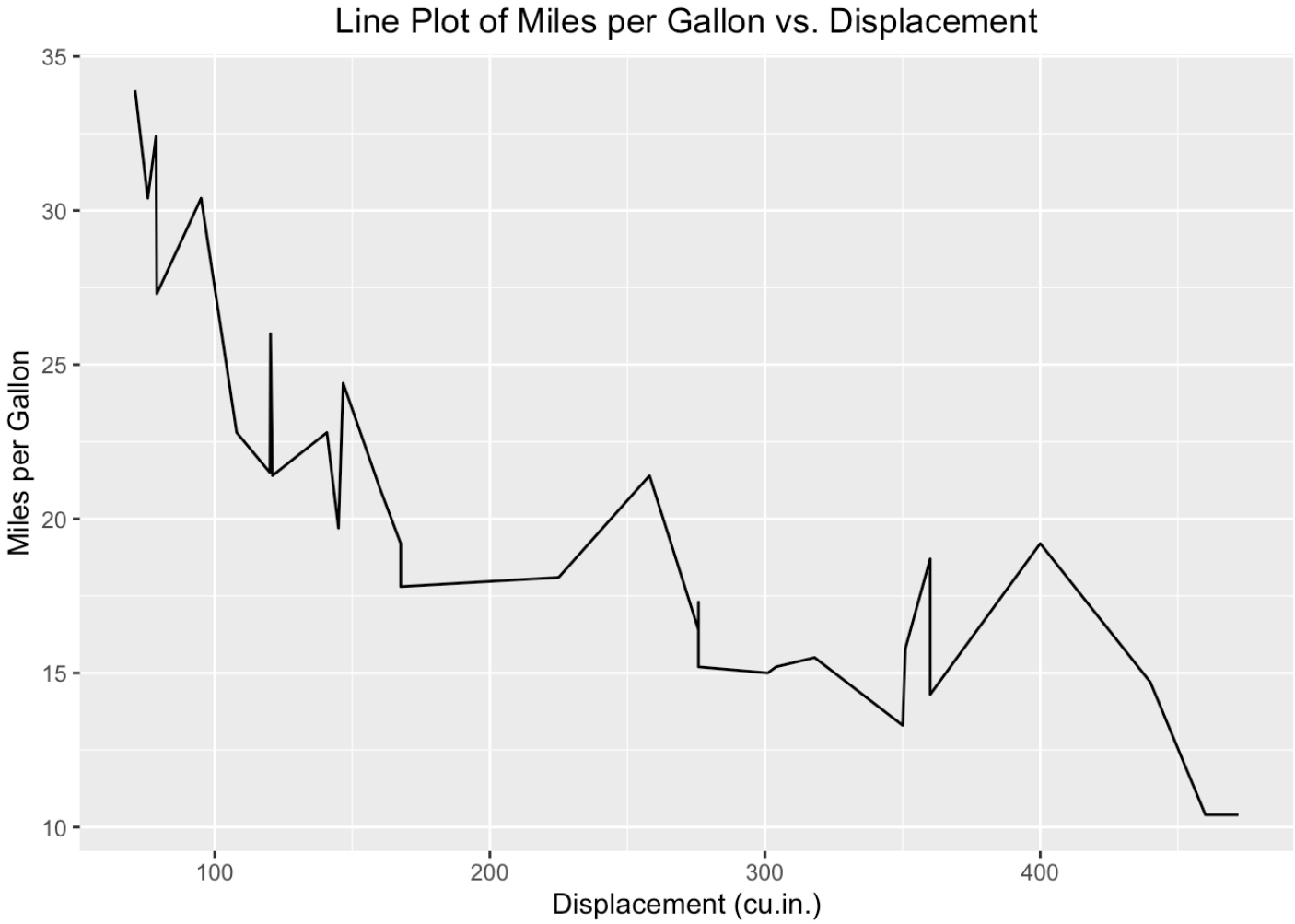
##loading and importing the necessary package

```
library(ggplot2)
```

#1. Create a simple line plot using ggplot2 with mpg (miles per gallon) on the y-axis and disp (displacement) on the x-axis from the 'mtcars' dataset.

##Interpretaton: there is negative correlation between miles per gallon and displacement as the engine displacement increases the miles per gallon generally decreases however, there are some anomalies that do not follow the general trend.

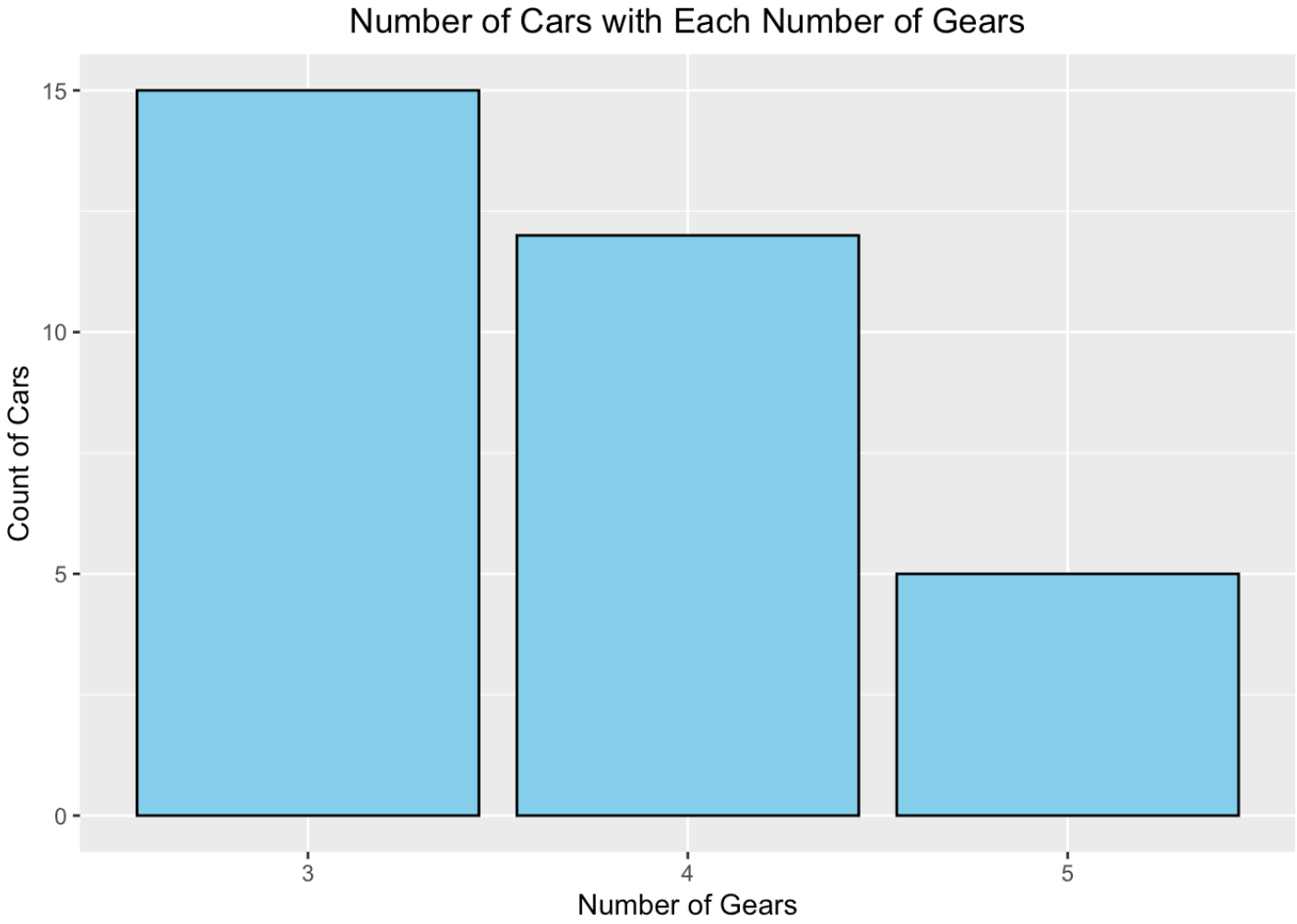
```
data(mtcars) #loading the required mtcars datset

ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_line() +
  labs(
    title = "Line Plot of Miles per Gallon vs. Displacement",
    x = "Displacement (cu.in.)",
    y = "Miles per Gallon"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```



#2. Using the ‘mtcars’ dataset, create a bar plot showing the number of cars with each number of gears (gear).

##Interpretation: The bar plot shows the distribution of cars based on the number of gears. It highlights that the most common configuration is 3 gears, followed by 4 gears, with 5-gear cars being the least common

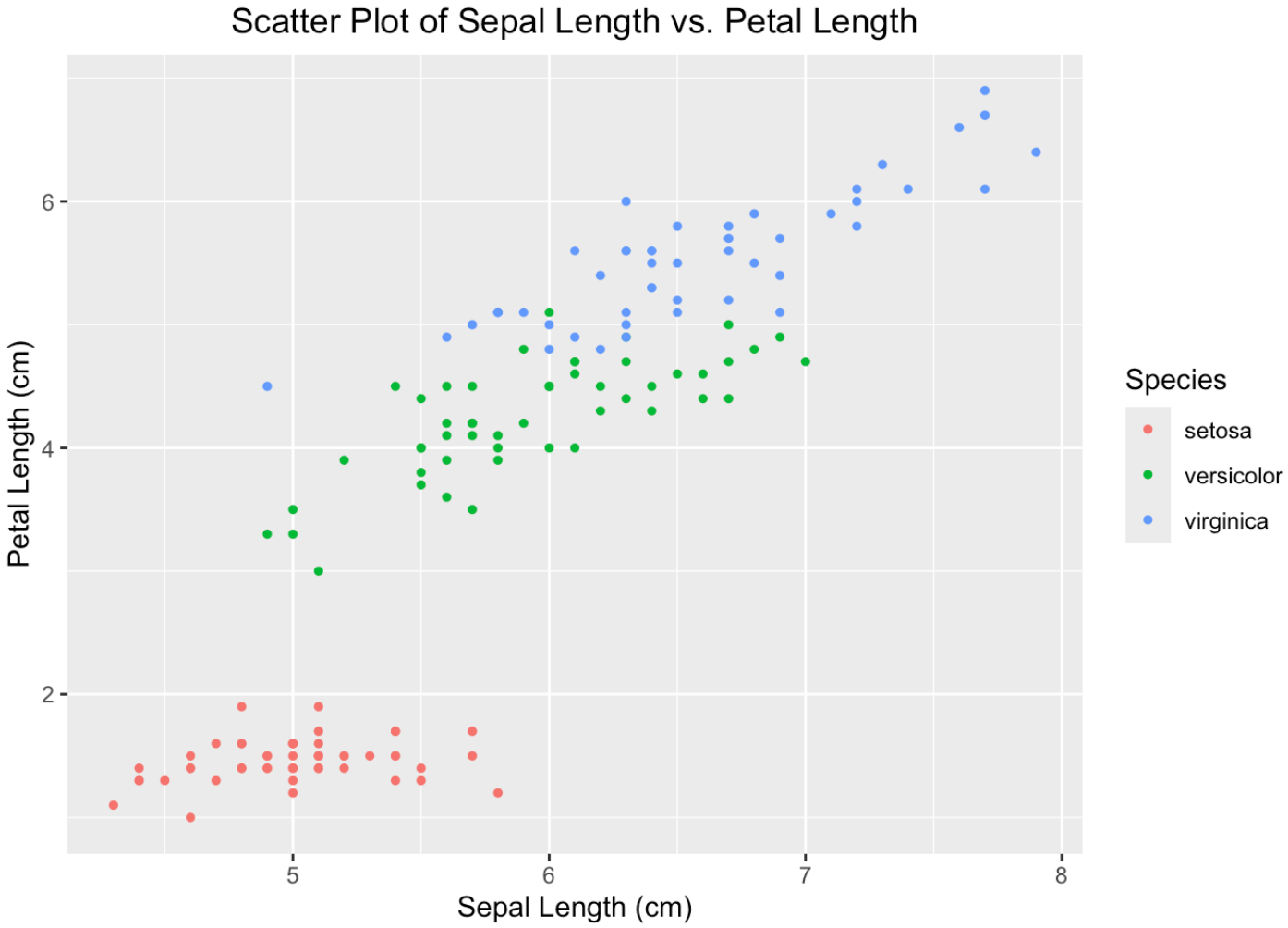
```
ggplot(mtcars, aes(x = factor(gear))) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(
    title = "Number of Cars with Each Number of Gears",
    x = "Number of Gears",
    y = "Count of Cars"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```



#3. Create a scatter plot using the iris dataset, mapping Sepal.Length to x, Petal.Length to y, and Species to color.

##Interpretation: there is a positive corellation between sepal length and petal length.This means that as the sepal length increases, the petal length also tends to increase. However, the correlation is not perfect, and there is a significant amount of scatter in the data. ris setosa flowers tend to have shorter sepals and petals than the other two species. Iris versicolor and Iris virginica flowers have a wider range of sepal and petal lengths, but they generally tend to have longer sepals and petals than Iris setosa flowers

```
data("iris") #loading the required iris datset

ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_point(size = 1) +
  labs(
    title = "Scatter Plot of Sepal Length vs. Petal Length",
    x = "Sepal Length (cm)",
    y = "Petal Length (cm)"
  ) +

  theme(
    plot.title = element_text(hjust = 0.5)
  )
```



#4. Create a bar plot showing the average mpg (miles per gallon) for each ‘cyl’ (number of cylinders) in the ‘mtcars’ dataset. Color bars based on ‘cyl’ and add error bars to represent the standard deviation

#Interpretation: the chart reveals that the cars with fewer cylinders tend to get better mileage than the cars with more cylinders. This is likely because cars with fewer cylinders typically have smaller engines, which require less fuel to operate.In the chart, four-cylinder cars get the best gas mileage, at around 30 MPG.ix-cylinder cars get an average of around 20 MPG.Eight-cylinder cars get the worst gas mileage, at around 10 MPG.

loading the necessary package

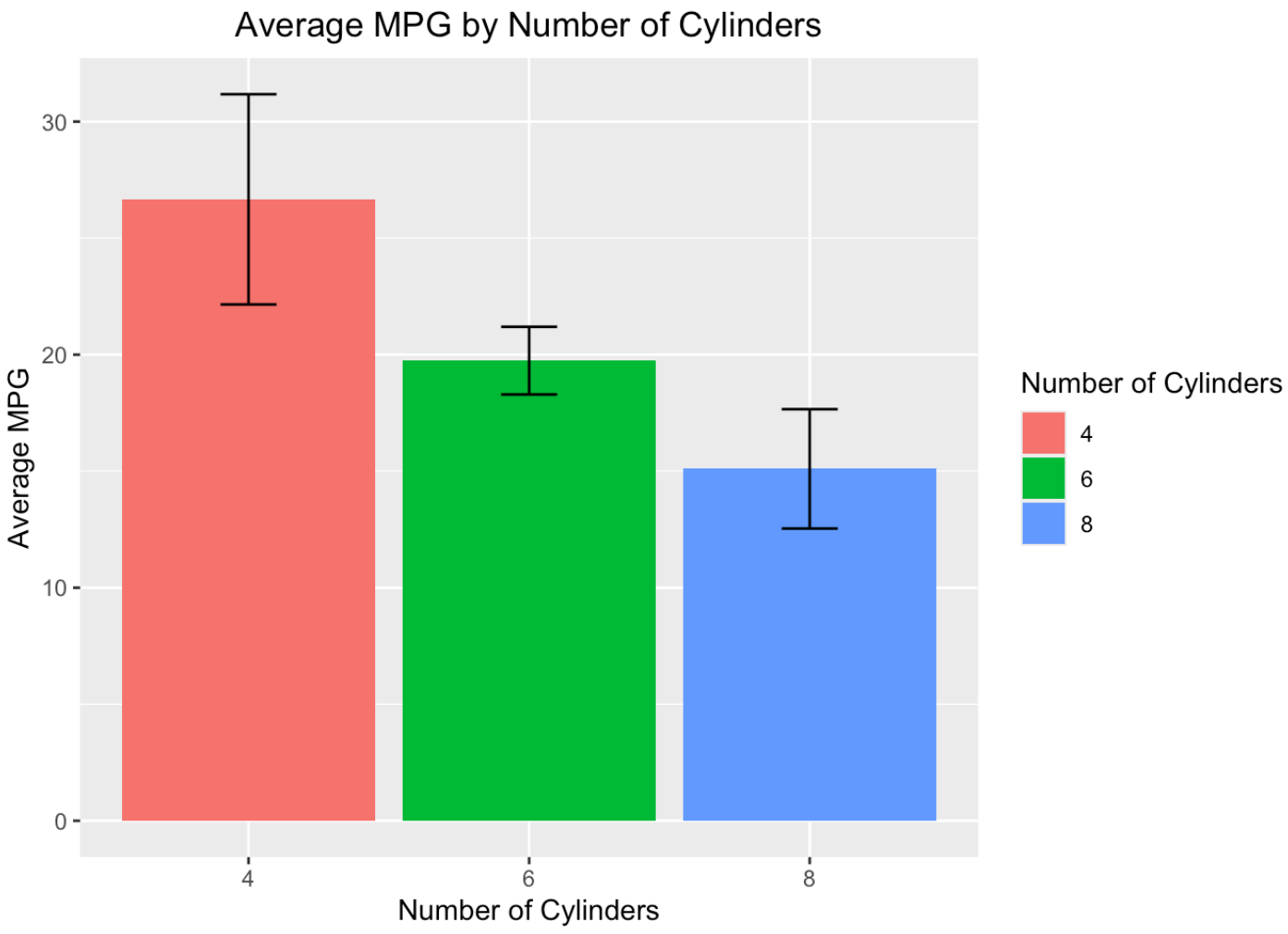
```
#install.packages("dplyr")
library(dplyr)
```

```
##
## 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
```

```
# Calculate mean mpg and standard deviation by cyl
mpg_summary <- mtcars %>%
  group_by(cyl) %>%
  summarise(mean_mpg = mean(mpg),
            sd_mpg = sd(mpg))

# Plotting
ggplot(mpg_summary, aes(x = factor(cyl), y = mean_mpg, fill = factor(cyl))) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar(aes(ymin = mean_mpg - sd_mpg, ymax = mean_mpg + sd_mpg),
                width = 0.2, position = position_dodge(width = 0.9)) +
  labs(x = "Number of Cylinders", y = "Average MPG", fill = "Number of Cylinders")
+
  ggtitle("Average MPG by Number of Cylinders") +
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```

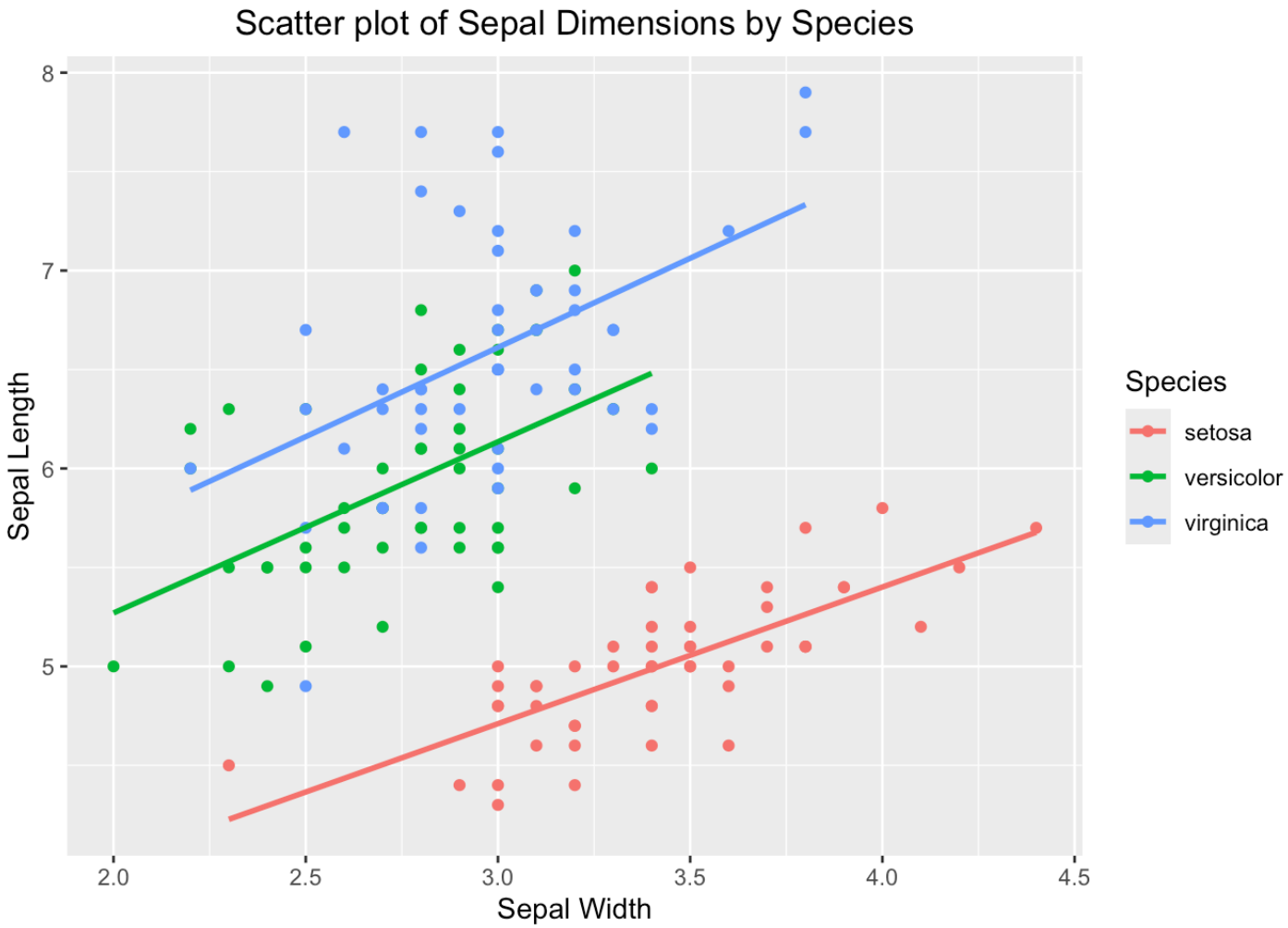


#5. Using the iris dataset, create a scatter plot mapping Sepal.Width to x and Sepal.Length to y, colored by Species. Add a linear regression line for each species.

##Interpretation: The regression line for all three variables has a positive slope, indicating that as Sepal.Width increases, Sepal.Length also tends to increase.All three species show a positive correlation between these two variables, but the strength of the correlation and the range of values vary between species. Setosa has the smallest Sepal.Length values, versicolor has intermediate values, and virginica has the largest Sepal.Length values.

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Scatter plot of Sepal Dimensions by Species",
       x = "Sepal Width",
       y = "Sepal Length") +
 theme(
    plot.title = element_text(hjust = 0.5)
  )
```

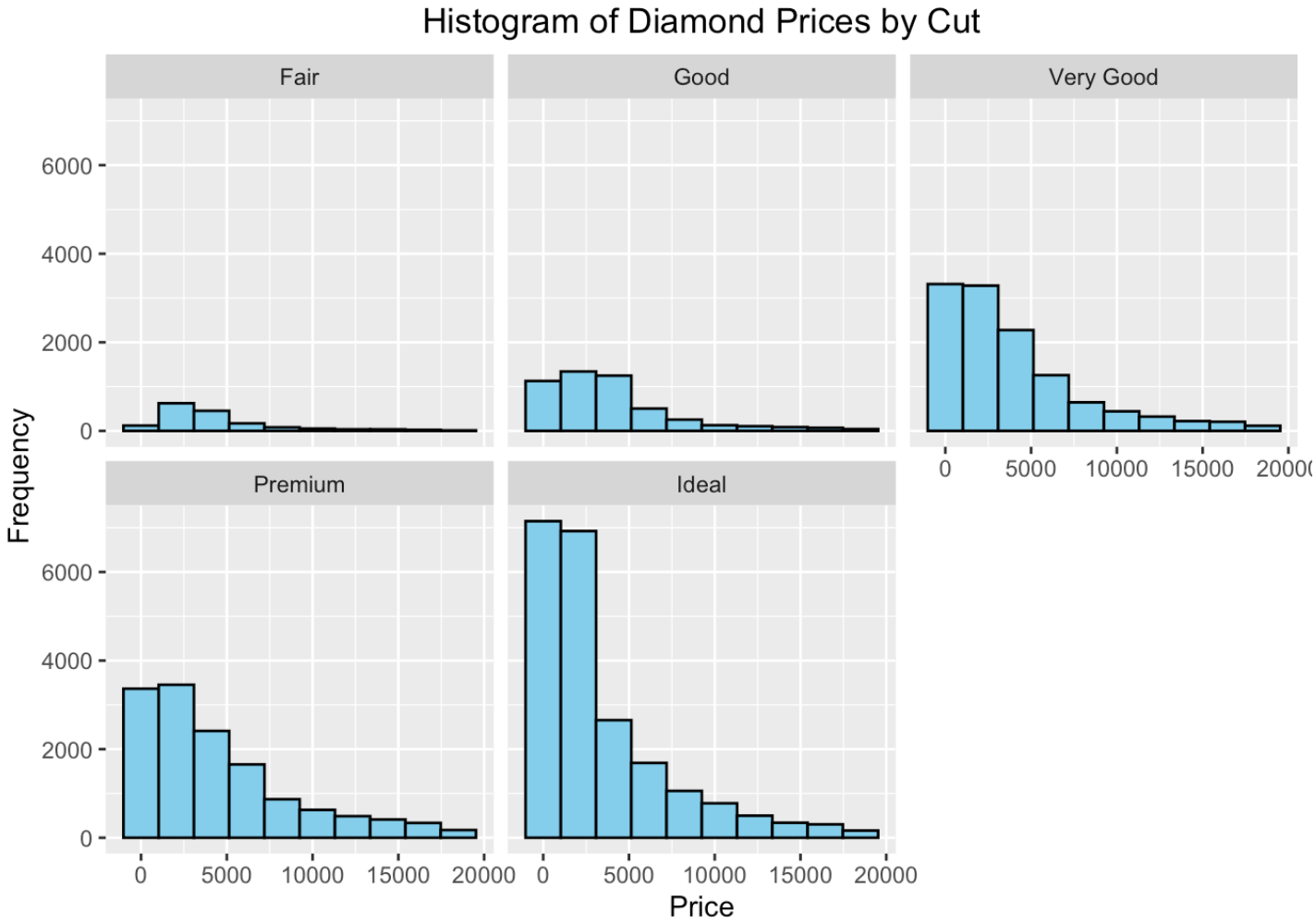
```
## `geom_smooth()` using formula = 'y ~ x'
```



#6. Using the diamonds dataset, create faceted histograms for price, split by cut. Adjust the bins so they are suitable for the data's distribution.

##Interpretation: The histogram shows that the price distribution varies across the different diamond cuts.the histogram demonstrate a trend between diamond cut and price. Diamonds with higher cut grades (Very Good, Premium, Ideal) tend to have a wider price range and potentially higher prices compared to diamonds with lower cut grades (Fair, Good).

```
ggplot(diamonds, aes(x = price)) +
  geom_histogram(bins = 10, fill = "skyblue", color = "black") +  # Adjust bins as
needed
  facet_wrap(~ cut) +
  labs(title = "Histogram of Diamond Prices by Cut",
       x = "Price",
       y = "Frequency") +
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```

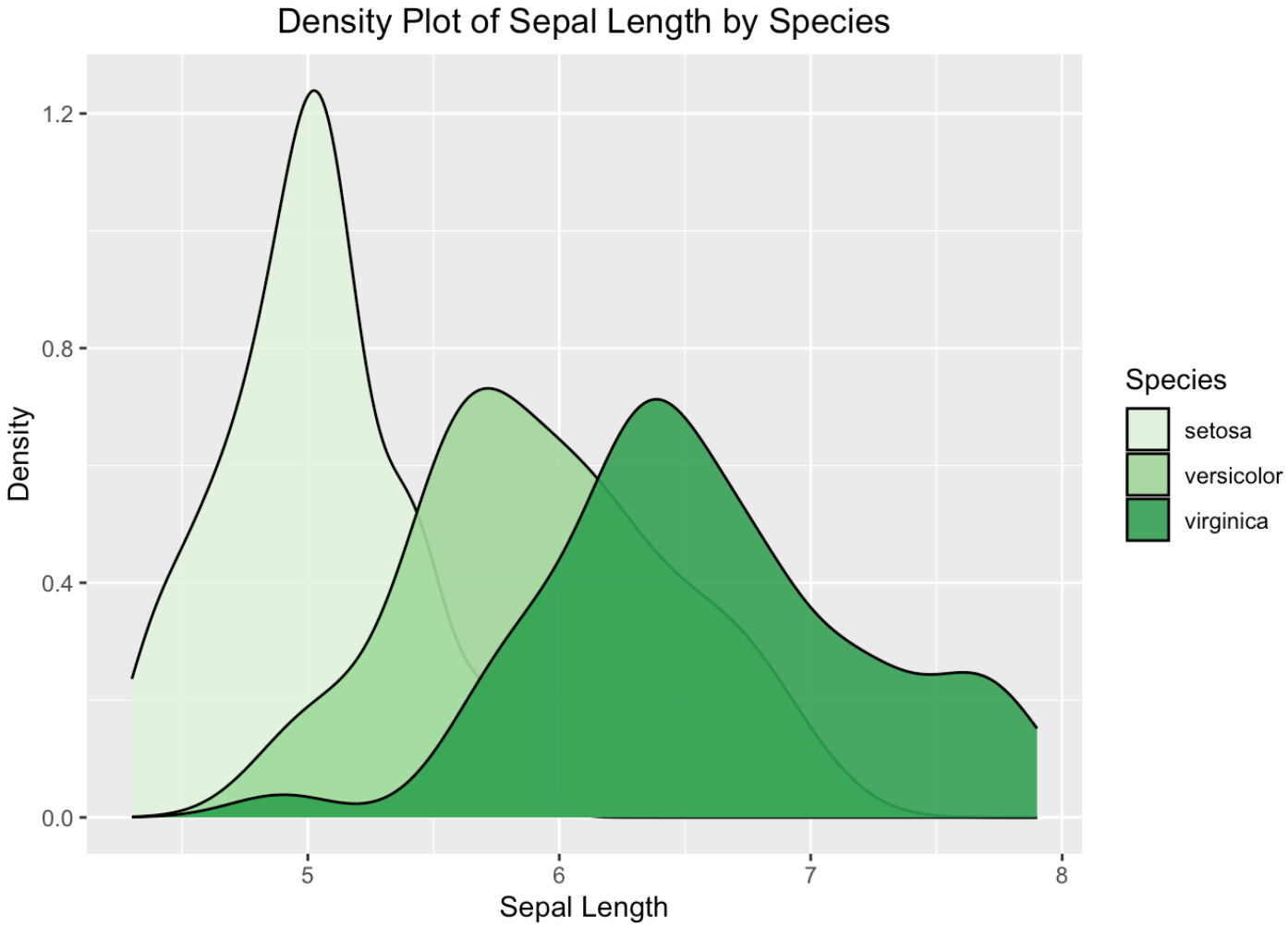


#7. Create a density plot for Sepal.Length from the iris dataset, fill based on Species, and apply a different color palette.

##Interpretation: the density plot reveals that setosa has the smallest sepals, versicolor has intermediate sepal lengths, and virginica has the largest sepals on average. There is likely some overlap in sepal length distribution between versicolor and setosa, but minimal overlap between versicolor and virginica

```
ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.9) +
  scale_fill_brewer(palette = "Dark1") +
  labs(title = "Density Plot of Sepal Length by Species",
       x = "Sepal Length",
       y = "Density") +
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```

```
## Warning: Unknown palette: "Dark1"
```



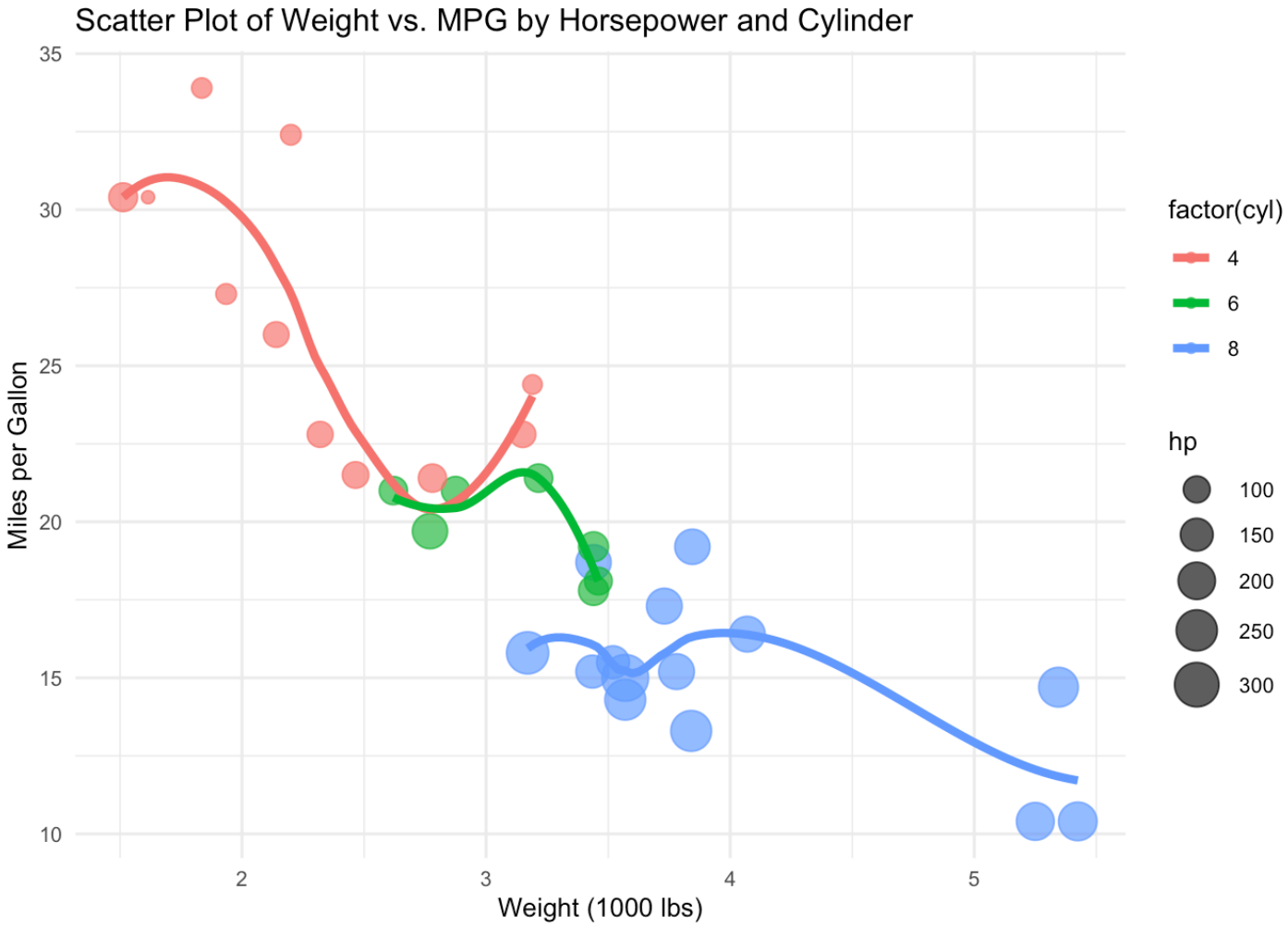
#8. Using the 'mtcars' dataset, create a scatter plot of wt (weight) vs. mpg (miles per gallon), size the points by hp (horsepower), and color them by 'cyl' (cylinders). Add smooth lines to show the trend for each cylinder group.

##Interpretation: the scatter plot reveals that there is negative correlation between mpg and wt, as the weight of the car increases, the fuel efficiency of the car decreases.the chart shows that weight and horsepower have an impact on the fuel efficiency of vehicles, with lighter and lower horsepower vehicles generally achieving higher mpg

```
# Load necessary library
library(ggplot2)

# Create scatter plot with size and color aesthetics
ggplot(mtcars, aes(x = wt, y = mpg, size = hp, color = factor(cyl))) +
  geom_point(alpha = 0.7) +  # Add points with transparency
  geom_smooth(method = "loess", se = FALSE, aes(group = cyl), size = 1.5) +  # Add
smooth lines by 'cyl'
  scale_size_continuous(range = c(2, 8)) +  # Adjust size range for better visibil
ity
  labs(x = "Weight (1000 lbs)", y = "Miles per Gallon") +  # Add axis labels
  ggtitle("Scatter Plot of Weight vs. MPG by Horsepower and Cylinder") +  # Add pl
ot title
  theme_minimal() +  # Apply a minimal theme
  theme(
    legend.position = "right",  # Adjust legend position
    text = element_text(size = 10)  # Adjust text size and font
  )
```

```
## 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.
```

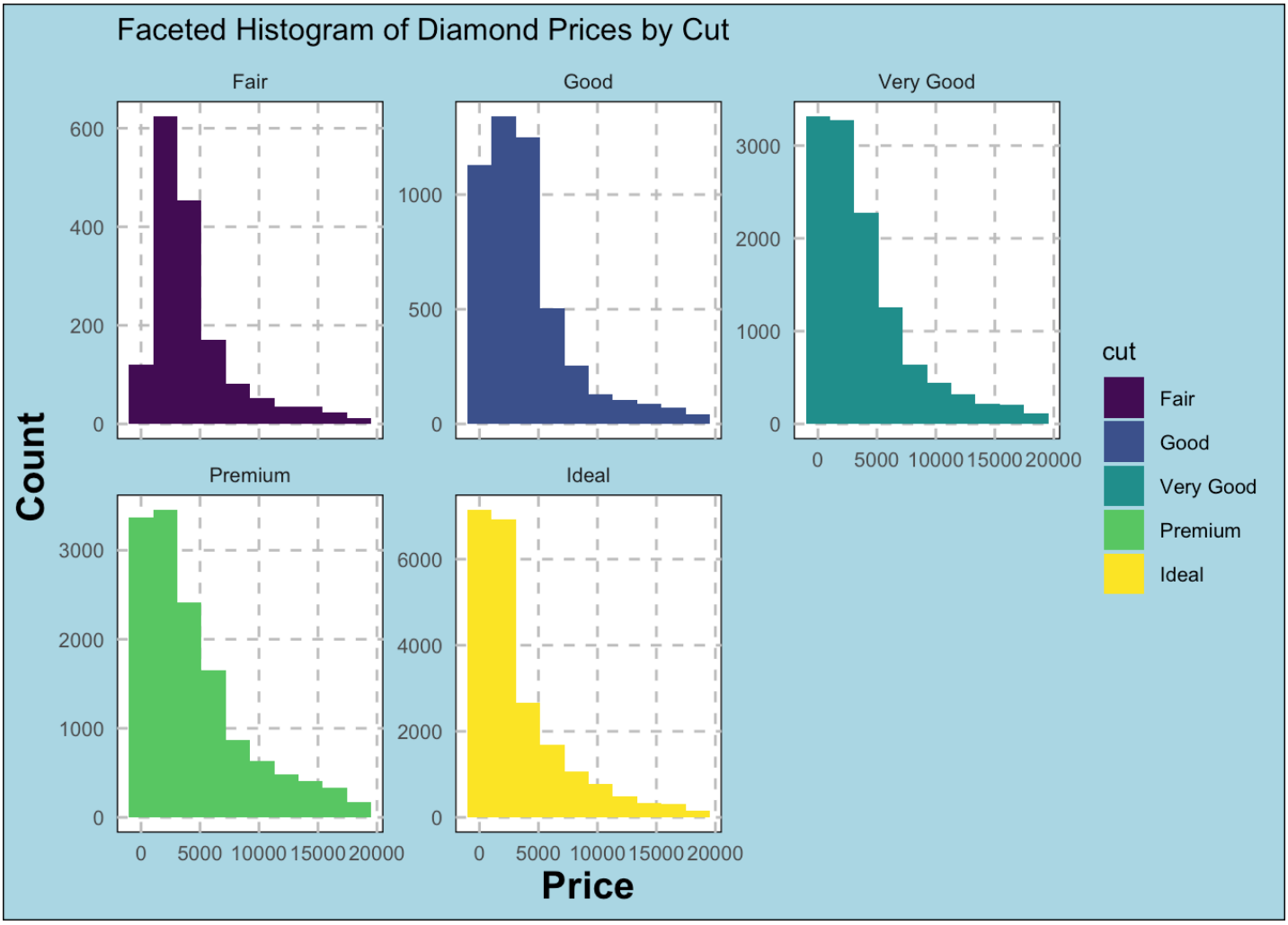
```
## `geom_smooth()` using formula = 'y ~ x'
```



#9. Create any plot of your choice using the ggplot2 package and then customize it using a theme() function. Modify at least five different aspects of the theme (e.g., text size, background color, grid lines).

##Interpretation: at first created a faceted histogram using diamonds dataset, then customize the plot usin theme() Sets the text size to 10 points and font family to Arial for all text elements in the plot. Sets the background color of the plot area to light blue. Modifies major grid lines to be gray and dashed. Removes minor grid lines. Positions the legend to the right of the plot. overall, created a faceted histogram where each facet represents a different diamond cut category (cut). It visualizes the distribution of diamond prices (price) within each cut category, with customizations applied to text, backgrounds, grid lines, axis titles, and legend position using the theme() function

```
# Create the basic plot
p <- ggplot(diamonds, aes(x = price, fill = cut)) +
  geom_histogram(bins = 10) +
  facet_wrap(~ cut, scales = "free_y") +
  labs(x = "Price", y = "Count") +
  ggtitle("Faceted Histogram of Diamond Prices by Cut") +
  theme_minimal()

# Customize the plot using theme() function
p + theme_minimal() +
  theme(
    text = element_text(size = 10, family = "Arial"),
    plot.background = element_rect(fill = "lightblue"),
    panel.background = element_rect(fill = "white"),
    panel.grid.major = element_line(color = "gray", linetype = "dashed"),
    panel.grid.minor = element_blank(),
    axis.title = element_text(size = 15, face = "bold"),
    legend.position = "right"
  )
```



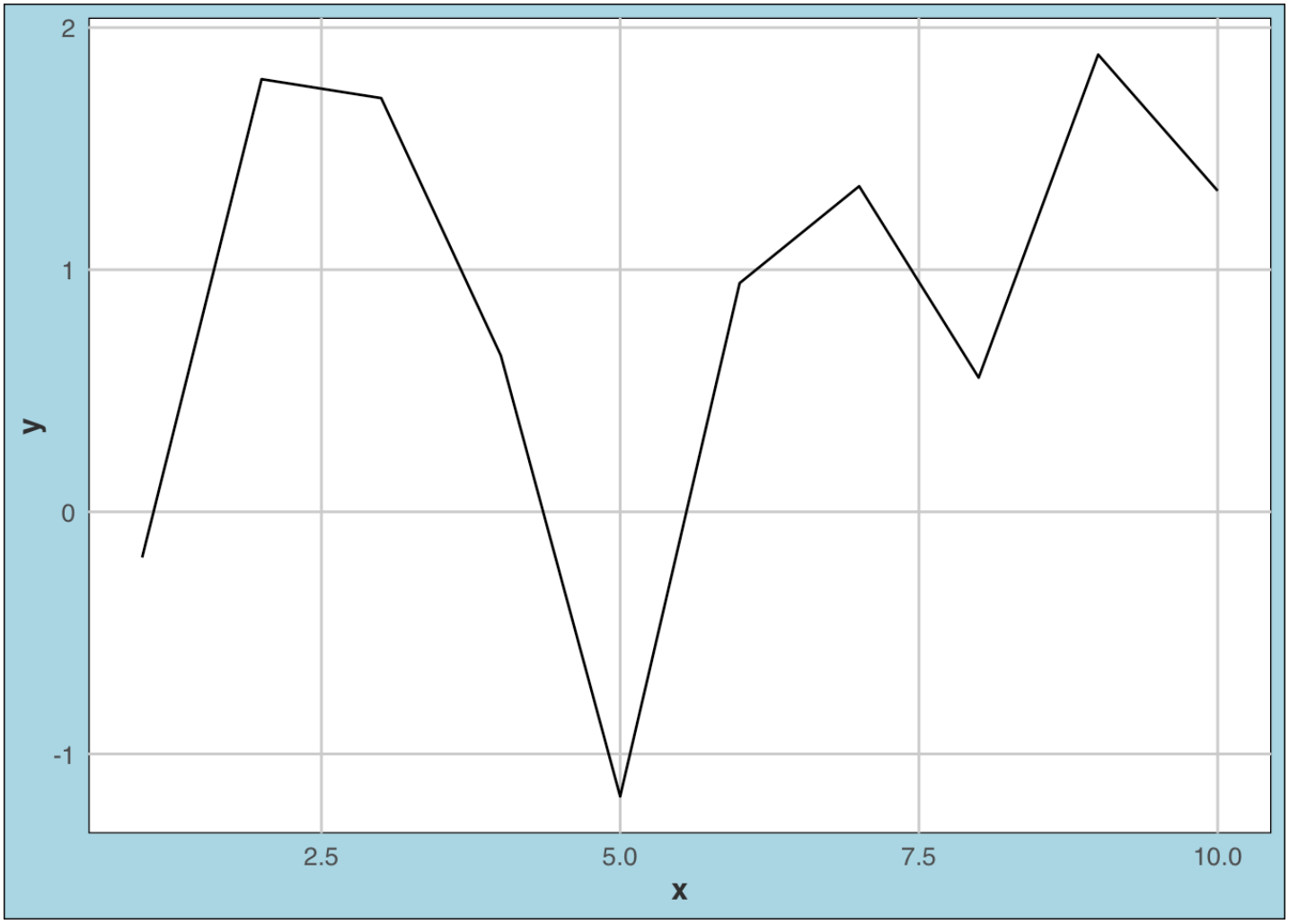
#10. How can you create your custom theme in ggplot2?Carry out the following steps and analyze the output.

##A. DefinetheCustomTheme:

```
my_custom_theme <- function() {
  theme_minimal() +  # Start with a minimal theme
    theme(
      text = element_text(family = "Helvetica", color = "#333333"),
      plot.background = element_rect(fill = "lightblue"),
      panel.background = element_rect(fill = "white"),
      panel.grid.major = element_line(color = "grey80"),
      panel.grid.minor = element_blank(),
      axis.title = element_text(size = 12, face = "bold"),
      axis.text = element_text(size = 10),
      legend.background = element_rect(fill = "white"),
      legend.title = element_text(face = "bold"),
      legend.text = element_text(size = 9)
    )
}
```

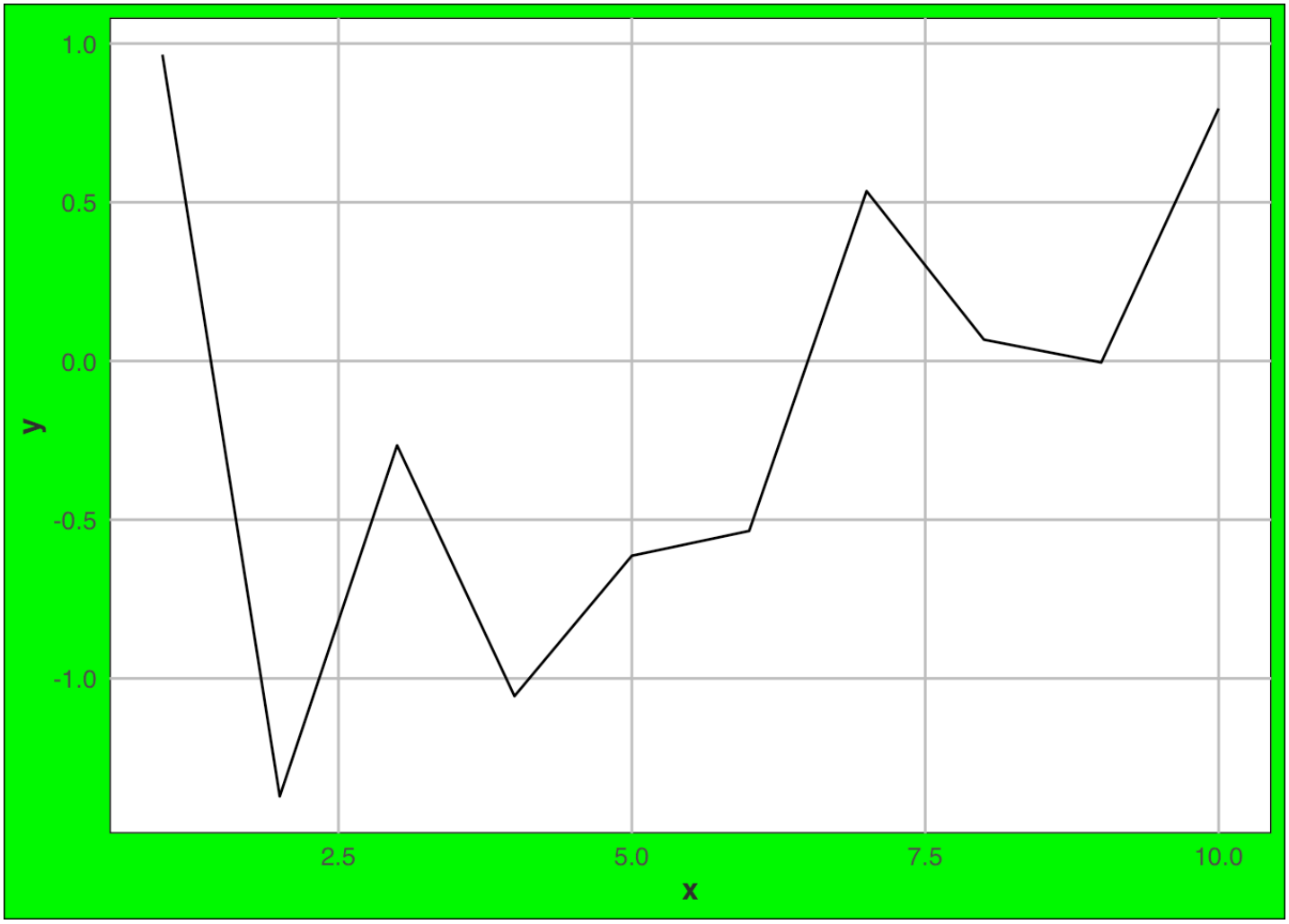
##B. ApplytheCustomTheme:

```
# Sample data
data <- data.frame(
  x = 1:10,
  y = rnorm(10)
)

# Create a plot and apply your custom theme
ggplot(data, aes(x, y)) +
  geom_line() +
  my_custom_theme()  # Apply your custom theme
```



##C. ModifyandReuseYourTheme: One of the benefits of creating your own theme function is that you can easily modify and reuse it. If you decide to change the font size or background color, for example, you can do so in your my_custom_theme function and those changes will apply to any plot where you use the theme.

```
my_custom_theme <- function() {
  theme_minimal() +  # Start with a minimal theme
    theme(
      text = element_text(family = "Helvetica", color = "#333333"),
      plot.background = element_rect(fill = "green"),
      panel.background = element_rect(fill = "white"),
      panel.grid.major = element_line(color = "grey"),
      panel.grid.minor = element_blank(),
      axis.title = element_text(size = 12, face = "bold"),
      axis.text = element_text(size = 10),
      legend.background = element_rect(fill = "white"),
      legend.title = element_text(face = "bold"),
      legend.text = element_text(size = 9)
    )
}

# Sample data
data <- data.frame(
  x = 1:10,
  y = rnorm(10)
)

# Create a plot and apply your custom theme
ggplot(data, aes(x, y)) +
  geom_line() +
  my_custom_theme()  # Apply your custom theme
```



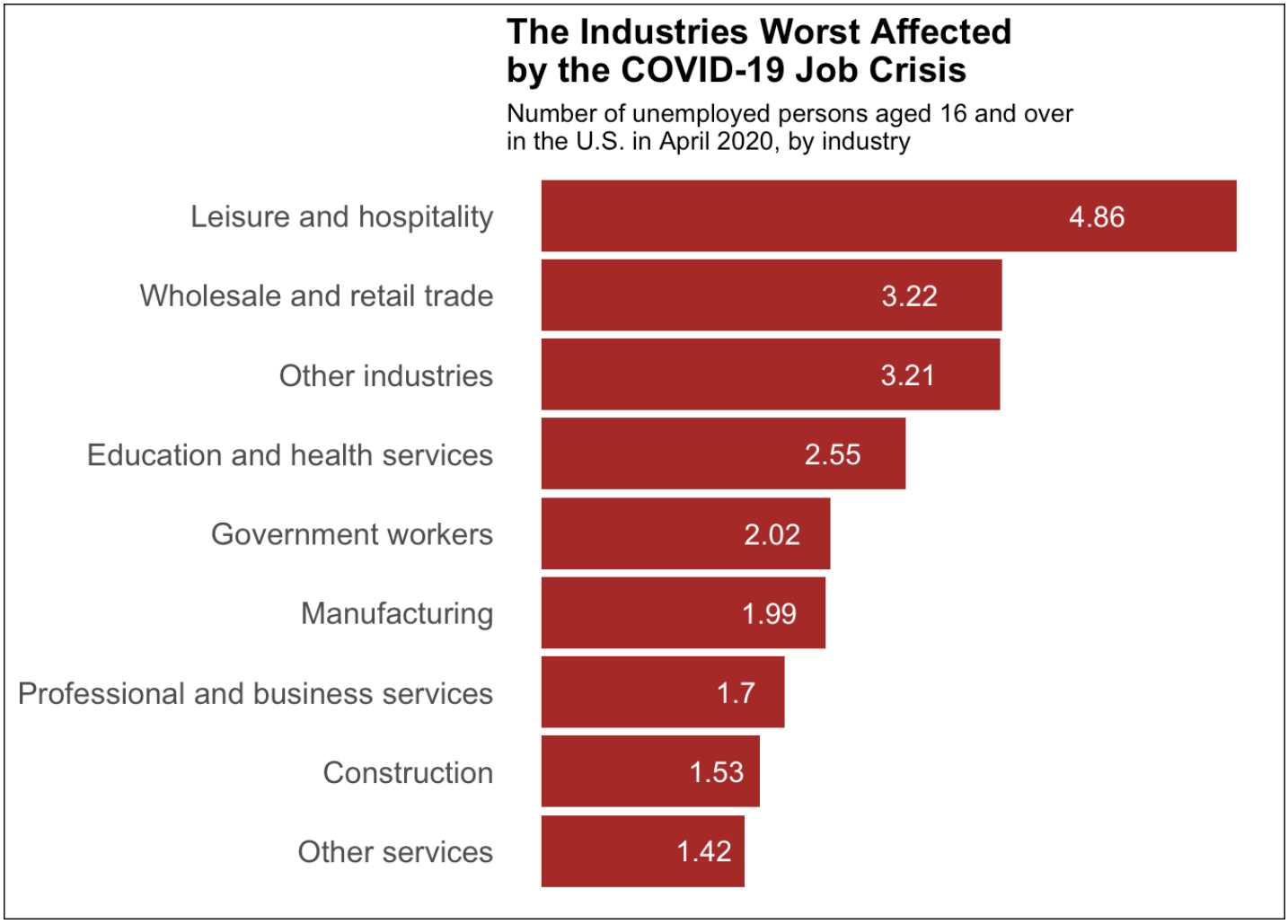
#Dataset

```
# Define the vectors
industries <- c("Leisure and hospitality", "Wholesale and retail trade", "Other in
dustries", "Education and health services",
                "Government workers", "Manufacturing",
                "Professional and business services", "Construction", "Other servi
ces")
unemployed_numbers <- c(4.86, 3.22, 3.21, 2.55, 2.02, 1.99, 1.70, 1.53, 1.42) # in
million
unemployment_rates <- c(0.39, 0.17, 0, 0.11, 0.09, 0.13, 0.10, 0.17, 0.23)  # Conv
ert percentages to proportions (if these were intended to be proportions)

# Create the data frame
job_crisis_data <- data.frame(Industry = industries, UnemployedNumbers = unemploye
d_numbers,
                              UnemploymentRate = unemployment_rates)

# Print the data frame
print(job_crisis_data)
```

```
##                              Industry UnemployedNumbers UnemploymentRate
## 1            Leisure and hospitality              4.86              0.39
## 2         Wholesale and retail trade              3.22              0.17
## 3                    Other industries              3.21              0.00
## 4      Education and health services              2.55              0.11
## 5                  Government workers              2.02              0.09
## 6                       Manufacturing              1.99              0.13
## 7 Professional and business services              1.70              0.10
## 8                        Construction              1.53              0.17
## 9                       Other services              1.42              0.23
```

##Interpetation: The graph illustrates that the leisure and hospitality industry was the most affected by the COVID-19 job crisis, with significantly higher unemployment numbers compared to other industries. Wholesale and retail trade, and other industries also faced substantial unemployment. The remaining industries, while still affected, had comparatively lower numbers of unemployed persons.

```
ggplot(job_crisis_data, aes(x = reorder(Industry, UnemployedNumbers),
                            y = UnemployedNumbers)) +
  geom_bar(stat = "identity", fill = "brown") +
  coord_flip() +
  geom_text(aes(label = UnemployedNumbers),
            position = position_stack(vjust = 0.8),
            hjust = 0.5, color = "white", size = 4) +
  labs(title = "The Industries Worst Affected
by the COVID-19 Job Crisis",
       subtitle = "Number of unemployed persons aged 16 and over
in the U.S. in April 2020, by industry",
       x = NULL, y = NULL) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    plot.subtitle = element_text(size = 10),
    axis.text.x = element_blank(),
    axis.text.y = element_text(size = 12),
    plot.background = element_rect(fill = "white"),
    panel.grid = element_blank()
  )
```


The Industries Worst Affected by the COVID-19 Job Crisis
Number of unemployed persons aged 16 and over in the U.S. in April 2020, by industry
Leisure and hospitality
4.86
Wholesale and retail trade
3.22
Other industries
3.21
Education and health services
2.55
Government workers
2.02
Manufacturing
1.99
Professional and business services
1.7
Construction
1.53
Other services
1.42