Part I

March 27, 2025

1 Part 1: Data Visualization with ggplot2

1.0.1 Instructor: Dr. Maryam Movahedifar

```
<img src="Uni_Logo.png" alt="Uni Logo" style="width: 100px; margin-right: 10px;">
<img src="DSC_Logo.png" alt="DSC Logo" style="width: 150px;">
```

1.1 Objective:

Learn how to create static, insightful visualizations using the `ggplot2` package.

1.2 Topics Covered:

- Installing and loading `ggplot2`
- Understanding the grammar of graphics
- Creating basic plots (scatter, line, bar)
- Faceting for multi-panel plots
- Hands-on exercise

1. **Install ggplot2 if not already installed**

[1]:
```
if (!requireNamespace("ggplot2", quietly = TRUE)) {
  install.packages("ggplot2")  # Install ggplot2 package
}
```

Load the ggplot2 library

[2]:
```
library(ggplot2)  # Load ggplot2 for data visualization
```

1.3 Understanding the Grammar of Graphics

`ggplot2` follows the **Grammar of Graphics**, which consists of:

- **Data**: The dataset you're working with.
- **Aesthetics (`aes`)**: Mapping variables to visual properties.
- **Geometries (`geom`)**: The type of plot (e.g., scatter, line, bar).
- **Facets**: Splitting data into subplots.

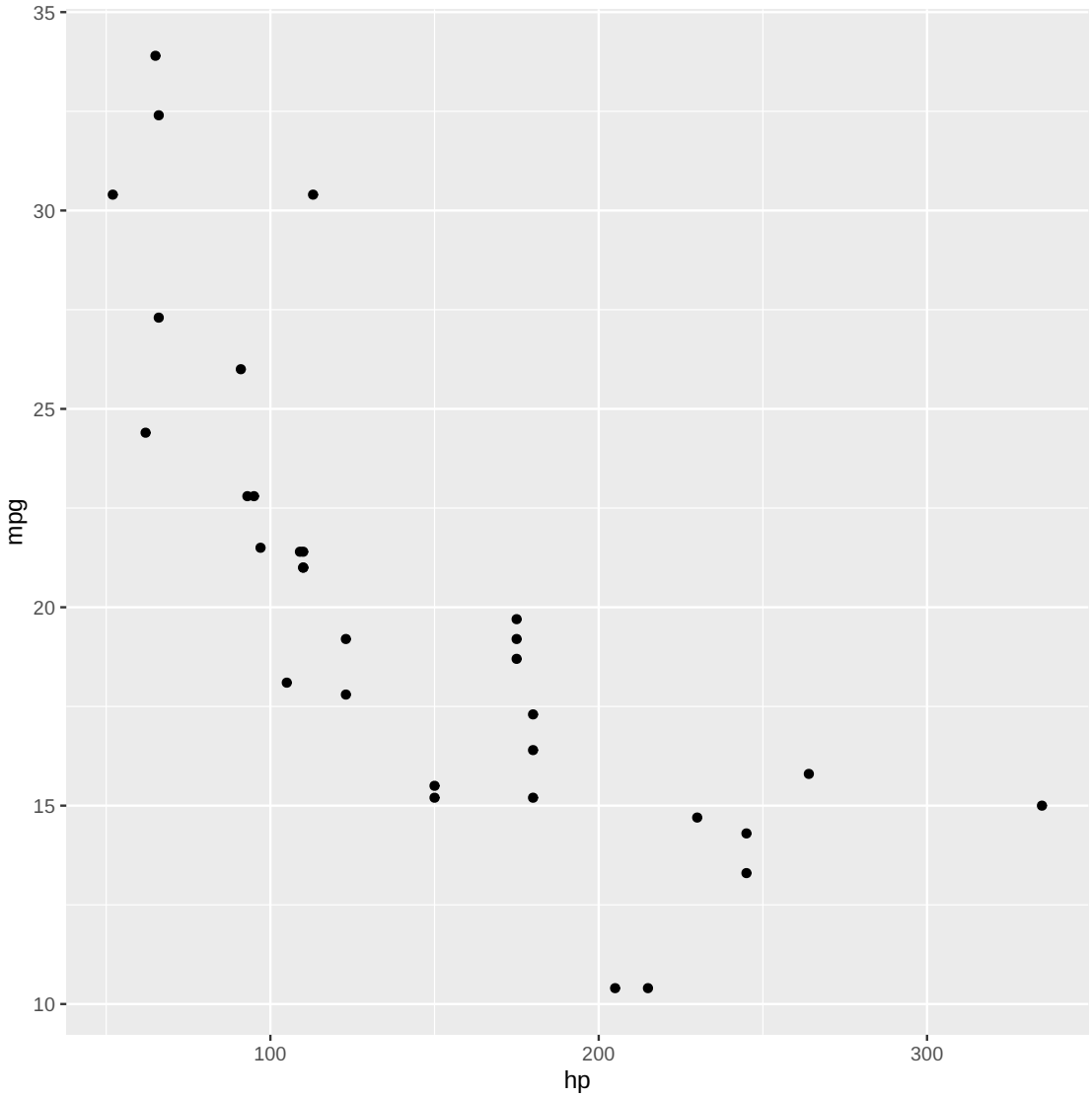
- **Themes**: Adjusting the appearance.

2. **Creating Basic Plots**

1.3.1 Example 1: Scatter Plot

This code creates a scatter plot of `mpg` vs `hp` from the `mtcars` dataset.

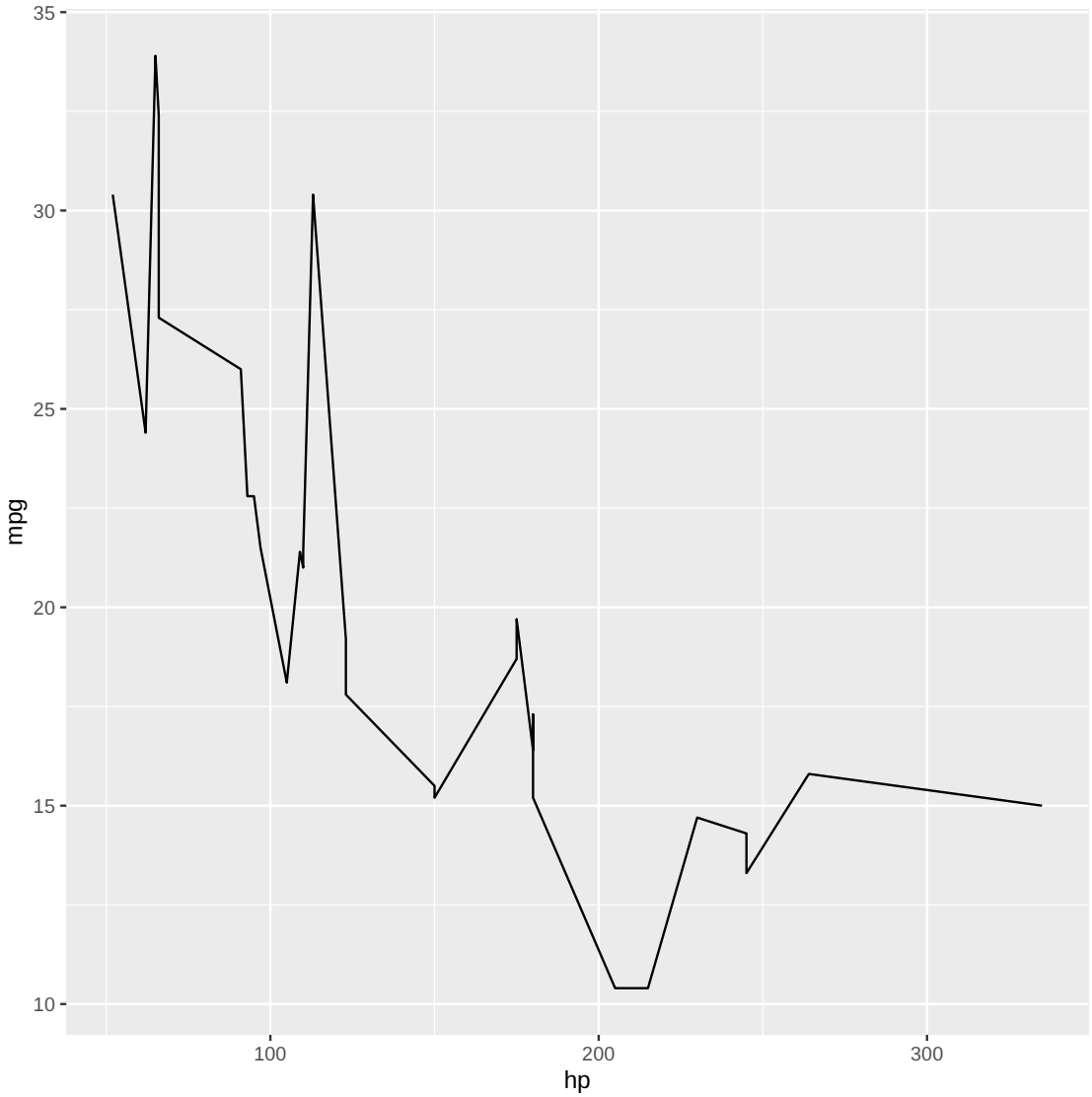
[3]:
```
# Load the built-in mtcars dataset
data(mtcars)  # Load dataset

# Create a basic scatter plot
ggplot(data = mtcars, aes(x = hp, y = mpg)) +  # Map 'hp' to x-axis and 'mpg'␣
↪to y-axis
  geom_point()  # Add points (scatter plot)
```



1.3.2 Example 2: Line Plot

This code creates a line plot showing the relationship between horsepower (`hp`) and miles per gallon (`mpg`).

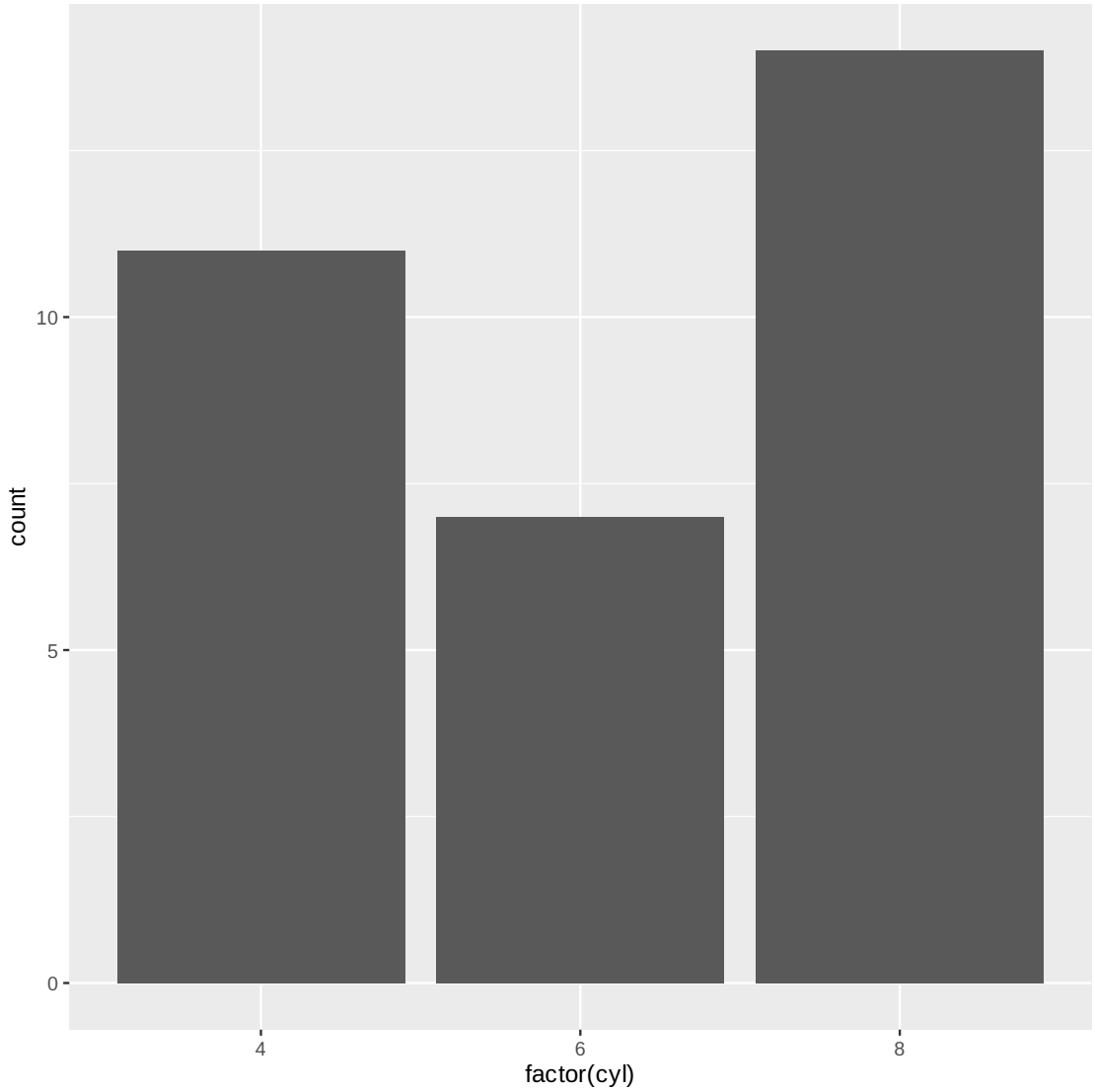
[4]:
```
ggplot(data = mtcars, aes(x = hp, y = mpg)) +  # Define the dataset and␣
↪aesthetics
 geom_line()  # Add a line plot
```



1.3.3 Example 3: Bar Plot

This code creates a bar plot showing the distribution of the number of cylinders (`cyl`) in `mtcars`.

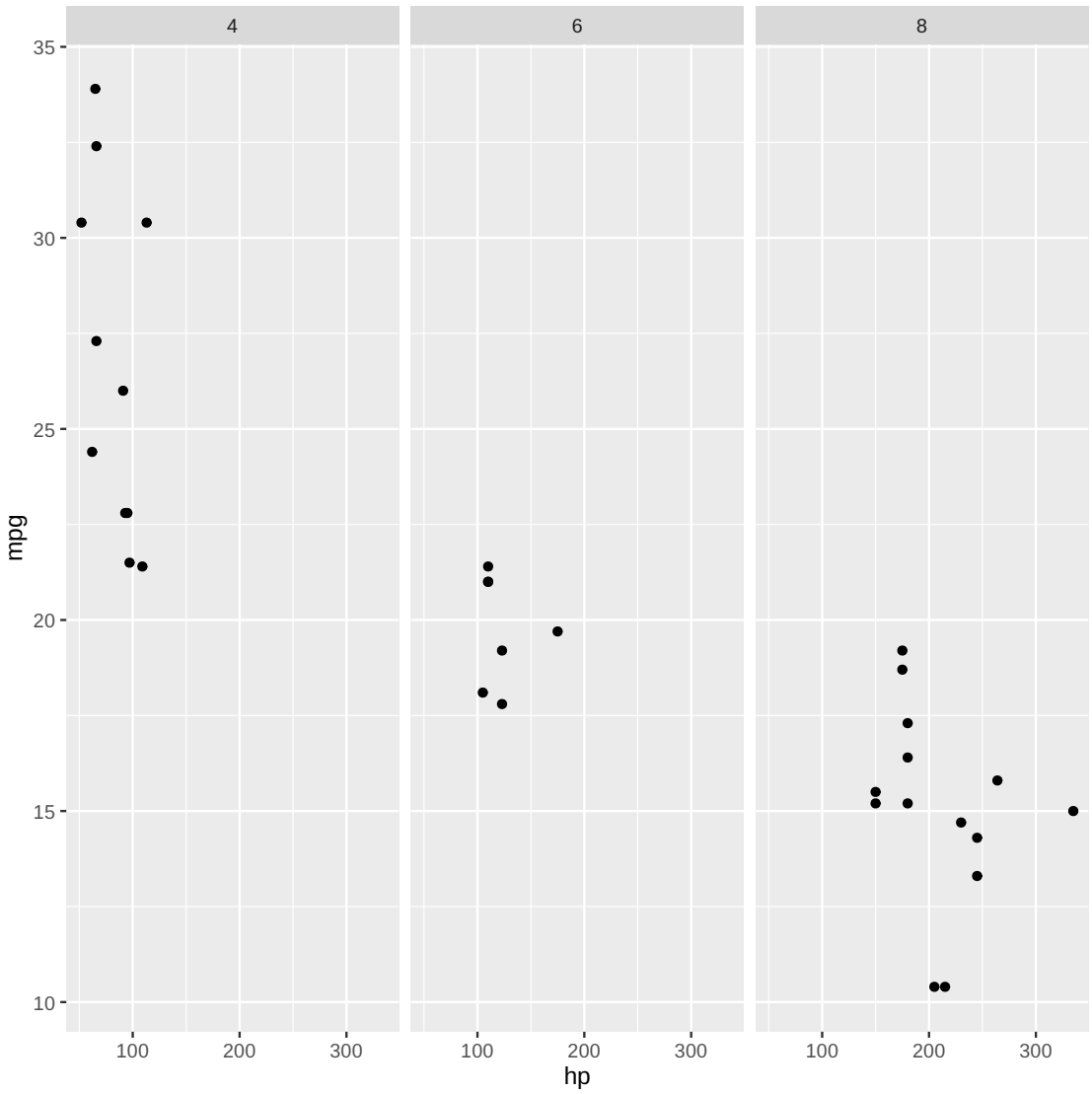
[5]:
```
ggplot(data = mtcars, aes(x = factor(cyl))) +  # Convert 'cyl' to a factor␣
↪(categorical variable)
 geom_bar()  # Create a bar plot
```



1.4 Faceting for Multi-panel Plots

Faceting allows us to create multiple small plots based on a categorical variable.
Here, we split the scatter plot by `cyl` (number of cylinders).

[6]:
```
ggplot(data = mtcars, aes(x = hp, y = mpg)) +  # Define data and mapping
  geom_point() +  # Add scatter plot points
  facet_wrap(~ cyl)  # Create separate plots for each cylinder category
```



1.5 Customizing ggplot2 Visualizations

Once you create your basic plots with ggplot2, the next step is **customizing** them to improve the clarity, style, and presentation of the visuals. Customization in ggplot2 involves:

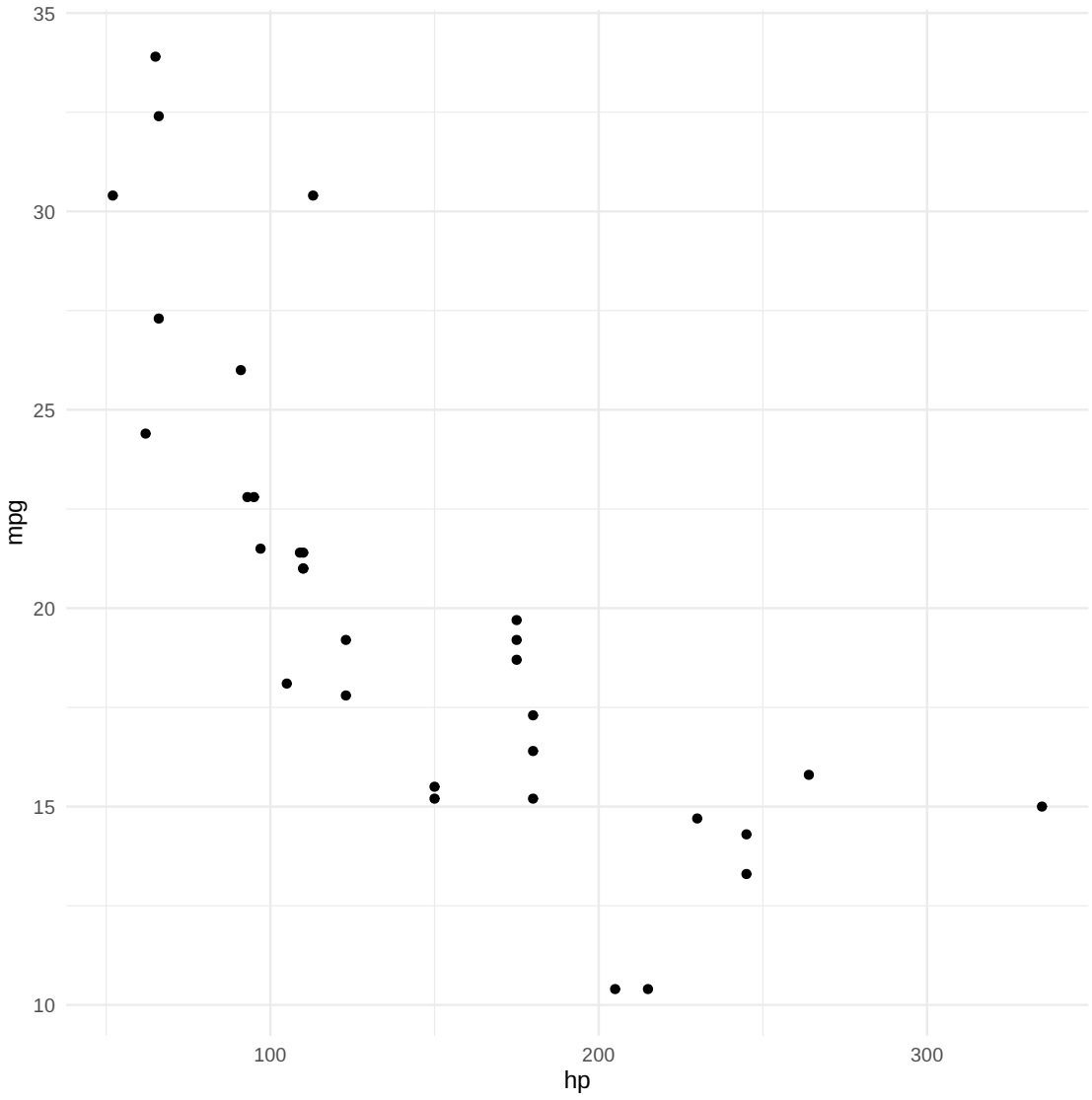
- **Adjusting themes** to modify the plot's background, gridlines, and overall style.
- **Adding labels** to axes, titles, and legends.
- **Changing color schemes** and **aesthetic properties** to make plots more informative and visually appealing.

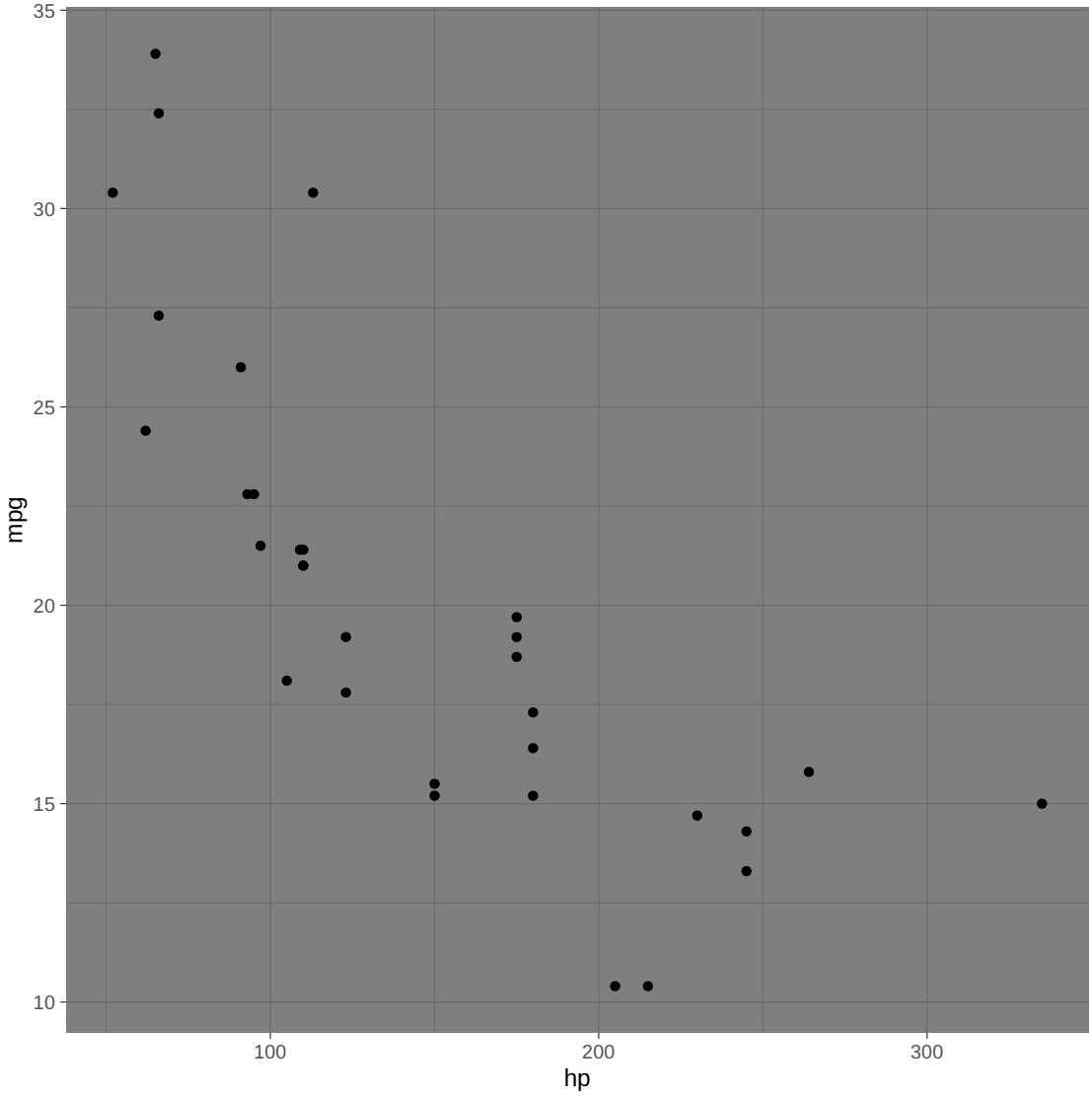
1.6 Customizing Plot Appearance

1.6.1 1. Changing Themes

ggplot2 comes with several built-in themes for adjusting the background, gridlines, and overall plot style. Here's how you can apply them:

[7]:
```
# Basic plot with default theme
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  theme_minimal()  # Apply a minimal theme

# Apply another built-in theme (dark background with light gridlines)
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  theme_dark()  # Apply a dark theme
```



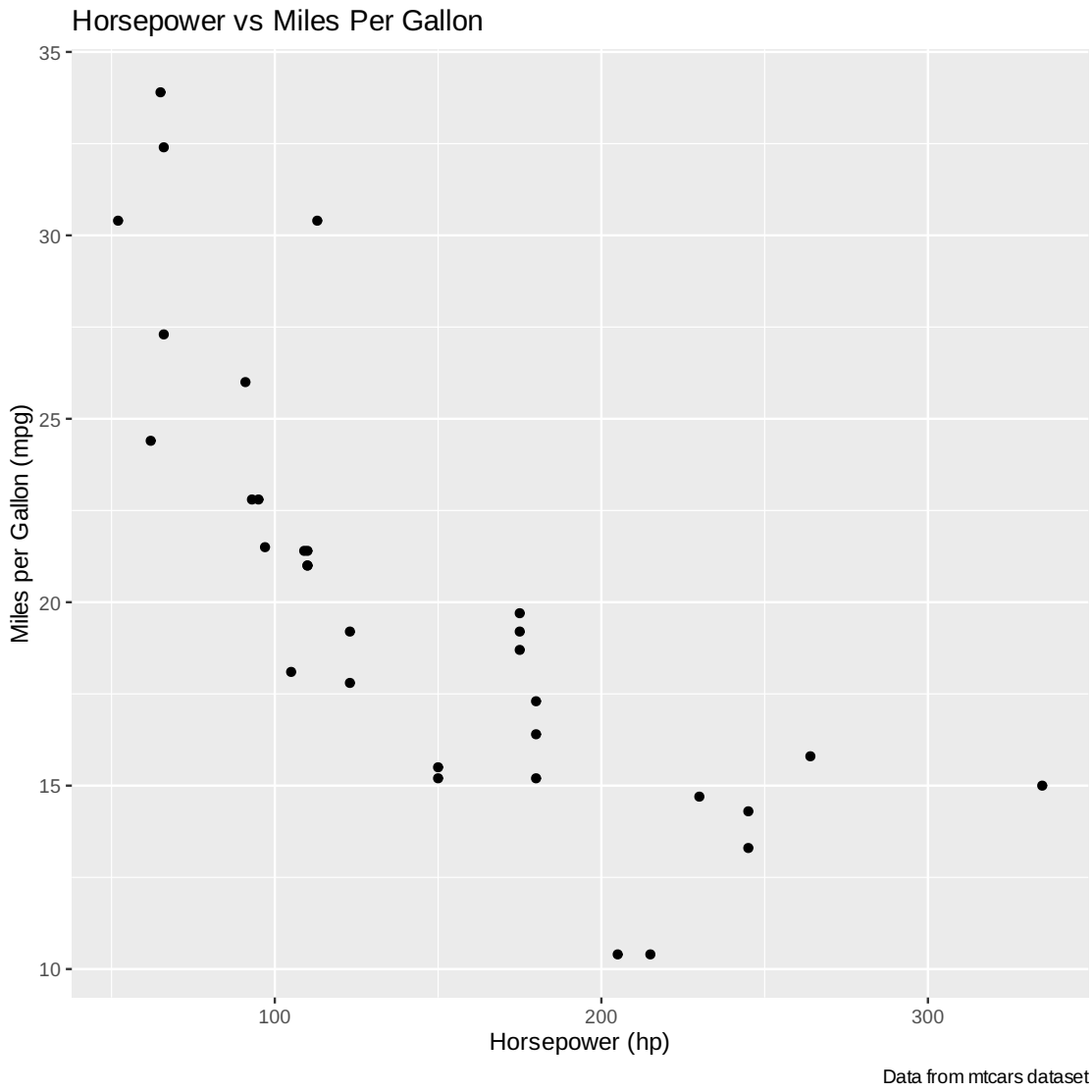


1.6.2 2. Adding Titles, Axis Labels, and Legends

You can add a title to your plot, as well as customize axis labels and legends for clarity:

[8]:
```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  labs(
    title = "Horsepower vs Miles Per Gallon",
    x = "Horsepower (hp)",  # Custom x-axis label
    y = "Miles per Gallon (mpg)",  # Custom y-axis label
    caption = "Data from mtcars dataset"  # Add a caption
```

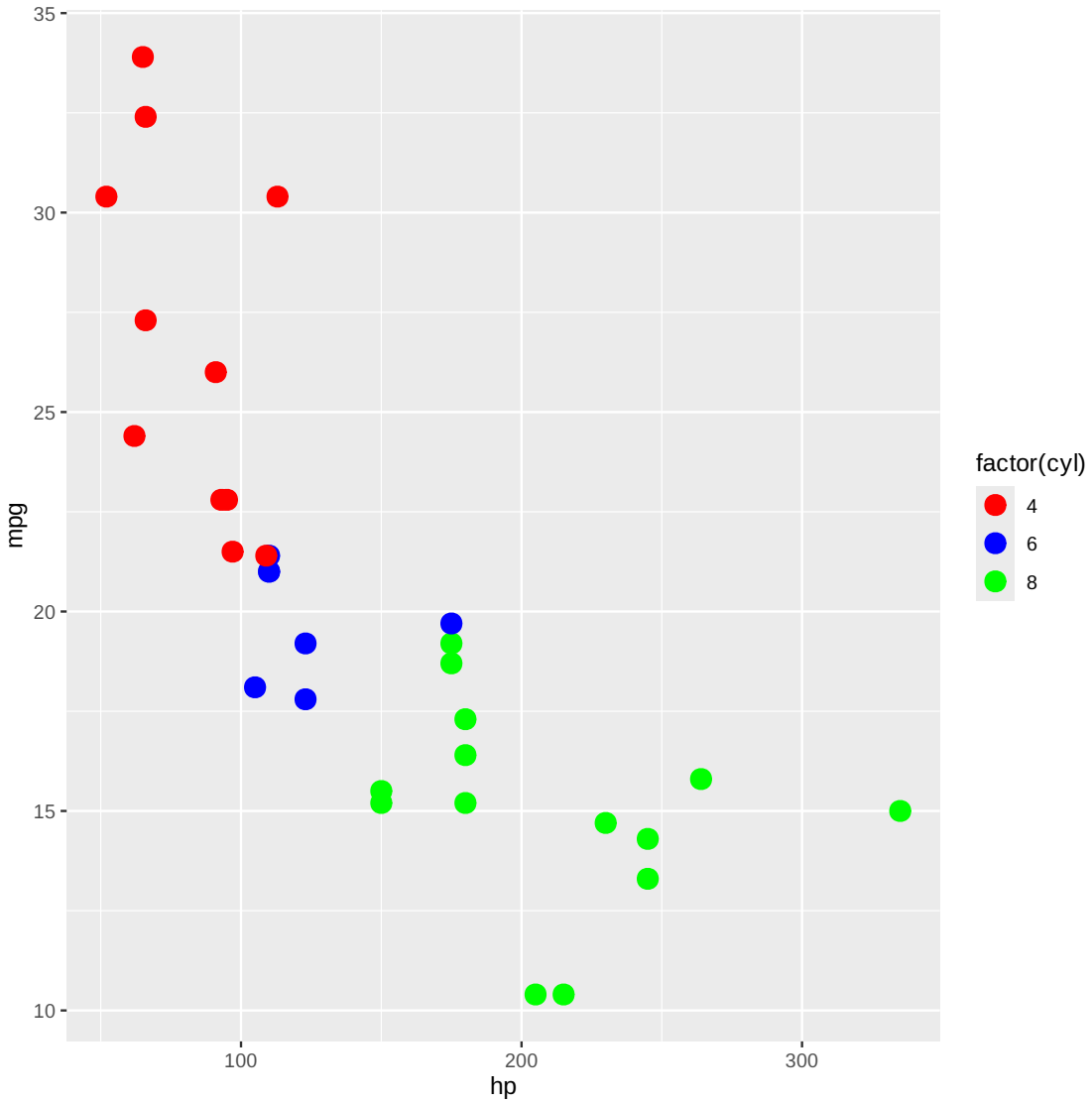
```
)
```



1.6.3 3. Changing Colors and Point Shapes

You can also customize the colors of points, lines, and other geometric objects:

[9]:
```
ggplot(mtcars, aes(x = hp, y = mpg, color = factor(cyl))) +  # Color by number␣
↪of cylinders
 geom_point(size = 4) +  # Change point size
 scale_color_manual(values = c("red", "blue", "green"))  # Manually set colors␣
↪for each group
```



1.7 Working with Multiple Geoms

Sometimes, we want to layer multiple types of plots to present different aspects of the data. This can be done by adding multiple geom_* functions together in a single plot.

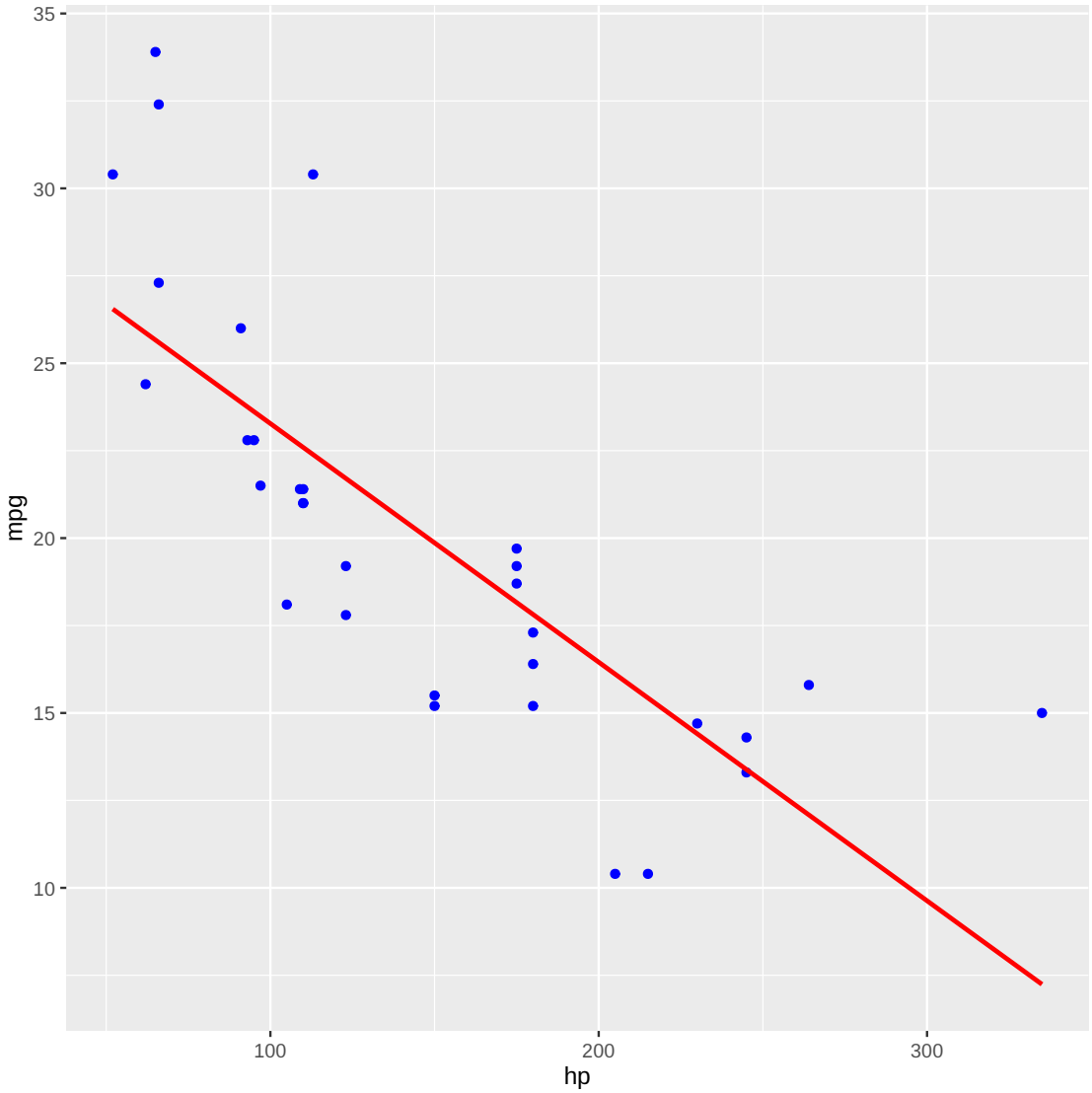
1.7.1 Example: Combining Points and a Line Plot

You can combine a scatter plot with a regression line to see the trend along with individual data points:

[10]:
```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(color = "blue") +  # Scatter plot
```

```
  geom_smooth(method = "lm", se = FALSE, color = "red")  # Add linear␣
↪regression line
```

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

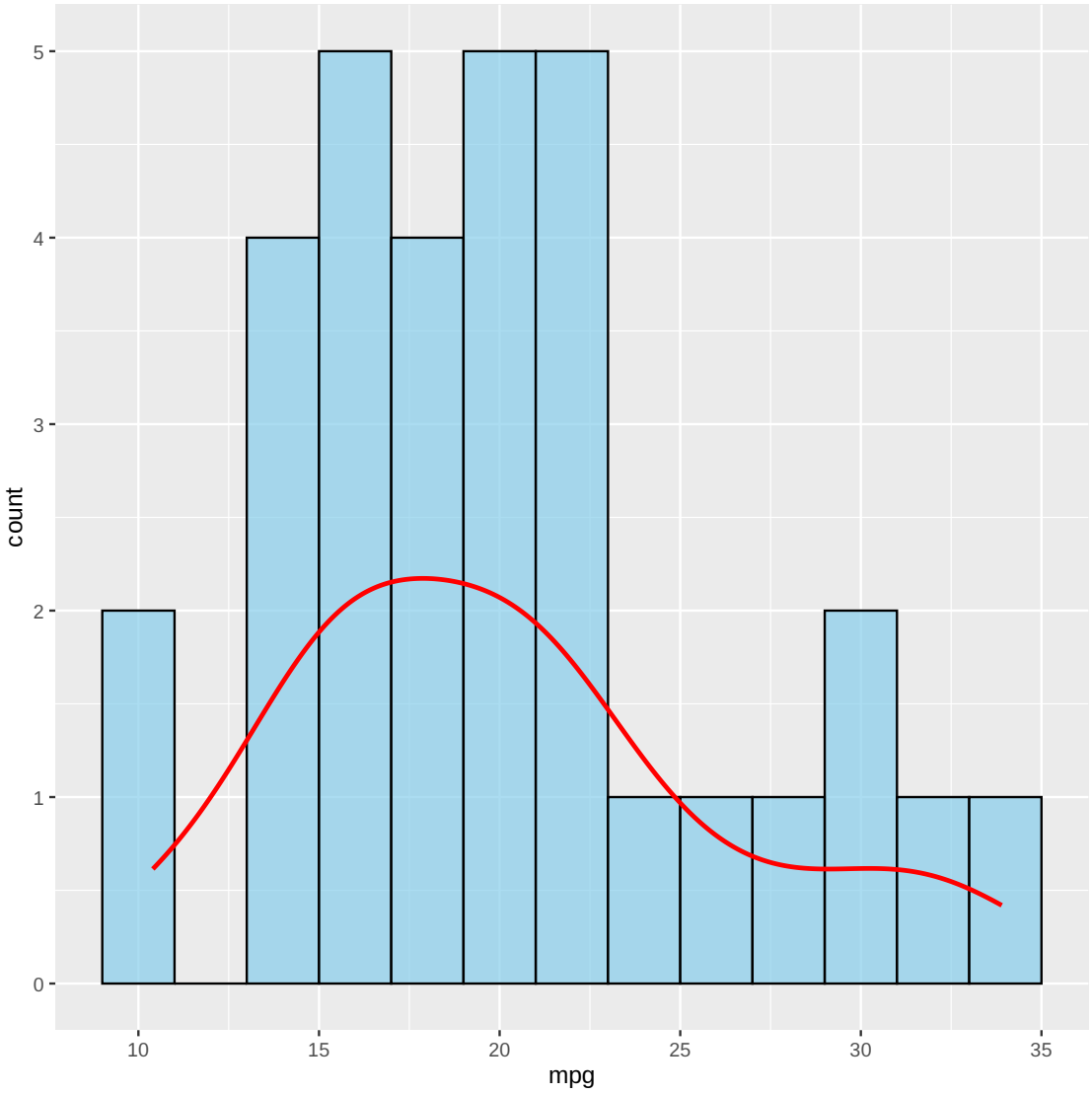


1.7.2 Example: Adding Histograms and Density Curves

You can also layer a histogram with a density curve to show the distribution of data:

[11]:
```
ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(binwidth = 2, fill = "skyblue", color = "black", alpha = 0.7)␣
↪+  # Histogram
  geom_density(aes(y = ..count..), color = "red", size = 1)  # Density curve
```

```
Warning message:
"Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
 Please use `linewidth` instead."
Warning message:
"The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
 Please use `after_stat(count)` instead."
```



1.8 Using Statistical Transformations

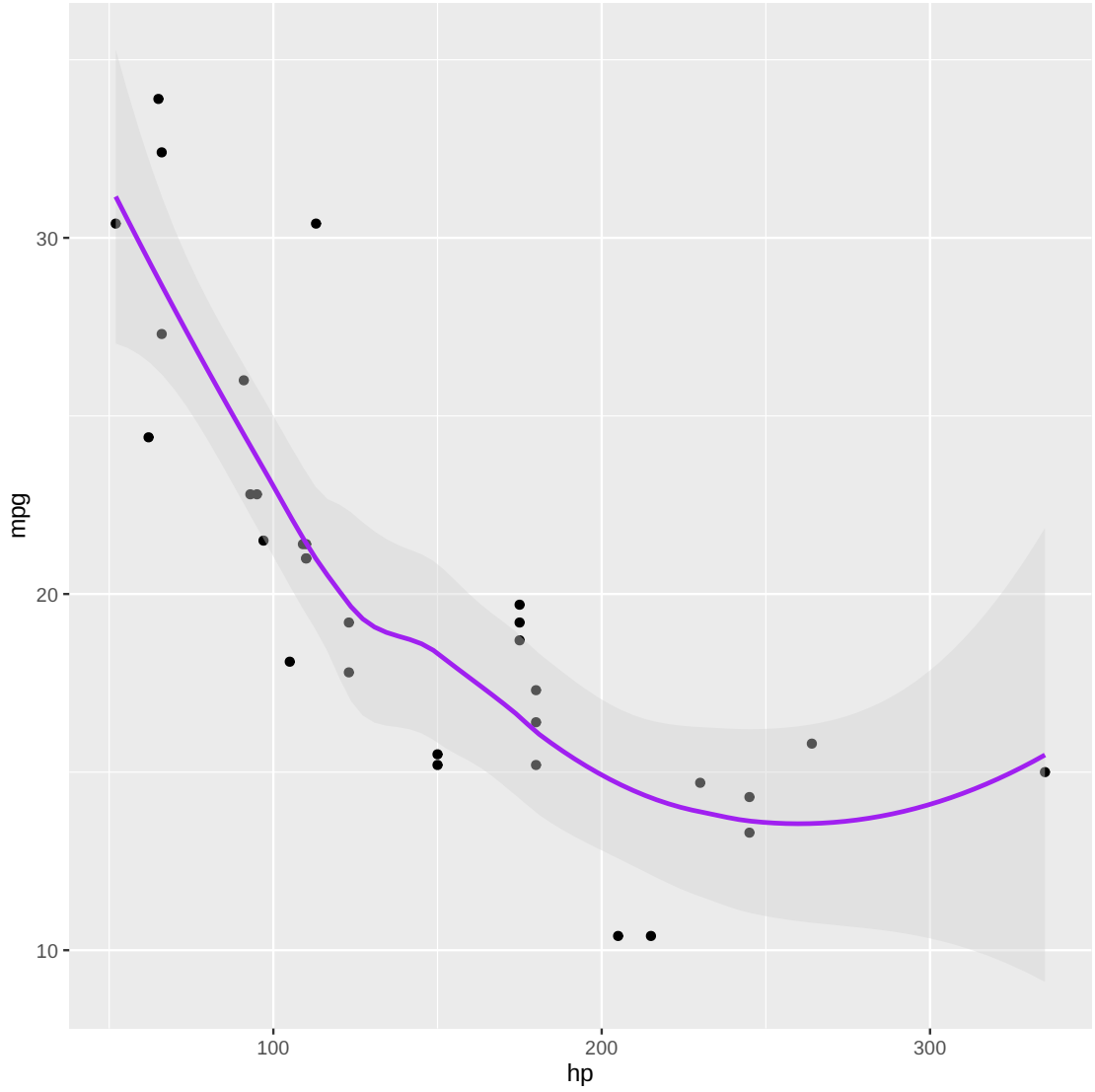
ggplot2 allows you to use built-in statistical transformations (`stat_*`) to quickly add statistical summaries to your plots. These transformations help you visualize trends, summaries, and distributions.

1.8.1 Example: Adding a Smooth Line (LOESS or Linear Model)

This code adds a smooth line to the plot using a local regression method (LOESS) or a linear model:

[12]:
```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +  # Scatter plot
  geom_smooth(method = "loess", color = "purple", fill = "lightgray")  # Add␣
↪LOESS line
```

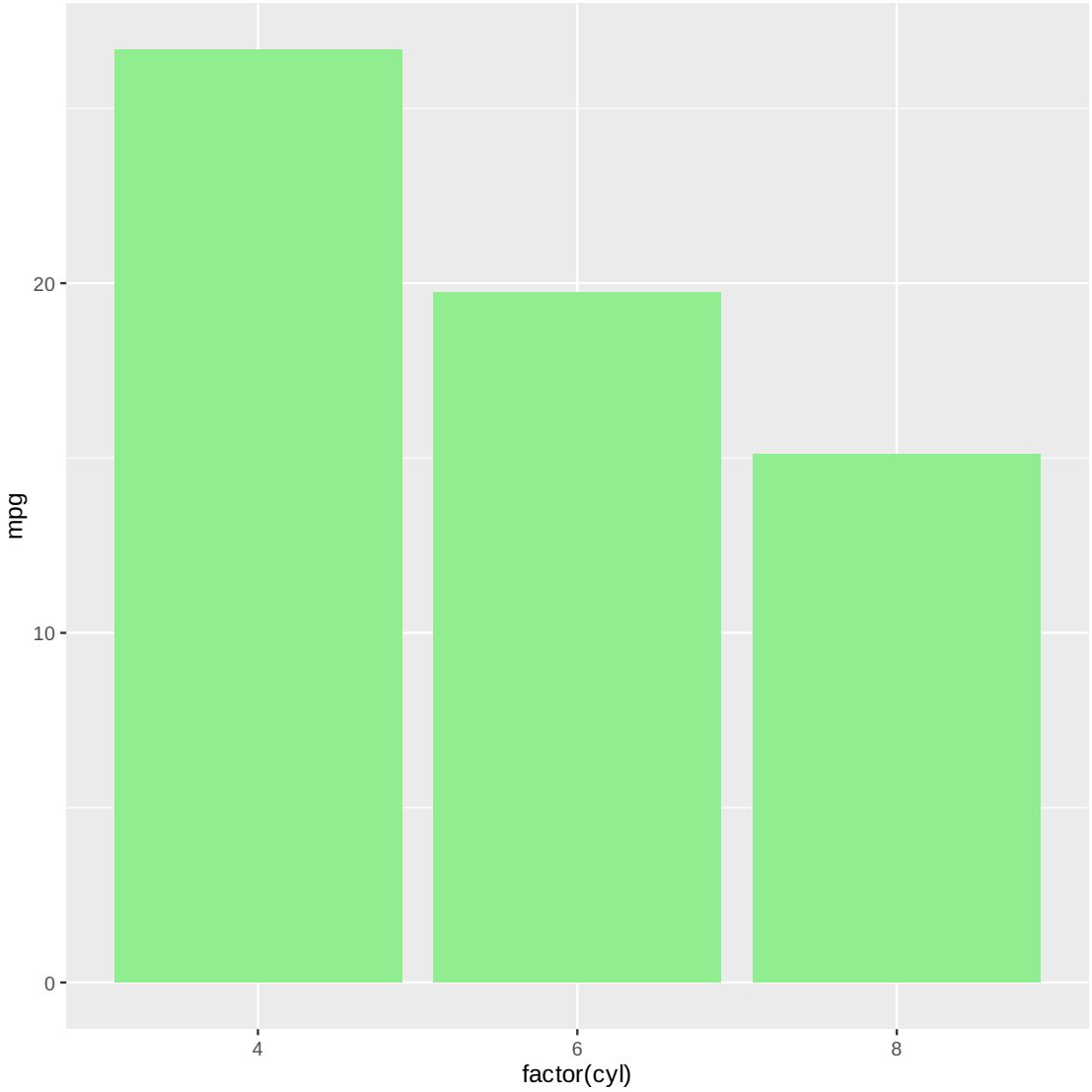
```
`geom_smooth()` using formula = 'y ~ x'
```



1.8.2 Example: Bar Plot with Summary Statistics

You can also use `stat_summary` to calculate and plot summary statistics (e.g., mean):

[13]:
```
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_bar(stat = "summary", fun = "mean", fill = "lightgreen")  # Bar plot␣
↪with mean
```



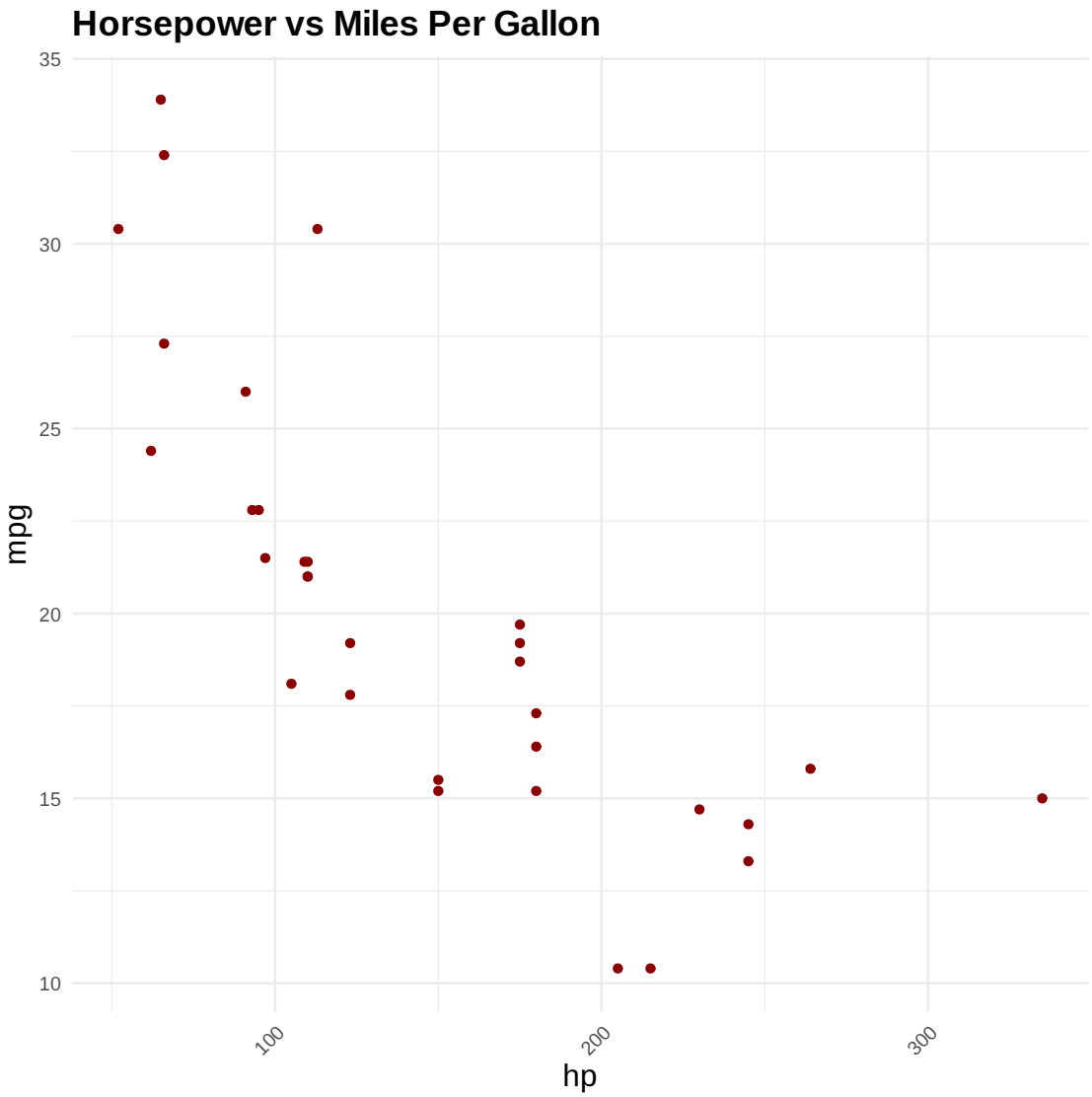
1.9 Themes and Styling for Polished Visualizations

Polishing your plots is an important step to make your visualizations more professional. ggplot2 provides several ways to adjust the appearance, including setting plot margins, text sizes, and axis styles.

1.9.1 Example: Adjusting Plot Theme

Here, we adjust the plot's background, text sizes, and axis angles for a polished, cleaner look:

[14]:
```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(color = "darkred") +
  theme_minimal() +  # Apply a clean minimal theme
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels
    plot.title = element_text(size = 16, face = "bold"),  # Customize title font
    axis.title = element_text(size = 14)  # Customize axis titles
  ) +
  labs(title = "Horsepower vs Miles Per Gallon")
```



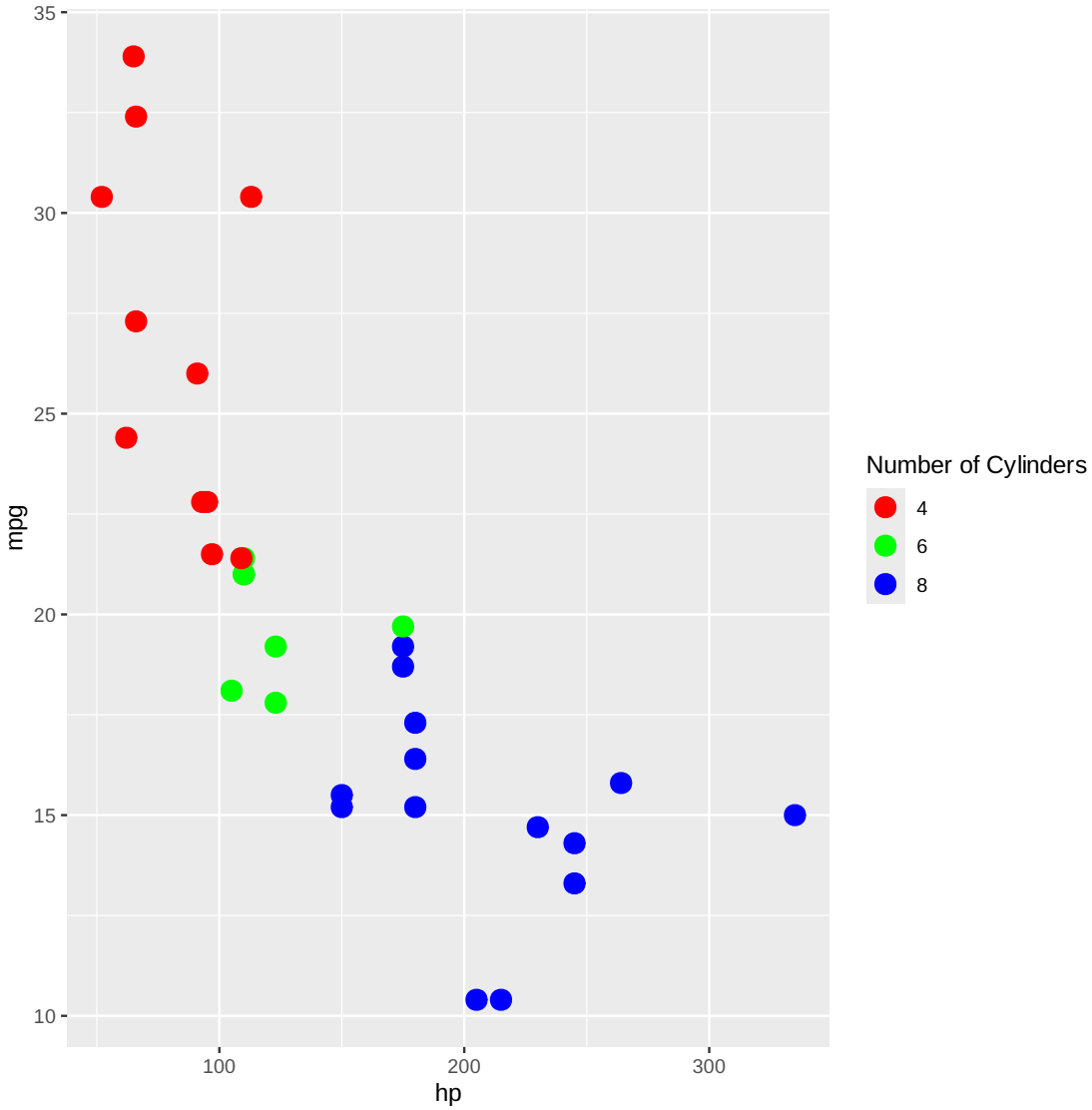
1.10 Customizing Legends and Color Scales

ggplot2 allows you to customize the legends and color scales used in the plot. You can set manual colors, adjust the legend title, or control its position.

1.10.1 Example: Customizing Legends

This code customizes the legend title and the color scale:

[15]:
```
ggplot(mtcars, aes(x = hp, y = mpg, color = factor(cyl))) +
  geom_point(size = 4) +
  scale_color_manual(values = c("red", "green", "blue")) +  # Customize color␣
↪scale
  labs(color = "Number of Cylinders")  # Rename the legend title
```



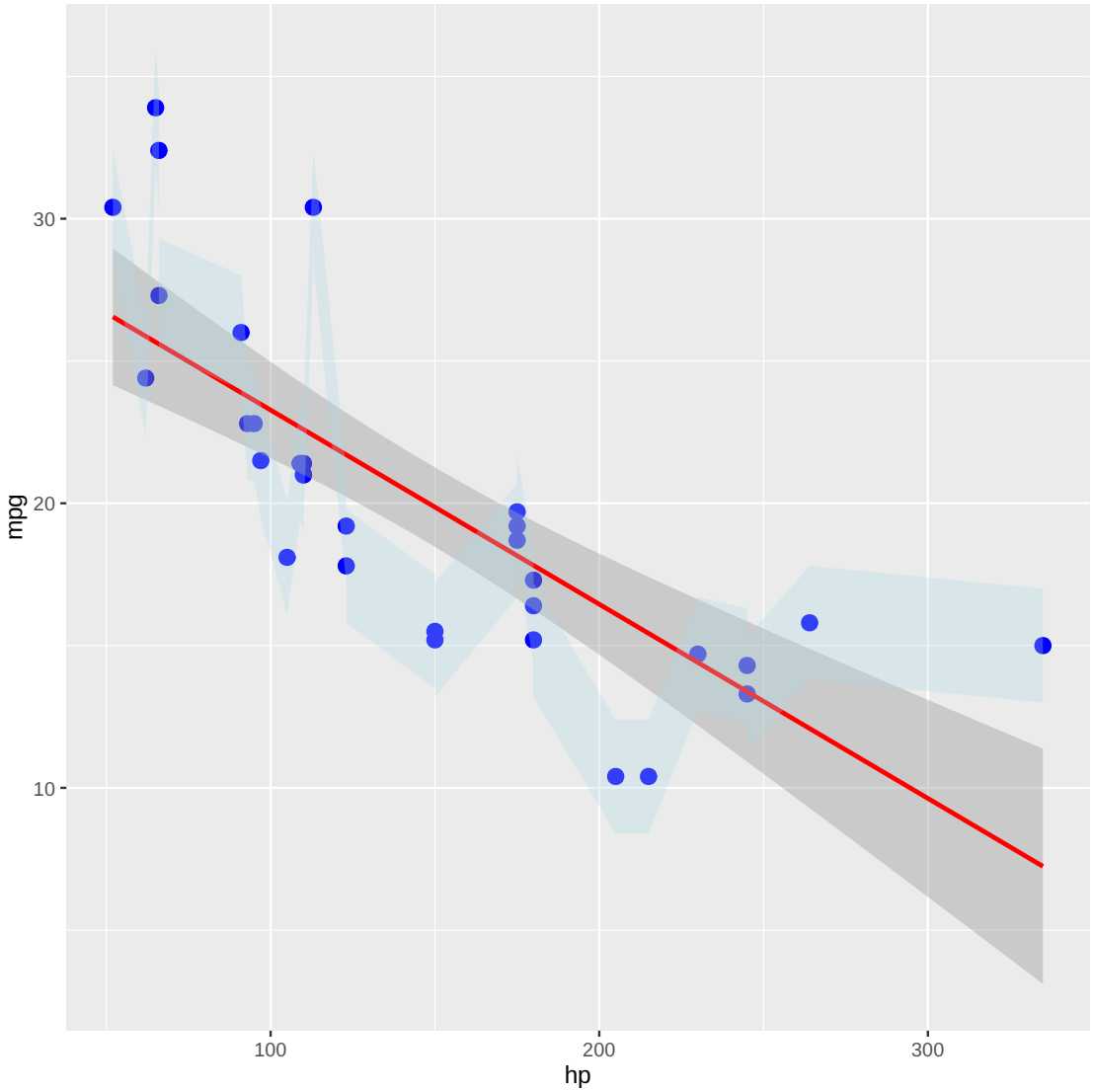
1.11 Multi-layered Plots with Complex Geoms

Advanced visualizations often require combining multiple geoms to communicate insights. This involves layering different plot types (e.g., scatter plots, lines, and ribbons) to create rich visualizations.

Example: Multi-layered Plot with `geom_point()`, `geom_smooth()`, and `geom_ribbon()`
This example demonstrates how to create a scatter plot, add a smooth regression line, and use a ribbon to show confidence intervals.

[16]:
```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(color = "blue", size = 3) +  # Scatter plot
  geom_smooth(method = "lm", color = "red", se = TRUE) +  # Linear regression␣
 ↪line with confidence interval
  geom_ribbon(aes(ymin = mpg - 2, ymax = mpg + 2), fill = "lightblue", alpha =␣
 ↪0.3)  # Confidence interval shaded area
#This plot shows both the scatter plot with a regression line and a ribbon to␣
 ↪indicate variability around the line.
```

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



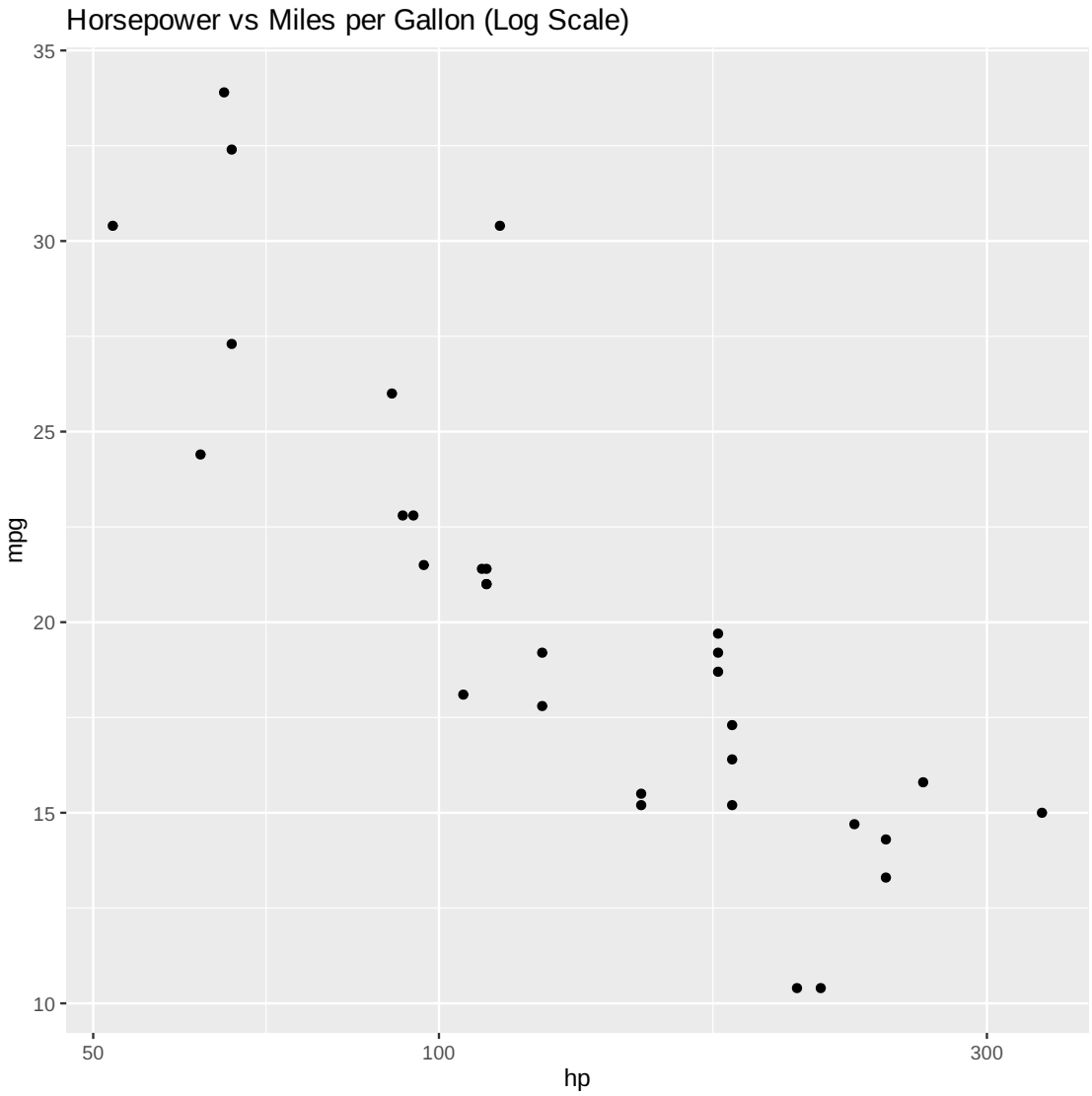
1.12 Customizing Axes and Scales

In advanced plots, it's important to control axis breaks, scales, and transformations to better represent your data.

Example: Logarithmic Scale and Custom Axis Ticks Logarithmic transformations are useful for visualizing data that spans multiple orders of magnitude, and custom axis ticks improve readability.

[17]:
```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  scale_x_log10() +  # Logarithmic transformation for x-axis
  scale_y_continuous(breaks = seq(10, 35, by = 5)) +  # Custom y-axis breaks
  labs(title = "Horsepower vs Miles per Gallon (Log Scale)")  # Custom title
```

```
#This plot transforms the x-axis to a logarithmic scale, which is helpful for␣
 ↪skewed data, and customizes the y-axis ticks.
```



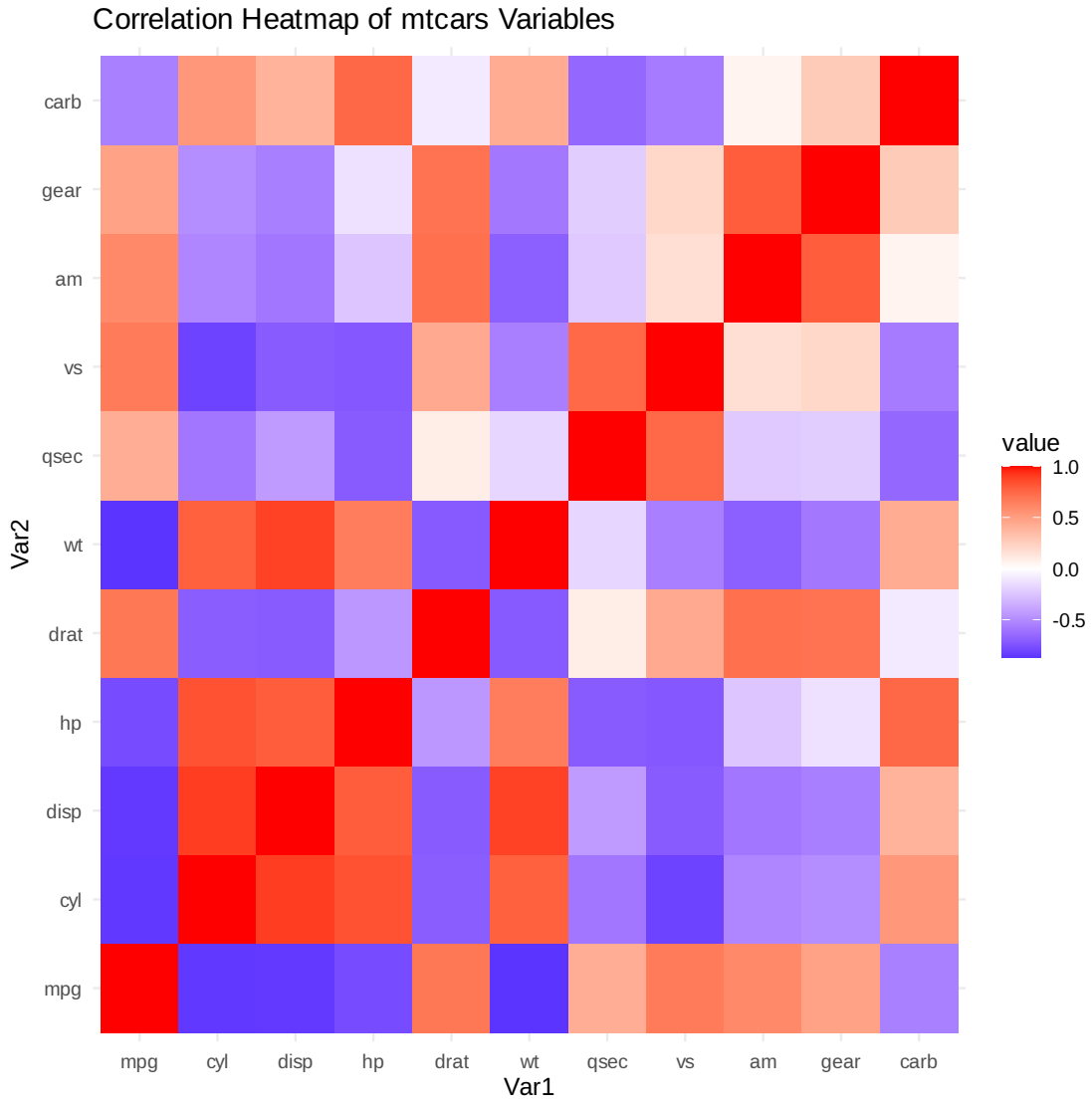
1.13 Creating Complex Heatmaps

Heatmaps are often used for visualizing correlations or complex relationships between two categorical variables.

Example: Correlation Heatmap This example demonstrates how to create a heatmap using `geom_tile()` to visualize correlations between different variables.

[18]:
```
library(reshape2)
```

```r
# Calculate correlation matrix
cor_matrix <- cor(mtcars)

# Convert the matrix to long format
cor_melted <- melt(cor_matrix)

# Create the heatmap
ggplot(cor_melted, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0) +  # Custom color scale
  theme_minimal() +  # Clean theme
  labs(title = "Correlation Heatmap of mtcars Variables")  # Add a title
```



This heatmap visualizes the correlation matrix between different variables in the `mtcars` dataset, with colors representing the strength and direction of the correlation.

1.14 Interactive Visualizations with `plotly`

ggplot2 is often combined with `plotly` to create interactive plots. This allows users to zoom, hover, and explore data in real time.

Example: Interactive ggplot2 Plot with `plotly`

[19]:
```
install.packages("plotly")
```

```
also installing the dependencies ‘lazyeval’, ‘crosstalk’


Updating HTML index of packages in '.Library'

Making 'packages.html' …
 done
```

[20]:
```
library(plotly)

# Create a ggplot object
p <- ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(color = factor(cyl)), size = 4) +
  labs(title = "Horsepower vs Miles per Gallon")

# Convert ggplot object to interactive plotly plot
ggplotly(p)
```

```
Attaching package: ‘plotly’


The following object is masked from ‘package:ggplot2’:

    last_plot


The following object is masked from ‘package:stats’:

    filter


The following object is masked from ‘package:graphics’:

    layout
```

```
HTML widgets cannot be represented in plain text (need html)
```

This creates an interactive plot where users can hover over points to see details and zoom in/out.

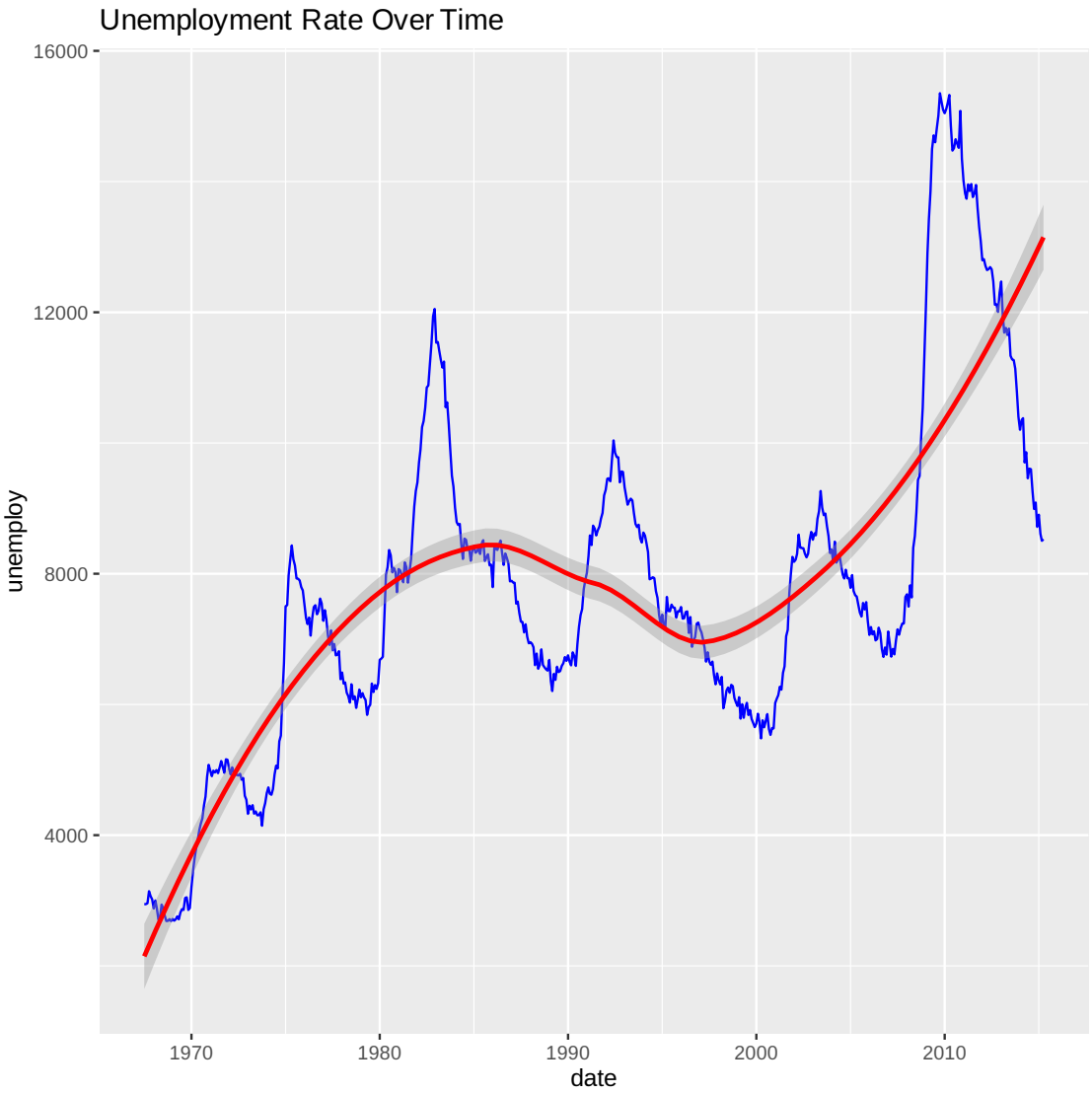
1.15 Advanced Time Series Visualization

Visualizing time series data requires using specialized techniques, such as smoothing or seasonal decomposition.

Example: Time Series Plot with Smoothing This plot uses `geom_smooth()` to display a smoothed time series trend, useful when working with fluctuating data over time.

[21]:
```
# Create a time series plot using a date variable
ggplot(economics, aes(x = date, y = unemploy)) +
  geom_line(color = "blue") +  # Line plot
  geom_smooth(method = "loess", color = "red", size = 1) +  # LOESS smoothing␣
 ↪line
  labs(title = "Unemployment Rate Over Time")
```

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



This visualization uses a smoothed curve to represent the trend in unemployment over time, making the underlying patterns more visible.

1.16 Ridgeline Plots

Ridgeline plots are great for visualizing the distribution of data across different groups, especially when you want to show the overlap between multiple distributions.

Example: Ridgeline Plot of mpg by Cylinders

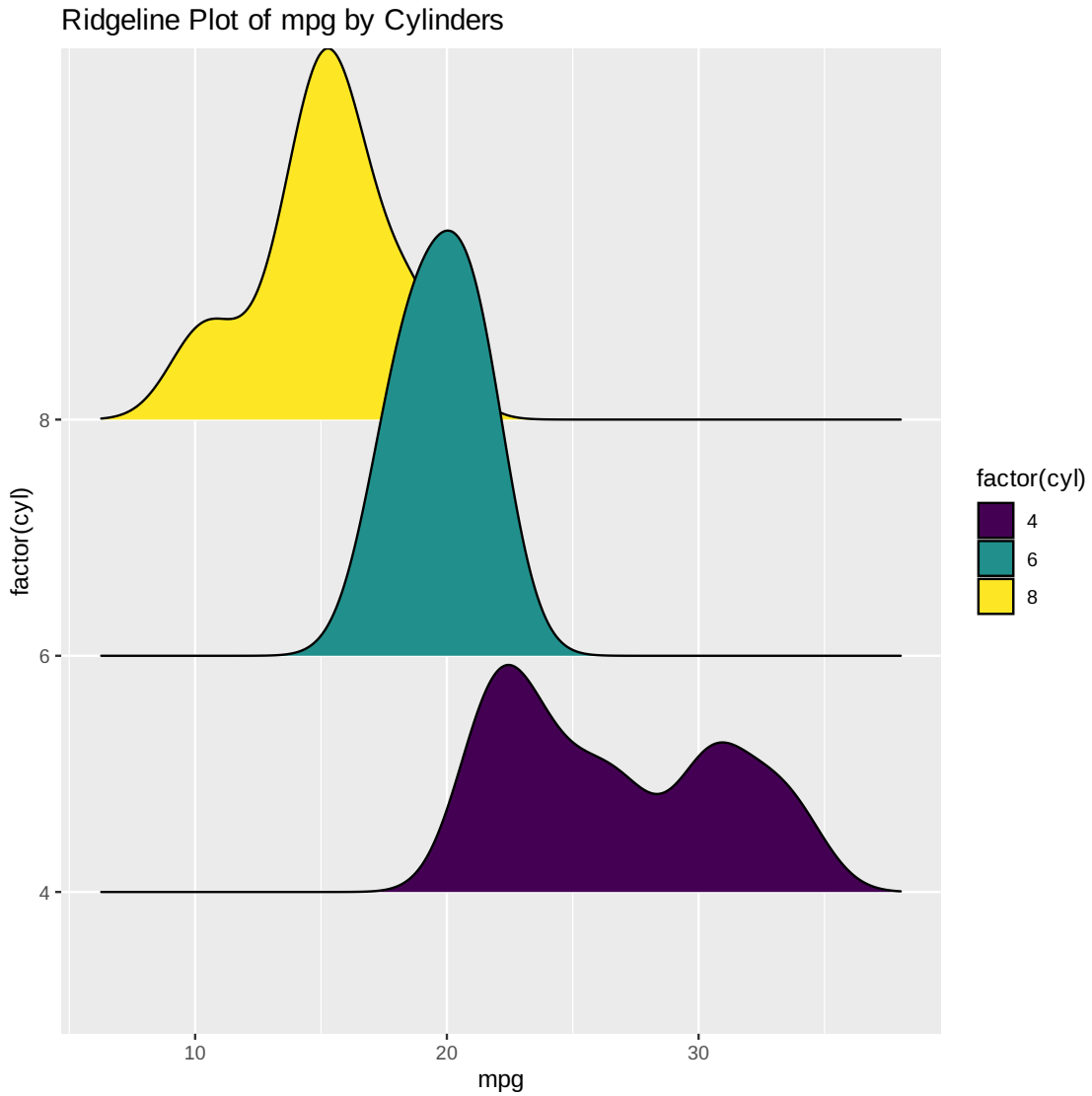
[22]:
```
install.packages("ggridges")
```

```
Updating HTML index of packages in '.Library'
```

```
Making 'packages.html' …
 done
```

[23]:
```
library(ggridges)

# Create a ridgeline plot
ggplot(mtcars, aes(x = mpg, y = factor(cyl), fill = factor(cyl))) +
  geom_density_ridges() +
  scale_fill_viridis_d() +  # Color scale
  labs(title = "Ridgeline Plot of mpg by Cylinders")
```

```
Picking joint bandwidth of 1.38
```



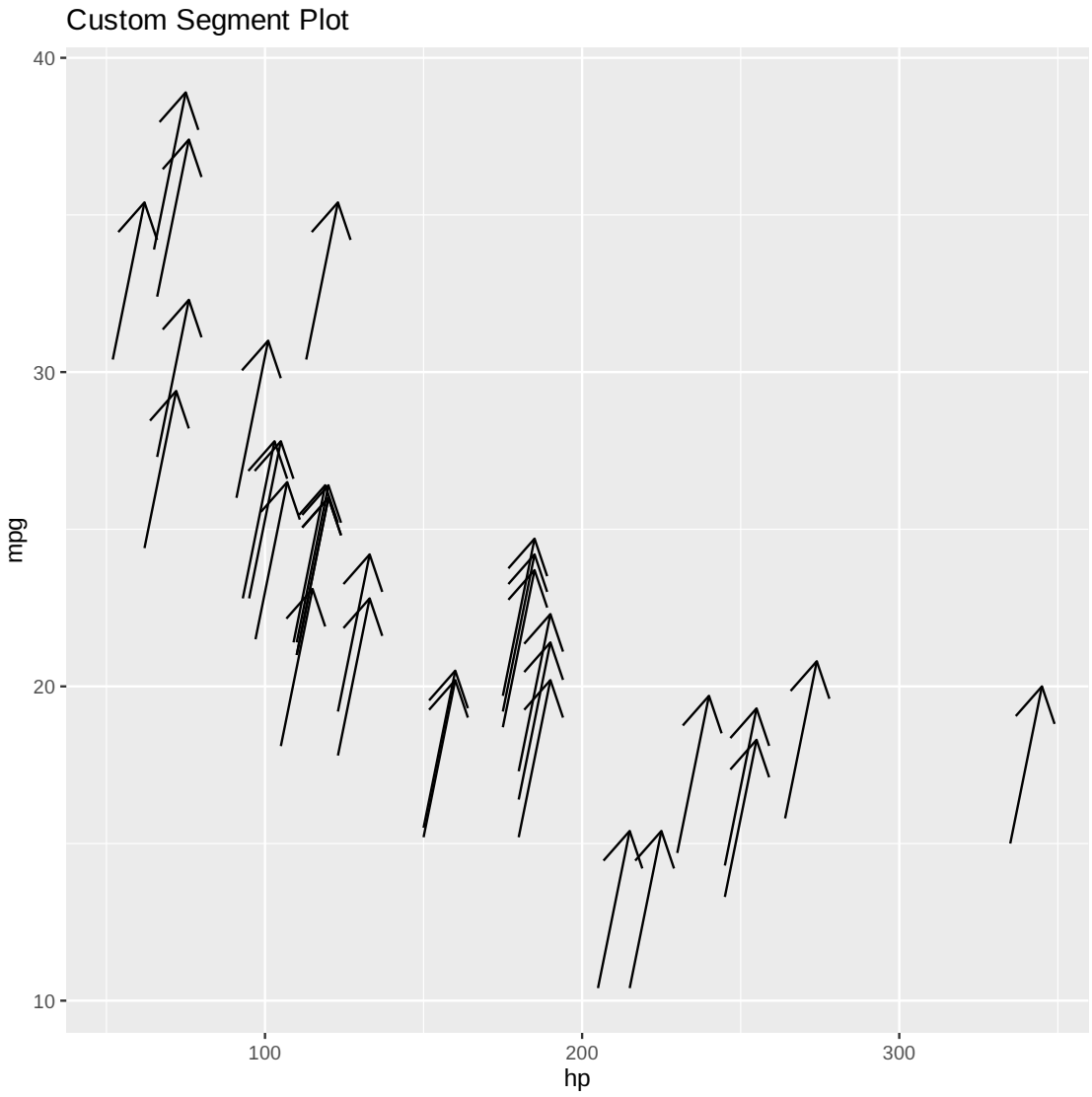
This ridgeline plot shows the distribution of `mpg` for each cylinder group in `mtcars`, with overlapping density curves.

1.17 Custom Geoms

You can also create your own custom ggplot geoms by using the `geom_raster()` or `geom_segment()` functions for more specialized plotting.

Example: Customizing with `geom_segment()`

[24]:
```
# Create custom segment plot
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_segment(aes(xend = hp + 10, yend = mpg + 5), arrow = arrow(type =␣
 ↪"open")) +  # Custom segments with arrows
  labs(title = "Custom Segment Plot")
```



This plot uses custom segments with arrows to represent changes or relationships between the `hp` and `mpg` variables.

[ ]: