Bivariate Continuous data (Part 4 of 4)

Chapter 15.

Exploring bivariate Continuous x Continuous data, using `ggplot2`

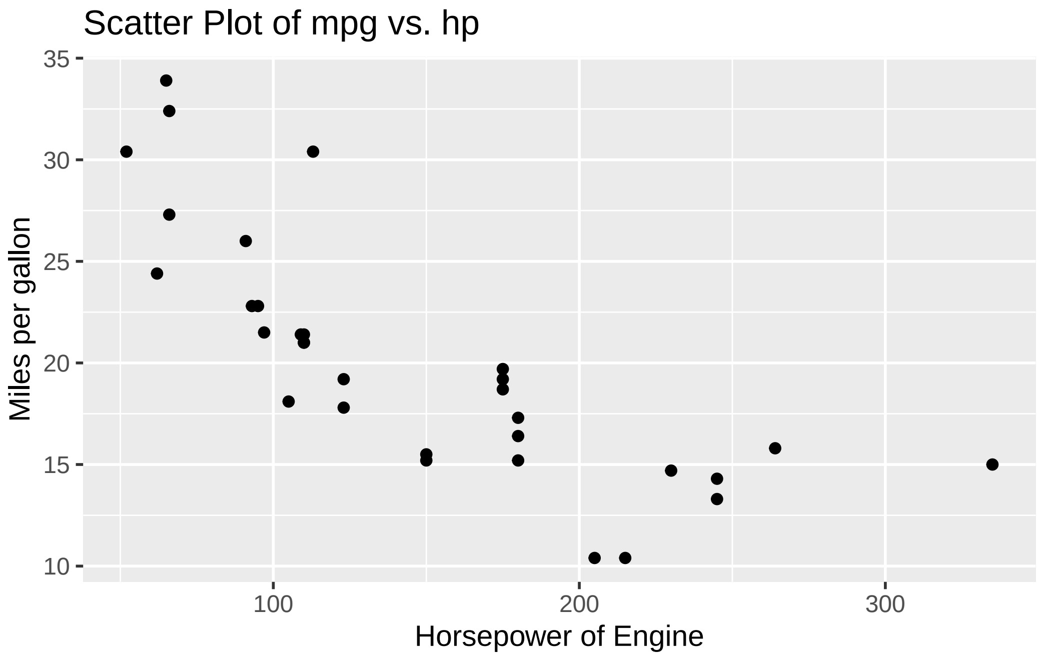
This chapter demonstrates the use of the popular **`ggplot2`** and **`ggpubr`** packages to further explore the interaction between *bivariate continuous data.*

Data: Suppose we run the following code to prepare the `mtcars` data for subsequent analysis and save it in a tibble called `tb`. [1]

```
# Load the required libraries, suppressing annoying startup messages
library(dplyr, quietly = TRUE, warn.conflicts = FALSE)
library(tibble, quietly = TRUE, warn.conflicts = FALSE)
library(ggplot2, quietly = TRUE, warn.conflicts = FALSE) # For data visualization
library(ggpubr, quietly = TRUE, warn.conflicts = FALSE) # For data visualization

# Read the mtcars dataset into a tibble called tb
data(mtcars)
tb <- as_tibble(mtcars)

# Convert relevant columns into factor variables
tb$cyl <- as.factor(tb$cyl) # cyl = {4,6,8}, number of cylinders
tb$am <- as.factor(tb$am) # am = {0,1}, 0:automatic, 1: manual transmission
tb$vs <- as.factor(tb$vs) # vs = {0,1}, v-shaped engine, 0:no, 1:yes
tb$gear <- as.factor(tb$gear) # gear = {3,4,5}, number of gears
```

Scatterplot using `ggplot2`

```
# Create a scatter plot using ggplot2 library  [2]
ggplot(data = tb,            # Specify the data frame 'tb' as the data source
       aes(x = hp,
           y = mpg)) +  # Define aesthetics: x-axis as 'hp' and y-axis as 'mpg'
  geom_point() +             # Add points to the plot
  xlab("Horsepower of Engine") +  # Label the x-axis
  ylab("Miles per gallon") +     # Label the y-axis
  ggtitle("Scatter Plot of mpg vs. hp")  # Set the title of the plot
```

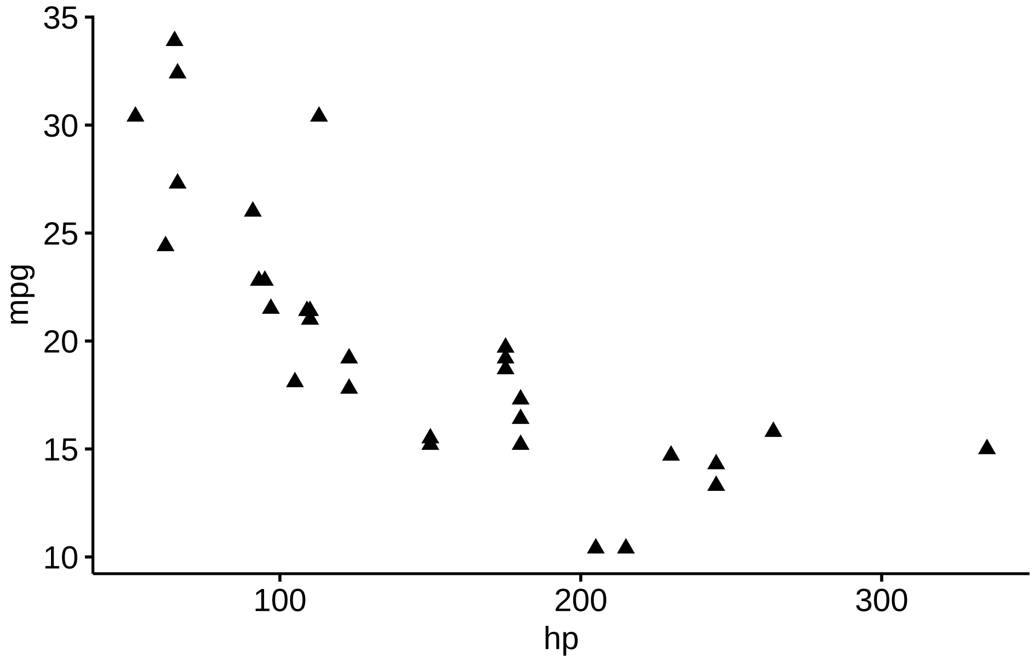


Discussion:

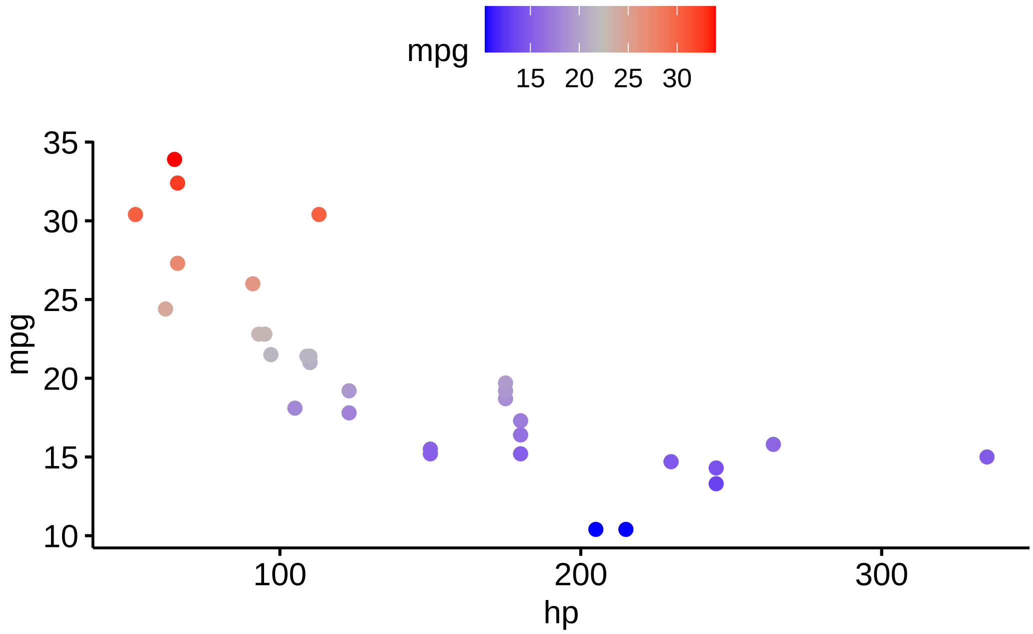
- The `ggplot2` package uses a layering approach, enabling users to build plots incrementally, piece by piece, using a combination of data, aesthetics, and geometric objects.
- The function `ggplot()` initializes the plotting system. It requires a dataset to operate on and an aesthetic mapping to determine how data variables will be plotted. Here, the dataset is represented by `tb`.
- Inside the `aes()` function, which stands for aesthetics, the code specifies that the variable `hp` from the `tb` data frame will be plotted on the x-axis and the variable `mpg` will be plotted on the y-axis. Hence, the resulting plot will display a relationship between horsepower (`hp`) and miles per gallon (`mpg`).
- The `geom_point()` function is an added layer, instructing `ggplot2` to render the relationship between `hp` and `mpg` as a scatter plot, with individual data points being represented as points.

- The functions `xlab()` and `ylab()` are used to set custom labels for the x and y axes, respectively. In this code, the x-axis is labeled as "Horsepower of Engine" and the y-axis is labeled as "Miles per gallon".
- Finally, the `ggtitle()` function is used to assign a title to the entire plot. In this instance, the title is set as "Scatter Plot of mpg vs. hp", clearly indicating the purpose and content of the visualization. [2]

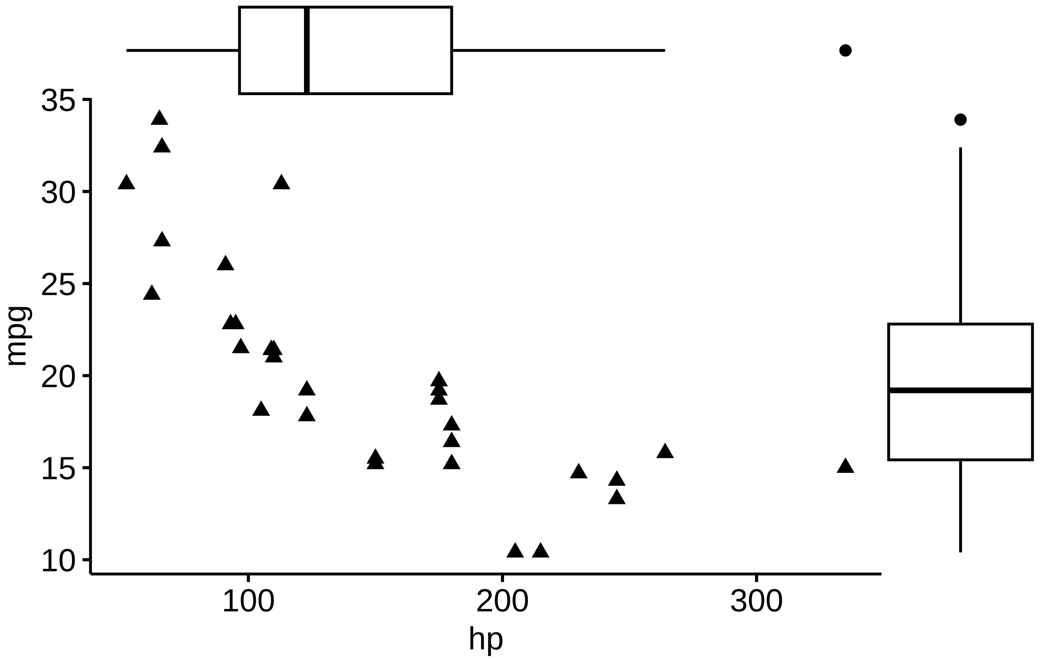
```
# Load the 'ggpubr' library, which contains the 'ggscatter()' function   [3]
library(ggpubr)

# Create a scatter plot using the 'ggscatter()' function
ggscatter(data = tb,        # Specify the data frame 'tb' as the data source
          x = "hp",         # Define the x-axis variable as "hp"
          y = "mpg",        # Define the y-axis variable as "mpg"
          shape = 17)       # Set the point shape to 17 (a filled circle)
```



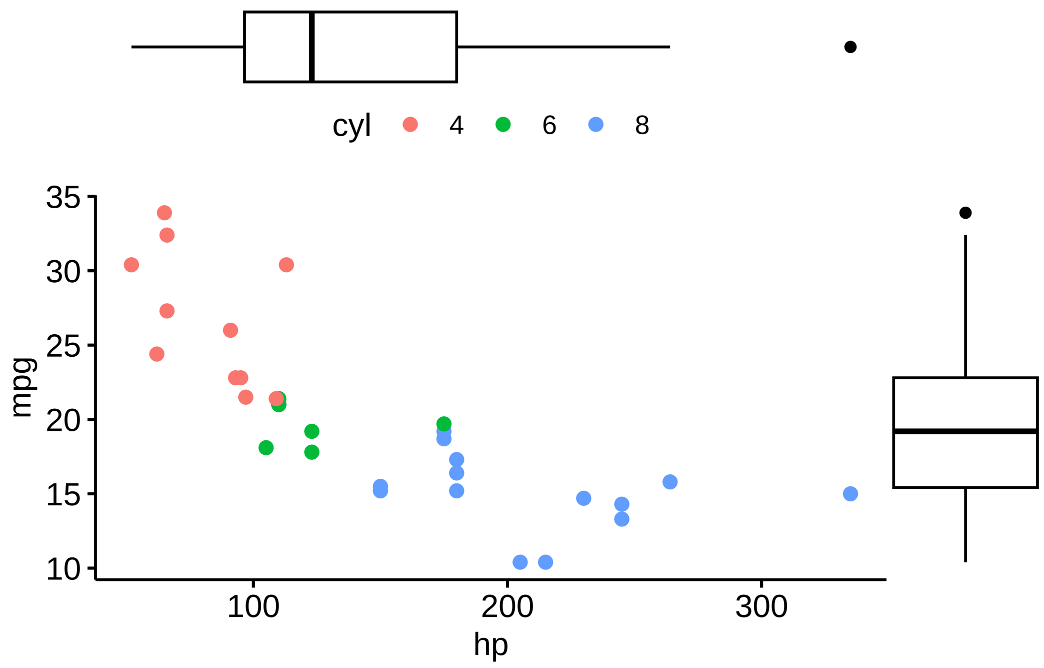
```
# Create a scatter plot using the 'ggscatter()' function from the 'ggpubr' package  [3]
# Set the x-axis variable to "hp" and the y-axis variable to "mpg"
# Additionally, specify the color aesthetic to "mpg" (color points based on "mpg")
ggscatter(data = tb, x = "hp", y = "mpg", color = "mpg") +
  gradient_color(c("blue", "gray", "red")) # Add a gradient color scheme [3]
```



```
# Load the 'ggExtra' library, which contains the 'ggMarginal()' function
library("ggExtra")

# Create a scatter plot using the 'ggscatter()' function from 'ggpubr'
p <- ggscatter(data = tb, x = "hp", y = "mpg", shape = 17)

# Change the marginal plot type to a boxplot using 'ggMarginal()'
ggMarginal(p, type = "boxplot")
```

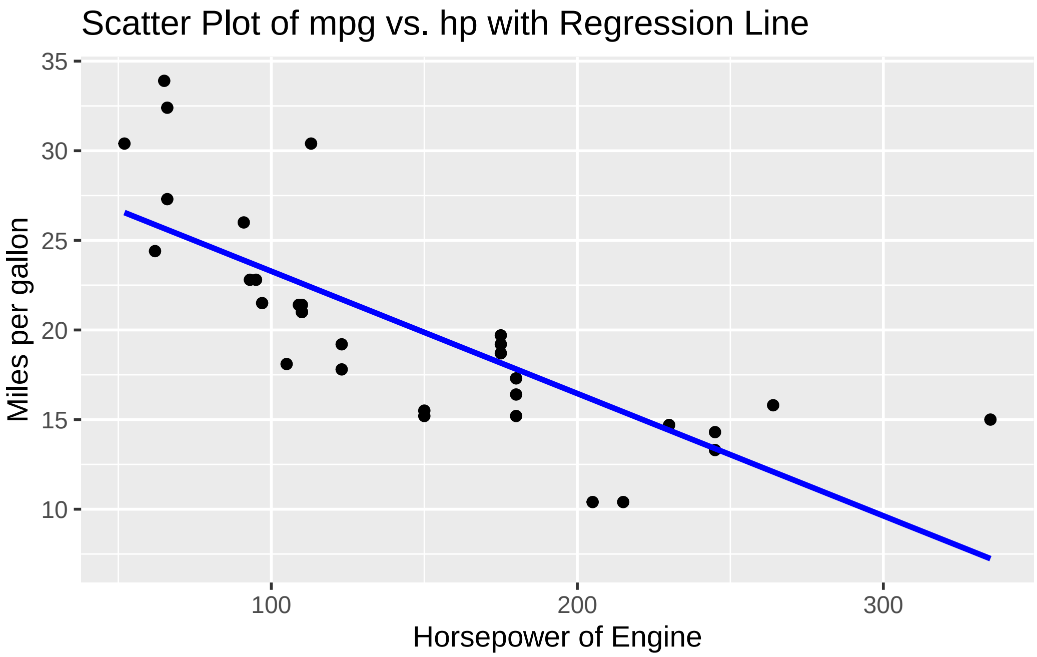


```
# Load the 'ggExtra' library, which contains the 'ggMarginal()' function
library("ggExtra")

# Create a scatter plot using the 'ggscatter()' function from the 'ggpubr' package
# Set the x-axis variable to "hp", the y-axis variable to "mpg", and color points based on
p <- ggscatter(data = tb, x = "hp", y = "mpg", color = "cyl")

# Change the marginal plot type to a boxplot using 'ggMarginal()'
ggMarginal(p, type = "boxplot")
```

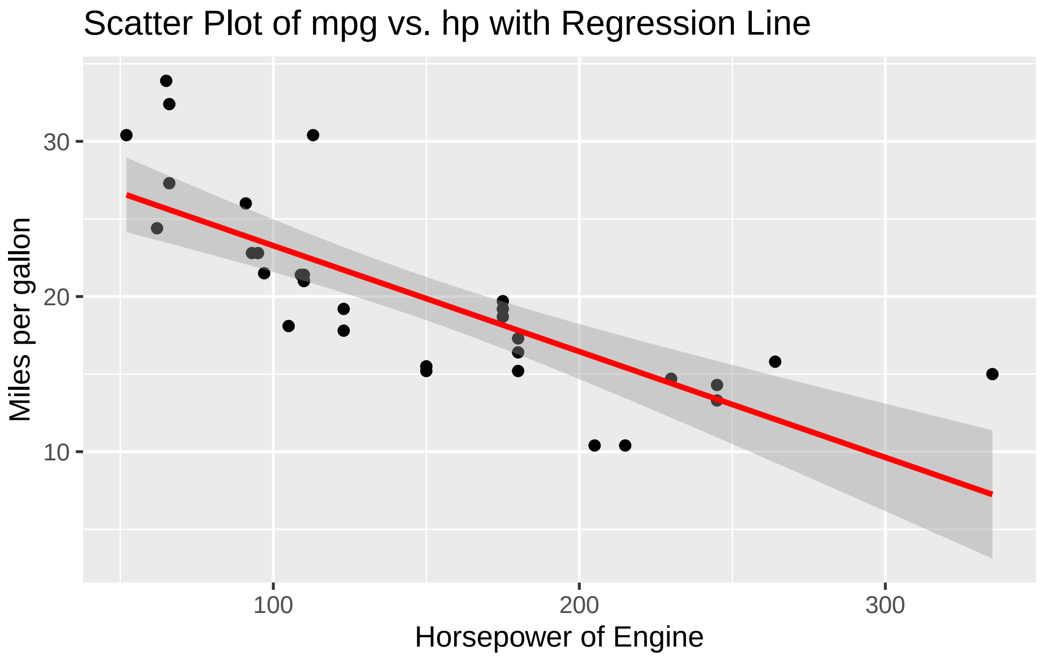


Scatterplot with Regression line using `ggplot2`

```
# Create a scatter plot using the 'ggplot()' function [2], [5]
ggplot(data = tb, aes(x = hp, y = mpg)) +
  geom_point() + # Add points to the plot
  geom_smooth(method = "lm",      # Add a linear regression line
              se = FALSE,         # Disable confidence intervals
              color = "blue") +   # set line color to blue
  xlab("Horsepower of Engine") + # Label the x-axis
  ylab("Miles per gallon") +      # Label the y-axis
  ggtitle("Scatter Plot of mpg vs. hp with Regression Line")
```



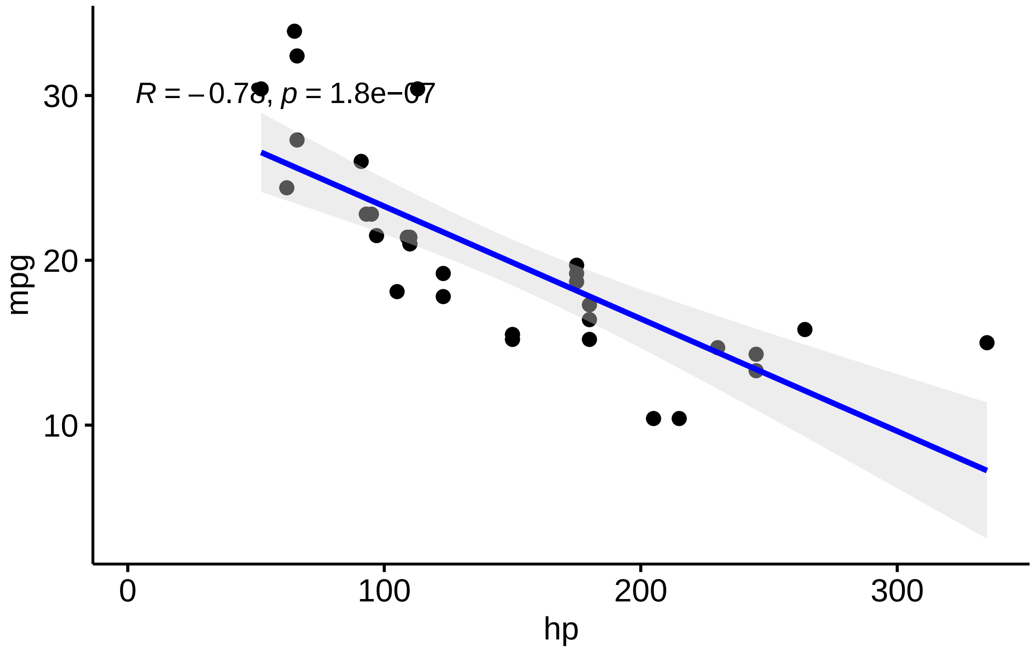
```r
ggplot(tb, aes(x = hp, y = mpg)) +
  geom_point() +                      # Add points to the plot  [2], [5]
  geom_smooth(method = "lm",          # Add a linear regression line
              se = TRUE,              # Enable confidence intervals
              color = "red") +        # Set line color to red
  xlab("Horsepower of Engine") +     # Label the x-axis
  ylab("Miles per gallon") +         # Label the y-axis
  ggtitle("Scatter Plot of mpg vs. hp with Regression Line")
```



Discussion:

- `geom_smooth(method = "lm", se = FALSE, color = "blue")`: This function adds a smoothed conditional mean.
    - The `method = "lm"` argument indicates that a linear model (i.e., a regression line) should be used for smoothing. This line will depict the overall trend in the data.
    - If `se = FALSE` then the standard error bands (which show the uncertainty around the regression line) aren't plotted. This determines whether or not the standard error bands (or confidence interval bands) are displayed around the smoothing line. In the case of linear regression (method = "lm"), these bands represent the 95% confidence interval around the predicted values. This means that if you were to repeatedly sample from the population and fit a regression model each time, you'd expect about 95% of the confidence intervals to contain the true regression line.

```
# Create a scatter plot using the 'ggscatter()' function from the 'ggpubr' package  [3], [
# Set the x-axis variable to "hp" and the y-axis variable to "mpg"
ggscatter(tb,
          x = "hp",
          y = "mpg",
          add = "reg.line",         # Add regression line
          conf.int = TRUE,          # Include confidence interval
          add.params = list(color = "blue",  # Customize color and fill
                            fill = "lightgray")) +

# Add the Pearson correlation coefficient to the plot
# Specify the label coordinates for the correlation coefficient
stat_cor(method = "pearson", label.x = 3, label.y = 30)
```

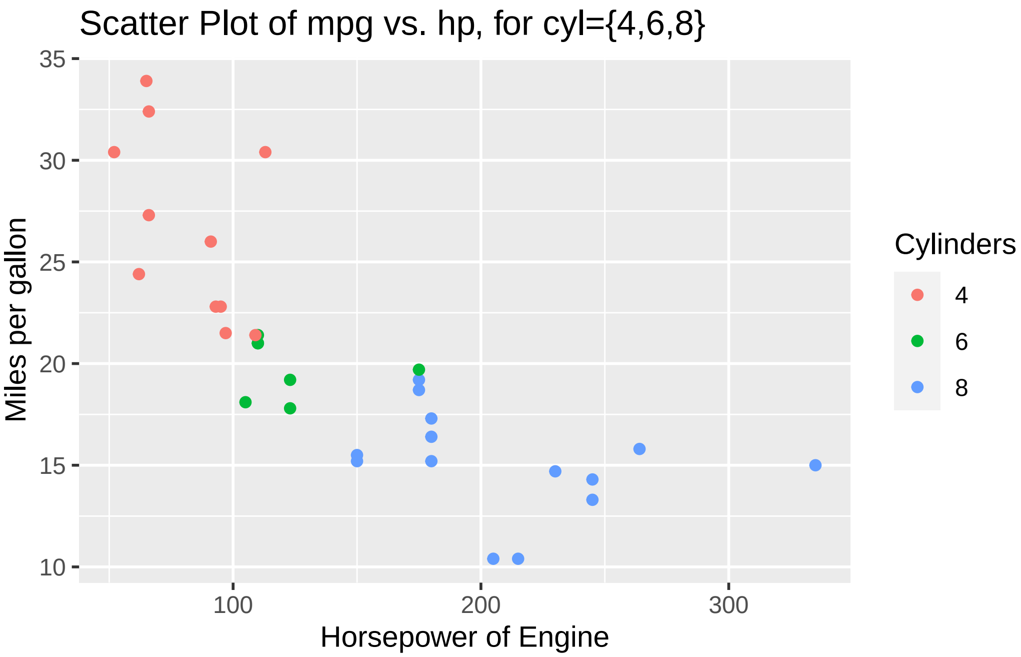


Scatterplots with Categorical Variables

Scatterplot colored by a Categorical variable, using `ggplot()`

This will create a scatterplot of miles per gallon (`mpg`) against horsepower (`hp`), with each point colored according to the number of cylinders (`cyl`) in the engine.

```
# Create a scatter plot using the 'ggplot()' function from the 'ggplot2' library
# Set the x-axis variable to "hp", the y-axis variable to "mpg", and color points by "cyl"
ggplot(tb, aes(x = hp,
               y = mpg,
               color = factor(cyl))) +
# Add points to the plot
geom_point() +
labs(x = "Horsepower of Engine",
     y = "Miles per gallon") + # Label x-axis and y-axis
scale_color_discrete(name = "Cylinders")  +  # Customize the legend title
ggtitle("Scatter Plot of mpg vs. hp, for cyl={4,6,8}") # Set the title
```

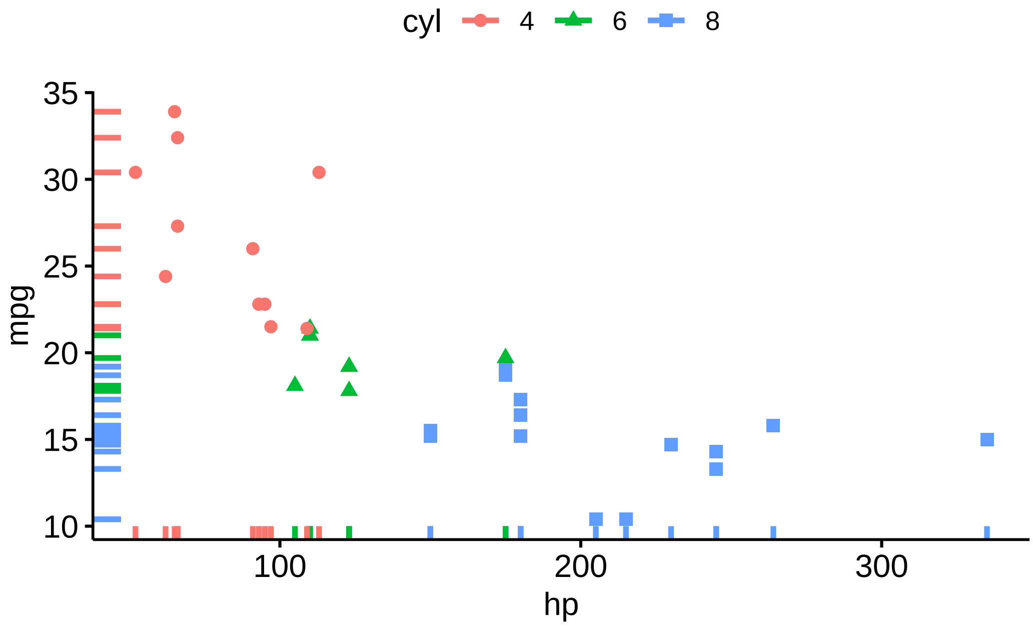


Discussion:

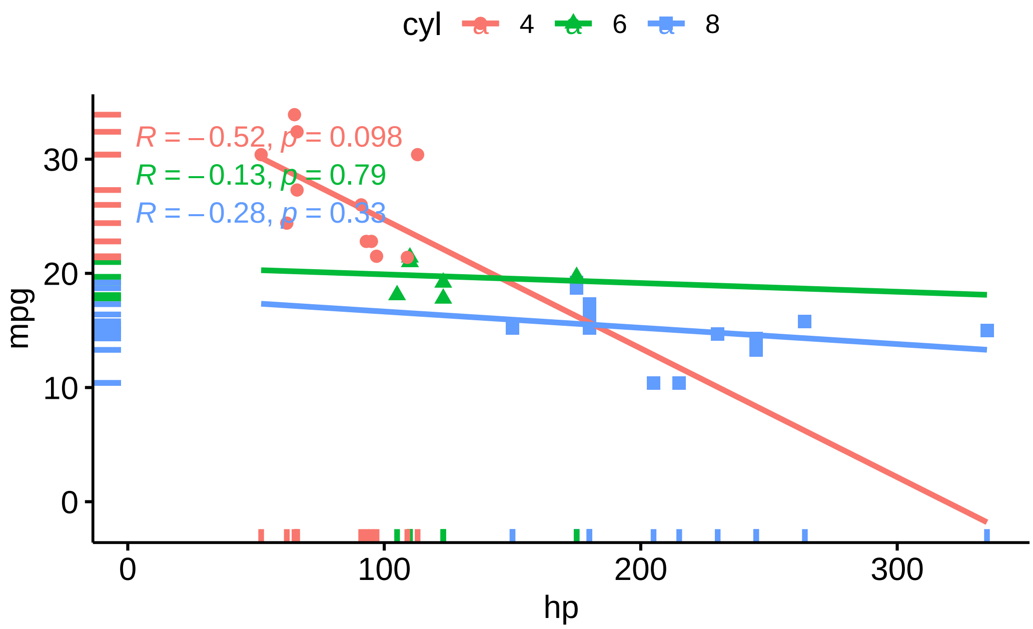
- The `aes()` function, short for aesthetics, designates the variables and their roles in the plot. In this code:
  - The `hp` variable is plotted on the x-axis.
  - The `mpg` variable is mapped to the y-axis.
  - The `color` attribute is set based on the `cyl` variable, which presumably indicates the number of cylinders in a car engine. The use of `factor(cyl)` ensures that the `cyl` variable is treated as a discrete factor rather than a continuous variable, which is essential for color differentiation.
- `geom_point()` introduces a scatter plot layer, meaning that the relationship between `hp` and `mpg` will be represented using individual points, with each point's color reflecting the number of cylinders as specified in the aesthetic mapping.
- The `labs()` function provides a convenient way to label the axes. Here, the x-axis receives the label "Horsepower" and the y-axis is labeled "Miles per gallon".

- The `scale_color_discrete()` function customizes the color scale for discrete variables. By specifying the `name` argument as "Cylinders", it ensures that the legend accompanying the color scale in the plot will be labeled as "Cylinders", making it clear to viewers that the colors of the points represent different cylinder counts.

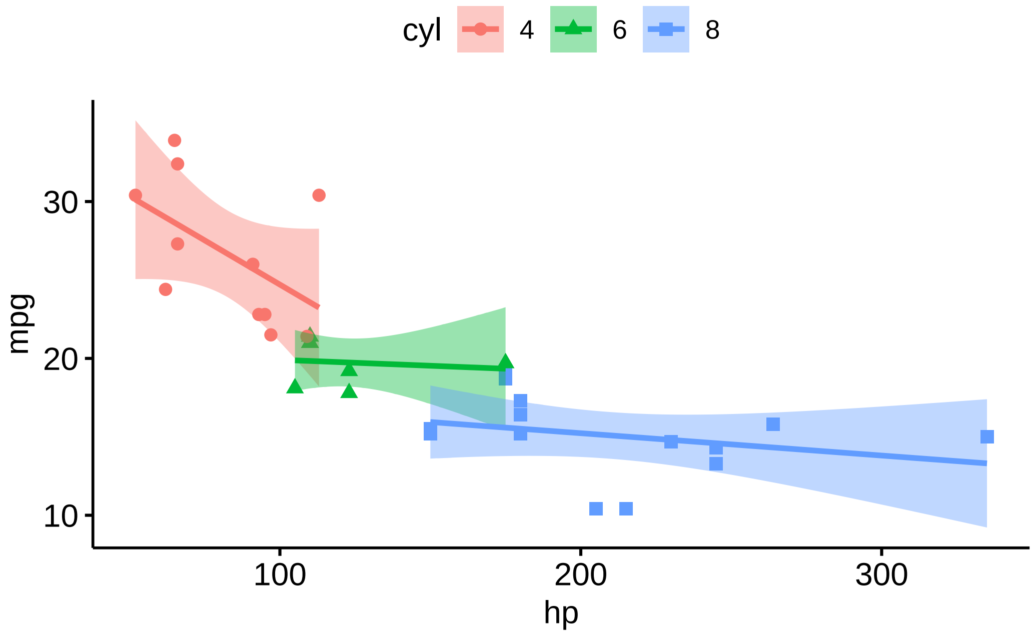
```r
# Load the 'ggpubr' library, which contains the 'ggscatter()' function  [3]
library(ggpubr)

# Create a scatter plot using the 'ggscatter()' function
ggscatter(tb,
          x = "hp", y = "mpg",
          color = "cyl",  # Color points by groups "cyl"
          shape = "cyl",  # Change point shape by groups "cyl"
          rug = TRUE)    # Add marginal rugs
```



```r
library(ggpubr)
# Create a scatter plot using the 'ggscatter()' function   [3]
ggscatter(tb,
          x = "hp", y = "mpg",
          add = "reg.line",    # Add regression line
          color = "cyl",       # Color by groups "cyl"
          shape = "cyl",       # Change point shape by groups "cyl"
          fullrange = TRUE,    # Extend the regression line
          rug = TRUE           # Add marginal rug (marginal density)
          ) +
# Add the Pearson correlation coefficient
# Color the label by groups "cyl" and specify the label coordinates
stat_cor(aes(color = cyl),
         label.x = 3)
```



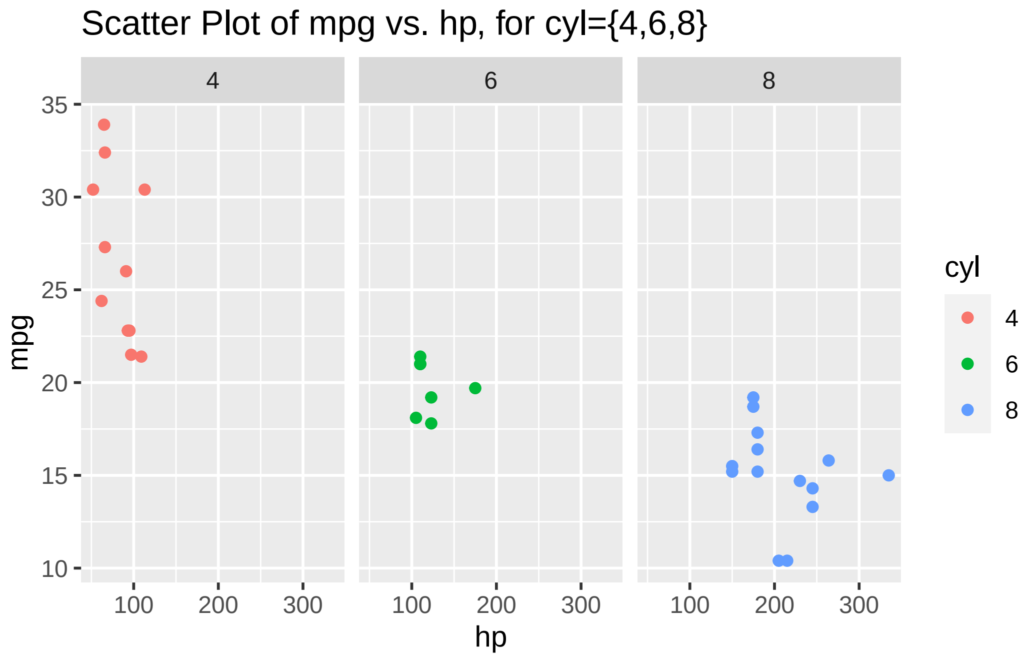
```
# Create a scatter plot using the 'ggscatter()' function  [3], [5]
# Set the x-axis variable to "hp" and the y-axis variable to "mpg"
ggscatter(tb,
          x = "hp", y = "mpg",
          add = "reg.line",  # Add regression line
          conf.int = TRUE,   # Include confidence interval
          color = "cyl",     # Color by groups "cyl"
          shape = "cyl"      # Change point shape by groups "cyl"
          )
```



Scatterplot faceted by a Categorical variable, using `ggplot()`

This will create a scatterplot of miles per gallon (mpg) against weight, with each plot faceted by the number of cylinders in the engine (cyl).

```r
# Create a scatter plot using the 'ggplot()' function from the 'ggplot2' library
# Set the x-axis variable to "hp", the y-axis variable to "mpg", and color points by "cyl"
ggplot(tb,
       aes(x = hp,
           y = mpg,
           color = cyl)) +
# Add points to the plot
geom_point() +
# Facet the plot by the variable "cyl" (separate plots by "cyl")
facet_grid(. ~ cyl) +
ggtitle("Scatter Plot of mpg vs. hp, for cyl={4,6,8}")
```



Discussion:

- The foundational layer is initialized with the `ggplot()` function. This function takes in a dataset, `tb`, and aesthetic mappings that determine how variables are displayed. In this piece of code:

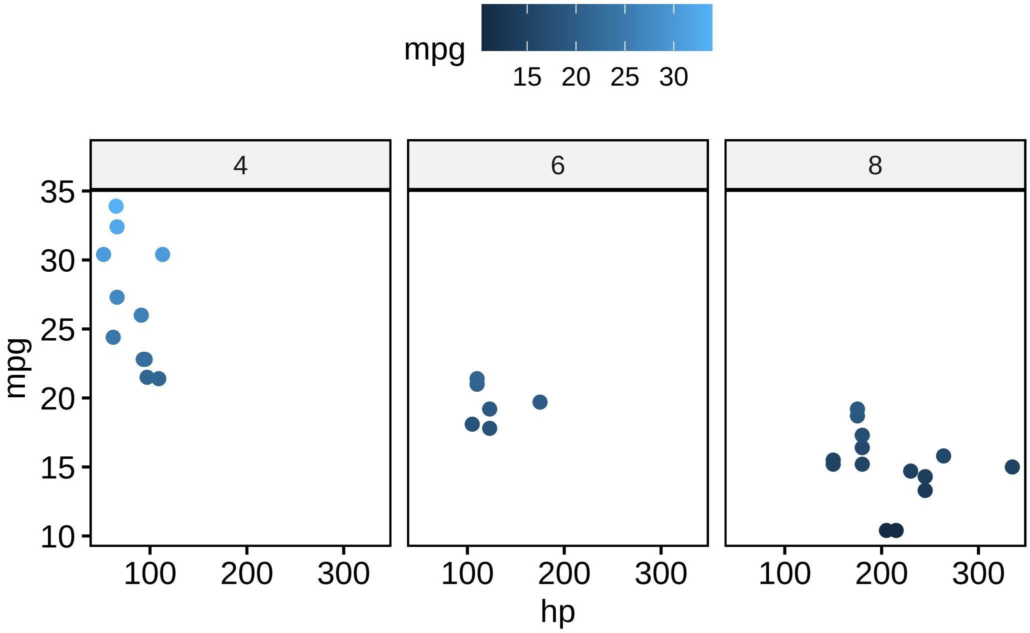
  - `hp` is chosen to be plotted on the x-axis.
  - `mpg` is selected for the y-axis.
  - The color of the points will be determined by the `cyl` variable.

- The addition of the `geom_point()` layer ensures that a scatter plot will represent the relationship between `hp` and `mpg`. Each point's color will correspond to the value of the `cyl` variable.

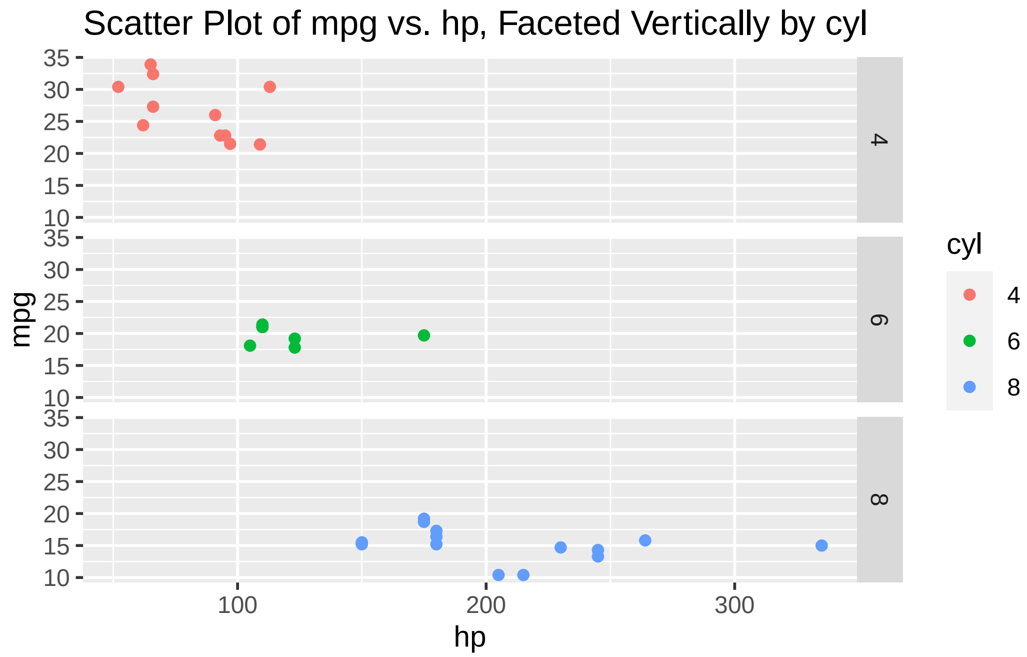
- The `facet_grid()` function introduces the concept of faceting. Faceting divides a plot into multiple panels based on the levels of one or more factors. In this case, the plot is faceted horizontally (`~ cyl`), meaning that separate panels are created for each unique value of `cyl`. The `.` before the `~` indicates that there's no faceting vertically.

- Finally, the `ggtitle()` function provides the entire plot with a title, which is "Scatter Plot of mpg vs. hp, for cyl={4,6,8}". This title clearly communicates the main theme of the plot and indicates that it showcases relationships for cars with 4, 6, or 8 cylinders.

```
# Create a scatter plot using the 'ggscatter()' function
ggscatter(tb,
          x = "hp",
          y = "mpg",
          color = "mpg", # Color points by the continuous variable "mpg"
          facet.by = "cyl") # Facet the plot by "cyl"
```



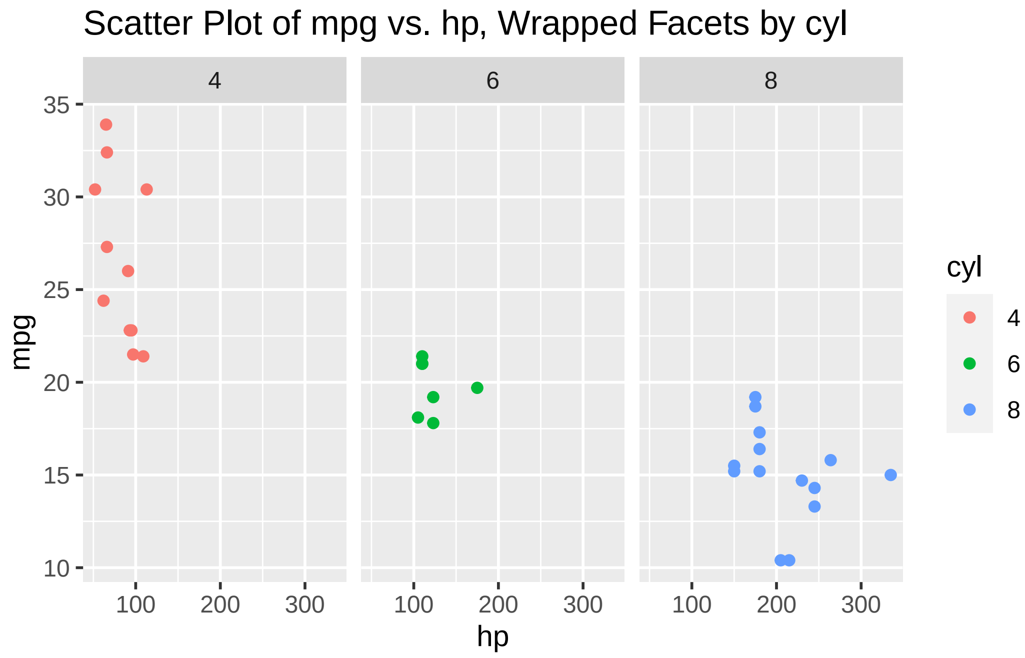
```
# Create a scatter plot using the 'ggplot()' function from the 'ggplot2' library
# Set the x-axis variable to "hp", the y-axis variable to "mpg", and color points by "cyl"
ggplot(tb,
       aes(x = hp,
           y = mpg,
           color = cyl)) +
# Add points to the plot
geom_point() +
# Facet the plot vertically by the variable "cyl" (separate plots by "cyl")
facet_grid(cyl ~ .) +
ggtitle("Scatter Plot of mpg vs. hp, Faceted Vertically by cyl")
```



Discussion:

- The primary difference between the two code snippets lies in how the faceting is implemented using the `facet_grid()` function.
- In the original code, `facet_grid(. ~ cyl)` is used, which means the scatter plots are faceted horizontally based on the unique values of the `cyl` variable; each unique cylinder count gets its own column.
- Conversely, in the updated code with `facet_grid(cyl ~ .)`, the scatter plots are faceted vertically based on the unique values of the `cyl` variable; each unique cylinder count gets its own row.

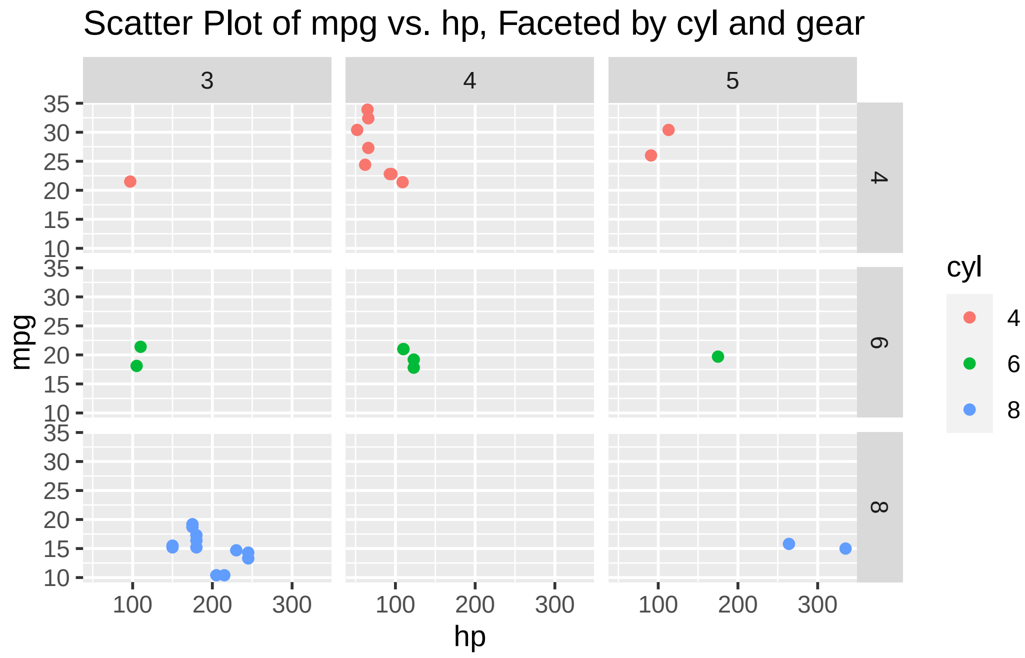
```r
# Create a scatter plot using the 'ggplot()' function from the 'ggplot2' library
# Set the x-axis variable to "hp", the y-axis variable to "mpg", and color points by "cyl"
ggplot(tb,
       aes(x = hp,
           y = mpg,
           color = cyl)) +
# Add points to the plot
geom_point() +
# Wrap facets by the variable "cyl" with 3 columns (separate plots by "cyl")
facet_wrap(~ cyl, ncol = 3) +
ggtitle("Scatter Plot of mpg vs. hp, Wrapped Facets by cyl")
```



Discussion:

- This approach creates a wrapped grid of facets based on `cyl`.
- The `ncol = 3` argument specifies that up to three facets will be placed in a row before wrapping to the next row. You can adjust this as needed based on the number of levels in the faceting variable and the desired layout.

```r
# Create a scatter plot using the 'ggplot()' function from the 'ggplot2' library
# Set the x-axis variable to "hp", the y-axis variable to "mpg", and color points by "cyl"
ggplot(tb,
       aes(x = hp,
           y = mpg,
           color = cyl)) +
# Add points to the plot
geom_point() +
# Facet the plot by the variables "cyl" and "gear" (separate plots by combinations of "cyl
facet_grid(cyl ~ gear) +
ggtitle("Scatter Plot of mpg vs. hp, Faceted by cyl and gear")
```



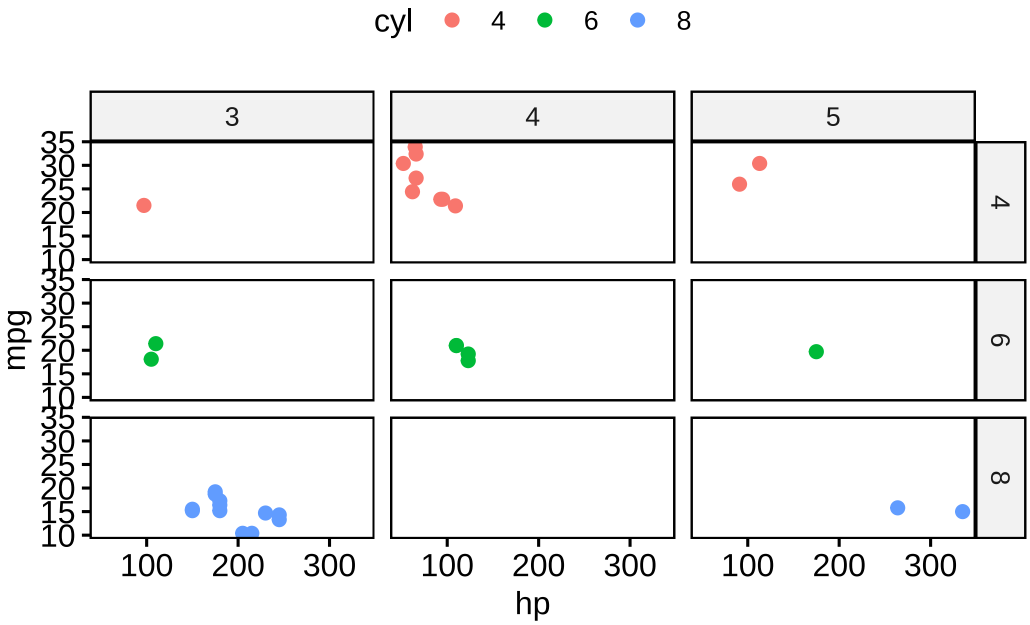
Discussion:

- In this code, within the `aes()` aesthetics function:
  - The variable `hp` is mapped to the x-axis.
  - The variable `mpg` is mapped to the y-axis.
  - The color of individual points is determined by the `cyl` variable, which probably represents the number of cylinders in an engine.

- The `geom_point()` function is introduced to represent the relationship between `hp` and `mpg` as a scatter plot. The colors of the individual points will correspond to the values of the `cyl` variable.

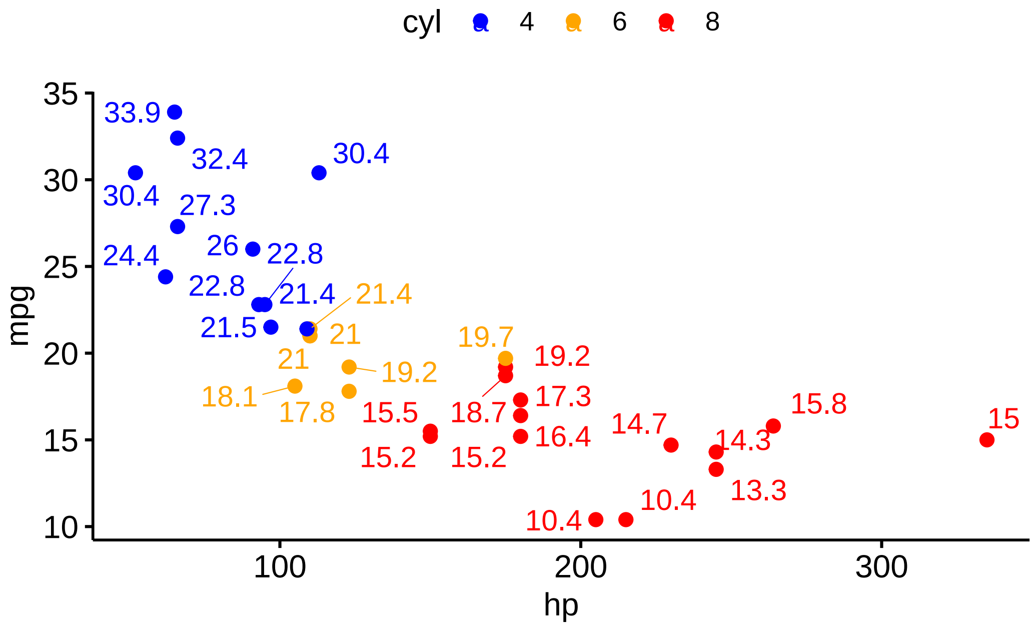
- The `facet_grid(cyl ~ gear)` function is the standout feature in this code. Here, the plots are faceted based on two categorical variables:
  - `cyl`, which is mapped to rows. Each unique value of `cyl` will generate a new row of plots.
  - `gear`, which is mapped to columns. Each unique value of `gear` will generate a new column of plots.
  - The resultant grid will represent combinations of `cyl` and `gear` values, with each cell in the grid showing the relationship between `hp` and `mpg` for a specific combination of `cyl` and `gear`. [4]

```
# Create a scatter plot using the 'ggscatter()' function    [3]
# Set the x-axis variable to "hp" and the y-axis variable to "mpg"
# Color points by the categorical variables "cyl" and "gear"
# Facet the plot by the combinations of "cyl" and "gear" (separate plots by combinations)
ggscatter(tb,
          x = "hp", y = "mpg",
          color = "cyl",
          facet.by = c("cyl", "gear"))
```

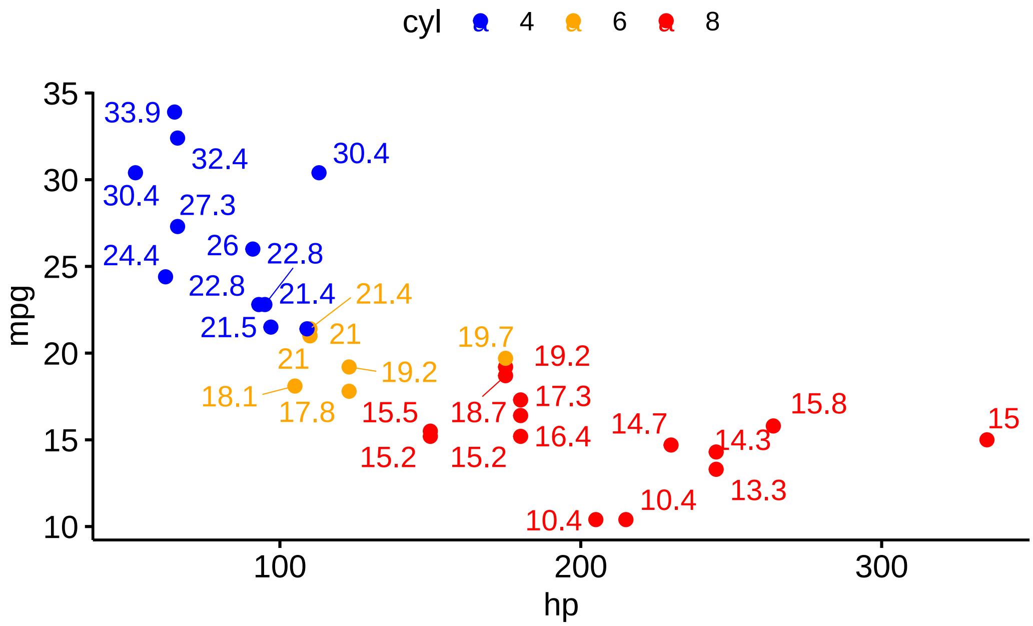


Scatterplot colored by a Categorical variable, with textual annotation, using ggpubr()

```
# Create a scatter plot using the 'ggscatter()' function   [3]
# Set the x-axis variable to "hp" and the y-axis variable to "mpg"
# Color points by the categorical variable "cyl" using the specified palette
# Add textual annotations using the "mpg" variable and enable point repelling
ggscatter(tb,
          x = "hp",
          y = "mpg",
          color = "cyl",
          palette = c("blue", "orange", "red"),
          label = "mpg",
          repel = TRUE)
```

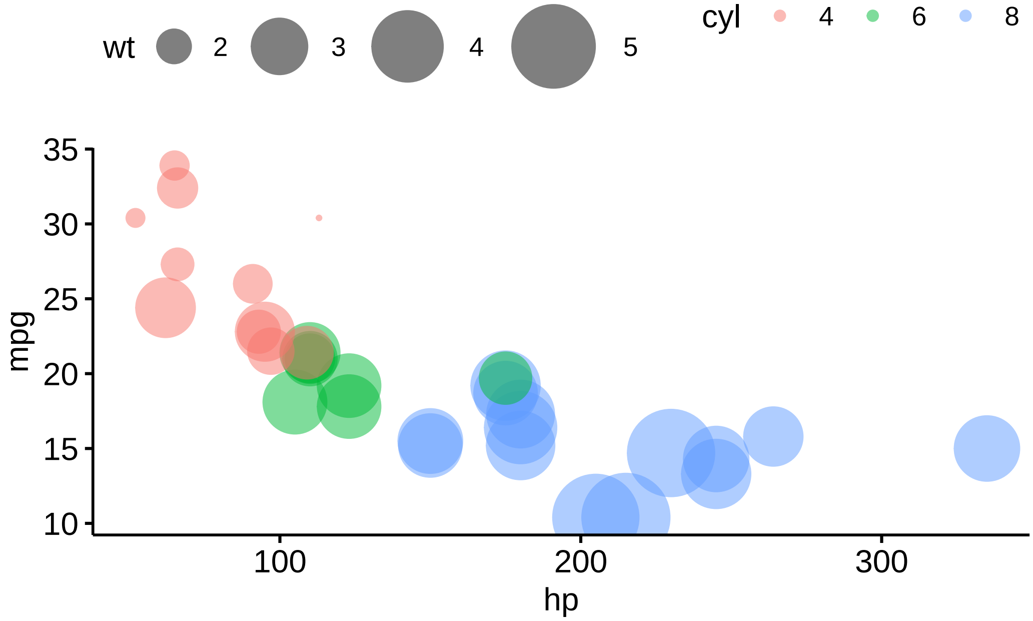


```
# Create a scatter plot using the 'ggscatter()' function    [3]
# Set the x-axis variable to "hp" and the y-axis variable to "mpg"
# Color points by the categorical variable "cyl" using the specified palette
# Add textual annotations using the "mpg" variable and enable point repelling
ggscatter(tb,
          x = "hp",
          y = "mpg",
          color = "cyl",
          palette = c("blue", "orange", "red"),
          label = "mpg",
          repel = TRUE) +

# Set limits to fit the entire plot inside a box
coord_cartesian(ylim = c(min(tb$mpg), max(tb$mpg)),
                xlim = c(min(tb$hp), max(tb$hp)))
```



Bubble Chart

```
# Create a scatter plot using the 'ggscatter()' function    [3]
# Set the x-axis variable to "hp" and the y-axis variable to "mpg"
# Color points by the categorical variable "cyl"
# Adjust point size based on the continuous variable "wt" and set transparency (alpha) to
ggscatter(tb,
          x = "hp", y = "mpg",
          color = "cyl",
          size = "wt", alpha = 0.5) +

# Adjust the range of point sizes for better visualization
scale_size(range = c(0.5, 15))
```



Summary of Chapter 15 – Bivariate Continuous data (Part 4 of 4)

This chapter provides a comprehensive guide to exploring bivariate continuous data using R's ggplot2 and ggpubr packages, focusing on the 'mtcars' dataset. Initially, the data is formatted into a tibble, and key variables are transformed into factors for nuanced analysis. The chapter emphasizes scatterplot creation using ggplot2, illustrating the relationship between variables like horsepower (hp) and miles per gallon (mpg). Techniques such as custom labeling, adding regression lines, and layering are demonstrated to enhance plot interpretability. ggplot2's layering approach is highlighted for its effectiveness in building complex plots incrementally.

Further, the ggpubr package's ggscatter function is introduced for advanced scatterplot customization. This includes altering point shapes, adding marginal rugs, regression lines, and correlation coefficients. The integration of categorical variables into scatter plots is extensively discussed. This involves coloring points by categories like the number of cylinders and employing various faceting techniques to compare data across categories. Advanced topics like point annotations, point repelling for clarity, and bubble charts where point sizes vary with a continuous variable are also covered. These techniques provide a multi-dimensional view of the data, showcasing the versatility of ggplot2 and ggpubr in visual data analysis.

Overall, the chapter equips readers with the skills to create and enhance scatter plots in R, offering insightful methods for detailed bivariate data analysis.

References

Basic R programmng

[1] Chambers, J. M. (2008). *Software for data analysis: programming with R* (Vol. 2, No. 1). New York: Springer.

Crawley, M. J. (2012). *The R book.* John Wiley & Sons.

Gardener, M. (2012). *Beginning R: the statistical programming language.* John Wiley & Sons.

Grolemund, G. (2014). *Hands-on programming with R: Write your own functions and simulations.* " O'Reilly Media, Inc.".

Kabacoff, R. (2022). *R in action: data analysis and graphics with R and Tidyverse.* Simon and Schuster.

Peng, R. D. (2016). *R programming for data science* (pp. 86-181). Victoria, BC, Canada: Leanpub.

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Tippmann, S. (2015). Programming tools: Adventures with R. *Nature*, *517*(7532), 109-110.

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for data science.* " O'Reilly Media, Inc.".

ggplot2:

[2] Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. https://ggplot2.tidyverse.org

Wickham, H., & Grolemund, G. (2016). R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media.

Wickham, H. (2020). ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics (Version 3.3.2) [Computer software]. Retrieved from https://CRAN.R-project.org/package=ggplot2

Wickham, H., et al. (2020). dplyr: A Grammar of Data Manipulation (Version 1.0.2) [Computer software]. Retrieved from https://CRAN.R-project.org/package=dplyr

Wilkinson, L. (2005). The Grammar of Graphics (2nd ed.). Springer-Verlag.

Wickham, H., et al. (2020). tibble: Simple Data Frames (Version 3.0.3) [Computer software]. Retrieved from https://CRAN.R-project.org/package=tibble

ggpubr:

[3] Kassambara A (2023). ggpubr: ‘ggplot2’ Based Publication Ready Plots. R package version 0.6.0, https://rpkgs.datanovia.com/ggpubr/.

Statistics using R

[4] Braun, W. J., & Murdoch, D. J. (2021). *A first course in statistical programming with R.* Cambridge University Press.

Cohen, Y., & Cohen, J. Y. (2008). *Statistics and Data with R: An applied approach through examples.* John Wiley & Sons.

Dalgaard, P. (2008). *Introductory statistics with R.* springer publication.

Davies, T. M. (2016). *The book of R: a first course in programming and statistics.* No Starch Press.

Everitt, B. S., & Hothorn, T. (2014). A Handbook of Statistical Analyses Using R. Chapman and Hall/CRC.

Field, A., Miles, J., & Field, Z. (2012). Discovering statistics using R. Sage Publications.

Fox, J., & Weisberg, S. (2018). *An R companion to applied regression.* Sage publications.

Hyndman, R. J., & Fan, Y. (1996). Sample quantiles in statistical packages. The American Statistician, 50(4), 361-365.

Matloff, N. (2011). *The art of R programming: A tour of statistical software design.* No Starch Press.

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Schumacker, R. E. (2014). *Learning statistics using R.* Sage Publications.

Schumacker, R., & Tomek, S. (2013). *Understanding statistics using R.* Springer Science & Business Media.

Regression:

[5] Fox, J. and Weisberg, S. (2011). An R Companion to Applied Regression, Second Edition. Thousand Oaks CA: Sage.