Continuous Data (2 of 2)

Aug 8, 2023

Exploring Univariate Continuous Data using `ggplot2` and `ggpubr`

THIS CHAPTER demonstrates the use of the popular **`ggplot2`** and **`ggpubr`** packages to further explore *univariate, continuous* data.

1. **`ggplot2`**: In the `ggplot2` package for instance, the function `geom_boxplot()` produces box plots, `geom_violin()` creates violin plots, and `geom_histogram()` and `geom_density()` generate histograms and density plots, respectively. The related `ggbeeswarm` package can be used for creating bee swarm plots.
2. **`ggpubr`**: The `ggpubr` package in R augments `ggplot2` by offering tools for creating publication-ready plots. It enables simplified plotting with easy-to-use functions like gghistogram(), ggdensity(), ggboxplot(), ggviolin, and makes it easy to merge multiple plots with `ggarrange()`, and provides specialized themes for a polished look. Essentially, `ggpubr` merges `ggplot2`'s extensive customization with the ease of creating visually appealing and informative plots.
3. **Data**: Suppose we run the following code to prepare the `mtcars` data for subsequent analysis and save it in a tibble called `tb`.

```r
# Load the required libraries, suppressing annoying startup messages
library(tibble)
suppressPackageStartupMessages(library(dplyr))
# 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
# Directly access the data columns of tb, without tb$mpg
```

```
attach(tb)
```

4. We load the ggplot2, dplyr and ggthemes packages. The package ggthemes allows us to use a variety of themes.

```
library(dplyr)
library(ggthemes)
suppressPackageStartupMessages(library(ggplot2))
```

5. Let's take a closer look at some of the most effective ways of Visualizing Univariate Continuous Data using ggplot2 and related packages, including

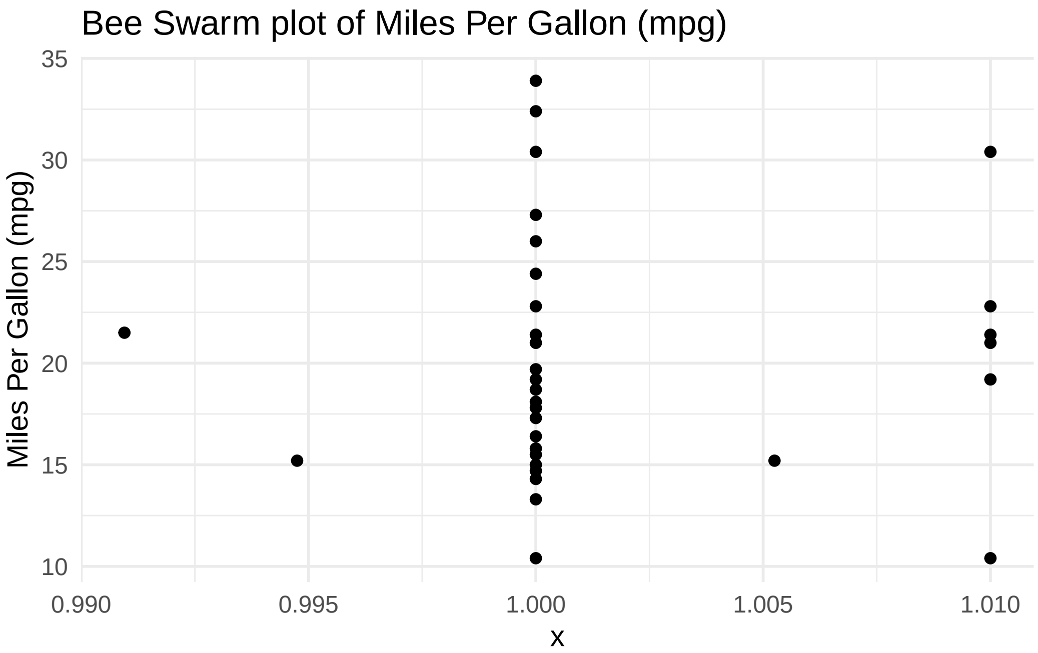
- Bee Swarm plots using ggbeeswarm
- Histograms using ggplot2 and ggpubr
- PDF and CDF Density plots using ggplot2 and ggpubr
- Box plots using ggplot2 and ggpubr
- Violin plots using ggplot2 and ggpubr
- Quantile-Quantile (Q-Q) Plots using ggplot2

Note that it is inconvenient to create Stem-and-Leaf plots using ggplot2.

Bee Swarm plot using ggbeeswarm

1. The bee swarm plot is an alternative to the box plot, where each point is plotted in a manner that avoids overlap.
2. We use the ggbeeswarm package on the mpg column of the tb tibble.

```
library(ggplot2)
library(ggbeeswarm) # Necessary for geom_beeswarm()
ggplot(tb,
       aes(x = 1,
           y = mpg)) +
  geom_beeswarm() +
  labs(title = "Bee Swarm plot of Miles Per Gallon (mpg)",
       y = "Miles Per Gallon (mpg)") +
  theme_minimal()
```



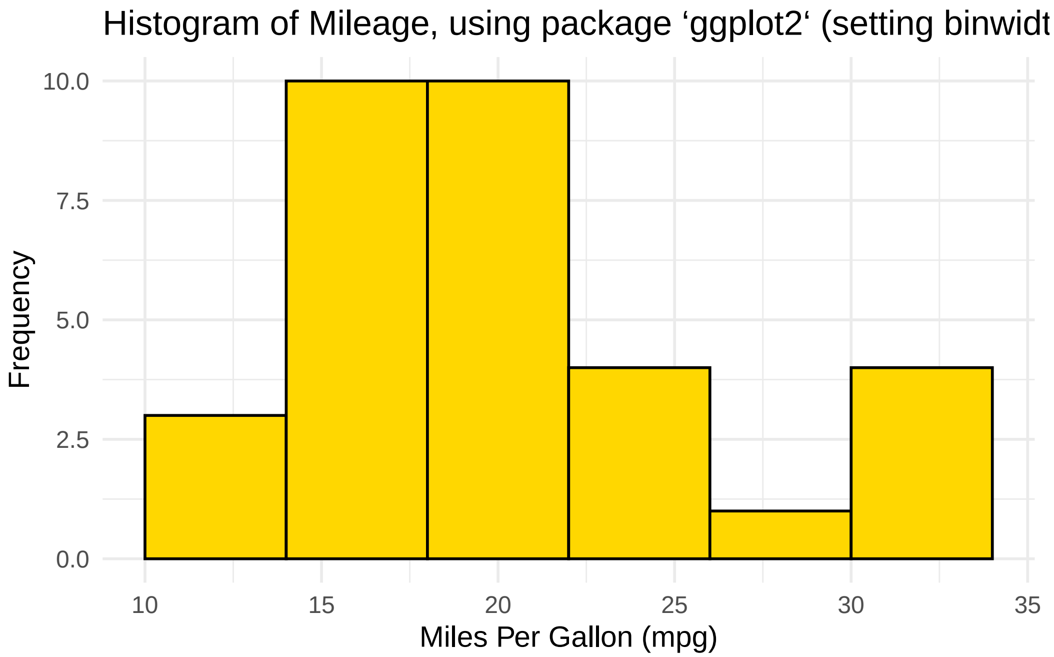
3. Discussion:

- Initially, we declare our dataset and the aesthetic mappings, defining how variables in the data are visually represented. For the bee swarm plot, we only need a y aesthetic, which is `mpg`. We set the x aesthetic to 1 as a placeholder, because bee swarm plots require an x aesthetic, but we only have one variable.
- Following that, we append a bee swarm plot using the `geom_beeswarm()` function.
- We use the `labs()` function to label the plot.
- We then adopt a minimalist theme by using `theme_minimal()` to give our plot a sleek and simple look.

Histogram using `ggplot2`

1. The following code creates a histogram using the `ggplot2` package. Here, we pre-specify the bin width and the resulting number of bins in the histogram depend on the range of the data.

```
ggplot(tb,
       aes(x = mpg)) +
  geom_histogram(binwidth = 4,
                 fill = "gold",
                 color = "black") +
  theme_minimal() +
  labs(title = "Histogram of Mileage, using package `ggplot2` (setting binwidth = 4)",
       x = "Miles Per Gallon (mpg)", y = "Frequency")
```

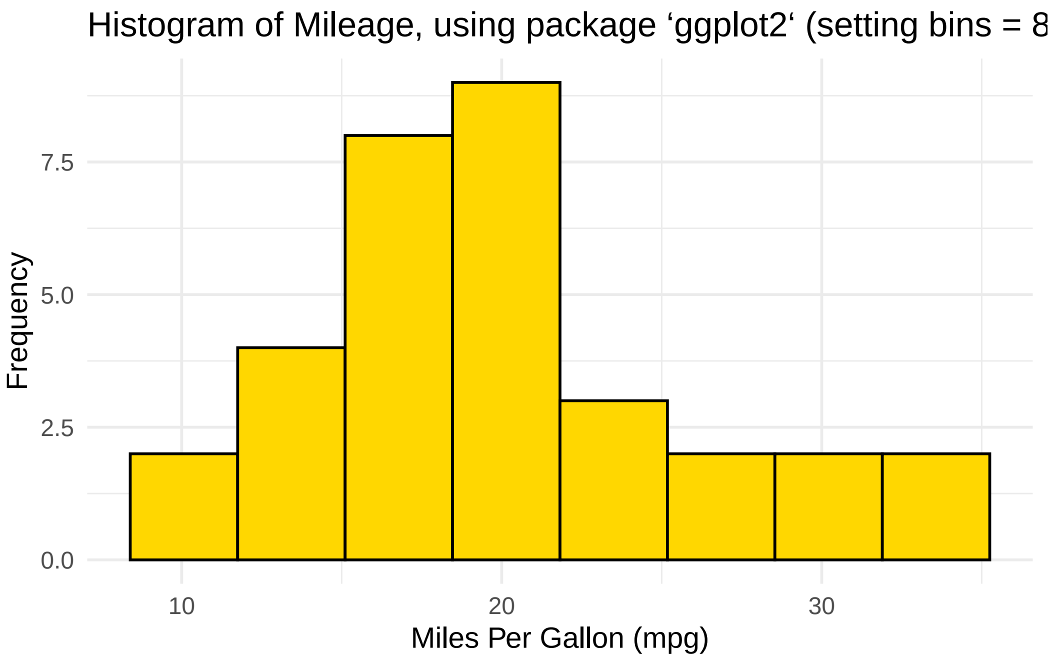


2. Discussion:

- The code `ggplot(tb, aes(x = mpg))` initializes a plot using the `tb` data frame, mapping the `mpg` column to the x-axis
- The histogram is created with `geom_histogram()`, using an adjustable `binwidth = 4`. Given this bin width, the resulting number of bins in the histogram depend on the range of `mpg`.
- The `binwidth` argument specifies the width of the bins in the histogram, and we have chosen `4` as an arbitrary width.
- We use `fill` and `color` to set the bar colors to be gold with a black border.
- A clean appearance is achieved with `theme_minimal()`, and titles and labels are added using `labs()`. [1]

3. We could alternately set the number of bins in the histogram, instead of specifying the bin width. In this case, the bin width gets calculated depending on the range of the data and the specified number of bins.

```r
ggplot(tb,
       aes(x = mpg)) +
  geom_histogram(bins = 8,
                 fill = "gold",
                 color = "black") +
  theme_minimal() +
  labs(title = "Histogram of Mileage, using package `ggplot2` (setting bins = 8)",
       x = "Miles Per Gallon (mpg)", y = "Frequency")
```



4. Discussion:

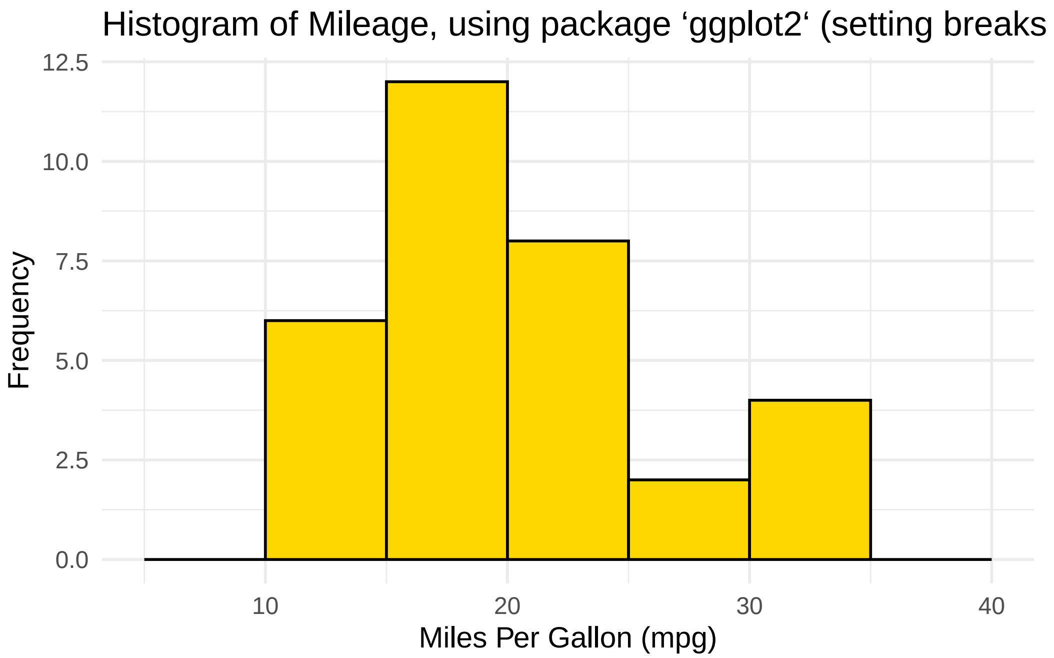
using the `bins` argument within the `geom_histogram()` function instead of `binwidth`. For example, in order to create a histogram with 12 bins of equal width, we can modify the code as follows:

- We instruct R to create a histogram having 8 bins of equal width, by setting `bins = 8` ingeom_histogram()‘
- The width of each bin is adjusted by dividing the range of `mpg` by the number of specified bins.

5. Alternately, we can specify custom bin ranges in a histogram. In this this approach, we supply a vector of breakpoints which defines the range of each bin. For example, the

following code defines histogram bins with ranges of 5-10, 10-15, 15-20, 20-25, 25-30, 30-35, 35-40, for the mpg variable

```
ggplot(tb,
       aes(x = mpg)) +
  geom_histogram(breaks = seq(5, 40, by = 5),
                 fill = "gold",
                 color = "black") +
  theme_minimal() +
  labs(title = "Histogram of Mileage, using package `ggplot2` (setting breaks of 5)",
       x = "Miles Per Gallon (mpg)", y = "Frequency")
```



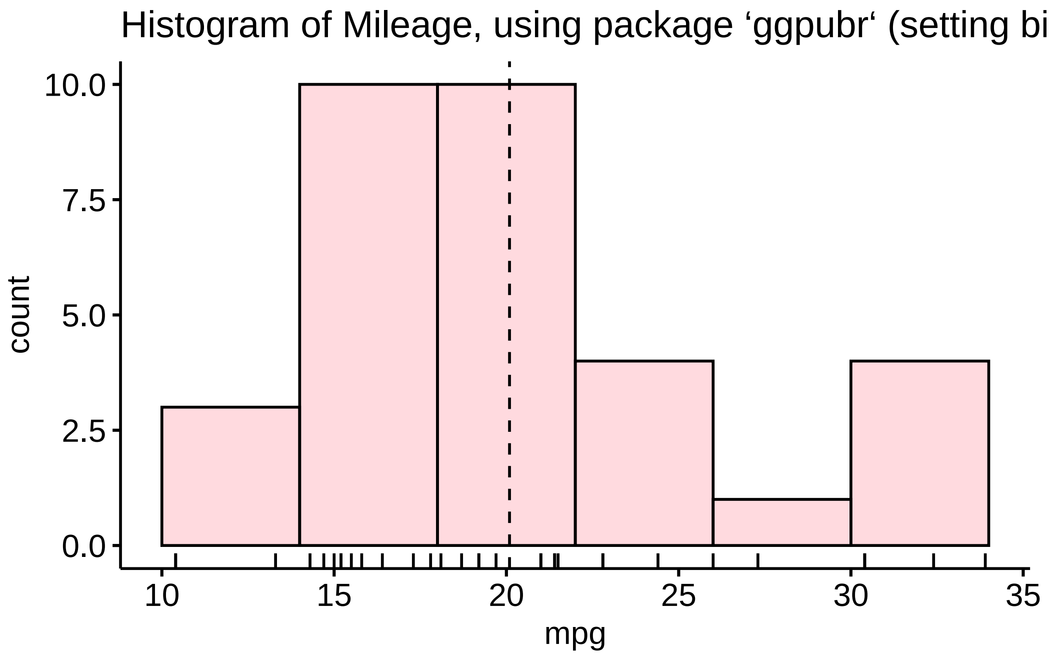
6. **Discussion:**

- `ggplot(tb, aes(x = mpg))` initializes a ggplot object with the `tb` data frame and sets the `mpg` column as the x-axis variable.
- `geom_histogram()` adds a histogram layer, in which `breaks = seq(5, 40, by = 5)` specifies bin edges using a sequence that starts at 5, ends at 40, and increases by 5 units. This results in bins like [5,10), [10,15), and so on.

Histogram using `ggpubr`

7. Recreating a histogram having binwidth of 4, using package `ggpubr`

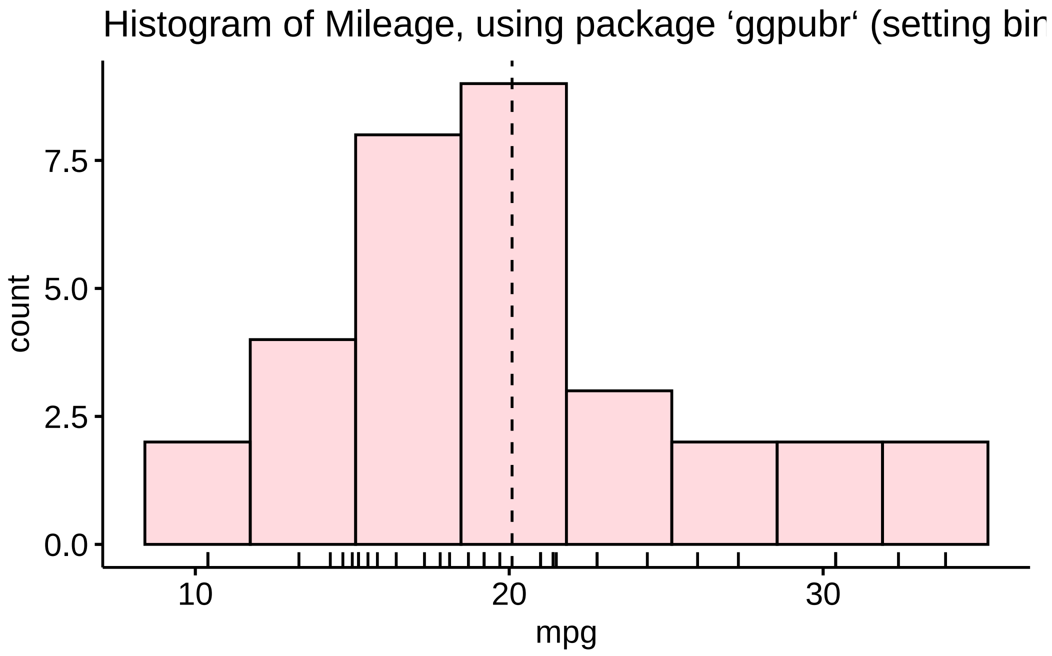
```
library(ggpubr)
gghistogram(tb,
            x = "mpg",
            binwidth = 4,
            add = "mean",
            rug = TRUE,
            color = "black" ,
            fill = "lightpink",
            title = "Histogram of Mileage, using package `ggpubr` (setting binwidth = 4) "
)
```



8. **Discussion:**

9. Recreating a histogram having 8 bins, using package `ggpubr`

```
library(ggpubr)
gghistogram(tb,
            x = "mpg",
            bins = 8,
            add = "mean",
            rug = TRUE,
            color = "black" ,
            fill = "lightpink",
            title = "Histogram of Mileage, using package `ggpubr` (setting bins = 8) "
)
```



9. **Discussion:**

gghistogram(tb, x = “mpg”, add = “mean”, bins = 6, rug = TRUE, color = “black”, fill = “gold”, title = “Histogram of Miles Per Gallon (mpg), using ggpubr”):

This line uses the gghistogram() function to create a histogram of the mpg variable from the tb dataset.

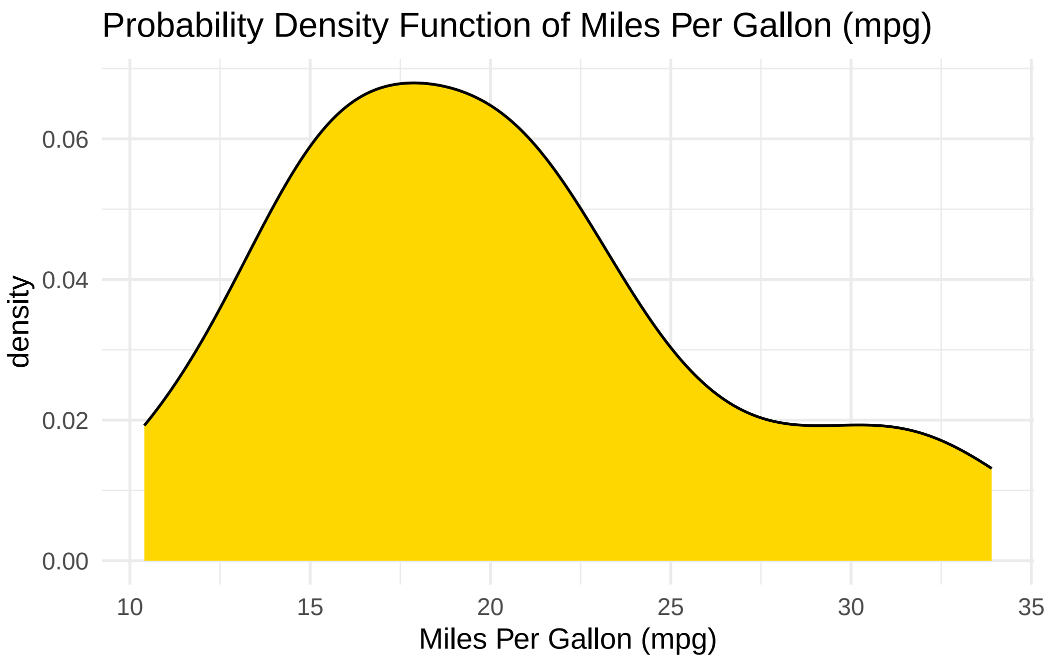
- tb: Specifies the dataset to use.
- x = “mpg”: Specifies the variable to create a histogram for.
- add = “mean”: Adds a vertical line at the mean of mpg.
- bins = 8: Specifies the number of bins in the histogram. This can be adjusted based on the specific data and desired level of granularity.
- rug = TRUE: Adds a rug plot at the bottom of the histogram, which displays a small vertical line for each observation along the range of mpg.

- color = “black”: Specifies the color of the border of the bars in the histogram.
- fill = “gold”: Specifies the fill color of the bars in the histogram.
- title = “Histogram of Miles Per Gallon (mpg), using ggpubr”: Specifies the title of the plot.

Probability Density Function (PDF) plot using `ggplot2`

- Recall that this type of plot shows the distribution of a single variable, and the area under the curve represents the probability of an observation falling within a particular range of values.

```
ggplot(tb,
       aes(x = mpg)) +
  geom_density(fill = "gold") +
  theme_minimal() +
  labs(title = "Probability Density Function of Miles Per Gallon (mpg)",
       x = "Miles Per Gallon (mpg)", y = "density")
```



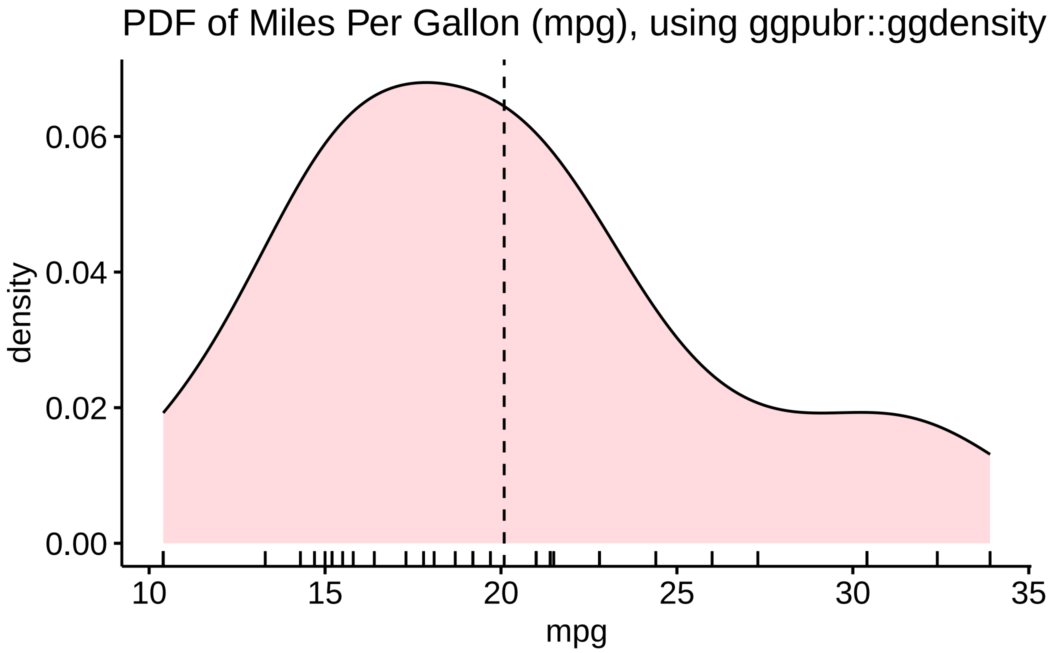
- We designate our data source and the aesthetic mappings using the `ggplot()` function. The aesthetic mapping for x is `mpg`.

- Subsequently, we append a density plot to our plot by using the `geom_density()` function. We fill the area under the curve with a light pink color by setting `fill` to "lightpink" and `alpha` to 0.5.

PDF using `ggpubr`

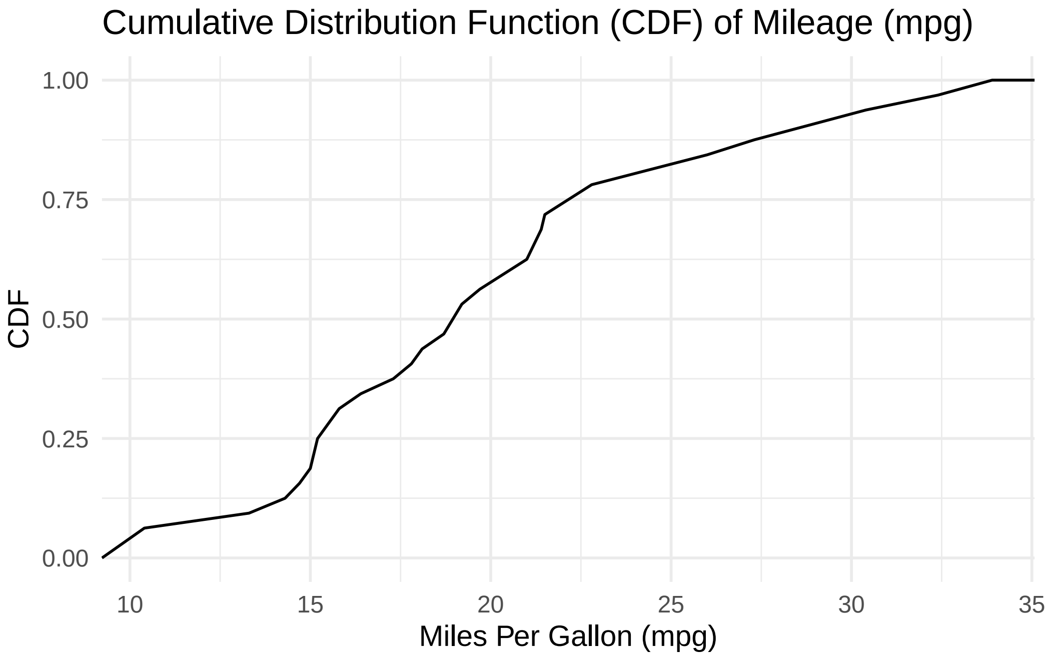
- The provided R code creates a PDF of the mpg (miles per gallon) variable in the tb dataset, using the `ggdensity()` function from the `ggpubr` package.

```r
library(ggpubr)
ggdensity(tb,
            x = "mpg",
            add = "mean",
            rug = TRUE,
            color = "black" ,
            fill = "lightpink",
            title = "PDF of Miles Per Gallon (mpg), using ggpubr::ggdensity()"
)
```



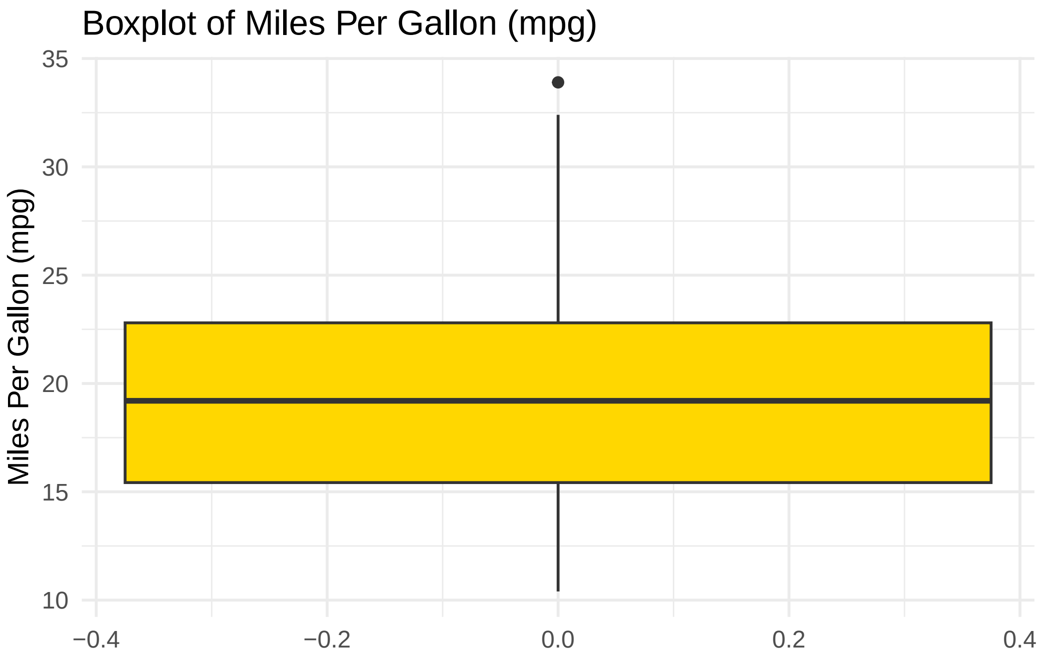
Cumulative Distribution Function (CDF) Plot using `ggplot2`

```r
# Load required library
library(ggplot2)
# Create a CDF plot
ggplot(tb, aes(x = mpg)) +
  stat_ecdf(geom = "line", color = "black") +
  labs(x = "Miles Per Gallon (mpg)", y = "CDF",
       title = "Cumulative Distribution Function (CDF) of Mileage (mpg)") +
  theme_minimal()
```



Boxplots using `ggplot2`

```r
ggplot(tb,
       aes(y = mpg)) +
  geom_boxplot(fill = "gold") +
  theme_minimal() +
  labs(title = "Boxplot of Miles Per Gallon (mpg)",
       y = "Miles Per Gallon (mpg)")
```

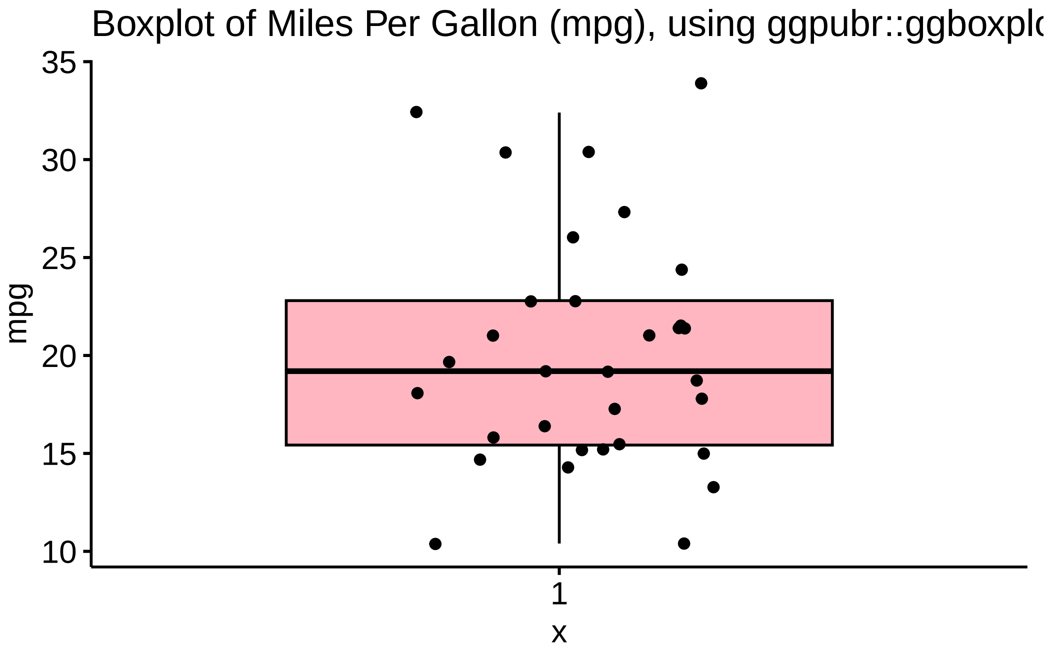


1. We're first specifying the source of our data and the aesthetic mappings, which define how variables in the data are mapped to visual properties. In this case, we're only specifying the y aesthetic, since a boxplot of a single variable doesn't need an x aesthetic. The y aesthetic is mapped to the mpg variable.
2. Then, we add a boxplot using `geom_boxplot()`.
3. After this, `theme_minimal()` is used to apply a minimalist theme to the plot, which has a clean and professional appearance.
4. Finally, we're adding some labels to the plot with the `labs()` function.

Boxplot using `ggpubr`

- The provided R code creates a Boxplot of the `mpg` (miles per gallon) variable in the `tb` dataset, using the `ggboxplot()` function from the `ggpubr` package.

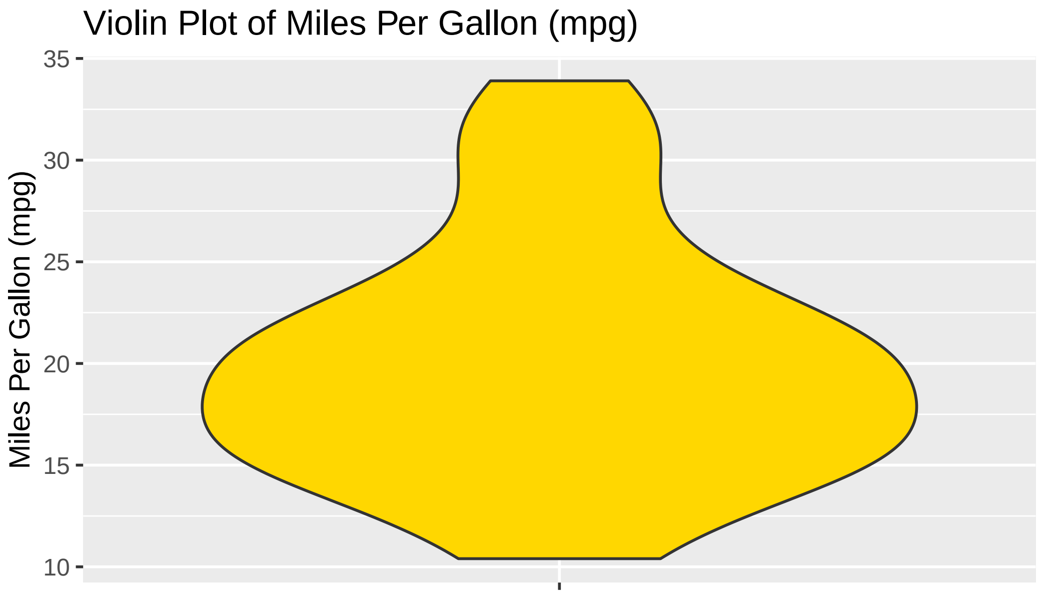
```
library(ggpubr)
ggboxplot(tb,
            y = "mpg",
            rug = TRUE,
            color = "black" ,
            fill = "lightpink",
            add = "jitter",
            title = "Boxplot of Miles Per Gallon (mpg), using ggpubr::ggboxplot()"
)
```



Violin plot using `ggplot2`

1. We will now generate a violin plot using the ggplot2 package for the `mpg` column

```
ggplot(tb,
       aes(x = "", y = mpg)) +
  geom_violin(fill = "gold") +
  labs(x = "", y = "Miles Per Gallon (mpg)", title = "Violin Plot of Miles Per Gallon (mpg
```

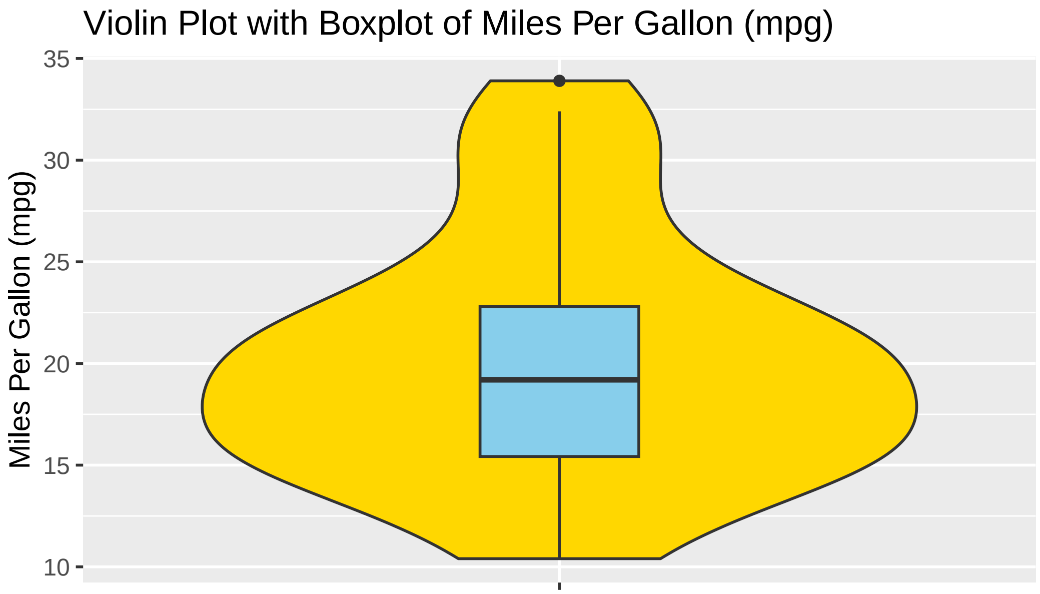


2. Discussion:

- Here, `aes()` defines aesthetic mappings, mapping `mpg` to the y-axis.
- `geom_violin()` generates the violin plot, and `labs()` adds a title for the plot and labels the y-axis

3. We can add a boxplot to the violin plot, as follows:

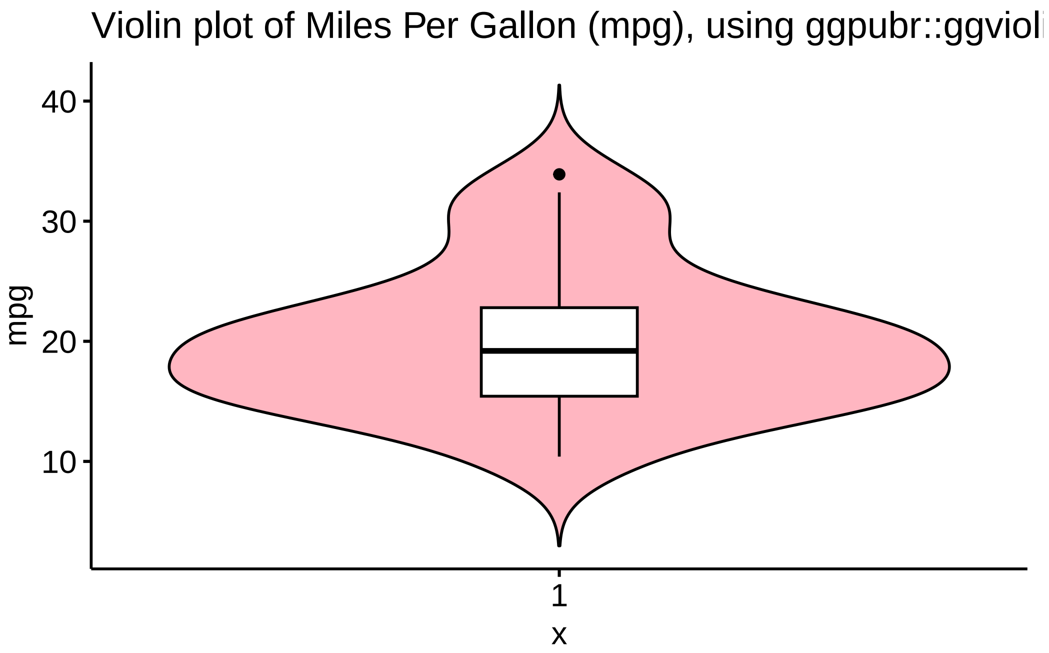
```
ggplot(tb, aes(x = "", y = mpg)) +
  geom_violin(fill = "gold") +
  geom_boxplot(fill = "skyblue", width = 0.2) +
  labs(x = "", y = "Miles Per Gallon (mpg)", title = "Violin Plot with Boxplot of Miles Pe
```



Violin plot using `ggpubr`

- The provided R code creates a Boxplot of the `mpg` (miles per gallon) variable in the `tb` dataset, using the `ggboxplot()` function from the `ggpubr` package.

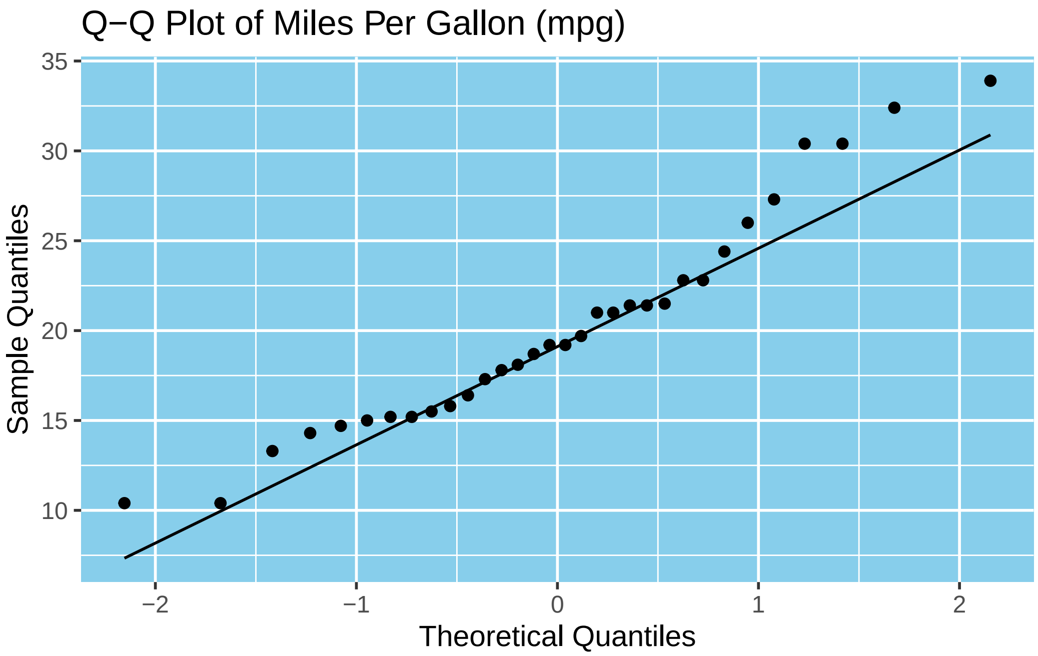
```r
library(ggpubr)
ggviolin(tb,
            y = "mpg",
            rug = TRUE,
            color = "black" ,
            fill = "lightpink",
            add = "boxplot", add.params = list(fill = "white"),
            title = "Violin plot of Miles Per Gallon (mpg), using ggpubr::ggviolin()"
)
```



Quantile-Quantile (Q-Q) Plots using `ggplot2`

- In order to create a Q-Q plot, we use the `ggplot()` function to specify our dataset and aesthetic mappings `aes()`. Subsequently, we use `stat_qq()` to generate the Q-Q plot and `stat_qq_line()` to add the reference line:

```
ggplot(tb,
       aes(sample = mpg)) +
  stat_qq() +
  stat_qq_line() +
  labs(x = "Theoretical Quantiles", y = "Sample Quantiles", title = "Q-Q Plot of Miles Per
  theme(panel.background = element_rect(fill = "skyblue"))
```

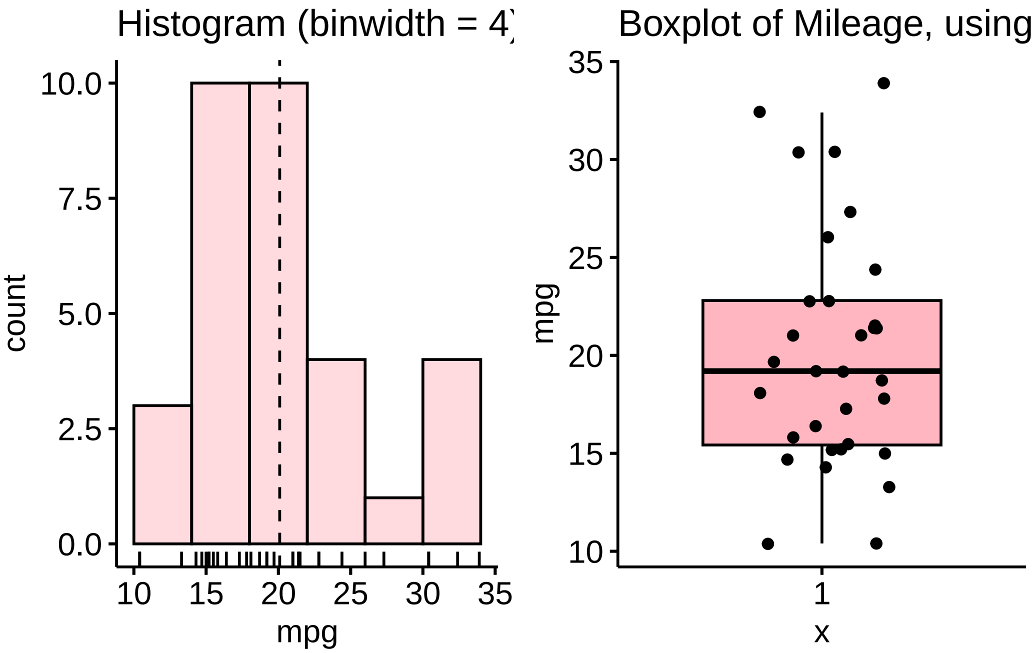


Combine Plots efficiently using ggarrange()

```
library(ggplot2)
library(ggpubr)
PlotHist <- gghistogram(tb,
            x = "mpg",
            binwidth = 4,
            add = "mean",
            rug = TRUE,
            color = "black" ,
            fill = "lightpink",
            title = "Histogram (binwidth = 4) "
)
PlotBox <- ggboxplot(tb,
            y = "mpg",
            rug = TRUE,
            color = "black" ,
            fill = "lightpink",
            add = "jitter",
            title = "Boxplot of Mileage, using ggpubr"
)
```

```
# Combine the plots using ggarrange()
combined_plot <- ggarrange(PlotHist, PlotBox, ncol = 2)

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



Summary of Chapter 13 – Continuous Data (2 of 6)

In this chapter, we explore how to visualize univariate continuous data using the ggplot2 package in R. We use the mtcars data set, converting it to a tibble called tb for easier manipulation.

The visualization methods we cover include histograms, density plots (Probability Density Function and Cumulative Density Function), box plots, bee swarm plots, violin plots, and Q-Q plots. These are created using functions like geom_histogram(), geom_density(), geom_boxplot(), geom_beeswarm(), geom_violin(), stat_qq(), and stat_qq_line().

For the histogram, we can adjust bin width, color, and number of bins, or define custom bin ranges. The density plots provide a visual representation of the distribution of a variable, and we can color the area under the curve. To create the CDF plot, we first arrange our data and calculate the cumulative distribution, which is plotted as a line graph. For the violin plot, we show how to add a box plot within the violin for additional information. Finally, we explore

Q-Q plots, which compare the quantiles of our data to a theoretical distribution, useful for assessing if the data follows a certain theoretical distribution.

References

[1]

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York.

Henderson, D. R. (1974). Motor Trend Car Road Tests. Motor Trend, 1974. Data retrieved from R mtcars dataset.

Eklund, A. (2020). ggbeeswarm: Categorical Scatter (Violin Point) Plots. R package version 0.6.0. https://CRAN.R-project.org/package=ggbeeswarm

[2]

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