

# Continuous Data (2 of 6)

Aug 4, 2023.

## Exploring Univariate Continuous Data using ggplot2

THIS CHAPTER demonstrates the use of the popular **ggplot2** package to further explore *univariate, continuous* data.

1. In R, we can create these plots for Visualizing Univariate Continuous Data using 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 **ggbeeswarm** package can be used for creating bee swarm plots with the `geom_beeswarm()` function. We will go into more detail about how to use these functions and interpret these plots in the upcoming sections.
2. **Data:** Let us work with the same **mtcars** data from the previous chapter. Suppose we run the following code to prepare the data for subsequent analysis. The data is now in a tibble called **tb**:

```
# 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)
attach(tb)
# Convert several numeric columns into factor variables
tb$cyl <- as.factor(tb$cyl)
tb$vs <- as.factor(tb$vs)
tb$am <- as.factor(tb$am)
tb$gear <- as.factor(tb$gear)
```

3. Let us load the **ggplot2**, **dplyr** and **ggthemes** packages. The package **ggthemes** allows us to use a variety of themes.

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

Attaching package: 'ggplot2'

The following object is masked from 'tb':

mpg

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

Histograms using ggplot2;

PDF and CDF Density plots using ggplot2;

Box plots using ggplot2;

Bee Swarm plots using ggbeeswarm;

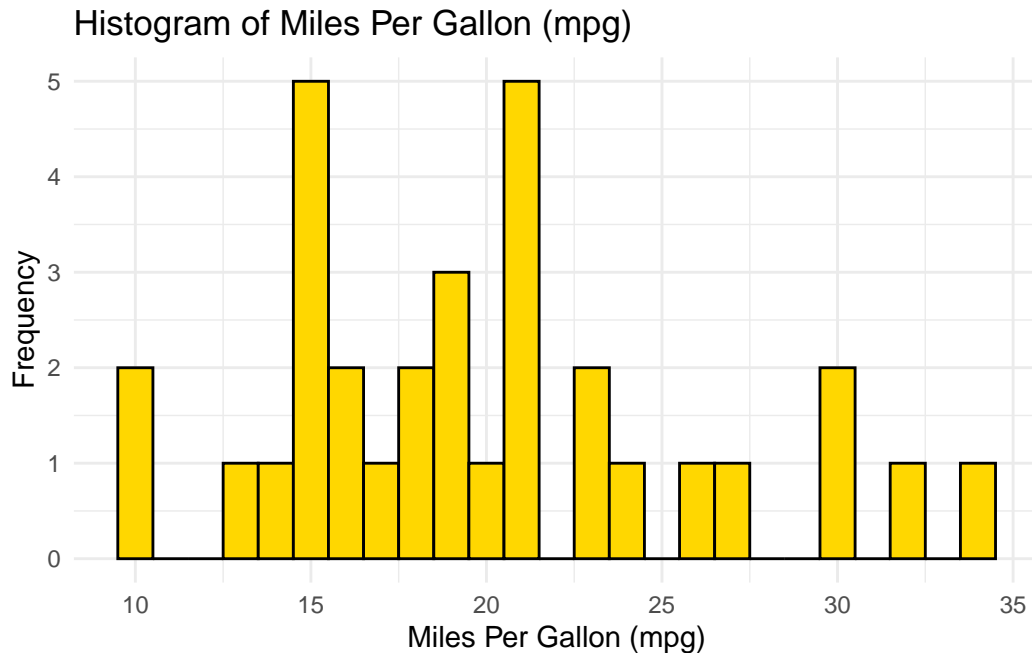
Violin plots using ggplot2;

Q-Q plots. Note that it is inconvenient to create Stem-and-Leaf plots using ggplot2 and hence we ignore this.

## Histogram using ggplot2

1. The following code creates a histogram using the ggplot2 package:

```
ggplot(tb,
       aes(x = mpg)) +
  geom_histogram(binwidth = 1,
                 fill = "gold",
                 color = "black") +
  theme_minimal() +
  labs(title = "Histogram of Miles Per Gallon (mpg)",
       x = "Miles Per Gallon (mpg)", y = "Frequency")
```



## 2. Discussion:

- First, we specify our data set and the aesthetic mappings with the `ggplot()` function. The x aesthetic, which stands for the variable that we are binning (`mpg`), is needed for a histogram.
- Next, we use the `geom_histogram()` function to draw the histogram. The `binwidth` argument specifies the width of the bins in the histogram, and we have chosen 1 as an arbitrary width. The `fill` color of the bars and border are set to gold and black respectively.
- We apply a minimalist theme to our plot with `theme_minimal()`, giving it a clean and polished look.
- Lastly, the `labs()` function allows us to add a title to our plot and labels for our x and y axes. [1]

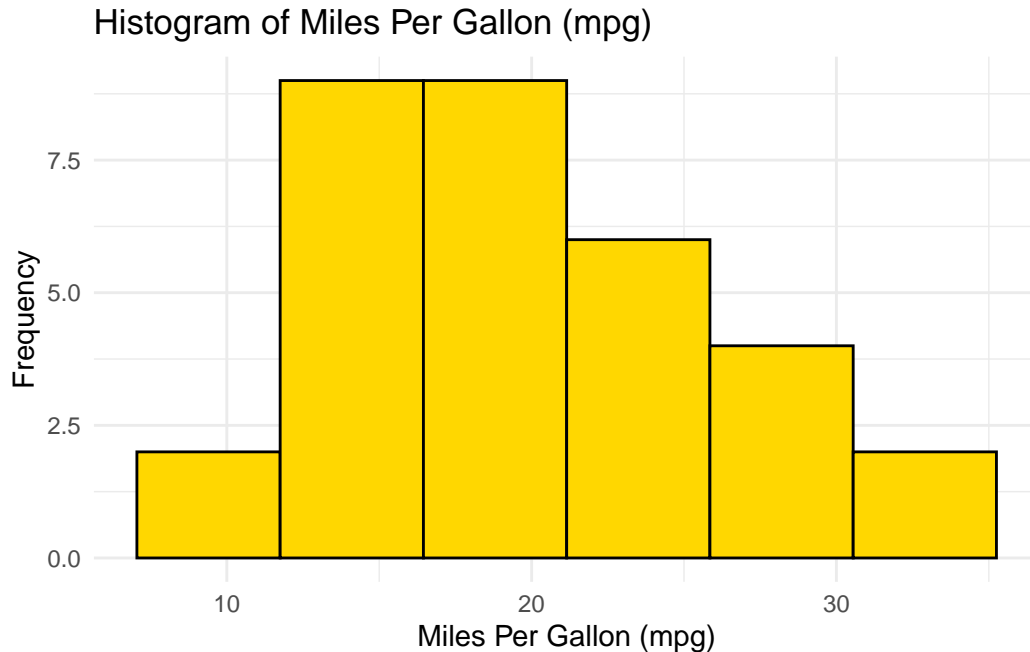
3. We can set the number of bins in the histogram using the `bins` argument within the `geom_histogram()` function instead of `binwidth`. For example, in order to create a histogram with 12 bins of equal width, we can modify the code as follows:

```
ggplot(tb,
       aes(x = mpg)) +
  geom_histogram(bins = 6,
                 fill = "gold",
```

```

    color = "black") +
  theme_minimal() +
  labs(title = "Histogram of Miles Per Gallon (mpg)", x = "Miles Per Gallon (mpg)", y = "F

```



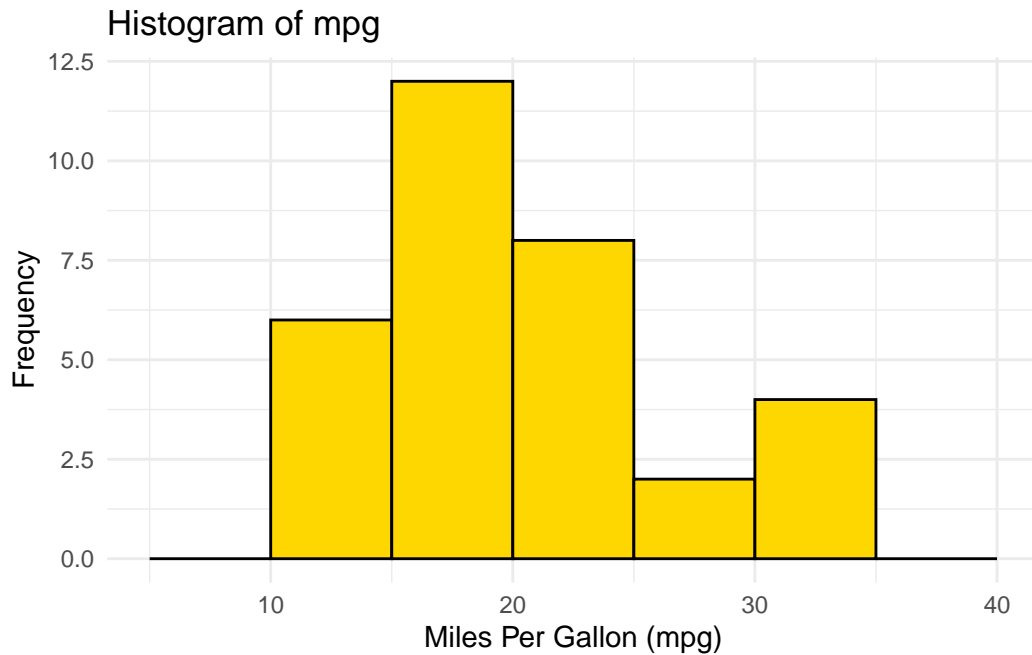
#### 4. Discussion:

- By replacing `binwidth = 2` with `bins = 12`, we instruct R to create 12 bins of equal width. The `bins` argument decides the number of bins in the histogram, and the width of the bins is calculated by dividing the range of the data by the number of bins.
- 5. We can specify custom bin ranges in a histogram. We typically use the `breaks` argument within the `geom_histogram()` function. We need to supply a vector of breakpoints which defines the range of each bin. For example, if we wanted to define bins with ranges of 5-10, 10-15, 15-20, 20-25, 25-30, 30-35, 35-40, for the `mpg` variable, we could write the following code:

```

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

```

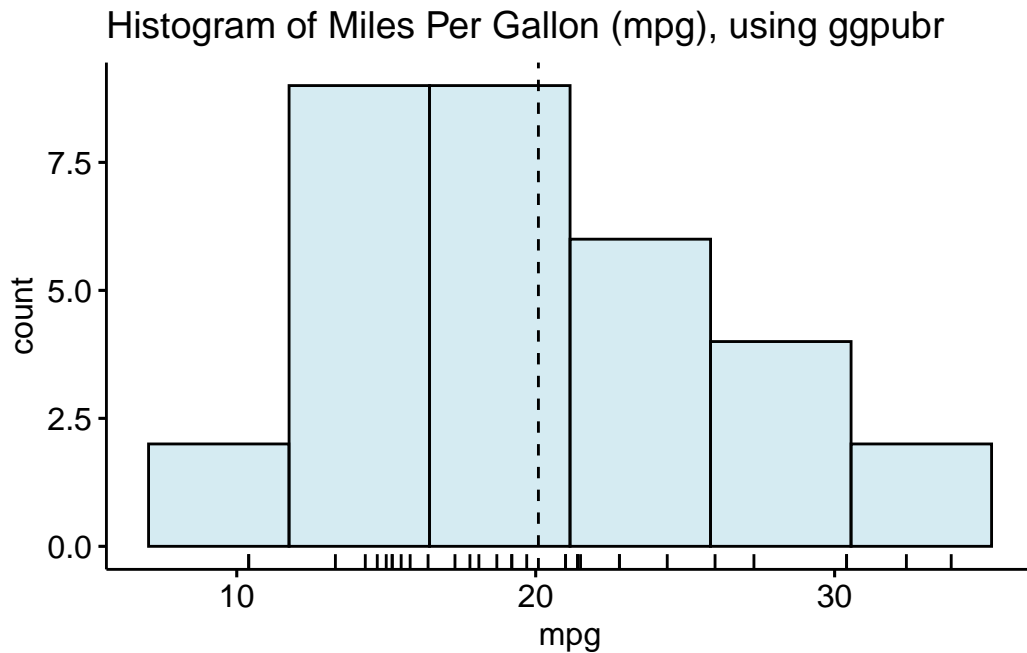


- In this code, the `seq()` function is used to generate a sequence of numbers from 5 to 40, with an increment of 5. These numbers serve as the breakpoints for the bins in the histogram. The `breaks` argument then uses these numbers to set the bin ranges. The plot aesthetics remain the same as before.

### Histogram using ggpubr

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

```
library(ggpubr)
gghistogram(tb,
  x = "mpg",
  add = "mean",
  bins = 6,
  rug = TRUE,
  color = "black" ,
  fill = "lightblue",
  title = "Histogram of Miles Per Gallon (mpg), using ggpubr"
)
```



Discussion:

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

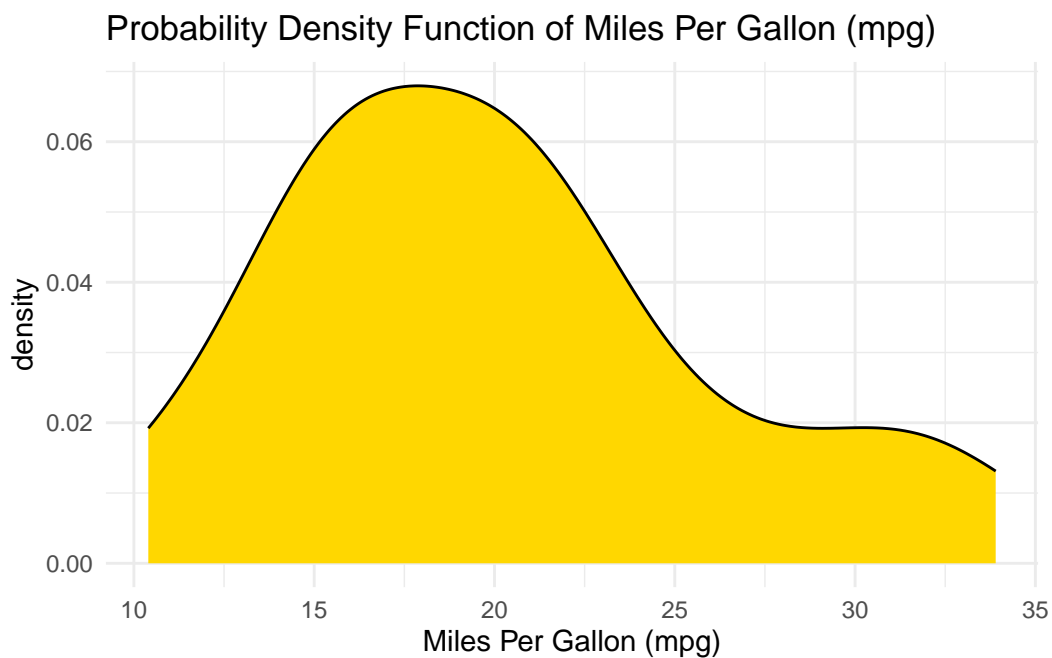
This line uses the `gghistogram()` function to create a histogram of the `mpg` variable from the `tb` dataset.

- `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 = 6`: 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")
```

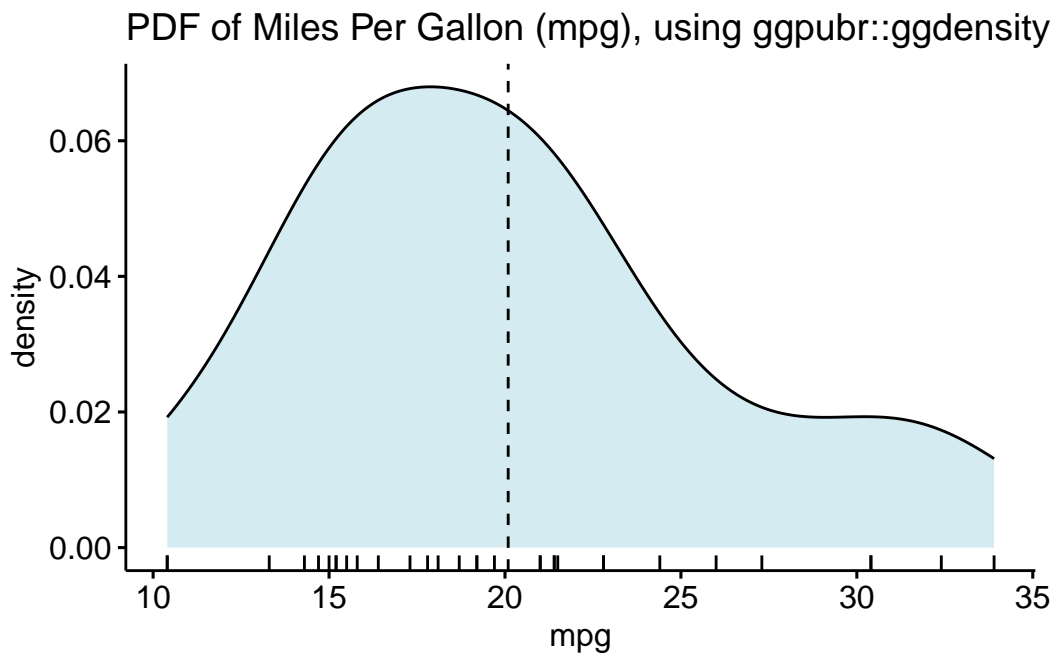


- We designate our data source and the aesthetic mappings using the `ggplot()` function. The aesthetic mapping for x is `mpg`.
- Subsequently, we append a density plot to our plot by using the `geom_density()` function. We fill the area under the curve with a light pink color by setting `fill` to “lightpink” 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.

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

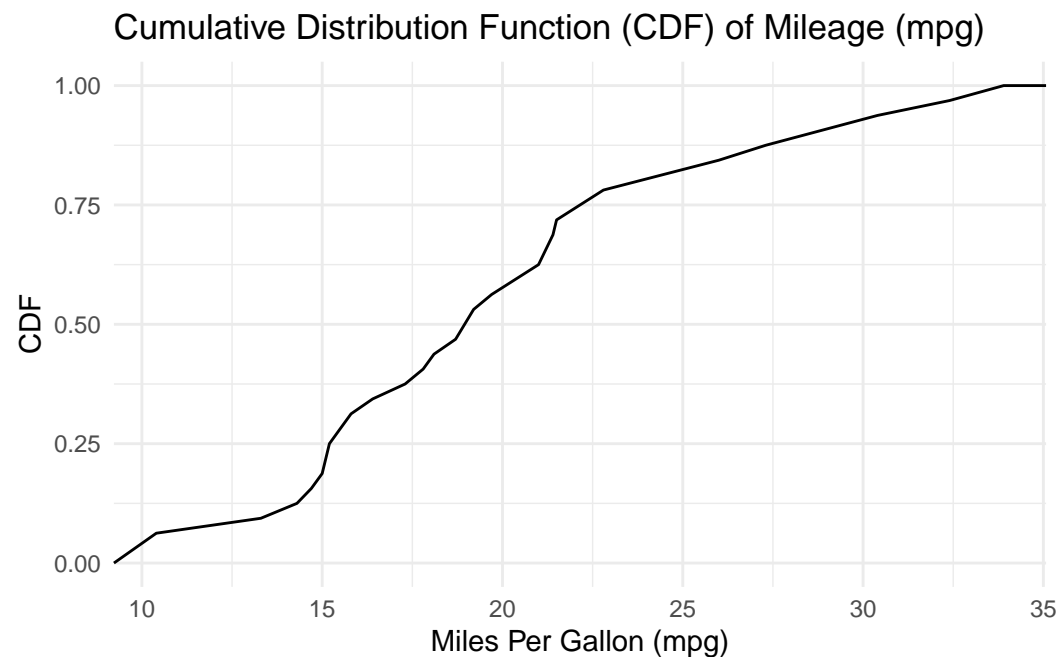


## Cumulative Distribution Function (CDF) Plot using ggplot2

```
# Load required library
library(ggplot2)
# Create a CDF plot
ggplot(tb, aes(x = mpg)) +
```

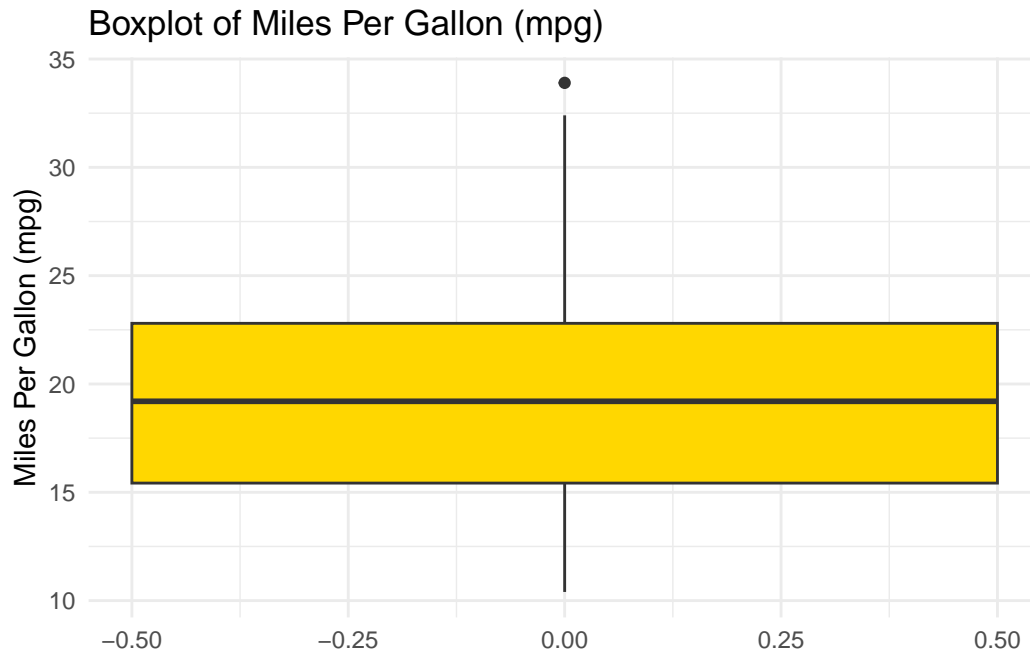


```
stat_ecdf(geom = "line", color = "black") +
labs(x = "Miles Per Gallon (mpg)", y = "CDF",
     title = "Cumulative Distribution Function (CDF) of Mileage (mpg)") +
theme_minimal()
```



## Boxplots using ggplot2

```
ggplot(tb,
       aes(y = mpg)) +
geom_boxplot(fill = "gold",
             width = 1) +
theme_minimal() +
labs(title = "Boxplot of Miles Per Gallon (mpg)", y = "Miles Per Gallon (mpg)")
```



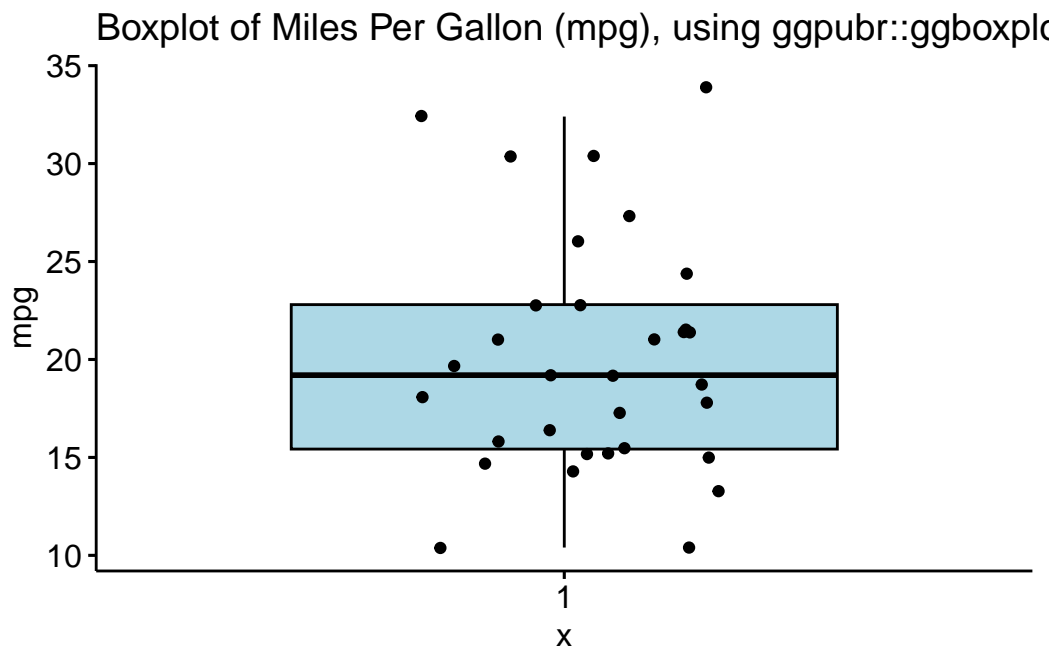
1. We're first specifying the source of our data and the aesthetic mappings, which define how variables in the data are mapped to visual properties. In this case, we're only specifying the y aesthetic, since a boxplot of a single variable doesn't need an x aesthetic. The y aesthetic is mapped to the 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 = "lightblue",
  add = "jitter",
```

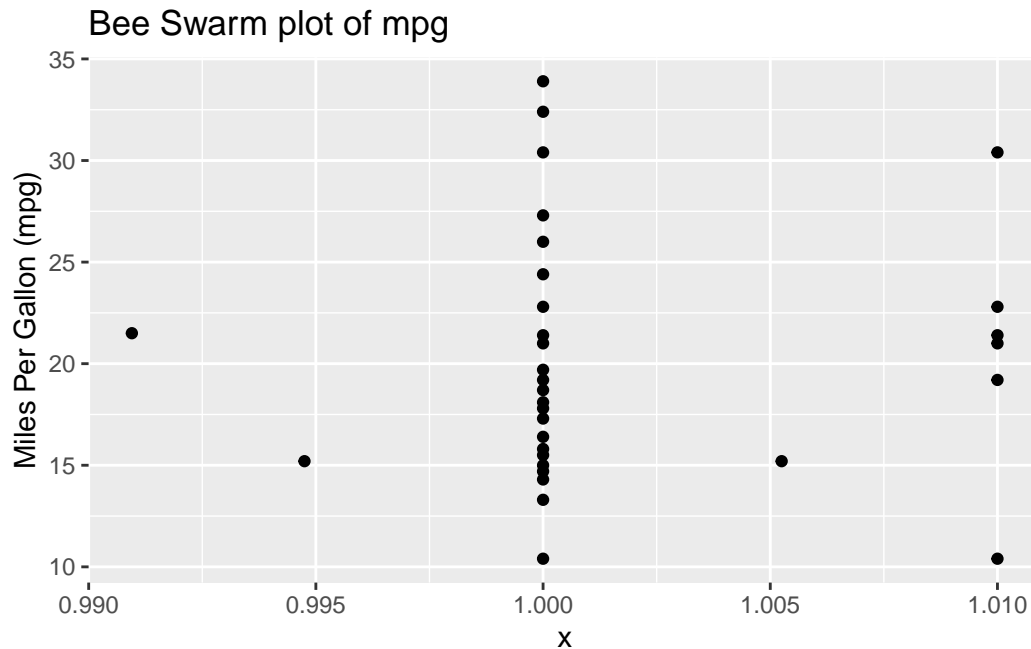
```
)
  title = "Boxplot of Miles Per Gallon (mpg), using ggpubr::ggboxplot()"
)
```



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



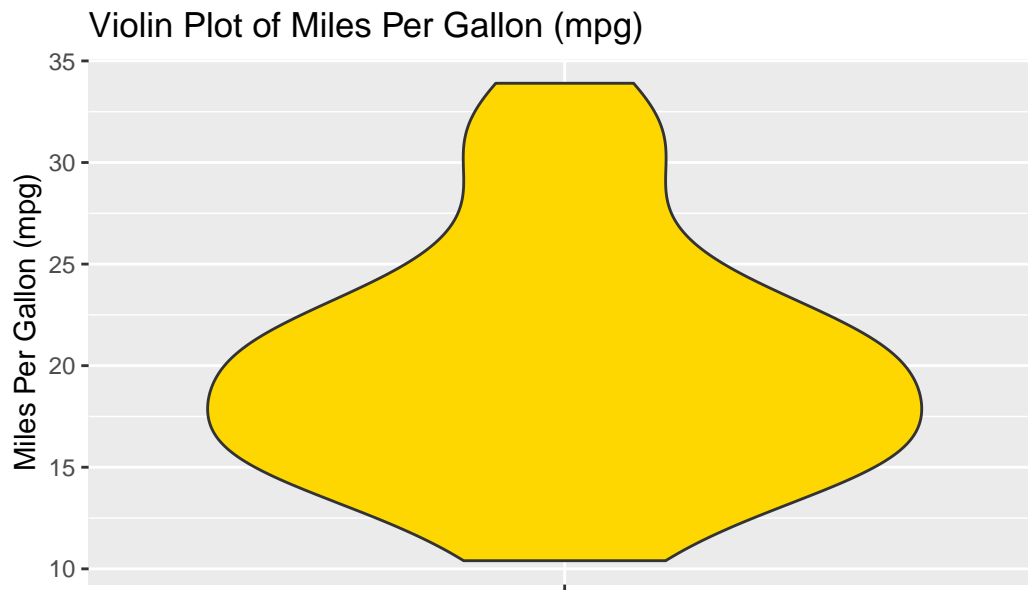
### 3. Discussion:

- 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 then adopt a minimalist theme by using `theme_minimal()` to give our plot a sleek and simple look.
- We use the `labs()` function to label the plot.

## Violin plot using ggplot2

- 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)")
```

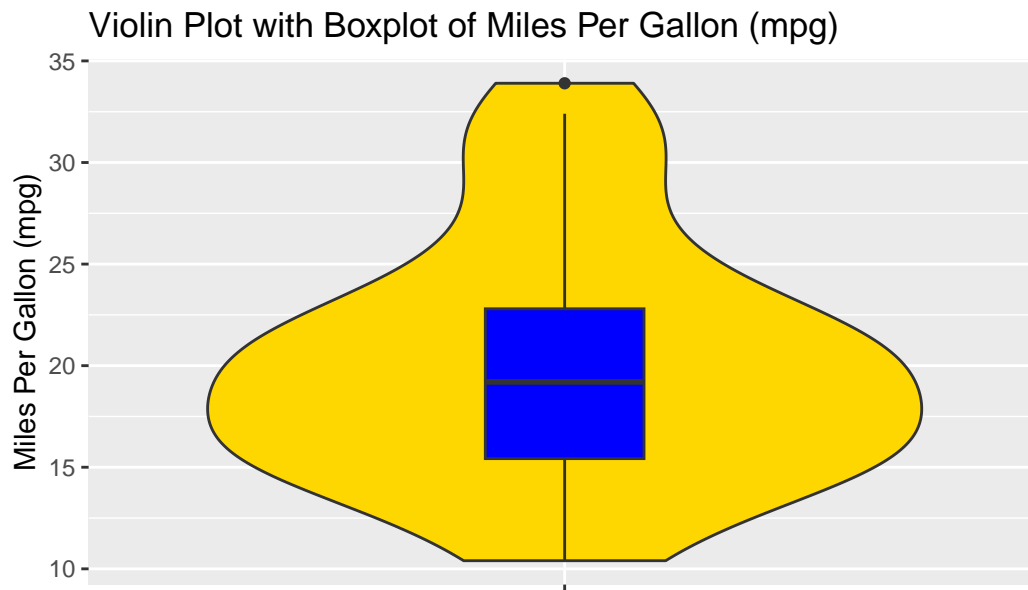


2. Discussion:

- 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 = "blue", width = 0.2) +  
  labs(x = "", y = "Miles Per Gallon (mpg)", title = "Violin Plot with Boxplot of Miles Per Gallon (mpg)")
```

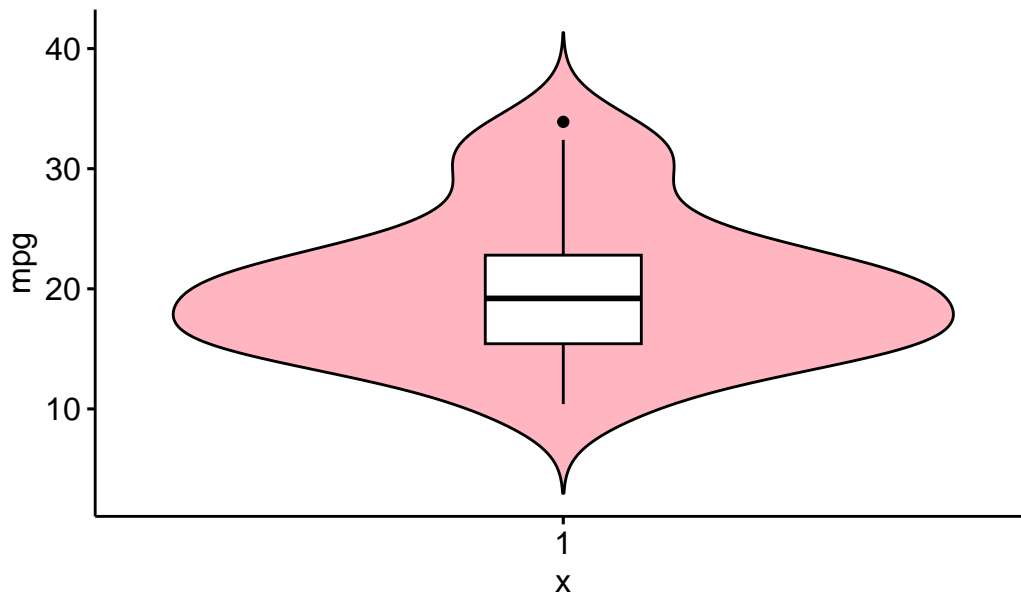


#### Violin plot using ggpubr

- 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.

```
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()"
)
```

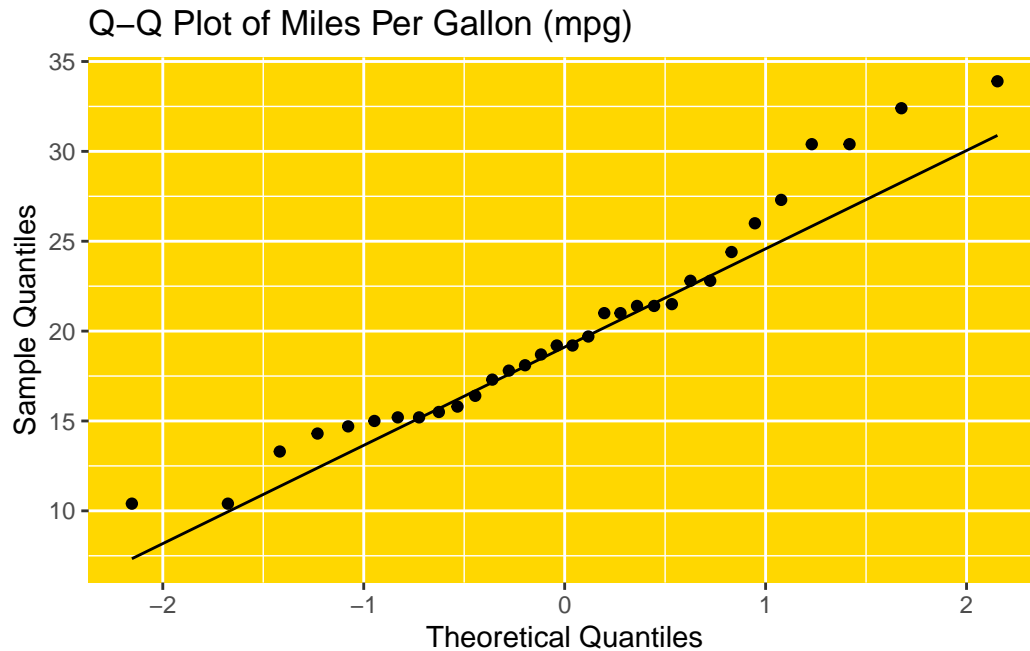
Violin plot of Miles Per Gallon (mpg), using `ggpubr::ggviolin`



### 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 = "gold"))
```



## 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/>.