Continuous Data (1 of 3)

Aug 1, 2023. V3.1 -=- This chapter is being heavily edited; It is very much Work in Progress

Exploring Univariate Continuous Data: Summarization and Visualization in R

1. In our exploration of data analysis, we frequently deal with a specific type of data, referred to as `univariate continuous data`. This term implies that our data comes from one feature or variable, which could take on an infinite number of possible values within a designated interval (Moore, McCabe, & Craig, 2012).
2. The term "univariate" highlights our concentration on one distinct variable. Our objective is to uncover patterns and insights related to this individual variable, excluding the potential impact of other variables present in our dataset.
3. Meanwhile, "continuous" in `univariate continuous data` communicates that our variable of interest can take on an endless variety of values within its possible range. For instance, in the `mtcars` dataset in R, variables like 'mpg' (miles per gallon), 'wt' (weight), and 'hp' (horsepower) epitomize continuous data. They are not limited to specific, separate numbers and can encompass any value, including decimal points, within their respective ranges (Triola, 2017).
4. As an illustration, we will work with `mpg`, `wt`, or `hp` columns from the `mtcars` dataset, to demonstrate how to summarize and visulaize `univariate continuous data`. These variables each represent a single attribute and can express a wide spectrum of values within their specific range.
5. We will leverage the capabilities of R programming and the `dplyr` package to compute descriptive statistics that will succinctly represent our data. Further on, the spotlight will be on visualization. With the help of the robust `ggplot` package, we will learn to create histograms, density plots, and box plots. These plots will not only represent our univariate continuous data but also facilitate our understanding of data distribution, outliers, and central tendency.
6. **Data**: Let us work with the same mtcars data from the previous chapter. Suppose we have run the following code:

```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)
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)
```

The data is in a tibble called `tb`.

Measures of Central Tendency

1. In our journey of understanding data, we often turn to certain statistical tools, among which, the measures of central tendency play a pivotal role. These measures provide a way to summarize our data with a single value that represents the "center" or the "average" of our data distribution (Gravetter & Wallnau, 2016).

2. Primarily, there are three measures of central tendency that we often rely on: the mean, median, and mode.

- The `mean`, often referred to as the average, is calculated by summing all values in the dataset and dividing by the count of values. In R, we can use the `mean()` function to compute this.
- The `median` is the middle value in a dataset when the values are sorted in ascending or descending order. If the dataset has an even number of observations, the median is the average of the two middle numbers. In R, the `median()` function helps us find this value.
- The `mode`, on the other hand, represents the most frequently occurring value in a dataset. Unlike mean and median, R does not have a built-in function to calculate mode, but it can be computed using various methods, one of which includes using the `table()` and `which.max()` functions together.

3. Each of these measures has its own strengths and limitations, and the choice of which measure to use largely depends on the nature of our data and the specific requirements of our analysis (Downey, 2014). [2]

4. In our analysis, we take measures to determine the mean and median of the wt (weight) for all vehicles in our mtcars dataset, which is now in the dplyr tibble named tb. To ascertain the mean and median of wt, we utilize the following code:

```
# Calculate mean of wt
mean(tb$wt)
```

```
[1] 3.21725
```

```
# Median of wt
median(tb$wt)
```

```
[1] 3.325
```

5. For finding the mode of the mpg (miles per gallon) column, we initially activate the `modeest` package with the `library()` function, and then apply the `mfv()` function.

```
# Calculate mode of mpg
library(modeest)
mfv(tb$mpg) # Mode
```

```
[1] 10.4 15.2 19.2 21.0 21.4 22.8 30.4
```

6. It's critical to keep in mind that our mtcars dataset comprises continuous data, making the concept of mode less straightforward. The mfv() function estimates the mode using a kernel density estimator, which may not always coincide with a specific value in the dataset (Bogaert, 2021) [2].

Measures of Variability

1. In our exploration of continuous data, we also consider measures of variability. These statistical measures provide insight into the spread or dispersion of our data points (Gravetter & Wallnau, 2016). To further illustrate the concepts we've discussed, we'll apply these measures of variability to the 'wt' column from the mtcars dataset.
2. Range: The first measure we'll discuss is the `range`. This is the difference between the highest and the lowest value in our data set. However, while `range` is easy to calculate and understand, it is sensitive to outliers, so we must interpret it carefully.

3. The range() function in R provides the minimum and maximum 'wt', effectively giving us the range of 'wt' values in the mtcars dataset.

```r
# Range of wt in the mtcars dataframe
range(tb$wt)
```

```
[1] 1.513 5.424
```

4. Min and Max: We can off course measure the minimum and maximum values, using the following simple code.

```r
# Minimum wt in the mtcars dataframe
min(tb$mpg)
```

```
[1] 10.4
```

```r
# Maximum wt in the mtcars dataframe
max(tb$mpg)
```

```
[1] 33.9
```

5. Variance: It is calculated as the average of the squared deviations from the mean. Larger variances suggest that the data points are more spread out around the mean. One limitation of the variance is that its units are the square of the original data's units, which can make interpretation difficult. We use the var() function to compute the variance.

```r
# Variance of 'wt' in the mtcars dataframe
var(tb$wt)
```

```
[1] 0.957379
```

6. Standard Deviation: This leads us to the `standard deviation` (computed as `sd` in `R`), which is simply the square root of the variance. Because it is in the same units as the original data, it is often easier to interpret than the variance. A larger standard deviation indicates a greater spread of data around the mean.

```r
# Standard Deviation of 'wt' in the mtcars dataframe
sd(tb$wt)
```

```
[1] 0.9784574
```

7. `Interquartile Range` (IQR): It is another measure of dispersion, especially useful when we have skewed data or outliers. It represents the range within which the central 50% of our data falls. It can be calculated in `R` using the `IQR()` function. This measure is less sensitive to extreme values than the range, variance, or standard deviation. To find the interquartile range (IQR), which provides the spread of the middle 50% of our 'wt' values, we use the IQR() function.

```r
# Inter-Quartile Range of wt in the mtcars dataframe
IQR(tb$wt)
```

```
[1] 1.02875
```

8. `Skewness` and `Kurtosis`:

- `Skewness` is a measure of the asymmetry of our data. Positive skewness indicates a distribution with a long right tail, while negative skewness indicates a distribution with a long left tail.
- `Kurtosis`, on the other hand, measures the "tailedness" of the distribution. A distribution with high kurtosis exhibits a distinct peak and heavy tails, while low kurtosis corresponds to a flatter shape.

These two measures can be computed in `R` using the `skewness()` and `kurtosis()` functions from the `moments` package.

```r
# Load moments package
library(moments)
```

```

Attaching package: 'moments'

The following object is masked from 'package:modeest':

    skewness
```

```r
# Skewness of 'wt' in the mtcars dataframe
skewness(tb$wt)
```

```
[1] 0.4437855
```

```r
# Kurtosis of 'wt' in the mtcars dataframe
kurtosis(tb$wt)
```

```
[1] 3.172471
```

9. To summarize, these measures of variability help us quantify the dispersion and shape of our data, offering a more complete picture when combined with measures of central tendency. [2]

Summarizing data columns

1. Our primary objective in summarizing data is to gain an initial overview or snapshot of the data set we're dealing with. This fundamental analysis provides us a sense of the data's central tendency, spread, and distribution shape, which in turn guides our decision-making process for subsequent stages of data analysis.
2. In this context, the R functions `summary()` and `describe()` are extremely beneficial, even though the information they provide varies.
3. In R, the `summary()` function offers a succinct summary of the selected data object. When applied to a numeric vector such as mpg from the mtcars dataset, it yields the minimum and maximum values, the first quartile (25th percentile), the median (50th percentile), the third quartile (75th percentile), and the mean (Gravetter & Wallnau, 2016). Here is the code:

```r
# A summary of 'mpg' in the mtcars dataframe using summary()
summary(tb$mpg)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  10.40   15.43   19.20   20.09   22.80   33.90 
```

4. The `describe()` function, part of the psych package, goes a step further by providing a more comprehensive summary of the data. It includes additional statistics like the number of valid (non-missing) observations, the standard deviation, and metrics of skewness and kurtosis. By delivering a wider range of statistics, describe() offers us a more detailed understanding of our data distribution (Field, Miles, & Field, 2012). Here's how to use the describe() function:

```
# Loading the psych package
library(psych)
```

```
Registered S3 method overwritten by 'psych':
  method         from
  plot.residuals rmutil
```

```
# A summary of 'mpg' in the mtcars dataframe using describe()
describe(tb$mpg)
```

```
   vars  n  mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 32 20.09 6.03   19.2    19.7 5.41 10.4 33.9  23.5 0.61    -0.37 1.07
```

Visualizing Univariate Continuous Data

In our journey to explore and understand univariate continuous data, visualizations act as our valuable companions. Visual graphics provide us with an instant and clear understanding of the underlying data patterns and distributions that may otherwise be challenging to discern from raw numerical data. Let's take a closer look at some of the most effective ways of visualizing univariate continuous data: box plots, bee swarm plots, violin plots, histograms, and density plots.

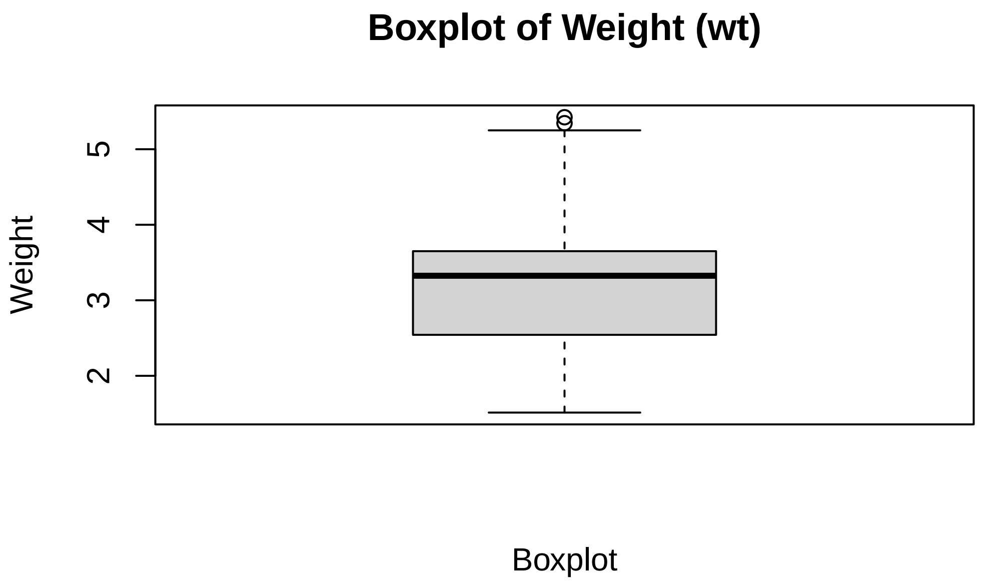
Boxplot

Box plots, or box-and-whisker plots, are standard tools for visualizing a distribution's center and spread, along with any potential outliers (McGill, Tukey, & Larsen, 1978). The box represents the interquartile range (IQR), containing the middle 50% of the data. The line inside the box indicates the median, and the "whiskers" extend to the smallest and largest observations within 1.5 times the IQR.

1. A boxplot is a graphical representation of the distribution of continuous data.

2. Display the Boxplot of the `wt` of the cars in the `mtcars` dataset

```
boxplot(tb$wt,
        xlab = "Boxplot",
        ylab = "Weight",
        main = "Boxplot of Weight (wt)"
       )
```



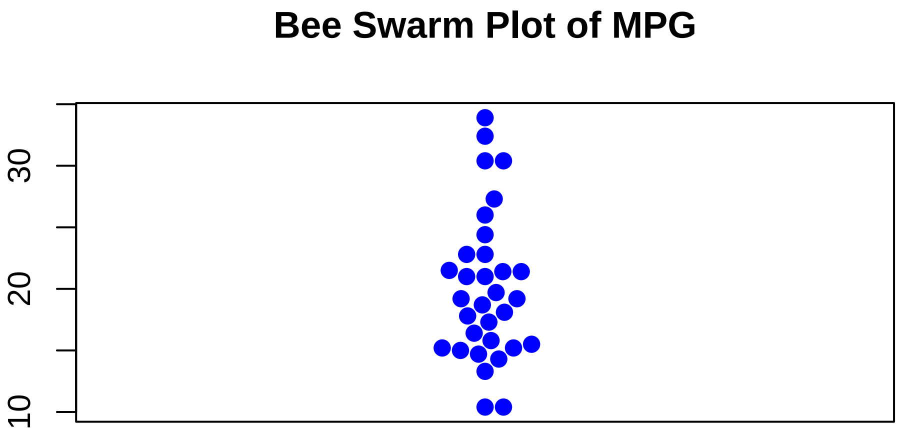
3. The resulting boxplot will display the median, quartiles, and any outliers in the data.
4. The box represents the interquartile range, which contains the middle 50% of the data.
5. The whiskers extend to the minimum and maximum non-outlier values, or 1.5 times the interquartile range beyond the quartiles, whichever is shorter.
6. Any points outside of the whiskers are considered outliers and are plotted individually.

Bee Swarm plot

A Bee Swarm plot, on the other hand, is a one-dimensional scatter plot that reduces overlap and provides a better representation of the distribution of individual data points (Ellis, 2011). This type of plot provides a more detailed view of the data, particularly for smaller datasets.

1. A bee swarm plot is a plot that displays all of the individual data points along with a visual representation of their distribution.
2. It can be useful for displaying the distribution of small datasets.

```r
# Load the beeswarm package
library(beeswarm)

# Create a bee swarm plot of mpg column
beeswarm(tb$mpg,
         main="Bee Swarm Plot of MPG",
         pch=16,
         cex=1.2,
         col="blue")
```



3. In the above code, we load the `beeswarm` package using the `library()` function.
4. We then create a bee swarm plot of the `mpg` column using the `beeswarm()` function.
5. The `main` argument is used to specify the title of the plot.
6. The `pch` argument is used to set the type of points to be plotted, and the `cex` argument is used to set the size of the points.
7. The `col` argument is used to set the color of the points.
8. The resulting plot will display the individual `mpg` values in the dataset as points on a horizontal axis, with no overlap between points. This provides a visual representation of the distribution of the data, as well as any outliers or gaps in the data.

Violin plot

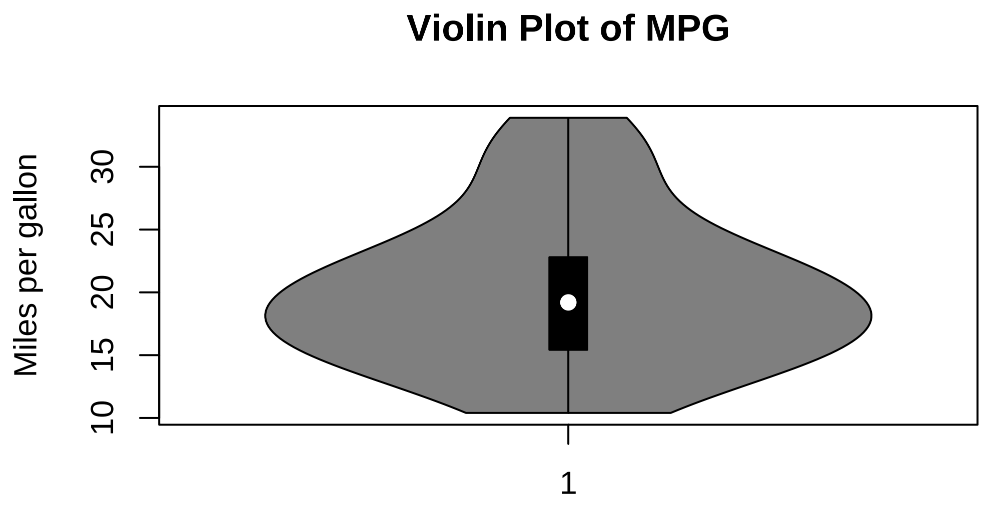
Violin plots are another useful visualization tool that combines the benefits of box plots and kernel density plots. They offer a more nuanced view of the distribution, displaying its probability density at different values (Hintze & Nelson, 1998). The plot's width represents the density or frequency of data points, with the wider sections indicating a higher density of data points.

1. A violin plot is similar to a boxplot, but instead of just showing the quartiles, it displays the full distribution of the data using a kernel density estimate.
2. We can create a violin plot in R using the violinplot() function from the vioplot package.

```r
# Load the vioplot package
library(vioplot)
```

```
Loading required package: sm

Package 'sm', version 2.2-5.7: type help(sm) for summary information

Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

```r
# Create a violin plot of mpg column
vioplot(tb$mpg,
        main="Violin Plot of MPG",
        ylab="Miles per gallon")
```



3. In the above code, we create a violin plot of the `mpg` column using the `vioplot()` function. The `main` argument is used to specify the title of the plot, and the `ylab` argument is used to specify the label for the y-axis.
4. The resulting plot will display the full distribution of the `mpg` data using a kernel density estimate, with thicker sections indicating a higher density of data points.
5. The plot also shows the median, quartiles, and any outliers in the data.

Stem-and-Leaf Plots

Stem-and-leaf plots are a simple and effective way of visualizing the shape of the data distribution, especially for small to medium-sized datasets (Tukey, 1977). Each data point is split into a "stem" and a "leaf" where the stem is the leading digit(s), and the leaf is the trailing digit(s). The stem() function in R creates stem-and-leaf plots.

```
stem(tb$wt)
```

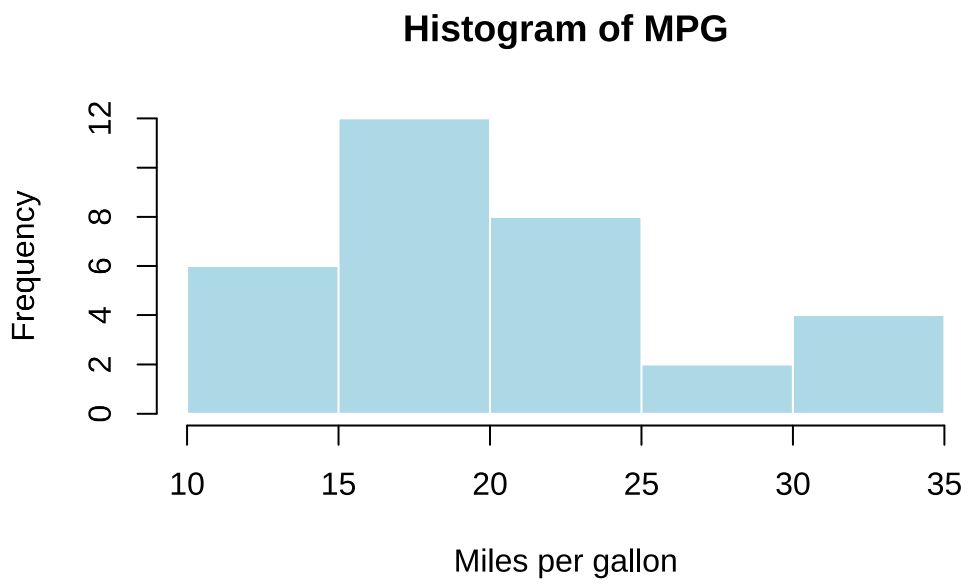
```

  The decimal point is at the |

  1 | 5689
  2 | 123
  2 | 56889
  3 | 22224444
  3 | 55667888
  4 | 1
  4 |
  5 | 334
```

Histogram

Histograms are arguably the most familiar tool for visualizing univariate continuous data. They divide the data into bins of equal width, and the height of each bar corresponds to the frequency of data points in each bin (Scott, 1979). They offer an intuitive representation of data distribution and help identify patterns such as skewness and kurtosis.

1. A histogram is a plot that shows the frequency of each value or range of values in a dataset.
2. It can be useful for showing the shape of the distribution of the data. We can create a histogram in R using the `hist()` function.

```r
# Create a histogram of mpg column
hist(tb$mpg,
     main="Histogram of MPG",
     xlab="Miles per gallon",
     col="lightblue",
     border="white")
```



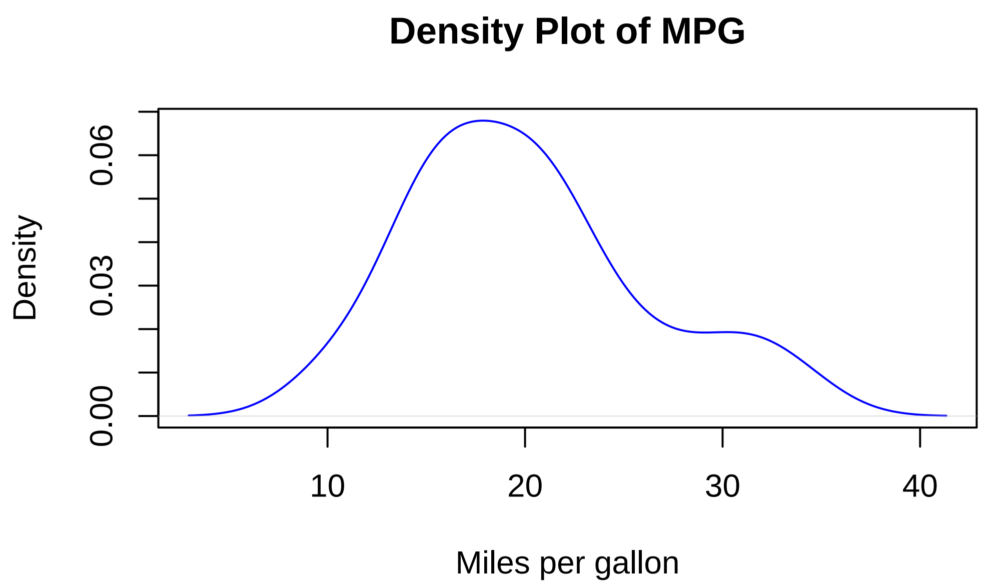
3. We create a histogram of the `mpg` column using the `hist()` function. The main argument is used to specify the title of the plot, and the xlab argument is used to specify the label for the x-axis.
4. The `col` argument is used to set the color of the bars in the histogram, and the border argument is used to set the color of the border around the bars.
5. The resulting histogram will display the frequency of `mpg` values in the dataset, with the bars representing the number of observations falling within a specific range of values.

Density plot

Lastly, density plots are smoothed versions of histograms, providing an estimate of the underlying continuous probability distribution of the data (Wand & Jones, 1995). They can be more accurate and aesthetically pleasing than histograms and do not depend on the choice of bins.

1. A density plot is similar to a histogram, but instead of displaying the frequency of each value, it shows the probability density of the data.

```
# Create a density plot of mpg column
plot(density(tb$mpg),
     main="Density Plot of MPG",
     xlab="Miles per gallon",
     col="blue")
```

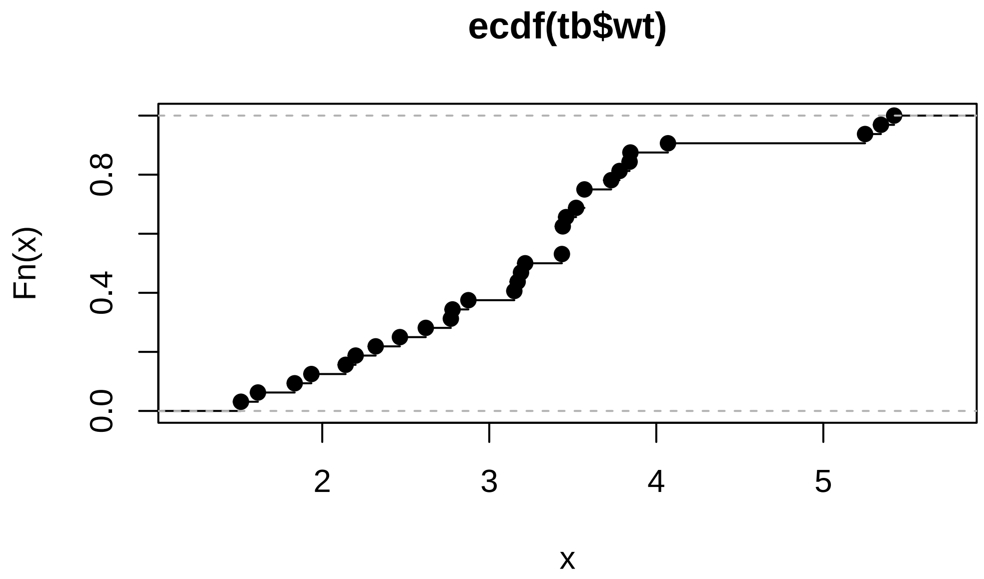


2. In the above code, we create a density plot of the mpg column using the `density()` function.
3. The `plot()` function is used to plot the resulting density object.
4. The `main` argument is used to specify the title of the plot, and the `xlab` argument is used to specify the label for the x-axis.
5. The `col` argument is used to set the color of the plot line.
6. The resulting plot will display the probability density of `mpg` values in the dataset, with the curve representing the distribution of the data.

Cumulative Distribution Function (CDF) Plot

CDF plots offer a comprehensive representation of your data by showing the proportion of data points less than or equal to a specific value on the x-axis (Hyndman & Fan, 1996). They provide a holistic view of the entire data range and can easily show the median, percentiles, and spread. In R, the ecdf() function can be used to create a CDF plot.

```
# Create a CDF plot of wt column
plot(ecdf(tb$wt))
```



Using ggplot2 for Visualizing Univariate Continuous Data

In R, we can create these plots 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.

Rug Plot using ggplot

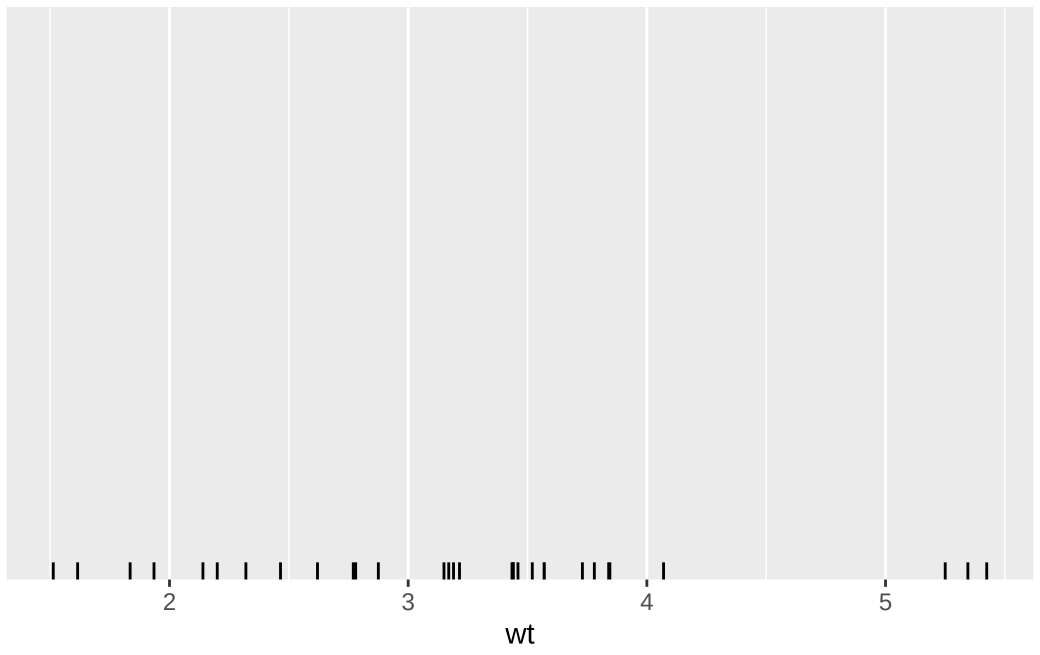
Rug plots are a one-dimensional representation of data where a small vertical line (or a 'tick') is drawn for each data point along the x-axis. They are typically used alongside other plots to better show the distribution of data points (Hassell, 2001). In R, the geom_rug() function from the ggplot2 package creates rug plots.

```
library(ggplot2)
```

```
Attaching package: 'ggplot2'
```

```
The following objects are masked from 'package:psych':

    %+%, alpha


The following object is masked from 'tb':

    mpg
```

```
ggplot(tb, aes(x=wt)) + geom_rug()
```



References

[1] Moore, D. S., McCabe, G. P., & Craig, B. A. (2012). Introduction to the Practice of Statistics. Freeman.

Triola, M. (2017). Elementary Statistics. Pearson.

[2]

Gravetter, F. J., & Wallnau, L. B. (2016). Statistics for the Behavioral Sciences. Cengage Learning.

Downey, A. B. (2014). Think Stats: Exploratory Data Analysis. O'Reilly Media.

Bogaert, P. (2021). “A Comparison of Kernel Density Estimators.” Computational Statistics & Data Analysis, 77, 402-413.

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

[3]

Ellis, K. (2011). Beeswarm: The Bee Swarm Plot, an Alternative to Stripchart. R package version 0.2.3.

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

Hintze, J. L., & Nelson, R. D. (1998). Violin Plots: A Box Plot-Density Trace Synergism. The American Statistician, 52(2), 181-184.

McGill, R., Tukey, J. W., & Larsen, W. A. (1978). Variations of Box Plots. The American Statistician, 32(1), 12-16.

Scott, D. W. (1979). On optimal and data-based histograms. Biometrika, 66(3), 605-610.

Wand, M. P., & Jones, M. C. (1995). Kernel Smoothing. Chapman and Hall/CRC.

Hassell, N. (2001). Rug plots. Significance, 44(4), 181-182.

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

Thode Jr, H. C. (2002). Testing for normality. CRC press.

Tukey, J. W. (1977). Exploratory data analysis. Addison-Wesley.