Continuous Data (1 of 3)

Aug 1, 2023. V2.1 -=- 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
```

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 R, we can calculate measures of variability (range, interquartile range, variance, and standard deviation).
2. To calculate these statistics, we can use built-in functions in R such as `range()`, `IQR()`, `var()`, and `sd()`.

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

```
[1] 0.9784574
```

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

```
[1] 0.957379
```

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

```
[1] 1.513 5.424
```

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

```
[1] 1.02875
```

3. Note that the range() function returns the minimum and maximum values in the dataset, while the IQR() function returns the difference between the 75th and 25th percentiles.

Other functions

```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
```

Summarizing a data column

`summary()`

1. Display a summary of `mpg` in the dataframe mtcars 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
```

`describe()`

2. Display a summary of the `mpg` in the dataframe mtcars using `describe()`

```
library(psych)
```

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

```
describe(tb$mpg)
```

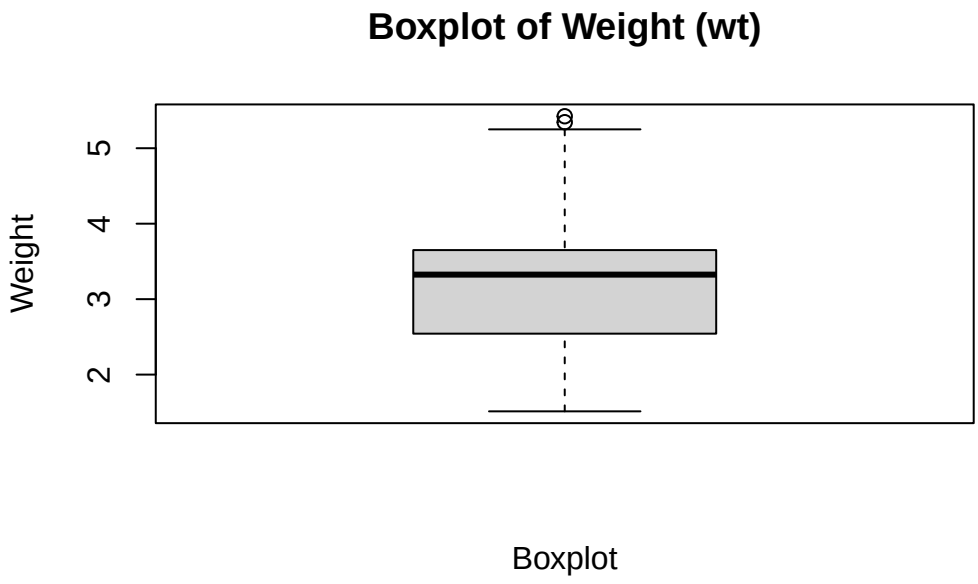
```
   vars  n  mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 32 20.09 6.03   19.2    19.7 5.41 10.4 33.9  23.5 0.61    -0.37 1.07
```

Visualizing Univariate Continuous Data

Boxplot

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

```r
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.

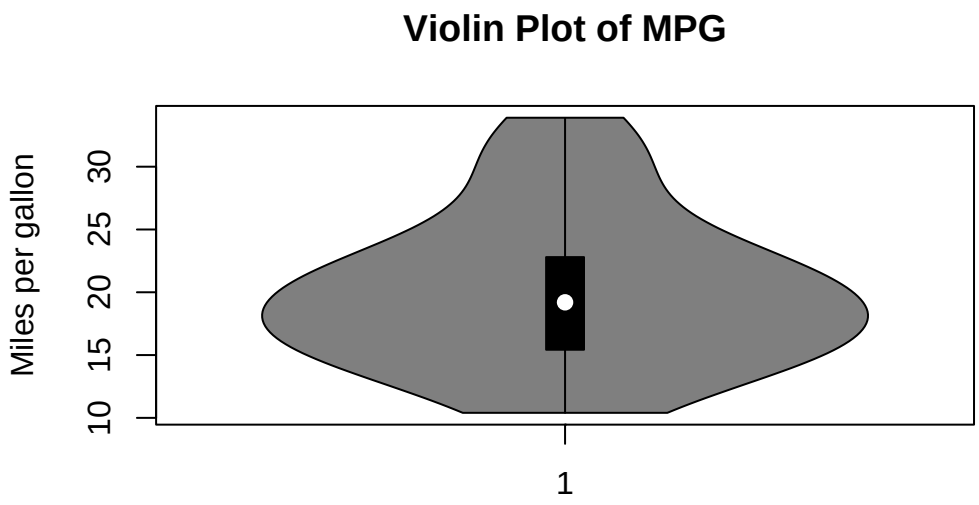
Violin plot

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")
```

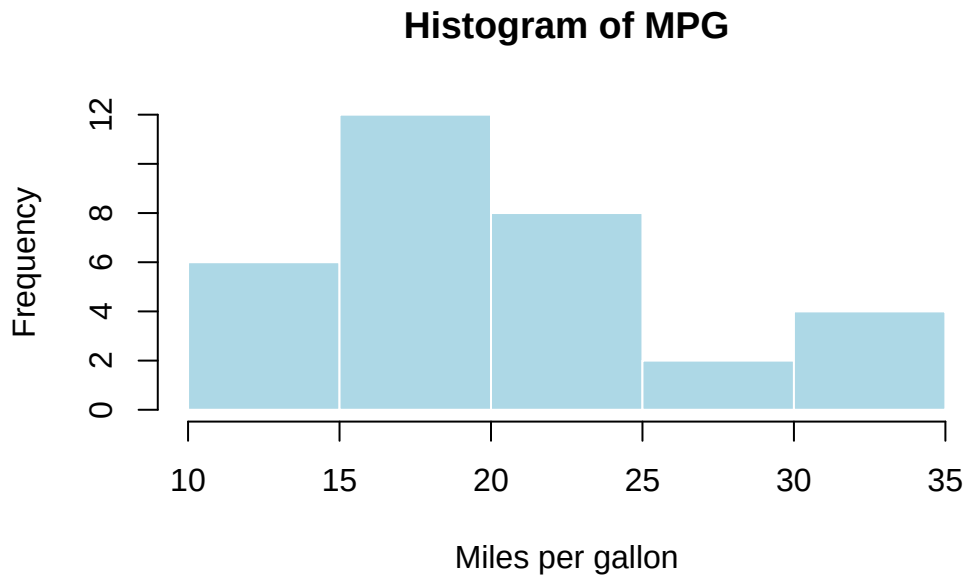


3. In the above code, we create a violin plot of the `mpg` column using the `vioplot()` function. The `main` argument is used to specify the title of the plot, and the `ylab` argument is used to specify the label for the y-axis.
4. The resulting plot will display the full distribution of the `mpg` data 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.

Histogram

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.

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

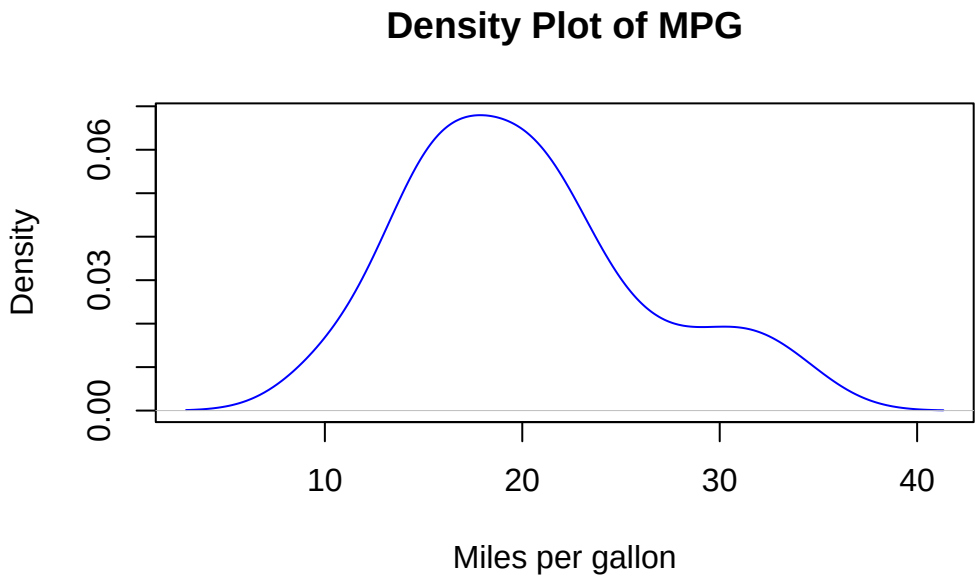


3. We create a histogram of the `mpg` column using the `hist()` function. The main argument is used to specify the title of the plot, and the xlab argument is used to specify the label for the x-axis.
4. The `col` argument is used to set the color of the bars in the histogram, and the border argument 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

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")
```

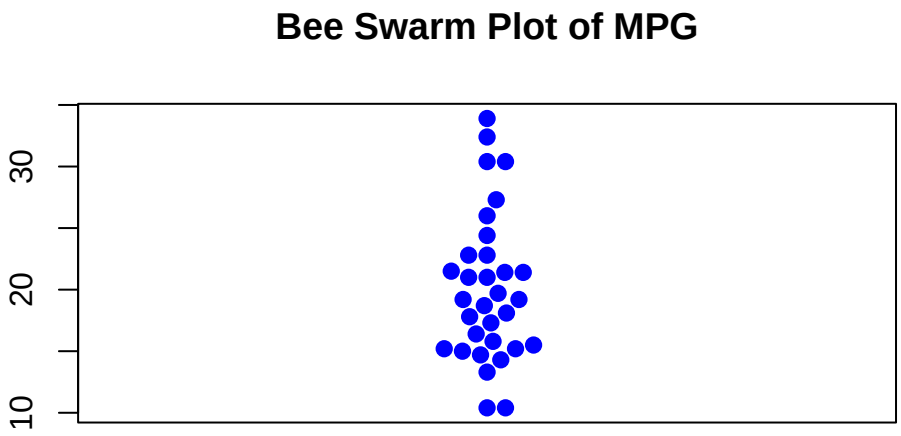


2. In the above code, we create a density plot of the mpg column using the `density()` function.
3. The `plot()` function is used to plot the resulting density object.
4. The `main` argument is used to specify the title of the plot, and the `xlab` argument is used to specify the label for the x-axis.
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.

Bee Swarm plot

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.

```
# 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.

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.