Topic 6: Basic Statistical Analysis

2023-08-08

In this topic, you will learn about :

- Summary statistics, frequency, and cross tabulation table

Summary statistics

Summary Statistics in R Programming

Summary statistics are numerical measures that provide a concise overview of the main characteristics of a dataset. They help to understand the central tendency, variability, and distribution of data. R provides various functions to compute summary statistics for data analysis.

Common Summary Statistics in R:

- Mean (**mean()**): The mean is the average value of a numeric variable.
- Median (**median()**): The median is the middle value of a numeric variable when the data is sorted.
- Minimum (**min()**), Maximum (**max()**): The minimum and maximum values are the smallest and largest values, respectively, in a numeric variable.
- Standard Deviation (**sd()**): The standard deviation measures the spread or variability of a numeric variable.
- Variance (**var()**): The variance is the square of the standard deviation and also measures variability.
- Quantiles (**quantile()**): Quantiles divide the data into equal parts, such as quartiles (Q1, Q2, Q3) and percentiles.

Example: Computing Summary Statistics

```r
# Sample data
data <- c(10, 15, 20, 25, 30, 35, 40)

# Compute summary statistics
mean_value <- mean(data)
median_value <- median(data)
min_value <- min(data)
max_value <- max(data)
sd_value <- sd(data)
var_value <- var(data)
quantiles <- quantile(data, probs = c(0.25, 0.5, 0.75))

print(mean_value)
```

```
## [1] 25
```

```r
print(median_value)
```

```
## [1] 25
```

```r
print(min_value)
```

```
## [1] 10
```

```r
print(max_value)
```

```
## [1] 40
```

```r
print(sd_value)
```

```
## [1] 10.80123
```

```r
print(var_value)
```

```
## [1] 116.6667
```

```r
print(quantiles)
```

```
##  25%  50%  75% 
## 17.5 25.0 32.5
```

Summary Statistics for Data Frames:

For data frames, you can use functions like **summary(), sapply(), and colMeans()** to compute summary statistics for each variable.

Example: Computing Summary Statistics for a Data Frame

```r
# Sample data frame
data <- data.frame(
  Age = c(25, 30, 22, 28, 35),
  Height = c(170, 180, 165, 175, 190),
  Weight = c(70, 80, 60, 75, 85)
)

# Compute summary statistics for each variable in the data frame
summary_stats <- summary(data)
means <- sapply(data, mean)
col_means <- colMeans(data)

print(summary_stats)
```

```
##       Age         Height        Weight  
##  Min.   :22   Min.   :165   Min.   :60  
##  1st Qu.:25   1st Qu.:170   1st Qu.:70  
##  Median :28   Median :175   Median :75  
##  Mean   :28   Mean   :176   Mean   :74  
##  3rd Qu.:30   3rd Qu.:180   3rd Qu.:80  
##  Max.   :35   Max.   :190   Max.   :85
```

```r
print(means)
```

```
##    Age Height Weight 
##     28    176     74
```

```r
print(col_means)
```

```
##    Age Height Weight 
##     28    176     74
```

Summary statistics are valuable tools to gain insights into your data quickly and can help you make informed decisions and observations during data analysis in R.

Frequency

Frequency in R Programming

Frequency refers to the count of occurrences of distinct values in a dataset. In R, you can calculate the frequency of values for categorical variables using various functions and techniques.

Calculating Frequency:

1. Using **table()**: The **table()** function creates a frequency table that shows the count of each unique value in a vector or data frame column.

Example: Calculating Frequency using table()

```r
# Sample data
gender <- c("Male", "Female", "Male", "Male", "Female", "Female", "Male", "Female")

# Calculate frequency of each unique value
frequency_table <- table(gender)
print(frequency_table)
```

```
## gender
## Female   Male 
##      4      4
```

2. Using **count()** from **dplyr**: If you are using the **dplyr** package, you can use the **count()** function to calculate the frequency of values in a data frame column.

Example: Calculating Frequency using count() from dplyr

```r
# Load dplyr library
library(dplyr)
```

```
## 
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union
```

```r
# Sample data frame
data <- data.frame(
  Gender = c("Male", "Female", "Male", "Male", "Female", "Female", "Male", "Female")
)

# Calculate frequency of each unique value in the Gender column
frequency_table <- count(data, Gender)
print(frequency_table)
```

```
##   Gender n
## 1 Female 4
## 2   Male 4
```

Frequency Table Interpretation:

The frequency table provides a summary of the counts for each unique value in the dataset.

Relative Frequency (Proportions):

You can also calculate relative frequency, which represents the proportion of each unique value relative to the total count.

Example: Calculating Relative Frequency using prop.table()

```
# Sample data
gender <- c("Male", "Female", "Male", "Male", "Female", "Female", "Male", "Female")

# Calculate relative frequency of each unique value
relative_frequency <- prop.table(table(gender))
print(relative_frequency)
```

```
## gender
## Female   Male
##    0.5    0.5
```

This means that "Female" has a relative frequency of 0.5 (50%), and "Male" also has a relative frequency of 0.5 (50%).

Summary:

1. Frequency refers to the count of occurrences of distinct values in a dataset.
2. Use **table()** or **count()** from **dplyr** to calculate the frequency of values for categorical variables.
3. Relative frequency can be obtained by dividing the frequency of each value by the total count.

Cross tabulation table

Cross Tabulation Table in R Programming

Cross tabulation (also known as a contingency table or crosstab) is a tabular representation that shows the distribution of one or more categorical variables against another. It allows you to analyze the relationship between two or more categorical variables in a data set.

Creating a Cross Tabulation Table:

1. Using **table()**: The **table()** function can be used to create a basic cross tabulation table in R.

Example: Creating a Cross Tabulation Table using table()

```
# Sample data
gender <- c("Male", "Female", "Male", "Male", "Female", "Female", "Male", "Female")
occupation <- c("Engineer", "Teacher", "Engineer", "Doctor", "Teacher", "Doctor", "Engineer", "Doctor")

# Create a cross tabulation table for gender and occupation
cross_tab <- table(gender, occupation)
print(cross_tab)
```

```
##         occupation
## gender   Doctor Engineer Teacher
##   Female      2        0       2
##   Male        1        3       0
```

2. Using **xtabs()**: The **xtabs()** function is another way to create cross-tabulation tables, providing more flexibility in defining formulas.

Example: Creating a Cross Tabulation Table using xtabs()

```r
# Sample data
gender <- c("Male", "Female", "Male", "Male", "Female", "Female", "Male", "Female")
occupation <- c("Engineer", "Teacher", "Engineer", "Doctor", "Teacher", "Doctor", "Engineer", "Doctor")

# Create a cross tabulation table for gender and occupation
cross_tab <- xtabs(~ gender + occupation)
print(cross_tab)
```

```
##         occupation
## gender   Doctor Engineer Teacher
##   Female      2        0       2
##   Male        1        3       0
```

Interpreting a Cross Tabulation Table:

A cross tabulation table displays the frequency of occurrences for combinations of values from two or more categorical variables. The rows represent the categories of one variable, while the columns represent the categories of the other variable.

Here, you can interpret that there are two females and one male in the "Doctor" category of occupation. In the "Engineer" category, there are no females but two males, and so on.

Summary:

1. A cross tabulation table shows the distribution of one or more categorical variables against another.
2. Use **table()** or **xtabs()** to create cross-tabulation tables in R.
3. Cross tabulation helps to analyze the relationships between categorical variables and is a useful tool for exploratory data analysis.