Topic 5: Data Cleaning and filtering

2023-08-16

In this topic, you will learn about :

- Keep and remove variables
- Filtering data and variables

Keep and remove variables

Keeping and Removing Variables in R Programming

In R, you can selectively keep or remove variables (columns) from a data frame using various functions and techniques. This allows you to manipulate and subset data to work with only the variables you need for analysis or visualization.

Keeping Variables:

1. Using $ Operator: You can keep specific variables by using the $ operator to select them from the data frame.

Example: Keeping Variables using $ Operator

```r
# Sample data frame
data <- data.frame(
  ID = 1:5,
  Age = c(25, 30, 22, 28, 35),
  Height = c(170, 180, 165, 175, 190),
  Weight = c(70, 80, 60, 75, 85)
)
print(data)
```

```
##   ID Age Height Weight
## 1  1  25    170     70
## 2  2  30    180     80
## 3  3  22    165     60
## 4  4  28    175     75
## 5  5  35    190     85
```

```r
# Keep only the Age and Height variables using $
a<-data$Age
a
```

```
## [1] 25 30 22 28 35
```

```r
# Keep only the Age and Height variables []
selected_data <- data[c("Age", "Height")]
print(selected_data)
```

```
##   Age Height
## 1  25    170
## 2  30    180
## 3  22    165
## 4  28    175
## 5  35    190
```

2. Using **Square Brackets**: You can use square brackets with column names to keep specific variables from the data frame.

Example: Keeping Variables using Square Brackets

```
# Sample data frame
data <- data.frame(
  ID = 1:5,
  Age = c(25, 30, 22, 28, 35),
  Height = c(170, 180, 165, 175, 190),
  Weight = c(70, 80, 60, 75, 85)
)

# Keep only the Age and Weight variables
selected_data <- data[, c("Age", "Weight")]
print(selected_data)
```

```
##   Age Weight
## 1  25     70
## 2  30     80
## 3  22     60
## 4  28     75
## 5  35     85
```

Removing Variables:

1. Using $ Operator or Square Brackets with Negation: You can remove specific variables by using the $ operator or square brackets with a negation sign (-) in front of the column names.

Example: Removing Variables using $ Operator and Square Brackets with Negation

```
# Sample data frame
data <- data.frame(
  ID = 1:5,
  Age = c(25, 30, 22, 28, 35),
  Height = c(170, 180, 165, 175, 190),
  Weight = c(70, 80, 60, 75, 85)
)

# Remove the ID variable
selected_data <- data[c("Age", "Height", "Weight")]
print(selected_data)
```

```
##   Age Height Weight
## 1  25    170     70
## 2  30    180     80
## 3  22    165     60
## 4  28    175     75
## 5  35    190     85
```

```
# Alternative method: Remove the ID variable
# Sample data frame
data <- data.frame(
  ID = 1:5,
```

```r
  Age = c(25, 30, 22, 28, 35),
  Height = c(170, 180, 165, 175, 190),
  Weight = c(70, 80, 60, 75, 85)
)
selected_data <- data[-1]
print(selected_data)
```

```
##   Age Height Weight
## 1  25    170     70
## 2  30    180     80
## 3  22    165     60
## 4  28    175     75
## 5  35    190     85
```

2. Using **select()** from **dplyr**: If you are using the **dplyr** package, you can use the **select()** function to remove variables.

Example: Removing Variables using **select()** 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(
  ID = 1:5,
  Age = c(25, 30, 22, 28, 35),
  Height = c(170, 180, 165, 175, 190),
  Weight = c(70, 80, 60, 75, 85)
)

# Remove the ID variable
selected_data <- select(data, -ID)
print(selected_data)
```

```
##   Age Height Weight
## 1  25    170     70
## 2  30    180     80
## 3  22    165     60
## 4  28    175     75
## 5  35    190     85
```

Summary:

1. You can keep specific variables using the $ operator, square brackets, or the **select()** function from **dplyr**.
2. To remove variables, use the negation sign with the $ operator, **square brackets**, or the **select()** function from **dplyr**.

3. Choose the method that best suits your data manipulation needs and the packages you are using.

Filtering data and variables

Filtering Data and Variables in R Programming

Filtering data and variables in R involves selecting specific rows or columns from a data frame based on certain conditions or criteria. This process allows you to extract subsets of the data for further analysis or visualization.

Filtering Data (Rows):

1. Using **Logical Indexing**: You can use logical indexing to filter rows based on specific conditions.

Example: Filtering Data using Logical Indexing

```
# Sample data frame
data <- data.frame(
  ID = 1:5,
  Age = c(25, 30, 22, 28, 35),
  Gender = c("Male", "Female", "Male", "Male", "Female")
)

# Filter rows where Age is greater than 25
filtered_data <- data[data$Age > 25, ]
print(filtered_data)
```

```
##   ID Age Gender
## 2  2  30 Female
## 4  4  28   Male
## 5  5  35 Female
```

Filtering Variables (Columns):

1. Using $ Operator or Square Brackets: You can use the $ operator or square brackets to select specific columns from a data frame.

Example: Filtering Variables using $ Operator and **Square Brackets**

```
# Sample data frame
data <- data.frame(
  ID = 1:5,
  Age = c(25, 30, 22, 28, 35),
  Gender = c("Male", "Female", "Male", "Male", "Female")
)

# Select only the Age and Gender columns
selected_variables <- data[c("Age", "Gender")]
print(selected_variables)
```

```
##   Age Gender
## 1  25   Male
## 2  30 Female
## 3  22   Male
## 4  28   Male
## 5  35 Female
```

2. Using **select()** from **dplyr**: If you are using the **dplyr** package, you can use the **select()** function to filter columns.

Example: Filtering Variables using **select()** from **dplyr**

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

# Sample data frame
data <- data.frame(
  ID = 1:5,
  Age = c(25, 30, 22, 28, 35),
  Gender = c("Male", "Female", "Male", "Male", "Female")
)
print(data)
```

```
##   ID Age Gender
## 1  1  25   Male
## 2  2  30 Female
## 3  3  22   Male
## 4  4  28   Male
## 5  5  35 Female
```

```r
# Select only the Age and Gender columns
selected_variables <- select(data, Age, Gender)
print(selected_variables)
```

```
##   Age Gender
## 1  25   Male
## 2  30 Female
## 3  22   Male
## 4  28   Male
## 5  35 Female
```

Summary:

1. Filtering data (rows) involves selecting specific rows based on conditions using logical indexing or the **filter()** function from dplyr.
2. Filtering variables (columns) involves selecting specific columns using the $ operator, square brackets, or the **select()** function from **dplyr**.
3. Choose the appropriate method based on your data manipulation needs and whether you are using the **dplyr** package for data manipulation.