R Lesson 2

Federico Reali

2024-10-18

Data Exploration and Manipulation in R

1. Loading Data

We will load and inspect four datasets:

- `Pima.tr` and `Pima.tr2`: Datasets from the `MASS` package.
- `BodyTemperature.txt`: A local dataset containing body temperature data.
- `Titanic.csv`: A dataset retrieved from a web source.

```r
# Load necessary library
library(MASS)

# Load Pima.tr dataset from MASS package
df <- MASS::Pima.tr
# View(df)

# Load the BodyTemperature dataset from local system
df_temp <- read.table(file = './Datasets/BodyTemperature.txt', header = TRU
# View(df_temp)

# Load Titanic dataset from local file
df_titanic <- read.csv(file = './Datasets/titanic.csv')
# View(df_titanic)

# Load Titanic dataset from web
df_from_web <- read.csv('https://web.stanford.edu/class/archive/cs/cs109/cs
# View(df_from_web)
```

2. Initial Data Exploration

Now, let's explore the `Pima.tr` and `BodyTemperature` datasets to get an overview of their structure and basic statistics.

```
{r}
# Explore the structure of Pima.tr dataset
str(df)

# Export data using write.csv/write.table if necessary
# write.csv(df, "Pima_tr_export.csv")
```

```
'data.frame':   200 obs. of  8 variables:
 $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
 $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
 $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
 $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
 $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
 $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
 $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
 $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

Summary

- **Summary** gives you a statistical summary of each variable in a dataset.
  - For numeric variables: Minimum, 1st quartile, median, mean, 3rd quartile, and maximum.
  - For factor or categorical variables: It counts occurrences of each level.

```
# Summary statistics of Pima.tr dataset
summary(df)
```

```
     npreg             glu              bp              skin      
 Min.   : 0.00   Min.   : 56.0   Min.   : 38.00   Min.   : 7.00  
 1st Qu.: 1.00   1st Qu.:100.0   1st Qu.: 64.00   1st Qu.:20.75  
 Median : 2.00   Median :120.5   Median : 70.00   Median :29.00  
 Mean   : 3.57   Mean   :124.0   Mean   : 71.26   Mean   :29.21  
 3rd Qu.: 6.00   3rd Qu.:144.0   3rd Qu.: 78.00   3rd Qu.:36.00  
 Max.   :14.00   Max.   :199.0   Max.   :110.00   Max.   :99.00  
      bmi             ped              age         type    
 Min.   :18.20   Min.   :0.0850   Min.   :21.00   No :132  
 1st Qu.:27.57   1st Qu.:0.2535   1st Qu.:23.00   Yes: 68  
 Median :32.80   Median :0.3725   Median :28.00            
 Mean   :32.31   Mean   :0.4608   Mean   :32.11            
 3rd Qu.:36.50   3rd Qu.:0.6160   3rd Qu.:39.25            
 Max.   :47.90   Max.   :2.2880   Max.   :63.00            
```

```r
# View the first and last rows of the dataset
head(df)
tail(df)
```

```
  npreg glu bp skin  bmi   ped age type
1     5  86 68   28 30.2 0.364  24   No
2     7 195 70   33 25.1 0.163  55  Yes
3     5  77 82   41 35.8 0.156  35   No
4     0 165 76   43 47.9 0.259  26   No
5     0 107 60   25 26.4 0.133  23   No
6     5  97 76   27 35.6 0.378  52  Yes
```

```
    npreg glu bp skin  bmi   ped age type
195     1 140 74   26 24.1 0.828  23   No
196     2 141 58   34 25.4 0.699  24   No
197     7 129 68   49 38.5 0.439  43  Yes
198     0 106 70   37 39.4 0.605  22   No
199     1 118 58   36 33.3 0.261  23   No
200     8 155 62   26 34.0 0.543  46  Yes
```

```r
# Dataset dimensions and column names
dim(df)
names(df)
```

```
[1] 200   8
```

```
[1] "npreg" "glu"   "bp"    "skin"  "bmi"   "ped"   "age"   "type"
```

3. Working with the BodyTemperature Dataset

We can also explore and manipulate the `BodyTemperature` dataset.

```r
df_temp <- read.table(file = './Datasets/BodyTemperature.txt', header = TRU
# View structure and summary of BodyTemperature dataset
str(df_temp)
```

```
'data.frame':	100 obs. of  4 variables:
 $ Gender     : chr  "M" "M" "M" "F" ...
 $ Age        : int  33 32 42 33 26 37 32 45 31 49 ...
 $ HeartRate  : int  69 72 68 75 68 79 71 73 77 81 ...
 $ Temperature: num  97 98.8 96.2 97.8 98.8 ...
```

```r
summary(df_temp)
```

```
    Gender               Age          HeartRate      Temperature    
 Length:100         Min.   :21.00   Min.   :61.00   Min.   : 96.20  
 Class :character   1st Qu.:33.75   1st Qu.:69.00   1st Qu.: 97.70  
 Mode  :character   Median :37.00   Median :73.00   Median : 98.30  
                    Mean   :37.62   Mean   :73.66   Mean   : 98.33  
                    3rd Qu.:42.00   3rd Qu.:78.00   3rd Qu.: 98.90  
                    Max.   :50.00   Max.   :87.00   Max.   :101.30  
```

What are Factors in R?

```r
# Let's look at an example
data <- c("Male", "Female", "Male", "Female", "Male")
factor_data <- as.factor(data)
factor_data
```

```
[1] Male   Female Male   Female Male
Levels: Female Male
```

- **Factors** in R are used to handle **categorical** data (data that has a limited number of unique values, like gender, colors, etc.).
- Factors are treated as **nominal (unordered)** or **ordinal (ordered)** categories.
- This is especially useful in statistical modeling because R treats factors differently than numeric data.

The `as.factor()` Function

```r
# Convert a character vector into a factor
char_vector <- c("red", "green", "blue", "green", "red")
factor_vector <- as.factor(char_vector)
factor_vector
```

```
[1] red   green blue  green red  
Levels: blue green red
```

```r
# Convert gender to factor and rename levels for better readability
df_temp$GenderFactor <- as.factor(df_temp$Gender)
levels(df_temp$GenderFactor) <- c("Female", "Male")

head(df_temp, 15)
```

```
   Gender Age HeartRate Temperature GenderFactor
1       M  33        69        97.0         Male
2       M  32        72        98.8         Male
3       M  42        68        96.2         Male
4       F  33        75        97.8       Female
5       F  26        68        98.8       Female
6       M  37        79       101.3         Male
7       F  32        71        97.8       Female
8       F  45        73        97.4       Female
9       F  31        77        99.2       Female
10      M  49        81        99.2         Male
11      M  40        69        97.5         Male
12      F  45        70        97.7       Female
13      F  49        71        98.3       Female
14      F  37        74        98.8       Female
```

```r
# Summary statistics for Age
mean(df_temp$Age)
```

```
[1] 37.62
```

```r
sd(df_temp$Age)
```

```
[1] 6.430326
```

```r
quantile(df_temp$Age, probs = seq(0, 1, 0.1))
```

```
  0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100% 
21.0 30.0 33.0 34.0 36.0 37.0 40.0 41.0 43.2 46.0 50.0 
```

```r
# Subsetting the data
df_temp[, c("Age", "HeartRate")]
```

```
   Age HeartRate
1   33        69
2   32        72
3   42        68
4   33        75
5   26        68
6   37        79
7   32        71
8   45        73
9   31        77
10  49        81
11  40        69
12  45        70
13  49        71
14  37        74
```

```r
df_temp[, 2:3]
```

```
   Age HeartRate
1   33        69
2   32        72
3   42        68
4   33        75
5   26        68
6   37        79
```

7	32	71
8	45	73
9	31	77
10	49	81
11	40	69
12	45	70
13	49	71
14	37	74

```r
df_temp[, c(2, 4)]
```

```
    Age Temperature
1    33        97.0
2    32        98.8
3    42        96.2
4    33        97.8
5    26        98.8
6    37       101.3
7    32        97.8
8    45        97.4
9    31        99.2
10   49        99.2
11   40        97.5
12   45        97.7
13   49        98.3
14   37        98.8
```

```r
mean(df_temp[, c(2)])
```

```
[1] 37.62
```

```r
mean(df_temp[, c(4)])
```

```
[1] 98.33
```

```r
mean(df_temp[, c(2, 4)])
```

```
[1] NA
```

```r
# does not work with multiple columns
```

```r
# Apply functions to multiple columns
apply(df_temp[, c(2, 4)], 2, mean)
```

```
     Age Temperature
   37.62       98.33
```

```r
apply(df_temp[, c(2, 4)], 2, sd)
```

```
      Age Temperature
6.4303259   0.9568995
```

```r
apply(df_temp[, c(2, 4)], 2, quantile)
```

```
       Age Temperature
0%   21.00        96.2
25%  33.75        97.7
50%  37.00        98.3
75%  42.00        98.9
100% 50.00       101.3
```

```r
# Sorting values
sort(df_temp$Age)
```

```
 [1] 21 22 23 25 25 26 28 29 30 30 30 30 31 31 31 31 32 32 32 33 33 33 33 33
33
[26] 34 34 34 34 34 34 34 34 35 35 35 35 36 36 36 36 36 36 36 36 37 37 37 37
37
[51] 37 37 38 38 38 38 38 39 40 40 40 40 40 40 41 41 41 41 41 41 41 42 42 42
42
[76] 42 43 43 43 43 44 44 44 44 44 45 45 45 45 46 46 47 47 48 48 49 49 49 49
50
```

```r
unique(df_temp$Gender)
```

```
[1] "M" "F"
```

```r
sort(unique(df_temp$Age))
```

```
 [1] 21 22 23 25 26 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47
[26] 48 49 50
```

4. Handling Missing Data

Missing data is a common issue in datasets, and it's essential to handle it properly for accurate analysis. Let's explore how to handle missing values in the `Pima.tr2` dataset by filling them with mean values.

```r
# Load dataset with missing data
df <- MASS::Pima.tr2
summary(df)
```

```
     npreg            glu              bp              skin      
 Min.   : 0.000   Min.   : 56.0   Min.   : 38.00   Min.   : 7.00  
 1st Qu.: 1.000   1st Qu.:101.0   1st Qu.: 64.00   1st Qu.:21.00  
 Median : 3.000   Median :121.0   Median : 72.00   Median :29.00  
 Mean   : 3.787   Mean   :123.7   Mean   : 72.32   Mean   :29.15  
 3rd Qu.: 6.000   3rd Qu.:142.0   3rd Qu.: 80.00   3rd Qu.:36.00  
 Max.   :14.000   Max.   :199.0   Max.   :114.00   Max.   :99.00  
                                  NA's   :13       NA's   :98     
      bmi             ped              age         type    
 Min.   :18.20   Min.   :0.0780   Min.   :21.0   No :194  
 1st Qu.:27.10   1st Qu.:0.2367   1st Qu.:24.0   Yes:106  
 Median :32.00   Median :0.3360   Median :29.0            
 Mean   :32.05   Mean   :0.4357   Mean   :33.1            
 3rd Qu.:36.50   3rd Qu.:0.5867   3rd Qu.:40.0            
 Max.   :52.90   Max.   :2.2880   Max.   :72.0            
```

```r
# Check for missing values
# is.na(df)
any(is.na(df))
```

```
[1] TRUE
```

```r
# Remove rows with missing data
dim(df)
```

```
[1] 300   8
```

```r
df_clean <- na.omit(df)
dim(df_clean)
```

```
[1] 200   8
```

```r
# Substitute missing blood pressure (bp) values with the mean
df$bp[is.na(df$bp)] <- mean(df$bp, na.rm = TRUE)

# Check if missing values were replaced
any(is.na(df$bp))
```

```
[1] FALSE
```

```r
# Substitute missing BMI values with the mean
which(is.na(df$bmi))
```

```
[1] 213 230 268
```

```r
index_NA_BMI <- which(is.na(df$bmi))
df$bmi[index_NA_BMI] <- mean(df$bmi, na.rm = TRUE)

# Verify no missing values for BMI remain
any(is.na(df$bmi))
```

```
[1] FALSE
```

5. Advanced Data Manipulation

In more advanced cases, we can substitute missing values based on conditions. For instance, we can fill missing values in the `skin` variable by computing the mean for specific groups (e.g., for different types).

```
# Example: Replace missing values in the variable 'skin' based on a conditi
# This approach involves calculating the mean for subsets of data (grouped
# df$skin[is.na(df$skin)] <- ... # Example to illustrate the logic
```

Try this at home!

6. Exercise: Exploring the Titanic Dataset

In this exercise, you will explore the Titanic dataset and apply the data exploration and cleaning techniques discussed in the previous lesson. Follow the guided steps to complete the tasks.

Step 1: Load the Titanic Dataset

Start by loading the Titanic dataset, which you can either use from a local file or from the web.

```
# Load Titanic dataset from web
df_titanic <- read.csv('https://web.stanford.edu/class/archive/cs/cs109/cs1
```

```r
# View the dataset structure and summary
str(df_titanic)
```

```
'data.frame':	887 obs. of  8 variables:
 $ Survived               : int  0 1 1 1 0 0 0 0 1 1 ...
 $ Pclass                 : int  3 1 3 1 3 3 1 3 3 2 ...
 $ Name                   : chr  "Mr. Owen Harris Braund" "Mrs. John Bradley (Florence Briggs Thayer) Cumings" "Miss. Laina Heikkinen" "Mrs. Jacques Heath (Lily May Peel) Futrelle" ...
 $ Sex                    : chr  "male" "female" "female" "female" ...
 $ Age                    : num  22 38 26 35 35 27 54 2 27 14 ...
 $ Siblings.Spouses.Aboard: int  1 1 0 1 0 0 0 3 0 1 ...
 $ Parents.Children.Aboard: int  0 0 0 0 0 0 0 1 2 0 ...
 $ Fare                   : num  7.25 71.28 7.92 53.1 8.05 ...
```

```r
summary(df_titanic)
```

```
    Survived          Pclass          Name               Sex           
 Min.   :0.0000   Min.   :1.000   Length:887         Length:887        
 1st Qu.:0.0000   1st Qu.:2.000   Class :character   Class :character  
 Median :0.0000   Median :3.000   Mode  :character   Mode  :character  
 Mean   :0.3856   Mean   :2.306                                        
 3rd Qu.:1.0000   3rd Qu.:3.000                                        
 Max.   :1.0000   Max.   :3.000                                        
      Age        Siblings.Spouses.Aboard Parents.Children.Aboard
 Min.   : 0.42   Min.   :0.0000          Min.   :0.0000         
 1st Qu.:20.25   1st Qu.:0.0000          1st Qu.:0.0000         
 Median :28.00   Median :0.0000          Median :0.0000         
 Mean   :29.47   Mean   :0.5254          Mean   :0.3833         
 3rd Qu.:38.00   3rd Qu.:1.0000          3rd Qu.:0.0000         
 Max.   :80.00   Max.   :8.0000          Max.   :6.0000         
      Fare       
```

Task 1: Inspect the Data

1. Use the `head()` and `tail()` functions to view the first and last 6 rows of the dataset.
2. Check the dimensions of the dataset (number of rows and columns) using the `dim()` function.
3. List all the column names using the `names()` function.

```r
# View the first and last rows of the dataset
head(df_titanic)
```

```
  Survived Pclass                                               Name    Sex
Age
1        0      3                             Mr. Owen Harris Braund   male
22
2        1      1 Mrs. John Bradley (Florence Briggs Thayer) Cumings female
38
3        1      3                              Miss. Laina Heikkinen female
26
4        1      1        Mrs. Jacques Heath (Lily May Peel) Futrelle female
35
5        0      3                            Mr. William Henry Allen   male
35
6        0      3                                    Mr. James Moran   male
27
  Siblings.Spouses.Aboard Parents.Children.Aboard    Fare
```

```
1 tail(df_titanic)
```

```
    Survived Pclass                                Name    Sex Age
882        0      3 Mrs. William (Margaret Norton) Rice female  39
883        0      2                Rev. Juozas Montvila   male  27
884        1      1        Miss. Margaret Edith Graham female  19
885        0      3     Miss. Catherine Helen Johnston female   7
886        1      1                Mr. Karl Howell Behr   male  26
887        0      3                  Mr. Patrick Dooley   male  32
    Siblings.Spouses.Aboard Parents.Children.Aboard   Fare
882                       0                       5 29.125
883                       0                       0 13.000
884                       0                       0 30.000
885                       1                       2 23.450
886                       0                       0 30.000
887                       0                       0  7.750
```

```r
# Get dimensions and column names
dim(df_titanic)
```

```
[1] 887   8
```

```r
names(df_titanic)
```

```
[1] "Survived"                "Pclass"
[3] "Name"                    "Sex"
[5] "Age"                     "Siblings.Spouses.Aboard"
[7] "Parents.Children.Aboard" "Fare"
```

Step 2: Explore Key Variables

Next, focus on exploring some key variables in the dataset, such as `Age`, `Fare`, and `Survived`.

Task 2: Statistical Summary of Key Variables

1. Calculate the mean and standard deviation of the `Age` and `Fare` columns.
2. Compute the quantiles of the `Age` variable in 10% intervals.

```r
# Mean and standard deviation for Age and Fare
mean(df_titanic$Age, na.rm = TRUE)
```

```
[1] 29.47144
```

```r
sd(df_titanic$Age, na.rm = TRUE)
```

```
[1] 14.12191
```

```r
mean(df_titanic$Fare, na.rm = TRUE)
```

```
[1] 32.30542
```

```r
sd(df_titanic$Fare, na.rm = TRUE)
```

```
[1] 49.78204
```

```r
# Quantiles of Age
quantile(df_titanic$Age, probs = seq(0, 1, 0.1), na.rm = TRUE)
```

```
   0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100% 
 0.42 14.80 19.00 22.00 24.00 28.00 31.00 35.00 40.40 49.00 80.00 
```

Step 3: Handling Missing Data

Now that we've explored the dataset, let's address missing data. For example, the `Age` variable has missing values.

Task 3: Handling Missing Values

1. Identify which variables have missing data using the `is.na()` and `summary()` functions.
2. If any NA is present, remove rows with missing data using the `na.omit()` function and check the new dimensions of the dataset.
3. If any NA is present, instead of removing, fill the missing values in the `Age` column with the mean of the column.

```r
# Check for missing values
summary(df_titanic)
```

```
    Survived          Pclass          Name               Sex           
 Min.   :0.0000   Min.   :1.000   Length:887         Length:887        
 1st Qu.:0.0000   1st Qu.:2.000   Class :character   Class :character  
 Median :0.0000   Median :3.000   Mode  :character   Mode  :character  
 Mean   :0.3856   Mean   :2.306                                        
 3rd Qu.:1.0000   3rd Qu.:3.000                                        
 Max.   :1.0000   Max.   :3.000                                        
      Age        Siblings.Spouses.Aboard Parents.Children.Aboard
 Min.   : 0.42   Min.   :0.0000          Min.   :0.0000         
 1st Qu.:20.25   1st Qu.:0.0000          1st Qu.:0.0000         
 Median :28.00   Median :0.0000          Median :0.0000         
 Mean   :29.47   Mean   :0.5254          Mean   :0.3833         
 3rd Qu.:38.00   3rd Qu.:1.0000          3rd Qu.:0.0000         
 Max.   :80.00   Max.   :8.0000          Max.   :6.0000         
      Fare       
```

```r
any(is.na(df_titanic))
```

```
[1] FALSE
```

```r
# No missing value!
```

Step 4: Factor Variables

Some columns, such as `Sex` and `Survived`, are categorical variables. Let’s convert them into factors and modify their levels to be more descriptive.

Task 4: Convert Categorical Variables to Factors

1. Convert the `Sex` and `Survived` columns into factors.
2. Rename the levels of the `Survived` column to `"Died"` and `"Survived"`.

```r
# Convert Sex and Survived to factors
df_titanic$Sex <- as.factor(df_titanic$Sex)
df_titanic$Survived <- as.factor(df_titanic$Survived)

# Rename the levels of Survived
levels(df_titanic$Survived) <- c("Died", "Survived")

# Check the structure of the modified columns
str(df_titanic$Sex)
```

```
Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
```

```r
str(df_titanic$Survived)
```

```
Factor w/ 2 levels "Died","Survived": 1 2 2 2 1 1 1 1 2 2 ...
```

Step 5: Advanced Data Manipulation

For more advanced tasks, you can apply functions to multiple columns or subsets of the data.

Task 5: Data Subsetting and Application of Functions

1. Extract the `Age` and `Fare` columns into a new subset.
2. Apply the `mean()` and `sd()` functions to both columns using the `apply()` function.
3. Sort the `Age` column in ascending order.

```r
# Subset Age and Fare columns
age_fare_subset <- df_titanic[, c("Age", "Fare")]

# Apply mean and standard deviation to the subset
apply(age_fare_subset, 2, mean, na.rm = TRUE)
```

```
     Age     Fare
29.47144 32.30542
```

```r
apply(age_fare_subset, 2, sd, na.rm = TRUE)
```

```
     Age     Fare
14.12191 49.78204
```

```r
# Sort the Age column
sorted_age <- sort(df_titanic$Age, na.last = TRUE)
head(sorted_age, 20)
```

```
 [1] 0.42 0.67 0.75 0.75 0.83 0.83 0.92 1.00 1.00 1.00 1.00 1.00 1.00 1.00
2.00
[16] 2.00 2.00 2.00 2.00 2.00
```

Step 6: Identify the Best Single Predictor of Survival

In this task, you will explore which categorical variable (such as `Sex` or `Pclass`) is the best single predictor for survival using the `table()` command.

Task 6: Use `table()` to Identify the Best Predictor of `Survived`

1. Use the `table()` function to create a contingency table between `Survived` and each of the following variables: `Sex` and `Pclass`.
2. Compare the results and determine which variable has the strongest relationship with `Survived`. You can use the function `prop.table`.
3. Explore the other variables.

```r
# Contingency table between Survived and Sex
table(df_titanic$Survived, df_titanic$Sex)
```

```
          female male
Died          81  464
Survived     233  109
```

```r
# Contingency table between Survived and Pclass
table(df_titanic$Survived, df_titanic$Pclass)
```

```
           1   2   3
Died      80  97 368
Survived 136  87 119
```

```r
# Based on the results, identify the variable with the strongest relationsh
```

Conclusion

In this section, we explored multiple datasets, including local and web-based datasets. We demonstrated the importance of understanding the structure and statistics of the data, handling missing values, and performing basic data manipulation. These are foundational steps before moving on to more advanced analysis or modeling.

Exercises

Exercise 1

Using the techniques used today, analyze the dataset `iris` available in R. You can find more information on the dataset using the `help`.

```r
# Iris dataset
data(iris)
```

Exercise 2

Using the techniques used today, analyze the `wine` dataset available at: https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

This dataset contains information about red wine samples, including characteristics such as acidity, sugar content, pH, alcohol percentage, and quality ratings.

```r
# Wine Quality dataset
wine_url <- 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine
wine <- read.csv(wine_url, sep = ";")
```

After having analyzed the dataset, can you find which is the best predictor of the variable *quality ratings*?