Data Cleaning in R

Felipe O. Cerezer

2024-09-29

Step 1. Required Libraries

We will use the following R packages: - `dplyr` For data manipulation - `tidyr` for handling missing values - `ggplot2` for data visualization

```
## Install required packages if not already installed
#install.packages(c("dplyr", "tidyr", "ggplot2"))

## Load libraries
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
```

```
library(tidyr)
library(ggplot2)
```

Step 2. Simulated Dataset

First, I will create a simulated dataset containing customer information to work with.

```
set.seed(42) # For reproducibility

# Simulated Dataset: Customer Information
data <- data.frame(
  Customer_ID = c(101, 102, 103, 104, 105, 102, 106, 107, 108, 109),
  Name = c("John Doe", "Jane Smith", "Sam Brown", "Sue Johson", "Mike White",
           "Jane Smith", "Emily Davis", "Michael Johnson", "Chris Lee", "Chris Lee"),
  Email = c("john@example.com", "jane@example.com", NA, "sue_j@example.com", "mike.w@example.com",
            "jane@example.com", "emily.d@example.com", "michael.j@example.com", "chris.l@example.com",
  Purchase_Amount = c(200, 300, 150, NA, 4000, 300, 250, 180, 190, 200),
  stringsAsFactors = FALSE
)

# Introduce missing values for demonstration
data$Purchase_Amount[c(5, 4)] <- NA  # Set Purchase_Amount for some customers as missing
```

```r
# View the original dataset
print("Original Data:")
```

```
## [1] "Original Data:"
```

```r
print(data)
```

```
##    Customer_ID            Name                 Email Purchase_Amount
## 1          101        John Doe      john@example.com             200
## 2          102      Jane Smith      jane@example.com             300
## 3          103       Sam Brown                  <NA>             150
## 4          104      Sue Johson     sue_j@example.com              NA
## 5          105      Mike White    mike.w@example.com              NA
## 6          102      Jane Smith      jane@example.com             300
## 7          106     Emily Davis   emily.d@example.com             250
## 8          107 Michael Johnson michael.j@example.com             180
## 9          108       Chris Lee   chris.l@example.com             190
## 10         109       Chris Lee   chris.l@example.com             200
```

Step 3: Fixing Typos

Next, I will address misspelled names in the dataset by using a correction table that maps incorrect names to their correct versions.

```r
# Create a lookup table for correcting name typos
correction_table <- data.frame(
  Incorrect = c("Sue Johson"),
  Correct = c("Sue Johnson"),
  stringsAsFactors = FALSE
)

# Correct the typo in the Name column
data_cleaned <- data %>%
  mutate(Name = ifelse(Name %in% correction_table$Incorrect,
                       correction_table$Correct[match(Name, correction_table$Incorrect)],
                       Name))

# Check the corrected dataset
print("Data after correcting typos:")
```

```
## [1] "Data after correcting typos:"
```

```r
print(data_cleaned)
```

```
##    Customer_ID            Name                 Email Purchase_Amount
## 1          101        John Doe      john@example.com             200
## 2          102      Jane Smith      jane@example.com             300
## 3          103       Sam Brown                  <NA>             150
## 4          104     Sue Johnson     sue_j@example.com              NA
## 5          105      Mike White    mike.w@example.com              NA
## 6          102      Jane Smith      jane@example.com             300
## 7          106     Emily Davis   emily.d@example.com             250
## 8          107 Michael Johnson michael.j@example.com             180
## 9          108       Chris Lee   chris.l@example.com             190
## 10         109       Chris Lee   chris.l@example.com             200
```

Explanation: Here, I created a correction table that contains the incorrect name "Sue Johson" alongside its correct version "Sue Johnson." I then used the mutate() function from the dplyr package to modify the Name column, replacing any occurrences of the incorrect name with the correct one. Finally, I printed the cleaned dataset to illustrate the successful correction.

Step 4: Handling Missing Values

I will identify missing values, analyze their extent, and decide on an appropriate method for handling them.

```
# Check for missing data
missing_data_summary <- colSums(is.na(data_cleaned))
print("Missing Data Summary:")
```

```
## [1] "Missing Data Summary:"
```

```
print(missing_data_summary)
```

```
##     Customer_ID            Name           Email Purchase_Amount
##               0               0               1               2
```

```
# Option 1: Impute missing Purchase_Amount with the median
median_purchase <- median(data_cleaned$Purchase_Amount, na.rm = TRUE)

data_cleaned <- data_cleaned %>%
  mutate(Purchase_Amount = ifelse(is.na(Purchase_Amount),
                                  median_purchase,
                                  Purchase_Amount))

# Option 2: Remove rows with missing Email (assuming Email is essential)
data_cleaned <- data_cleaned %>%
  filter(!is.na(Email))

print("Data after handling missing values:")
```

```
## [1] "Data after handling missing values:"
```

```
print(data_cleaned)
```

```
##   Customer_ID            Name                 Email Purchase_Amount
## 1         101        John Doe      john@example.com             200
## 2         102      Jane Smith      jane@example.com             300
## 3         104     Sue Johnson     sue_j@example.com             200
## 4         105      Mike White    mike.w@example.com             200
## 5         102      Jane Smith      jane@example.com             300
## 6         106     Emily Davis   emily.d@example.com             250
## 7         107 Michael Johnson michael.j@example.com             180
## 8         108       Chris Lee   chris.l@example.com             190
## 9         109       Chris Lee   chris.l@example.com             200
```

Explanation: In this section, I began by checking for missing data using the is.na() function and summarizing it with colSums(), allowing us to understand the extent of missing values in the dataset. I then addressed the missing values in the Purchase_Amount column by calculating the median of the available amounts and imputing the missing entries with this median. Additionally, I removed any rows with missing values in the Email column, assuming this information is essential for our analysis. The cleaned dataset was printed to show the updated information after addressing missing values.

Step 5: Removing Duplicates

Now, I will identify and remove duplicate entries based on the Customer_ID.

```
# Identify and count the number of duplicate entries based on Customer_ID
num_duplicates <- data %>%
  group_by(Customer_ID) %>%
  filter(n() > 1) %>%
  summarise(duplicate_count = n())

# Print the number of duplicates found
print(paste("Number of duplicate entries based on Customer_ID:", nrow(num_duplicates)))
```

```
## [1] "Number of duplicate entries based on Customer_ID: 1"
```

```
# Remove duplicates, keeping the first occurrence
data_cleaned <- data_cleaned %>%
  distinct(Customer_ID, .keep_all = TRUE)

print("Data after removing duplicates:")
```

```
## [1] "Data after removing duplicates:"
```

```
print(data_cleaned)
```

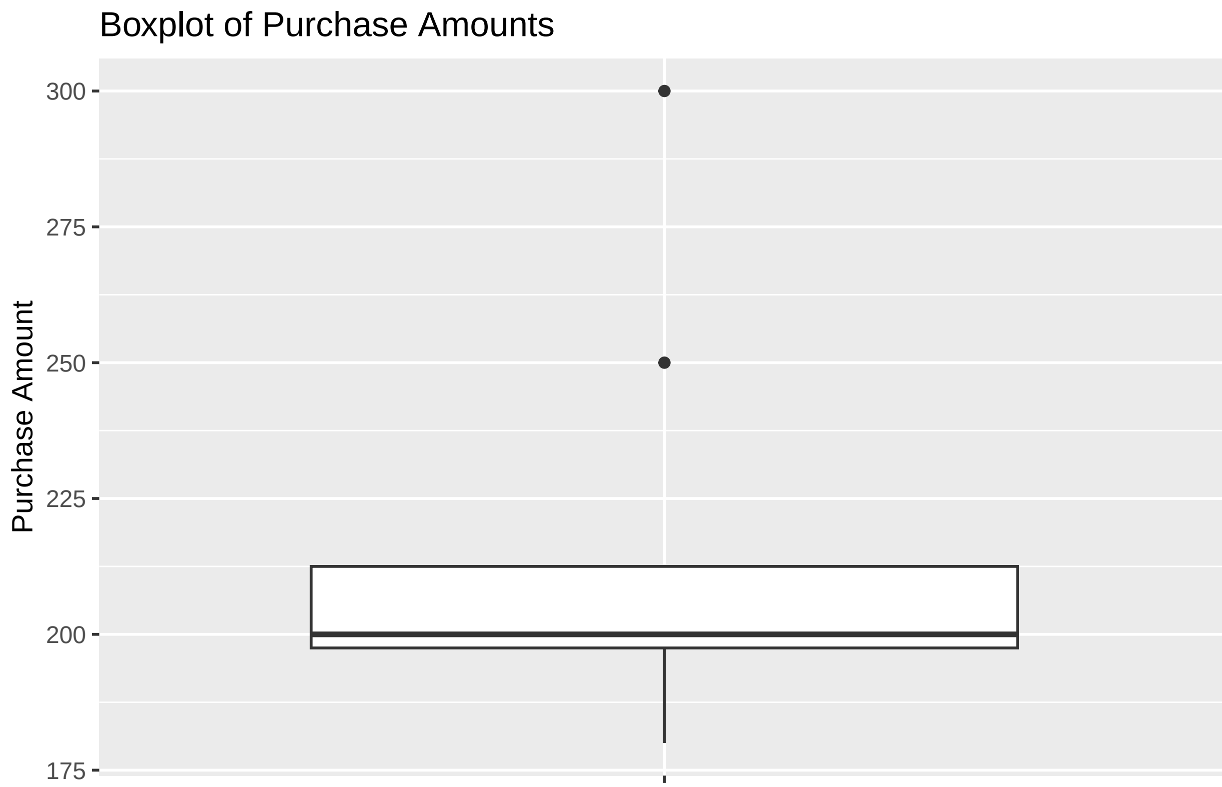
```
##   Customer_ID            Name                 Email Purchase_Amount
## 1         101        John Doe      john@example.com             200
## 2         102      Jane Smith      jane@example.com             300
## 3         104     Sue Johnson     sue_j@example.com             200
## 4         105      Mike White    mike.w@example.com             200
## 5         106     Emily Davis   emily.d@example.com             250
## 6         107 Michael Johnson michael.j@example.com             180
## 7         108       Chris Lee   chris.l@example.com             190
## 8         109       Chris Lee   chris.l@example.com             200
```

Explanation: In this part, I first identified duplicate entries by grouping the dataset by Customer_ID and filtering groups with more than one entry. I summarized the counts to understand how many duplicates existed and printed this information for awareness. After identifying duplicates, I removed them using the distinct() function, keeping only the first occurrence of each Customer_ID. The cleaned dataset was printed again to confirm the successful removal of duplicates.

Step 6: Detecting and Removing Outliers

In this step, I will identify outliers in the Purchase_Amount using the Interquartile Range (IQR) method and visualize the results

```
# Visualize Purchase_Amount to detect outliers
ggplot(data_cleaned, aes(x = "", y = Purchase_Amount)) +
  geom_boxplot() +
  ggtitle("Boxplot of Purchase Amounts") +
  ylab("Purchase Amount") +
  xlab("")
```




```r
# Calculate IQR to identify outliers
Q1 <- quantile(data_cleaned$Purchase_Amount, 0.25)
Q3 <- quantile(data_cleaned$Purchase_Amount, 0.75)
IQR <- Q3 - Q1

# Define bounds for outliers
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
data_cleaned <- data_cleaned %>%
  filter(Purchase_Amount >= lower_bound & Purchase_Amount <= upper_bound)

print("Data after removing outliers:")
```

```
## [1] "Data after removing outliers:"
```

```r
print(data_cleaned)
```

```
##   Customer_ID            Name                  Email Purchase_Amount
## 1         101        John Doe       john@example.com             200
## 2         104     Sue Johnson      sue_j@example.com             200
## 3         105      Mike White     mike.w@example.com             200
## 4         107 Michael Johnson michael.j@example.com             180
## 5         108       Chris Lee    chris.l@example.com             190
## 6         109       Chris Lee    chris.l@example.com             200
```

Explanation: In this section, I began by visualizing the Purchase_Amount data with a boxplot to identify potential outliers. The boxplot allows us to quickly assess the distribution and see extreme values. I then calculated the first (Q1) and third (Q3) quartiles of the Purchase_Amount, along with the IQR, to define bounds for outliers. Any entry falling below the lower bound or above the upper bound was removed from the dataset. The cleaned dataset was printed to show the result of this outlier removal.