TSAR Project Part 2

Michèle Fille

April 9, 2024

Table of contents

List of Figures

Introduction

This project focuses on data wrangling using R, specifically utilising the dplyr and tidyr packages along with their functions. The objective is to process and visualise a dataset containing special values. The initial step involves loading the data and making it “dirty” by applying the function `dirtyfy.rds`. Consequently, the dataset becomes contaminated with various special values. These include NAs (Not Available) for all value types, for character type values, “n/a”, which are not automatically recognised as NA, as well as space (” “) and empty string (”“). For numeric values, special values include NaN (not a number) and outliers, identified using Tukey’s fences. Tukey’s fences define outliers as values lying outside the whiskers of the boxplot. Subsequently, the process involves counting the occurrences of special values per attribute and visualising the occurrences as percentages. Below, each step of the project is illustrated, demonstrating the application of dplyr and tidyr functions in data processing with R.

Task 1: Data Loading and Preparation

In this section, the private dataset from Project 1 is loaded and intentionally contaminated with special values, making it "dirty".

The below R code chunk is intended to install the required packages. However, since all of them are already installed on my device, they are commented out. Afterwards, the respective libraries are loaded.

```r
# Install and load the required packages and libraries for this project
#install.packages("data.table") # already installed
#install.packages("tidyr") # already installed
#install.packages("dplyr") # already installed
#install.packages("DataExplorer") # already installed
#install.packages("ggplot2") # already installed
library(data.table)
library(tidyr)
library(dplyr)
library(DataExplorer)
library(ggplot2)
```

Load myLCdata

```r
# Read the data
myLCdata <- fread("myLCdata.csv")
```

The R code chunk above loads the data into the variable `myLCdata`.

Apply "dirtyfy.rds" Function

```r
# Load the dirtyfy function from the RDS object
dirtyfy <- readRDS("dirtyfy.rds")

# Apply the dirtyfy function to your data set
myLCdata_dirty <- dirtyfy(myLCdata, 4)
```

The R code chunk above loads the function `dirtyfy.rds` into the variable `dirtyfy`, and then applies this function to the data in `myLCdata` using the value of my personal TSAR ID, which is 4.

Generate EDA report with "create_report()"

```
# Apply create_report() to myLCdata_dirty *
# create_report(myLCdata_dirty) # please only excute in the console
```

The R code snippet provided is intended solely for documentation purposes. It is important not to execute the `create_report()` function from the DataExplorer library within the Quarto file; instead, it should be run in the console.

The `create_report()` function automatically generates an EDA (Exploratory Data Analysis) report. One of the visualisations included in this EDA report is represented by Figure 1, which illustrates the percentage of missing rows per attribute.

Missing Data Profile Screenshot

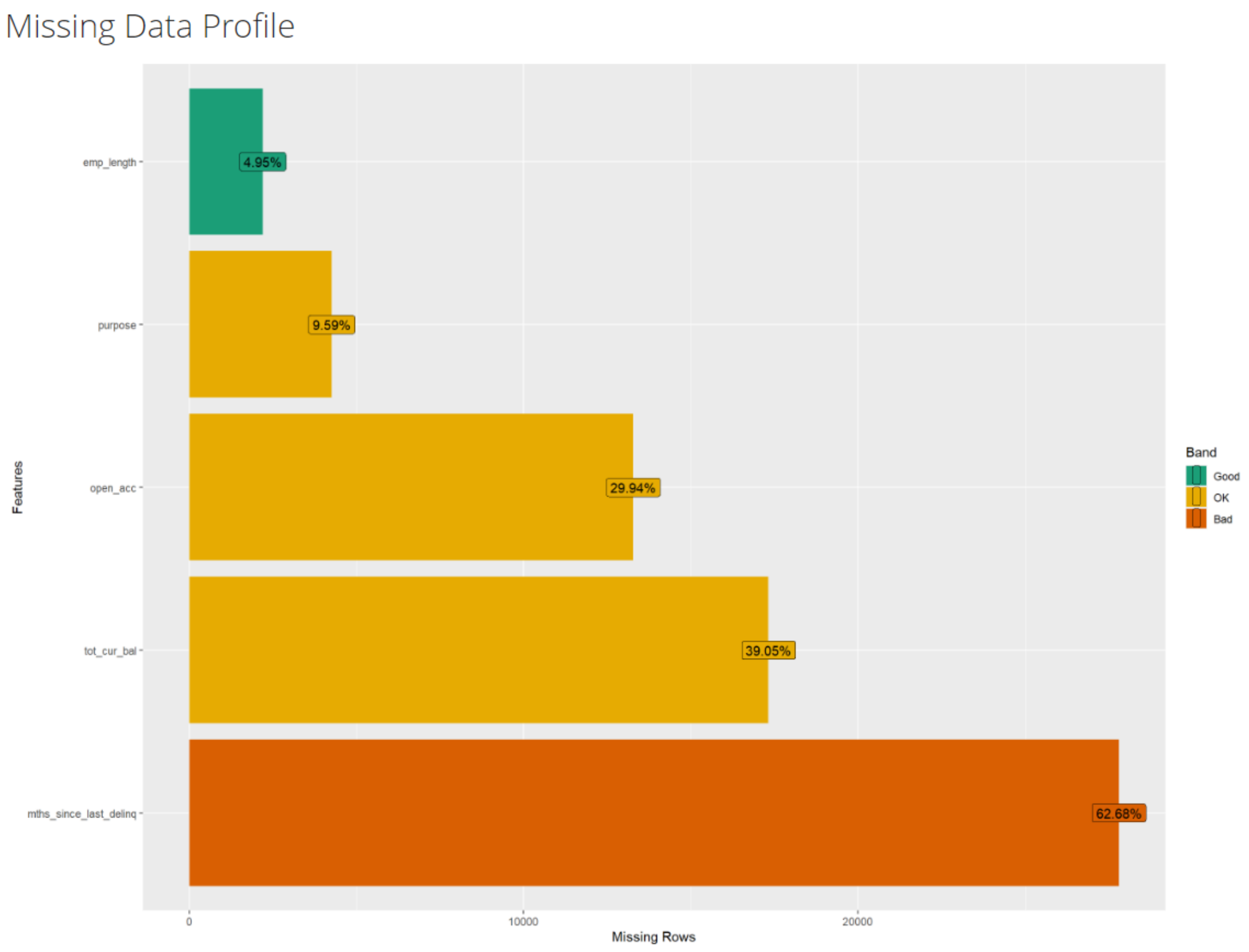

Figure 1: Missing Data Profile

Task 2: Counting and Visualising Special Values

Here, the occurrences of special values per attribute will be counted, and their percentages will be depicted in grouped bar charts. One chart will be dedicated to attributes with character data type, while another will represent attributes with numeric data type.

Special values for attributes with character data type may include: NA (Not Available), "n/a" string, space (" "), and empty string (""). In contrast, for attributes with numeric data type, special values may encompass NA (Not Available), NaN (Not a Number), and outliers (Tukey's fences).

```r
# Overview of data: column data-types
print(sapply(myLCdata_dirty, class))
```

```
mths_since_last_delinq             emp_length                purpose 
             "numeric"            "character"            "character" 
              open_acc            tot_cur_bal 
             "numeric"              "numeric" 
```

The above R code chunk provides an overview of the data, specifying all attributes along with their data types. Since the dataset used in this project is identical to the one used in Project 1, it contains the same attributes, namely: emp_length, mths_since_last_delinq, open_acc, purpose, and tot_cur_bal. Among these attributes, emp_length and purpose are of datatype character, while mths_since_last_delinq, open_acc, and tot_cur_bal are of numeric data type.

In the subsequent parts, the attributes of character data type and numeric data type are examined separately.

Character Attributes

In this part, the occurrences of special values (NA, “n/a” string, space (” “), empty string (”“)) in the attributes of data type character are counted. The percentages are calculated based on the total number of occurrences in relation to the total number of rows in the dataset, and they are visualised in a grouped bar chart.

Compute Special Value Counts

```
# Calculate special value counts for character attributes
character_counts <- myLCdata_dirty %>%
  summarise(
    across(
      where(is.character), # Operate only on character columns
      list(
        count_NA = ~sum(is.na(.)), # count NAs
        count_na_string = ~sum(. == "n/a", na.rm = TRUE), # Count "n/a" strings
        count_space = ~sum(. == " ", na.rm = TRUE), # Count spaces
        count_empty_string = ~sum(. == "", na.rm = TRUE) # Count empty strings
      ), .names = "{.col}.{fn}" # Name resulting columns (column name . function name)
    )
  )

# Print the resulting dataframe
print(character_counts)
```

```
  emp_length.count_NA emp_length.count_na_string emp_length.count_space
1                2197                       5997                   6401
  emp_length.count_empty_string purpose.count_NA purpose.count_na_string
1                          3412             4255                    7429
  purpose.count_space purpose.count_empty_string
1                3528                       3991
```

The above R code chunk computes the counts of special values for character attributes in the dataset ‘myLCdata_dirty’. It specifically counts the occurrences of NA (Not Available), “n/a” string, space (” “), and empty string (”“). The code utilises the dplyr package to summarise the data, employing functions like `summarise()` and `across()` to calculate counts for each type of special value. Finally, the results are displayed on the console using `print()`.

For calculations, such as counting special strings in character columns, NA values need to be excluded before performing the calculations. This is because functions like `sum()` cannot handle NA values by default, and including them would lead to inaccurate calculations. By setting `na.rm = TRUE`, it ensures that NA values are not considered in the calculations, thereby allowing accurate results to be obtained.

Restructure DataFrame Format

```r
# Pivot the dataframe to a longer format using pivot_longer()
character_table_counts <- character_counts %>%
  pivot_longer(
    cols = everything(),  # Pivot all columns
    names_to = "Attribute",  # Create a new column 'Attribute' to store the attribute names
    values_to = "Value"  # Create a new column 'Value' to store the counts
  ) %>%

  # Separate the 'Attribute' column into 'Attribute' and 'Calculation' columns
  separate(Attribute, into = c("Attribute", "Calculation"), sep = "\\.", remove = FALSE) %>%

  # Pivot the dataframe back to a wider format using pivot_wider()
  pivot_wider(
    names_from = "Calculation",  # Use the 'Calculation' column values as new column names
    values_from = "Value"  # Use the 'Value' column values as counts for each calculation type
  ) %>%

  arrange(Attribute) # Arrange the dataframe by attribute name

# Print the resulting dataframe
print(character_table_counts)
```

```
# A tibble: 2 x 5
  Attribute  count_NA count_na_string count_space count_empty_string
  <chr>         <int>           <int>       <int>              <int>
1 emp_length     2197            5997        6401               3412
2 purpose        4255            7429        3528               3991
```

The above R code chunk pivots the dataframe to the desired format for analysis. Initially, it utilises `pivot_longer()` to convert the dataframe into a longer format, separating the 'Attribute' and 'Calculation' columns. Subsequently, it applies `pivot_wider()` to revert the dataframe back to a wider format, organising the counts by each type of special value. Finally, the resulting dataframe, 'character_table_counts', is printed to the console.

Calculate Special Value Percentage

```r
# Calculate the total number of rows in myLCdata_dirty
total_rows <- nrow(myLCdata_dirty)

# Convert counts to relative counts
character_table_perc <- character_table_counts %>%
  mutate(across(
    where(is.numeric), # Select numeric columns
    ~ ./total_rows # Divide each count by the total number of rows
  ))

# Rename the headers replacing "count" with "perc"
character_table_perc <- character_table_perc %>%
  rename_with(~ gsub("count", "perc", .x), contains("count"))

# Print the resulting dataframe
print(character_table_perc)
```

```
# A tibble: 2 x 5
  Attribute  perc_NA perc_na_string perc_space perc_empty_string
  <chr>        <dbl>          <dbl>      <dbl>             <dbl>
1 emp_length  0.0495          0.135     0.144             0.0769
2 purpose     0.0959          0.167     0.0795            0.0900
```

The above R code chunk calculates the relative occurrence percentages of special values for character attributes in the dataset 'myLCdata_dirty'. After determining the total number of rows in the dataset, the counts of special values are converted into relative percentages by dividing each count by the total number of rows. Then, the column headers are renamed to reflect percentages.

Visualise Special Value Percentage

```r
# Convert data to a longer format
character_table_long <- pivot_longer(character_table_perc,
                                     cols = starts_with("perc_"),
                                     names_to = "Metric", values_to = "Percentage")

# Define colors for different metrics
char_metric_colors <- c("perc_NA" = "#A4D8D8", "perc_na_string" = "#A6CEE3",
                        "perc_space" = "#C1BCE4", "perc_empty_string" = "#CAB2D6")

# Define the desired order of metrics
desired_char_metric_order <- c("perc_NA", "perc_na_string", "perc_space", "perc_empty_string")

# Convert the Metric variable to a factor with the desired order
character_table_long$Metric <- factor(character_table_long$Metric, levels = desired_char_metric_order)

# Create plot with customized colors and sorted metrics
ggplot(character_table_long, aes(x = Attribute, y = Percentage, fill = Metric)) +
  geom_col(position = "dodge") +
  geom_text(aes(label = scales::percent(Percentage)), position = position_dodge(width = 0.9),
            hjust= 0 , size = 3) +  # Add text labels with percentage values
  coord_flip() +
  scale_y_continuous(labels = scales::percent,
                     limits = c(0, 0.2)) +  # Set limits for the y-axis
  labs(title = "Percentage of special values in the character attributes of 'myLCdata_dirty'",
       x = NULL, y = NULL) +  # Remove axis titles
  scale_fill_manual(values = char_metric_colors) +
  facet_wrap(~ Metric, scales = "free_y", ncol = 1) +
  theme(legend.position = "none")  # Remove legend
```

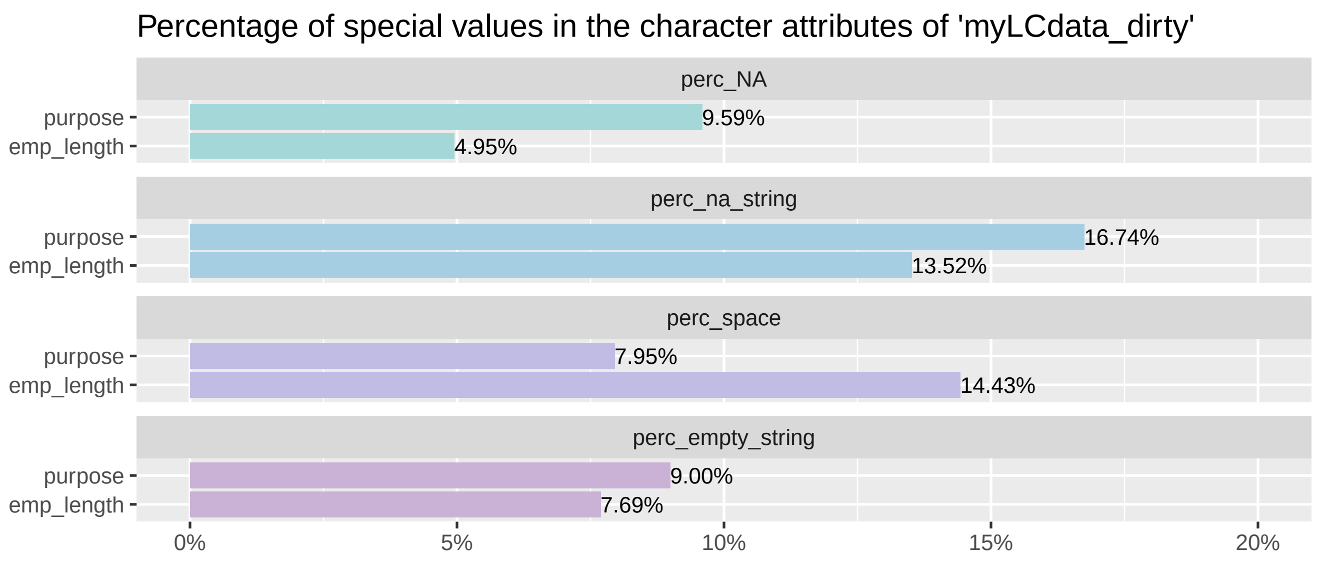

Figure 2: Grouped bar plot showing the distribution of special values in character attributes.

The above R code chunk generates a grouped bar chart displayed in Figure 2. Each bar in the chart represents the percentage of special values for every character attribute in the dataset, grouped according to the type of special value.

Numeric Attributes

In this part, the occurrences of special values (NA, NaN, outliers) in the attributes of data type numeric are counted. The percentages are calculated based on the total number of occurrences in relation to the total number of rows in the dataset, and they are visualised in a grouped bar chart.

Compute Special Value Counts

```r
# Calculate special value counts for numeric attributes
numeric_counts <- myLCdata_dirty %>%
  summarise(
    across(
      where(is.numeric), # Operate only on numeric columns
      list(
        count_NA = ~sum(is.na(.)), # count NAs
        count_NaN = ~sum(is.nan(.), na.rm = TRUE), #  count NANs
        count_outliers = ~sum(. < boxplot.stats(.)$stats[1] |
                              . > boxplot.stats(.)$stats[5], na.rm = TRUE) #  count outliers
      ), .names = "{.col}.{fn}" # Name resulting columns (column name . function name)
    )
  )

# Print the resulting dataframe
print(numeric_counts)
```

```
  mths_since_last_delinq.count_NA mths_since_last_delinq.count_NaN
1                           27810                             5954
  mths_since_last_delinq.count_outliers open_acc.count_NA open_acc.count_NaN
1                                     8             13283               8644
  open_acc.count_outliers tot_cur_bal.count_NA tot_cur_bal.count_NaN
1                     977                17328                  8501
  tot_cur_bal.count_outliers
1                        923
```

The above R code chunk computes the counts of special values for numeric attributes in the dataset 'myLCdata_dirty'. It specifically counts the occurrences of NA (Not Available), NaNs (Not a Number), and outliers (Tukey's fences). The code utilises the dplyr package to summarise the data, employing functions like `summarise()` and `across()` to calculate counts for each type of special value. Finally, the results are displayed on the console using `print()`.

For calculations, such as counting NANs and outliers in numeric columns, NA values need to be excluded before performing the calculations. This is because functions like `sum()` and `boxplot.stats()` cannot handle NA values by default, and including them would lead to inaccurate calculations. By setting `na.rm = TRUE`, it ensures that NA values are not considered in the calculations, thereby allowing accurate results to be obtained.

Restructure DataFrame Format

```r
# Pivot the dataframe to a longer format using pivot_longer()
numeric_table_counts <- numeric_counts %>%
  pivot_longer(
    cols = everything(), # Pivot all columns
    names_to ="Attribute",# Create a new column 'Attribute' to store the attribute names
    values_to ="Value"# Create a new column 'Value' to store the counts
  ) %>%

  # Separate the 'Attribute' column into 'Attribute' and 'Calculation' columns
  separate(Attribute, into = c("Attribute", "Calculation"), sep = "\\.", remove = FALSE) %>%

  # Pivot the dataframe back to a wider format using pivot_wider()
  pivot_wider(
    names_from ="Calculation",# Use the 'Calculation' column values as new column names
    values_from ="Value"# Use the 'Value' column values as counts for each calculation type
  ) %>%

  arrange(Attribute) # Arrange the dataframe by attribute name

# Print the resulting dataframe
print(numeric_table_counts)
```

```
# A tibble: 3 x 4
  Attribute              count_NA count_NaN count_outliers
  <chr>                     <int>     <int>          <int>
1 mths_since_last_delinq    27810      5954              8
2 open_acc                  13283      8644            977
3 tot_cur_bal               17328      8501            923
```

The above R code chunk pivots the dataframe to the desired format for analysis. Initially, it utilises `pivot_longer()` to convert the dataframe into a longer format, separating the ‘Attribute’ and ‘Calculation’ columns. Subsequently, it applies `pivot_wider()` to revert the dataframe back to a wider format, organising the counts by each type of special value. Finally, the resulting dataframe, ‘numeric_table_counts’, is printed to the console.

Calculate Special Value Percentage

```r
# Calculate the total number of rows in myLCdata_dirty
total_rows <- nrow(myLCdata_dirty)

# Convert counts to relative counts
numeric_table_perc <- numeric_table_counts %>%
  mutate(across(
    where(is.numeric), # Select numeric columns
    ~ ./total_rows # Divide each count by the total number of rows
  ))

# Rename the headers replacing "count" with "perc"
numeric_table_perc <- numeric_table_perc %>%
  rename_with(~ gsub("count", "perc", .x), contains("count"))

# Print the resulting dataframe with relative counts
print(numeric_table_perc)
```

```
# A tibble: 3 x 4
  Attribute              perc_NA perc_NaN perc_outliers
  <chr>                    <dbl>    <dbl>         <dbl>
1 mths_since_last_delinq   0.627    0.134      0.000180
2 open_acc                 0.299    0.195      0.0220
3 tot_cur_bal              0.391    0.192      0.0208
```

The above R code chunk calculates the relative occurrence percentages of special values for numeric attributes in the dataset 'myLCdata_dirty'. After determining the total number of rows in the dataset, the counts of special values are converted into relative percentages by dividing each count by the total number of rows. Then, the column headers are renamed to reflect percentages.

Visualise Special Value Percentage

```
# Convert data to a longer format
numeric_table_long <- pivot_longer(numeric_table_perc, cols = starts_with("perc_"),
                                   names_to = "Metric", values_to = "Percentage")

# Define colors for different metrics
num_metric_colors <- c("perc_NA" = "#A4D8D8", "perc_NaN" = "#A6CEE3",
                    "perc_outliers" = "#C1BCE4")

# Define the desired order of metrics
desired_num_metric_order <- c("perc_NA", "perc_NaN", "perc_outliers")

# Convert the Metric variable to a factor with the desired order
numeric_table_long$Metric <- factor(numeric_table_long$Metric, levels = desired_num_metric_order)

# Create plot with customized colors and sorted metrics
ggplot(numeric_table_long, aes(x = Attribute, y = Percentage, fill = Metric)) +
  geom_col(position = "dodge") +
  geom_text(aes(label = scales::percent(Percentage)), position = position_dodge(width = 0.9),
            hjust= 0 , size = 3) +  # Add text labels with percentage values
  coord_flip() +
  scale_y_continuous(labels = scales::percent,
                     breaks = seq(0, 1, by = 0.05), # Axis labels in 5% increments
                     limits = c(0, 0.65)) +  # Set limits for the y-axis
  labs(title = "Percentage of special values in the numeric attributes of 'myLCdata_dirty'",
       x = NULL, y = NULL) +  # Remove axis titles
  scale_fill_manual(values = num_metric_colors) +
  facet_wrap(~ Metric, scales = "free_y", ncol = 1) +
  theme(legend.position = "none")  # Remove legend
```

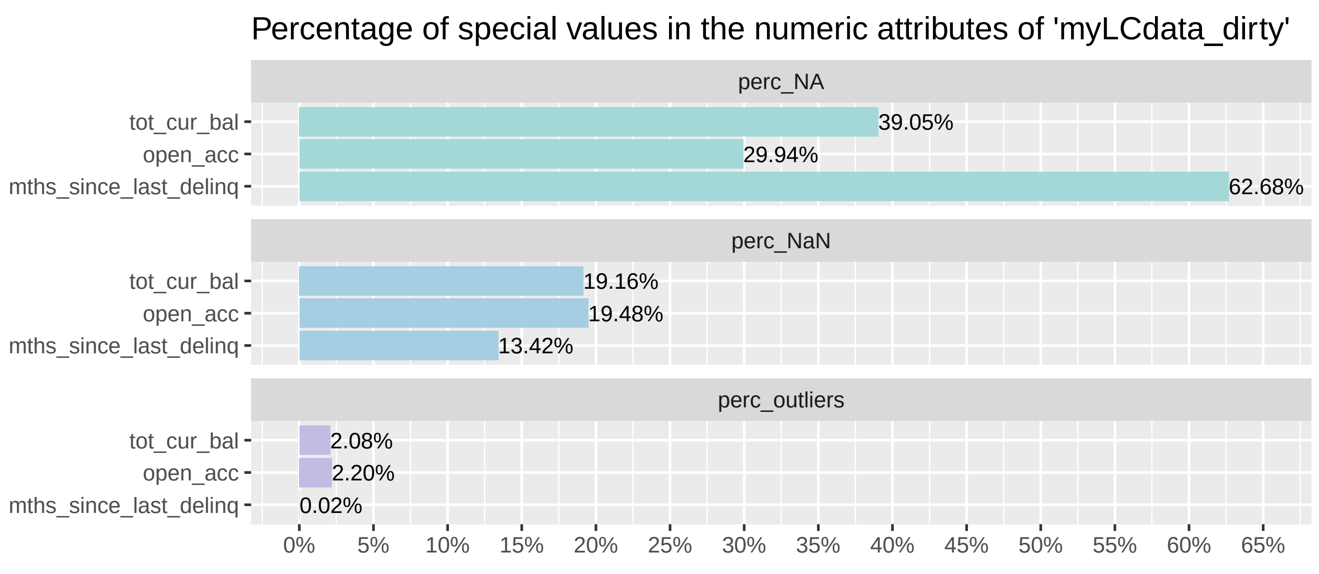

Figure 3: Grouped bar plot showing the distribution of special values in numeric attributes.

The above R code chunk generates a grouped bar chart displayed in Figure 3. Each bar in the chart represents the percentage of special values for every numeric attribute in the dataset, grouped according to the type of special value.

Conclusion

In conclusion, this project aimed to illustrate the process of data wrangling using R, focusing on identifying and handling special values within a dataset. By intentionally contaminating the dataset with various special values and employing techniques from the dplyr and tidyr packages, the special values were successfully counted and the percentages visualized. The analysis revealed that special values such as NA, “n/a” strings, spaces, and empty strings were present in attributes of data type character, while NA, NaNs, and outliers were present in numeric attributes. The generated grouped bar charts provided clear insights into the relative occurrence of these special values across different attributes.utes.