Data Wrangling Assessment Task 3: Dataset challenge

Aliia Gismatullina s4051304

Table of Contents

Assessment Brief

For this assessment, the task at hand involves producing an R markdown report and an accompanying overview presentation, providing an opportunity to refine proficiency in R programming. Moreover, it necessitates the articulation of well-founded justifications and explanations for the processes implemented.

Building on the foundation laid in previous assessments, the primary objective is to transform disorderly datasets, strategically addressing challenges such as missing values and outliers. This assignment aligns directly with several key course learning outcomes (RMIT University, School of Science, 2023):

- Utilise leading open-source software, R, to address and resolve data wrangling tasks.
- Select, perform, and justify data validation processes for raw datasets to satisfy quality requirements.
- Apply and evaluate the best practice standards of Tidy Data Principles.
- Critically analyse data integration procedures for combining data with different types and structures into a suitable format.

These outcomes serve as a framework for evaluating the success and effectiveness of the applied methodologies in this data pre-processing task.

Setup

```
library(tidyverse)
library(readr)
library(readxl)
library(stringr)
```

```r
library(lubridate)
library(reshape2)
library(patchwork)
library(imputeTS)
library(car)
library(gridExtra)
library(pscl)
```

Data Description

For this Assessment, I am working with two datasets prepared by The Bureau of Transport and Regional Economics (BTRE). These datasets contain time series information on international airlines operating to and from Australia.

The first dataset covers the years 1999 to 2003, while the second spans from 2004 to 2008. I will be merging these two separate datasets into a single comprehensive data frame. This combined dataset will capture data on international airline flights to and from Australia, providing a more extensive dataset for analysis covering the period from 1999 to 2008.

The data includes passenger counts, freight weights, and mail movements, as well as flight details, available seats, and seat utilization. The information is broken down by airline, country, and city for easier analysis.

The variables are as following:

- Month - the observation month;
- Scheduled Operator - International airline name;
- Country to/from - Country of port (inbound/outbound);
- Passengers In - Passengers flying to Australia;
- Freight In - Freight inbound to Australia in tonnes;
- Mail In - Mail inbound to Australia in tonnes;
- Passengers Out - Passengers flying out of Australia;
- Freight Out - Freight outbound from Australia in tonnes;
- Mail Out - Mail outbound from Australia in tonnes;
- Year - the observation year.

BTRE (2023). International airline activity Table1 2004to2008 web.xls. [ReadMe], B16-B24. Available at: https://www.bitre.gov.au/publications/ongoing/international_airline_activity-time_series.

To start with, I will import the MS Excel files.

```r
# To import the data, I will use the 'readxl' package.

df1 <- read_excel("data/International_airline_activity_Table1_99to03.xls",
    sheet = "Data")
df2 <- read_excel("data/International_airline_activity_Table1_2004to2008_web.xls",
    sheet = "Data")

head(df1)

## # A tibble: 6 × 10
##   Month               `Scheduled Operator`  `Country to/from` `Passengers In`
##   <dttm>              <chr>                 <chr>             <chr>
## 1 1999-01-01 00:00:00 Aerolineas Argentinas Argentina         2266
## 2 1999-01-01 00:00:00 Aerolineas Argentinas New Zealand       798
## 3 1999-01-01 00:00:00 Air Caledonie         New Caledonia     4011
## 4 1999-01-01 00:00:00 Air China             China             2890
## 5 1999-01-01 00:00:00 Air Mauritius         Mauritius         1744
## 6 1999-01-01 00:00:00 Air Nauru             Nauru             844
## # ℹ 6 more variables: `Freight In` <chr>, `Mail In` <chr>,
## #   `Passengers Out` <chr>, `Freight Out` <chr>, `Mail Out` <chr>,
## #   CalYear <dbl>

head(df2)

## # A tibble: 6 × 10
##   Month               `Scheduled Operator`  `Country to/from` `Passengers In`
```

```
##   <dttm>              <chr>                <chr>             <chr>
## 1 2004-01-01 00:00:00 Aerolineas Argentinas Argentina         2259
## 2 2004-01-01 00:00:00 Aerolineas Argentinas New Zealand       402
## 3 2004-01-01 00:00:00 Air Caledonie        New Caledonia     5361
## 4 2004-01-01 00:00:00 Air Canada           Canada            7248
## 5 2004-01-01 00:00:00 Air Canada           USA               3159
## 6 2004-01-01 00:00:00 Air China            China             8392
## # ℹ 6 more variables: `Freight In` <chr>, `Mail In` <chr>,
## #   `Passengers Out` <chr>, `Freight Out` <chr>, `Mail Out` <chr>, Year <dbl>
```

As we can see, the datasets have 9 common variables out of 10:

- 'df1' has a CalYear;
- 'df2' has the Year.

In fact, they mean the same, only the naming varies. Before proceeding with the merging, I find it reasonable to rename the 'CalYear' variable in df1 to 'Year' to match with the df2.

```
# For renaming a variable, I will use 'rename()' function from dplyr package.

df1 <- df1 %>% rename(Year = CalYear)
head(df1)

## # A tibble: 6 × 10
##   Month               `Scheduled Operator`  `Country to/from` `Passengers In`
##   <dttm>              <chr>                <chr>             <chr>
## 1 1999-01-01 00:00:00 Aerolineas Argentinas Argentina         2266
## 2 1999-01-01 00:00:00 Aerolineas Argentinas New Zealand       798
## 3 1999-01-01 00:00:00 Air Caledonie        New Caledonia     4011
## 4 1999-01-01 00:00:00 Air China            China             2890
## 5 1999-01-01 00:00:00 Air Mauritius        Mauritius         1744
## 6 1999-01-01 00:00:00 Air Nauru            Nauru             844
## # ℹ 6 more variables: `Freight In` <chr>, `Mail In` <chr>,
## #   `Passengers Out` <chr>, `Freight Out` <chr>, `Mail Out` <chr>, Year <dbl>
```

Next, I will join these 2 datasets into 1 and name it "airline_df", using the 'full_join' function. A full outer join returns all rows from both data frames, matching them where possible and filling in missing values with NA where there is no match. This ensures that all the information is retained from both datasets.

```
# I will use 'full_join' function, as I need to include all rows from both datasets.

airline_df <- full_join(df1, df2, by = join_by(Month, `Scheduled Operator`, `Country to/from`, `Passengers In`, `Freight In`, `Mail In`, `Passengers Out`, `Freight Out`, `Mail Out`, Year))

head(airline_df)

## # A tibble: 6 × 10
##   Month               `Scheduled Operator`  `Country to/from` `Passengers In`
##   <dttm>              <chr>                <chr>             <chr>
## 1 1999-01-01 00:00:00 Aerolineas Argentinas Argentina         2266
## 2 1999-01-01 00:00:00 Aerolineas Argentinas New Zealand       798
## 3 1999-01-01 00:00:00 Air Caledonie        New Caledonia     4011
## 4 1999-01-01 00:00:00 Air China            China             2890
## 5 1999-01-01 00:00:00 Air Mauritius        Mauritius         1744
## 6 1999-01-01 00:00:00 Air Nauru            Nauru             844
## # ℹ 6 more variables: `Freight In` <chr>, `Mail In` <chr>,
## #   `Passengers Out` <chr>, `Freight Out` <chr>, `Mail Out` <chr>, Year <dbl>
```

Understanding and Tidying the Data

Now, let's examine and comprehend the resultant dataset. Initially, I'd like to ascertain the total count of variables, observations, and data types. The most effective method to obtain this information is by employing the str() function.

```
str(airline_df)

## tibble [11,966 × 10] (S3: tbl_df/tbl/data.frame)
##  $ Month             : POSIXct[1:11966], format: "1999-01-01" "1999-01-01" ...
##  $ Scheduled Operator: chr [1:11966] "Aerolineas Argentinas" "Aerolineas Argentinas" "Air Caledonie" "Air China" ...
##  $ Country to/from   : chr [1:11966] "Argentina" "New Zealand" "New Caledonia" "China" ...
##  $ Passengers In     : chr [1:11966] "2266" "798" "4011" "2890" ...
##  $ Freight In        : chr [1:11966] "7.4779999999999998" "33.274000000000001" "6.5910000000000002" "40.369999999999997"
...
##  $ Mail In           : chr [1:11966] "1.827" "0" "0.60999999999999999" "2.6150000000000002" ...
##  $ Passengers Out    : chr [1:11966] "1689" "713" "2882" "2623" ...
##  $ Freight Out       : chr [1:11966] "59.137" "24.882000000000001" "20.407" "112.07899999999999" ...
##  $ Mail Out          : chr [1:11966] "1.0640000000000001" "0" "3.032" "3.3700000000000001" ...
##  $ Year              : num [1:11966] 1999 1999 1999 1999 1999 ...
```

Tidy Data Principles

1. Overview

Subsequently, there are 11,966 observations and 10 variables. The initial observation reveals that the data does not adhere to the Tidy Data Principles. Let’s go step by step:

- “Each variable forms a column”: The first variable “Month” contains both year and month values, which is against the Tidy Data Principles. To solve this, we can retain only the month values and eliminate the year component, considering the presence of a dedicated “Year” variable in the dataset.

- “Each observation forms a row”: Consequently, after we modify the “Month” variable as mentioned above, this Principle will be valid, as the rest of the rows represent a unique observation.

- “Each type of observational unit forms a table”: Data pertaining to a specific observation is contained within its own table, avoiding mixing multiple types of data within the same table.

- “Variable names are informative and not too long”: Variable names should be clear, concise, and descriptive - in our case, the variable names require formatting, as there are spaces ” ” and special characters “/” present in the dataset. I will address this in the next step.

- “Data is organized to facilitate analysis”: The structure of the dataset should be optimized for analysis, with clear relationships between variables and observations. In our case, almost all the variable types are characters, instead of numeric types. To adhere to Tidy Data Principles, these should be converted to numeric types (integers or doubles) since they represent numerical quantities.

2. Variable names

Handling spaces ” ” in variable names can pose challenges during analysis. I propose changing all spaces to underscores “_” for consistency and ease of analysis.

```
# Replacing spaces with underscores in variable names. Here, gsub(" ", "_", .) represents a regular expression substitution;
and 'everything()' represents modification of the names in all columns.

airline_df <- airline_df %>%
  rename_with(~ gsub(" ", "_", .), everything())
```

Additionally, a variable “Country_to/from” could potentially cause issues in certain situations due to a special charachter “/”, and it might be more convenient to rename it to “Country_To_From”.

```
# Rename the specific variable "Country_to/from" to "Country_To_From"
airline_df <- airline_df %>%
  rename("Country_To_From" = "Country_to/from")

head(airline_df)

## # A tibble: 6 × 10
##   Month               Scheduled_Operator    Country_To_From Passengers_In
```

```
##   <dttm>              <chr>                 <chr>           <chr>
## 1 1999-01-01 00:00:00 Aerolineas Argentinas Argentina       2266
## 2 1999-01-01 00:00:00 Aerolineas Argentinas New Zealand     798
## 3 1999-01-01 00:00:00 Air Caledonie         New Caledonia   4011
## 4 1999-01-01 00:00:00 Air China             China           2890
## 5 1999-01-01 00:00:00 Air Mauritius         Mauritius       1744
## 6 1999-01-01 00:00:00 Air Nauru             Nauru           844
## # ℹ 6 more variables: Freight_In <chr>, Mail_In <chr>, Passengers_Out <chr>,
## #   Freight_Out <chr>, Mail_Out <chr>, Year <dbl>
```

As a result, we have consistent and clear variable names.

3. Variable Types

All the variable types are currently set as characters, requiring modification. In particular:

- Month - (e.g. 1999-01-01) encompasses both year and month values, along with the first day of each month. Furthermore, it is presented in 'POSIXct' format, specifically designed for precise representation of date and time values. In our context, this level of precision is unnecessary, as we require solely the month value without the need for such detailed accuracy. And, as I mentioned above, the separate 'Year' variable already exists.

```
# Extracting the month value from a POSIXct date-time object using 'month' function from the 'lubridate' package.
airline_df$Month <- month(airline_df$Month)
unique(airline_df$Month)

##  [1]  1  2  3  4  5  6  7  8  9 10 11 12

str(airline_df$Month)

##  num [1:11966] 1 1 1 1 1 1 1 1 1 1 ...
```

The output above shows that the values have been successfully extracted.

When it comes to the data type of the variable "Month", I believe it represents a categorical aspect of the data, so converting it to a factor can be a good choice. Factors can help performing analyses that treat the months as distinct categories, such as seasonal patterns.

```
airline_df$Month <- factor(airline_df$Month)
str(airline_df$Month)

##  Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
```

- Scheduled_Operator, Country_To_From - can remain as a character, as it includes strings of names.

- Passengers_In, Passengers_Out - should be converted into integers, as they represent the whole numbers of passengers. To achieve this, I will use the 'str_detect' function from the 'stringr' package, where:

– "str_detect(Passengers_In,"\D")" checks if there are any non-digit characters in the "Passengers_In" column;
– The condition "is.na(Passengers_In) | str_detect(Passengers_In,"\D")" checks if the value is either NA or contains non-digit characters.
– If the condition is true, it replaces the value with NA using "ifelse".

```
# Using the 'str_detect' function from the 'stringr' package.

airline_df <- airline_df %>%
  mutate(Passengers_In = as.integer(ifelse(is.na(Passengers_In) | str_detect(Passengers_In, "\\D"), NA, Passengers_In)),
         Passengers_Out = as.integer(ifelse(is.na(Passengers_Out) | str_detect(Passengers_Out, "\\D"), NA, Passengers_Out)))

summary(airline_df$Passengers_In)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       0    1398    3574    9938    9581  134894    2376
```

```r
summary(airline_df$Passengers_Out)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       0    1341    3482    9730    9215  142993    2297
```

We can see that the variables have been successfully converted into integers, and there are over 2000 NA's in them.

- Freight In, Freight Out, Mail In, Mail Out - should be transformed into double data types, considering the given values are in tonnes. The best way to ensure that only valid numeric values are converted to double, and non-numeric or invalid values are replaced with NA, is using regular expressions.

– The regular expression "^\d+\.?\d*$" checks if the value is a non-negative decimal number (integer or floating).
– The condition "is.na(Freight_In) **|** !grepl("^\d+\.?\d*$", Freight_In)" checks if the value is either NA or does not match the specified pattern.
– If the condition is true, it replaces the value with NA using "ifelse".

```r
# Converting the variables using "as.double" function.
airline_df <- airline_df %>%
  mutate(Freight_In = as.double(ifelse(is.na(Freight_In) | !grepl("^\\d+\\.?\\d*$", Freight_In), NA, Freight_In)),
         Freight_Out = as.double(ifelse(is.na(Freight_Out) | !grepl("^\\d+\\.?\\d*$", Freight_Out), NA, Freight_Out)),
         Mail_In = as.double(ifelse(is.na(Mail_In) | !grepl("^\\d+\\.?\\d*$", Mail_In), NA, Mail_In)),
         Mail_Out = as.double(ifelse(is.na(Mail_Out) | !grepl("^\\d+\\.?\\d*$", Mail_Out), NA, Mail_Out)))
summary(airline_df[c("Freight_In", "Freight_Out", "Mail_In", "Mail_Out")])

##    Freight_In       Freight_Out         Mail_In           Mail_Out
##  Min.   :   0.00   Min.   :   0.00   Min.   :  0.000   Min.   :  0.000
##  1st Qu.:  11.55   1st Qu.:  12.34   1st Qu.:  0.000   1st Qu.:  0.000
##  Median :  92.89   Median :  59.72   Median :  0.171   Median :  0.000
##  Mean   : 339.62   Mean   : 284.43   Mean   : 15.835   Mean   : 11.908
##  3rd Qu.: 320.79   3rd Qu.: 238.24   3rd Qu.:  6.222   3rd Qu.:  2.699
##  Max.   :5921.89   Max.   :5910.32   Max.   :434.599   Max.   :492.977
##  NA's   :1017      NA's   :752       NA's   :1017      NA's   :752
```

The grouped summary above confirms the successful conversion of the variables.

- Year - should be a factor variable, and needs to be ordered, as it explicitly represents the ordinal nature of the years (1999-2008). By converting it to a factor and ordering the levels, it is specifying that the years have a meaningful order, rather than treating them as nominal categories.

Let's examine the summary for the 'Year' variable to confirm its range spanning from 1999 to 2008.

```r
summary(airline_df$Year)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1999    2001    2004    2003    2006    2008
```

I will use the 'mutate()' function from the 'dplyr' package to convert Year variable into a factor with levels.

```r
# Converting 'Year' to a factor and order the levels

airline_df <- airline_df %>%
  mutate(Year = factor(Year, levels = unique(Year), ordered = TRUE))

str(airline_df$Year)

##  Ord.factor w/ 10 levels "1999"<"2000"<..: 1 1 1 1 1 1 1 1 1 1 ...

summary(airline_df$Year)

## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008
## 1395 1265 1134 1028 1102 1187 1214 1200 1222 1219
```

As a result, we have the Year variable with 10 levels (from 1999 to 2008).

Let's now review the final data types we have successfully achieved through the above modifications.

```
str(airline_df)

## tibble [11,966 × 10] (S3: tbl_df/tbl/data.frame)
##  $ Month             : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Scheduled_Operator: chr [1:11966] "Aerolineas Argentinas" "Aerolineas Argentinas" "Air Caledonie" "Air China" ...
##  $ Country_To_From   : chr [1:11966] "Argentina" "New Zealand" "New Caledonia" "China" ...
##  $ Passengers_In     : int [1:11966] 2266 798 4011 2890 1744 844 58006 719 11124 3714 ...
##  $ Freight_In        : num [1:11966] 7.48 33.27 6.59 40.37 24.79 ...
##  $ Mail_In           : num [1:11966] 1.827 0 0.61 2.615 0.227 ...
##  $ Passengers_Out    : int [1:11966] 1689 713 2882 2623 1328 642 56422 607 8310 4400 ...
##  $ Freight_Out       : num [1:11966] 59.1 24.9 20.4 112.1 25.6 ...
##  $ Mail_Out          : num [1:11966] 1.064 0 3.032 3.37 0.013 ...
##  $ Year              : Ord.factor w/ 10 levels "1999"<"2000"<..: 1 1 1 1 1 1 1 1 1 1 ...
```

Consequently, the characher variables (Freight_In, Mail_In, Freight_Out, and Mail_Out) have been successfully parsed to double data types; "Passengers_In", "Passengers_Out" - to integers; the "Month" values have been extracted and labeled; and the "Year" variable is represented as an ordinal factor with 10 levels corresponding to the years from 1999 to 2008.

```
summary(airline_df)

##      Month      Scheduled_Operator Country_To_From    Passengers_In
##  3      :1021   Length:11966       Length:11966       Min.   :     0
##  12     :1005   Class :character   Class :character   1st Qu.:  1398
##  1      :1003   Mode  :character   Mode  :character   Median :  3574
##  2      :1001                                         Mean   :  9938
##  6      : 997                                         3rd Qu.:  9581
##  10     : 997                                         Max.   :134894
##  (Other):5942                                         NA's   :2376
##    Freight_In         Mail_In        Passengers_Out    Freight_Out
##  Min.   :   0.00   Min.   :  0.000   Min.   :     0   Min.   :   0.00
##  1st Qu.:  11.55   1st Qu.:  0.000   1st Qu.:  1341   1st Qu.:  12.34
##  Median :  92.89   Median :  0.171   Median :  3482   Median :  59.72
##  Mean   : 339.62   Mean   : 15.835   Mean   :  9730   Mean   : 284.43
##  3rd Qu.: 320.79   3rd Qu.:  6.222   3rd Qu.:  9215   3rd Qu.: 238.24
##  Max.   :5921.89   Max.   :434.599   Max.   :142993   Max.   :5910.32
##  NA's   :1017      NA's   :1017      NA's   :2297     NA's   :752
##     Mail_Out            Year
##  Min.   :  0.000   1999   :1395
##  1st Qu.:  0.000   2000   :1265
##  Median :  0.000   2007   :1222
##  Mean   : 11.908   2008   :1219
##  3rd Qu.:  2.699   2005   :1214
##  Max.   :492.977   2006   :1200
##  NA's   :752       (Other):4451
```

The summary above provides a quick snapshot of the distribution and characteristics of the data, where all the variables are in the correct type and all the Tidy Data Principles have been accomplished.

Manipulating Data

In this step of the assessment, I am required to create or mutate at least one variable from the existing ones. In alignment with this requirement, I propose to create a new variable named "Total_Passengers_Carried." This variable will signify the sum of passengers carried by the airline for a specific month, encompassing both inbound and outbound passenger movements.

```
# As the variables are now the correct type (integer), we can perform a basic mathematical operation.
airline_df <- airline_df %>%
  mutate(Total_Passengers_Carried = Passengers_In + Passengers_Out)

summary(airline_df$Total_Passengers_Carried)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       6    2837    7180   19790   18896  269640    2399
```

Following this, a similar methodology can be applied to generate another variable, "Total_Freight_Carried." This variable will encapsulate the aggregate sum of freight, measured in tonnes, encompassing both inbound and outbound shipments carried by the airline for a given month.

```
airline_df <- airline_df %>%
  mutate(Total_Freight_Carried = Freight_In + Freight_Out)

summary(airline_df$Total_Freight_Carried)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##     0.00    35.27   172.80   632.61   544.67 10070.31     1769

head(airline_df)

## # A tibble: 6 × 12
##   Month Scheduled_Operator    Country_To_From Passengers_In Freight_In Mail_In
##   <fct> <chr>                 <chr>                   <int>      <dbl>   <dbl>
## 1 1     Aerolineas Argentinas Argentina                2266       7.48   1.83
## 2 1     Aerolineas Argentinas New Zealand               798      33.3    0
## 3 1     Air Caledonie         New Caledonia            4011       6.59   0.61
## 4 1     Air China             China                    2890      40.4    2.62
## 5 1     Air Mauritius         Mauritius                1744      24.8    0.227
## 6 1     Air Nauru             Nauru                     844       0.631  0.086
## # ℹ 6 more variables: Passengers_Out <int>, Freight_Out <dbl>, Mail_Out <dbl>,
## #   Year <ord>, Total_Passengers_Carried <int>, Total_Freight_Carried <dbl>
```

As a result, there are now a total of 12 variables, each in the correct data type. The dataset is structured and organized, with each variable allocated to its own column and each observation to its own row.

Scan I. Missing Values

To scan all variables for missing values and inconsistencies in the dataset, I will use the following steps:

1. Firstly, I will use the 'summary' function to get a quick overview of missing values in each variable:

```
summary(airline_df)

##      Month      Scheduled_Operator Country_To_From    Passengers_In
##  3      :1021   Length:11966       Length:11966       Min.   :     0
##  12     :1005   Class :character   Class :character   1st Qu.:  1398
##  1      :1003   Mode  :character   Mode  :character   Median :  3574
##  2      :1001                                         Mean   :  9938
##  6      : 997                                         3rd Qu.:  9581
##  10     : 997                                         Max.   :134894
##  (Other):5942                                         NA's   :2376
##    Freight_In         Mail_In        Passengers_Out    Freight_Out
##  Min.   :   0.00   Min.   :  0.000   Min.   :     0   Min.   :   0.00
##  1st Qu.:  11.55   1st Qu.:  0.000   1st Qu.:  1341   1st Qu.:  12.34
##  Median :  92.89   Median :  0.171   Median :  3482   Median :  59.72
##  Mean   : 339.62   Mean   : 15.835   Mean   :  9730   Mean   : 284.43
##  3rd Qu.: 320.79   3rd Qu.:  6.222   3rd Qu.:  9215   3rd Qu.: 238.24
##  Max.   :5921.89   Max.   :434.599   Max.   :142993   Max.   :5910.32
##  NA's   :1017      NA's   :1017      NA's   :2297     NA's   :752
##     Mail_Out           Year       Total_Passengers_Carried
##  Min.   :  0.000   1999   :1395   Min.   :     6
##  1st Qu.:  0.000   2000   :1265   1st Qu.:  2837
##  Median :  0.000   2007   :1222   Median :  7180
##  Mean   : 11.908   2008   :1219   Mean   : 19790
##  3rd Qu.:  2.699   2005   :1214   3rd Qu.: 18896
##  Max.   :492.977   2006   :1200   Max.   :269640
##  NA's   :752       (Other):4451   NA's   :2399
##  Total_Freight_Carried
##  Min.   :    0.00
##  1st Qu.:   35.27
##  Median :  172.80
##  Mean   :  632.61
##  3rd Qu.:  544.67
```

```
## Max.   :10070.31
## NA's   :1769
```

2. To get the total count of missing values for each variable, I will use the 'colSums()' function.

```
colSums(is.na(airline_df))

##                  Month      Scheduled_Operator          Country_To_From
##                      0                       0                        0
##          Passengers_In              Freight_In                  Mail_In
##                   2376                    1017                     1017
##         Passengers_Out             Freight_Out                 Mail_Out
##                   2297                     752                      752
##                   Year Total_Passengers_Carried    Total_Freight_Carried
##                      0                    2399                     1769
```

Here we can see that all the numeric variables have big amounts of missing values:

- Month: No missing values (0).
- Scheduled_Operator: No missing values (0).
- Country_To_From: No missing values (0).
- Passengers_In: 2376 missing values.
- Freight_In: 1017 missing values.
- Mail_In: 1017 missing values.
- Passengers_Out: 2297 missing values.
- Freight_Out: 752 missing values.
- Mail_Out: 752 missing values.
- Year: No missing values (0).
- Total_Passengers_Carried: 2399 missing values.
- Total_Freight_Carried: 1769 missing values.

Addressing a large number of missing values requires careful consideration and the chosen approach should align with the nature of the analysis. Several methods can be considered:

1. Imputation:

Estimating or predicting missing values based on the observed data - mean or median imputation, as well as regression imputation. Given the prevalence of missing values in our dataset, it's crucial to assess the impact of missingness on the analysis. We can employ imputation methods and compare the results with and without imputation. Consistent results may indicate that missing values are missing completely at random (MCAR).

2. Removing Rows with Missing Values:

It is reasonable to consider the removal of rows where missing values are simultaneously present in 4 or more variables (more than half). This decision is justified by the understanding that observations with extensive missing data across multiple variables may not contribute significantly to the valuable information essential for our analysis.

```
rows_to_delete <- which(rowSums(is.na(airline_df)) > 4)

# Deleting rows
airline_df <- airline_df[-rows_to_delete, ]

# Resetting row names
rownames(airline_df) <- NULL

summary(airline_df)

##      Month      Scheduled_Operator Country_To_From    Passengers_In
##  3      : 861   Length:10197       Length:10197       Min.   :     0
##  1      : 859   Class :character   Class :character   1st Qu.:  1414
##  2      : 855   Mode  :character   Mode  :character   Median :  3595
##  6      : 853                                         Mean   :  9962
##  4      : 849                                         3rd Qu.:  9614
```

```
##  5      : 849                                           Max.   :134894  
##  (Other):5071                                          NA's   :630     
##    Freight_In         Mail_In        Passengers_Out    Freight_Out    
##  Min.   :   0.00   Min.   :  0.000   Min.   :     2   Min.   :   0.00  
##  1st Qu.:  11.36   1st Qu.:  0.000   1st Qu.:  1378   1st Qu.:  13.72  
##  Median :  87.17   Median :  0.300   Median :  3532   Median :  61.91  
##  Mean   : 333.79   Mean   : 16.803   Mean   :  9823   Mean   : 298.82  
##  3rd Qu.: 282.92   3rd Qu.:  7.014   3rd Qu.:  9350   3rd Qu.: 242.57  
##  Max.   :5921.89   Max.   :434.599   Max.   :142993   Max.   :5910.32  
##                                      NA's   :625                       
##     Mail_Out            Year      Total_Passengers_Carried
##  Min.   :  0.000   1999   :1256   Min.   :     6          
##  1st Qu.:  0.000   2000   :1119   1st Qu.:  2837          
##  Median :  0.000   2008   :1066   Median :  7180          
##  Mean   : 13.088   2007   :1060   Mean   : 19790          
##  3rd Qu.:  3.438   2006   :1014   3rd Qu.: 18896          
##  Max.   :492.977   2005   :1012   Max.   :269640          
##                    (Other):3670   NA's   :630             
##  Total_Freight_Carried
##  Min.   :    0.00     
##  1st Qu.:   35.27     
##  Median :  172.80     
##  Mean   :  632.61     
##  3rd Qu.:  544.67     
##  Max.   :10070.31     
## 
```

```
colSums(is.na(airline_df))
```

```
##                    Month       Scheduled_Operator          Country_To_From 
##                        0                        0                        0 
##            Passengers_In               Freight_In                  Mail_In 
##                      630                        0                        0 
##           Passengers_Out              Freight_Out                 Mail_Out 
##                      625                        0                        0 
##                     Year Total_Passengers_Carried    Total_Freight_Carried 
##                        0                      630                        0
```

After deleting these rows, that were not informative for the analysis, we can see that there are now 630 NA's in Passengers_In, 625 NA's in Passengers_Out and 630 NA's in Total_Passengers_Carried left to further deal with.

3. Creating a Missingness Indicator:

Creating a binary indicator variable that flags whether a value is missing. This can be useful for understanding the impact of missingness on the analysis.

```
# Creating a missingness indicator for the numeric variables
airline_df$Passengers_Out_missing <- ifelse(is.na(airline_df$Passengers_Out), 1, 0)

airline_df$Passengers_In_missing <- ifelse(is.na(airline_df$Passengers_In), 1, 0)

airline_df$Total_Passengers_Carried_missing <- ifelse(is.na(airline_df$Total_Passengers_Carried), 1, 0)

head(airline_df)
```

```
## # A tibble: 6 × 15
##   Month Scheduled_Operator    Country_To_From Passengers_In Freight_In Mail_In
##   <fct> <chr>                 <chr>                   <int>      <dbl>   <dbl>
## 1 1     Aerolineas Argentinas Argentina                2266      7.48    1.83 
## 2 1     Aerolineas Argentinas New Zealand               798     33.3     0    
## 3 1     Air Caledonie         New Caledonia            4011      6.59    0.61 
## 4 1     Air China             China                    2890     40.4     2.62 
## 5 1     Air Mauritius         Mauritius                1744     24.8     0.227
## 6 1     Air Nauru             Nauru                     844      0.631   0.086
## # ℹ 9 more variables: Passengers_Out <int>, Freight_Out <dbl>, Mail_Out <dbl>,
## #   Year <ord>, Total_Passengers_Carried <int>, Total_Freight_Carried <dbl>,
## #   Passengers_Out_missing <dbl>, Passengers_In_missing <dbl>,
## #   Total_Passengers_Carried_missing <dbl>
```

Here I would like to filter out and remove the rows, where Passengers_Out and Passengers_In have missing values, and at the same time Freight_In, Freight_Out, Mail_In and Mail_Out equal 0, as these rows will not be informative as well.

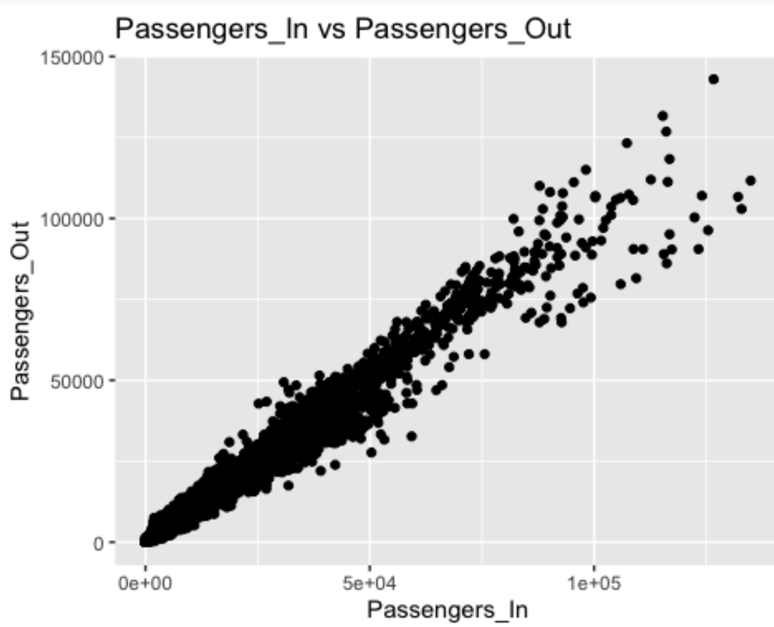
```
airline_df <- airline_df %>%
  filter(!(Passengers_Out_missing == 1 & Passengers_In_missing == 1 & Total_Passengers_Carried_missing == 1 &
           Freight_In == 0 & Freight_Out == 0 & Mail_In == 0 & Mail_Out == 0))

colSums(is.na(airline_df))

##                        Month               Scheduled_Operator
##                            0                                0
##              Country_To_From                    Passengers_In
##                            0                              497
##                   Freight_In                          Mail_In
##                            0                                0
##               Passengers_Out                      Freight_Out
##                          492                                0
##                     Mail_Out                             Year
##                            0                                0
##     Total_Passengers_Carried             Total_Freight_Carried
##                          497                                0
##       Passengers_Out_missing             Passengers_In_missing
##                            0                                0
## Total_Passengers_Carried_missing
##                            0
```

Now we can further investigate the relationship of these variables and take necessary actions to deal with the remaining missing values.

4. Identifying errors:

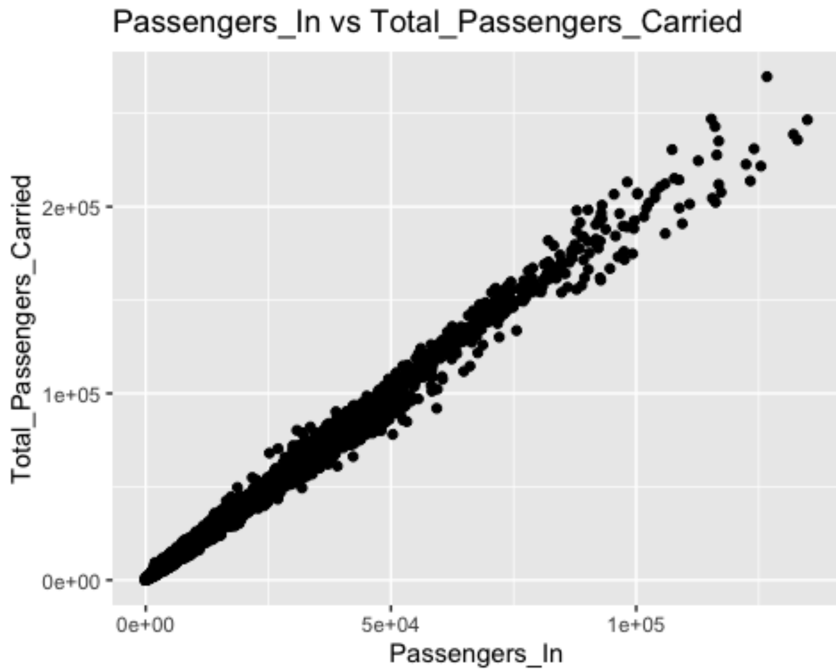
First, I would like to visualize these variables in correlation.

```
# Scatter plot for Passengers_In vs Passengers_Out
ggplot(airline_df, aes(x = Passengers_In, y = Passengers_Out)) +
  geom_point() +
  labs(title = "Passengers_In vs Passengers_Out",
       x = "Passengers_In", y = "Passengers_Out")

## Warning: Removed 497 rows containing missing values (`geom_point()`).
```

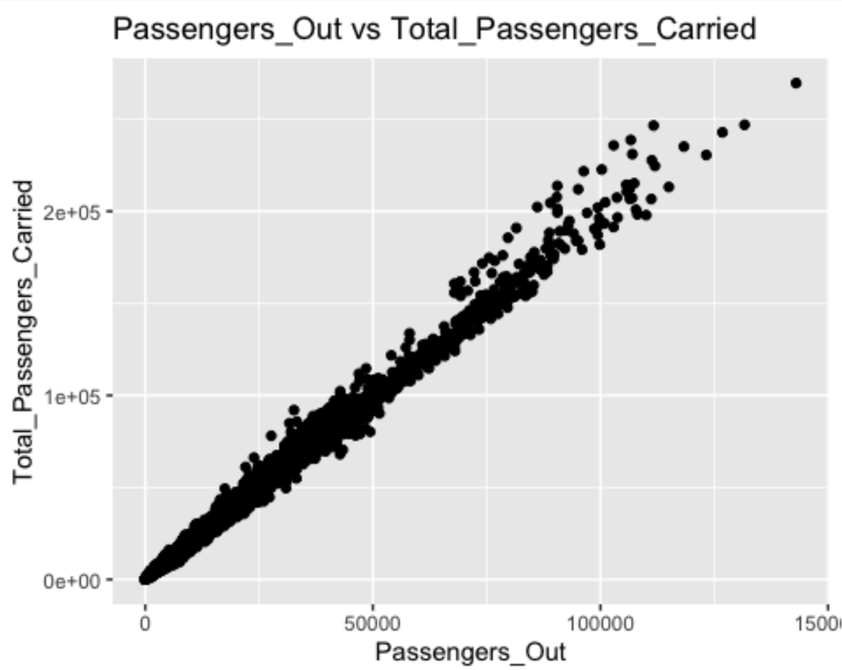



```
# Scatter plot for Passengers_In vs Total_Passengers_Carried
ggplot(airline_df, aes(x = Passengers_In, y = Total_Passengers_Carried)) +
  geom_point() +
```

```r
  labs(title = "Passengers_In vs Total_Passengers_Carried",
       x = "Passengers_In", y = "Total_Passengers_Carried")

## Warning: Removed 497 rows containing missing values (`geom_point()`).
```




```r
# Scatter plot for Passengers_Out vs Total_Passengers_Carried
ggplot(airline_df, aes(x = Passengers_Out, y = Total_Passengers_Carried)) +
  geom_point() +
  labs(title = "Passengers_Out vs Total_Passengers_Carried",
       x = "Passengers_Out", y = "Total_Passengers_Carried")

## Warning: Removed 497 rows containing missing values (`geom_point()`).
```



In the scatter plot we can clearly see the relationship of the variables, suggesting that they move together in a similar fashion, and changes in one are associated with changes in the others. However, upon analyzing the dataset I found that the Regression Imputation cannot be performed when dealing with missing values, as all 3 of them have the NA's on the same rows simultaneously. Therefore, I came to a conclusion that replacing the missing values with the mean values of the variables is the most suitable solution.

However, before calculating the mean values, it is essential to check if there are any outliers in these variables, as it may significantly affect the mean values. Let's take a quick glance at the data summary.

```
# Calculating summary statistics for Passengers_In
summary_passengers_in <- summary(airline_df$Passengers_In)

# Calculating summary statistics for Passengers_Out
summary_passengers_out <- summary(airline_df$Passengers_Out)

# Calculating summary statistics for Total_Passengers_Carried
summary_total_passengers_carried <- summary(airline_df$Total_Passengers_Carried)

# Displaying the results
print("Summary Statistics for Passengers_In:")

## [1] "Summary Statistics for Passengers_In:"

print(summary_passengers_in)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##       0    1414    3595    9962    9614  134894     497

print("Summary Statistics for Passengers_Out:")

## [1] "Summary Statistics for Passengers_Out:"

print(summary_passengers_out)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##       2    1378    3532    9823    9350  142993     492

print("Summary Statistics for Total_Passengers_Carried:")

## [1] "Summary Statistics for Total_Passengers_Carried:"

print(summary_total_passengers_carried)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##       6    2837    7180   19790   18896  269640     497
```

When examining the maximum values of the 'Passengers_In' a notable figure of 134,894 and 'Passengers_Out' max value of 142,993 caught my attention. I will select these values and analyse them.

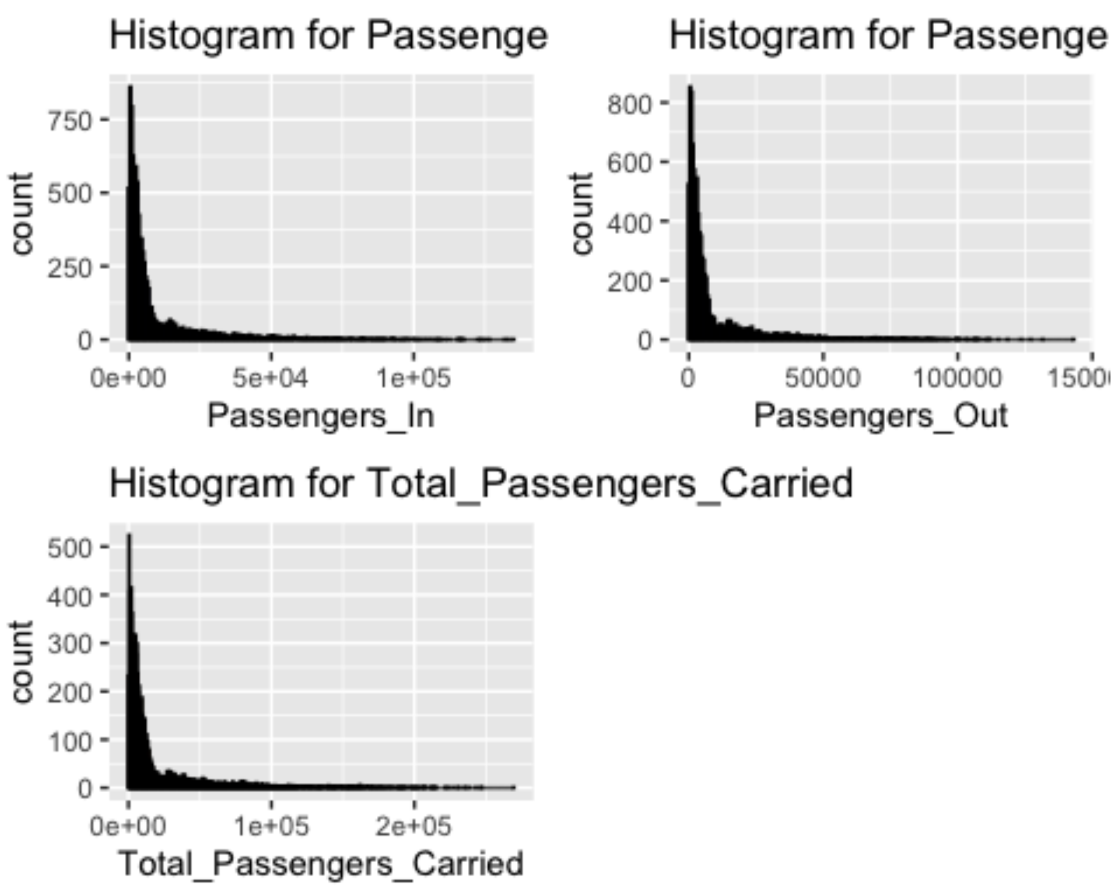
```
max_passengers_in_row <- airline_df %>%
  filter(Passengers_In == max(Passengers_In, na.rm = TRUE))

max_passengers_out_row <- airline_df %>%
  filter(Passengers_Out == max(Passengers_Out, na.rm = TRUE))

print(max_passengers_in_row)

## # A tibble: 1 × 15
##   Month Scheduled_Operator Country_To_From Passengers_In Freight_In Mail_In
##   <fct> <chr>              <chr>                   <int>      <dbl>   <dbl>
## 1 7     Singapore Airlines Singapore              134894      5922.    123.
## # ℹ 9 more variables: Passengers_Out <int>, Freight_Out <dbl>, Mail_Out <dbl>,
## #   Year <ord>, Total_Passengers_Carried <int>, Total_Freight_Carried <dbl>,
## #   Passengers_Out_missing <dbl>, Passengers_In_missing <dbl>,
## #   Total_Passengers_Carried_missing <dbl>

print(max_passengers_out_row)

## # A tibble: 1 × 15
##   Month Scheduled_Operator Country_To_From Passengers_In Freight_In Mail_In
##   <fct> <chr>              <chr>                   <int>      <dbl>   <dbl>
## 1 12    Singapore Airlines Singapore              126647      4373.    147.
## # ℹ 9 more variables: Passengers_Out <int>, Freight_Out <dbl>, Mail_Out <dbl>,
```

```
## #   Year <ord>, Total_Passengers_Carried <int>, Total_Freight_Carried <dbl>,
## #   Passengers_Out_missing <dbl>, Passengers_In_missing <dbl>,
## #   Total_Passengers_Carried_missing <dbl>
```

As we can see on the output, the maximum value of 'Passengers_In' happened in June, 2008. The research on the official Singapore Airlines website revealed that in 2008, the airline documented a '6.7% year-on-year growth' in passenger carriage (Singapore Company Registration, 2008). Notably, July 2008 coincided with the introduction of Airbus A380 aircraft. Subsequently, the maximum value of 'Passengers_Out' 142,993 happened in December, 2008, the same year with the above. In conclusion, the increase in passenger numbers in this case is logical.

As we proved that the max values are valid, we can continue with the scanning. For the next step, I would like to create histograms for these 3 variables and visually see the distribution of values.

```r
# Histogram for Passengers_In
hist_plot_passengers_in <- ggplot(airline_df, aes(x = Passengers_In)) +
  geom_histogram(binwidth = 500, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Histogram for Passengers_In")

# Plot for Passengers_Out
hist_plot_passengers_out <- ggplot(airline_df, aes(x = Passengers_Out)) +
  geom_histogram(binwidth = 500, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Histogram for Passengers_Out")

# Histogram for Total_Passengers_Carried
hist_plot_total_passengers_carried <- ggplot(airline_df, aes(x = Total_Passengers_Carried)) +
  geom_histogram(binwidth = 500, fill = "orange", color = "black", alpha = 0.7) +
  labs(title = "Histogram for Total_Passengers_Carried")

# Display the plots
grid.arrange(hist_plot_passengers_in, hist_plot_passengers_out, hist_plot_total_passengers_carried, ncol = 2)
```



The histograms look almost identical, right-skewed, and contain a lot of 0 values, so some data transformation can be relevant here to gain insights into the distribution of the data.

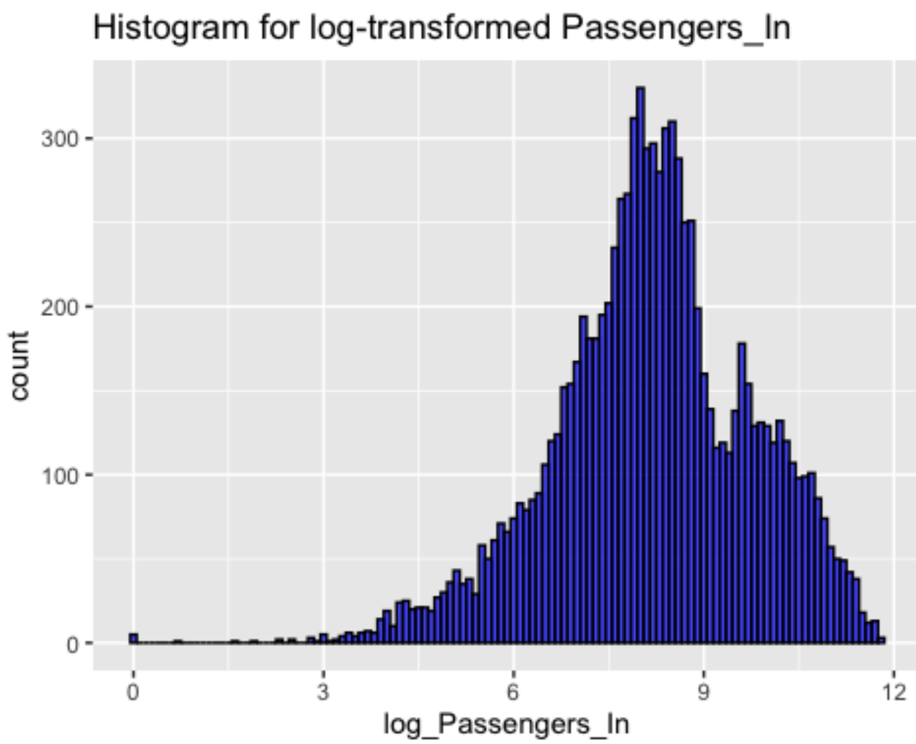
Based on the visualization above, replacing the missing values with the mean values does not seem to be the right approach, as the NA amounts are quite large - 497, 492 and 497. If we simply replace them with mean values (as an example, for 'Passengers_In', replacing 497 NA values with 9962), it will significantly affect the dataset and introduce bias.

In such a case, alternative imputation methods, such as K-Nearest Neighbors (KNN) imputation, might be more suitable, keeping in mind that the consecutive values have a linear relationship between them, as we proved earlier. To continue with the selected method, I will use Data Transformation technique as below.

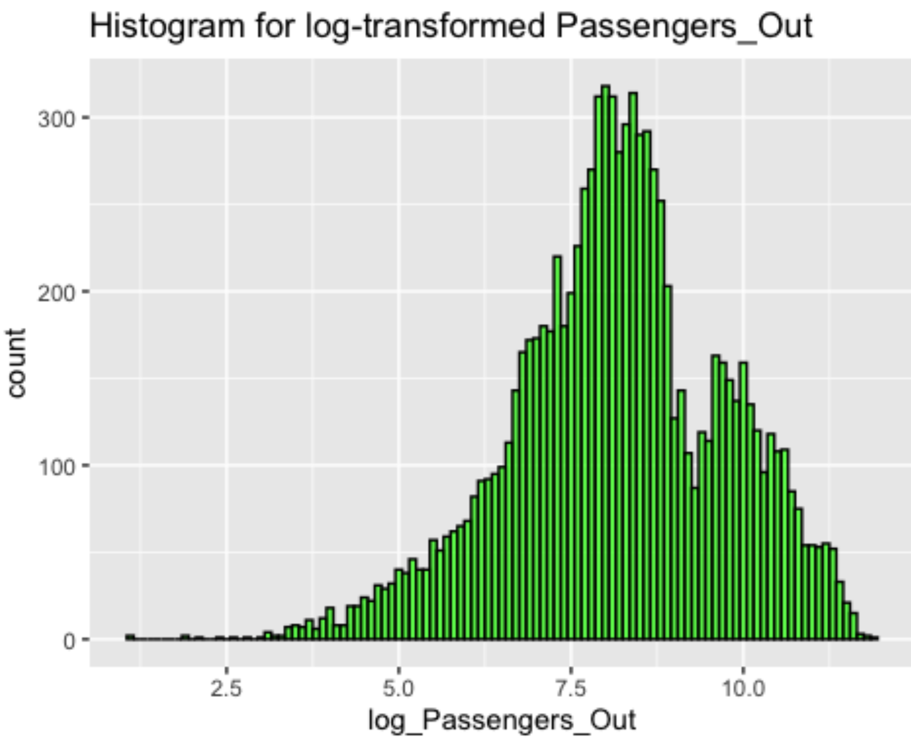
5. Data Transformation:

I will apply log transformations to the 3 variables to reduce right-skewness.

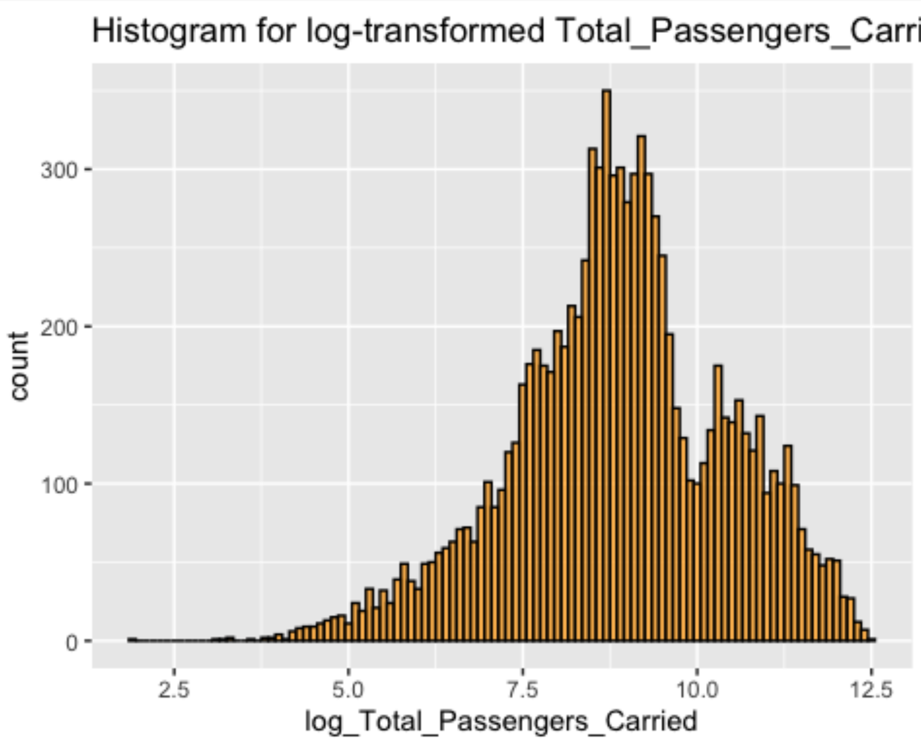
```
# Log-transforming the variables
airline_df$log_Passengers_In <- log1p(airline_df$Passengers_In)
airline_df$log_Passengers_Out <- log1p(airline_df$Passengers_Out)
airline_df$log_Total_Passengers_Carried <- log1p(airline_df$Total_Passengers_Carried)

# Histogram for log-transformed Passengers_In
ggplot(airline_df, aes(x = log_Passengers_In)) +
  geom_histogram(binwidth = 0.1, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Histogram for log-transformed Passengers_In")
```




```
# Histogram for log-transformed Passengers_Out
ggplot(airline_df, aes(x = log_Passengers_Out)) +
  geom_histogram(binwidth = 0.1, fill = "green", color = "black", alpha = 0.7) +
  labs(title = "Histogram for log-transformed Passengers_Out")

## Warning: Removed 492 rows containing non-finite values (`stat_bin()`).
```




```r
# Histogram for log-transformed Total_Passengers_Carried
ggplot(airline_df, aes(x = log_Total_Passengers_Carried)) +
  geom_histogram(binwidth = 0.1, fill = "orange", color = "black", alpha = 0.7) +
  labs(title = "Histogram for log-transformed Total_Passengers_Carried")
```

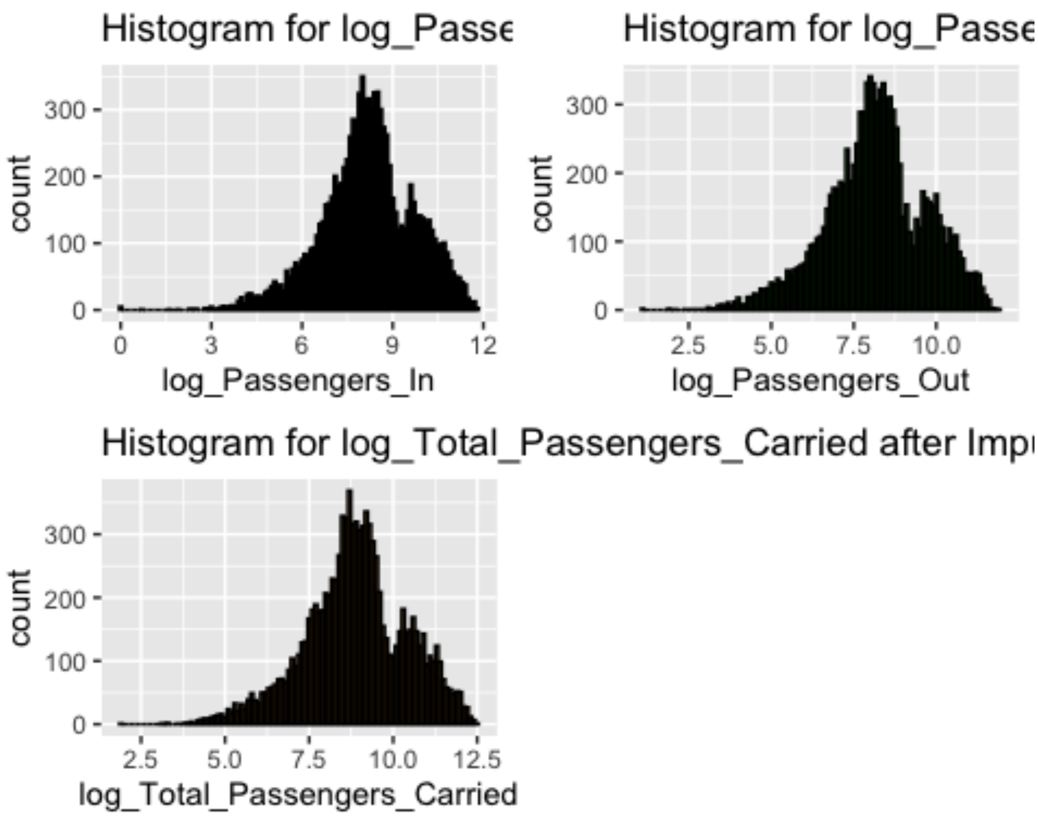


The transformed values look much clearer and provide more insights compared to the values prior to transformation.

Next, I will implement a K-Nearest Neighbors method, where it considers the values of the nearest neighbors to impute missing values.

```r
# Imputing missing values using linear interpolation
airline_df$log_Passengers_In <- na_interpolation(airline_df$log_Passengers_In, option = "linear")
```

```r
airline_df$log_Passengers_Out <- na_interpolation(airline_df$log_Passengers_Out, option = "linear")
airline_df$log_Total_Passengers_Carried <- na_interpolation(airline_df$log_Total_Passengers_Carried, option = "linear")


# Histogram for log_Passengers_In after imputation
hist_imputed_passengers_in <- ggplot(airline_df, aes(x = log_Passengers_In)) +
  geom_histogram(binwidth = 0.1, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Histogram for log_Passengers_In after Imputation")

# Histogram for log_Passengers_Out after imputation
hist_imputed_passengers_out <- ggplot(airline_df, aes(x = log_Passengers_Out)) +
  geom_histogram(binwidth = 0.1, fill = "green", color = "black", alpha = 0.7) +
  labs(title = "Histogram for log_Passengers_Out after Imputation")

# Histogram for log_Total_Passengers_Carried after imputation
hist_imputed_total_passengers_carried <- ggplot(airline_df, aes(x = log_Total_Passengers_Carried)) +
  geom_histogram(binwidth = 0.1, fill = "orange", color = "black", alpha = 0.7) +
  labs(title = "Histogram for log_Total_Passengers_Carried after Imputation")

# Displaying the histograms
grid.arrange(hist_imputed_passengers_in, hist_imputed_passengers_out, hist_imputed_total_passengers_carried, ncol = 2)
```



In the histograms now we see a positive outcome of K-Nearest Neighbors (KNN) method successfully handled missing values without significantly altering the distribution of the variables. This suggests that the imputed values align well with the patterns observed in the existing data.

```r
print("Summary Statistics for log_Passengers_In:")

## [1] "Summary Statistics for log_Passengers_In:"

summary(airline_df$log_Passengers_In)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   7.280   8.189   8.178   9.152  11.812

print("Summary Statistics for log_Passengers_Out:")
```

```
## [1] "Summary Statistics for log_Passengers_Out:"

summary(airline_df$log_Passengers_Out)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.099   7.258   8.173   8.165   9.122  11.871

print("Summary Statistics for log_Total_Passengers_Carried:")

## [1] "Summary Statistics for log_Total_Passengers_Carried:"

summary(airline_df$log_Total_Passengers_Carried)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.946   7.979   8.880   8.876   9.819  12.505
```

As we can see on the output, we have successfully eliminated the missing values in log transformed variables.

Scan II. Outliers

First of all, I would like to check the categorical variables for unique values and identify any inconsistencies, using the 'unique' function.

```
unique(airline_df$Scheduled_Operator)

##  [1] "Aerolineas Argentinas"      "Air Caledonie"               
##  [3] "Air China"                  "Air Mauritius"               
##  [5] "Air Nauru"                  "Air New Zealand"             
##  [7] "Air Niugini"                "Air Pacific"                 
##  [9] "Air Vanuatu"                "Air Zimbabwe"                
## [11] "Alitalia"                   "All Nippon Airways"          
## [13] "Ansett International"       "AOM French Airlines"         
## [15] "Asian Express Airlines"     "Asiana Airlines"             
## [17] "British Airways"            "Canadian Airlines Intl"      
## [19] "Cathay Pacific Airways"     "China Eastern Airlines"      
## [21] "Continental Micronesia"     "EgyptAir"                    
## [23] "Emirates"                   "Eva Air"                     
## [25] "Flight West Airlines"       "Freedom Air International"   
## [27] "Garuda Indonesia"           "Gulf Air"                    
## [29] "Japan Airlines"             "KLM Royal Dutch Airlines"    
## [31] "Korean Air"                 "Lan Chile"                   
## [33] "Lauda Air"                  "Malaysia Airlines"           
## [35] "Mandarin Airlines"          "Martinair Holland"           
## [37] "Merpati Nusantara Airlines" "Olympic Airways"             
## [39] "Polynesian Airlines"        "Qantas Airways"              
## [41] "Royal Brunei Airlines"      "Singapore Airlines"          
## [43] "Solomon Airlines"           "South African Airways"       
## [45] "Swissair"                   "Thai Airways International"  
## [47] "United Airlines"            "Virgin Atlantic Airways"     
## [49] "Polar Air Cargo"            "Royal Tongan Airlines"       
## [51] "China Airlines"             "Connie Kalitta Services"     
## [53] "SriLankan Airlines"         "Vietnam Airlines"            
## [55] "Philippine Airlines"        "China Southern Airlines"     
## [57] "Air Canada"                 "Lufthansa German Airlines"   
## [59] "Gemini Air Cargo"           "Australian Airlines"         
## [61] "Air Paradise International" "Virgin Australia"            
## [63] "Virgin Samoa"               "Hawaiian Airlines"           
## [65] "Valuair"                    "Transair"                    
## [67] "Air Tahiti Nui"             "Austrian Airlines"           
## [69] "Airlines PNG"               "Jetstar"                     
## [71] "Tiger Airways"              "Cargolux Airlines Intl"      
## [73] "JALways"                    "Pacific Air Express"         
## [75] "Etihad Airways"             "Airnorth"                    
## [77] "AirAsia X"                  "Silk Air"                    
## [79] "LAN Airlines"               "Our Airline"                 
## [81] "Tasman Cargo Airlines"      "SkyAirWorld"                 
## [83] "OzJet"

unique(airline_df$Country_To_From)

##  [1] "Argentina"            "New Zealand"          "New Caledonia"       
##  [4] "China"                "Mauritius"            "Nauru"
```

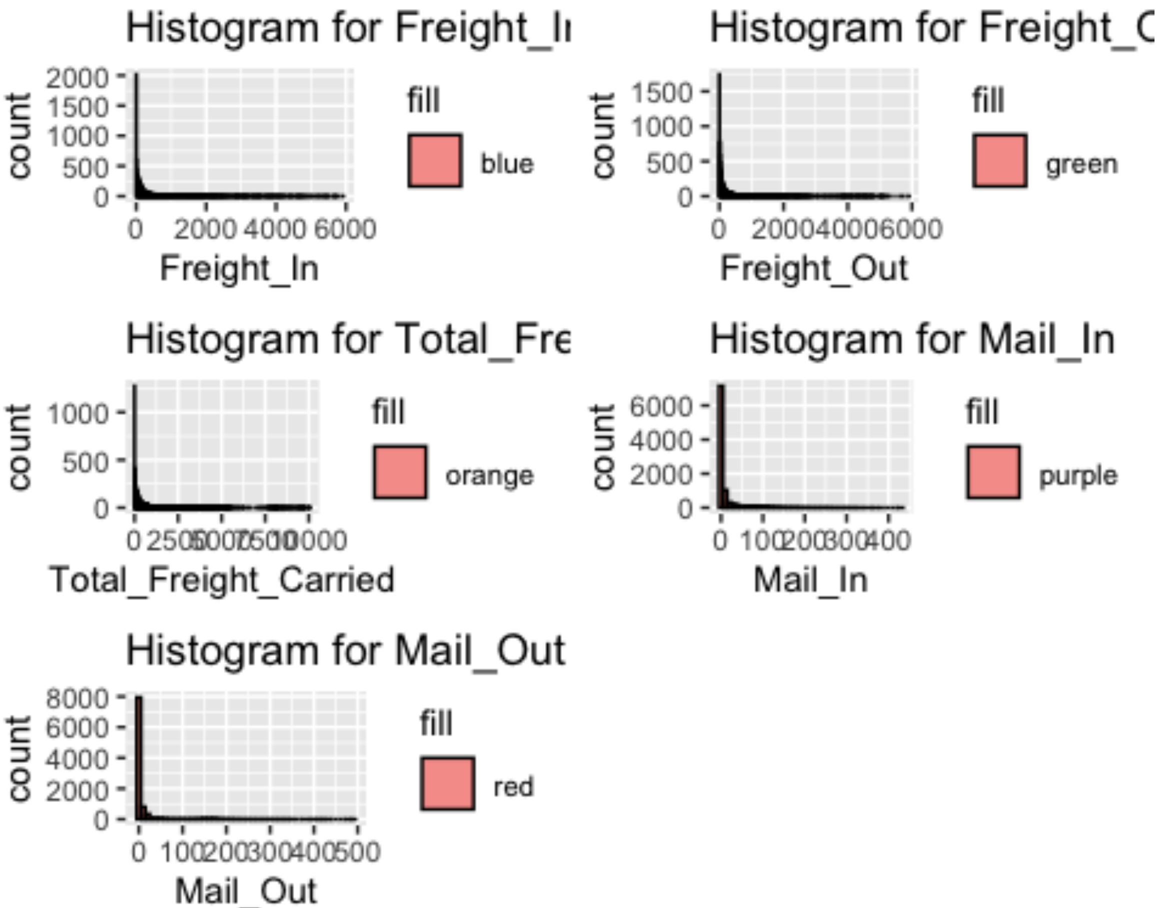
```
##  [7] "Taiwan"               "USA"                  "Papua New Guinea"    
## [10] "Fiji"                 "Vanuatu"              "Zimbabwe"            
## [13] "Italy"                "Singapore"            "Japan"               
## [16] "Hong Kong"            "Indonesia"            "Malaysia"            
## [19] "France"               "Sri Lanka"            "Korea"               
## [22] "Thailand"             "UK"                   "Canada"              
## [25] "Guam"                 "Egypt"                "United Arab Emirates"
## [28] "Bahrain"              "Netherlands"          "Chile"               
## [31] "Austria"              "Greece"               "Tonga"               
## [34] "Western Samoa"        "Germany"              "India"               
## [37] "Philippines"          "Solomon Islands"      "South Africa"        
## [40] "Switzerland"          "Tahiti"               "Vietnam"             
## [43] "Brunei"               "East Timor"           "Cook Islands"        
## [46] "Luxembourg"           "Hong Kong (SAR)"      "Kiribati"
```

The unique values for both of the variables look relevant, so we can continue with the numerical variables.

As we already scanned, identified and transformed the 3 numeric variables: 'Passengers_In', 'Passengers_Out' and 'Total_Passengers_Carried', I will do the similar to the rest of the numeric variables:

- Freight_In,
- Freight_Out,
- Total_Freight_Carried,
- Mail_In,
- Mail_Out.

```r
# Creating histograms for Freight_In, Freight_Out, Total_Freight_Carried, Mail_In, Mail_Out.
histogram_freight_in <- ggplot(airline_df, aes(x = Freight_In, fill = "blue")) +
  geom_histogram(binwidth = 10, color = "black", alpha = 0.7) +
  labs(title = "Histogram for Freight_In")

histogram_freight_out <- ggplot(airline_df, aes(x = Freight_Out, fill = "green")) +
  geom_histogram(binwidth = 10, color = "black", alpha = 0.7) +
  labs(title = "Histogram for Freight_Out")

histogram_total_freight_carried <- ggplot(airline_df, aes(x = Total_Freight_Carried, fill = "orange")) +
  geom_histogram(binwidth = 10, color = "black", alpha = 0.7) +
  labs(title = "Histogram for Total_Freight_Carried")

histogram_mail_in <- ggplot(airline_df, aes(x = Mail_In, fill = "purple")) +
  geom_histogram(binwidth = 10, color = "black", alpha = 0.7) +
  labs(title = "Histogram for Mail_In")

histogram_mail_out <- ggplot(airline_df, aes(x = Mail_Out, fill = "red")) +
  geom_histogram(binwidth = 10, color = "black", alpha = 0.7) +
  labs(title = "Histogram for Mail_Out")

# Displaying the histograms.
grid.arrange(histogram_freight_in, histogram_freight_out, histogram_total_freight_carried,
             histogram_mail_in, histogram_mail_out, ncol = 2)
```



The histograms generated lack informativeness due to a considerable number of instances with zero values (0). These zeros signify months where the airline exclusively transported passengers without any freight or mail, reflecting a real scenario. In order to reduce the impact of extreme values (such as 0), I suggest to apply Zero-Inflated models (Zero-Inflated Negative Binomial (ZINB) in particular) for handling a dataset a large number of zeros.

For fitting this model, I will use the 'pscl' package.

```
# Rounding 'Mail_In' to the nearest integer (as this model works with integers only).
airline_df$Rounded_Mail_In <- round(airline_df$Mail_In)

# Fitting a zero-inflated negative binomial model with the rounded variable with the independent predictor variable -
'Passengers_In'.
zinb_model <- zeroinfl(Rounded_Mail_In ~ Passengers_In, data = airline_df, dist = "negbin")

# Summary of the model
summary(zinb_model)

##
## Call:
## zeroinfl(formula = Rounded_Mail_In ~ Passengers_In, data = airline_df,
##     dist = "negbin")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -0.5782 -0.4799 -0.2811 -0.1309 11.9301
##
## Count model coefficients (negbin with log link):
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.770e+00  3.168e-02   55.88   <2e-16 ***
## Passengers_In  6.920e-05  9.175e-07   75.42   <2e-16 ***
## Log(theta)    -1.094e+00  2.176e-02  -50.28   <2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.725e+00  8.022e-02   33.96   <2e-16 ***
```

```
## Passengers_In -1.578e-03  3.331e-05  -47.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta = 0.3348
## Number of iterations in BFGS optimization: 20
## Log-likelihood: -2.461e+04 on 5 Df
```

Let's interpret the key insights from the zero-inflated negative binomial model:

- Excess Zeros: In our model, the excess zeros in the count of rounded mail items are not random; they are systematically influenced by certain factors.

- Association with the Number of Passengers:

The coefficient is 6.920e-05 for 'Passengers_In'

The presence of excess zeros is linked to the number of passengers. When the number of passengers increases, the odds of observing zero counts in rounded mail items also increase.

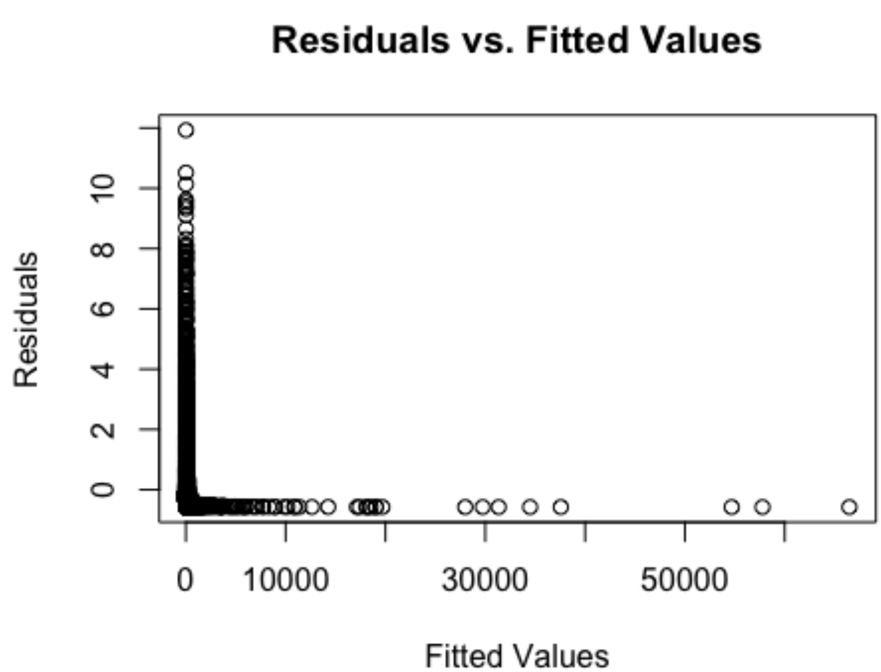
- Zero-Inflated Distribution:

Zero-Inflation Model Coefficients (binomial with logit link): The intercept for the zero-inflation model is estimated to be 2.725e+00. The coefficient for Passengers_In in the zero-inflation model is estimated to be -1.578e-03.

This pattern suggests a zero-inflated distribution, where zero counts are not just random occurrences but are influenced by specific predictors. In this case, the excess zeros are associated with the predictor variable Passengers_In.

Practically, this could mean that certain conditions related to the number of passengers lead to a higher likelihood of having no mail items (rounded to zero) in a given month, irrespective of other factors in the model.

```r
# Residuals vs. Fitted Values Plot
plot(zinb_model$fitted.values, resid(zinb_model),
     xlab = "Fitted Values", ylab = "Residuals",
     main = "Residuals vs. Fitted Values")
```



Analysing the plot, there is a point in Fitted values, that can be an outlier (< 60000). I suggest to identify this specific data point corresponding to the outlier in the fitted values, following these steps:

1. Identifying Outlier Index:

```
# The fitted values are stored in zinb_model$fitted.values
outlier_index <- which(zinb_model$fitted.values > 60000)

# Print the index
print(outlier_index)

## 9611
## 9142
```

2. Retrieving the corresponding rows from the dataset - 9611 and 9142:

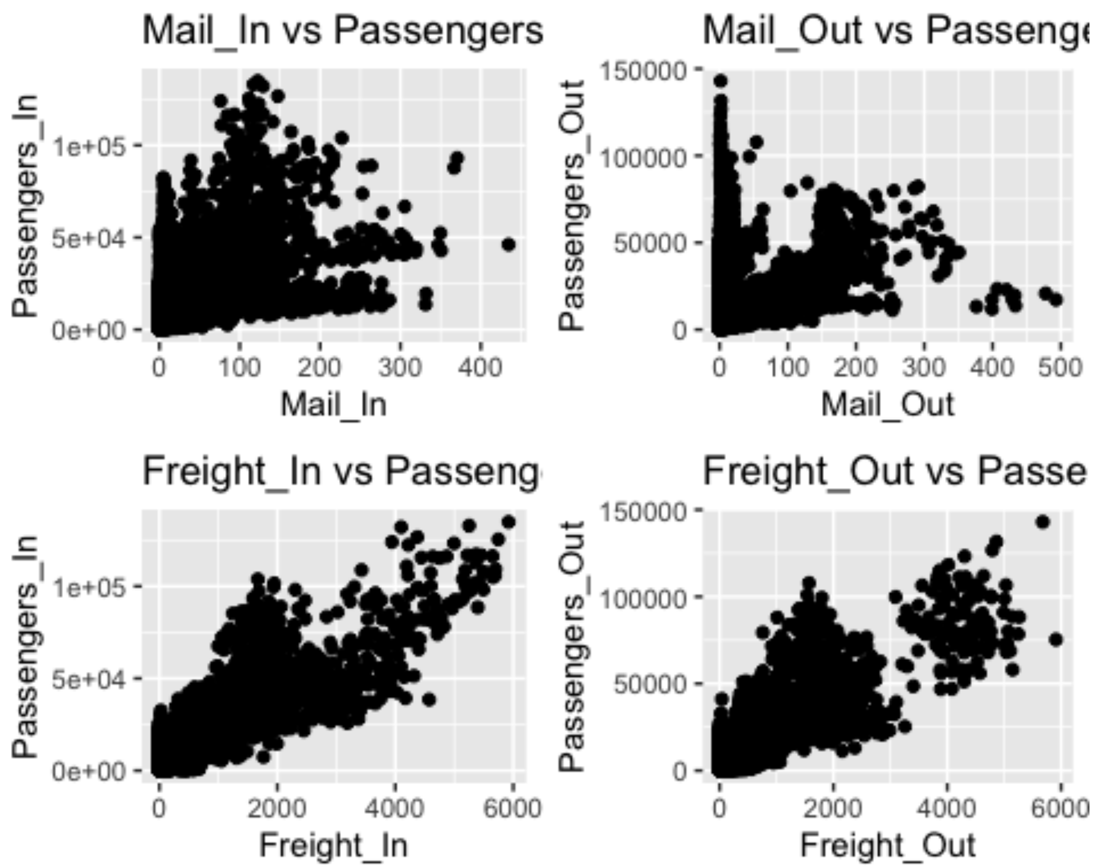
```
outlier_observation_1 <- airline_df[9611, ]
outlier_observation_2 <- airline_df[9142, ]

# Printing out the observations
print(outlier_observation_1)

## # A tibble: 1 × 19
##   Month Scheduled_Operator Country_To_From Passengers_In Freight_In Mail_In
##   <fct> <chr>              <chr>                   <int>      <dbl>   <dbl>
## 1 7     Singapore Airlines Singapore              134894      5922.    123.
## # ℹ 13 more variables: Passengers_Out <int>, Freight_Out <dbl>, Mail_Out <dbl>,
## #   Year <ord>, Total_Passengers_Carried <int>, Total_Freight_Carried <dbl>,
## #   Passengers_Out_missing <dbl>, Passengers_In_missing <dbl>,
## #   Total_Passengers_Carried_missing <dbl>, log_Passengers_In <dbl>,
## #   log_Passengers_Out <dbl>, log_Total_Passengers_Carried <dbl>,
## #   Rounded_Mail_In <dbl>

print(outlier_observation_2)

## # A tibble: 1 × 19
##   Month Scheduled_Operator Country_To_From Passengers_In Freight_In Mail_In
##   <fct> <chr>              <chr>                   <int>      <dbl>   <dbl>
## 1 2     Our Airline        Kiribati                   41          0       0
## # ℹ 13 more variables: Passengers_Out <int>, Freight_Out <dbl>, Mail_Out <dbl>,
## #   Year <ord>, Total_Passengers_Carried <int>, Total_Freight_Carried <dbl>,
## #   Passengers_Out_missing <dbl>, Passengers_In_missing <dbl>,
## #   Total_Passengers_Carried_missing <dbl>, log_Passengers_In <dbl>,
## #   log_Passengers_Out <dbl>, log_Total_Passengers_Carried <dbl>,
## #   Rounded_Mail_In <dbl>
```

After examining these observations, we can conclude that the observations are not outliers and represent a valid and realistic situation.

In addition, I will conduct a comprehensive analysis of the data for potential outliers by generating scatter plots. These plots will illustrate the relationship between 'Passengers_In' and 'Passengers_Out' and their impact on other variables. This exploration aims to understand the real-life dynamics, specifically how the presence of passengers influences the transportation of freight or mail.

```
# Scatter plot for Mail_In vs Passengers_In
plot10 <- ggplot(airline_df, aes(x = Mail_In, y = Passengers_In)) +
  geom_point() +
  labs(title = "Mail_In vs Passengers_In",
       x = "Mail_In", y = "Passengers_In")

# Scatter plot for Mail_Out vs Passengers_Out
plot11 <- ggplot(airline_df, aes(x = Mail_Out, y = Passengers_Out)) +
  geom_point() +
  labs(title = "Mail_Out vs Passengers_Out",
       x = "Mail_Out", y = "Passengers_Out")
```

```r
# Scatter plot for Freight_In vs Passengers_In
plot12 <- ggplot(airline_df, aes(x = Freight_In, y = Passengers_In)) +
  geom_point() +
  labs(title = "Freight_In vs Passengers_In",
       x = "Freight_In", y = "Passengers_In")

# Scatter plot for Freight_Out vs Passengers_Out
plot13 <- ggplot(airline_df, aes(x = Freight_Out, y = Passengers_Out)) +
  geom_point() +
  labs(title = "Freight_Out vs Passengers_Out",
       x = "Freight_Out", y = "Passengers_Out")

# Arranging plots in a 2x2 grid
grid.arrange(plot10, plot11, plot12, plot13, ncol = 2)
```



In analyzing these plots, it becomes evident that there are no outliers present in these numeric variables, and the following observations can be made:

- In relation to freight carriage, there is a positive correlation with the presence of passengers; as the number of passengers increases, so does the amount of freight carried.
- Conversely, in the case of mail, there appears to be a negative correlation; fewer passengers seem to be associated with an increase in the volume of mail transported on the plane.

Reflective journal

Initial Plan

My initial plan for this assessment was to effectively address and resolve data wrangling tasks using R programming. The task involved merging two datasets related to International Airlines Operations to and from Australia, spanning years from 1999 to 2008. The key objectives were to adhere to Tidy Data Principles, validate and clean the data, and apply necessary transformations to make it suitable for analysis.

Key Questions

- How can I ensure that the merged dataset adheres to Tidy Data Principles?
- What challenges might arise in terms of data types, variable names, and missing values?
- How can I handle missing values and outliers effectively?
- What insights can be gained from the data to inform the data preprocessing decisions?

Difficulties Encountered

- Tidying Data Principles: The initial dataset did not adhere to Tidy Data Principles. The "Month" variable contained both year and month values, and variable names required formatting. This posed a challenge in organizing the data appropriately.
- Variable Types: Converting variable types posed challenges, especially when dealing with dates and ensuring that the dataset structure was optimized for analysis.
- Missing Values: A significant number of missing values were present in numeric variables, requiring careful consideration of the impact on analysis and the choice of imputation methods.
- Outliers: Identifying and handling outliers, especially in variables like passenger counts and freight weights, required a nuanced approach to prevent bias in the dataset.

Solutions Used

- Tidy Data Principles: I addressed the issues by modifying the "Month" variable, formatting variable names, and converting variables to appropriate types, ensuring adherence to Tidy Data Principles.
- Missing Values: Employed multiple strategies, including imputation methods (mean and K-Nearest Neighbors), removing rows with extensive missing values, and creating missingness indicators.
- Outliers: Utilized a Zero-Inflated Negative Binomial model to handle variables with a large number of zeros, and carefully examined potential outliers through scatter plots and statistical modeling.

Insights Gained

- Data Patterns: Discovered that the excess zeros in mail counts were systematically influenced by the number of passengers, suggesting a zero-inflated distribution.
- Outliers: Recognized that seemingly extreme values in passenger counts were valid and aligned with real-world events, such as the introduction of new aircraft.
- Relationships: Explored the relationships between passenger counts, freight weights, and mail movements, revealing interesting dynamics in how the presence of passengers influenced cargo transportation.

Reflective Conclusion

This assessment has been a comprehensive learning experience, providing insights into the complexities of real-world datasets. Navigating challenges in data tidying, handling missing values, and addressing outliers required a combination of technical skills and critical thinking. The iterative process of exploration, analysis, and decision-making was instrumental in arriving at effective solutions. Moving forward, this experience will undoubtedly contribute to my proficiency in R programming and data wrangling tasks, enhancing my ability to extract meaningful insights from diverse datasets.

Presentation link

https://rmit-arc.instructuremedia.com/embed/c489dfdb-a6cd-4f50-a948-506ee396a312

References

BTRE (2023). International airline activity Table1 2004to2008 web.xls., https://www.bitre.gov.au/publications/ongoing/international_airline_activity-time_series.

Singapore Company Registration. (2008). Page 1 of 1. https://www.singaporeair.com/saar5/pdf/Investor-Relations/Operating-Stats/opstats-jul08.pdf.