Data Wrangling Assessment Task 2: Creating and pre-processing synthetic data

Michael Teixeira S4133975

Introduction

We created two synthetic datasets: `gyg_info` and `financial_info`. These datasets were created based on the chain of restaurants called Guzman and Gomez (GYG).The data has been adjusted to generate results that are as realistic as possible.[1]

The `gyg_info` dataset contains general information about each restaurant, including:

- RestaurantID: A unique identifier for each restaurant.
- Postcode: The postal code where the restaurant is located.
- YearFounded: The year the restaurant was established.
- Capacity: The seating capacity of the restaurant.
- Reviews: Customer ratings (from 3 to 5 stars).
- DriveThru: Whether the restaurant has a drive-through service.
- State: The Australian state where the restaurant is located.
- NumberReviews: Number of reviews, variable correlated with YearFounded.

The `financial_info` dataset contains financial data for each restaurant, including:

- RestaurantID: Same as 'gyg_info' dataset, it's the common variable.
- Revenue: The total sales for the restaurant.
- NumberOrdersMonth: The number of orders per month.
- AvgOrderPrice: The average price per order.
- Expenses: The total expenses for the restaurant.
- Profit: The difference between revenue and expenses.

Setup

Insert and load the packages you need to produce the report here:

```
# Load Packages and set seed used for this report:

library(tidyverse)    # [2]  Tidyverse collection of packages. It assists with data import, tidying, manipulation,and data visualisation. For example:"dplyr", "readr", "tidyr",ggplot2, and magrittr for pipe operators.
library(ggplot2)      # Data visualisation
```

```r
library(magrittr)     # Pipes Operators
library(here)         # [4]  Easier file path
library(deducorrect)  # [5]  To impute values
library(deductive)    # [6]  Validation and error checking
library(validate)     # [7]  Validating and checking
library(Hmisc)        # [8]  Delete outliers
library(openxlsx)     # [9]  For reading, writing, and manipulating Excel
files
library(outliers)     # [10] To work with outliers.

# Set seed
set.seed(2024)  # Seed number chosen for this report
```

Step1

Create the Synthetic Datasets

```r
# set parameters
nrestaurants <- 75

# Synthetic Dataset 1: Guzman and Gomez restaurants info
gyg_info <- data.frame(
  RestaurantID = paste0("GYG", sprintf("%03d", 1:nrestaurants)),    # paste0
to concatenate GYG with ID; sprintf to keep 3 digits in the key number.
  Postcode = sample(c(0:100, 800:900, 2000:8000), nrestaurants,
                    replace = TRUE),                                 # 0:100
to create a small chance of NA value.
  YearFounded = sample(2006:2023, nrestaurants, replace = TRUE),    # Sample
of number since year first restaurant was founded in 2006.
  Capacity = sample(30:100, nrestaurants, replace = TRUE),          # Random
capacity inbetwenn 30 to 100.
  Reviews = factor(round(runif(nrestaurants, 3, 5), 1)),            # Random
number in between 3 to 5 with one decimal number.
  DriveThru = sample(c(TRUE, FALSE), size = nrestaurants,
                     replace = TRUE, prob = c(0.3, 0.7))            # Logical
vector.Added probabilities of TRUE = 30% and FALSE = 70%.
)


# Assign states based on postcodes
gyg_info$state <- case_when(
    gyg_info$Postcode >= 2000 & gyg_info$Postcode <= 2999 ~ "NSW",
    (gyg_info$Postcode >= 2600 & gyg_info$Postcode <= 2618) |
    (gyg_info$Postcode >= 2900 & gyg_info$Postcode <= 2920) ~ "ACT",
    gyg_info$Postcode >= 3000 & gyg_info$Postcode <= 3999 ~ "VIC",
    gyg_info$Postcode >= 4000 & gyg_info$Postcode <= 4999 ~ "QLD",
    gyg_info$Postcode >= 5000 & gyg_info$Postcode <= 5999 ~ "SA",
    gyg_info$Postcode >= 6000 & gyg_info$Postcode <= 6999 ~ "WA",
    gyg_info$Postcode >= 7000 & gyg_info$Postcode <= 7999 ~ "TAS",
    gyg_info$Postcode >= 800 & gyg_info$Postcode <= 899 ~ "NT",
```

```
    TRUE ~ NA)  # Return NA for any other postcode


# Correlated variable NumberReviews based on YearFounded:
gyg_info <- gyg_info %>%
  arrange(YearFounded) %>%
  mutate(NumberReviews = round((2024 - YearFounded) * runif(nrestaurants, 50,
200), 0))  # Higher number of reviews for older years. We get the restaurant
age, to then multiply the result for a variable number in between 50 to 200.

# Structure of Dataset 1
str(gyg_info)

## 'data.frame':    75 obs. of  8 variables:
##  $ RestaurantID : chr  "GYG041" "GYG056" "GYG057" "GYG059" ...
##  $ Postcode     : int  5817 5657 4238 6610 2952 6916 6677 3263 6593 3721
...
##  $ YearFounded  : int  2006 2006 2006 2006 2006 2007 2007 2007 2007 2007
...
##  $ Capacity     : int  85 41 64 64 58 99 70 39 83 45 ...
##  $ Reviews      : Factor w/ 18 levels "3","3.1","3.3",..: 17 15 8 2 14 17
7 3 18 14 ...
##  $ DriveThru    : logi  FALSE TRUE FALSE FALSE FALSE FALSE ...
##  $ state        : chr  "SA" "SA" "QLD" "WA" ...
##  $ NumberReviews: num  1142 2409 3168 2077 1341 ...

# Add missing values
gyg_info$Reviews[sample(1:nrestaurants, 5)] <- NA # Convert 5 random index
numbers from Reviews to NA.

# Add outliers to NumberReviews:
outliers_nreviews <- sample(1:nrestaurants, 1)

# Display the first few rows of the dataset
head(gyg_info)

##   RestaurantID Postcode YearFounded Capacity Reviews DriveThru state
## 1       GYG041     5817        2006       85    <NA>     FALSE    SA
## 2       GYG056     5657        2006       41     4.6      TRUE    SA
## 3       GYG057     4238        2006       64    <NA>     FALSE   QLD
## 4       GYG059     6610        2006       64     3.1     FALSE    WA
## 5       GYG068     2952        2006       58     4.5     FALSE   NSW
## 6       GYG011     6916        2007       99     4.8     FALSE    WA
##   NumberReviews
## 1          1142
## 2          2409
## 3          3168
## 4          2077
## 5          1341
## 6          2660
```

As we can see from the `str()` function, Dataset 1 contains:

- 2 chr variables.
- 3 int variables.
- 1 factor variable.
- 1 logical variable.
- 1 num variable.

Total of 8 variables and 75 observations.

We now create the Financial Information Dataset:

```
# Generate dataset 2: Financial Information
financial_info <- data.frame(
  RestaurantID = paste0("GYG", sprintf("%03d", 1:nrestaurants)),    # Common
variable.
  Revenue = round(rnorm(nrestaurants, mean = 400000, sd = 60000)),  # Random
controlled revenue, average of 400k with standard variation of 60k.
  AvgOrderPrice = rnorm(nrestaurants, mean = 25, sd = 5))           # Random
controlled average order price of 25 with variation of 5.


# Calculate NumberOrdersMonth:
financial_info$NumberOrdersMonth <- round(financial_info$Revenue /
financial_info$AvgOrderPrice)

# Calculate Expenses (correlated variable with variation):
financial_info$Expenses <- round(financial_info$Revenue * rnorm(75, mean =
0.8, sd = 0.05)) # 80% of Revenue on average, with 5% of variation.


# Add missing values to NumberOrdersMonth:
financial_info$NumberOrdersMonth[sample(1:75, 3)] <- NA

# Add outliers to Revenue:
outliers_rev <- sample(1:75, 3)
financial_info$Revenue[outliers_rev] <- c(5253354, 2387800, 4400133)

# Add outliers to AvgOrderPrice:
outliers_avg <- sample(1:75, 1)
financial_info$AvgOrderPrice[outliers_avg] <- c(800)

# Add outliers to NumberOrdersMonth:
outliers_nord <- sample(1:75, 2)
financial_info$NumberOrdersMonth[outliers_nord] <- c(50, 899888)
```

```r
# Add outliers to Expenses:
outliers_exp <- sample(1:75, 2)
financial_info$Expenses[outliers_exp] <- c(1544030, 2357700)

# Summary statistics for financial data:
summary(financial_info)
```

```
##  RestaurantID          Revenue        AvgOrderPrice    NumberOrdersMonth
##  Length:75          Min.   : 287672   Min.   : 14.84   Min.   :    50
##  Class :character   1st Qu.: 356716   1st Qu.: 22.46   1st Qu.: 14196
##  Mode  :character   Median : 400770   Median : 24.90   Median : 16320
##                     Mean   : 541480   Mean   : 35.15   Mean   : 28604
##                     3rd Qu.: 437652   3rd Qu.: 26.51   3rd Qu.: 18714
##                     Max.   :5253354   Max.   :800.00   Max.   :899888
##                                                        NA's   :3
##     Expenses
##  Min.   : 230139
##  1st Qu.: 287335
##  Median : 316426
##  Mean   : 361025
##  3rd Qu.: 356312
##  Max.   :2357700
##
```

```r
# Structure of Dataset 2
str(financial_info)
```

```
## 'data.frame':    75 obs. of  5 variables:
##  $ RestaurantID     : chr  "GYG001" "GYG002" "GYG003" "GYG004" ...
##  $ Revenue          : num  2387800 434089 423168 448618 424426 ...
##  $ AvgOrderPrice    : num  29 24.1 27 26 17.7 ...
##  $ NumberOrdersMonth: num  16618 17980 899888 17241 23942 ...
##  $ Expenses         : num  368224 342452 343514 357038 338713 ...
```

```r
# Display the first few rows of the dataset:
head(financial_info)
```

```
##   RestaurantID Revenue AvgOrderPrice NumberOrdersMonth Expenses
## 1       GYG001 2387800      28.99197             16618   368224
## 2       GYG002  434089      24.14349             17980   342452
## 3       GYG003  423168      26.96763            899888   343514
## 4       GYG004  448618      26.01992             17241   357038
## 5       GYG005  424426      17.72748             23942   338713
## 6       GYG006  434959      24.90310             17466   359127
```

Dataset 2 contains:

- 1 chr variable.
- 4 num variables.

Total of 5 variables and 75 observations.

Based on an initial analysis of `summary(financial_info)`, it seems that the dataset includes outliers in the maximum values of all numeric variables, as well as in the minimum value of NumberOrdersMonth. Additionally, NumberOrdersMonth has 3 missing values (NA's).

Merge

We now merge both datasets using the common variable `RestaurantID`.

```
# Left join both datasets to combined_data using common variable:
combined_data <- left_join(gyg_info, financial_info, by = "RestaurantID")
```

Understand

We will ow inspect the structure of the new database combined and make modifications as needed.

```
# Structure of combined_data.

str(combined_data)
```

```
## 'data.frame':    75 obs. of  12 variables:
##  $ RestaurantID     : chr  "GYG041" "GYG056" "GYG057" "GYG059" ...
##  $ Postcode         : int  5817 5657 4238 6610 2952 6916 6677 3263 6593
3721 ...
##  $ YearFounded      : int  2006 2006 2006 2006 2006 2007 2007 2007 2007
2007 ...
##  $ Capacity         : int  85 41 64 64 58 99 70 39 83 45 ...
##  $ Reviews          : Factor w/ 18 levels "3","3.1","3.3",..: NA 15 NA 2
14 17 7 3 18 14 ...
##  $ DriveThru        : logi  FALSE TRUE FALSE FALSE FALSE FALSE ...
##  $ state            : chr  "SA" "SA" "QLD" "WA" ...
##  $ NumberReviews    : num  1142 2409 3168 2077 1341 ...
##  $ Revenue          : num  359757 319326 374235 464773 424555 ...
##  $ AvgOrderPrice    : num  21.9 26.6 25.6 30.7 26.4 ...
##  $ NumberOrdersMonth: num  16404 12024 14633 15143 16087 ...
##  $ Expenses         : num  296224 247744 310944 368000 317476 ...
```

We analyse that `state` is char data type, we want to modify to a factor type. all the other variables are in a adequate data type.

Scan I

Scan for missing values in our combined dataset:

```
# Scan the data for missing values.
colSums(is.na(combined_data)) # Total of 9 missing values on 3 different
variables.
```

```
##      RestaurantID          Postcode       YearFounded          Capacity
##                 0                 0                 0                 0
##           Reviews         DriveThru             state     NumberReviews
##                 5                 0                 1                 0
##           Revenue     AvgOrderPrice NumberOrdersMonth          Expenses
##                 0                 0                 3                 0
```

```
# Missing State
# We can approach to solve this issue by requesting the info of the
restaurant ID that it is missing the state or use the impute(fun=mode) if the
information is not relevant for our analysis.

missing_state<- which(sapply(combined_data$state, is.na))
combined_data$RestaurantID[missing_state]
```

```
## [1] "GYG023"
```

```
# combined_data$state <- impute(combined_data$state, fun = mode)
combined_data$state[missing_state] <- "WA" # We assume the correct state is
WA

# Handle missing values of reviews using Hmisc package
combined_data$Reviews <- impute(combined_data$Reviews, fun = mode)    #
Replacing values for mode, in this case NA replaced by 4.4.

# Check values that have been imputed:
is.imputed(combined_data$Reviews)                                     # TRUE
for the values that has been replaced.

# Handling missing values for NumberOrdersMonth:
# In this case we could recalculate the missing values. We will use impute()
function instead for practicing purposes.

combined_data$NumberOrdersMonth <- impute(combined_data$NumberOrdersMonth,
fun = mean)    # Mean for NA values
which(is.imputed(combined_data$NumberOrdersMonth))
# Which rows on the column has been imputed.
```

```
## [1] 29 35 72
```

```
# Scan the data for missing values.
colSums(is.na(combined_data)) # Total of 0 missing values.
```

```
##      RestaurantID          Postcode       YearFounded          Capacity
##                 0                 0                 0                 0
##           Reviews         DriveThru             state     NumberReviews
##                 0                 0                 0                 0
##           Revenue     AvgOrderPrice NumberOrdersMonth          Expenses
##                 0                 0                 0                 0
```

We can conclude that using `impute()`, can be a safe option of replacing missing values, in some situations when we are unsure that all the other values required to do calculations are correct. Also being a reduce number of missing values will unlikely affect the final results.

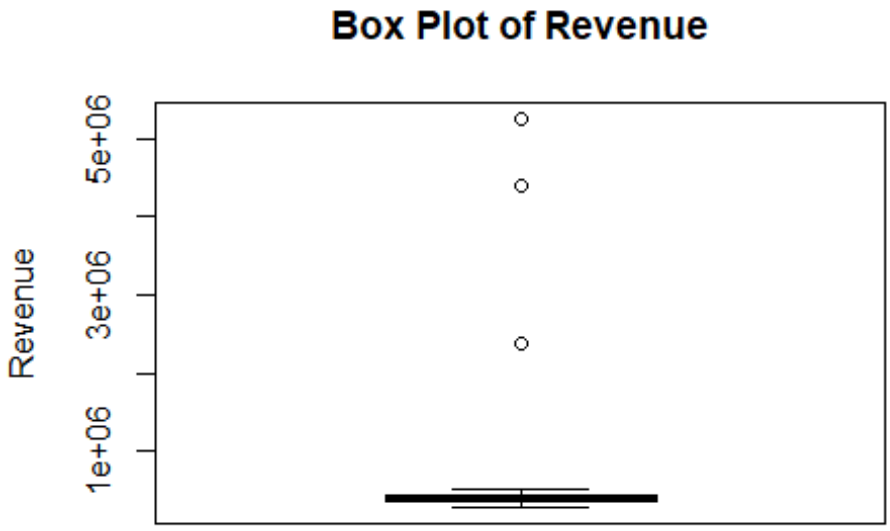
Scan II

In this stage we will be detecting outliers using **Tukey's Method**: We will then remove then because we intend to make calculations. We will give examples of how replace them by median value or by capping them to lower fence or upper fence.

```
# Function to  cap outliers:
cap <- function(x){
  quantiles <- quantile( x, c(0.05, 0.25, 0.75, 0.95 ) , na.rm = TRUE)
  x[ x < quantiles[2] - 1.5 * IQR(x, na.rm = TRUE) ] <- quantiles[1]
  x[ x > quantiles[3] + 1.5 * IQR(x, na.rm = TRUE) ] <- quantiles[4]
  x}
```

combined_data$Revenue*

```
# Use Tukey's Boxplot and IQR approach to detect Revenue outliers:
combined_data$Revenue %>%
  boxplot(main = "Box Plot of Revenue", ylab = "Revenue", col = "lightblue")
# There are at least 3 obvious outliers.
```




```
summary(combined_data$Revenue)  # Max. value is significantly higher than
median.
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  287672  356716  400770  541480  437652 5253354

# Calculate q1 and q3 to get iqr:
q1 <- quantile(combined_data$Revenue, probs = 0.25)
q3 <- quantile(combined_data$Revenue, probs = 0.75)
iqr <- q3 - q1

# Calculate lower and upper fence:
lower_fence <- q1 - (1.5 * iqr) # Lower fence is Q1 minus the inter-quartile
range.
upper_fence <- q3 + (1.5 * iqr) # Upper fence is Q3 plus the inter-quartile
range.

low_outliers <- which(combined_data$Revenue < lower_fence)
up_outliers <- which(combined_data$Revenue > upper_fence)

length(low_outliers) # There are 0 outliers.

## [1] 0

length(up_outliers)  # There are 3 outliers.

## [1] 3

low_outliers # no outliers.

## integer(0)

up_outliers  # This gives the locations (observation numbers) of the
outliers.

## [1] 12 55 75

# All outliers together
total_outliers <- unique(c(low_outliers, up_outliers))

# Remove outliers from the data
combined_data <- combined_data[-total_outliers, ]

summary(combined_data)

##
##  5 values imputed to 4.4
##
##
##  3 values imputed to 28604.28

##  RestaurantID          Postcode     YearFounded      Capacity
Reviews
##  Length:72          Min.   :  34   Min.   :2006   Min.   : 30.00   4.4
:14
```

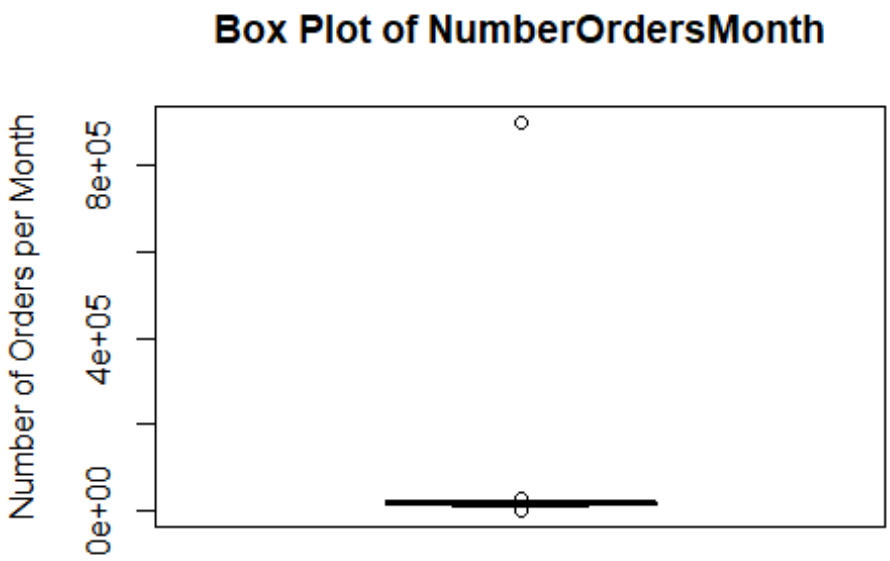
```
##  Class :character   1st Qu.:3696   1st Qu.:2009   1st Qu.: 48.00   4.5
: 6
##  Mode  :character   Median :5166   Median :2014   Median : 63.00   3.1
: 5
##                     Mean   :4913   Mean   :2014   Mean   : 65.74   3.5
: 5
##                     3rd Qu.:6161   3rd Qu.:2019   3rd Qu.: 81.50   4.8
: 5
##                     Max.   :7947   Max.   :2023   Max.   :100.00   5
: 5
##
(Other):32
##  DriveThru          state           NumberReviews     Revenue
##  Mode :logical   Length:72          Min.   : 108   Min.   :287672
##  FALSE:49        Class :character   1st Qu.: 655   1st Qu.:353494
##  TRUE :23        Mode  :character   Median :1171   Median :400172
##                                     Mean   :1291   Mean   :396802
##                                     3rd Qu.:1930   3rd Qu.:434307
##                                     Max.   :3290   Max.   :529581
##
##  AvgOrderPrice    NumberOrdersMonth    Expenses
##  Min.   : 14.84   Min.   :    50    Min.   : 230139
##  1st Qu.: 22.28   1st Qu.: 14239    1st Qu.: 285633
##  Median : 24.81   Median : 16850    Median : 315337
##  Mean   : 35.45   Mean   : 29196    Mean   : 345511
##  3rd Qu.: 26.43   3rd Qu.: 19145    3rd Qu.: 355076
##  Max.   :800.00   Max.   :899888    Max.   :2357700
##
```

In case we wanted to cap the outliers:

combined_data$Revenue < -cap(combined_data$Revenue) # Capping outliers
combined_data$Revenue[up_outliers] # Values that has been replaced to.
combined_data$Revenue[up_outliers] # This gives the values of the outliers.

Replacing `combined_data$NumberOrdersMonth` for a median value:

```
# Detect and handle outliers in NumberOrdersMonth by using cap function.
combined_data$NumberOrdersMonth %>%
  boxplot(main = "Box Plot of NumberOrdersMonth", ylab = "Number of Orders
per Month", col = "lightblue")
```




```r
summary(combined_data$NumberOrdersMonth)
```

```
##
##  3 values imputed to 28604.28
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      50   14239   16850   29196   19145  899888
```

```r
q1 <- quantile(combined_data$NumberOrdersMonth, 0.25, na.rm = TRUE)
q3 <- quantile(combined_data$NumberOrdersMonth, 0.75, na.rm = TRUE)
iqr <- q3 - q1

lower_fence <- q1 - 1.5 * iqr
upper_fence <- q3 + 1.5 * iqr

low_outliers <- which(combined_data$NumberOrdersMonth < lower_fence)
up_outliers <- which(combined_data$NumberOrdersMonth > upper_fence)

length(low_outliers) # There are 1 outliers.
```

```
## [1] 1
```

```r
length(up_outliers)  # There are 4 outliers.
```

```
## [1] 4
```

```r
low_outliers # This gives the locations (observation numbers) of the
outliers.
```

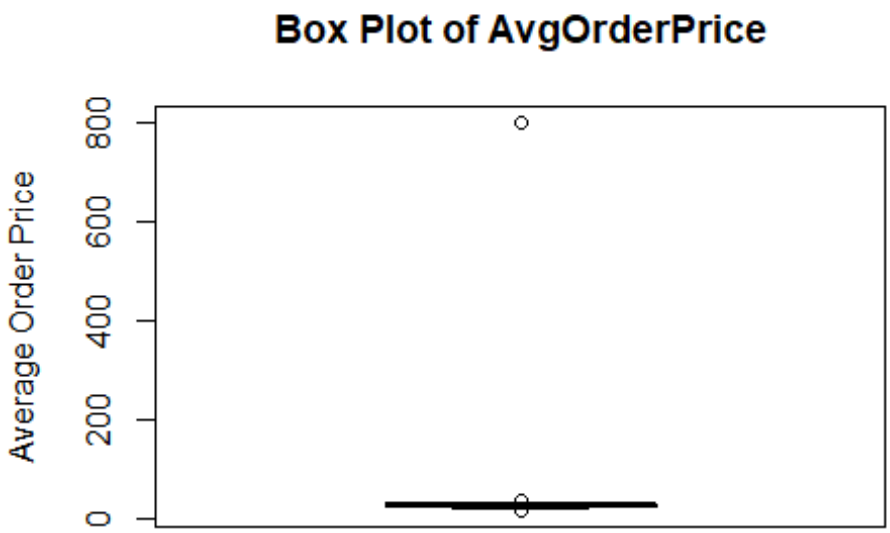
```
## [1] 68

up_outliers  # This gives the locations (observation numbers) of the
outliers.

## [1] 17 28 34 70

combined_data$NumberOrdersMonth[low_outliers]  # This gives the values of the
outliers.

## [1] 50

combined_data$NumberOrdersMonth[up_outliers]  # This gives the values of the
outliers.

## [1] 899888.00   28604.28*  28604.28*  28604.28*

# All outliers together
total_outliers <- unique(c(low_outliers, up_outliers))

# Remove outliers from the data
combined_data <- combined_data[-total_outliers, ]

summary(combined_data$NumberOrdersMonth)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10482   14228   16404   16662   18720   25016
```

Replace outliers by median values instead:

combined_data %<>% mutate(NumberOrdersMonth = case_when(NumberOrdersMonth > upper_fence ~ median(combined_data$NumberOrdersMonth), $NumberOrdersMonth <$ $lower_fence\ median(combined_data$NumberOrdersMonth), TRUE ~ NumberOrdersMonth))

combined_data$AvgOrderPrice:

```
# Detect and handle outliers in AvgOrderPrice
combined_data$AvgOrderPrice %>%
  boxplot(main = "Box Plot of AvgOrderPrice", ylab = "Average Order Price",
col = "lightblue")
```




```r
summary(combined_data$AvgOrderPrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.84   22.22   24.66   36.23   26.41  800.00
```

```r
q1 <- quantile(combined_data$AvgOrderPrice, 0.25, na.rm = TRUE)
q3 <- quantile(combined_data$AvgOrderPrice, 0.75, na.rm = TRUE)
iqr <- q3 - q1

lower_fence <- q1 - 1.5 * iqr
upper_fence <- q3 + 1.5 * iqr

low_outliers <- which(combined_data$AvgOrderPrice < lower_fence)
up_outliers <- which(combined_data$AvgOrderPrice > upper_fence)

length(low_outliers) # There are 1 outliers.
```

```
## [1] 1
```

```r
length(up_outliers)  # There are 5 outliers.
```

```
## [1] 5
```

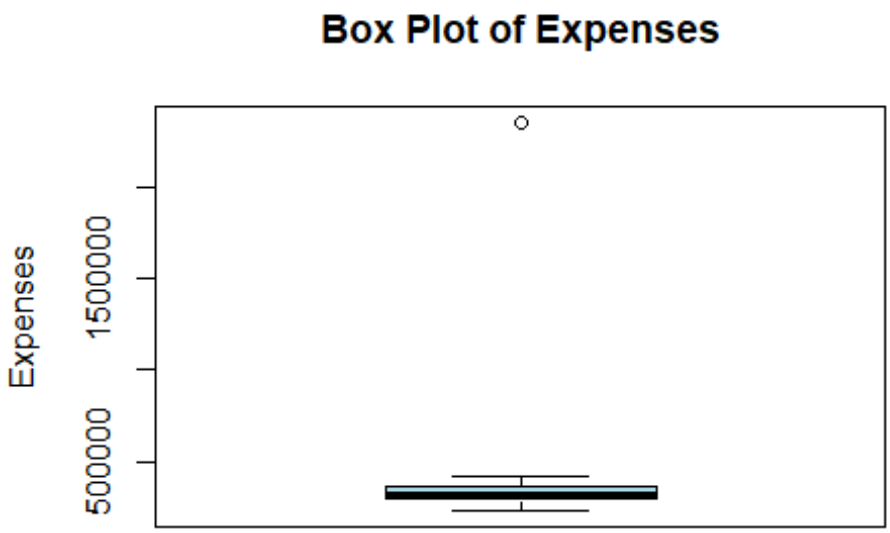
```r
low_outliers # This gives the locations (observation numbers) of the
outliers.
```

```
## [1] 33
```

```
up_outliers  # This gives the locations (observation numbers) of the
outliers.

## [1]  6 17 24 26 41

combined_data$AvgOrderPrice[low_outliers]  # This gives the values of the
outliers.

## [1] 14.84219

combined_data$AvgOrderPrice[up_outliers]  # This gives the values of the
outliers.

## [1]  36.82087  33.49605  33.67293  34.41755 800.00000

# All outliers together
total_outliers <- unique(c(low_outliers, up_outliers))

# Remove outliers from the data
combined_data <- combined_data[-total_outliers, ]

summary(combined_data$AvgOrderPrice)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   17.71   22.09   24.32   24.16   26.02   31.65
```

combined_data$Expenses:

```
# Detect and handle outliers in Expenses
combined_data$Expenses %>%
  boxplot(main = "Box Plot of Expenses", ylab = "Expenses", col =
"lightblue")
```




```r
summary(combined_data$Expenses)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  230139  294735  317476  353786  357038 2357700
```

```r
q1 <- quantile(combined_data$Expenses, 0.25, na.rm = TRUE)
q3 <- quantile(combined_data$Expenses, 0.75, na.rm = TRUE)
iqr <- q3 - q1

lower_fence <- q1 - 1.5 * iqr
upper_fence <- q3 + 1.5 * iqr

low_outliers <- which(combined_data$Expenses < lower_fence)
up_outliers <- which(combined_data$Expenses > upper_fence)

length(low_outliers) # There are 0 outliers.
```

```
## [1] 0
```

```r
length(up_outliers)  # There are 2 outliers.
```

```
## [1] 1
```

```r
low_outliers # This gives the locations (observation numbers) of the
outliers.
```

```
## integer(0)
```

```r
up_outliers  # This gives the locations (observation numbers) of the
outliers.

## [1] 14

combined_data$Expenses[low_outliers]  # This gives the values of the
outliers.

## numeric(0)

combined_data$Expenses[up_outliers]  # This gives the values of the outliers.

## [1] 2357700

# All outliers together
total_outliers <- unique(c(low_outliers, up_outliers))

# Remove outliers from the data
combined_data <- combined_data[-total_outliers, ]

# Check the summary
summary(combined_data$Expenses)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  230139  294484  316951  320387  355949  414906

str(combined_data)

## 'data.frame':    60 obs. of  12 variables:
##  $ RestaurantID     : chr  "GYG041" "GYG056" "GYG057" "GYG059" ...
##  $ Postcode         : int  5817 5657 4238 6610 2952 6677 3263 6593 3721
6906 ...
##  $ YearFounded      : int  2006 2006 2006 2006 2006 2007 2007 2007 2007
2007 ...
##  $ Capacity         : int  85 41 64 64 58 70 39 83 45 52 ...
##  $ Reviews          : Factor w/ 18 levels "3","3.1","3.3",..: 13 15 13 2
14 7 3 18 14 13 ...
##   ..- attr(*, "imputed")= int [1:3] 1 3 56
##  $ DriveThru        : logi  FALSE TRUE FALSE FALSE FALSE FALSE ...
##  $ state            : chr  "SA" "SA" "QLD" "WA" ...
##  $ NumberReviews    : num  1142 2409 3168 2077 1341 ...
##  $ Revenue          : num  359757 319326 374235 464773 424555 ...
##  $ AvgOrderPrice    : num  21.9 26.6 25.6 30.7 26.4 ...
##  $ NumberOrdersMonth: num  16404 12024 14633 15143 16087 ...
##  $ Expenses         : num  296224 247744 310944 368000 317476 ...
```

We deleted total of 15 observations which allow us now to perform the following calculations.

Manipulate Data

Now that our datasets are clean from missing values and outliers we can procede to make calculations. We will proceed to create a `combined_data$Profit`, `combined_data$ProfitPer` variables and convert the `combined_data$state` variable to factor.

```
# Calculate Profit:
combined_data$Profit <- combined_data$Revenue - combined_data$Expenses

# Calculate Profit in percentage:
combined_data$ProfitPer <- round(combined_data$Profit /
combined_data$Expenses *100,2)

# Transform State variable to factor and create levels.
combined_data$state <- factor(combined_data$state,
                              levels = c("NT", "NSW", "ACT", "VIC", "QLD",
"SA", "WA", "TAS"))

# Structure of combined_data.
str(combined_data)

## 'data.frame':    60 obs. of  14 variables:
##  $ RestaurantID     : chr  "GYG041" "GYG056" "GYG057" "GYG059" ...
##  $ Postcode         : int  5817 5657 4238 6610 2952 6677 3263 6593 3721
6906 ...
##  $ YearFounded      : int  2006 2006 2006 2006 2006 2007 2007 2007 2007
2007 ...
##  $ Capacity         : int  85 41 64 64 58 70 39 83 45 52 ...
##  $ Reviews          : Factor w/ 18 levels "3","3.1","3.3",..: 13 15 13 2
14 7 3 18 14 13 ...
##   ..- attr(*, "imputed")= int [1:3] 1 3 56
##  $ DriveThru        : logi  FALSE TRUE FALSE FALSE FALSE FALSE ...
##  $ state            : Factor w/ 8 levels "NT","NSW","ACT",..: 6 6 5 7 2 7
4 7 4 7 ...
##  $ NumberReviews    : num  1142 2409 3168 2077 1341 ...
##  $ Revenue          : num  359757 319326 374235 464773 424555 ...
##  $ AvgOrderPrice    : num  21.9 26.6 25.6 30.7 26.4 ...
##  $ NumberOrdersMonth: num  16404 12024 14633 15143 16087 ...
##  $ Expenses         : num  296224 247744 310944 368000 317476 ...
##  $ Profit           : num  63533 71582 63291 96773 107079 ...
##  $ ProfitPer        : num  21.4 28.9 20.4 26.3 33.7 ...

# Summary stats
summary(combined_data) #

##
##  3 values imputed to 4.4
```

```
##  RestaurantID          Postcode     YearFounded      Capacity
Reviews
##  Length:60          Min.   :  34   Min.   :2006   Min.   : 30.00   4.4
:10
##  Class :character   1st Qu.:3536   1st Qu.:2009   1st Qu.: 51.00   4.5
: 6
##  Mode  :character   Median :5166   Median :2014   Median : 65.00   3.1
: 5
##                     Mean   :4833   Mean   :2014   Mean   : 66.53   3.5
: 5
##                     3rd Qu.:6161   3rd Qu.:2019   3rd Qu.: 81.50   3.3
: 3
##                     Max.   :7947   Max.   :2023   Max.   :100.00   3.8
: 3
##
(Other):28
##  DriveThru           state    NumberReviews     Revenue
AvgOrderPrice
##  Mode :logical   WA     :15   Min.   : 108   Min.   :287672   Min.
:17.71
##  FALSE:41        SA     :12   1st Qu.: 595   1st Qu.:369077   1st
Qu.:22.05
##  TRUE :19        QLD    :11   Median :1155   Median :405445   Median
:24.29
##                  NSW    : 9   Mean   :1283   Mean   :400099   Mean
:24.07
##                  VIC    : 7   3rd Qu.:1930   3rd Qu.:435083   3rd
Qu.:25.88
##                  TAS    : 5   Max.   :3290   Max.   :529581   Max.
:31.65
##                  (Other): 1
##  NumberOrdersMonth    Expenses          Profit         ProfitPer
##  Min.   :10541     Min.   :230139   Min.   : 35149   Min.   : 9.01
##  1st Qu.:14685     1st Qu.:294484   1st Qu.: 63473   1st Qu.:20.81
##  Median :16850     Median :316951   Median : 77543   Median :24.80
##  Mean   :16900     Mean   :320387   Mean   : 79712   Mean   :25.33
##  3rd Qu.:18742     3rd Qu.:355949   3rd Qu.: 93138   3rd Qu.:28.79
##  Max.   :25016     Max.   :414906   Max.   :137385   Max.   :46.09
##
```

If we replaced the outliers with other values, the integrity of the results could have been compromised. `combined_data$state` is now a factor. `combined_data$Profit` and `combined_data$ProfitPer` columns has been created.

Transform

Write your plain text here.

```
# log Transformation
```

```r
combined_data$LogRevenue = log(combined_data$Revenue) # log Transformation of Revenue to LogRevenue
```

Summary statistics

```r
# Group by state and DriveThru, and calculate multiple statistics
grouped_summary <- combined_data %>%
  group_by(state, DriveThru) %>%
  summarise(
    Mean_Profit = mean(Profit, na.rm = TRUE),
    Median_Profit = median(Profit, na.rm = TRUE),
    SD_Profit = sd(Profit, na.rm = TRUE),
    Total_Revenue = sum(Revenue, na.rm = TRUE),
    Total_Expenses = sum(Expenses, na.rm = TRUE))

## `summarise()` has grouped output by 'state'. You can override using the
## `.groups` argument.

# Display the grouped summary
head(grouped_summary)

## # A tibble: 6 × 7
## # Groups:   state [4]
##   state DriveThru Mean_Profit Median_Profit SD_Profit Total_Revenue
##   <fct> <lgl>           <dbl>         <dbl>     <dbl>         <dbl>
## 1 NT    FALSE         137385        137385        NA         435453
## 2 NSW   FALSE          86365.        87086.    29366.       2475778
## 3 NSW   TRUE           82401.        83904     13368.       1199594
## 4 VIC   FALSE          75061.        78002.    24692.       2432332
## 5 VIC   TRUE           87588         87588        NA         384846
## 6 QLD   FALSE          74785.        71040     24123.       3156303
## # ℹ 1 more variable: Total_Expenses <dbl>
```

We can analyse different values such mean of profit, Total Revenue, Total Expenses. Grouped by state and driveTru.

Save File

Save in xlxs:

```r
# Create a new workbook
wb <- createWorkbook()

# Add worksheets
addWorksheet(wb, "gyg_info")
addWorksheet(wb, "financial_info")
addWorksheet(wb, "combined_data")

# Write data to worksheets
writeData(wb, sheet = "gyg_info", gyg_info)
writeData(wb, sheet = "financial_info", financial_info)
writeData(wb, sheet = "combined_data", combined_data)
```

```r
# Save the workbook to a file
saveWorkbook(wb, file = "MichaelTeixeiraS4133975.xlsx", overwrite = TRUE)
```

References

[1] Nichols N (2021) *The rise of Guzman y Gomez: The making of a global brand*, Business News Australia website, accessed 27 July 2024. https://www.businessnewsaustralia.com/articles/the-rise-of-guzman-y-gomez--the-making-of-a-global-brand.html

[2] Wickham H, Averick M, Bryan J, et al. (2019) *Welcome to the tidyverse. Journal of Open Source Software*, 4(43), 1686, https://doi.org/10.21105/joss.01686

[3] R Core Team (2019) *R: A language and environment for statistical computing*, R Foundation for Statistical Computing website, accessed 27 July 2024. https://www.R-project.org/

[4]Wickham, H. (2021). *here: A Simpler Way to Find Your Files (Version 1.0.1)*. R package. Available at: https://CRAN.R-project.org/package=here

[5]Neuhaus, J., & Cooper, M. (2020). *deducorrect: Data Deduplication and Correction (Version 1.0.0).* R package. Available at: https://CRAN.R-project.org/package=deducorrect

[6]Dupont, C., & Robert, P. (2021). *deductive: Data Validation and Deduction (Version 1.2.0).* R package. Available at: https://CRAN.R-project.org/package=deductive

[7]Berger, R., & Thiel, J. (2018). *validate: Validate Data According to a Specification (Version 1.1.0).* R package. Available at: https://CRAN.R-project.org/package=validate

[8]Harrell, F. E. (2023). *Hmisc: Harrell Miscellaneous (Version 4.7.0). R package.* Available at: https://CRAN.R-project.org/package=Hmisc

[9]Walker, S. (2021). openxlsx: Read, Write and Edit Excel xlsx Files (Version 4.2.5). R package. Available at: https://CRAN.R-project.org/package=openxlsx

[10]Iglewicz, B., & Hoaglin, D. C. (1993). How to Detect and Handle Outliers. Wiley Series in Probability and Statistics. Available at: https://CRAN.R-project.org/package=outliers