# Data Wrangling (Data Preprocessing)

Data Cleaning Using R

Vasitha Tilakumara

## Setup

```
# Load the necessary packages required to reproduce the report.
# Load necessary libraries
library(dplyr)
library(magrittr)
library(randomNames)
library(Hmisc)
#library(knitr)
library(formatR)
```

```
## Warning: package 'formatR' was built under R version 4.3.2
```

```
library(knitr)
hook_output = knit_hooks$get('output')
knit_hooks$set(output = function(x, options) {
  # this hook is used only when the linewidth option is not NULL
  if (!is.null(n <- options$linewidth)) {
    x = xfun::split_lines(x)
    # any lines wider than n should be wrapped
    if (any(nchar(x) > n)) x = strwrap(x, width = n)
    x = paste(x, collapse = '\n')
  }
  hook_output(x, options)
})
```

## Data generation

```
# Define vectors for product names and categories (These
# two vectors include real world product name and there
# categories according to bunning store website)
Product_Names <- c("Premium Interior Paint", "Taubmans Exterior Paint",
    "Paint Rollers and Brushes", "Dulux Colorbond Paint", "Spray Paints",
    "Ryobi Cordless Drill", "Stanley Screwdriver Set", "Makita Circular Saw",
    "Power Tool Accessories", "Nails and Screws", "Garden Soil and Mulch",
    "Outdoor Plants and Flowers", "GARDENA Hose Reel", "BBQ Grills and Accessories",
    "Solar Garden Lights", "Treated Pine Timber", "Plywood Sheets",
    "Roofing Iron", "Cement and Concrete Mix", "Bricks and Pavers",
```

```
    "Kitchen Cabinet Hardware", "Bathroom Fixtures", "Flooring Options (Tiles, Laminate)",
    "Lighting Fixtures", "Shelving Units", "LED Light Bulbs",
    "Electrical Switches and Outlets", "Extension Cords", "Ceiling Fans",
    "Solar Panels", "Bathroom Taps and Faucets", "PVC Pipes and Fittings",
    "Toilet Seats and Cisterns", "Sink and Shower Accessories",
    "Hot Water Systems", "Smoke Alarms", "Security Cameras",
    "Padlocks and Safes", "Fire Extinguishers", "Safety Helmets and Gear",
    "Plastic Storage Containers", "Garage Shelving", "Tool Chests and Boxes",
    "Wardrobe Organizers", "Storage Racks", "Patio Sets", "Outdoor Dining Tables and Chairs",
    "Garden Benches", "Sun Lounges", "Umbrellas and Shade Sails",
    "Behr Interior Paint", "Valspar Exterior Paint", "Paint Thinner",
    "Paint Trays and Rollers", "Rust-Oleum Spray Paint", "DeWalt Cordless Drill",
    "Craftsman Screwdriver Set", "Bosch Jigsaw", "Tool Belts and Pouches",
    "Wood Screws and Fasteners", "Potting Soil Mix", "Flowering Shrubs",
    "Hose Nozzles and Attachments", "Outdoor Fire Pits", "Solar Pathway Lights",
    "Pressure-Treated Lumber", "Drywall Sheets", "Galvanized Roofing Sheets",
    "Ready-Mix Concrete Bags", "Concrete Blocks", "Cabinet Knobs and Pulls",
    "Bathroom Vanities", "Laminate Flooring", "Pendant Light Fixtures",
    "Closet Organizers", "Incandescent Light Bulbs", "Dimmer Switches",
    "Power Strips", "Ceiling Light Fixtures", "Outdoor Solar Lights",
    "Shower Heads and Sprays", "Copper Pipes and Fittings", "Bidet Seats",
    "Bathroom Shelves and Organizers", "Tankless Water Heaters",
    "Carbon Monoxide Alarms", "Motion-Activated Security Lights",
    "Keyed Entry Door Knobs", "Fireproof Safes", "Reflective Safety Vests",
    "Clear Plastic Bins", "Garage Workbenches", "Rolling Tool Chests",
    "Shoe Racks and Organizers", "Wall-Mounted Bike Racks", "Adirondack Chairs",
    "Hammocks", "Picnic Tables", "Patio Umbrella Stands", "Outdoor Cushions and Pillows")

Product_Categories <- c("Paint and Paint Supplies", "Paint and Paint Supplies",
    "Paint and Paint Supplies", "Paint and Paint Supplies", "Paint and Paint Supplies",
    "Tools and Hardware", "Tools and Hardware", "Tools and Hardware",
    "Tools and Hardware", "Tools and Hardware", "Gardening and Outdoor",
    "Gardening and Outdoor", "Gardening and Outdoor", "Gardening and Outdoor",
    "Gardening and Outdoor", "Building Materials", "Building Materials",
    "Building Materials", "Building Materials", "Building Materials",
    "Home Improvement", "Home Improvement", "Home Improvement",
    "Home Improvement", "Home Improvement", "Electrical and Lighting",
    "Electrical and Lighting", "Electrical and Lighting", "Electrical and Lighting",
    "Electrical and Lighting", "Plumbing and Bathroom", "Plumbing and Bathroom",
    "Plumbing and Bathroom", "Plumbing and Bathroom", "Plumbing and Bathroom",
    "Safety and Security", "Safety and Security", "Safety and Security",
    "Safety and Security", "Safety and Security", "Storage and Organization",
    "Storage and Organization", "Storage and Organization", "Storage and Organization",
    "Storage and Organization", "Outdoor Furniture", "Outdoor Furniture",
    "Outdoor Furniture", "Outdoor Furniture", "Outdoor Furniture",
    "Paint and Paint Supplies", "Paint and Paint Supplies", "Paint and Paint Supplies",
    "Paint and Paint Supplies", "Paint and Paint Supplies", "Tools and Hardware",
    "Tools and Hardware", "Tools and Hardware", "Tools and Hardware",
    "Tools and Hardware", "Gardening and Outdoor", "Gardening and Outdoor",
    "Gardening and Outdoor", "Gardening and Outdoor", "Gardening and Outdoor",
    "Building Materials", "Building Materials", "Building Materials",
    "Building Materials", "Building Materials", "Home Improvement",
    "Home Improvement", "Home Improvement", "Home Improvement",
```

```
    "Home Improvement", "Electrical and Lighting", "Electrical and Lighting",
    "Electrical and Lighting", "Electrical and Lighting", "Electrical and Lighting",
    "Plumbing and Bathroom", "Plumbing and Bathroom", "Plumbing and Bathroom",
    "Plumbing and Bathroom", "Plumbing and Bathroom", "Safety and Security",
    "Safety and Security", "Safety and Security", "Safety and Security",
    "Safety and Security", "Storage and Organization", "Storage and Organization",
    "Storage and Organization", "Storage and Organization", "Storage and Organization",
    "Outdoor Furniture", "Outdoor Furniture", "Outdoor Furniture",
    "Outdoor Furniture", "Outdoor Furniture")
```

```
# Define weight ranges for each product category
weight_ranges <- list(
  "Paint and Paint Supplies" = c(0.5, 4),  # Example range for paint products in kg
  "Tools and Hardware" = c(0.5, 8),      # Example range for tools and hardware in kg
  "Gardening and Outdoor" = c(0.1, 4),  # Example range for gardening products in kg
  "Building Materials" = c(5, 20),       # Example range for building materials in kg
  "Home Improvement" = c(0.5, 4),        # Example range for home improvement products in kg
  "Electrical and Lighting" = c(0.2, 2),# Example range for electrical products in kg
  "Plumbing and Bathroom" = c(0.3, 4),  # Example range for plumbing products in kg
  "Safety and Security" = c(0.1, 4),     # Example range for safety and security products in kg
  "Storage and Organization" = c(0.2, 4),# Example range for storage products in kg
  "Outdoor Furniture" = c(0.5, 4)          # Example range for outdoor furniture in kg
)
```

```
suppliers <- c("Bunnings Wholesale", "Bunnings Direct", "Bunnings Pro Supplies",
    "Bunnings Trade Center", "Bunnings Distributors", "Bunnings Home Essentials",
    "Bunnings Hardware Solutions", "Bunnings Building Materials",
    "Bunnings Tools and More")
```

This R code simulates a scenario related to a Bunning store[1], It involves the generation of synthetic data for product inventory, employees, and transaction logs. Here's an explanation of the code:

- Defining Product Names and Categories:
  - **Product_Names** vector contains a list of product names available in a Bunning store outlet[2].
  - **Product_Categories** vector contains corresponding product categories for each product mentioned in the **Product_Names** vector.
- Defining Weight Ranges for Categories:
  - **weight_ranges** is a list that defines weight ranges for each product category. These ranges are in kilograms (kg) and represent the allowable weight of products within each category.
- Defining Suppliers:
  - **suppliers** is a vector containing different supplier names providing products to Bunning stores.
- The following packages are used in this analysis,
  - `dplyr` - following functions are used in the analysis from this package[3]

[1]Bunnings Group Limited (2023) Our range of products - Bunnings Australia,Bunning Company Website - www.bunnings.com.au, Accessed 07 September 2023. https://www.bunnings.com.au/products

[2]Bunnings Group Limited (2023) Our range of products - Bunnings Australia,Bunning Company Website - www.bunnings.com.au, Accessed 07 September 2023. https://www.bunnings.com.au/products

[3]RMIT University (2023) Module 4 Overview Summary Learning Objectives Tidy Data Principles Common problems with messy data sets The tidyr, RMIT Canva website, Accessed 12 September 2023. http://rare-phoenix-161610.appspot.com/secured/Module_04.html

  - `randomeName` - Use create randome employee names

- Mostly **Base R functions** are used in this analysis[4] plus to generate sysnthetic data different type of probability distribution functions are used here[5].

```
# Generate random customer names
EmployeeCount <- 160  # Number of names to generate
EmployeeName <- character(EmployeeCount)

# Generate random Product names
ProductCount <- 100
ProductName <- character(ProductCount)  # Number of product names to generate
UnitWeightRandom <- character(ProductCount)  # Number of unit weight depending on
# the category to generate
TotalWeight <- character(ProductCount)  # Number of Total weight to calculate

# Generate random transactions
transactionCount <- 200
```

- Generating Random Customer Names:
  - **EmployeeCount** specifies the number of employee names to generate.
  - **EmployeeName** variable is an empty character vector that will store generated employee names in the store.
- Generating Random Product Data:
  - **ProductCount** specifies the number of product names to generate.
  - **ProductName** is an empty character vector that will store generated product names in the store.
  - **UnitWeightRandom** is an empty character vector that will store randomly generated unit weights of each product in the store.
  - **TotalWeight** is an empty character vector that will store calculated total weights of each product in the system.
- Generating Random Transactions:
  - **transactionCount** specifies the number of transactions records to generate in the inventory system in the Bunning store.

```
# Set seed for reproducibility
set.seed(123)

# Generate random indices samples vector of the same length
# as product names and categories vector
sample_indices <- sample(1:length(Product_Names), ProductCount,
    replace = FALSE)

# Generate random unit weights based on category weight
# ranges specified
i = 0
for (k in Product_Categories[sample_indices]) {
    UnitWeightRandom[i] <- runif(1, min = weight_ranges[[k]][1],
```

[4]RMIT University (2023) Module 4 Overview Summary Learning Objectives Tidy Data Principles Common problems with messy data sets The tidyr, RMIT Canva website, Accessed 12 September 2023. http://rare-phoenix-161610.appspot.com/secured/Module_04.html

[5]RMIT University (2023) Generating Synthetic Data. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)

```
        max = weight_ranges[[k]][2])
    i = i + 1
}

# Convert the unitWeightRandom which is the type of
# character to double
UnitWeight <- as.double(UnitWeightRandom)

# Generate random number of products
NumOfItems <- sample(1:250, ProductCount, replace = TRUE)

# Calculate the Total weight based on the random unit
# weight and random number products
TotalWeight = NumOfItems * UnitWeight
```

- Generating Random Unit Weights and Total Weights[6]:
  - Initially a vector containing sample indices are create with the same length as product names vector.
  - Random unit weights are generated for each product item based on the specified weight ranges specified for their respective categories using a `for()` loop.
  - Random unit weight is generated drawn from uniform distribution of specified weight ranges in each category using function `runif()`. Then create **UnitWeight** vector is converted to numeric.
  - Random number of products (NumOfItems) are generated using sample method.
  - Total weights are calculated by multiplying the random unit weights by the random number of products.

```
# Set seed for reproducibility
set.seed(123)

# Create the first synthetic dataset for
# warehouse inventory
inventory_data <- data.frame(ProductID = 1:ProductCount,
    ProductName = Product_Names[sample_indices],
    Category = Product_Categories[sample_indices],
    NumOfItems, TotalWeight, ArrivalDate = sample(seq(from = as.Date("2022-01-01"),
        to = as.Date("2022-12-31"), by = "1 day"),
        ProductCount, replace = TRUE), Supplier = sample(suppliers,
        ProductCount, replace = TRUE), Status = sample(c("In Stock",
        "Out of Stock"), ProductCount, replace = TRUE),
    Priority = sample(c("High", "Medium", "Low"),
        ProductCount, replace = TRUE), ShelfLocation = sample(1:10,
        ProductCount, replace = TRUE))
head(inventory_data[, 1:4])
```

```
##   ProductID                ProductName                 Category NumOfItems
## 1         1  Bathroom Taps and Faucets    Plumbing and Bathroom        173
## 2         2     Ceiling Light Fixtures  Electrical and Lighting        142
## 3         3        Behr Interior Paint Paint and Paint Supplies        245
## 4         4 BBQ Grills and Accessories    Gardening and Outdoor         16
## 5         5             Drywall Sheets       Building Materials        215
## 6         6            Garage Shelving Storage and Organization        125
```

[6]RMIT University (2023) Generating Synthetic Data. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)

```r
head(inventory_data[, 4:8])
```

```
##   NumOfItems TotalWeight ArrivalDate                    Supplier       Status
## 1        173   131.66407  2022-06-28       Bunnings Trade Center Out of Stock
## 2        142   274.50905  2022-01-14          Bunnings Wholesale     In Stock
## 3        245    34.50133  2022-07-14    Bunnings Home Essentials     In Stock
## 4         16   124.12389  2022-11-02       Bunnings Pro Supplies Out of Stock
## 5        215   731.50985  2022-04-28 Bunnings Building Materials     In Stock
## 6        125   163.63328  2022-10-26       Bunnings Pro Supplies Out of Stock
```

```r
head(inventory_data[, 8:10])
```

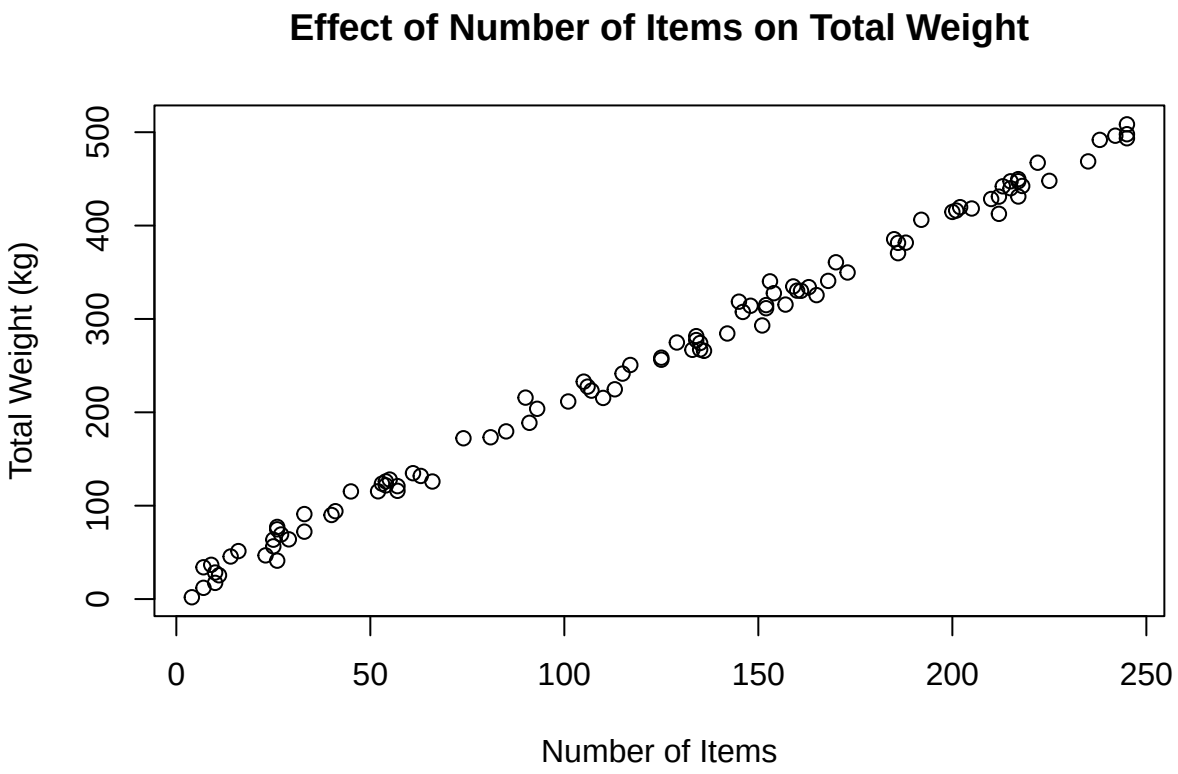
```
##         Status Priority ShelfLocation
## 1 Out of Stock     High             4
## 2     In Stock      Low            10
## 3     In Stock   Medium             9
## 4 Out of Stock     High             6
## 5     In Stock     High             3
## 6 Out of Stock      Low             4
```

- Creating Warehouse Inventory Data[7]:
  - The **inventory_data** data frame is created to store synthetic data of all the records of each product in the Bunning store outlets.
  - It includes columns such as,
    - **ProductID** = The "ProductID" variable represents a unique identifier assigned to each product in the Bunnings store. Integer type variable is created here.
    - **ProductName** = The "ProductName" holds the information about all the name of the items in the Bunning store. `sample()` functions is used to randomly create index numbers vector with same length as Product name and Product Category vector.Then using the randomly created index, The Product Names were randomly selected. Character type variable is created here.
    - **Category** = The "Category" variable contains the information about the category of the item. `sample()` functions method is used to create random respective category names for the above generated random product names. Categorical type variable is created here.
    - **NumOfItems** = Number of items for each product is picked by `sample()` functions method from a vector containing integer values from 1:250 with replacement. Integer type variable is created here.
    - **TotalWeight** = The TotalWeight is the Total weight of the each items in store. Above calculate Total weight is assigned here. Numerical type variable is created here.
    - **Supplier** = The "Supplier" variable indicates the name of the supplier company that provided the products to the Bunnings store. Random supplier names are picked from pre created vector supplier company names using `sample()` function. Character type variable is created here.
    - **ArrivalDate** = The "ArrivalDate" variable denotes the date when a specific batch or shipment of products was received and added to the store's inventory. The arrival date is picked between year 2022 (Janurary 1st to 31st of December). In order to generate random samples functions `sample()` function in combinations with `seq()` function ( create a sequence of date values in a specified range with a specified intervals of one day).

[7]RMIT University (2023) Generating Synthetic Data. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)

* **Status** = The "Status" variable describes the current availability of a product in the store's inventory. Random product availability status are picked from pre created vector and then using `sample()` function. The values in the pre define vector are "In Stock" and "Out of Stock" which describe the availability of the product in the inventory system.
* **Priority** = Represents the level of importance in restocking the relevant item in the inventory.
* **ShelfLocation** = The "ShelfLocation" variable represents the aisle number within the Bunnings store where a particular item is stored or displayed on the shelves.

```r
# Set seed for reproducibility
set.seed(250)

# Set the correlation strength
correlation_strength <- 2

# Create random error values bases
inventory_data %<>%
    mutate(error = rnorm(n = ProductCount, mean = 0, sd = 10))

# Generate correlated TotalWeight based on numofItems
inventory_data %<>%
    mutate(TotalWeight = inventory_data$NumOfItems * correlation_strength +
        10 + inventory_data$error)

# Plotting of correlated synthetic data between number of
# item and Weight
plot(x = inventory_data$NumOfItems, y = inventory_data$TotalWeight,
    main = "Effect of Number of Items on Total Weight", xlab = "Number of Items",
    ylab = "Total Weight (kg)")
```




- Generating Correlated Total Weight[8]:
  - Normal distribution is used in generating correlated random data between **NumOfItems** and **UnitWeight** variables.
  - Positive correlation is introduced between the **TotalWeight** and **NumOfItems** variables. Based on the following equation, The study found that a Total weight of the items was expected to increase based on Number of items of each product in the store, **TotalWeight** of the each item will be later be useful in handling the product inside the store and for shipping purpose, according to the following relationship:

$$TotalWeight = 2 * NumOfItems + 10$$

  - Random **error** is generated using normal distribution function `rnorm()` mean of 0 and a standard deviation of 10 kg. and added to the **TotalWeight** variable to create messiness in correlation between **NumOfItems** and **UnitWeight** variables.The equation is modified to include a random and messiness **TotalWeight** variable,

$$TotalWeight = 2 * NumOfItems + 10 + error$$

- Plotting Data: Using `plot()` a scatter plot is created to visualize the relationship between the number of items and total weight of the products in the Bunnings store.

```
# Set seed for reproducibility
set.seed(98765)
```

[8]RMIT University (2023) Generating Synthetic Data. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)

```r
# Introduce missing values (approximately 5% values in the
# Weight column)
inventory_data %<>%
    mutate(rand = runif(ProductCount, min = 0, max = 1))
inventory_data %<>%
    mutate(TotalWeight = case_when(inventory_data$rand >= 0.09 ~
        inventory_data$TotalWeight))
```

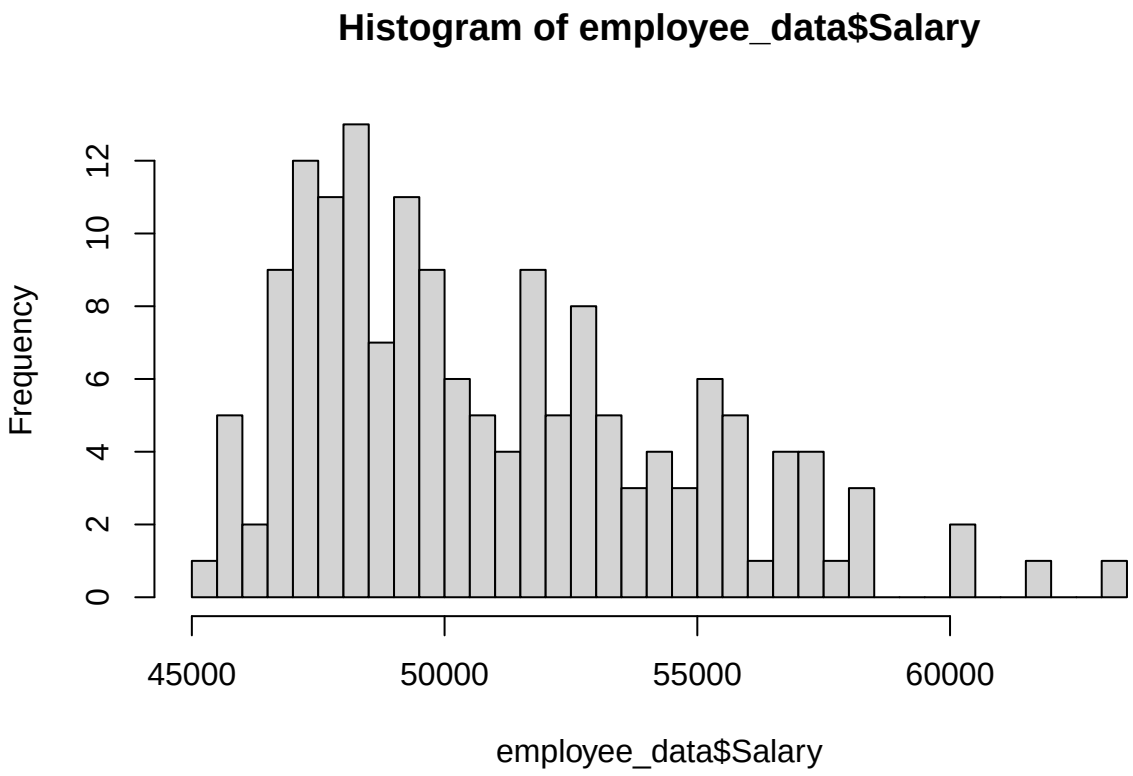
- Introducing Missing Values[9]:
  - Approximately 5% of values in the **TotalWeight** variable are randomly replaced with missing values. Check the next section to view a clear explanation.

```r
# Set seed for reproducibility
set.seed(250)

# Create a second synthetic dataset for
# Bunning store employees
employee_data <- data.frame(EmployeeID = 1:EmployeeCount,
    EmployeeName = paste(randomNames(EmployeeCount,
        which.names = "first"), randomNames(EmployeeCount,
        which.names = "last")), Department = sample(c("Shipping",
        "Receiving", "Inventory", "Management"),
        EmployeeCount, replace = TRUE), Age = sample(25:60,
        EmployeeCount, replace = TRUE), Gender = sample(c("Male",
        "Female"), EmployeeCount, replace = TRUE),
    Salary = rbeta(EmployeeCount, shape1 = 2,
        shape2 = 15) * 51035 + 45000, EmploymentType = sample(c("Full-Time",
        "Part-Time"), EmployeeCount, replace = TRUE))

head(employee_data)
```

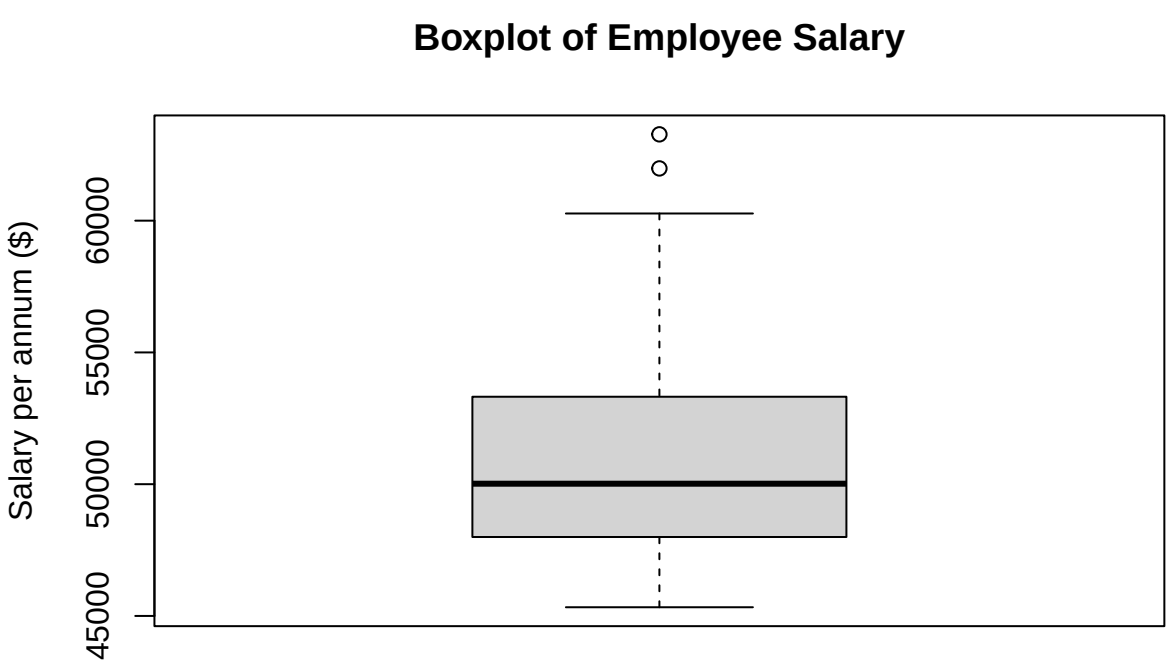
```
##   EmployeeID      EmployeeName Department Age Gender   Salary EmploymentType
## 1          1  Briana Hernandez  Receiving  25   Male 52570.31      Part-Time
## 2          2    Mareno Alarcon  Inventory  43 Female 45840.07      Full-Time
## 3          3     Kelly Johnson   Shipping  37   Male 48873.34      Part-Time
## 4          4 Krystal Sheffield Management  50   Male 57137.24      Full-Time
## 5          5           Yee Lee   Shipping  52   Male 49881.69      Part-Time
## 6          6    Yamiles Pineda  Inventory  25   Male 47598.28      Full-Time
```

```r
# Plot a hist for employee salary to check
# the normal distribution
hist(employee_data$Salary, breaks = 50)
```

[9]RMIT University (2023) Generating Synthetic Data. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)




```r
# Z score will not be used to check of
# outliers since employee salary is not
# approximately normally distributed
boxplot(employee_data$Salary, main = "Boxplot of Employee Salary",
    ylab = "Salary per annum ($)")
```




- Creating Employee Data[10]:
  - The **employee_data** data frame is synthetically generated to simulate employee information at a Bunnings store.
  - It includes variables such as,
    - **EmployeeID** = The "EmployeeID" variable represents a unique identifier assigned to each employee working in the Bunnings store.
    - **EmployeeName** = The "EmployeeName" variable represents the full name of each employee working at the Bunnings store. Employee Full name including the first name and the last name created using function `randomNames()` from package `randomNames`, Later joined them together using `paste()` function.
    - **Department** = The "Department" variable specifies the department within the Bunnings store where the employee is assigned to work. Assigned department of the each employee, Vector of department names were created, then using `sample()` function all the data were sample from the vector created.
    - **Age** = The "Age" variable records the age of each employee. Random age values are generated using a vector of age between 25 and 60, then sample the age of each Employee from the created vector with replacement.
    - **Gender** = The "Gender" variable captures the gender of each employee. To generate the random gender of the each employee, same sample method is used with vector which include the genders.
    - **Salary** = The "Salary" variable represents the monetary compensation, typically on an annual or monthly basis, that an employee receives for their work. To obtain a salary variable values

[10]RMIT University (2023) Generating Synthetic Data. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)

which are not uniform or normal,beta distribution is used. This generated the salary of all the team members in the store ranging from $45,000 to $60,000 per annum.

  * **EmploymentType** = The "EmploymentType" variable specifies the type of employment or contractual arrangement that an employee has with the Bunnings store. Record the employment type of each employee is Full-Time or Part-Time. Once again the same sample method is used to generate the employement type of each employee from a vector of employement type values such as "Full-Time" or "Part-Time".

- Normal distribution of employee salary is plotted here to check if it is approximately normally distributed to determine which outlier method to use to scan outliers in the employee salary variable.

- To introduce a random and messiness into the data frame, Outliers are generated in **Salary** variable using beta distribution. The `hist()` plots the normal distribution and then using `boxplot()` the outliers are identified.

```
# Set seed for reproducibility
set.seed(965)

# Introduce missing values (approximately 5% values in the
# Salary column)
employee_data %<>%
    mutate(rand = runif(EmployeeCount, min = 0, max = 1))
employee_data %<>%
    mutate(Salary = case_when(employee_data$rand >= 0.05 ~ employee_data$Salary))
```

- Introducing Missing Values[11]:
  - Approximately 5% missing values are introduced in the **Salary** variable of the employee data frame.To obtain this outcome `runif()` is used to draw the random numbers between 0 and 1, `case_when()` is used to conditionally update values in the **Salary** variable based on the **rand** variable. `employee_data$rand >= 0.05` checks if the value in the **rand** variable is greater than or equal to 0.05 (which is approximately 5%).
  - As a result, this line of code introduces missing values into the **Salary** variable for rows where the **rand** value is less than 0.05. rows where **rand** is greater than or equal to 0.05 will retain their original Salary values.
  - This method used to generate missing values in other variables in each data sets.

```
# Set seed for reproducibility
set.seed(757)

# Create a synthetic transaction log
transaction_log <- data.frame(ProductID = sample(1:150, transactionCount,
    replace = TRUE), EmployeeID = sample(1:100, transactionCount,
    replace = TRUE), ActivityType = factor(sample(c("Receiving",
    "Shipping", "Stocking"), transactionCount, replace = TRUE)),
    ActivityDate = sample(seq(from = as.Date("2022-01-01"), to = as.Date("2022-12-31"),
        by = "1 day"), transactionCount, replace = TRUE))

head(transaction_log)
```

```
##   ProductID EmployeeID ActivityType ActivityDate
```

[11]RMIT University (2023) Generating Synthetic Data. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)

```
## 1        120          72     Shipping   2022-02-10
## 2         64          59    Receiving   2022-06-30
## 3         84          78     Stocking   2022-11-24
## 4         66          39     Stocking   2022-09-29
## 5        126          40    Receiving   2022-02-05
## 6         77          25     Shipping   2022-07-26
```

- Creating Transaction Log Data[12]:
  - The **transaction_log** data frame is created to store synthetic data for transaction logs. Transaction Log in a Bunnings store's inventory system is a systematic record that captures essential details about product transactions, employee involvement, activity types, and the timing of these events.
  - It includes columns such as,
    - **ProductID** = The "ProductID" variable is a unique identifier assigned to each product or item in the Bunnings store's inventory.
    - **EmployeeID** = The "EmployeeID" variable represents a unique identifier assigned to each employee or staff member working in the Bunnings store.
    - **ActivityType** = The "ActivityType" variable indicates the type of activity being logged in the transaction record. Activity types are, Receiving = Receiving of the product by the supplier at the store, Shipping = Shipping of the product from the store to the customer and Stocking = Stocking of the product in the store once product are supplied by the supplier.
    - **ActivityDate** = The "ActivityDate" variable specifies the date when the above mentioned activity type occurred.

```
# Introduce missing values (approximately 5% values in the
# Activity Date)
transaction_log %<>%
    mutate(rand = runif(transactionCount, min = 0, max = 1))
transaction_log %<>%
    mutate(ActivityType = case_when(transaction_log$rand >= 0.05 ~
        transaction_log$ActivityType))
```

- Introducing Missing Values:
  - Approximately 5% of missing values are introduced in the **ActivityType** variable of the transaction log using same method mentioned in the missing value generation in the **Salary** variable.

```
# Checking all the missing values
inventory_data$TotalWeight %>% is.na()  %>% sum()
```

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

```
employee_data$Salary %>% is.na()  %>% sum()
```

```
## [1] 8
```

```
transaction_log$ActivityType %>% is.na()  %>% sum()
```

```
## [1] 10
```

[12]RMIT University (2023) Generating Synthetic Data. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)

- Checking the Summary of Missing Values[13]:
  - The code checks and counts the missing values in the **TotalWeight** variables of the **inventory** data frame, the **Salary** variable of the **employee** data frame, and the **ActivityType** variable of the **transaction** log data frame.

In summary, this code generates synthetic data to simulate a Bunning store outlet scenario, including inventory information of products, employee information, and transaction records of the products. It introduces missing values to mimic real-world data scenarios and creates correlations between certain variables. The code provides an example of data generation and manipulation for analytical purposes.

## Merging data sets

```
# Merge the two datasets based on the transaction ProductID
# and EmployeeID
merged_data <- transaction_log %>%
    select(-c("rand")) %>%
    inner_join(inventory_data, by = "ProductID") %>%
    select(-c("rand")) %>%
    inner_join(employee_data[, c("EmployeeID", "EmployeeName",
        "Department", "Salary")], by = "EmployeeID")

head(merged_data[, 1:5])
```

```
##   ProductID EmployeeID ActivityType ActivityDate              ProductName
## 1        64         59    Receiving   2022-06-30            Plywood Sheets
## 2        84         78     Stocking   2022-11-24             Storage Racks
## 3        66         39     Stocking   2022-09-29     Dulux Colorbond Paint
## 4        77         25     Shipping   2022-07-26    Carbon Monoxide Alarms
## 5        27         34     Stocking   2022-11-06 Copper Pipes and Fittings
## 6        51         64    Receiving   2022-11-10         Laminate Flooring
```

```
head(merged_data[, 5:8])
```

```
##                ProductName                 Category NumOfItems TotalWeight
## 1            Plywood Sheets       Building Materials         26    77.25363
## 2             Storage Racks Storage and Organization        217   431.12637
## 3     Dulux Colorbond Paint Paint and Paint Supplies        213   442.06214
## 4    Carbon Monoxide Alarms      Safety and Security         93   203.77509
## 5 Copper Pipes and Fittings    Plumbing and Bathroom        235   468.71457
## 6         Laminate Flooring         Home Improvement        101   211.58621
```

```
head(merged_data[, 9:12])
```

```
##   ArrivalDate                   Supplier       Status Priority
## 1  2022-12-08     Bunnings Tools and More Out of Stock   Medium
## 2  2022-06-07     Bunnings Tools and More     In Stock   Medium
```

[13]RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values

```
## 3  2022-06-08 Bunnings Hardware Solutions     In Stock     High
## 4  2022-03-30 Bunnings Hardware Solutions     In Stock      Low
## 5  2022-02-12       Bunnings Pro Supplies     In Stock     High
## 6  2022-06-02    Bunnings Home Essentials     In Stock     High
```

```
head(merged_data[, 12:17])
```

```
##   Priority ShelfLocation       error          EmployeeName Department   Salary
## 1   Medium             6  15.2536257         Trevor Goosby  Inventory 53930.43
## 2   Medium             2 -12.8736253          David Macias Management 52955.56
## 3     High             5   6.0621442         Nu'ma Judkins  Inventory 47406.55
## 4      Low             6   7.7750870     Ricardo Humphries   Shipping 52076.08
## 5     High             7 -11.2854285 Addison Albani-Burgio Management 51558.67
## 6     High             1  -0.4137897        Tyler al-Murad  Receiving 60232.14
```

The following merged data set contain values from 3 different data frames each with number of observation greater than 100 and number of variables greater than 5. The purpose of this merge is to obtain all the employee IDs, name, department and salary who handle the products recorded in the transaction log. Also, it includes all the details of the products handled by the employees and their ID is recorded in the transaction log.

Initially, **transaction_log** data frame is merged with **inventory_data** data frame based on their common variable **ProductID** using `inner_join()` function and ignore the **rand** variable in the both data frames using `select()` function[14]. Using the result of the **transaction_log** and the **inventory_data** data frames, then **employee_data** data frame is merged based on the common variable **EmployeeID** but only including "EmployeeID","EmployeeName","Department" and"Salary" variables from the **employee_data** data frame.

`inner_join()` function[15] takes the all the observation with same common variable in the data frames and ignore the rest.

## Checking structure of combined data

```
# Check the dimensions of the merged data frame
dim(merged_data)
```

```
## [1] 134  17
```

```
# Check structure of combined data
str(merged_data)
```

```
## 'data.frame': 134 obs. of 17 variables:
## $ ProductID : int 64 84 66 77 27 51 44 97 3 36 ...
## $ EmployeeID : int 59 78 39 25 34 64 36 52 20 91 ...
## $ ActivityType : Factor w/ 3 levels
```

[14]RMIT University (2023) Module 4 Overview Summary Learning Objectives Tidy Data Principles Common problems with messy data sets The tidyr, RMIT Canva website, Accessed 12 September 2023. http://rare-phoenix-161610.appspot.com/secured/Module_04.html

[15]RMIT University (2023) Module 4 Overview Summary Learning Objectives Tidy Data Principles Common problems with messy data sets The tidyr, RMIT Canva website, Accessed 12 September 2023. http://rare-phoenix-161610.appspot.com/secured/Module_04.html

```
"Receiving","Shipping",..: 1 3 3 2 3 1 3 3 1 2 ...
## $ ActivityDate : Date, format: "2022-06-30" "2022-11-24"
...
## $ ProductName : chr "Plywood Sheets" "Storage Racks"
"Dulux Colorbond Paint" "Carbon Monoxide Alarms" ...
## $ Category : chr "Building Materials" "Storage and
Organization" "Paint and Paint Supplies" "Safety and
Security" ...
## $ NumOfItems : int 26 217 213 93 235 101 90 85 245 107
...
## $ TotalWeight : num 77.3 431.1 442.1 203.8 468.7 ...
## $ ArrivalDate : Date, format: "2022-12-08" "2022-06-07"
...
## $ Supplier : chr "Bunnings Tools and More" "Bunnings
Tools and More" "Bunnings Hardware Solutions" "Bunnings
Hardware Solutions" ...
## $ Status : chr "Out of Stock" "In Stock" "In Stock" "In
Stock" ...
## $ Priority : chr "Medium" "Medium" "High" "Low" ...
## $ ShelfLocation: int 6 2 5 6 7 1 5 5 9 8 ...
## $ error : num 15.25 -12.87 6.06 7.78 -11.29 ...
## $ EmployeeName : chr "Trevor Goosby" "David Macias"
"Nu'ma Judkins" "Ricardo Humphries" ...
## $ Department : chr "Inventory" "Management" "Inventory"
"Shipping" ...
## $ Salary : num 53930 52956 47407 52076 51559 ...
```

Dimension of the merged data set is inspected using `dim()` function. The output of this function produced,

- No. of observations = 134
- No. of variables = 15.

The structure of the merged data frame display variables of data type,

- Integer variables
- Date variables
- Character variables
- Numeric variables
- Factor variables

```r
# Convert all the character type variables into factors
merged_data <- data.frame(lapply(merged_data[, !names(merged_data) %in%
    c("ProductName", "Supplier", "EmployeeName")], function(x) if (is.character(x)) {
    as.factor(x)
} else {
    x
}), ProductName = merged_data$ProductName, Supplier = merged_data$Supplier,
    EmployeeName = merged_data$EmployeeName)

# Convert the variable Activity Type to factors and order
# the levels
merged_data$ActivityType <- factor(merged_data$ActivityType,
    levels = c("Receiving", "Stocking", "Shipping"))
```

```r
# Check the level of the factors in the data frame
lapply(merged_data, function(x) if (is.factor(x)) {
    levels(x)
})
```

```
## $ProductID
## NULL
##
## $EmployeeID
## NULL
##
## $ActivityType
## [1] "Receiving" "Stocking" "Shipping"
##
## $ActivityDate
## NULL
##
## $Category
## [1] "Building Materials" "Electrical and Lighting"
## [3] "Gardening and Outdoor" "Home Improvement"
## [5] "Outdoor Furniture" "Paint and Paint Supplies"
## [7] "Plumbing and Bathroom" "Safety and Security"
## [9] "Storage and Organization" "Tools and Hardware"
##
## $NumOfItems
## NULL
##
## $TotalWeight
## NULL
##
## $ArrivalDate
## NULL
##
## $Status
## [1] "In Stock" "Out of Stock"
##
## $Priority
## [1] "High" "Low" "Medium"
##
## $ShelfLocation
## NULL
##
## $error
## NULL
##
## $Department
## [1] "Inventory" "Management" "Receiving" "Shipping"
##
## $Salary
## NULL
##
## $ProductName
## NULL
```

```
##
## $Supplier
## NULL
##
## $EmployeeName
## NULL
```

```
# Check structure of after data conversion
str(merged_data)
```

```
## 'data.frame': 134 obs. of 17 variables:
## $ ProductID : int 64 84 66 77 27 51 44 97 3 36 ...
## $ EmployeeID : int 59 78 39 25 34 64 36 52 20 91 ...
## $ ActivityType : Factor w/ 3 levels
"Receiving","Stocking",..: 1 2 2 3 2 1 2 2 1 3 ...
## $ ActivityDate : Date, format: "2022-06-30" "2022-11-24"
...
## $ Category : Factor w/ 10 levels "Building
Materials",..: 1 9 6 8 7 4 1 4 6 8 ...
## $ NumOfItems : int 26 217 213 93 235 101 90 85 245 107
...
## $ TotalWeight : num 77.3 431.1 442.1 203.8 468.7 ...
## $ ArrivalDate : Date, format: "2022-12-08" "2022-06-07"
...
## $ Status : Factor w/ 2 levels "In Stock","Out of Stock":
2 1 1 1 1 1 1 1 1 2 ...
## $ Priority : Factor w/ 3 levels "High","Low","Medium": 3
3 1 2 1 1 2 2 3 2 ...
## $ ShelfLocation: int 6 2 5 6 7 1 5 5 9 8 ...
## $ error : num 15.25 -12.87 6.06 7.78 -11.29 ...
## $ Department : Factor w/ 4 levels
"Inventory","Management",..: 1 2 1 4 2 3 3 3 3 1 ...
## $ Salary : num 53930 52956 47407 52076 51559 ...
## $ ProductName : chr "Plywood Sheets" "Storage Racks"
"Dulux Colorbond Paint" "Carbon Monoxide Alarms" ...
## $ Supplier : chr "Bunnings Tools and More" "Bunnings
Tools and More" "Bunnings Hardware Solutions" "Bunnings
Hardware Solutions" ...
## $ EmployeeName : chr "Trevor Goosby" "David Macias"
"Nu'ma Judkins" "Ricardo Humphries" ...
```

After identifying all the variables require data type conversion, `lapply()` function[16] was used to convert all the necessary categorical type variables into factors, user defined function is invoked coupled with `is.character` and `as.factor`. Both of these functions check if the variables is character and once it is satisfied the character variables is converted using respective functions. Above chunk, it is visible the removal of character type variables which are not categorical variables from the data frame before converting to factor. Therefor, those variable won't be effected. Later, after the required character variables are converted the removed character type variables(e.g:"ProductName","Supplier" & "EmployeeName" ) are added back to the data frame. Next, after observing the resulted data frame, **ActivityType** variable need to be ordered

[16]RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values

according to the order of the process which the product are handle once they are received by the store up until it is shipped to the customer[17].

Once all the data type conversions are completed, invoke `lapply()` function combination with a user define function which checks the if the variable is of type factor, then produce the levels of those factor variables[18]. All the levels were remained same since it is not required to order and level names is clear for the analysis.

## Generate summary statistics

```
# Generate summary statistics and group them
# by product category
stat <- merged_data %>%
    group_by(Category) %>%
    summarise(min = min(TotalWeight, na.rm = TRUE),
        first_quantile = quantile(TotalWeight,
            0.25, na.rm = TRUE), Median = median(TotalWeight,
            na.rm = TRUE), Mean = mean(TotalWeight,
            na.rm = TRUE), third_quantile = quantile(TotalWeight,
            0.75, na.rm = TRUE), max = max(TotalWeight,
            na.rm = TRUE), stand_dev = sd(TotalWeight,
            na.rm = TRUE), n = n(), missing = sum(is.na(TotalWeight)),
        IQR = third_quantile - first_quantile)
head(stat[, 1:4])
```

```
## # A tibble: 6 x 4
##   Category                    min first_quantile Median
##   <fct>                     <dbl>          <dbl>  <dbl>
## 1 Building Materials         77.3           125.    216.
## 2 Electrical and Lighting   115.            189.    284.
## 3 Gardening and Outdoor      69.2           126.    315.
## 4 Home Improvement          124.            180.    218.
## 5 Outdoor Furniture           1.95           72.2   326.
## 6 Paint and Paint Supplies   17.3           308.    387.
```

```
head(stat[, c(1, 5:7)])
```

```
## # A tibble: 6 x 4
##   Category                   Mean third_quantile   max
##   <fct>                    <dbl>          <dbl> <dbl>
## 1 Building Materials         221.           277.  415.
## 2 Electrical and Lighting    281.           368.  448.
## 3 Gardening and Outdoor      265.           371.  467.
## 4 Home Improvement           265.           368.  498.
## 5 Outdoor Furniture          233.           328.  442.
## 6 Paint and Paint Supplies   356.           508.  508.
```

[17]RMIT University (2023) Module 3 overview summary learning objectives to check attributes of R objects Learn how to convert between data types/structures. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_03.html#Summary

[18]RMIT University (2023) Module 3 overview summary learning objectives to check attributes of R objects Learn how to convert between data types/structures. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_03.html#Summary

```r
head(stat[, c(1, 8:11)])
```

```
## # A tibble: 6 x 5
##   Category                 stand_dev     n missing   IQR
##   <fct>                        <dbl> <int>   <int> <dbl>
## 1 Building Materials            110.    10       0  153.
## 2 Electrical and Lighting       113.    15       0  179.
## 3 Gardening and Outdoor         139.    10       0  245.
## 4 Home Improvement              132.    15       1  189.
## 5 Outdoor Furniture             157.    13       0  255.
## 6 Paint and Paint Supplies      162.    16       0  201.
```

In this section, summary statistics are shown and grouped according to the product category.

- "Building Materials" has a relatively small weight range with a minimum of 77.253626 and a maximum of 414.6209 , while "Tools and Hardware" has a wider range with a minimum of 29.75 and a maximum of 892.5.
- The "stand_dev" (standard deviation) reflects the dispersion in Total weight of products within each category. Categories with higher standard deviations, such as "Plumbing and Bathroom," have more Total weight variability compared to those with lower standard deviations, like "Safety and Security"
- The "Mean" and "Median" values indicate the average total weigh of products in each category. "Electrical and Lighting" has a mean price of 281.38, which is very close to its median of 284.42, suggesting a relatively symmetric distribution. In contrast, "Paint and Paint Supplies" has a mean of 355.72 and a median of 386.63, indicating a potential right-skewed distribution.
- The first and third quantile ("first_quantile" and "third_quantile") provide information about the spread of total weight of the products. A larger interquartile range (IQR = third quantile minus first quantile) suggests a wider spread of total weights within the category. For instance, "Plumbing and Bathroom" has a substantial spread with a first quantile of 74.72005 and a third quantile of 406.1935.
- The "n" (sample size) indicates the number of observations for each category. Categories with larger sample sizes tend to have more reliable summary statistics.
- It's important to note the "missing" variable, which shows the number of missing values in each category. This can affect the accuracy of the summary statistics, especially when comparing categories. For example, "Tools and Hardware" has missing data for 5 observations, which may impact the reliability of its statistics.

## Scanning data

```r
# Check all the missing values in each column and get there index
missingValues <- merged_data %>%  sapply(function(x) which(is.na(x)))

# check the number of missing value in each variable
merged_data %>%  sapply(function(x) length(which(is.na(x))))
```

```
##     ProductID    EmployeeID  ActivityType  ActivityDate      Category
##             0             0             8             0             0
##    NumOfItems   TotalWeight   ArrivalDate        Status      Priority
##             0             6             0             0             0
```

```
## ShelfLocation         error    Department        Salary   ProductName
##             0             0             0             8             0
##      Supplier  EmployeeName
##             0             0
```

To scan for missing values following steps are taken[19],

1. `sapply()`: sapply is a function in R used to apply a specified function to each column in the merged data frame.
2. `function(x) which(is.na(x))`[20]: This is an user define function that is applied to each column in *merged_data* data frame. The purpose of this function is to find the positions where missing values (NA) occur in the column.
3. `is.na(x)`: This part of the function checks if each element of the column is NA and returns a logical vector of the same length, with TRUE indicating NA values and FALSE indicating non-NA values.
4. `which(is.na(x))`: This part of the function takes the logical vector from the previous step and returns the indices (positions) where TRUE values occur, indicating the positions of missing values in each column.
5. `length()`: calculate length of the resulted vector of values containing positions of missing values in each column.

So, when you apply sapply to the data frame merged_data, it effectively applies the user define function to each column of merged_data. The result is a list where each element corresponds to a column in merged_data, and each element contains the indices of missing values in that column.

```
# mode imputation (for categorical/factor variables)
merged_data$ActivityType <- impute(merged_data$ActivityType, fun = mode)
```

Next, Explains how these missing values are handled,

- Mode Imputation for Categorical/Factor Variable **ActivityType**[21]:
  - Missing values in the **ActivityType** variable are imputed with the mode (most frequent value) using the `impute()` function from an `Hmisc` package in r. Checks if there are any remaining missing values in the **ActivityType** variable.

```
# Handling missing values in Total weight of the
# items Subset the merged data frame to get the
# missing values in total weight and there
# respective 'Category' and'NumOfItems'
subsetMergedData <- merged_data[missingValues$TotalWeight,
    c("Category", "NumOfItems", "TotalWeight")]

# Convert the weight range list to a data frame
weightRange <- as_tibble(weight_ranges)
```

[19]RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values

[20]RMIT University (2023) Module 6 overview summary learning objectives outliers types of outliers most common causes of outliers detecting. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)

[21]RMIT University (2023) Module 6 overview summary learning objectives outliers types of outliers most common causes of outliers detecting. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)

```
# calculate the mean of each weight ranges in the
# respective category and create a data frame
meanWeightRange <- data.frame(sapply(weightRange, mean))
```

```
# Mutate the Total weight column in the subset
# data frame 'subsetMergedData', by using values
# mean unit weight and number of items
TotalWeightM <- meanWeightRange[as.character(subsetMergedData$Category),
    ] * subsetMergedData$NumOfItems
MeanUnitWeightM <- meanWeightRange[as.character(subsetMergedData$Category),
    ]
subsetMergedData <- mutate(subsetMergedData, TotalWeight = TotalWeightM,
    MeanUnitWeight = MeanUnitWeightM)

# Row Id of missing values in total weight
# variable in the merged data frame
rowIDTotalWeight <- which(is.na(merged_data$TotalWeight))

# Replace the missing values in the Total weight
# column with calculated mean values of the
# weight ranges of the categories
merged_data$TotalWeight[rowIDTotalWeight] <- subsetMergedData$TotalWeight
```

- Handling Missing Values in **TotalWeight** variable using Weight Ranges in their respective categories:
  - This section deals with missing values in the **TotalWeight** variable by utilizing a predefined **weight_ranges** list.
  - First the **merged_data** data frame is subset to extract missing values of the **TotalWeight** variable along with corresponding **Category** and **NumOfItems** variable values.
  - Converts the **weight_ranges** list to a data frame.
  - After the conversion of the list to data frame. Mean of the each category is calculated in the converted data frame **weightRange**.
  - Using these mean values in each category, the missing values in **TotalWeight** of each product in the merged data frame is amended.
  - Using this method we get rid of the missing values in the total weight variable with the help of the predefined **weight_ranges** list.

```
# median imputation (for numerical variables)
merged_data$Salary <- impute(merged_data$Salary, fun = median)
```

- Mode Imputation for numerical variables **Salary**[22]:
  - Missing values in the **Salary** variable are imputed with the median (the value in the middle of a data set) using the `impute()` function from an `Hmisc` package in r.

```
# Check if there are any missing values in Variable ActivityType
is.na(merged_data$ActivityType) %>% sum()
```

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

[22]RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values

```r
# Check if there are any missing values in Variable TotalWeight
is.na(merged_data$TotalWeight) %>% sum()
```

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

```r
# Check if there are any missing values in Variable Salary
is.na(merged_data$Salary) %>% sum()
```

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

checking if there any missing values left in the **ActivityType**, **TotalWeight** and **Salary**.

## Reference


1. Bunnings Group Limited (2023) Our range of products - Bunnings Australia,Bunning Company Website - www.bunnings.com.au, Accessed 07 September 2023. https://www.bunnings.com.au/products
2. RMIT University (2023) Module 4 Overview Summary Learning Objectives Tidy Data Principles Common problems with messy data sets The tidyr, RMIT Canva website, Accessed 12 September 2023. http://rare-phoenix-161610.appspot.com/secured/Module_04.html
3. RMIT University (2023) Module 5 overview summary learning objectives missing data identifying missing data recode missing data excluding missing. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_05.html#Special_values
4. RMIT University (2023) Module 6 overview summary learning objectives outliers types of outliers most common causes of outliers detecting. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)
5. RMIT University (2023) Module 3 overview summary learning objectives to check attributes of R objects Learn how to convert between data types/structures. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_03.html#Summary
6. RMIT University (2023) Generating Synthetic Data. Accessed 12 September 2023. https://rare-phoenix-161610.appspot.com/secured/Module_06.html#Capping_(aka_Winsorising)