Code

Basic Data Wrangling 2 (Creating a mock dataset)

Phankawee Mukdaprakorn

30 April 2024

Introduction

This report has been conducted from a data analyst in one of top-tier australian banking institution with over 20 years of establishment. The bank got its first customer to create the first account on 03/04/2000. In this report, there are the procedures of how to generate synthetic datasets about banking industry.

Among the essential datasets maintained by banks, there are two significant datasets standing out: the **`customer dataset`** and the **`saving account dataset`**.

- The **`customer data set`** contains the information of the bank customers such as firstname, lastname, gender, age, highest level of education, profession, salary, and address. These information are uniquely identified by customer ID numbers.
- The **`saving account data set`** contains the information of the saving accounts such as the date of account opening, average monthly current balance, and the interest gained from deposits. These details are uniquely identified by account numbers and linked to the **`customer data set`** through customer ID numbers.

However, there are many assumptions which need to be made during this synthetic-data generating alongside the procedures in this report.

Setup

Hide

```
library(readr)
library(dplyr)
library(openxlsx)
library(tidyr)
library(rvest)
library(tidyverse)
library(ggplot2)
library(here)
```

Data generation

Customer Data set

Generate Customer IDs

The following code chuck is used to generate the customer IDs (**Cust_ID**) which are 8-digit numbers used to uniquely identify each row of customer information. By coding `runif(200,min = 0.1)*10^8`, it guarantees that the resulting numbers will not fall below 10,000,000, thereby preventing leading zeros in the **Cust_ID**.

Hide

```
set.seed(1111)
Cust_ID <- round(runif(200,min = 0.1)*10^8,digits = 0)
```

Generate Firstnames, Lastnames, and Genders

According from Babycenter's article (2024) (https://www.babycenter.com/baby-names/most-popular/top-baby-names-2024), I have replicated the excel file named `Top_baby_names_of_2024` which contains 100 popular baby names for both boys and girls. Subsequently, I read the excel file using `read.xlsx` function before using `pivot_longer` function to convert it in to a suitable format so that I can get `baby_names_long` which contains only **Genders** and **Firstnames**.

Hide

```
setwd(here())
baby_names <- read.xlsx("ref/Top_baby_names_of_2024.xlsx")
head(baby_names,n=5)
```

	Number	Girls	Boys
	<dbl>	<chr>	<chr>
1	1	Olivia	Noah
2	2	Emma	Liam
3	3	Amelia	Oliver
4	4	Sophia	Mateo
5	5	Charlotte	Elijah

5 rows

Hide

```
baby_names_long <- baby_names %>% pivot_longer(.,c("Girls","Boys"),names_to = "Genders", values_to = "FirstName
s") %>% select(Genders, FirstNames)
head(baby_names_long,n=5)
```

Genders	FirstNames
<chr>	<chr>
Girls	Olivia
Boys	Noah
Girls	Emma
Boys	Liam
Girls	Amelia

5 rows

For **surnames**, I sourced the data from Kimberly's article (2020) (https://www.thoughtco.com/most-common-us-surnames-1422656) which details the most common US surnames in 2020. I used `read_html` function from `rvest` package to scrape the relevant table from the website which I store the url in the variable `top_surnames_url`.

Hide

```
#Getting Surnames
top_surnames_url <- "https://www.thoughtco.com/most-common-us-surnames-1422656"
html_code <- read_html(top_surnames_url)
table_html <- html_code %>% html_nodes("table") %>% .[[1]]
top_100_surnames <- table_html %>% html_table()
#Checking top five rows of the table
head(top_100_surnames, n=5)
```

Rank	Surname	Surname Origin	Estimated Population
<int>	<chr>	<chr>	<chr>
1	Smith	English	2,442,977
2	Johnson	English, Scottish	1,932,812
3	Williams	English, Welsh	1,625,252
4	Brown	English, Scottish, Irish	1,437,026
5	Jones	English, Welsh	1,425,470

5 rows

The combination of all **firstnames** and **surnames** was achieved by using `cross_join` function from `dplyr` package. This step facilitated the generation of all possible fullnames pairs, which will be sampled from in the next step.

Hide

```
All_possible_names <- select(cross_join(baby_names_long,top_100_surnames),c(Genders, FirstNames, Surname))
All_possible_names[,"FullNames"] <- paste(All_possible_names$FirstNames," ",All_possible_names$Surname)
```

Hide

```
length(All_possible_names$FullNames)
```

```
[1] 20000
```

Hide

```
length(unique(All_possible_names$FullNames))
```

```
[1] 19900
```

After executing the cross-join operation, I rechecked the result and found out that there are some **Fullnames** which are duplicated in the `All_possible_names` dataset so I checked the duplicated **Fullnames** using `count()` and sorted it.

Hide

```
All_possible_names %>% group_by(FullNames) %>% count(sort = TRUE) %>% filter(n>1)
```

FullNames <chr>	**n** <int>
Charlie Adams	2
Charlie Allen	2
Charlie Alvarez	2
Charlie Anderson	2
Charlie Bailey	2
Charlie Baker	2
Charlie Bennet	2
Charlie Brooks	2
Charlie Brown	2
Charlie Campbell	2

1-10 of 100 rows Previous **1** 2 3 4 5 6 ... 10 Next

As a result, “Charlie” exists in both for girl and boy names which makes it duplicated. However, I did not intend to use all **Fullnames** in this data set so I did not solve any thing for this problem. I just need 200 **Fullnames** to be my banking customers.

I `set.seed()` to be `1111` so that the next time I sample this project again, it still gives me the sample set which I use now. After the sampling, I checked that the 200 sampling **Fullnames** I got are all unique from each other and put them with their genders into the new data set called `Cust_FullName` .

Hide

```
set.seed(1111)
Cust_FullName <- All_possible_names[sample(nrow(All_possible_names), 200), ]
length(unique(Cust_FullName$FullNames))
```

```
[1] 200
```

Hide

```
Cust_FullName %>% group_by(Genders) %>% count()
```

Genders <chr>	**n** <int>
Boys	91
Girls	109

2 rows

Generate Age

I made the assumption that people typically open their own accounts at the age of 15. However, the majority of customers in this bank age between 20-39 years old. Consequently, the proportions of age ranges of the customers in this bank from 15-19 years old, 20-39 years old, 40-69 years old, and 70 years old and older are 13%, 45%, 25%, and 17% respectively. Using these proportions, I can sample the **Age_range**, assign it to each customer, and store the results in the `Cust_Age` data set.

Hide

```
Age_Demographics <- data.frame(Age_range = c("15-19","20-39","40-69","70+"),
                               prevalence = c(0.13,0.45,0.25,0.17))
set.seed(1111)
Cust_Age <- data.frame(Cust_num = 1:200,
                           AgeRange = sample(Age_Demographics$Age_range,
                                             size = 200,
                                             replace = TRUE,
                                             prob = Age_Demographics$prevalence))
Cust_Age %>% count(AgeRange)
```

AgeRange <chr>	n <int>
15-19	33
20-39	88
40-69	49
70+	30

4 rows

I used `case_when` function to evaluate the **AgeRange** condition and let `R` draw the values between those ranges. Additionally, I plot the histogram to see the distribution of my customer age. The histogram looks quite right skewed since all customers falling within the 15-19 age range were all 19 years old.

Hide

```
set.seed(1111)
Cust_Age$Age <- case_when(Cust_Age$AgeRange == "15-19" ~ round(runif(nrow(Cust_Age), min = 15, max = 19)),
                          Cust_Age$AgeRange == "20-39" ~ round(runif(nrow(Cust_Age), min = 20, max = 39)),
                          Cust_Age$AgeRange == "40-69" ~ round(runif(nrow(Cust_Age), min = 40, max = 69)),
                          Cust_Age$AgeRange == "70+" ~ round(runif(nrow(Cust_Age), min = 70, max = 90)))
hist(Cust_Age$Age, breaks = 50)
```

Generate Highest Education

According to the age sampling, the minimum age of the banking customer data set is 19 so in this report, the education will be categorized into 3 groups: secondary school (SS), vocational education (VE), and university or higher education (HE), assuming that the minimum standard education for every people is secondary school. Moreover, according from the information from Australian Bureau of Statistics (2021)

(https://www.abs.gov.au/statistics/people/education/education-and-training-census/2021), I used the combination percentage of people who study secondary school, vocational education, and university or higher educations which are 1,629,622(47%), 601,891(18%), and 1,185,457(35%) people, respectively to generate the highest education for the customer data set.

Hide

```
TypeofEdu <- c("SS","VE","HE")
Edu_demographic <- data.frame(Edu_Type = c("SS","VE","HE"),
                              prevalence = c(0.47,0.18,0.35))

set.seed(1111)
Cust_Highest_Edu <- data.frame(Cust_num = 1:200,
                           High_Edu = sample(Edu_demographic$Edu_Type,
                                             size = 200,
                                             replace = TRUE,
                                             prob = Edu_demographic$prevalence))
Cust_Highest_Edu %>% count(High_Edu)
```

High_Edu <chr>	**n** <int>
HE	68
SS	93
VE	39

3 rows

Generate Profession and Salary

Based on the given highest education levels and according from Indeed Website (2023) (https://au.indeed.com/career-advice/pay-salary/average-salary-australia-by-profession) , I have created the Excel file called “Avg_weekly_salary_by_professions” to indicate the average income by qualifications and occupations.

Hide

```
Avg_Sal_by_Professions <- read.xlsx("ref/Avg_weekly_salary_by_professions.xlsx")
head(Avg_Sal_by_Professions,5)
```

	Qualification <chr>	**Professions** <chr>	**Avg_Weekly_Salary** <dbl>
1	No tertiary qualification	Barista	900
2	No tertiary qualification	Bartender	900
3	No tertiary qualification	Courier	900
4	No tertiary qualification	Cleaner	900
5	No tertiary qualification	Sales assistant	900

5 rows

Firstly, I categorized the professions listed in the Excel file into three educational tiers: secondary school(SS), vocational education(VE), and university or higher education (HE). In this case, I grouped people with “No tertiary qualification” with a secondary school group (`SS_educations`), “Certificate I-II”,“Certificate III-IV”,“Diploma” with a vocational education group (`VE_educations`), and “Bachelor degree”,“Graduate diploma”, “Postgraduate degree” with a high education group (`HE_educations`). Subsequently, I used `case_when()` to assign the highest educational levels to each profession in the `Avg_Sal_by_Professions` dataset, creating a new column labeled **High_Edu**.

Hide

```
SS_educations <- c("No tertiary qualification")
VE_educations <- c("Certificate I-II","Certificate III-IV","Diploma")
HE_educations <- c("Bachelor degree","Graduate diploma", "Postgraduate degree")

Avg_Sal_by_Professions$High_Edu <- case_when(Avg_Sal_by_Professions$Qualification %in% SS_educations ~ "SS",
                                        Avg_Sal_by_Professions$Qualification %in% VE_educations ~ "VE",
                                        Avg_Sal_by_Professions$Qualification %in% HE_educations ~ "HE")
```

I also extracted the professions in each highest educational levels and store them in separate vector variables, which are now ready for use in the next sampling process.

Hide

```
SS_Prof <- Avg_Sal_by_Professions %>% filter(Qualification %in% SS_educations) %>% pull(Professions)
VE_Prof <- Avg_Sal_by_Professions %>% filter(Qualification %in% VE_educations) %>% pull(Professions)
HE_Prof <- Avg_Sal_by_Professions %>% filter(Qualification %in% HE_educations) %>% pull(Professions)
```

After the preparation, I generated the new dataframe called `Cust_Prof_Salary` which stores the information about the occupations and their averaged weekly salary based on their occupations. In this step, to be more realistic, I used the numbers of size when sampling the occupations based on their highest educational level, assuming that it should result in some relations between educations, occupations, and salaries.

Hide

```
set.seed(1111)
Cust_Prof_Salary <- data.frame(Professions = sample(SS_Prof,size = 93,replace = TRUE))
Cust_Prof_Salary <- rbind(Cust_Prof_Salary, data.frame(Professions = sample(VE_Prof,size = 39,replace = TRUE)))
Cust_Prof_Salary <- rbind(Cust_Prof_Salary, data.frame(Professions = sample(HE_Prof,size = 68,replace = TRUE)))
Cust_Prof_Salary$Num <- c(1:200)

## Getting other information of these professions such as highest educational levels and average weekly salary
Cust_Prof_Salary <- Cust_Prof_Salary %>% left_join(.,Avg_Sal_by_Professions) %>% select(Num,Professions,Avg_Wee
kly_Salary,High_Edu)
```

```
Joining with `by = join_by(Professions)`
```

Hide

```
head(Cust_Prof_Salary,5)
```

	Num	Professions	Avg_Weekly_Salary	High_Edu
	<int>	<chr>	<dbl>	<chr>
1	1	Cleaner	900	SS
2	2	Pet sitter	900	SS
3	3	Bartender	900	SS
4	4	Bartender	900	SS
5	5	Cleaner	900	SS

5 rows

And these were top ten occupations from my banking customer data set.

Hide

```
Cust_Prof_Salary %>% group_by(Professions) %>% summarise(count=n(),percent = count / nrow(.) * 100) %>% arrange
(desc(count)) %>% head(n=10)
```

Professions	count	percent
<chr>	<int>	<dbl>
Bartender	20	10.0
Cleaner	20	10.0
Barista	17	8.5
Sales assistant	16	8.0
Pet sitter	13	6.5
Courier	7	3.5
Occupational therapist	7	3.5
Chemical engineer	6	3.0
Data scientist	6	3.0
Exercise physiologist	6	3.0

1-10 of 10 rows

What I needed to do next with `Cust_Prof_Salary` was converting the average weekly wage into the monthly salary. In addition, I generated some differences between each row's monthly salary using `rnorm` function.

Hide

```
Cust_Prof_Salary$Avg_Monthly_Salary <- Cust_Prof_Salary$Avg_Weekly_Salary*4

set.seed(1111)
Cust_Prof_Salary$Difference <- rnorm(n=200,mean = 0,sd=2)*100
Cust_Prof_Salary$Monthly_Salary <- round(Cust_Prof_Salary$Avg_Monthly_Salary+ Cust_Prof_Salary$Difference,digit
s = -2)
```

Generate Address

I used the dataset from City of Melbourne (2015) (https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/street-names/exports/csv?lang=en&timezone=Australia%2FSydney&use_labels=true&delimiter=%2C) which contains the street names and some addresses in Melbourne to sample the random addresses.

Hide

```
Street_names_Melb <- read_csv("https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/street-name
s/exports/csv?lang=en&timezone=Australia%2FSydney&use_labels=true&delimiter=%2C", col_select = c("mccid_gis",
"name"))
dim(Street_names_Melb)
```

```
[1] 2876    2
```

Hide

```
head(Street_names_Melb,n=5)
```

mccid_gis	name
<dbl>	<chr>
39	BUCKHURST LA
65	GIBSON AV
63	BARKLY AV
21	WRECKYN PLACE
31	CHAUVEL ST

5 rows

I put the **mccid_gis** and **name** of the streets with “Melbourne, Australia” altogether to generate the full addresses and put them into **Address** column.

Hide

```
Street_names_Melb$Address <- paste(Street_names_Melb$mccid_gis," ",Street_names_Melb$name," ", "Melbourne, Aust
ralia")
```

After that, I sampled 200 addresses which will be the customer addresses and put them in the `Cust_Address` data set.

Hide

```
set.seed(1111)
Cust_Address <- data.frame(Num = c(1:200),
                           Address = sample(Street_names_Melb$Address,size = 200,replace=FALSE))
```

I also made some information in this address column to be missing values by setting 5% of observations to be missing.

Hide

```
set.seed(1111)
Cust_Address$rand <- runif(200,min = 0, max = 1)
Cust_Address$Address <- case_when(Cust_Address$rand >= 0.05 ~ Cust_Address$Address)
```

Finally, I combined all customer information together to compile the banking `Customer_Dataset` .

Hide

```
Customer_Dataset <- data.frame(Cust_ID = Cust_ID,
                               Firstname = Cust_FullName$FirstNames,
                               Lastname = Cust_FullName$Surname,
                               Gender = case_when(Cust_FullName$Genders == "Girls" ~ "Female",
                                                  Cust_FullName$Genders == "Boys" ~ "Male"),
                               Age = Cust_Age$Age,
                               Highest_Education = Cust_Prof_Salary$High_Edu,
                               Profession = Cust_Prof_Salary$Professions,
                               Monthly_Salary = Cust_Prof_Salary$Monthly_Salary,
                               Address = Cust_Address$Address)
```

Deposit Account Data set

Generate Account Numbers

I used the same method to generate 200 random 8-digit numbers and put them in the variable vector **Acc_Num**.

Hide

```
set.seed(2222)
Acc_Num <- round(runif(200,min = 0.1)*10^8,digits = 0)
```

Generate Account_open_date

For the account opening date, I used the date from 1/1/2000 until 30/12/2023 as the bank has operated for over 20 years. After that, I sampled 200 values from **day**, **month**, **year** variables, put them together using `paste()` function, and stored them in `Acc_open_date` . I also used `dmy()` function from `lubridate` package to convert those dates into a proper format. Subsequently, I generated the `end_date` variable to store the date `01/04/2024` aiming to calculate the number of months that each **Acc_open_date** has been deposited their money with the bank. In order to calculate **Month_diff**, I used `interval()` function from `lubridate` package to do it.

Hide

```
year <- c(2000:2023)
month <- c(1:12)
day <- c(1:30)

#Sample the Acc_open_date
set.seed(2222)
Acc_open_date <- data.frame(Date = paste(sample(day, size = 200,replace = TRUE),sample(month,size = 200,replace
= TRUE),sample(year,size = 200,replace = TRUE)))

#Convert the format
Acc_open_date$Open_date <- lubridate::dmy(Acc_open_date$Date)

#Calculate the Month_diff
end_date <- lubridate::dmy("01/04/2024")
Acc_open_date$Month_diff <- lubridate::interval(Acc_open_date$Open_date, end_date) %/% months(1)
```

Generate Current Balance

I generated monthly average current balance (**Current_Bal**) based on the customer income (**Monthly_Salary**) which I generated from the previous data set. According to Cameron's (2024) (https://mozo.com.au/savings-accounts/guides/how-much-of-my-income-should-i-save#:~:text=The%2050%2F30%2F20%20approach%20to%20saving%20is%20one%20of,%2C%20and%2020%25%20towards%20savings), people should save 20% of their monthly income. I can create some correlation between savings (**Current_Bal**) and income (**Monthly_Salary**) with the following relationship:

$$CurrentBalance = 0.2 * (Numberof DepositMonth * MonthlyIncome)$$

Hide

```
Acc_Current_Balance <- data.frame(Current_Bal = 0.2*Acc_open_date$Month_diff*Customer_Dataset$Monthly_Salary)
```

Before moving on to the next step, I put some missing values into the account current balance for 5% of total rows and stored them to the new column named **Current_Balance**.

Hide

```
set.seed(2222)
Acc_Current_Balance$rand <- runif(200, min = 0, max = 1)
Acc_Current_Balance$Current_Balance <- case_when(Acc_Current_Balance$rand >= 0.05 ~ Acc_Current_Balance$Current
_Bal)
```

Generate Interest

Given all accounts get 3% annual interest rate as a simple interest rate, I can calculate the overall interest for each account with the following code chunk.

Hide

```
Acc_Current_Balance$Interest <- Acc_Current_Balance$Current_Balance*(Acc_open_date$Month_diff/12)*0.03
```

I combined all account information together to get the `Saving_Acc_Dataset` .

Hide

```
Saving_Acc_Dataset <- data.frame(Acc_Num = Acc_Num,
                                 Cust_ID = Cust_ID,
                                 Acc_open_date = Acc_open_date$Open_date,
                                 Average_Current_Balance = Acc_Current_Balance$Current_Balance,
                                 Interest = Acc_Current_Balance$Interest)
```

Merging data sets

Hide

```
Banking_Cust_Acc <- merge(Customer_Dataset,Saving_Acc_Dataset, by = "Cust_ID")
```

This was the combination of `Customer_Dataset` and `Saving_Acc_Dataset` joining with 1-to-1 relationship through the **Cust_ID**. Each row has information of the customer and his/her own saving account.

Checking structure of combined data

Hide

```
str(Banking_Cust_Acc)
```

```
'data.frame':   200 obs. of  13 variables:
 $ Cust_ID               : num  10152118 10863158 11475396 11650515 12355360 ...
 $ Firstname             : chr  "Michael" "Riley" "Dylan" "Grayson" ...
 $ Lastname              : chr  "Scott" "Brooks" "Sanchez" "Mitchell" ...
 $ Gender                : chr  "Male" "Female" "Male" "Male" ...
 $ Age                   : num  36 32 33 30 39 38 38 24 26 26 ...
 $ Highest_Education     : chr  "SS" "HE" "HE" "SS" ...
 $ Profession            : chr  "Barista" "Pharmacist" "Architect" "Sales assistant" ...
 $ Monthly_Salary        : num  3400 6700 6800 3700 3600 3700 7200 5100 6100 3300 ...
 $ Address               : chr  NA NA NA NA ...
 $ Acc_Num               : num  43871310 17747391 21074002 29915132 78105574 ...
 $ Acc_open_date         : Date, format: "2003-04-28" "2015-11-24" "2020-03-05" ...
 $ Average_Current_Balance: num  170680 134000 65280 25900 90720 ...
 $ Interest              : num  107102 33500 7834 2266 28577 ...
```

According to the structure of the `Banking_Cust_Acc` data set, there were 200 rows with 13 variables containing the banking customer data. However, there were some variables which datatypes were needed to be converted. I converted customer IDS (**Cust_ID**) and account numbers (**Acc_Num**) to be integer variables. Moreover, for the highest educational levels (**Highest_Education**), I would take it as an ordered factor given the highest education are HE>VE>SE respectively. furthermore, I converted genders (**Gender**) and professions (**Profession**) to be factor variables.

Hide

```
Banking_Cust_Acc$Cust_ID <- as.integer(Banking_Cust_Acc$Cust_ID)
Banking_Cust_Acc$Gender <- as.factor(Banking_Cust_Acc$Gender)
Banking_Cust_Acc$Highest_Education <- Banking_Cust_Acc$Highest_Education %>% factor(.,
                                                                                   levels = c("SS","VE","HE"),
labels = c("Secondary School or lower", "Vocational Educations","University or Higher educations"), ordered = T
RUE)
Banking_Cust_Acc$Profession <- as.factor(Banking_Cust_Acc$Profession)
Banking_Cust_Acc$Acc_Num <- as.integer(Banking_Cust_Acc$Acc_Num)
```

I rechecked some results after the conversions.

Hide

```
Banking_Cust_Acc$Profession %>% levels()
```

```
 [1] "Accountant"                    "Accounting clerk"              "Actuary"
 [4] "Aircraft maintenance engineer" "Architect"                     "Barista"
 [7] "Bartender"                     "Call centre worker"            "Carpenter"
[10] "Chemical engineer"             "Cleaner"                       "Coach driver"
[13] "Courier"                       "Crane operator"                "Data scientist"
[16] "Dental hygienist"              "Digital marketer"              "Economist"
[19] "Electrical engineer"           "Exercise physiologist"         "Finance manager"
[22] "Information officer"           "Legal clerk"                   "Mechanical engineer"
[25] "Nurse manager"                 "Occupational therapist"        "Paramedic"
[28] "Payroll officer"               "Personal trainer"              "Pet sitter"
[31] "Pharmacist"                    "Power plant operator"          "Project administrator"
[34] "Psychiatrist"                  "Restaurant manager"            "Sales assistant"
[37] "School principal"              "Sonographer"                   "Stonemason"
[40] "Urban planner"
```

Hide

```
class(Banking_Cust_Acc$Highest_Education)
```

```
[1] "ordered" "factor"
```

Hide

```
str(Banking_Cust_Acc)
```

```
'data.frame':   200 obs. of  13 variables:
 $ Cust_ID               : int  10152118 10863158 11475396 11650515 12355360 13114759 13478665 13578776 138553
94 15439670 ...
 $ Firstname             : chr  "Michael" "Riley" "Dylan" "Grayson" ...
 $ Lastname              : chr  "Scott" "Brooks" "Sanchez" "Mitchell" ...
 $ Gender                : Factor w/ 2 levels "Female","Male": 2 1 2 2 2 2 2 1 2 1 ...
 $ Age                   : num  36 32 33 30 39 38 38 24 26 26 ...
 $ Highest_Education     : Ord.factor w/ 3 levels "Secondary School or lower"<..: 1 3 3 1 1 1 3 2 3 1 ...
 $ Profession            : Factor w/ 40 levels "Accountant","Accounting clerk",..: 6 31 5 36 11 6 15 9 26 30
...
 $ Monthly_Salary        : num  3400 6700 6800 3700 3600 3700 7200 5100 6100 3300 ...
 $ Address               : chr  NA NA NA NA ...
 $ Acc_Num               : int  43871310 17747391 21074002 29915132 78105574 10163913 97907349 87308660 933489
31 45889632 ...
 $ Acc_open_date         : Date, format: "2003-04-28" "2015-11-24" "2020-03-05" ...
 $ Average_Current_Balance: num  170680 134000 65280 25900 90720 ...
 $ Interest              : num  107102 33500 7834 2266 28577 ...
```

Generate summary statistics

I wanted to compare the summary statistics among the different tiers of highest education. I wanted to check if there was any difference in monthly salary among this dimension. Thus I grouped `Banking_Cust_Acc` by **Highest_Education** and used `summarise()` function to calculate minimum values, 1st quantiles, medians, 3rd quantiles, maximum values, mean values, standard deviations, number of observations, and number of missing values (if any).

Hide

```
Banking_Cust_Acc %>% group_by(Highest_Education) %>% summarise(Min = min(Monthly_Salary, na.rm = TRUE),
                                                               Q1 = quantile(Monthly_Salary, probs = .25, na.rm
= TRUE),
                                                               Median = median(Monthly_Salary, na.rm = TRUE),
                                                               Q3 = quantile(Monthly_Salary, probs = .75, na.rm
= TRUE),
                                                               Max = max(Monthly_Salary, na.rm = TRUE),
                                                               Mean = mean(Monthly_Salary, na.rm = TRUE),
                                                               SD = sd(Monthly_Salary, na.rm = TRUE),
                                                               n = n(),
                                                               Missing = sum(is.na(Monthly_Salary)))
```

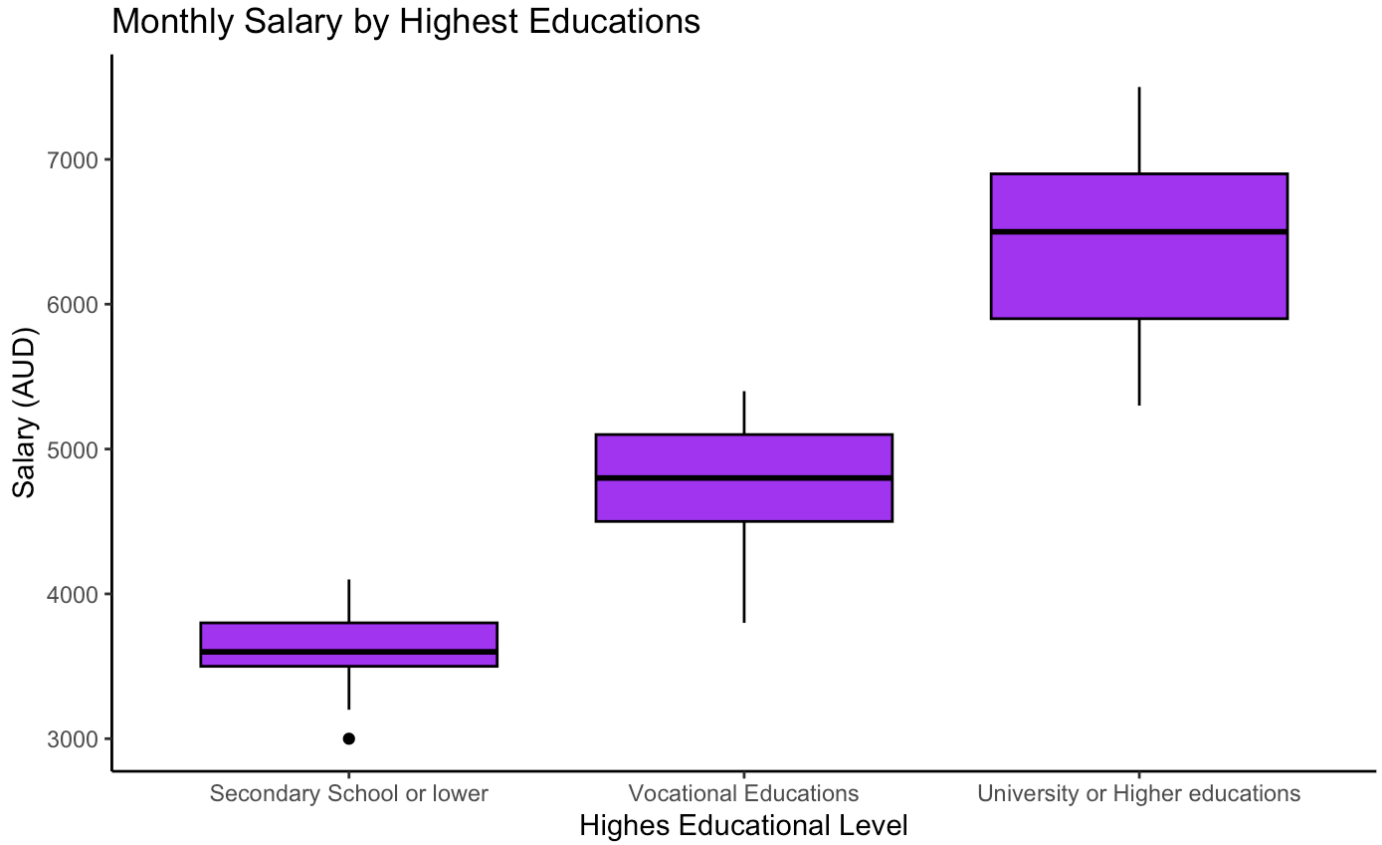
Highest_Education	Min	Q1	Median	Q3	Max	Mean	SD	n	Missing
<ord>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>
Secondary School or lower	3000	3500	3600	3800	4100	3636.559	213.0447	93	0

Highest_Education	Min	Q1	Median	Q3	Max	Mean	SD	n	Missing
<ord>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>
Vocational Educations	3800	4500	4800	5100	5400	4756.410	409.6178	39	0
University or Higher educations	5300	5900	6500	6900	7500	6405.882	583.3210	68	0

3 rows

Hide

```
ggplot(Banking_Cust_Acc, aes(x = Highest_Education, y = Monthly_Salary)) +
  geom_boxplot(fill = "purple", color = "black") +
  xlab("Highes Educational Level") +
  ylab("Salary (AUD)") +
  theme_classic() +
  labs(title = "Monthly Salary by Highest Educations")
```



As be seen from the boxplots, customers who graduated from university or higher educations tend to have higher range of monthly salary while customers with the secondary school or lower educational levels tend to have narrow range of salary. There is an outlier in the customers in secondary school or lower group.

I also wanted to check if there is any different salary among different professions.

Hide

```
Banking_Cust_Acc %>% group_by(Profession) %>% summarise(Min = min(Age, na.rm = TRUE),
                                                        Q1 = quantile(Age, probs = .25, na.rm = TRUE),
                                                        Median = median(Age, na.rm = TRUE),
                                                        Q3 = quantile(Age, probs = .75, na.rm = TRUE),
                                                        Max = max(Age, na.rm = TRUE),
                                                        Mean = mean(Age, na.rm = TRUE),
                                                        SD = sd(Age, na.rm = TRUE),
                                                        n = n(),
                                                        Missing = sum(is.na(Age))) %>% arrange(desc(n))
%>% head(10)
```

Profession	Min	Q1	Median	Q3	M...	Mean	SD	n	Missing
<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>
Bartender	19	19.75	28.0	48.25	88	37.15000	22.17223	20	0

Profession	Min	Q1	Median	Q3	M…	Mean	SD	n	Missing
<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>
Cleaner	19	21.25	37.5	55.00	88	41.50000	21.02505	20	0
Barista	19	22.00	36.0	50.00	83	41.76471	22.96064	17	0
Sales assistant	19	28.50	35.5	68.75	90	47.18750	25.49567	16	0
Pet sitter	19	30.00	49.0	65.00	89	48.38462	23.71546	13	0
Courier	24	27.50	30.0	56.00	77	42.57143	20.43573	7	0
Occupational therapist	19	23.50	37.0	53.50	84	42.00000	24.15229	7	0
Chemical engineer	32	34.25	35.0	55.25	67	44.16667	15.86716	6	0
Data scientist	25	37.25	52.0	66.75	87	53.33333	23.60226	6	0
Exercise physiologist	25	49.75	57.5	74.25	81	58.00000	20.78461	6	0

1-10 of 10 rows

As be seen from the summary statistics, from top ten banking customer professions, Bartender and Cleaner tend to be popular. Moreover, bartender profession have many customers aged approximately 37 years old working which is relatively low comparing to other top-tens. On the other hand, from this data set, exercise physiologist is the last top-ten profession and has many elderly customers working because the mean value is 58 years old. Sales assistant tends to be the widest range of age of customers working this occupation since the standard deviation is highest (25.49567).

I also wanted to check if there is any different salary among different professions.

Hide

```
Banking_Cust_Acc %>% group_by(Profession) %>% summarise(Min = min(Monthly_Salary, na.rm = TRUE),
                                                        Q1 = quantile(Monthly_Salary, probs = .25, na.rm
= TRUE),
                                                        Median = median(Monthly_Salary, na.rm = TRUE),
                                                        Q3 = quantile(Monthly_Salary, probs = .75, na.rm
= TRUE),
                                                        Max = max(Monthly_Salary, na.rm = TRUE),
                                                        Mean = mean(Monthly_Salary, na.rm = TRUE),
                                                        SD = sd(Monthly_Salary, na.rm = TRUE),
                                                        n = n(),
                                                        Missing = sum(is.na(Monthly_Salary))) %>% arrang
e(desc(Mean)) %>% head(10)
```

Profession	Min	Q1	Median	Q3	Max	Mean	SD	n	Missing
<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>
Data scientist	6800	6925	7100	7200	7400	7083.333	222.8602	6	0
Architect	6800	6950	7050	7175	7400	7075.000	250.0000	4	0
Economist	6800	6875	7000	7200	7500	7075.000	309.5696	4	0
Urban planner	6900	6975	7050	7125	7200	7050.000	212.1320	2	0
Psychiatrist	7000	7000	7000	7000	7000	7000.000	*NA*	1	0
Chemical engineer	6700	6825	6900	6900	7200	6900.000	167.3320	6	0
Mechanical engineer	6600	6600	6600	6600	6600	6600.000	*NA*	1	0
Sonographer	6400	6500	6500	6700	6800	6580.000	164.3168	5	0
Nurse manager	6300	6400	6500	6600	6700	6500.000	282.8427	2	0
Occupational therapist	6100	6350	6500	6550	6600	6428.571	179.9471	7	0

1-10 of 10 rows

As be seen from the summary statistics, data scientist is the profession which earns the highest salary as its mean is the highest. Furthermore, the profession that tends to have a widest range of salary among is the economist since its standard deviation is 309.5696.

Scanning data for Missing Values

Firstly, I used `sapply` function to execute `anyNA()` function over the `Banking_Cust_Acc` data set to see if there is any NA value. After locating the columns which contain the missing values, I used `which()` function in combination with `is.na()` function to locate the specific locations of those NAs.

Hide

```
sapply(Banking_Cust_Acc, anyNA)
```

```
               Cust_ID              Firstname               Lastname                 Gender 
                 FALSE                  FALSE                  FALSE                  FALSE 
                   Age      Highest_Education             Profession         Monthly_Salary 
                 FALSE                  FALSE                  FALSE                  FALSE 
               Address                Acc_Num          Acc_open_date Average_Current_Balance 
                  TRUE                  FALSE                  FALSE                   TRUE 
              Interest 
                  TRUE 
```

Hide

```
which(is.na(Banking_Cust_Acc$Address))
```

```
[1] 1 2 3 4 5 6 7 8 9
```

Hide

```
which(is.na(Banking_Cust_Acc$Average_Current_Balance))
```

```
[1]   6  35  55  84 120 154 186 196 198
```

Hide

```
which(is.na(Banking_Cust_Acc$Interest))
```

```
[1]   6  35  55  84 120 154 186 196 198
```

Overall, there were 18 rows that contained at least 1 missing value so I can't just omit all 18 rows of data as it accounts for 9% of my data set. I needed to apply the basic missing value imputation techniques for **Average_Current_Balance** and **Interest**. However, I can't impute the **Address** variable so I decided to omit them off if the address variable will be used.

Hide

```
library(Hmisc)
Banking_Cust_Acc$Cleaned_Current_Balance <- impute(Banking_Cust_Acc$Average_Current_Balance, fun = median)
Banking_Cust_Acc$Cleaned_Interest <- impute(Banking_Cust_Acc$Interest, fun = median)
```

I chose median instead of mean to prevent the bias which could happen if there is any outlier in column **Average_Current_Balance** and **Interest**. However, doing so might not be the best way as median is calculated from the overall **Average_Current_Balance** and **Interest** which come from various occupations and many highest educational levels. It might misleads the findings. A better method would be finding the common features among those missing-value rows and grouping them by that common feature with other completed observations to calculate the mean or median value specifically for that group. Then using that specific mean or median to replace the NA values would be the better way to deal with these missing values.

After the imputations, the two cleaned columns should be usable. If I wanted to use `sum()` function with **Average_Current_Balance** and **Interest**, the results would be NA because these two columns contained NA values. However, if I used the same function with **Cleaned_Current_Balance** and **Cleaned_Interest**, it should be able to generate the results since there were no NA value in the columns.

Hide

```
#Use sum() with Average_Current_Balance and Interest
sum(Banking_Cust_Acc$Average_Current_Balance)
```

```
[1] NA
```

Hide

```
sum(Banking_Cust_Acc$Intereset)
```

```
[1] 0
```

Hide

```
#Use sum() with Cleaned_Current_Balance and Cleaned_Interest
sum(Banking_Cust_Acc$Cleaned_Current_Balance)
```

```
[1] 24770920
```

Hide

```
sum(Banking_Cust_Acc$Cleaned_Interest)
```

```
[1] 11025966
```

To recheck precisely, I used `which()` and `is.na()` to check for NA values with `Cleaned_Current_Balance` and `Cleaned_Interest`.

Hide

```
which(is.na(Banking_Cust_Acc$Cleaned_Current_Balance))
```

```
integer(0)
```

Hide

```
which(is.na(Banking_Cust_Acc$Cleaned_Interest))
```

```
integer(0)
```

Scanning data for Correlations

I also discovered some correlations between some variables as followed. Firstly, I needed to calculate the number of months the customers had deposited their money with our bank and stored them in **Deposit_Month_Length**.

Hide

```
end_date <- lubridate::dmy("01/04/2024")
Banking_Cust_Acc$Deposit_Month_Length <- lubridate::interval(Banking_Cust_Acc$Acc_open_date, end_date) %/% mont
hs(1)
```

I was going to create a correlation matrix to check if in `Banking_Cust_Acc` data set, there is any correlation among the numeric variables. I put all numeric variables into the new data set named `Cor_Data`.

Hide

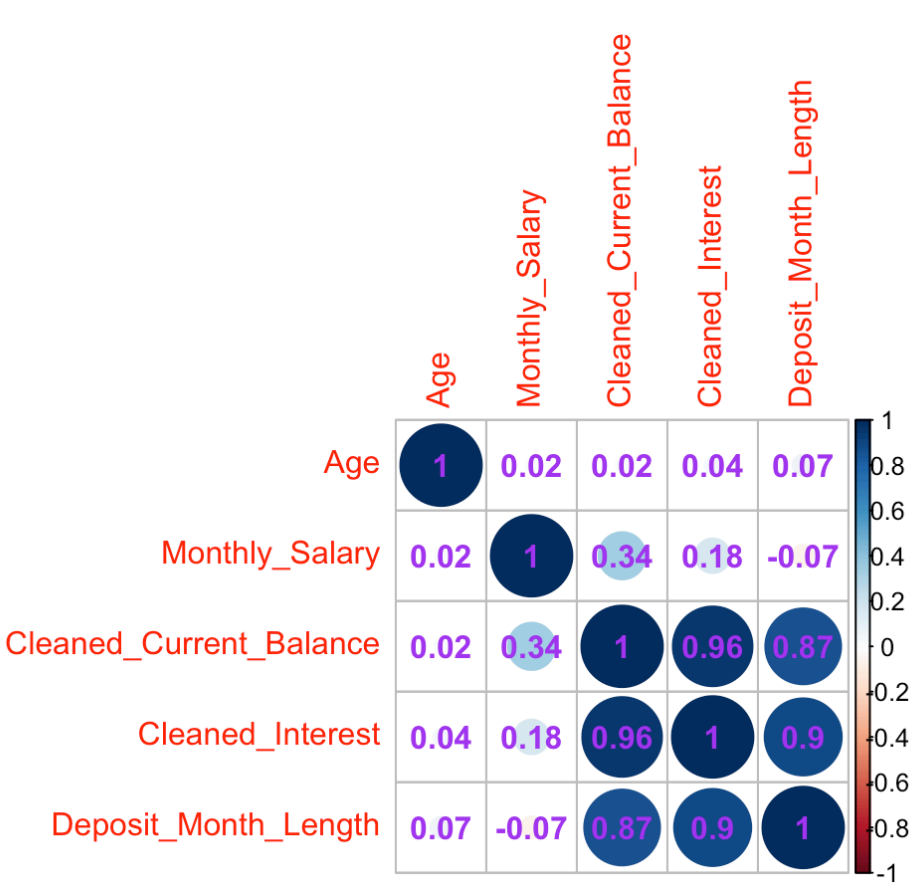
```
Cor_Data <- Banking_Cust_Acc %>% select(Age, Monthly_Salary, Cleaned_Current_Balance, Cleaned_Interest, Deposit
_Month_Length) %>% cor() %>% round(3)
Cor_Data
```

```
                        Age Monthly_Salary Cleaned_Current_Balance Cleaned_Interest Deposit_Month_Length
Age                   1.000          0.022                   0.022           0.036                0.066
Monthly_Salary        0.022          1.000                   0.335           0.179               -0.070
Cleaned_Current_Balance 0.022          0.335                   1.000           0.956                0.866
Cleaned_Interest      0.036          0.179                   0.956           1.000                0.895
Deposit_Month_Length  0.066         -0.070                   0.866           0.895                1.000
```

Let's see in a correlogram to be easier. I used `corrplot()` function from `corrplot` package to demonstrate this correlation values. The big blue circles showed the strong positive correlations between two variables while if there are any red circles, it indicated the negative correlations between two variables.

Hide

```
corrplot::corrplot(Cor_Data, method = 'circle', addCoef.col = 'purple')
```



According from the correlogram, there are strong positive correlations among **Cleaned_Current_Balance**, **Cleaned_Interest**, and **Deposit_Month_Length** because the higher current balance and loner deposit month length tend to result in the higher interest. Moreover, there are slight positive correlations between **Monthly_Salary** and **Interest** as people tend to deposit more money if they earn a higher salary to get more interest.

References

ABS. (Statistics ABo) (2021) *Education and training: Census*, Australian Bureau of Statistics website, accessed 19 April 2024. https://www.abs.gov.au/statistics/people/education/education-and-training-census/2021#cite-window1 (https://www.abs.gov.au/statistics/people/education/education-and-training-census/2021#cite-window1)

Babycenter (2024) *Most popular baby names of 2024*, Babycenter website, accessed 19 April 2024. https://www.babycenter.com/baby-names/most-popular/top-baby-names-2024 (https://www.babycenter.com/baby-names/most-popular/top-baby-names-2024)

benson23 (6 April 2023) 'How to round up to the nearest 10 (or 100 or X)?', *stackoverflow*, accessed 25 April 2024. https://stackoverflow.com/questions/6461209/how-to-round-up-to-the-nearest-10-or-100-or-x (https://stackoverflow.com/questions/6461209/how-to-round-up-to-the-nearest-10-or-100-or-x)

Chris (28 December 2017) 'sorting a table in R by count', *stackoverflow*, accessed 25 April 2024. https://stackoverflow.com/questions/48001805/sorting-a-table-in-r-by-count (https://stackoverflow.com/questions/48001805/sorting-a-table-in-r-by-count)

Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. *Journal of Statistical Software*, 40(3), 1-25. URL https://www.jstatsoft.org/v40/i03/ (https://www.jstatsoft.org/v40/i03/).

GG (Geeks Gf) (2023) Correlation Matrix in R Programming, Geeks for Geeks website, accessed 28 April 2024. https://www.geeksforgeeks.org/correlation-matrix-in-r-programming/ (https://www.geeksforgeeks.org/correlation-matrix-in-r-programming/)

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Harrell Jr F (2024). *Hmisc: Harrell Miscellaneous*. R package version 5.1-2, https://CRAN.R-project.org/package=Hmisc (https://cran.r-project.org/package=Hmisc).

II SPS (23 January 2024) 'Mastering Date Calculations in R: A Guide to Calculating Months with Base R and lubridate', *R-bloggers*, accessed 23 April 2024. https://www.r-bloggers.com/2024/01/mastering-date-calculations-in-r-a-guide-to-calculating-months-with-base-r-and-lubridate/ (https://www.r-bloggers.com/2024/01/mastering-date-calculations-in-r-a-guide-to-calculating-months-with-base-r-and-lubridate/)

Indeed (Indeed) (2023) *Average Salary in Australia by Profession and Industry*, Indeed website, accessed 20 April 2024. https://au.indeed.com/career-advice/pay-salary/average-salary-australia-by-profession (https://au.indeed.com/career-advice/pay-salary/average-salary-australia-by-profession)

Melbourne Co (Team CoMOD) (2015) *Street names* [CSV file], City of Melbourne website, accessed 20 April 2024. https://data.melbourne.vic.gov.au/explore/dataset/street-names/information/ (https://data.melbourne.vic.gov.au/explore/dataset/street-names/information/)

Powell K (19 Febuary 2020) ‘Top 100 Most Common Last Names in the United States’, *ThoughCo.*, accessed 19 April 2024. https://www.thoughtco.com/most-common-us-surnames-1422656 (https://www.thoughtco.com/most-common-us-surnames-1422656)

R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/ (https://www.R-project.org/).

Schauberger P, Walker A (2023). *openxlsx: Read, Write and Edit xlsx Files*. R package version 4.2.5.2, https://CRAN.R-project.org/package=openxlsx (https://cran.r-project.org/package=openxlsx).

Taheri S (2024) ‘Module 2 Get: Importing, Scraping and Exporting Data with R’ [Course Website], Royal Melbourne Institute of Technology, 124 La Trobe St, Melbourne VIC 3000.

Taheri S (2024) ‘Module 3 Understand: Understanding Data and Data Structures’ [Course Website], Royal Melbourne Institute of Technology, 124 La Trobe St, Melbourne VIC 3000.

Taheri S (2024) ‘Module 4 Tidy and Manipulate: Tidy Data Principles and Manipulating Data’ [Course Website], Royal Melbourne Institute of Technology, 124 La Trobe St, Melbourne VIC 3000.

Taheri S (2024) ‘Module 5 Scan: Missing Values’ [Course Website], Royal Melbourne Institute of Technology, 124 La Trobe St, Melbourne VIC 3000.

Taheri S (2024) ‘Generating Synthetic Data’ Royal Melbourne Institute of Technology, 124 La Trobe St, Melbourne VIC 3000.

Taiyun Wei and Viliam Simko (2021). R package ‘corrplot’: Visualization of a Correlation Matrix (Version 0.92). Available from https://github.com/taiyun/corrplot (https://github.com/taiyun/corrplot)

Thomson C (24 January 2024) ‘How much of my income should I save?’, *Mozo*, accessed 21 April 2024. https://mozo.com.au/savings-accounts/guides/how-much-of-my-income-should-i-save# (https://mozo.com.au/savings-accounts/guides/how-much-of-my-income-should-i-save#):~:text=The%2050%2F30%2F20%20approach%20to%20saving%20is%20one%20of,%2C%20and%2020%25%20towards%20savings

Wickham H, Hester J, Bryan J (2024). *readr: Read Rectangular Text Data*. R package version 2.1.5, https://CRAN.R-project.org/package=readr (https://cran.r-project.org/package=readr).

Wickham H, François R, Henry L, Müller K, Vaughan D (2023). *dplyr: A Grammar of Data Manipulation*. R package version 1.1.4, https://CRAN.R-project.org/package=dplyr (https://cran.r-project.org/package=dplyr).

Wickham H, Vaughan D, Girlich M (2024). *tidyr: Tidy Messy Data*. R package version 1.3.1, https://CRAN.R-project.org/package=tidyr (https://cran.r-project.org/package=tidyr).

Wickham H (2024). *rvest: Easily Harvest (Scrape) Web Pages*. R package version 1.0.4, https://CRAN.R-project.org/package=rvest (https://cran.r-project.org/package=rvest).

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). “Welcome to the tidyverse.” *Journal of Open Source Software*, *4*(43), 1686. doi:10.21105/joss.01686 (doi:10.21105/joss.01686) https://doi.org/10.21105/joss.01686 (https://doi.org/10.21105/joss.01686).