Predicitive Analytics

Daniel Ma

31/05/2021

```
library(tidyverse)
library(fpp3)
library(readxl)
library(lubridate)
my_data <- read_excel("ANZ synthesised transaction dataset.xlsx")
my_data$date <- as.Date(my_data$date, "yyyy-mm-dd") # will convert the date column into a time value
```

Data Cleaning

As we are starting with the same dataset, we will need to clean it first.

```
library(factoextra)
fact_cols <- names(dplyr::select_if(subset(my_data,select = -c(account,long_lat,merchant_id,first_name,
my_data[,fact_cols] <- lapply(my_data[,fact_cols], factor)
```

Annual Salary

To predict salary we need to find where salary exists in the dataset.

```
unique(my_data$txn_description)
```

```
## [1] POS         SALES-POS   PAYMENT     INTER BANK PAY/SALARY PHONE BANK
## Levels: INTER BANK PAY/SALARY PAYMENT PHONE BANK POS SALES-POS
```

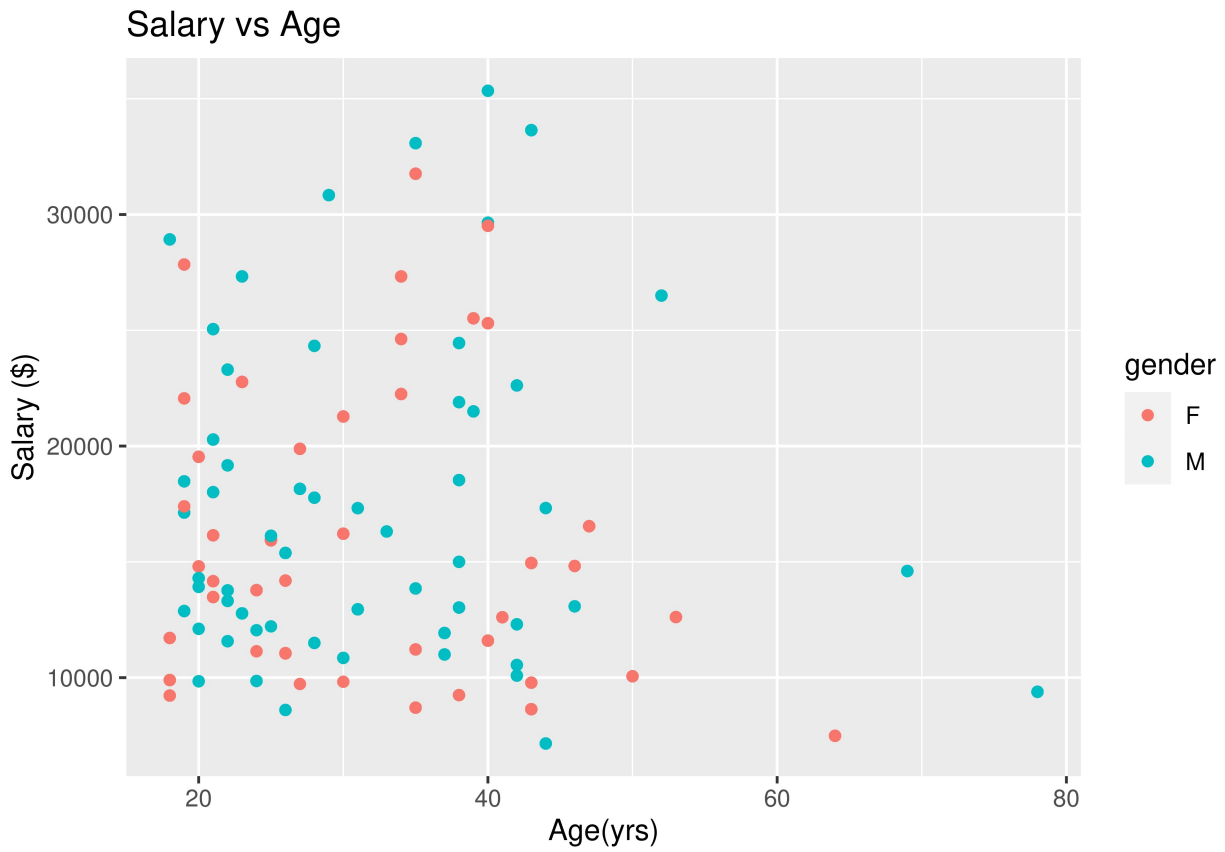
We notice that there is a desciption of "PAY/SALARY" in txn_description column. And so salary exists under the txn_description column. We need to filter it out before we do anything else

```
my_data_salary <- my_data %>%
  filter(txn_description == "PAY/SALARY") %>%
  select(customer_id, gender, age, amount, date, balance)
my_data_total <- my_data_salary %>%
  group_by(customer_id) %>%
  summarise(amount = sum(amount))# %>%
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
my_data_salary_total <- left_join(my_data_total, my_data_salary, by =c("customer_id" = "customer_id"))
my_data_salary_total <- my_data_salary_total[-c(5)]
```

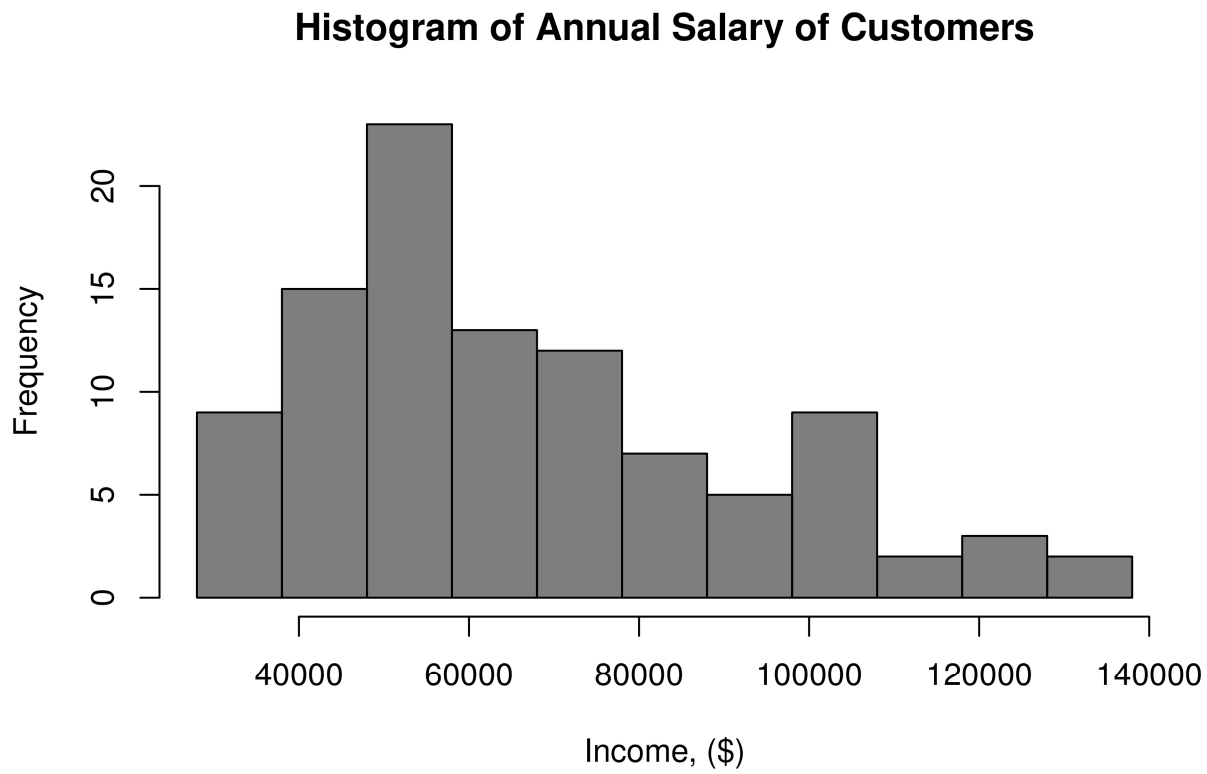
Now with salary filtered out, we can make some observations by looking at different attributes



While this graph shows total amount, it actually doesn't show how many times a customer transferred money into the bank. So let's see how we can put a trend into it.

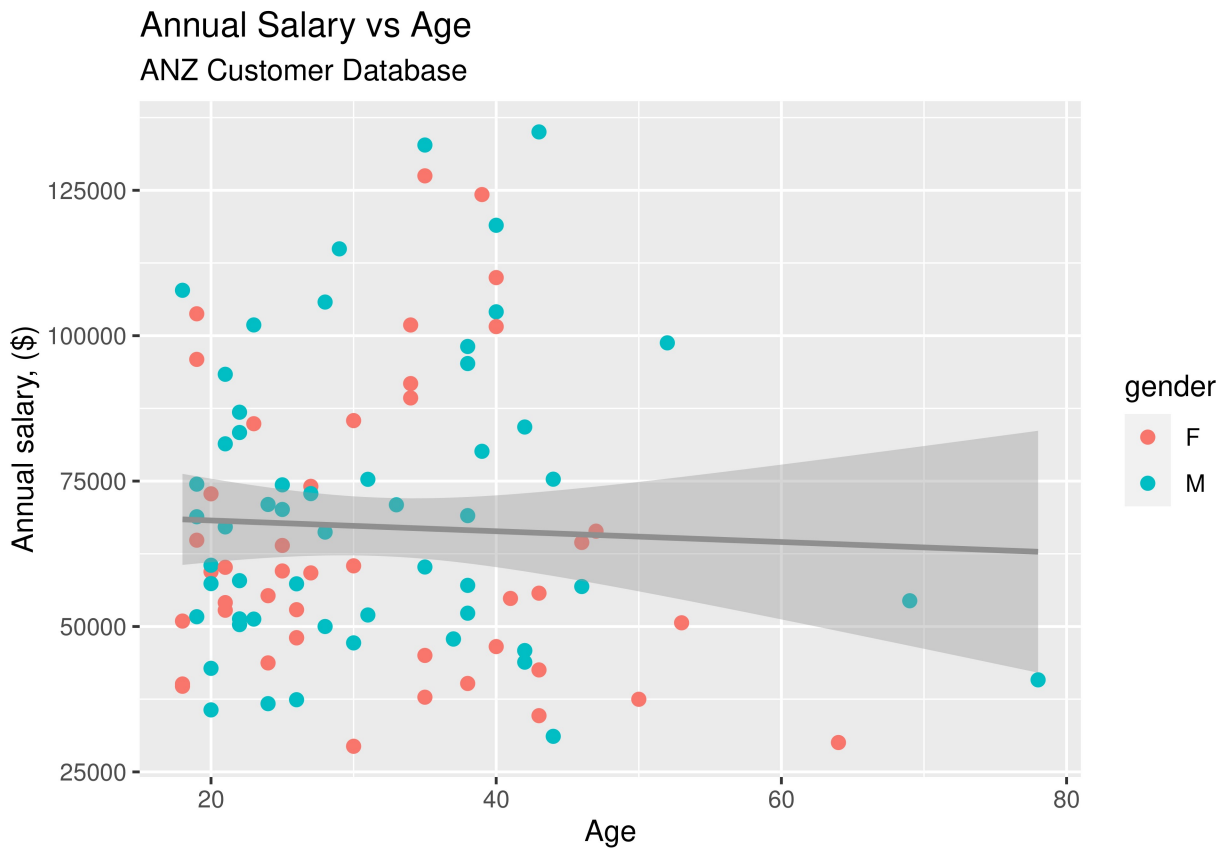
```
# Create an initial dataframe to store the results
df_customer = data.frame(customer_id = unique(my_data$customer_id))

# Create a mode function to find out what is the salary payment frequency
mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
  }

# Create a loop to process through all salary payments for each customer
for (i in seq(nrow(df_customer))) {
  transaction_data <- my_data[my_data$customer_id == df_customer$customer_id[i]
                              & my_data$txn_description ==    #find those where txn_description is Sal
                              "PAY/SALARY",c("amount","date")] %>%
    group_by(date) %>%
    summarise(amount = sum(amount))
    total_sum <- sum(transaction_data$amount)
    count = dim(transaction_data)[1]
```

```
    if (count == 0) {
    df_customer$frequency[i] = NA
    df_customer$level[i] = NA
  } else {
    s = c()
    lvl = c()
    for (j in seq(count - 1)) {
      s = c(s,(transaction_data$date[j + 1] - transaction_data$date[j]))
      lvl = c(lvl,transaction_data$amount[j])}
    lvl = c(lvl,tail(transaction_data$amount,n = 1))
    df_customer$frequency[i] = mode(s)
    df_customer$level[i] = mode(lvl)
  }
}

df_customer$annual_salary = df_customer$level / df_customer$frequency * 365.25
df_customer_table <- as.data.frame(df_customer)

# Distribution of customers' annual salary
hist(df_customer$annual_salary[!is.na(df_customer$annual_salary)],breaks = c(seq(28000,140000,by = 1000
     col = rgb(0,0,0,0.5), main = "Histogram of Annual Salary of Customers", xlab = 'Income, ($)')
```




```
# Create a dataframe to summarize customers' consumption behavior
df_attributes <- my_data %>%
  # use my_data to summarize customers' consumption behavior
```

```
  dplyr::select(customer_id, gender, age, amount, date, balance) %>%
  group_by(customer_id) %>%
  mutate(num_trans = n(),
             avg_weekly_trans = round(7*n()/length(unique(date)),0),
             max_amount = max(amount),
             num_large_trans = sum(amount > 100),
             # an arbitrary $100 benchmark is selected
             use_num_day = length(unique(date)),
             avg_trans_amount = mean(amount, na.rm = TRUE),
             med_balance = median(balance,na.rm = TRUE)) %>%
  select(-c("amount","date","balance")) %>%
  unique()

# Assign gender as binary numeric variable
df_attributes$gender_num <- ifelse(df_attributes$gender == "M",1,0)

# Assign age by groups as binary numeric variables
df_attributes$age_below20 <- ifelse(df_attributes$age < 20,1,0)
df_attributes$age_btw20n40 <- ifelse(df_attributes$age >= 20 & df_attributes$age < 40,1,0)
df_attributes$age_btw40n60 <- ifelse(df_attributes$age >= 40 & df_attributes$age < 60,1,0)

# Merge all the attributes into  a single dataframe and select relevant attributes
customer_data <- merge(df_customer, df_attributes)
# Remove columns: customer_id, frequency and level
customer_data <- customer_data[ ,c(4:17)] # 14 columns left


#### Below is an example comparing salary and age

# Scatter plot of annual salary versus age
ggplot(customer_data, aes(x = age, y = annual_salary,
                                    color = gender)) +  geom_point(size = 2) +
  stat_smooth(aes(group = 1), method = "lm", se = TRUE, position = "identity", col = "grey56") +
  labs(title = "Annual Salary vs Age",
       subtitle = "ANZ Customer Database", y = "Annual salary, ($)", x = "Age")
```



This graph is able to show the trend, which was not done previously.

Forecast

To do a forecast, we will need to split the model (where 1/5 of the data will be used to test the accuracy of the model that we choose)

```
#Using the customer_data

# First to eliminate categorical gender variable that was kept for plotting scatter plots above
customer_data <- customer_data[ ,-2] # 13 columns left
# Create train and test datasets
dt = sort(sample(nrow(customer_data), nrow(customer_data)*.8))
# select 20% of the data for validation
validation <- customer_data[-dt,]
# use the remaining 80% of data to training and testing the models
dataset <- customer_data[dt,]
```

```
# fit <- dataset %>%
#   model(TSLM(annual_salary))
```

Note: Not sure how to forecast with a data frame. I have only used forecasting with a time series.