Lab04

Vaibhav Saini - 301386847

2023-02-09

Question 1a

Create my.lm function to return slope and Y-intercept

```
#my.lm that takes a vector x and a vector y
my.lm <- function(x, y) {
  # find Beta
  b <- sum((x - mean(x)) * (y - mean(y))) / sum((x - mean(x)) ^ 2)
  a <- mean(y) - b * mean(x)
  #returns the slope a and y-intercept b
  return(list(a = a, b = b))
}
```

Question 1b

Test my.lm against lm function

```
# create vector x and y of length 3
x <- c(3, 0, 1)
y <- c(8, 4, 7)

# call my.lm
res <- my.lm(x, y)

# print slope and y-intercept decimal values for my.lm
sprintf("Slope for my.lm func: %.2f", res$a)
```

```
## [1] "Slope for my.lm func: 4.71"
```

```
sprintf("Y-intercept for my.lm func: %.2f", res$b)
```

```
## [1] "Y-intercept for my.lm func: 1.21"
```

```
model = lm(y ~ x, data = data.frame(x = x, y = y))

# print slope and y-intercept decimal values for lm
sprintf("Slope for lm func: %.2f", summary(model)$coefficients[1,1])
```

```
## [1] "Slope for lm func: 4.71"
```

```
sprintf("Y-intercept for lm func: %.2f", summary(model)$coefficients[2,1])
```

```
## [1] "Y-intercept for lm func: 1.21"
```

Question 2

Plot total revenue / month for Revenue

```
# import libraries
library(RSQLite)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

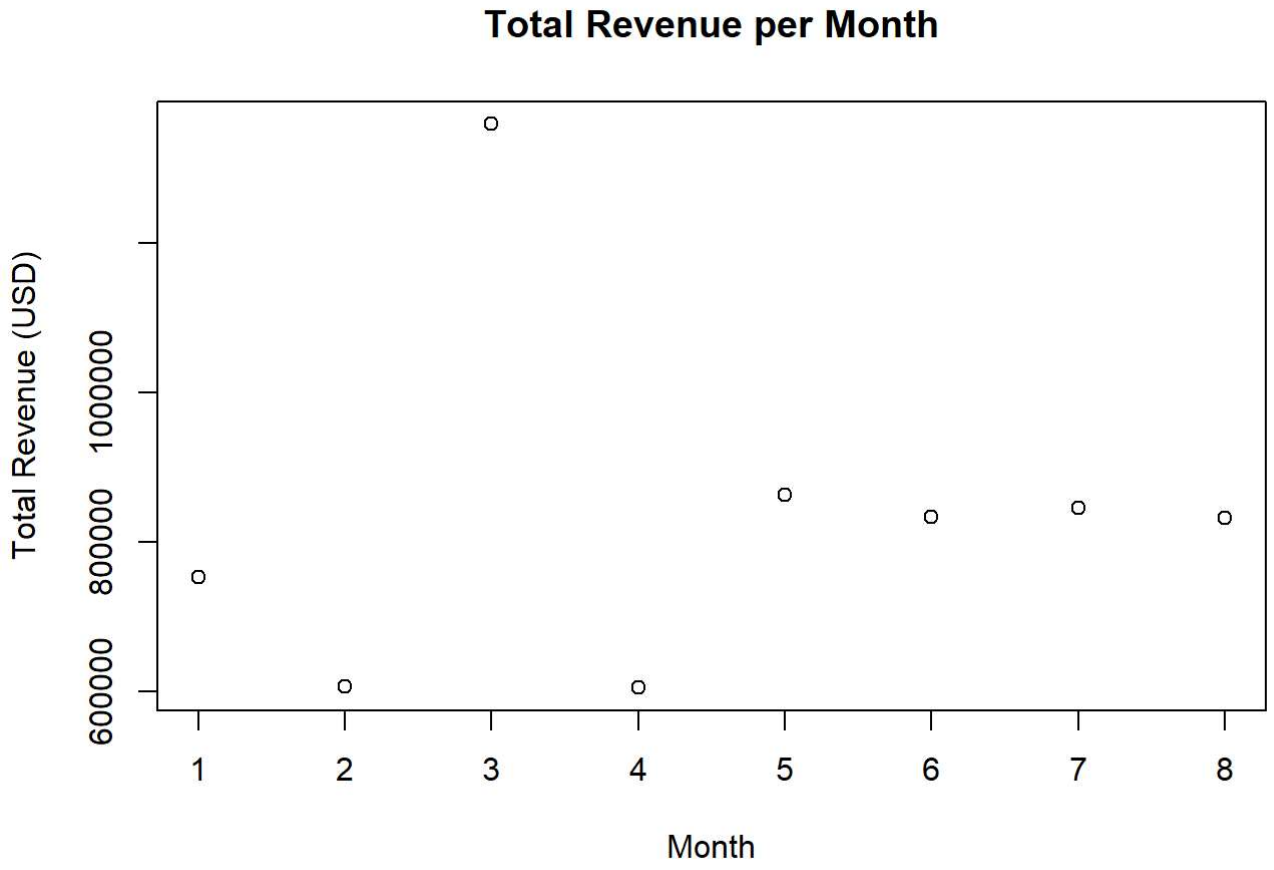
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
# create database connection
db <- dbConnect(SQLite(), dbname = "xcoretail.sqlite")

# read sales table
data_sales <- dbReadTable(db, "sales")

# read prices table
data_prices <- dbReadTable(db, "prices")

# convert InvoiceDate to date format
data_sales$InvoiceDate <- as.Date(data_sales$InvoiceDate, format = "%Y-%m-%d")

# filter data_sales between 2011-01-01 and 2011-08-31
data_sales <- data_sales %>% filter(InvoiceDate >= "2011-01-01" & InvoiceDate <= "2011-08-31")

# join data_sales and data_prices on StockCode
data <- left_join(data_sales, data_prices, by = "StockCode")

# clean up data with description "" and UnitPrice 0
data <- data %>% filter(Description != "" & UnitPrice != 0)

# calculate total price
data$total_price <- data$Quantity * data$UnitPrice

# calculate total price per day
data_day <- data %>% group_by(InvoiceDate) %>% summarise(total_price = sum(total_price))

# calculate total price per month
data_month <- data %>% group_by(month = format(InvoiceDate, "%Y-%m")) %>% summarise(total_price
 = sum(total_price))

# convert yyyy-mm to mm
data_month$month <- as.numeric(substr(data_month$month, 6, 7))

# plot total price per month
plot(data_month$month, data_month$total_price, xlab = "Month", ylab = "Total Revenue (USD)", mai
n = "Total Revenue per Month")
```



Bonus Question

The function take in 2 values (Stock Code and Target Month) and returns number of units to be purchased along with the plot

```
my.prodpredict <- function(code, month) {

  # generate a list of months 3 months before the month of interest using the lubridate package
  months <- c(substr(as.character(ym(month) - months(3)), start = 1, stop = 7), substr(as.charac
ter(ym(month) - months(2)), start = 1, stop = 7), substr(as.character(ym(month) - months(1)), st
art = 1, stop = 7))

  # create a vector of length 3 to store the total price of the product in the 3 months before t
he month of interest
  counts <- rep(NA, 3)

  # loop through the months
  for (i in 1:3) {

    # create a query to select the total price of the product in the month of interest
    query <- paste0("SELECT SUM(Quantity) FROM sales WHERE StockCode = '", code, "' AND InvoiceD
ate LIKE '", months[i], "%'")

    # send the query to the database
    request <- dbSendQuery(db, query)

    # fetch the result
    counts[i] <- strtoi(dbFetch(request))

    # clear the result
    dbClearResult(request)
  }

  # get the slope and y-intercept of the line of best fit
  res <- my.lm(c(1, 2, 3), counts)

  # calculate the total units product in the month of interest
  counts[4] <- res$a + res$b * 4

  # plot the total units product in the 3 months before the month of interest
  plot(c(1, 2, 3), counts[-4], main = paste0("Prediction for Product ", code, ": ", month, " is
 ", counts[4] ," units"), ylab = "Units", xlab = "Month", pch = 19, xlim = c(0, 5), ylim = c(cou
nts[1], counts[4]))

  # plot the total units product in the month of interest
  points(4, counts[4], pch = 3)

  # plot the line for the Linear regression
  abline(a = res$a, b = res$b)

  return(counts[4])

}
```

Using predictions

Usage Instructions:

Call function directly with Stock code and target month as parameters

function will return the total amount of units and plot a graph for the prediction

Understanding the Graph plot

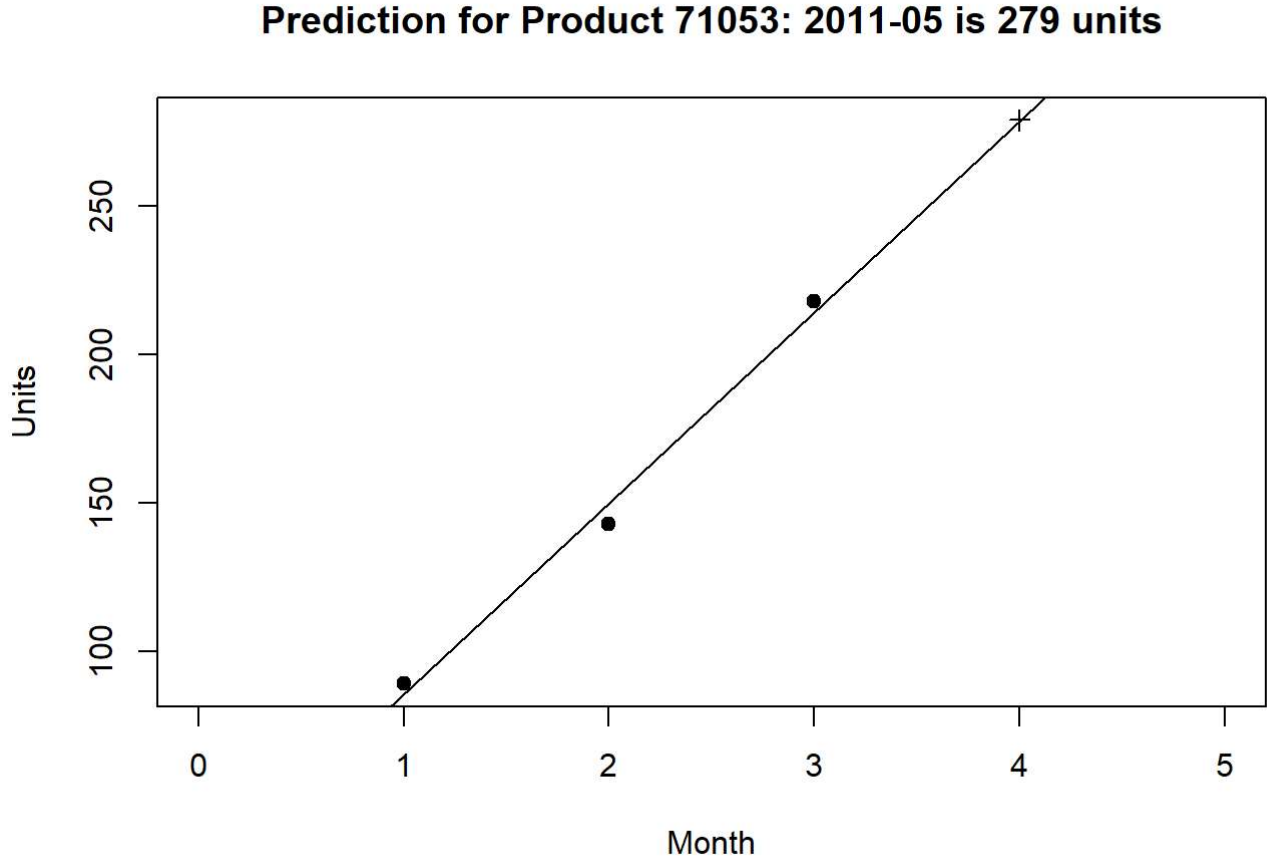
The y-axis represent the number of stock units, whereas x-axis represent months series

Each dot on the plot is the number of stock units purchased in a given month with ‘+’ point being the number of stock units that should be purchased for the target month

The line y = a + bX is the linear regression the months

Does your exploration suggest that rolling out this procedure on all product codes may be indicated?

While the prediction model seems work perfectly for the given example, it might not always be able to predict accurately in the case of an volalite stock given we are only insterested in the linear regression over past 3 months.




```
## [1] "Total units of Stock 71053 for the month 2011-05 that should be purchased are : 279.00"
```