ProjetR

Introduction :

Nous avons sélectionné le jeu de données “Superket sales” [1] pour ce projet afin de procéder à l’analyse, à la visualisation et à la prédiction des données.

Informations sur les attributs :

- Invoice id : Numéro d’identification de la facture de vente générée par ordinateur
- Branch: Succursale du supercentre (3 succursales disponibles identifiées par A, B et C)
- City: Emplacement des supercentres
- Customer type : Type de clients, enregistré par Membres pour les clients utilisant la carte de membre et Normal pour ceux sans carte de membre
- Gender : Genre du client
- Product line : Groupes de catégorisation générale des articles - Accessoires électroniques, Accessoires de mode, Aliments et boissons, Santé et beauté, Maison et style de vie, Sports et voyages
- Unit price : Prix de chaque produit en $
- Quantity : Nombre de produits achetés par le client
- Tax: Frais de taxe de 5 % pour les clients qui achètent
- Total : Prix total incluant la taxe
- Date : Date d’achat (Enregistrée de janvier 2019 à mars 2019)
- Time: Heure d’achat (de 10h à 21h)
- Payment: Mode de paiement utilisé par le client pour l’achat (3 méthodes disponibles - Espèces, Carte de crédit et Ewallet)
- COGS : Coût des marchandises vendues
- Gross margin percentage : Pourcentage de marge brute
- Gross income : Revenu brut
- Rating: Note de stratification du client sur son expérience d’achat globale (Sur une échelle de 1 à 10)

Chargement des packages:

Avant d’entamer l’analyse, nous chargerons les bibliothèques nécessaires :

```r
# Chargement des packages
library(readr)
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
```

```r
library(ggplot2)
library(lubridate)
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

Collecte des Données :

```r
library(readr)
supermarket_sales <- read_csv("C:/Users/Sabrine/Downloads/archive/supermarket_sales - Sheet1.c
                              col_types = cols(
                                `Invoice ID` = col_character(),
                                Branch = col_character(),
                                City = col_character(),
                                `Customer type` = col_character(),
                                Gender = col_character(),
                                `Product line` = col_character(),
                                Date = col_character(),
                                Payment = col_character(),
                                `Unit price` = col_double(),
                                Quantity = col_double(),
                                `Tax 5%` = col_double(),
                                Total = col_double(),
                                cogs = col_double(),
                                `gross margin percentage` = col_double(),
                                `gross income` = col_double(),
                                Rating = col_double(),
                                Time = col_time(format = "")
                              )
)
View(supermarket_sales)
```

Nettoyage des Données :

Après l'affichage des premières lignes des données, on constate que que ces dernières sont soigneusement nettoyées et prêtes à être utilisées. Voici la commande utilisée pour afficher les premières lignes du jeu de données `supermarket_sales` :

```
head(supermarket_sales)
```

```
# A tibble: 6 x 17
  `Invoice ID` Branch City    `Customer type` Gender `Product line` `Unit price`
  <chr>        <chr>  <chr>   <chr>           <chr>  <chr>                 <dbl>
1 750-67-8428  A      Yangon  Member          Female Health and be~         74.7
2 226-31-3081  C      Naypyi~ Normal          Female Electronic ac~         15.3
3 631-41-3108  A      Yangon  Normal          Male   Home and life~         46.3
4 123-19-1176  A      Yangon  Member          Male   Health and be~         58.2
5 373-73-7910  A      Yangon  Normal          Male   Sports and tr~         86.3
6 699-14-3026  C      Naypyi~ Normal          Male   Electronic ac~         85.4
# i 10 more variables: Quantity <dbl>, `Tax 5%` <dbl>, Total <dbl>, Date <chr>,
#   Time <time>, Payment <chr>, cogs <dbl>, `gross margin percentage` <dbl>,
#   `gross income` <dbl>, Rating <dbl>
```

les dimensions des données sont (1000 lignes et 18 colonnes)

```
dim(supermarket_sales)
```

```
[1] 1000   17
```

Résumé statistique des donnée

```
summary(supermarket_sales)
```

```
  Invoice ID           Branch             City           Customer type
 Length:1000        Length:1000        Length:1000        Length:1000
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character


    Gender          Product line         Unit price       Quantity
 Length:1000        Length:1000        Min.   :10.08   Min.   : 1.00
 Class :character   Class :character   1st Qu.:32.88   1st Qu.: 3.00
 Mode  :character   Mode  :character   Median :55.23   Median : 5.00
                                       Mean   :55.67   Mean   : 5.51
                                       3rd Qu.:77.94   3rd Qu.: 8.00
                                       Max.   :99.96   Max.   :10.00
     Tax 5%              Total              Date               Time
 Min.   : 0.5085   Min.   :  10.68   Length:1000        Length:1000
 1st Qu.: 5.9249   1st Qu.: 124.42   Class :character   Class1:hms
 Median :12.0880   Median : 253.85   Mode  :character   Class2:difftime
 Mean   :15.3794   Mean   : 322.97                      Mode  :numeric
 3rd Qu.:22.4453   3rd Qu.: 471.35
 Max.   :49.6500   Max.   :1042.65
```

```
   Payment               cogs        gross margin percentage  gross income
 Length:1000        Min.   : 10.17   Min.   :4.762           Min.   : 0.5085
 Class :character   1st Qu.:118.50   1st Qu.:4.762           1st Qu.: 5.9249
 Mode  :character   Median :241.76   Median :4.762           Median :12.0880
                    Mean   :307.59   Mean   :4.762           Mean   :15.3794
                    3rd Qu.:448.90   3rd Qu.:4.762           3rd Qu.:22.4453
                    Max.   :993.00   Max.   :4.762           Max.   :49.6500
     Rating
 Min.   : 4.000
 1st Qu.: 5.500
 Median : 7.000
 Mean   : 6.973
 3rd Qu.: 8.500
 Max.   :10.000
```

Pour gérer les valeurs manquantes, nous allons vérifier les colonnes des ensembles de données. Si nous trouvons des données manquantes dans les colonnes, cela génère des valeurs NA en sortie, ce qui peut ne pas être bon pour chaque modèle. Nous allons donc vérifier cela en utilisant la méthode `mean()` pour chaque colonne numérique.

```
mean(supermarket_sales$`Unit price`)
```

```
[1] 55.67213
```

```
mean(supermarket_sales$Quantity)
```

```
[1] 5.51
```

```
mean(supermarket_sales$`Tax 5%`)
```

```
[1] 15.37937
```

```
mean(supermarket_sales$Total)
```

```
[1] 322.9667
```

```
mean(supermarket_sales$cogs)
```

```
[1] 307.5874
```

```
mean(supermarket_sales$`gross margin percentage`)
```

```
[1] 4.761905
```

```
mean(supermarket_sales$`gross income`)
```

```
[1] 15.37937
```

```
mean(supermarket_sales$Rating)
```

```
[1] 6.9727
```

=> On constate qu'il n' y a pas de dans les colonnes numériques

On Vérifie les valeurs uniques dans une colonne spécifique pour traiter les cas ou les données sont écrites dans différent format:

```r
unique(supermarket_sales$City)
```

```
[1] "Yangon"    "Naypyitaw" "Mandalay"
```

```r
unique(supermarket_sales$Branch)
```

```
[1] "A" "C" "B"
```

```r
unique(supermarket_sales$Gender)
```

```
[1] "Female" "Male"
```

```r
unique(supermarket_sales$`Customer type`)
```

```
[1] "Member" "Normal"
```

```r
unique(supermarket_sales$`Product line`)
```

```
[1] "Health and beauty"      "Electronic accessories" "Home and lifestyle"
[4] "Sports and travel"      "Food and beverages"     "Fashion accessories"
```

```r
unique(supermarket_sales$Payment)
```

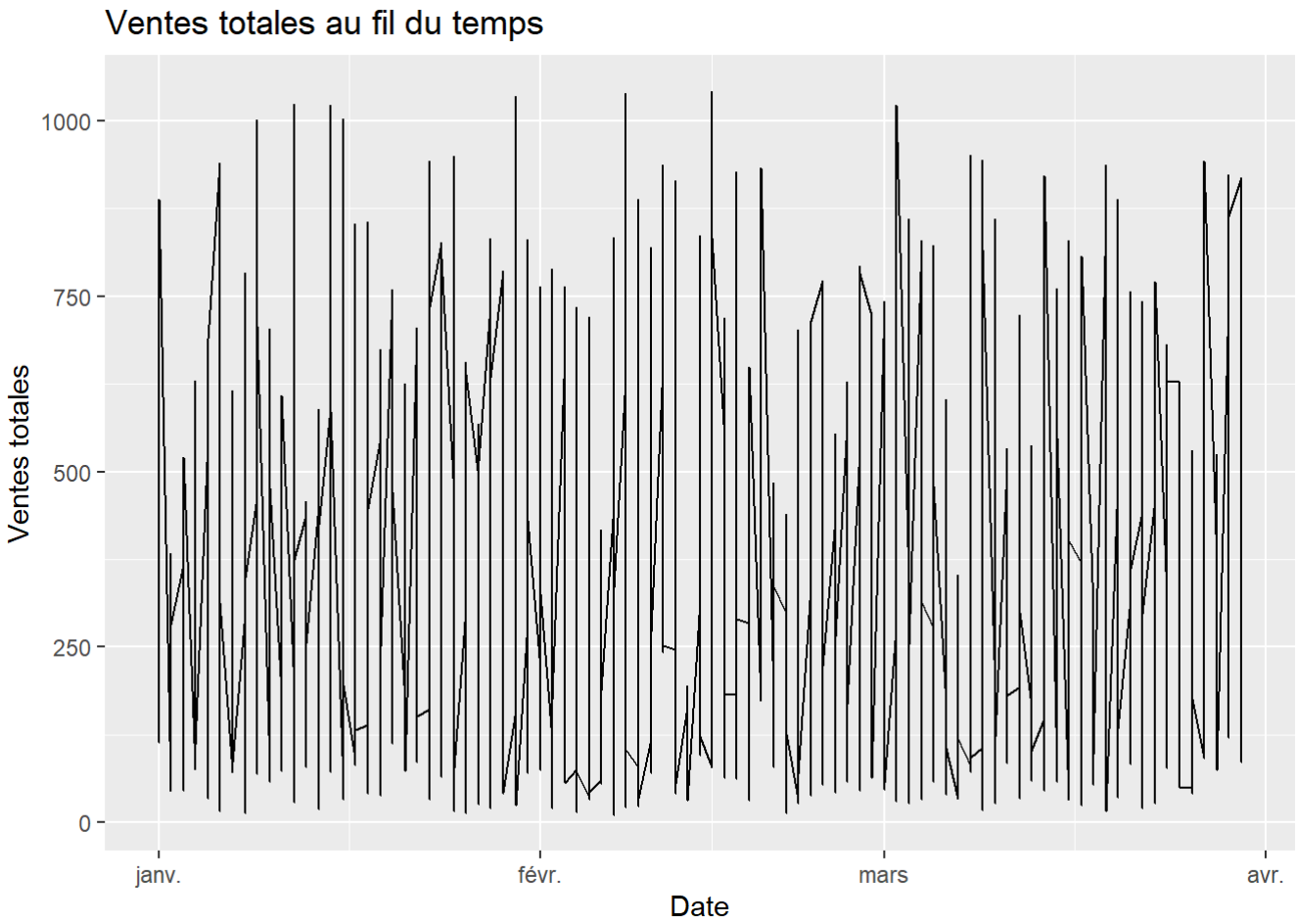
```
[1] "Ewallet"     "Cash"        "Credit card"
```

Analyse des Données :

Analyses de séries chronologiques :

1. Visualisation de la série chronologique des ventes totales au fil du temps :

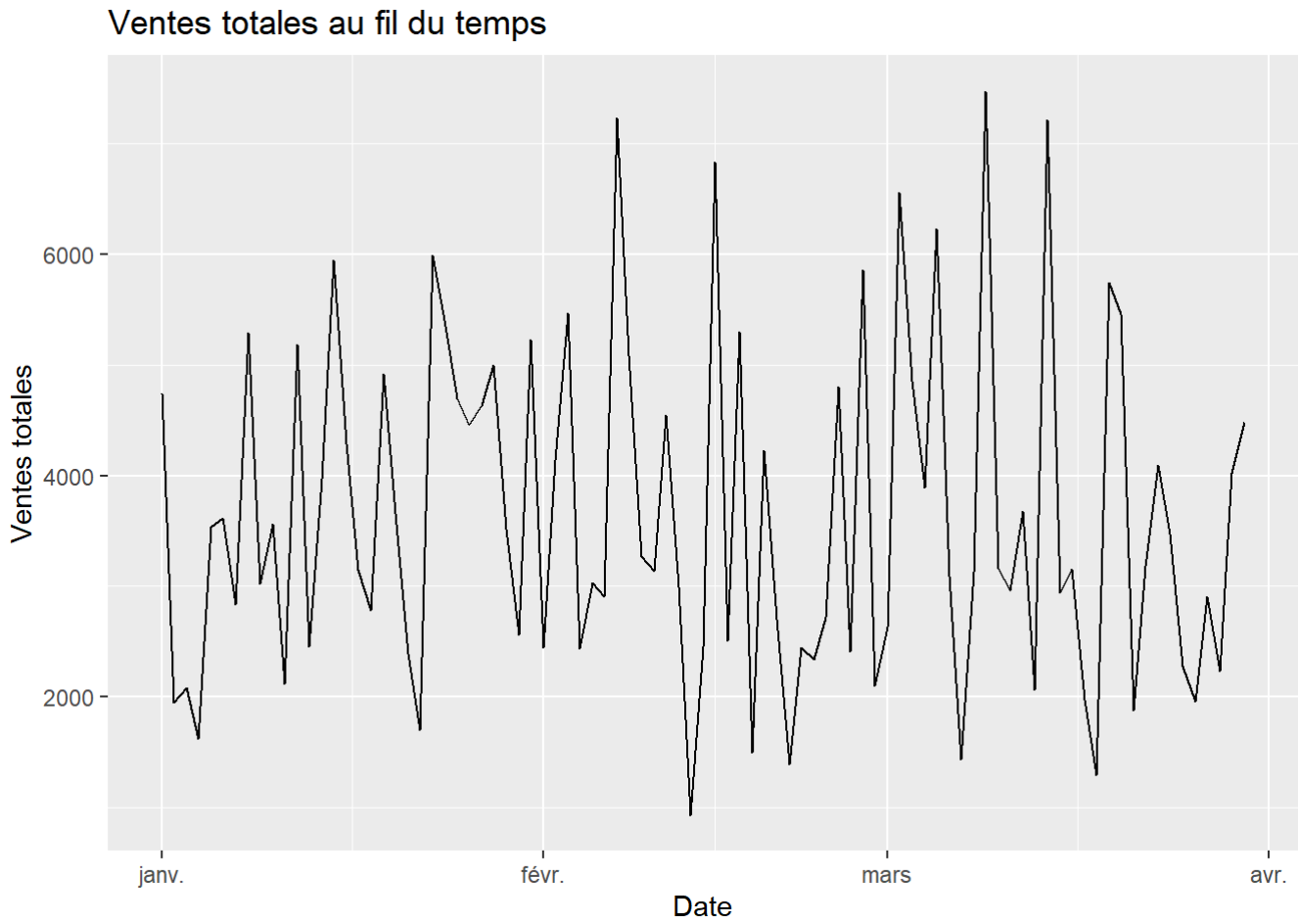
```r
library(ggplot2)

# Convertir la colonne Date en format Date
supermarket_sales$Date <- as.Date(supermarket_sales$Date, format = "%m/%d/%Y")

# Créer un graphique de la série chronologique des ventes totales
ggplot(supermarket_sales, aes(x = Date, y = Total)) +
  geom_line() +
  labs(title = "Ventes totales au fil du temps", x = "Date", y = "Ventes totales")
```



Analyse statistique

a. Analyse de série temporelle

```r
# Convertir la colonne 'Date' en format Date
supermarket_sales$Date <- as.Date(supermarket_sales$Date, format = "%m/%d/%Y")

# Aggréger les ventes par date
sales_time_series <- supermarket_sales %>%
  group_by(Date) %>%
  summarise(Total_Sales = sum(Total))

# Afficher la série temporelle
ggplot(sales_time_series, aes(x = Date, y = Total_Sales)) +
  geom_line() +
  labs(title = "Ventes totales au fil du temps",
       x = "Date",
       y = "Ventes totales")
```



=>La série temporelle montre une tendance générale des ventes au fil du temps.

b. Corrélations entre différentes variables économiques

```
# Calculer les corrélations
numerics <- select(supermarket_sales, -c(`Invoice ID`, Branch, City, Payment, Time,`Customer t
correlation_matrix <- cor(numerics)
```

```
Warning in cor(numerics): the standard deviation is zero
```

```
# Calculate standard deviation for each column
standard_deviations <- sapply(numerics, sd)

# Exclude variables with zero standard deviation
non_zero_std_variables <- numerics[, standard_deviations != 0]

# Calculate correlations
correlation_matrix <- cor(non_zero_std_variables)

# Display correlation matrix
print(correlation_matrix)
```

```
             Unit price    Quantity     Tax 5%      Total       cogs
Unit price   1.000000000 0.01077756 0.6339621 0.6339621 0.6339621
Quantity     0.010777564 1.00000000 0.7055102 0.7055102 0.7055102
Tax 5%       0.633962089 0.70551019 1.0000000 1.0000000 1.0000000
```

```
Total         0.633962089  0.70551019  1.0000000  1.0000000  1.0000000
cogs          0.633962089  0.70551019  1.0000000  1.0000000  1.0000000
gross income  0.633962089  0.70551019  1.0000000  1.0000000  1.0000000
Rating       -0.008777507 -0.01581490 -0.0364417 -0.0364417 -0.0364417
             gross income       Rating
Unit price      0.6339621 -0.008777507
Quantity        0.7055102 -0.015814905
Tax 5%          1.0000000 -0.036441705
Total           1.0000000 -0.036441705
cogs            1.0000000 -0.036441705
gross income    1.0000000 -0.036441705
Rating         -0.0364417  1.000000000
```

=>La matrice de corrélation révéle des relations entre différentes variables économiques

c. Modèle économétrique (exemple: régression linéaire)

```r
# Exemple de modèle économétrique simple (régression linéaire)
model <- lm(Total ~ Quantity + `Unit price` + `Tax 5%` + Rating, data = supermarket_sales)

# Afficher le résumé du modèle
summary(model)
```

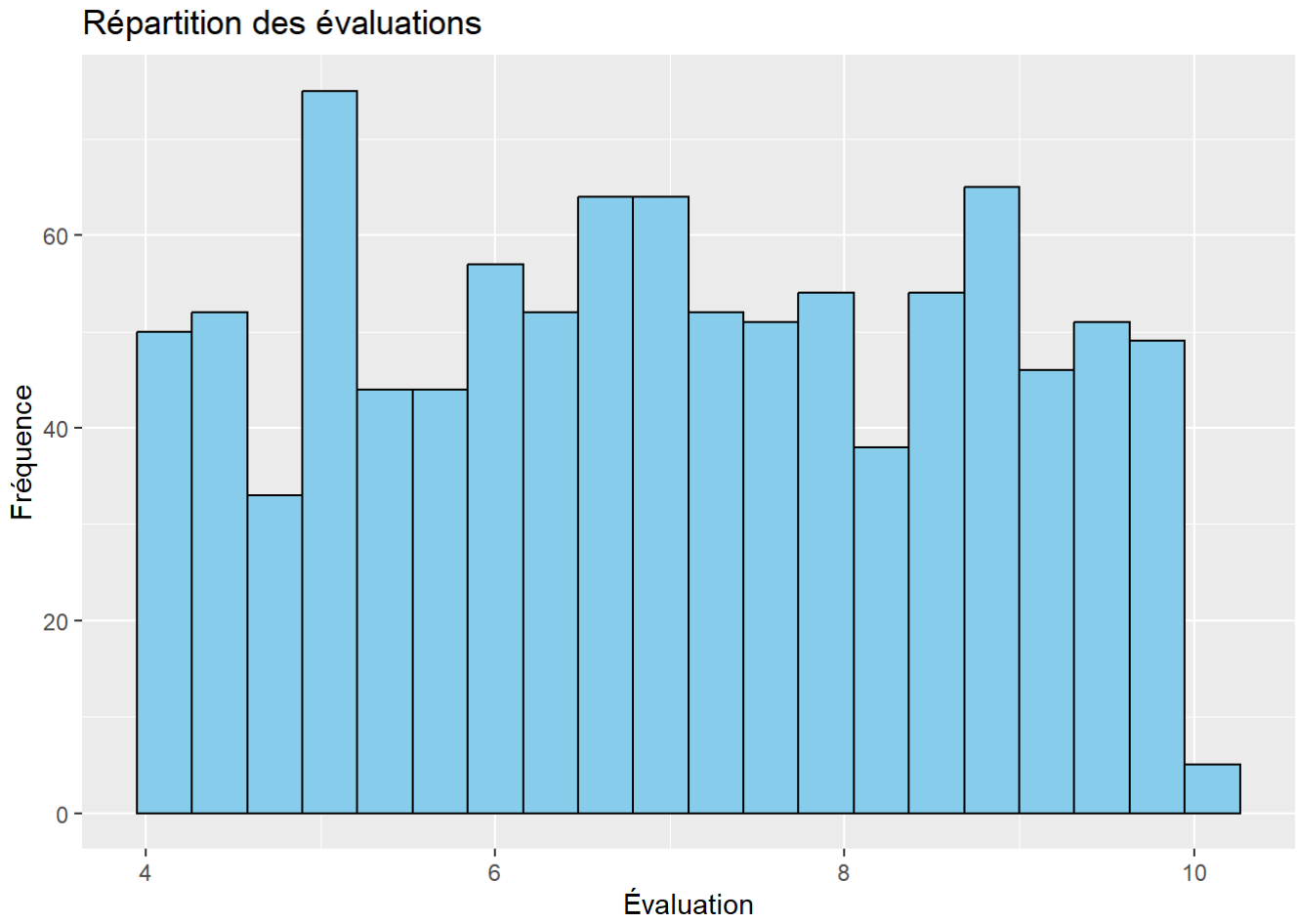
```
Call:
lm(formula = Total ~ Quantity + `Unit price` + `Tax 5%` + Rating,
    data = supermarket_sales)

Residuals:
       Min         1Q     Median         3Q        Max
-2.352e-12 -6.860e-14 -2.680e-14  6.900e-15  2.900e-11

Coefficients:
               Estimate Std. Error    t value Pr(>|t|)
(Intercept)   8.274e-13  1.878e-13  4.406e+00 1.17e-05 ***
Quantity      5.916e-14  2.380e-14  2.486e+00   0.0131 *
`Unit price` -1.086e-15  2.407e-15 -4.510e-01   0.6519
`Tax 5%`      2.100e+01  7.689e-15  2.731e+15  < 2e-16 ***
Rating        1.922e-14  1.737e-14  1.107e+00   0.2686
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.416e-13 on 995 degrees of freedom
Multiple R-squared:      1, Adjusted R-squared:      1
F-statistic: 1.703e+31 on 4 and 995 DF,  p-value: < 2.2e-16
```

=>Le modèle économétrique peut être utilisé pour prédire les ventes en fonction d'autres variables

Visualisation des données

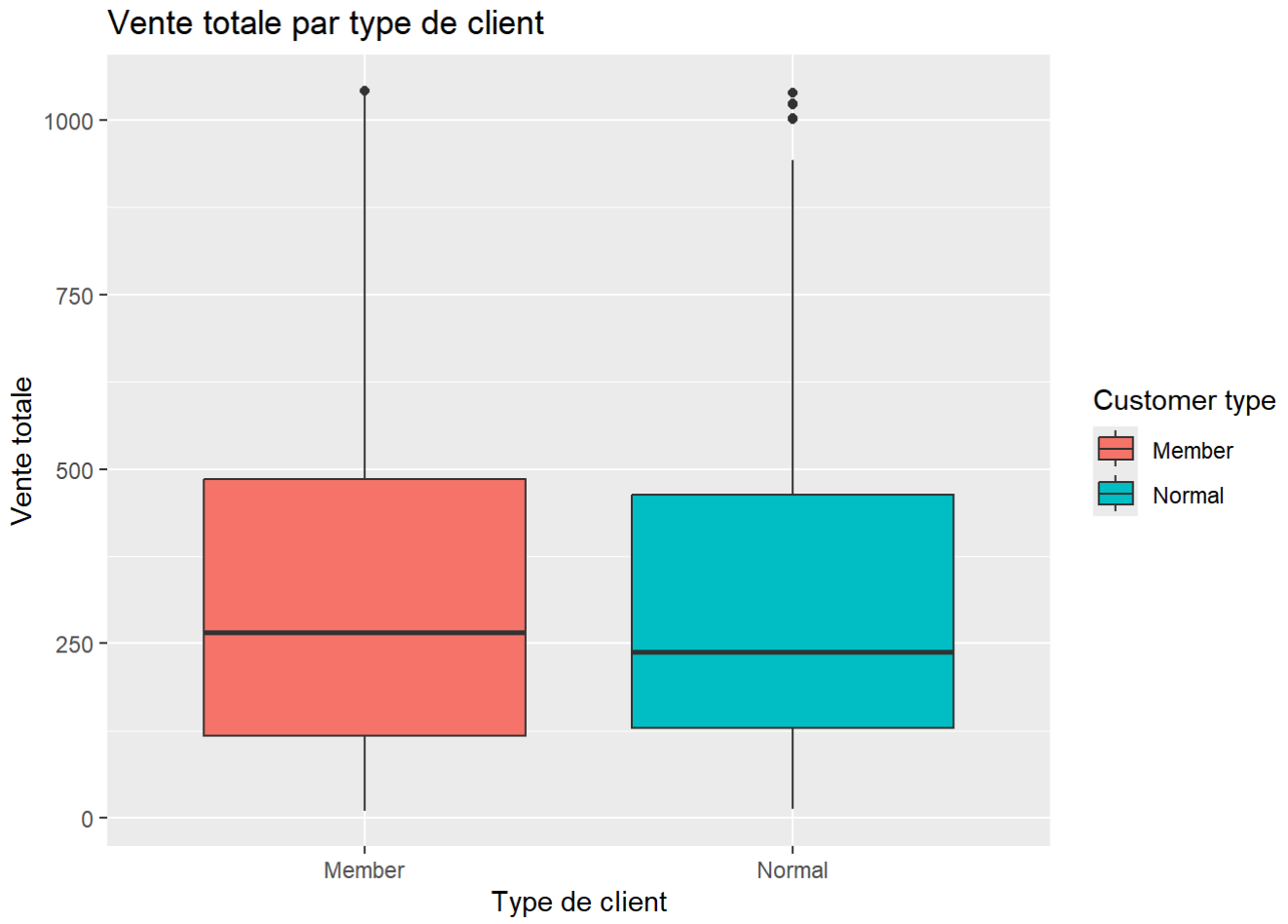
a. Histogramme des évaluations (Ratings)

```r
ggplot(supermarket_sales, aes(x = Rating)) +
  geom_histogram(fill = "skyblue", color = "black", bins = 20) +
  labs(title = "Répartition des évaluations",
       x = "Évaluation",
       y = "Fréquence")
```



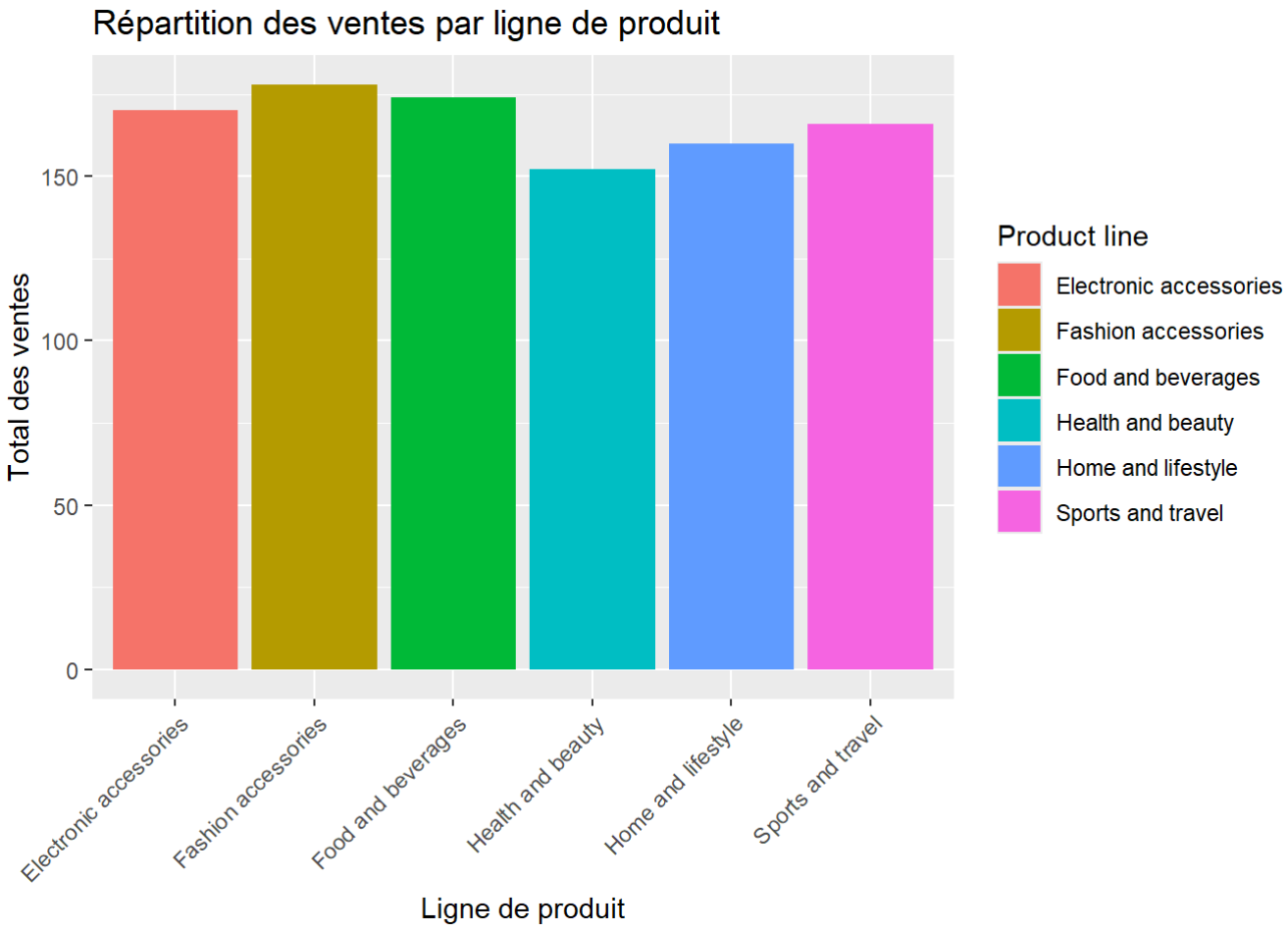
b. Boxplot de la vente totale par type de client

```r
ggplot(supermarket_sales, aes(x = `Customer type`, y = Total, fill = `Customer type`)) +
  geom_boxplot() +
  labs(title = "Vente totale par type de client",
       x = "Type de client",
       y = "Vente totale")
```



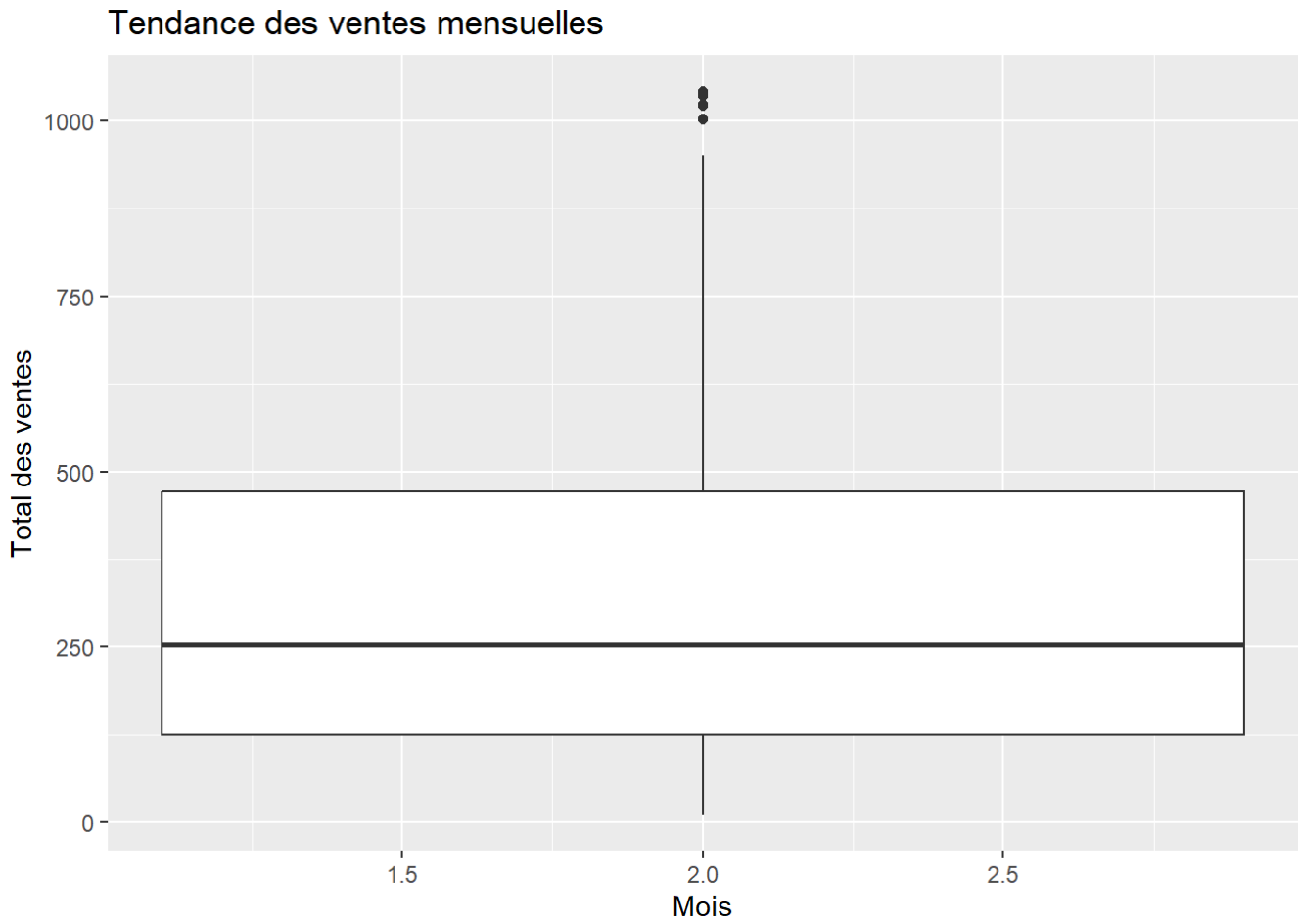
c. Diagramme à barres pour montrer la répartition des ventes par ligne de produit

```
ggplot(supermarket_sales, aes(x = `Product line`, fill = `Product line`)) +
  geom_bar() +
  labs(title = "Répartition des ventes par ligne de produit", x = "Ligne de produit", y = "Tot
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
library(lubridate)

# Extraire le mois de la colonne Date
supermarket_sales$Month <- month(supermarket_sales$Date)

# Créer un graphique de la tendance des ventes mensuelles
ggplot(supermarket_sales, aes(x = Month, y = Total)) +
  geom_boxplot() +
  labs(title = "Tendance des ventes mensuelles", x = "Mois", y = "Total des ventes")
```

```
Warning: Continuous x aesthetic
i did you forget `aes(group = ...)`?
```



Prédiction

```r
# Sélectionner les variables pour la prédiction (par exemple, 'Total' comme variable cible et
prediction_data <- select(supermarket_sales, Total, Quantity, `Unit price`, Rating)

# Définir la taille de l'ensemble d'entraînement
train_size <- 0.8 * nrow(prediction_data)

# Créer un index aléatoire pour l'ensemble d'entraînement
set.seed(123) # Pour la reproductibilité
train_index <- sample(seq_len(nrow(prediction_data)), size = train_size)

# Séparer les données en ensembles d'entraînement et de test
train_data <- prediction_data[train_index, ]
test_data <- prediction_data[-train_index, ]

# Créer le modèle de régression linéaire
lm_model <- lm(Total ~ ., data = train_data)

# Faire des prédictions sur l'ensemble de test
predictions <- predict(lm_model, newdata = test_data)

# Évaluer les performances du modèle (par exemple, erreur quadratique moyenne)
mse <- mean((test_data$Total - predictions)^2)
```

```
rmse <- sqrt(mse)
print(paste("Root Mean Squared Error (RMSE):", rmse))
```

```
[1] "Root Mean Squared Error (RMSE): 85.8606036390738"
```

=>85 est proche à 100 donc le modèle est bon

Sitographie :

[1]"Supermarket sales" sur *https://www.kaggle.com/datasets/aungpyaeap/supermarket-sales* (consulté le 20/3/2024)