Module 7 - Exemples d'applications des sciences des données

Exercices - Solutions

Exercice 1 :

Nous allons d'abord charger le fichier "Wiki-Vote.txt" dans R, pour ce faire, nous procéderons comme suit :

```
# Chargeons d'abord la librairie igraph
library(igraph)

# Lecture du fichier 1912.edges
MyGraphData <- read.graph(file = "Wiki-Vote.txt", directed = TRUE)
```

Ici, le paramètre "directed = TRUE" signifie que le graphe est dirigé, contrairement aux données de Facebook par exemple.

Une fois que le fichier est chargé, nous pouvons maintenant effectuer des opérations sur le graphe. Par exemple, pour calculer le nombre de noeuds et d'arcs dans le graphe, nous utilisons les fonctions suivantes :

```
# Nombre de noeuds dans le graphe
> vcount(MyGraphData)
[1] 8298

# Nombre d'arcs dans le graphe
> ecount(MyGraphData)
[1] 103689
```

Pour calculer la transitivité du graphe, nous utilisons la fonction suivante :

```
# Transitivité du graphe: peut être interprétée comme la probabilité que les noeuds
    adjacents à un noeud donné soient connectés
> transitivity(MyGraphData, type = "localaverage",isolates = "zero")
[1] 0.1208108
```

Une transitivité de 0.12, veut dire que la probabilité que les noeuds adjacents à un noeud donné soient connectés est très faible.

Pour calculer la longueur moyenne des chemins entre deux noeuds dans le graphe, nous pouvons utiliser la fonction suivante :

```
# Longueur moyenne des chemins entre deux noeuds dans le graphe
```

```
> average.path.length(MyGraphData)
[1] 3.341011
```

De la même façon, pour calculer la centralité moyenne normalisée des noeuds dans le graphe, nous allons utiliser la fonction suivante :

```
# Centralité moyenne des noeuds dans le graphe
centralite_moyenne <- betweenness(MyGraphData)

# Normalisation
centralite_moyenne_normalisee <- (centralite_moyenne - min(centralite_moyenne))/(
    max(centralite_moyenne) - min(centralite_moyenne))

# La valeur moyenne
> mean(centralite_moyenne_normalisee)
[1] 0.003772477
```

La visualisation des valeurs de centralités moyennes normalisées peut fait comme suite :

```
# Visualisation
plot(Ci_norm, xlab="Noeuds", ylab="Centralité moyenne normalisée", col ="red")
```

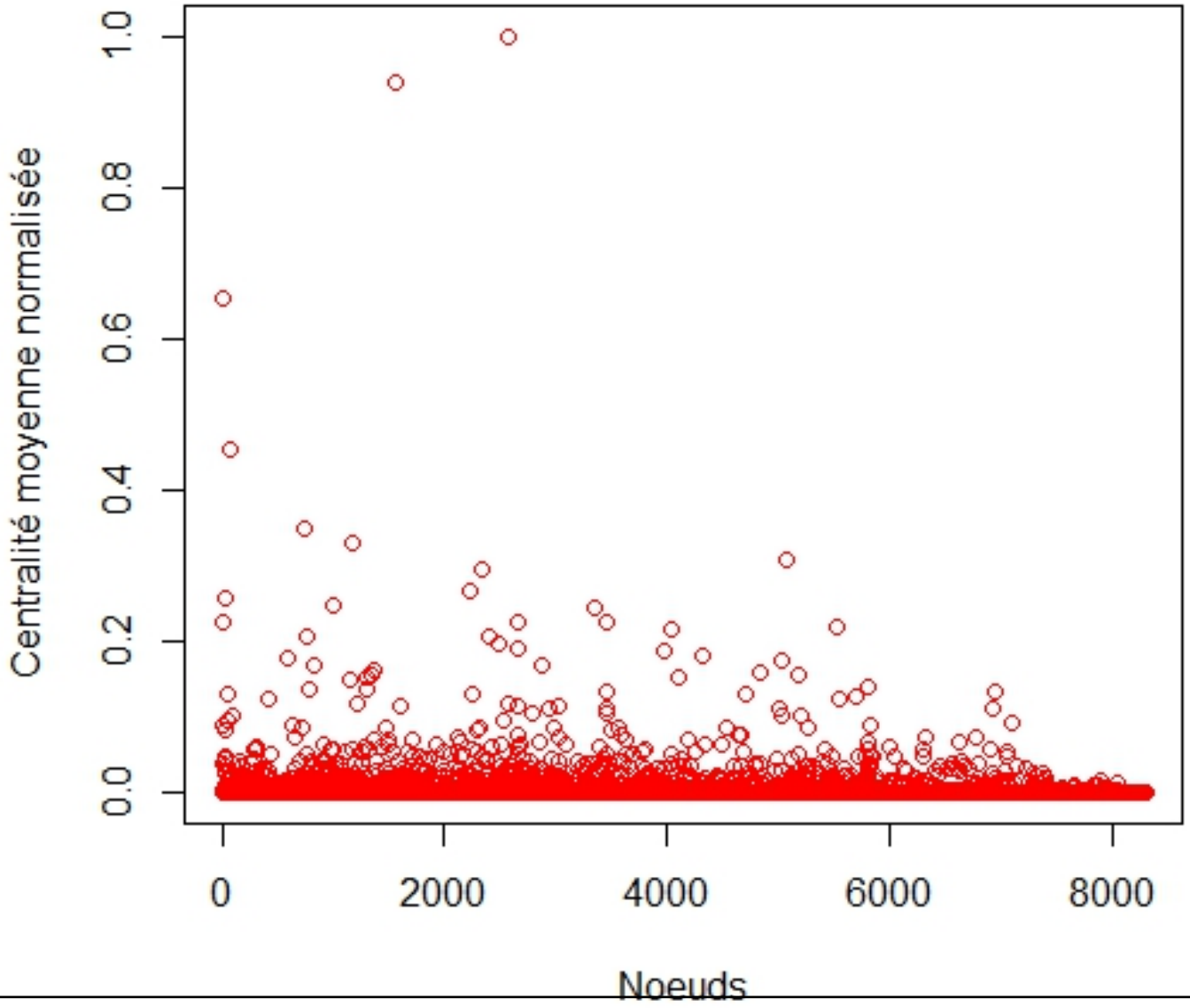

FIGURE 1 – Centralités moyennes normalisées des noeuds dans le graphe.

Pour identifier les noeuds ayant les plus grandes valeurs de degré entrant, nous allons procéder comme suit :

```
#  Noeuds ayant les plus grandes valeurs de degré entrant
# Afficher d'abord un résumé sur les statistiques des degrés des noeuds dans le
    graphe
> summary(degree(MyGraphData, mode = "in")) # ici le mode = "in" signifie les arcs
    entrants pour un noeud.
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0     0.0    12.5     6.0   457.0
# Le résultat indique qu'en moyenne, nous avons une valeur de 12 pour le degré
    entrant, et une valeur maximale de 457. Nous allons donc choisir les noeuds
    ayant un degré entrant supérieur à 300.
> V(MyGraphData)[degree(MyGraphData, mode = "in") > 300]$label
[1] "16"   "1298" "2399" "2626" "4038"  # Nous avons donc 5 noeuds.
```

Pour visualiser par exemple les noeuds ayant un degré entrant plus grand, nous allons isoler tous les autres noeuds ayant un degré entrant inférieur à 200 par exemple, pour avoir un petit graphe, de la façon suivante :

```
#  Isoler d'abord les noeuds
noeuds_isoles<- V(MyGraphData)[degree(MyGraphData, mode = "in") < 200]

# Créer le nouveau graphe en supprimant les noeuds isolés
nouveau_graphe <- delete.vertices(MyGraphData, noeuds_isoles)

# Visualiser le nouveau graphe
plot(nouveau_graphe) # Voir figure en bas
```

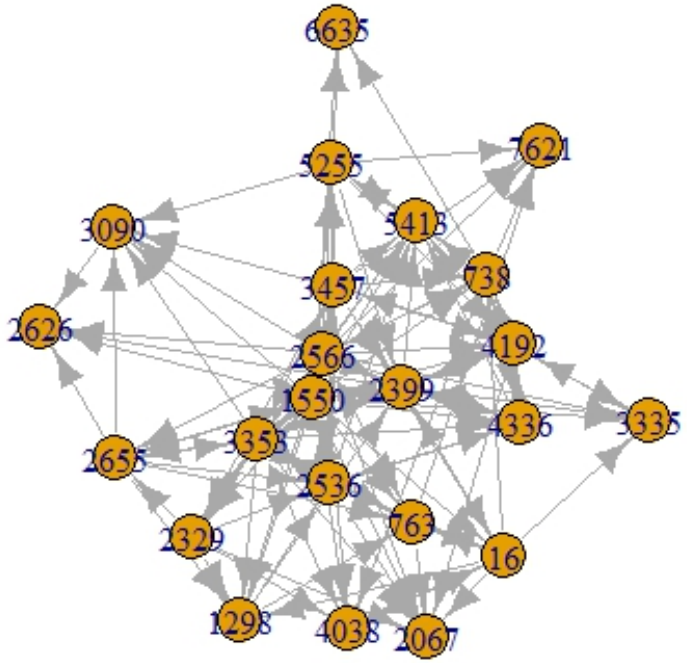

FIGURE 2 – Graphe des noeuds ayant plus grand degré entrant.

Exercice 2 :

Pour commencer, nous allons d'abord charger le fichier "data.csv" dans R.

```
# Chargeons d'abord la librairie dplyr
library(dplyr)

# Lecture du fichier 1912.edges
df <- read.csv(file="data.csv", sep = ",")

# Examiner les colonnes dans cet ensemble de données
> glimpse(df)
Observations: 541,909
Variables: 8
$ InvoiceNo   <fctr> 536365, 536365, 536365, 536365, 536365, 536365, 536365...
$ StockCode   <fctr> 85123A, 71053, 84406B, 84029G, 84029E, 22752, 21730, 2...
$ Description <fctr> WHITE HANGING HEART T-LIGHT HOLDER, WHITE METAL LANTER...
$ Quantity    <int> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2, 3, ...
$ InvoiceDate <fctr> 12/1/2010 8:26, 12/1/2010 8:26, 12/1/2010 8:26, 12/1/2...
$ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1.85, 1...
$ CustomerID  <int> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17850,...
$ Country     <fctr> United Kingdom, United Kingdom, United Kingdom, United...
```

Nous allons ensuite nettoyer les données, par exemple ne pas garder des quantités ou des montants inférieurs à 0. Pour vérifier l'existence de ces erreurs dans les données, nous utilisons la fonction suivante :

```
# Vérifier l'existence des valeurs négatives
> any(df$Quantity < 0)
[1] TRUE

# Même chose pour les montants
> any(df$UnitPrice < 0)
[1] TRUE
```

Dans les deux cas, nous avons des réponses positives qui indiquent l'existence des valeurs négatives dans les deux colonnes. Nous devons donc remplacer ces valeurs par des "NA" par exemple, ou supprimer complètement les lignes contenant ces valeurs. Dans cet exercice, nous préférons faire les deux, c'est-à-dire, remplacer les valeurs négatives par des NA, puis les supprimer complètement. Pour ce faire, nous procéderons comme suit :

```
# Remplacer les valeurs négatives par des NA
df <- df %>%
  mutate(Quantity = replace(df$Quantity, df$Quantity <= 0, NA),
         UnitPrice = replace(df$UnitPrice, df$UnitPrice <= 0, NA))
```

Une fois que le changement est fait, nous pouvons maintenant utiliser la fonction drop_na() de la librairie **tidyr** pour supprimer complètement les lignes avec NA. Pour ce faire, nous procédons comme suit :

```
# Supprimer les lignes avec des NA
df <- df %>% drop_na()
```

Nous pouvons vérifier si les lignes contenants des NA ont été supprimées en utilisant encore la fonction any() comme suit :

```
# Vérifier l'existence des NA
> any(df$Quantity < 0)
[1] FALSE
```

Avant de d'appliquer la méthode RFM, il est important de convertir le format des colonnes en facteur pour faciliter les choses par la suite. Pour ce faire, nous utilisons les fonctions mutate(), as.Date() et as.factor() suivantes :

```
# Convertir le type de colonnes en facteur
df <- df %>%
  mutate(InvoiceNo = as.factor(InvoiceNo), StockCode = as.factor(StockCode),
         InvoiceDate = as.Date(InvoiceDate, '%m/%d/%Y %H:%M'), CustomerID = as.
             factor(CustomerID),
         Country = as.factor(Country))
```

Pour savoir le montant réel payé par chaque client, nous pouvons créer une nouvelle colonne, "Total", qui va contenir le produit de la quantité avec le prix unitaire comme suit :

```
# Créer la colonne "Total"
df <- df %>%
  mutate(Total = Quantity*UnitPrice)
```

Nous pouvons vérifier maintenant le résultat de tous ces changements comme suit :

```
# Vérifier les nouveaux changements
> glimpse(df)

Observations: 397,884
Variables: 9
$ InvoiceNo   <fctr> 536365, 536365, 536365, 536365, 536365, 536365, 536365...
$ StockCode   <fctr> 85123A, 71053, 84406B, 84029G, 84029E, 22752, 21730, 2...
$ Description <fctr> WHITE HANGING HEART T-LIGHT HOLDER, WHITE METAL LANTER...
$ Quantity    <int> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2, 3, ...
$ InvoiceDate <date> 2010-12-01, 2010-12-01, 2010-12-01, 2010-12-01, 2010-1...
$ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1.85, 1...
$ CustomerID  <fctr> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17850...
$ Country     <fctr> United Kingdom, United Kingdom, United Kingdom, United...
$ Total       <dbl> 15.30, 20.34, 22.00, 20.34, 20.34, 15.30, 25.50, 11.10,...
```

Maintenant, nous pouvons calculer les trois critères de la méthode RFM. La récence pour un client pourrait être calculée par la soustraction de la date de la plus récente facture de la de la date la plus récente dans l'ensemble de données. La plus récente date dans l'ensemble de données est "2011-12-09", on ajoute un jour à cette date pour inclure les achats effectués durant durant le jour de "2011-12-09". Par conséquent, la date devient "2011-12-10". Le résultat de la soustraction est en nombre de jours. La fréquence pourrait être calculée en comptant le nombre de factures distinctes, c'est-à-dire, le nombre d'achats que le client a effectué. Le montant pourrait être calculé en additionnant les montants que le clients a dépensé et les divisant sur la fréquence pour avoir la moyenne des dépenses de chaque client par transaction.

```
# Calcul des trois critères de la méthode RFM
df_RFM <- df %>%
  group_by(CustomerID) %>%
  summarise(Recence   = as.numeric(as.Date("2012-01-01")-max(InvoiceDate)),
            Frequence = n_distinct(InvoiceNo),
            Montant   = sum(Total)/n_distinct(InvoiceNo))
```

Nous pouvons visualiser un résumé du dataframe "df_RFM" en utilisant la fonction suivante :

```
# Afficher le résumé des résultats
> summary(df_RFM)
CustomerID       Recence          Frequence          Montant
 12346   :   1   Min.   :  1.00   Min.   :  1.000   Min.   :    3.45
 12347   :   1   1st Qu.: 18.00   1st Qu.:  1.000   1st Qu.:  178.62
 12348   :   1   Median : 51.00   Median :  2.000   Median :  293.90
 12349   :   1   Mean   : 93.06   Mean   :  4.272   Mean   :  419.17
 12350   :   1   3rd Qu.:142.75   3rd Qu.:  5.000   3rd Qu.:  430.11
 12352   :   1   Max.   :374.00   Max.   :209.000   Max.   :84236.25
 (Other):4332
```

Pour montrer la distribution de la récence pour tous les clients, nous pouvons afficher l'histogramme de chacun des critères comme suit :

```
# Afficher l'histogramme de récence
hist(df_RFM$Recence,
     main="Histogramme de récence",
     xlab="Récence",
     ylab="Client",
     border="blue",
     col="green",
     las=1) # voir la figure en bas.
```

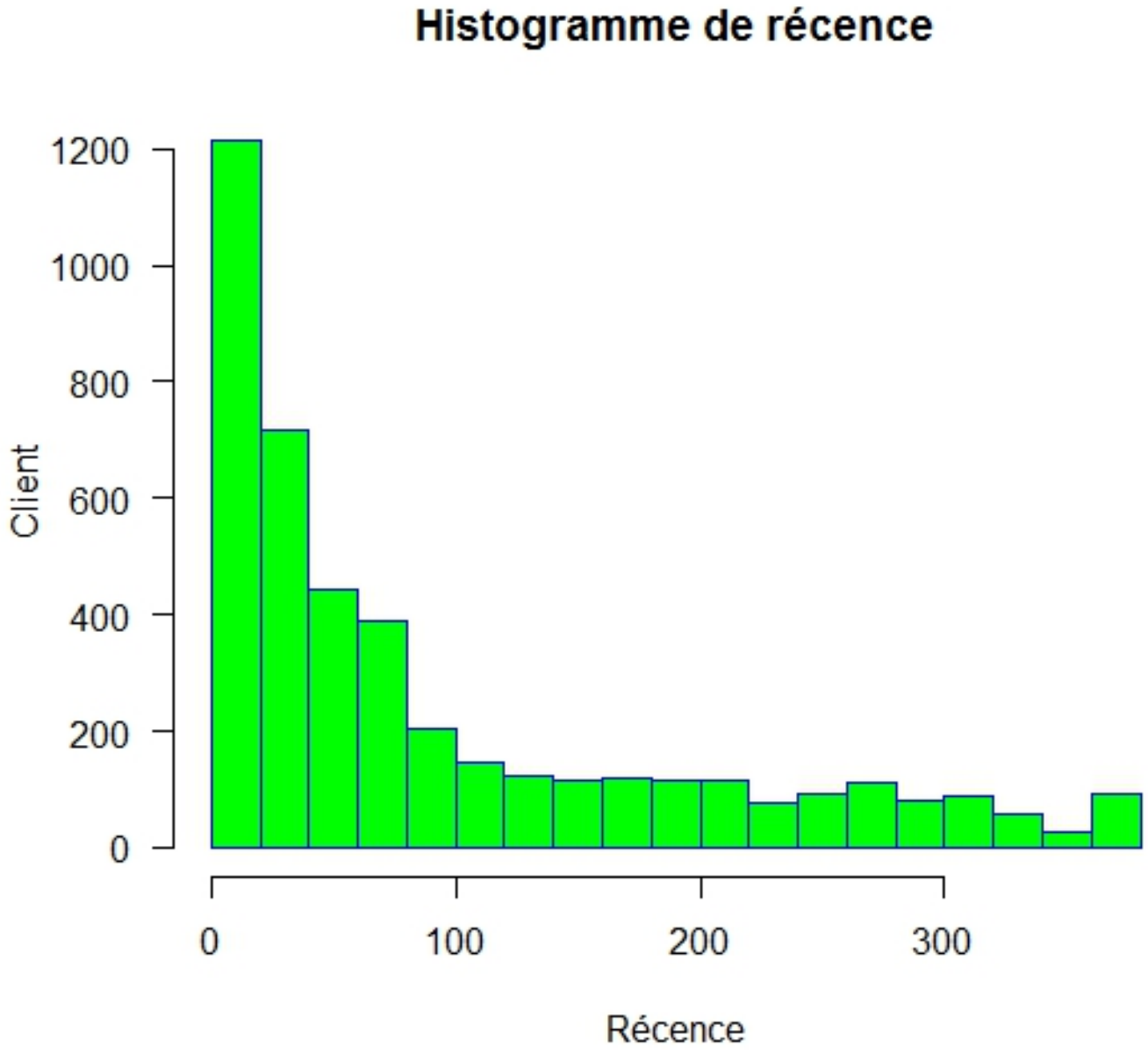

FIGURE 3 – Histogramme de récence.

De la même façon, l'histogramme de fréquence est affiché comme suit :

```
# Afficher l'histogramme de fréquence
hist(df_RFM$Frequence,
     main="Histogramme de fréquence",
     xlab="Fréquence",
     ylab="Client",
     border="blue",
     col="green",
     las=1) # voir la figure en bas.
```

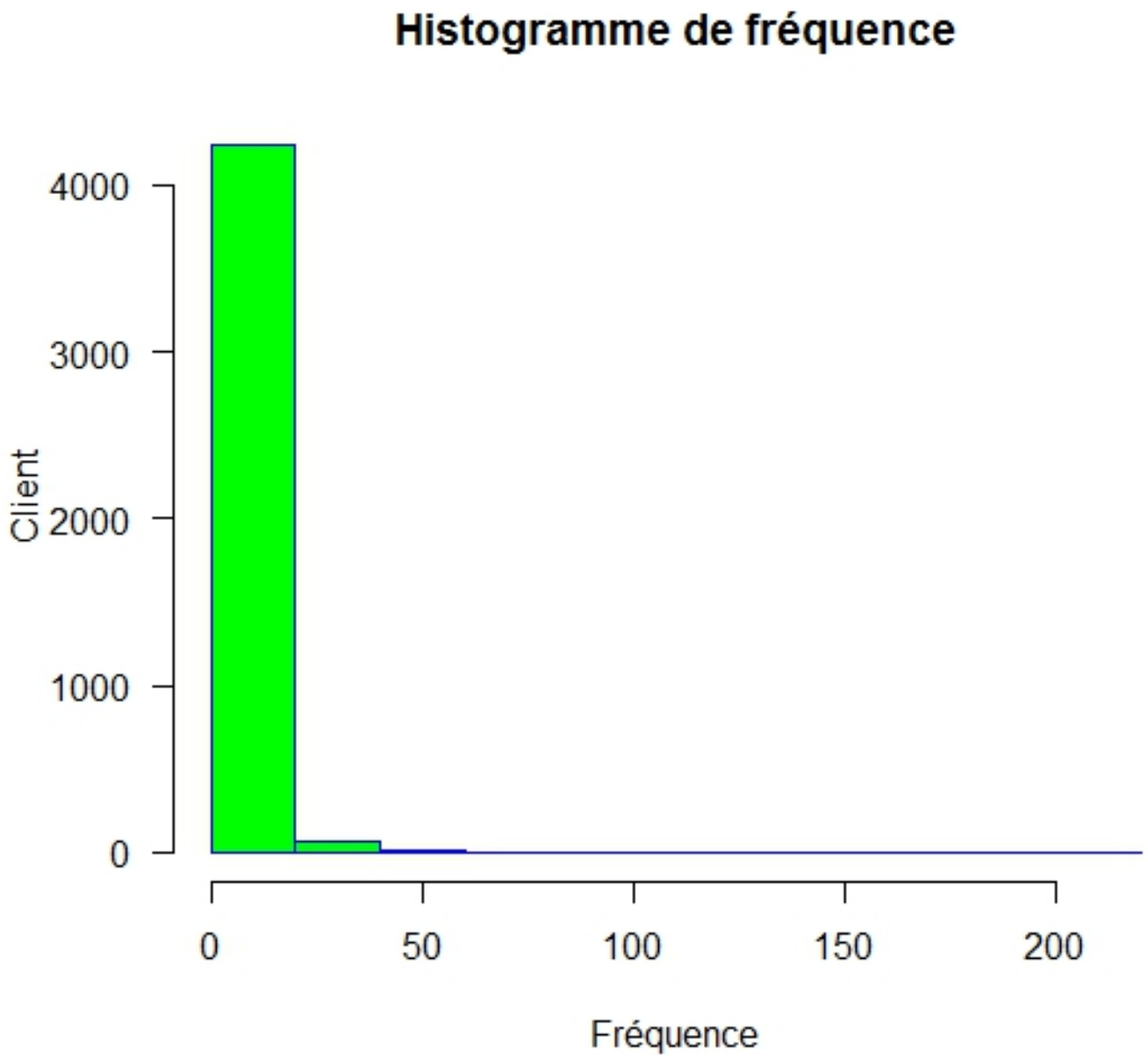

FIGURE 4 – Histogramme de fréquence.

Finalement, l'histogramme de montant est affiché comme suit :

```
# Afficher l'histogramme de montant
hist(df_RFM$Montant,
     main="Histogramme de montant",
     xlab="Montant",
     ylab="Client",
     border="blue",
     col="green",
     las=1) # voir la figure en bas.
```

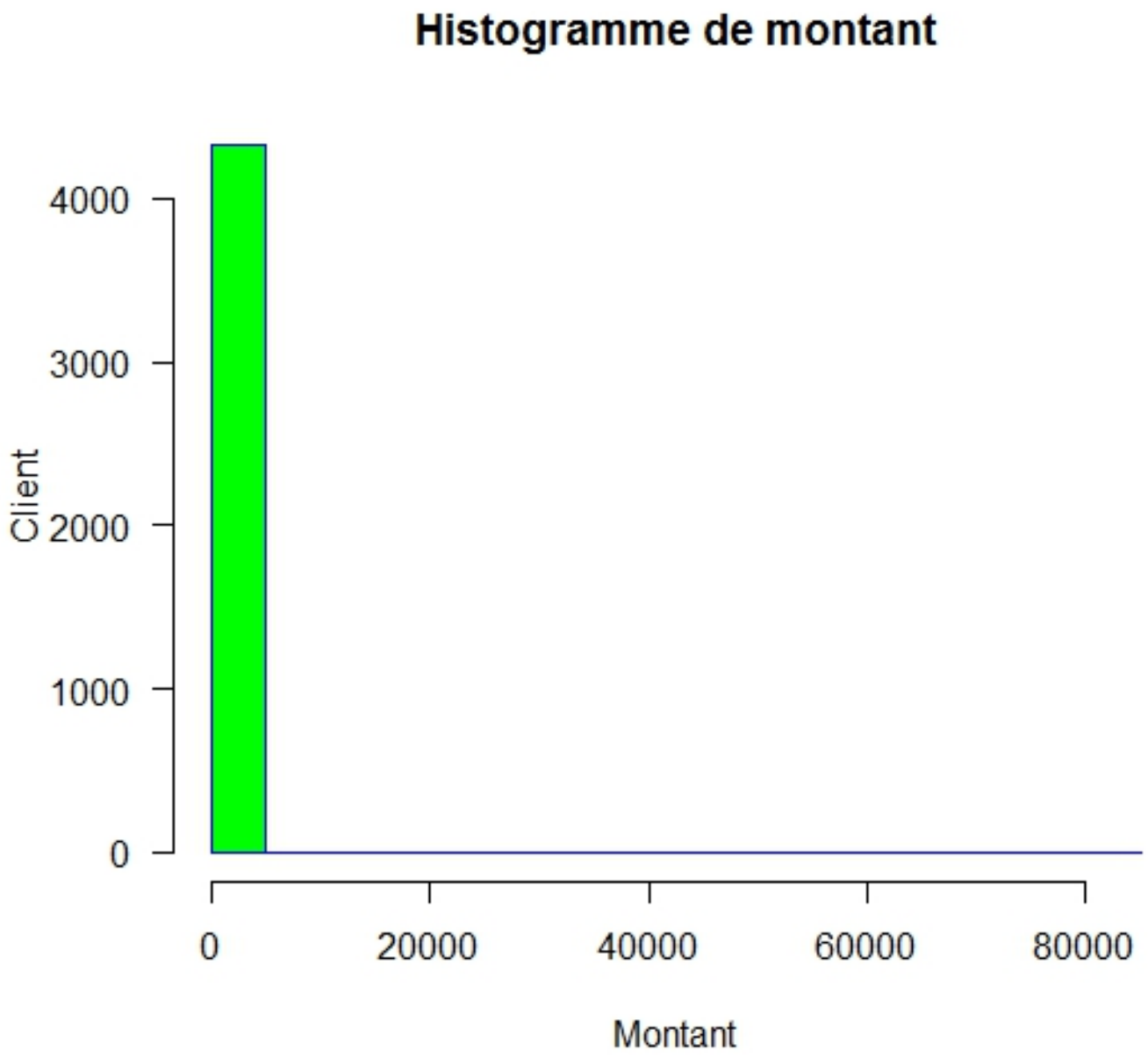

FIGURE 5 – Histogramme de montant.