DESCO - Knowledge Discovery - Association Rules

1141074 - Sérgio Silva | 1970400 - Pedro Neves | 1040706 - Sérgio Castro

5/3/2018

Neste documento pretende-se adquirir um melhor conhecimento sobre os perfis de grupos de clientes. Com esta informação, o objetivo final é a recomendação dos artigos mais indicados a cada grupo de clientes.

1. Exploração e preparação dos dados

Para o cálculo do valor de RFM dos clientes, foi efetuado tratamento dos dados das tabelas **TRANSACTION.dat**, **TRANSACTION_ITEM.dat** e **CARD.DAT**, idêntico ao realizado para a previsão de resposta a campanhas. Detalhes sobre a utilização do algoritmo RFM são descritos na outra parte do trabalho.

Ao conjunto de dados resultante, foi adicionado uma categoria para dividir os clientes por intervalos de idades. Os intervalos considerados foram: menor de 50 anos, maior ou igual a 50 e menor de 65 anos, maior ou igual a 65 anos.

```
df_customers$ageInterval <- cut(df_customers$age,
                      breaks = c(0, 50, 65, +Inf),
                      labels = c("< 50", "< 65", ">= 65"),
                      right = FALSE)
```

Verificação dos dados dos clientes.

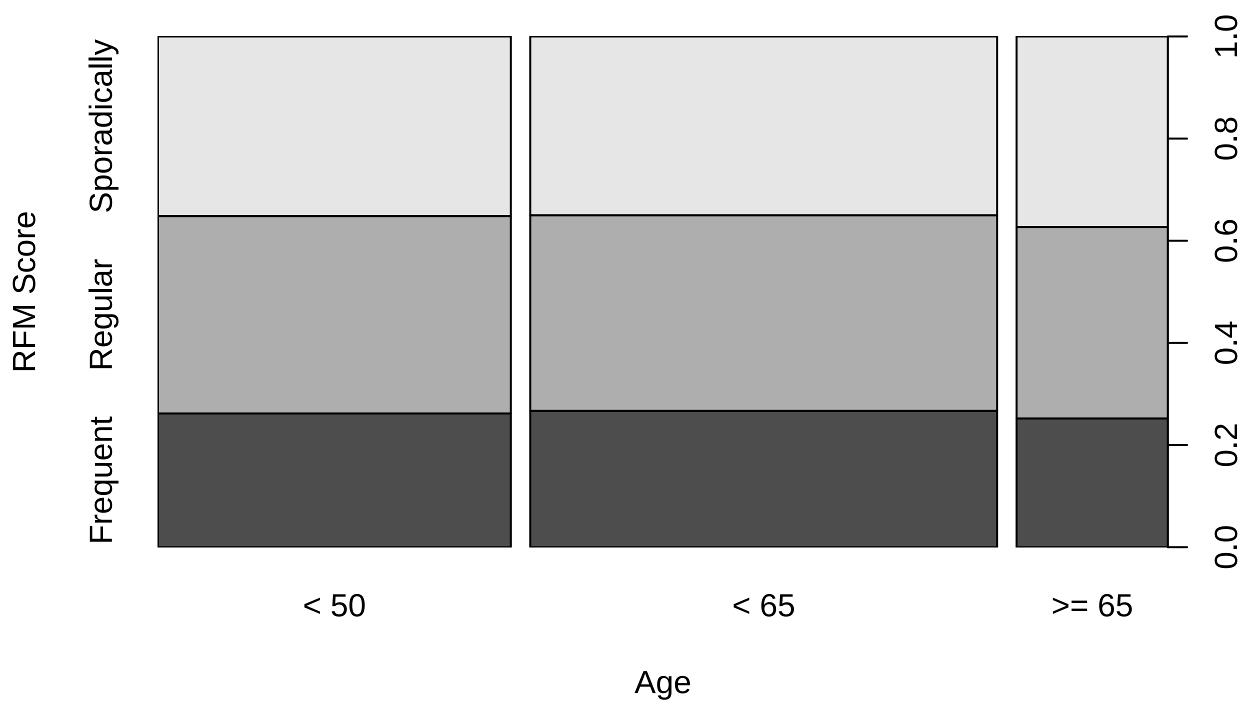
```
summary(df_customers)
```

```
##     CardID                  City           Region        PostalCode
##  Length:60519       Catburg    :10302   Central:30176   A039798 : 4467
##  Class :character   Foxton     : 9987   Eastern:30343   A001761 :  538
##  Mode  :character   Kingsville :10130                   A024496 :  445
##                     Princeton  :10042                   A0104173:  286
##                     Queensbury :10004                   A049814 :  280
##                     Ravensville:10054                   A0117302:  279
##                                                         (Other) :54224
##  CardStartDate              Gender       DateOfBirth
##  Min.   :1998-01-01   Feminino :30412   Min.   :1902-02-13
##  1st Qu.:1998-11-01   Masculino:30107   1st Qu.:1954-11-12
##  Median :1999-09-02                     Median :1962-01-26
##  Mean   :1999-09-18                     Mean   :1961-06-08
##  3rd Qu.:2000-06-29                     3rd Qu.:1969-02-08
##  Max.   :2001-12-30                     Max.   :1991-12-11
##
##   MaritalStatus   HasChildren  NumChildren    YoungestChild
##  Casado  :20033   Sim:34122   Min.   :0.000   Min.   : 0.000
##  Solteiro:20196   Não:26397   1st Qu.:0.000   1st Qu.: 0.000
##  Outro   :20290               Median :1.000   Median : 0.000
##                               Mean   :1.147   Mean   : 6.344
##                               3rd Qu.:2.000   3rd Qu.:11.000
##                               Max.   :7.000   Max.   :68.000
##
##    rfm_score        age         clientYears         rfm_score_cat
```

```
##  Min.   :111   Min.   : 23.0   Min.   :13.00   Frequent    :15904
##  1st Qu.:214   1st Qu.: 46.0   1st Qu.:15.00   Regular     :23179
##  Median :324   Median : 53.0   Median :15.00   Sporadically:21436
##  Mean   :330   Mean   : 53.6   Mean   :15.28
##  3rd Qu.:453   3rd Qu.: 60.0   3rd Qu.:16.00
##  Max.   :555   Max.   :113.0   Max.   :17.00
##
##  ageInterval
##  < 50 :21992
##  < 65 :29097
##  >= 65: 9430
##
##
##
##
```

Para a criação dos *clusters*, foi criado um conjunto de dados com os atributos mais relevantes. A variável 'CardID' é mantida para permitir a identifição das transações após a criação, mas não é utilizado para aquando da criação dos *clusters

```
dataCustomers <- df_customers[, c("CardID", "Region", "Gender", "MaritalStatus", "HasChildren",
                                  "rfm_score_cat", "clientYears", "age", "ageInterval")]

with(dataCustomers,
  plot( ageInterval, rfm_score_cat, xlab = "Age", ylab = "RFM Score")
  )
```



2. Clustering

Como a maioria dos atributos do conjunto de dados são categóricos e o algoritmo **k-means** não é diretamente aplicável a este tipo de dados, é necessário recorrer a outros tipos de algoritmos. Após pesquisa, encontramos algumas soluções que a seguir se descrevem. No entanto, apenas com o algoritmo **k-modes** foi utilizado a totalidade do conjunto de dados.

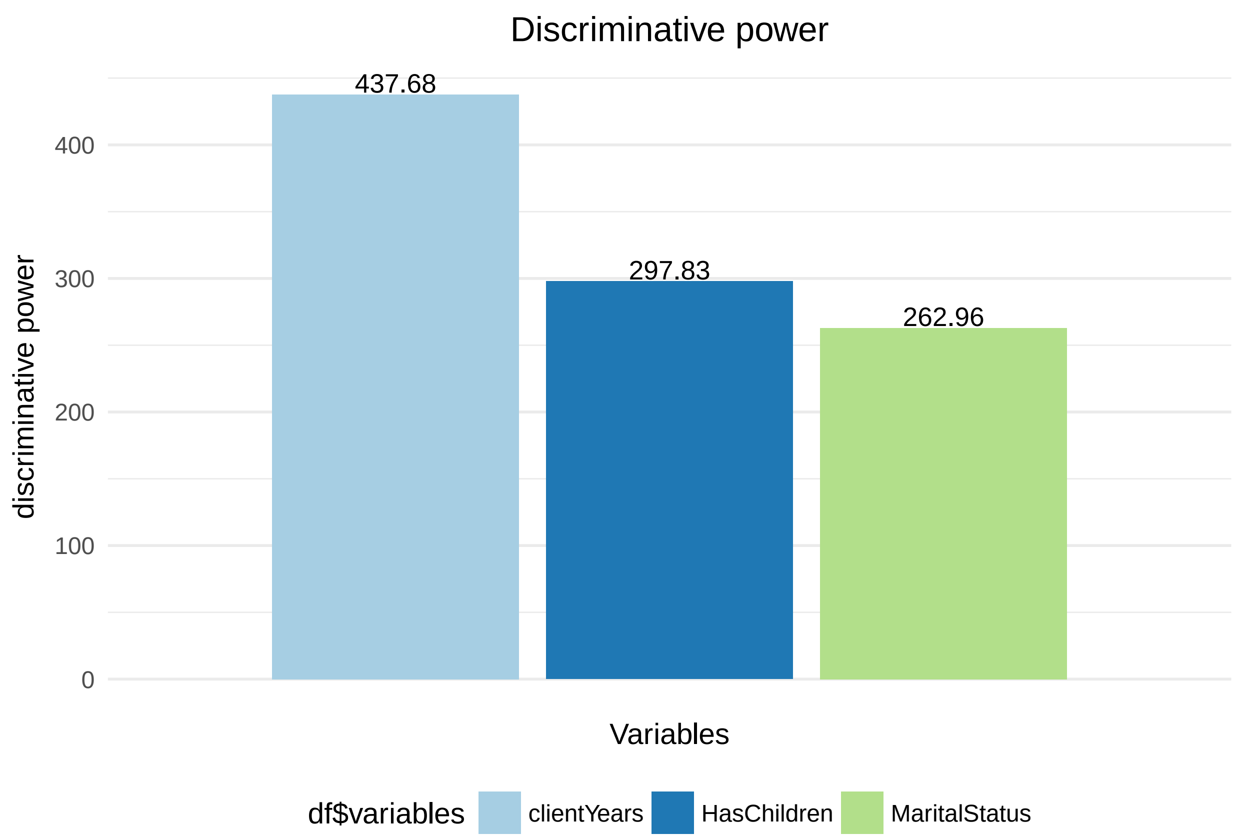
Model-based Clustering

VarSelLCM é um *package* que implementa *clustering* baseado em modelos (deteção das características relevantes e seleção do número de *clusters*), recorrendo a critérios de informação. Dados podem ser compostos por valores contínuosm, inteiros ou numéricos (Ref. **???**). Para a criação do cluster, utilizaram-se os atributos 'age' e 'clientYears', que permitiram obter melhores resultados, pelo que não é utilizado o atributo 'ageInterval'.

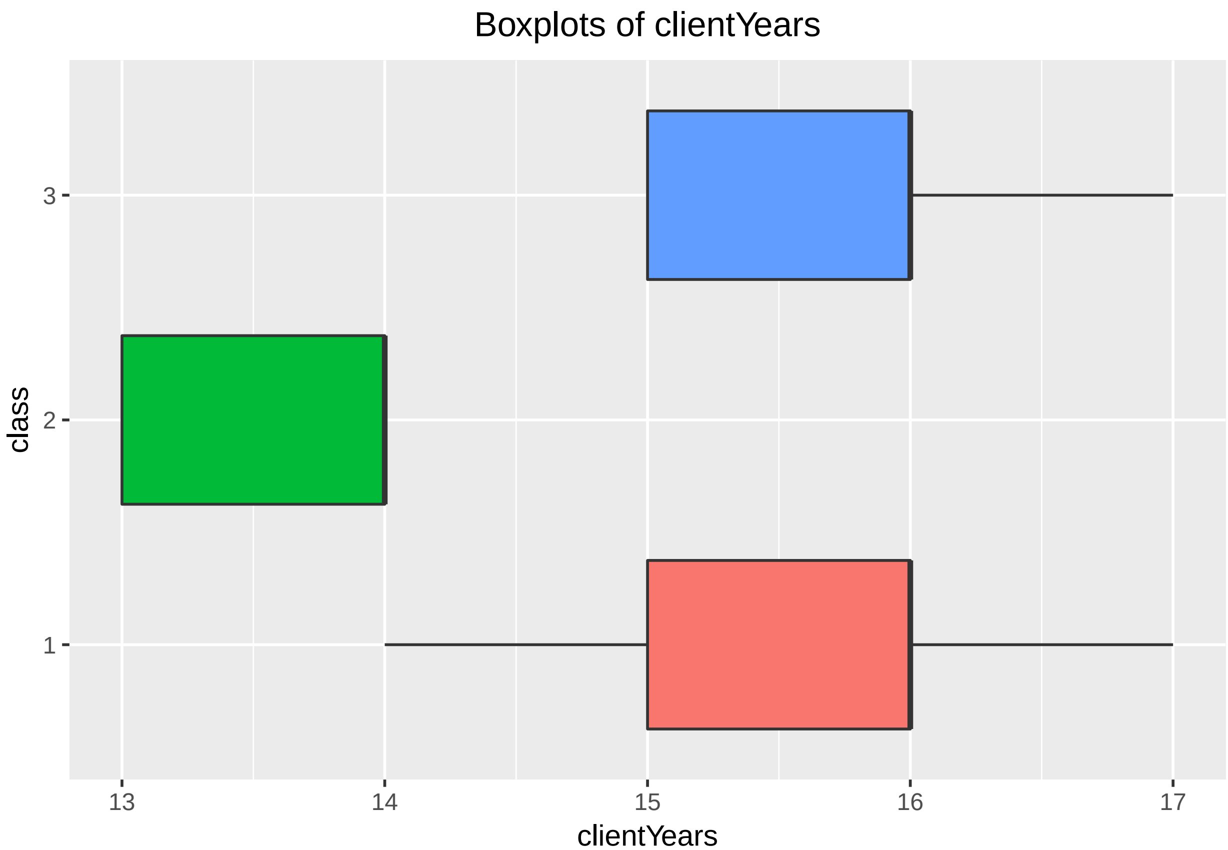
```
library(VarSelLCM)
set.seed(123)

cluster.model_base <- VarSelCluster(dataCustomers[1:1000, -c("CardID", "ageInterval")], gvals = 3, nbco

#VarSelShiny(out)

summary(cluster.model_base)
```

```
## Data set:
##    Number of individuals: 1000
##    Number of continuous variables: 2
##    Number of categorical variables: 5
##
## Model:
##    Number of components: 3
##    Model selection has been performed according to the BIC  criterion
##    Variable selection has been performed, 3  ( 42.86 % ) of the variables are relevant for clusterin
##
## Information Criteria:
##    loglike: -9494.642
##    AIC:     -9517.642
##    BIC:     -9574.081
##    ICL:     -9701.244
```

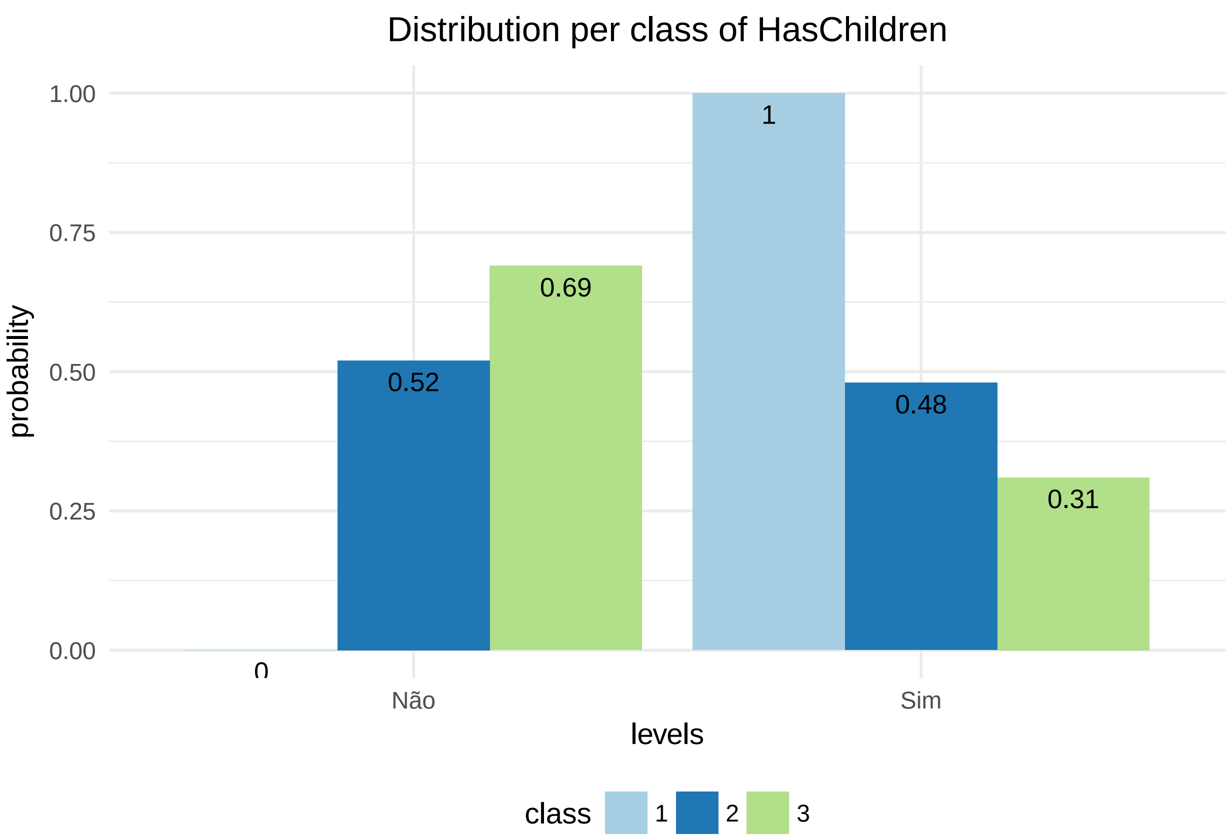
```
# As variáveis mais discriminativas do modelo podem ser visualizadas
plot(cluster.model_base, type = "bar")
```



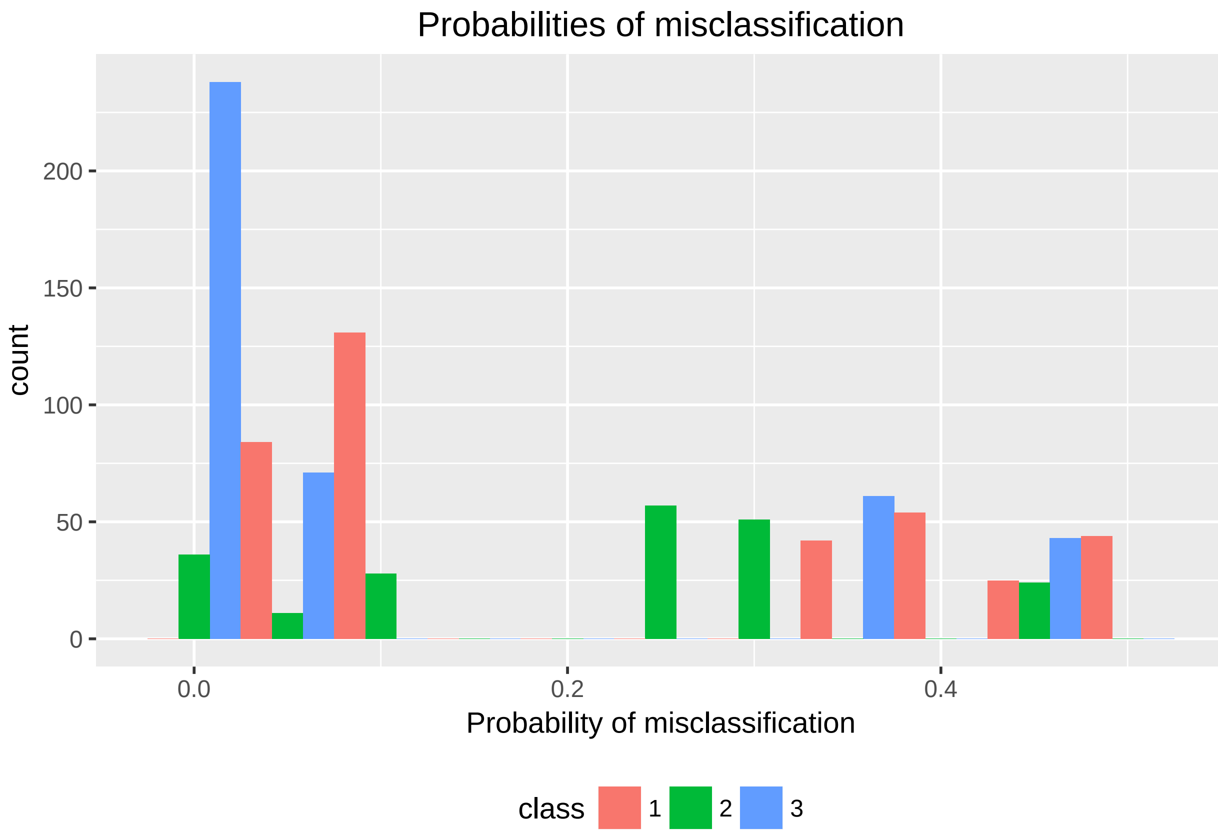

```r
# Por exemplo, a distribuição por cluster da variável HasChildren.
plot(cluster.model_base, y = "clientYears", type = "boxplot")
```




```
plot(cluster.model_base, y = "HasChildren", type = "boxplot")
```




```r
# Probabilidades de má classificação
plot(cluster.model_base, type="probs-class")
```



Analisando as duas variáveis mais discriminativa, pode-se verificar que no caso da variável 'clientYears', dois clusters pertencem à mesma gama de valores. No caso da variável 'HasChildren', também se verifica que dois clusters não têm dissemelhança significativa.

Clustering hierárquico

Novamente, como a maioria dos atributos não são númericos, é necessário utilizar uma métrica que seja possível a estes dados. Uma possibilidade é a utilização da métrica de Gower (Ref. **???**). A função 'daisy()' do package 'cluster' contem uma implementação desta métrica. Para o calculo da matriz de distância não foram utilizados os atributos 'age' e 'clientYears'. De notar que a utilização desta métrica obriga a manter uma matriz NxN em memória, o que muito rapidamente se torna , pelo que apenas consideramos um subconjunto dos dados.

```
library(cluster)
set.seed(123)

gower.dist <- daisy(dataCustomers[1:2000, -c("CardID", "clientYears", "age")], metric = "gower")
summary(gower.dist)
```

```
## 1999000 dissimilarities, summarized :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.5000  0.5000  0.5656  0.6667  1.0000
## Metric :  mixed ;  Types = N, N, N, N, N, N
## Number of objects : 2000
```

```
gower.mat <- as.matrix(gower.dist)
```

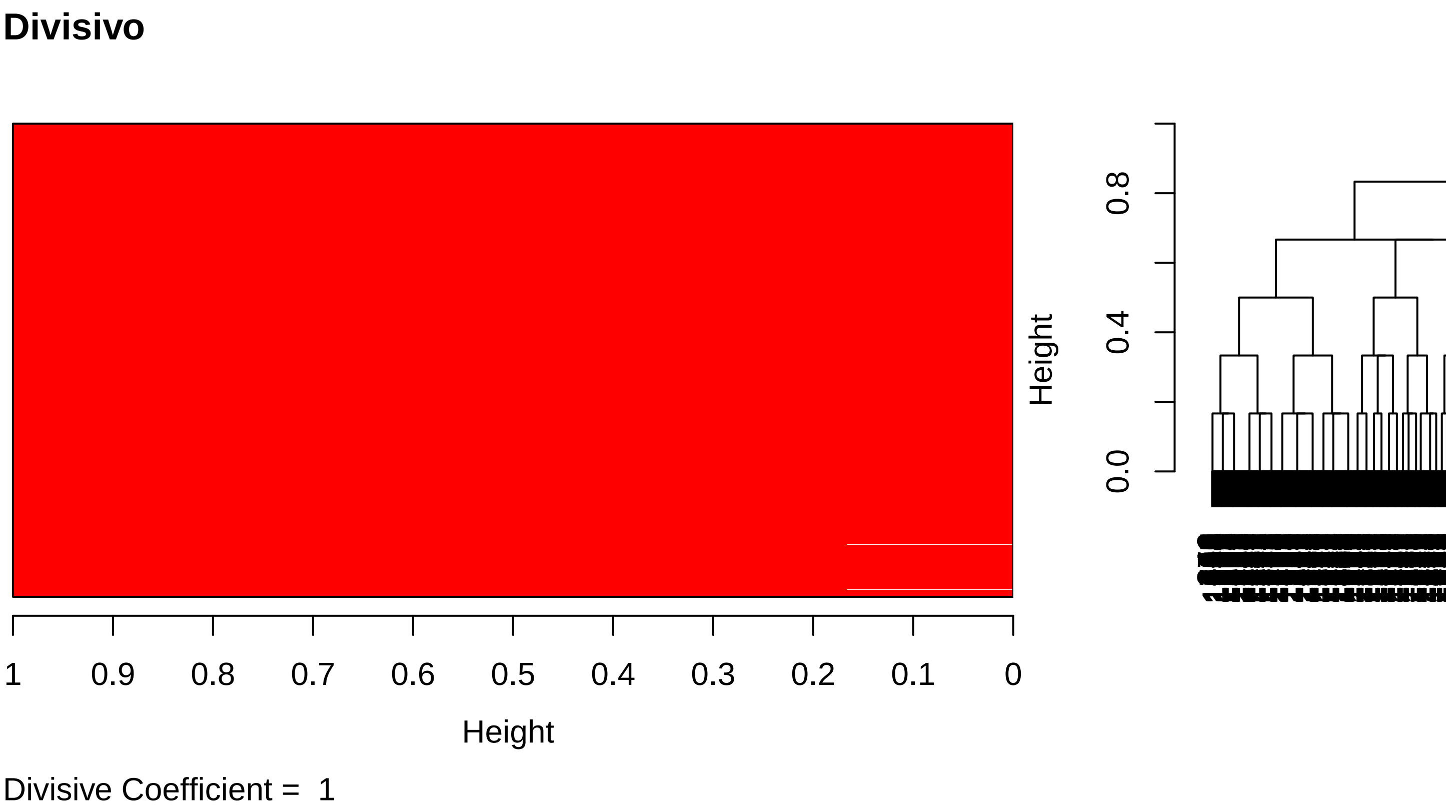
```r
# Par mais "semelhante"
dataCustomers[
  which(gower.mat == min(gower.mat[gower.mat != min(gower.mat)]),
        arr.ind = TRUE)[1, ], ]
```

```
##          CardID  Region   Gender MaritalStatus HasChildren rfm_score_cat
## 1: C0100000726 Eastern Feminino        Casado         Sim      Frequent
## 2: C0100000111 Eastern Feminino        Casado         Sim  Sporadically
##    clientYears age ageInterval
## 1:          14  59        < 65
## 2:          14  51        < 65
```

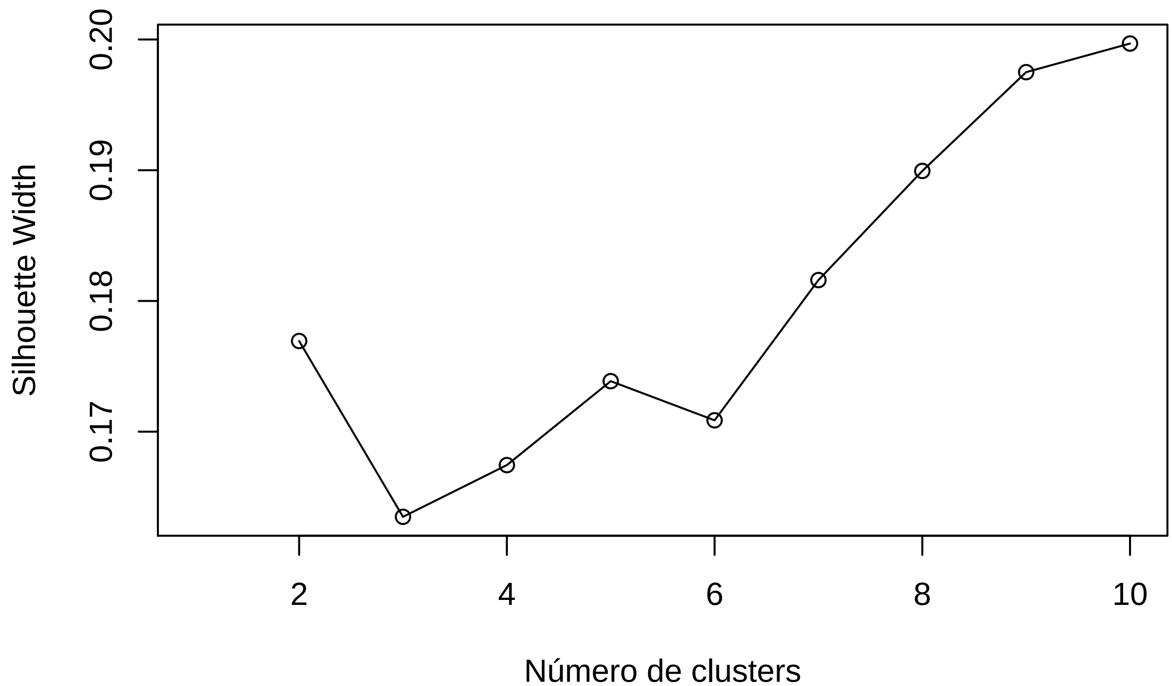
```r
# Para menos "semelhante"
dataCustomers[
  which(gower.mat == max(gower.mat[gower.mat != max(gower.mat)]),
        arr.ind = TRUE)[1, ], ]
```

```
##          CardID  Region    Gender MaritalStatus HasChildren rfm_score_cat
## 1: C0100000375 Eastern Masculino      Solteiro         Não       Regular
## 2: C0100000111 Eastern  Feminino        Casado         Sim  Sporadically
##    clientYears age ageInterval
## 1:          14  47        < 50
## 2:          14  51        < 65
```

```r
# Clustering hierárquico "divisivo"" (DIANA)
divisive.clust <- diana(as.matrix(gower.dist), diss = TRUE, keep.diss = TRUE)

plot(divisive.clust, main = "Divisivo")
```



```r
# Clustering PAM (Partition around medoids)
sil_width <- c(NA)

for(i in 2:10) {

  pam_fit <- pam(gower.dist, diss = TRUE, k = i)

  sil_width[i] <- pam_fit$silinfo$avg.width
}

# Plot sihouette width (higher is better)

plot(1:10, sil_width, xlab = "Número de clusters", ylab = "Silhouette Width")
lines(1:10, sil_width)
```



Considerando este subconjunto de dados, constituído por 2000 observações, o número recomendado de clusters seria 10:

```r
# ver https://www.r-bloggers.com/clustering-mixed-data-types-in-r/

pam_fit <- pam(gower.dist, diss = TRUE, k = 10)

dataCustomers[pam_fit$medoids, ]
```

```
##          CardID  Region    Gender MaritalStatus HasChildren rfm_score_cat
##   1: C0100104746 Eastern Masculino        Casado         Sim  Sporadically
##   2: C0100100325 Central Masculino         Outro         Sim      Frequent
##   3: C0100001139 Eastern  Feminino         Outro         Sim  Sporadically
##   4: C0100001872 Eastern Masculino        Casado         Sim       Regular
##   5: C0100003292 Eastern  Feminino      Solteiro         Sim      Frequent
##   6: C0100097331 Central  Feminino        Casado         Sim       Regular
##   7: C0100104155 Central Masculino      Solteiro         Não  Sporadically
```

```
##  8: C0100018670 Eastern  Feminino      Solteiro          Não      Frequent
##  9: C0100011948 Central  Feminino      Solteiro          Não       Regular
## 10: C0100099715 Eastern Masculino         Outro          Não       Regular
##     clientYears age ageInterval
##  1:          14  50        < 65
##  2:          15  64        < 65
##  3:          16  47        < 50
##  4:          14  48        < 50
##  5:          16  63        < 65
##  6:          17  63        < 65
##  7:          17  64        < 65
##  8:          15  55        < 65
##  9:          13  40        < 50
## 10:          16  56        < 65
```

```
pam_fit$clustering
```

```
##    [1]  1  2  3  4  5  6  1  4  5  1  2  2  5  7  6  5  1  2  8  5  2  3  4
##   [24]  9  3  6  3  4  8  3  3  9 10  2  9  6  1  8  4  9  6  8 10  8  8  6
##   [47]  4  4  2  2 10  2  6  2  8  7  3  8  3  3  7 10  9  3  8  5  6  4  7
##   [70] 10  6  6 10  2  9  3  3  4  6  9  5  2  5  4  8  3 10  4  2  1  1  9
##   [93]  1  3  1  2  7  1  3  7 10  3  4  7  3  4  7  3  5  3  1  2 10  6  4
##  [116]  9  7  4 10  7  8  4  6  4  4  2  4  7  7  7  3  6  3  7  7  7  5  7
##  [139]  2  6 10 10  6  5  5  3  2  4  9  4  8  1  3  6  5  8  7  9  6  6  4
##  [162] 10  3  4  7  1  5  8  5  8  8  1  2  3  1 10  6  4 10  7  7  1  2 10
##  [185]  6 10 10  7  3  3  5  1  4  3  1  1  3  6  7  6 10  4  6  4  4  6  8
##  [208]  1  4  6  1  9  4  5  2  2  7  5  6  7  1  7  1 10  9  8  2  9  6  2
##  [231]  3  4  4  6  8  7 10  9  5  8  2  5 10  2  9  1  7 10  3  1  9  4  7
##  [254]  4 10  7  3  3  9  9  4  2  8  3  8  6  6  6  3  3  8  7  3  7  9  2
##  [277] 10  4  2  8  4  3  6  7  8  6  6  5 10  4  5  4  4  4  9  3  7  6  4
##  [300]  4  7  5 10  1  3  9  4  9 10  7  7  7  9  3  2  8  9  4 10 10  3  4
##  [323]  1  2  7  4  1  6  7  2  9  4  9  7  8  5  2  3  4  3  4  8  9  9  1
##  [346]  9  3  4  1  7  2  4  1  6  9  5  3  5  9  7  1  6  8  3  9 10  7  5
##  [369]  7  2  9  1  7  2  4  4  7  1  7  3  6  5  2  6  7  6  5  5  2  9  7
##  [392]  9  5  2  8  8  3  2  1  8  4  7  9  7  8  3  1  2  6  2  6  1  4  3
##  [415]  4  7  8  8  1  3  2  3  4  8  4  9  6  7  3  2  9  9  2  3  3  1 10
##  [438]  1 10  7  2  1  8 10  1  4  7  3  7  3  4  5  3  1  9  2  8 10  9  8
##  [461]  9  4  2  6  2  1  5  4  3  9  3  6  6  1  4  4  4  7 10 10  7  4  4
##  [484]  9  1  3  9  4  6  8  3  6 10  3  5  8  2  6  6  9  1  4  3  3  2  3
##  [507]  9  4  7  2  8  8  3 10  3  7  8  6  2  5  2  3  7  8  5  6  2  9  7
##  [530]  9  4  4 10  4 10  9  9  9  8  5  2  6  4  3  7  7  7  3  3  4  3  4
##  [553]  4  6  9  6  9  3  9  2  3  2  5  8  8  8  8  2  8  3  6  8  3  4  6
##  [576]  5 10  3  4  4  4 10  7  3  7  4  6  6  7  6  5  3  7  6  7  7  5  3
##  [599]  9  9  1  6  3  7  2  3  4 10  5  4  4  9  7  3  2  8  4  3  9  2  4
##  [622]  4 10  3  3  5  6  4  3  1  2  5  3  8  3  7  5  3  1  4  5  6  6  9
##  [645]  8  5  4  8  8 10  2  9  3  8  3  4  1  6  7  3  8  3  5  6  3  3  3
##  [668]  6  4  2  3  6  9  6  9  9  2  8  2  8  3  6  7  6  9  4  6  5  7  6
##  [691]  6  6  6  3  3  3  3  4  3  4  8  9  6  6  9  3  3  7  9  1  9  3  6
##  [714]  1  7  5  6  5  9  9  3  5  9  7  4  4  1  3  8  4  6  1  2 10  1  7
##  [737]  4  2  6  7  3  8  3  1  7 10  1  5  4  8  2  2  4  8  2  4  3  2  7
##  [760]  3  3  4  7 10  9  9  6  2  8  6  6  3  7  4  7  9  3  8  4  2  2  3
##  [783]  3  1  6  4  6  9  6  3  3  4  8  5  2  3  1  3  3  2  5  2  2  2  5
##  [806]  7  9  3  4  4  4  9  3  1  6  6  7  4  6  6  7  7  3  4  9 10  8  2
##  [829]  8  3 10  6 10  2  8  4  2  6  6  6 10  5  8 10  2  3  2  3  3  8  8
##  [852]  3  8  1  7  9  1  2  8  5  5  3  9  4  4  9 10  6  4  7  6  4  8  1
```

```
##  [875]  6  2  4 10 10  4  6  1  6  2 10  8  4  2  2  2  6  3  3  1  1  6  5
##  [898]  1  5 10  4  6  4  4 10  7  9  9  4  5  2  4  9  9  8  6  3 10  7  4
##  [921] 10  7  6  6  6  9  1  6  3  9  3  3  1  8  8  6  7  1  5  4  3  2  8
##  [944]  3  3  6  1  3  3  7  7  5  8  2  3  6  8  4  6  7  7  6 10  5  3  2
##  [967] 10  3  3  7  8  7  4  6  2  6  9  3  9  9 10  5  7  6 10  2  5  1  3
##  [990]  1 10  1  9  3  2  7  1  6  9  8  2  9 10  5 10 10  8  6 10  4  4  2
## [1013]  5  4  2 10  3  6  7 10  9  3  6  3  9  3  8  2  9  1  6  3 10  6  2
## [1036]  6 10  8  3  8  6  4  4  4 10  3  9  7  4  6  5  7  6  2  8 10  8  8
## [1059]  3  4  8  1  8  1  4  9 10 10  6  3  4  7  4  3  6  2  4  3  6  6  6
## [1082]  7  3  8  7  4 10  7  5  4  9  2  1 10  3  6  3  3  2  6  4 10  7  7
## [1105]  7 10  8 10  9  9  9  6  6  7 10  1  4  5  8  3  3  8  8  8  5  4  4
## [1128]  5  6  4  7  2  4  3  8 10  7 10  6  9  9  4  3 10  7  3  1  3  7  3
## [1151]  5  6 10  9  9  7  8  6  9  3  3  9 10  6  6  2  7  9  7 10 10 10  7
## [1174] 10  3  4  6 10  6  3  3 10  3  3  5  4  3  6  9  1  3 10  6  3  4  6
## [1197]  7  5  9  7  4  5  8  2  8  6  9  3  6  2  1  1  4  3  2  7  4  3  5
## [1220]  6  7  4  2  8 10  8  8  7  8  2  4  3  4  2  1  8  1  5  4  4  9 10
## [1243]  8  5  9  4  6  5  6  9  2  4  3  9  7  3  7  8  7  1  9  7  7  2  6
## [1266]  3  9  3  9 10  4  5  9  8  9 10  4  3  8  3 10  5  7  6  4  8  5  7
## [1289]  3  8  3 10 10  7  4  6  7 10  4  6 10  7  3  3  4  2  2  9  4  3  4
## [1312] 10  9  6  2 10  8  7  4  9  3  2  9  1  5  3  6  2  8 10 10  6  9  4
## [1335]  7  8  9  6  7  6  7  3  3  8  8  2 10  3  8  8  6  3  3  5  7  3  1
## [1358]  3  3  5  1  7  3  3  5 10  3  8  7  4  1  1  9  8  8  9  4  3  3  1
## [1381]  3  4  4  8  2  1  6  9  3  9  3  3  5  1  6  5  2  1  6  1  8  6  3
## [1404]  3  4  8  7  6  7  9  6  3  2  6 10  1  1  3  3  3  4  4  5  2  9  9
## [1427]  6  2  1  8  2  7  5  9  7  9  6  6  9  6 10  3  3  7  2  6  4  2  9
## [1450]  5  9  6  3  5  7  6  4  7  5  2  2  8  4  3  5  3  1  2  2  9  1  7
## [1473]  4 10  2  7  9  8  5  5  7  3  8  3  8  6  2  2  2  8  1  3  4  9  4
## [1496]  3  5  6  7  2 10  8  3  6  9  3  5  4  9  3  2  4  2  8 10  5  1  6
## [1519]  3  1  9  7  5 10 10  6  7  1  8  9  9  3  9  9  4 10  1  7  5  6  8
## [1542]  2  8  8  7  2 10  8  6  9 10  4 10  8  2  1 10  8  7  6  8  1  8  1
## [1565]  7  3 10  3  6  1  7  7  6  6  3  4  7 10  4  2  6  9  5  8  5  3  3
## [1588]  3  4  2 10  4 10  9  3  8  7  5  3  8  5  6  7  3  7  9  9  2  6  2
## [1611]  6  9  6  3  3  3 10 10 10  5  3  4 10  4 10  4  6  1  7  4  3  4 10
## [1634]  9  5 10  6  3  3  6  4  9  2  1 10  7  8  3  2  2  8  8  9  4  9  8
## [1657]  4  8  1  2  4  5 10  1  6  8  1  5  2  6  4  7  9  1  2  6  6  3  4
## [1680]  6  8  8  8  5  7  8  7  2  2 10  7  7  6  4  7  7 10  3  8  1  9  7
## [1703] 10  1  2  8  3 10  4  4  1  8  3  5  4  4  5  4 10  5  1  2  4  3  8
## [1726] 10 10  4  5  8  6 10  9  6  4  9  3  8  9  9  9  4  3  3  5  3  2  8
## [1749]  7 10  9  9  1  8  5  9  4  7  3  9  6  6  5 10  1  9  3  4  8  1  9
## [1772]  4  7 10  3  1  6 10  6  1 10  5  7  2  6  4 10  2  2  9  3  9  5  8
## [1795]  1  7  2  3  9  5  1  7  2  1  5  9  1  1  5  6  4  3 10  6 10  5  2
## [1818]  4  3  9  3  2  7  6  8  2  2  2  5  8 10  7  4  4 10  3 10  6  1  9
## [1841] 10  2 10  8  6  3 10  3  3  7 10 10  7  4  9  7  2  7  9  8  4  7  4
## [1864]  2  7  8  2  9  4 10  1  4  9  4  4 10  5 10  4  5  2  5  1 10  4  9
## [1887]  3  3  9  6  7  8  8 10  7  7  9  4  4  1  5  6  3  4  3  6  1  8  1
## [1910]  6  6  5  6  9  8  7  2  2  8  3  6  2  8  2  1  8  3  3  7  6  9  5
## [1933]  8  8  4  7  2  3  7  3  6  1  7  8  4  4  1  9  2  6 10  5  2  5  2
## [1956]  8  4  1  2  1  1  7  7  2  7  3  9  6  7  8  1  9  4  9  1  6  9  4
## [1979]  2  8  6 10 10  1  1  3  9  3  8  9  3  2  6 10  5  6  4  6  6  3
```

Clustering com o algoritmo k-modes

k-modes é uma variante do *k-means* que é aplicável a dados categóricos (Ref. **???**).

```r
library(klaR)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
set.seed(123)

# Set number of clusters
kNumberClusters <- 3

# without age and clientYears
clusters.kmodes <- kmodes(dataCustomers[, -c("CardID", "age", "clientYears")], modes = kNumberClusters,

# Place customer in its cluster
dataCustomers$cluster <- clusters.kmodes$cluster
clusters <- split(dataCustomers, dataCustomers$cluster)
```

Visualize differences between clusters

```r
dataCustomers[, .N, by = .(cluster, Gender)][order(cluster, Gender)]
```

```
##    cluster    Gender     N
## 1:       1  Feminino 19734
## 2:       1 Masculino  6046
## 3:       2  Feminino  4725
## 4:       2 Masculino 12155
## 5:       3  Feminino  5953
## 6:       3 Masculino 11906
```

```r
dataCustomers[, .N, by = .(cluster, Region)][order(cluster, Region)]
```

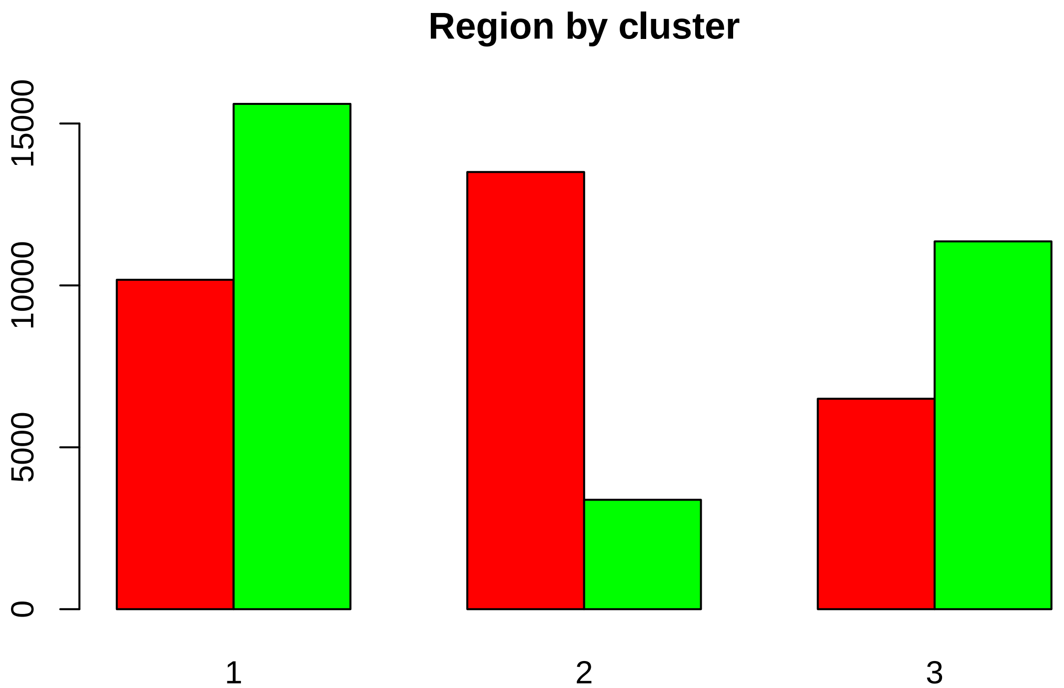
```
##    cluster  Region     N
## 1:       1 Central 10174
## 2:       1 Eastern 15606
## 3:       2 Central 13502
## 4:       2 Eastern  3378
## 5:       3 Central  6500
## 6:       3 Eastern 11359
```

```r
dataCustomers[, .N, by = .(cluster, rfm_score_cat)][order(cluster, rfm_score_cat)]
```

```
##    cluster rfm_score_cat     N
## 1:       1      Frequent  3818
## 2:       1       Regular  7695
## 3:       1  Sporadically 14267
## 4:       2      Frequent  8128
## 5:       2       Regular  4367
## 6:       2  Sporadically  4385
## 7:       3      Frequent  3958
```

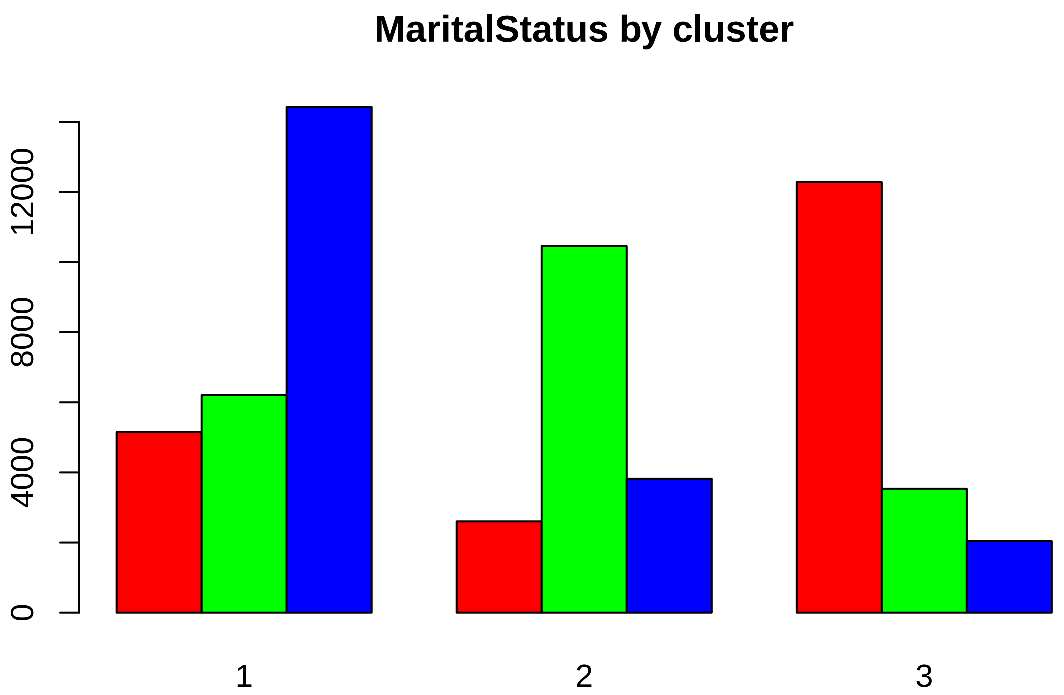
```
## 8:       3       Regular 11117
## 9:       3  Sporadically  2784
```

```r
barplot(table(dataCustomers$Region, dataCustomers$cluster),
        beside = T, col = c("red", "green"),
        main = "Region by cluster")
```

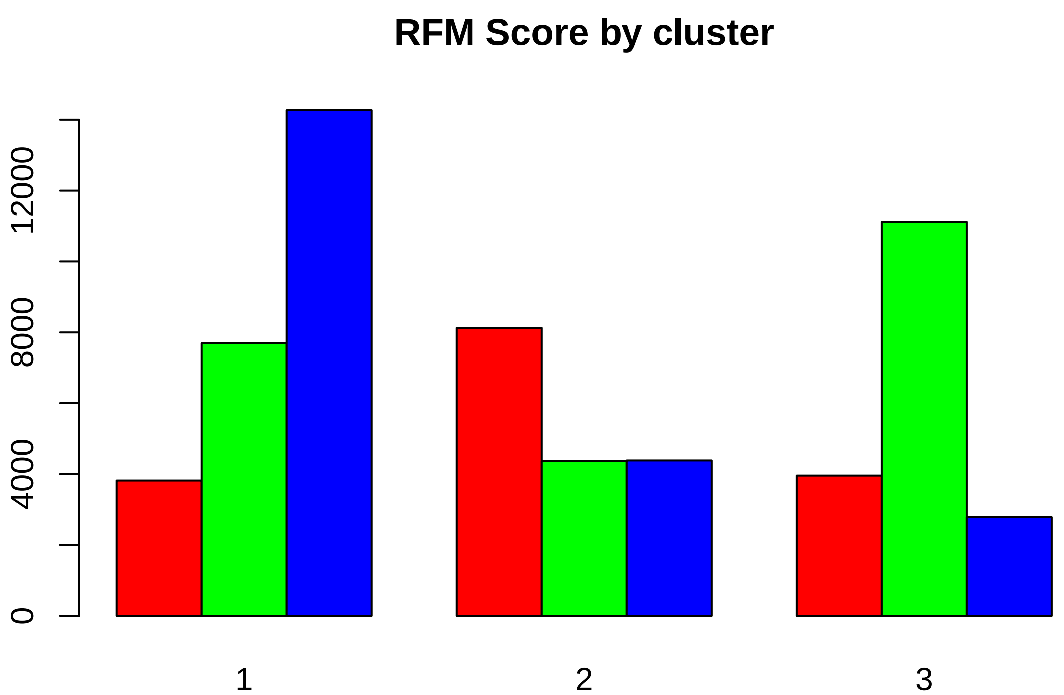



```r
barplot(table(dataCustomers$Gender, dataCustomers$cluster),
        beside = T, col = c("red", "green"),
        main = "Gender by cluster")
```

```r
barplot(table(dataCustomers$MaritalStatus, dataCustomers$cluster),
        beside = T, col = c("red", "green", "blue"),
        main = "MaritalStatus by cluster")
```

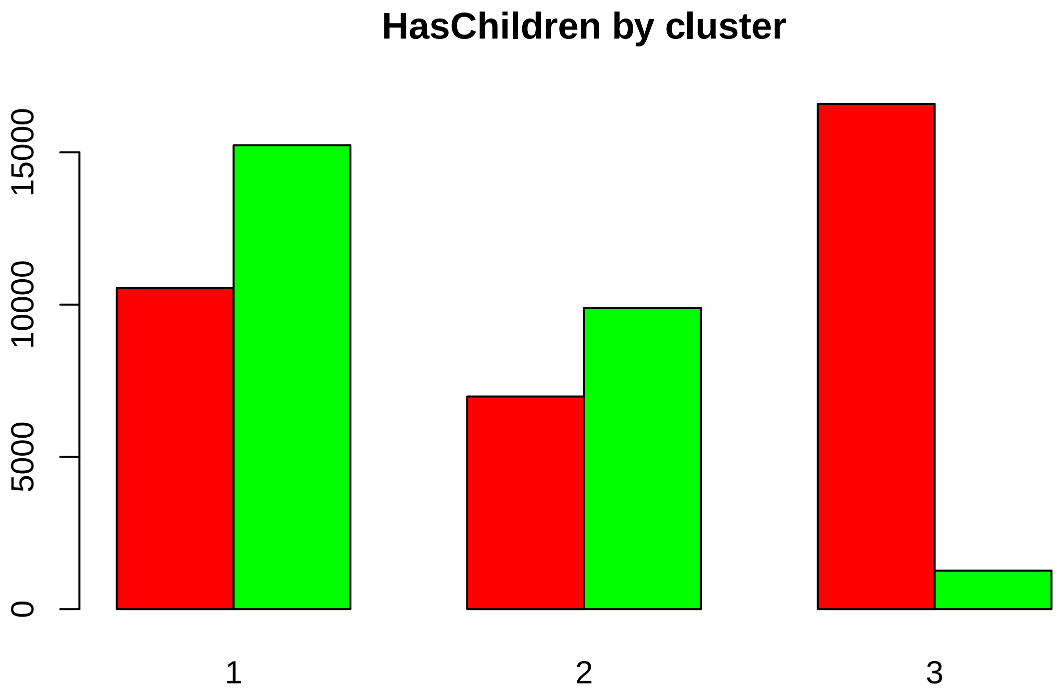



```r
barplot(table(dataCustomers$rfm_score_cat, dataCustomers$cluster),
        beside = T, col = c("red", "green", "blue"),
        main = "RFM Score by cluster")
```

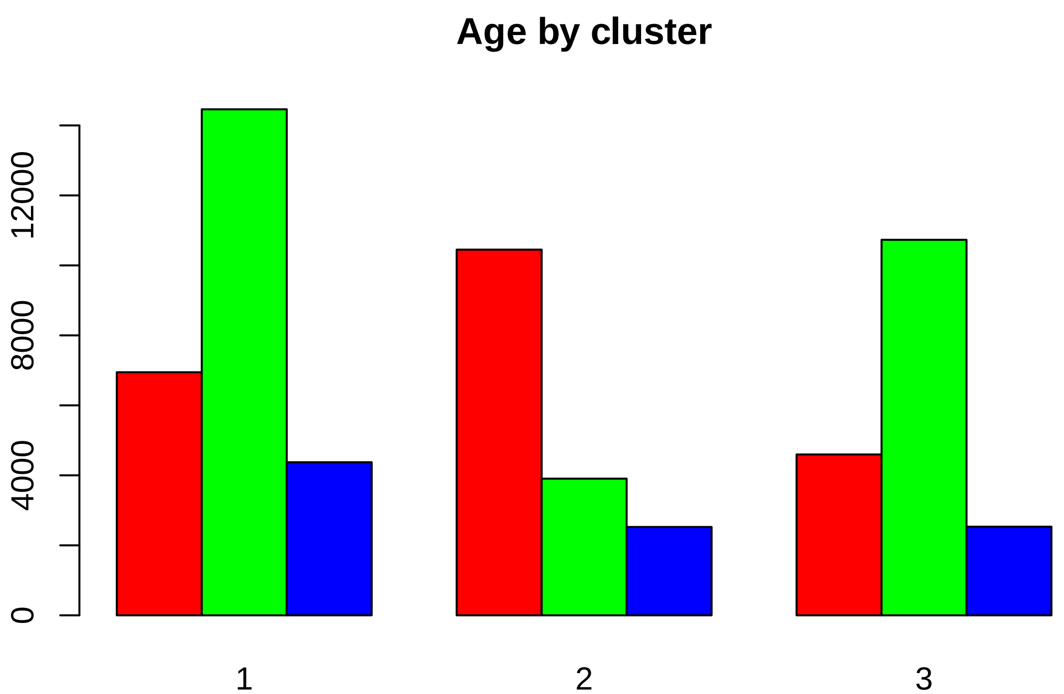



```r
barplot(table(dataCustomers$HasChildren, dataCustomers$cluster),
        beside = T, col = c("red", "green"),
```

```
        main = "HasChildren by cluster")
```




```
barplot(table(dataCustomers$ageInterval, dataCustomers$cluster),
        beside = T, col = c("red", "green", "blue"),
        main = "Age by cluster")
```



Differences between clusters

```
dataCustomers.cl1 <- clusters[[1]]
round(prop.table(table(dataCustomers.cl1$Region))*100, digits = 2)
```

```
##
## Central Eastern
##   39.46   60.54
```

```
round(prop.table(table(dataCustomers.cl1$Gender))*100, digits = 2)
```

```
##
##  Feminino Masculino
##     76.55     23.45
```

```
round(prop.table(table(dataCustomers.cl1$MaritalStatus))*100, digits = 2)
```

```
##
##   Casado Solteiro    Outro
##    19.97    24.07    55.97
```

```
round(prop.table(table(dataCustomers.cl1$HasChildren))*100, digits = 2)
```

```
##
##   Sim   Não
## 40.91 59.09
```

```
round(prop.table(table(dataCustomers.cl1$rfm_score_cat))*100, digits = 2)
```

```
##
##     Frequent      Regular Sporadically
##        14.81        29.85        55.34
```

```
round(prop.table(table(dataCustomers.cl1$ageInterval))*100, digits = 2)
```

```
##
##  < 50  < 65 >= 65
## 26.94 56.09 16.97
```

```
dataCustomers.cl2 <- clusters[[2]]
round(prop.table(table(dataCustomers.cl2$Region))*100, digits = 2)
```

```
##
## Central Eastern
##   79.99   20.01
```

```
round(prop.table(table(dataCustomers.cl2$Gender))*100, digits = 2)
```

```
##
##  Feminino Masculino
##     27.99     72.01
```

```
round(prop.table(table(dataCustomers.cl2$MaritalStatus))*100, digits = 2)
```

```
##
##   Casado Solteiro    Outro
##    15.42    61.94    22.64
```

```
round(prop.table(table(dataCustomers.cl2$HasChildren))*100, digits = 2)
```

```
##
##   Sim   Não
## 41.37 58.63
```

```r
round(prop.table(table(dataCustomers.cl2$rfm_score_cat))*100, digits = 2)
```

```
##
##     Frequent      Regular Sporadically
##        48.15        25.87        25.98
```

```r
round(prop.table(table(dataCustomers.cl2$ageInterval))*100, digits = 2)
```

```
##
##  < 50  < 65 >= 65
## 61.91 23.13 14.95
```

```r
dataCustomers.cl3 <- clusters[[3]]
round(prop.table(table(dataCustomers.cl3$Region))*100, digits = 2)
```

```
##
## Central Eastern
##    36.4    63.6
```

```r
round(prop.table(table(dataCustomers.cl3$Gender))*100, digits = 2)
```

```
##
##  Feminino Masculino
##     33.33     66.67
```

```r
round(prop.table(table(dataCustomers.cl3$MaritalStatus))*100, digits = 2)
```

```
##
##   Casado Solteiro    Outro
##    68.77    19.80    11.43
```

```r
round(prop.table(table(dataCustomers.cl3$HasChildren))*100, digits = 2)
```

```
##
##   Sim   Não
## 92.91  7.09
```

```r
round(prop.table(table(dataCustomers.cl3$rfm_score_cat))*100, digits = 2)
```

```
##
##     Frequent      Regular Sporadically
##        22.16        62.25        15.59
```

```r
round(prop.table(table(dataCustomers.cl3$ageInterval))*100, digits = 2)
```

```
##
##  < 50  < 65 >= 65
## 25.73 60.09 14.18
```

Clustering by RFM Score

```r
# Divide customers by its RFM Score
rfm.clusters <- split(dataCustomers, dataCustomers$rfm_score_cat)
```

```r
dataCustomers.rfmFrequent <- rfm.clusters$Frequent
dataCustomers.rfmRegular <- rfm.clusters$Regular
dataCustomers.rfmSporadically <- rfm.clusters$Sporadically
```

Dados das compras

```r
## Tabela ITEM.dat
items <- fread("DATA-CRM/ITEM.dat", quote = "'")

### Verificação dos dados da tabela item, tal como o número de colunas e linhas, bem como se os dados f
summary(items)
```

```
##    ItemCode         ItemDescription    CategoryCode
##  Length:819         Length:819         Length:819
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##  SubCategoryCode     BrandCode         UpmarketFlag
##  Length:819         Length:819         Length:819
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```r
dim(items)
```

```
## [1] 819   6
```

```r
str(items)
```

```
## Classes 'data.table' and 'data.frame':   819 obs. of  6 variables:
##  $ ItemCode       : chr  "I0000000001" "I0000000002" "I0000000003" "I0000000004" ...
##  $ ItemDescription: chr  "BXT - Listen2This1" "BXT - Listen2This2" "BXT - Listen2This3" "ENDOS - ENS
##  $ CategoryCode   : chr  "MACC" "MACC" "MACC" "MACC" ...
##  $ SubCategoryCode: chr  "PMSPE" "PMSPE" "PMSPE" "PMSPE" ...
##  $ BrandCode      : chr  "BBXT" "BBXT" "BBXT" "BENDOS" ...
##  $ UpmarketFlag   : chr  "F" "F" "F" "F" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
#Verificar se a tabela possui dados nulos
table(is.na(items))
```

```
##
## FALSE
##  4914
```

```r
## Tabelas CATEGORY.dat e SUBCATEGORY.dat
categories <- fread("DATA-CRM/CATEGORY.dat", quote = "'")
subcategories <- fread("DATA-CRM/SUBCATEGORY.dat", quote = "'")
```

Join com a tabela de transações + cardID

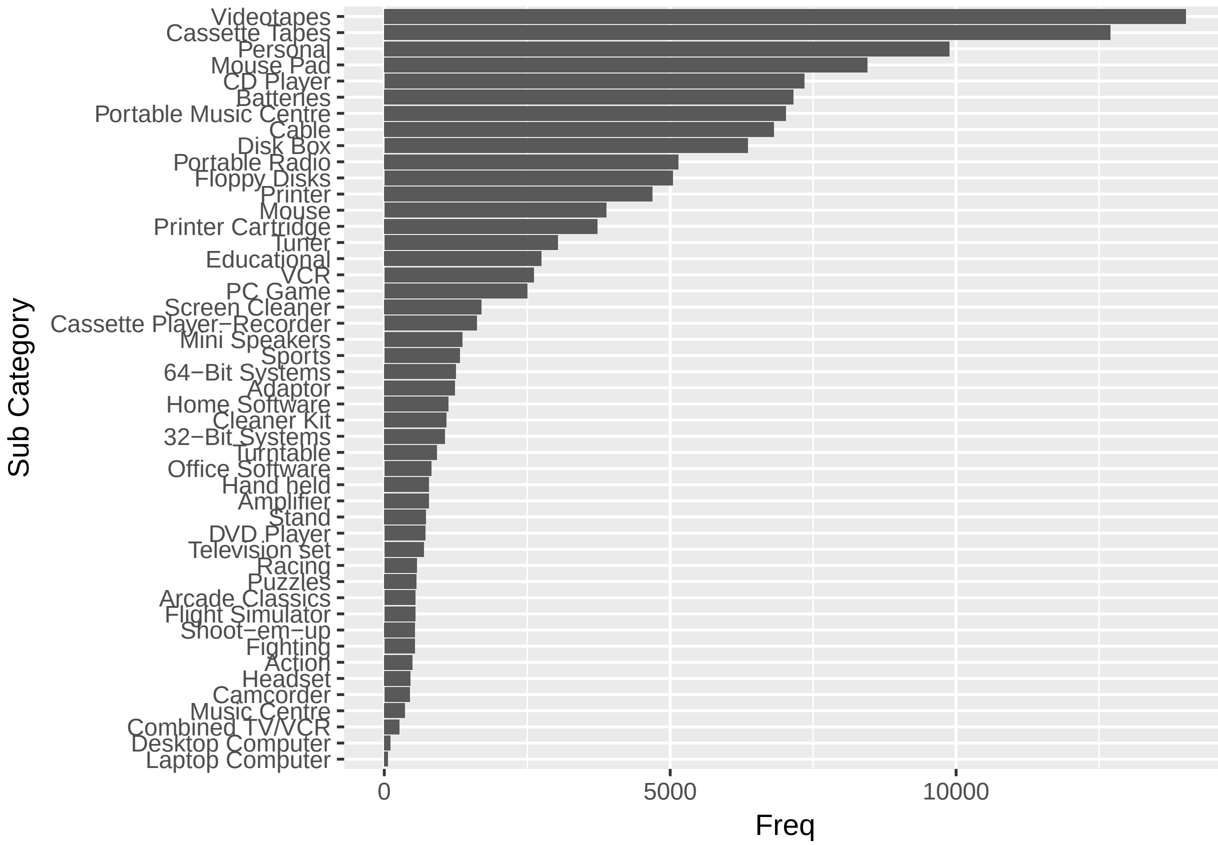
```r
result.aux <- merge(items, categories, all.x = TRUE, by = 'CategoryCode')
result.aux <- merge(result.aux, subcategories, all.x = TRUE, by = 'SubCategoryCode')

result.purchases <- merge(result.transactions, result.aux[ ,c(3:5, 7:8)], all.x = TRUE, by = 'ItemCode')
```

```r
# Se retirados 'ItemNumber' e 'TransactionID' passam a existir observações repetidas
dataPurchases <- result.purchases[, c("CardID", "Date", "PaymentMethod", "Amount", "ItemDescription", "C
```

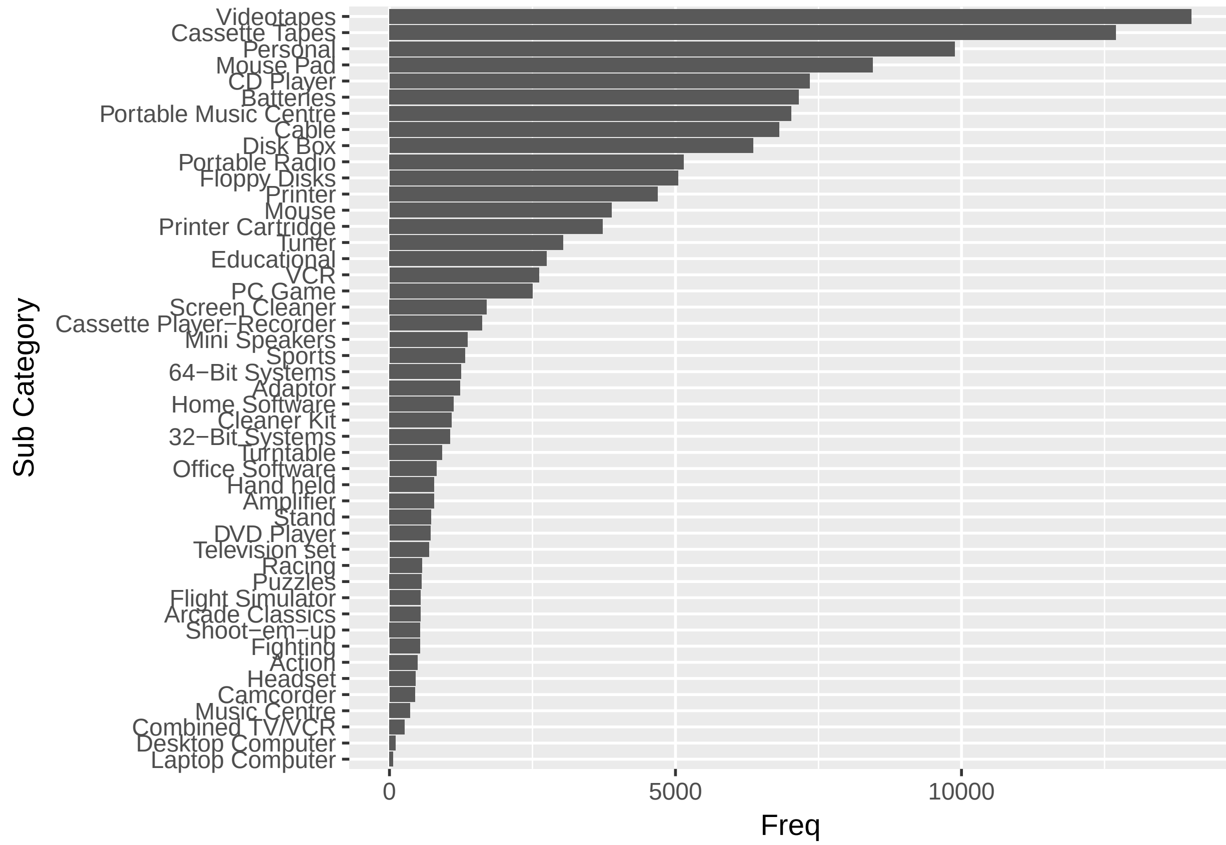
```r
dataPurchases$PaymentMethod <- as.factor(dataPurchases$PaymentMethod)
dataPurchases$ItemDescription <- as.factor(dataPurchases$ItemDescription)
dataPurchases$CategoryDescription <- as.factor(dataPurchases$CategoryDescription)
dataPurchases$SubCategoryDescription <- as.factor(dataPurchases$SubCategoryDescription)
dataPurchases$BrandCode <- as.factor(dataPurchases$BrandCode)


# Split dataPurchases by clusters
dataPurchases.cl1 <- merge(dataPurchases, dataCustomers.cl1[, c("CardID")], by = "CardID")
dataPurchases.cl2 <- merge(dataPurchases, dataCustomers.cl2[, c("CardID")], by = "CardID")
dataPurchases.cl3 <- merge(dataPurchases, dataCustomers.cl3[, c("CardID")], by = "CardID")

# Split dataPurchases by rfm clusters
dataPurchases.rmfFrequent <- merge(dataPurchases, dataCustomers.rfmFrequent[, c("CardID")], by = "CardII
dataPurchases.rfmRegular <- merge(dataPurchases, dataCustomers.rfmRegular[, c("CardID")], by = "CardID")
dataPurchases.rfmSporadically <- merge(dataPurchases, dataCustomers.rfmSporadically[, c("CardID")], by =
```

Vendas por subcategorias de produtos

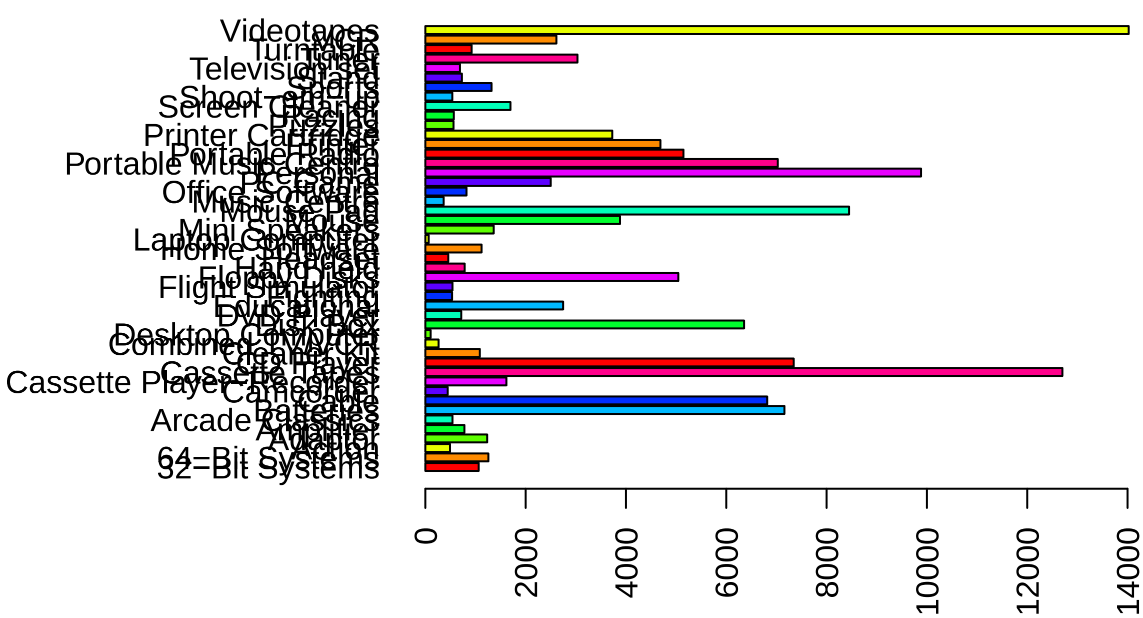
```r
# Frequência das subcategorias de produtos no cluster 1
# ordenado por ordem descrescente
sub_ord <- factor(dataPurchases.cl1$SubCategoryDescription,
                  levels = rev(levels(fct_infreq(dataPurchases.cl1$SubCategoryDescription))))

ggplot(as.data.frame(dataPurchases.cl1$SubCategoryDescription), aes(x = sub_ord)) +
  geom_bar() + labs(x = "Sub Category", y = "Freq") + coord_flip()
```



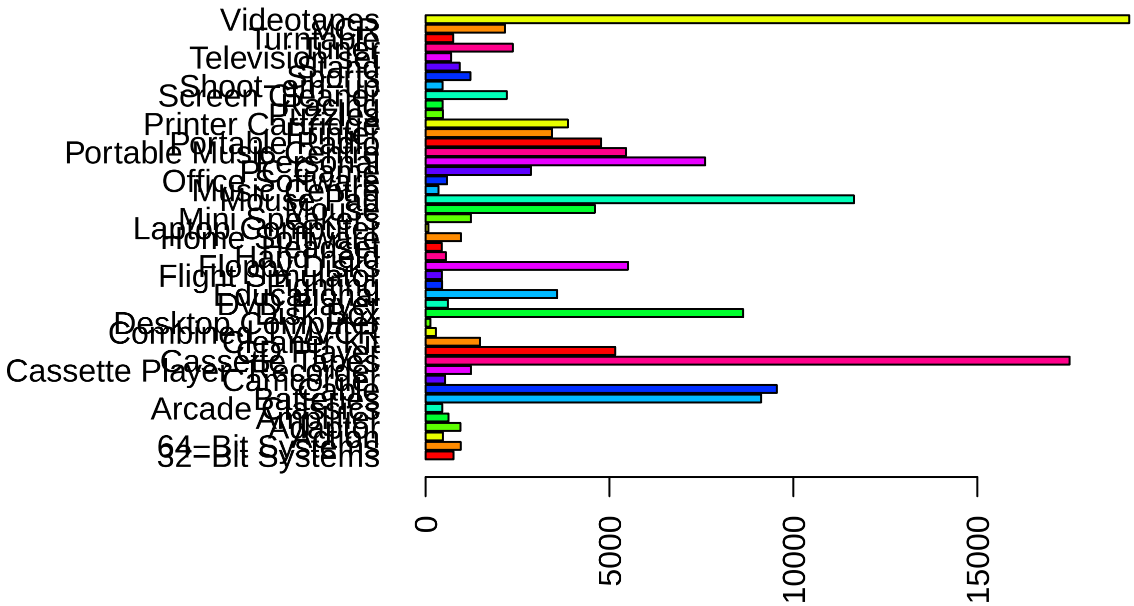

```r
# Alternativamente
x <- as.data.frame(sort(
  table(dataPurchases.cl1$SubCategoryDescription, dnn = c("SubCategory")), decreasing = F))

ggplot(x, aes(x = reorder(SubCategory, Freq), y = Freq)) +
  geom_bar(stat = 'identity') + labs(x = "Sub Category", y = "Freq") + coord_flip()
```



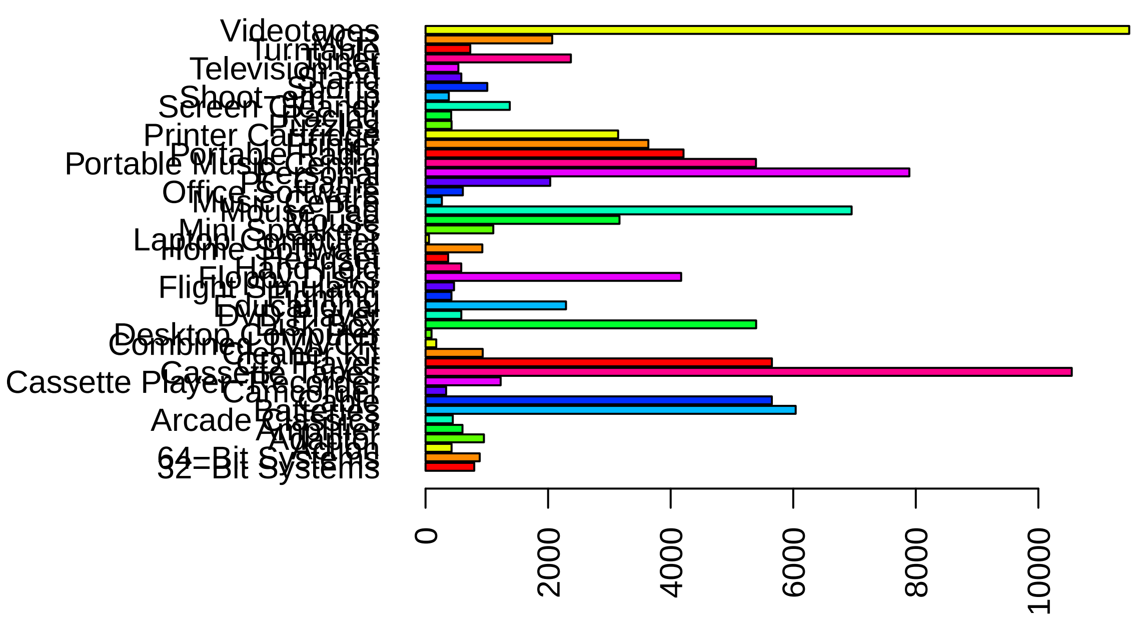

```r
par(las = 2)
par(mar = c(5, 12, 5, 2))

plot(dataPurchases.cl1$SubCategoryDescription, col=rainbow(11), horiz = TRUE)
```



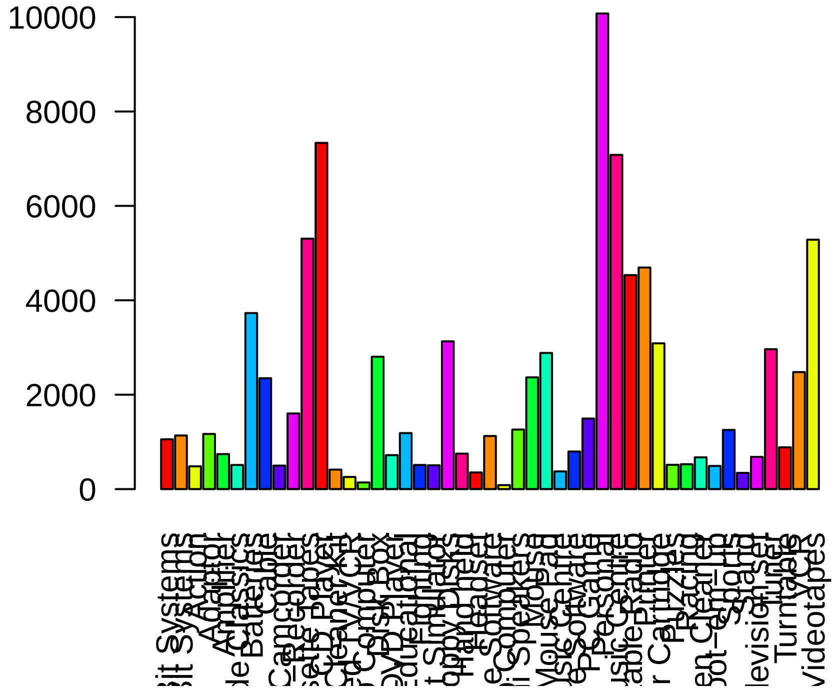

```r
plot(dataPurchases.cl2$SubCategoryDescription, col=rainbow(11), horiz = TRUE)
```



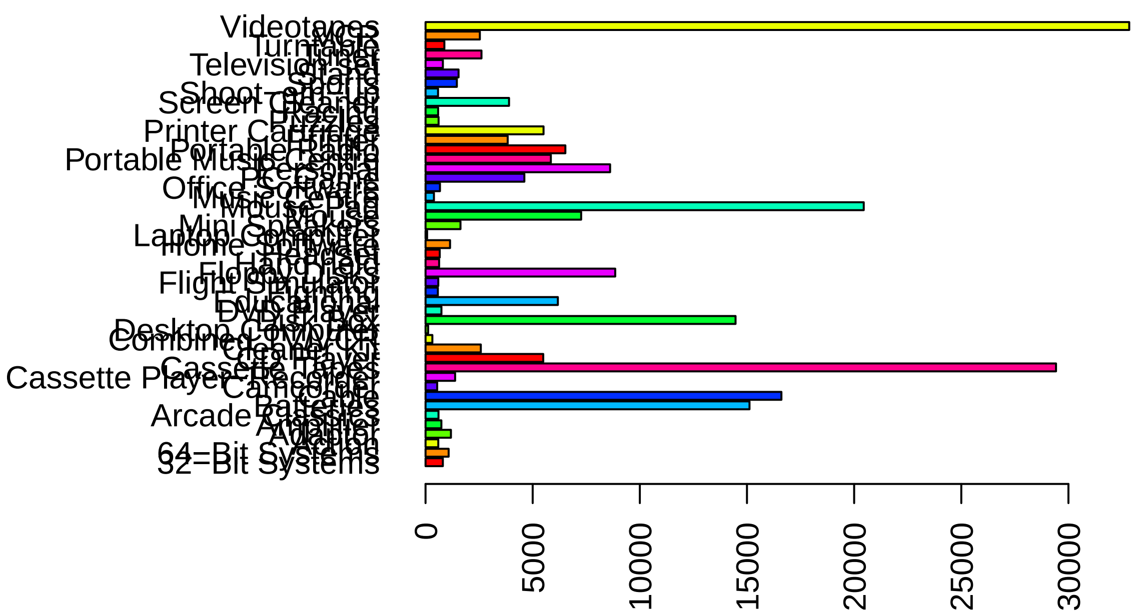

```
plot(dataPurchases.cl3$SubCategoryDescription, col=rainbow(11), horiz = TRUE)
```



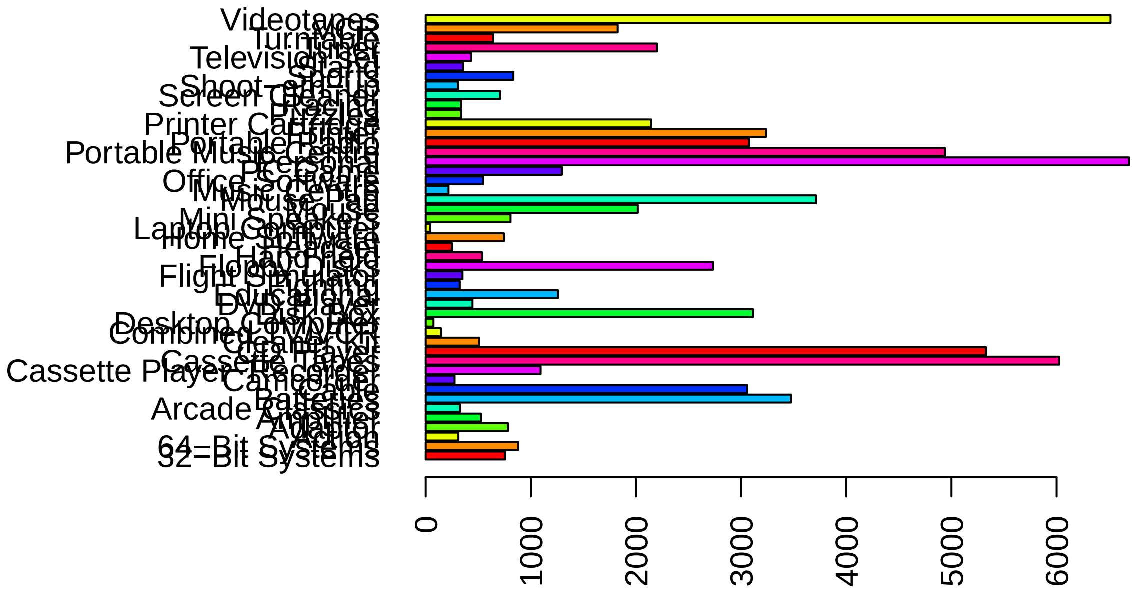

```
plot(dataPurchases.rfmRegular$SubCategoryDescription, col=rainbow(11))
```




```
plot(dataPurchases.rmfFrequent$SubCategoryDescription, col=rainbow(11), horiz = TRUE)
```




```
plot(dataPurchases.rfmSporadically$SubCategoryDescription, col=rainbow(11), horiz = TRUE)
```




```r
# Todas as subcategorias de produtos
levels(dataPurchases$SubCategoryDescription)
```

```
##  [1] "32-Bit Systems"            "64-Bit Systems"
##  [3] "Action"                    "Adaptor"
##  [5] "Amplifier"                 "Arcade Classics"
##  [7] "Batteries"                 "Cable"
##  [9] "Camcorder"                 "Cassette Player-Recorder"
## [11] "Cassette Tapes"            "CD Player"
## [13] "Cleaner Kit"               "Combined TV/VCR"
## [15] "Desktop Computer"          "Disk Box"
## [17] "DVD Player"                "Educational"
## [19] "Fighting"                  "Flight Simulator"
## [21] "Floppy Disks"              "Hand held"
## [23] "Headset"                   "Home Software"
## [25] "Laptop Computer"           "Mini Speakers"
## [27] "Mouse"                     "Mouse Pad"
## [29] "Music Centre"              "Office Software"
## [31] "PC Game"                   "Personal"
## [33] "Portable Music Centre"     "Portable Radio"
## [35] "Printer"                   "Printer Cartridge"
## [37] "Puzzles"                   "Racing"
## [39] "Screen Cleaner"            "Shoot-em-up"
## [41] "Sports"                    "Stand"
## [43] "Television set"            "Tuner"
## [45] "Turntable"                 "VCR"
## [47] "Videotapes"
```

```r
# Número de vendas por subcategorias
baskets.subcat <- count(dataPurchases, c("dataPurchases$SubCategoryDescription"))
baskets.subcat <- baskets.subcat[order(-baskets.subcat$freq), ]
```

```
## Warning: Unknown or uninitialised column: 'freq'.
```

```
## Error in -baskets.subcat$freq: invalid argument to unary operator
```

```r
colnames(baskets.subcat) <- c("subcategory", "freq")

length(unique(baskets.subcat$subcategory))       # 47 subcategorias
```

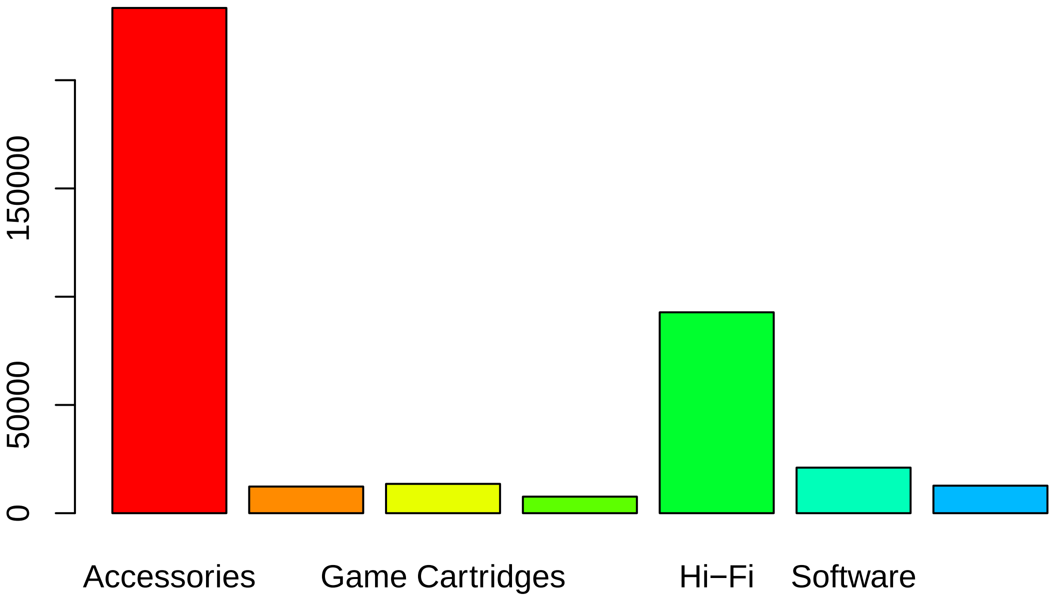
```
## [1] 1
```

```r
# Número médio de itens por basket (subcategoria)
summary(baskets.subcat$freq)
```

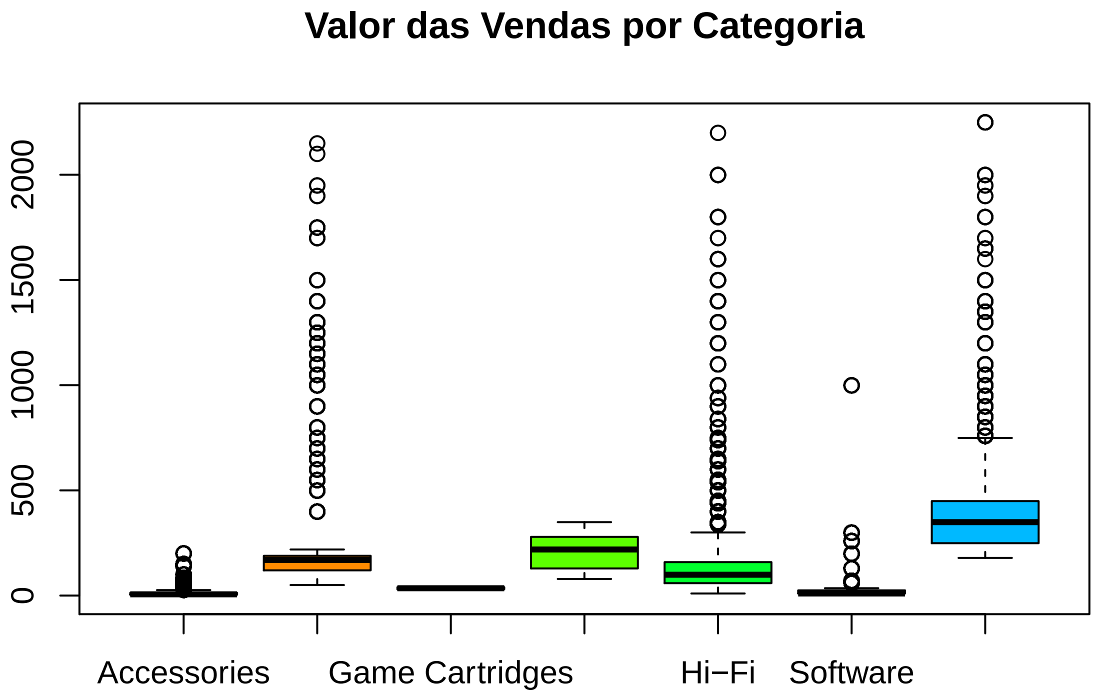
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  393381  393381  393381  393381  393381  393381
```

Vendas por categorias de produtos

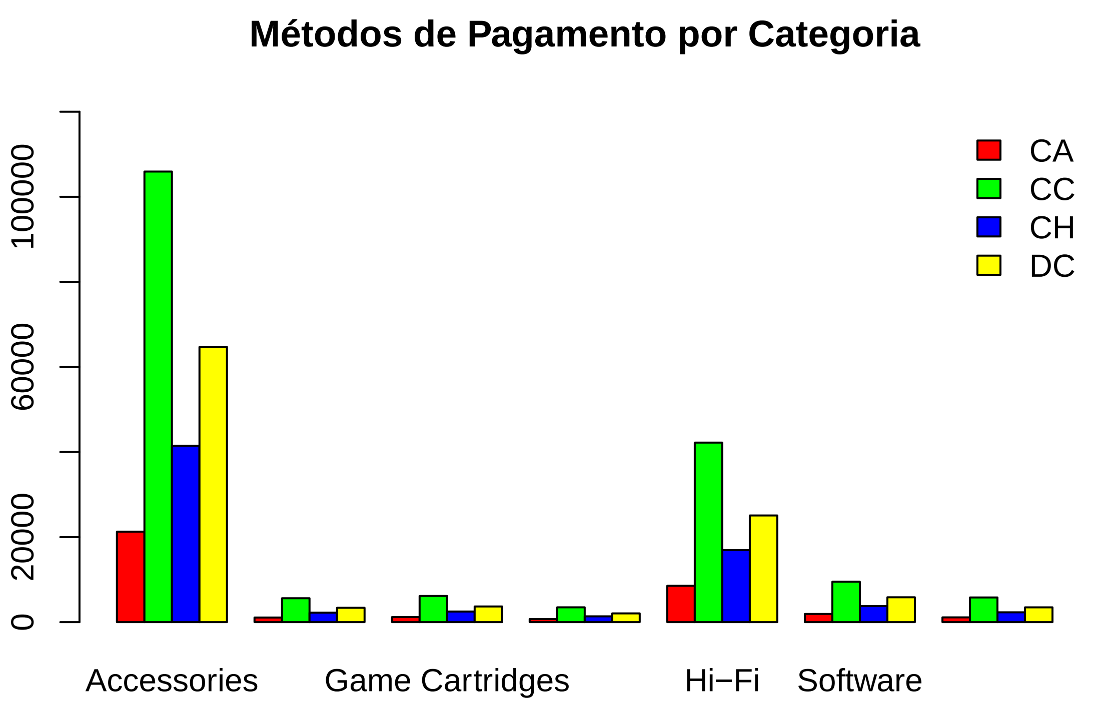
```r
plot(dataPurchases$CategoryDescription, col=rainbow(11))
```




```r
boxplot( Amount ~ CategoryDescription, data = dataPurchases, main = "Valor das Vendas por Categoria", c
```




```r
barplot(table(dataPurchases$PaymentMethod, dataPurchases$CategoryDescription),
        beside = T, col = c("red", "green", "blue", "yellow"),
        main = "Métodos de Pagamento por Categoria", ylim = c(0, 120000))

legend("topright", levels(dataPurchases$PaymentMethod), bty = "n", fill=c("red", "green", "blue", "yell
```




```r
# Todas as categorias de produtos
levels(dataPurchases$CategoryDescription)
```

```
## [1] "Accessories"     "Computers"       "Game Cartridges" "Game Consoles"
```

```
## [5] "Hi-Fi"           "Software"        "TV & Video"
```

```r
# Número de vendas por categorias
baskets.cat <- count(dataPurchases, c("dataPurchases$CategoryDescription"))
baskets.cat <- baskets.cat[order(-baskets.cat$freq), ]
```

```
## Warning: Unknown or uninitialised column: 'freq'.
```

```
## Error in -baskets.cat$freq: invalid argument to unary operator
```

```r
colnames(baskets.cat) <- c("category", "freq")

length(unique(baskets.cat$category))       # 7 categorias
```

```
## [1] 1
```

```r
# Número médio de itens por basket
summary(baskets.cat$freq)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  393381  393381  393381  393381  393381  393381
```

```r
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
basket <- as(split(as.vector(dataPurchases$SubCategoryDescription), as.vector(dataPurchases$CardID)), "
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

```r
class(basket)
```

```
## [1] "transactions"
## attr(,"package")
## [1] "arules"
```

```r
summary(basket)
```

```
## transactions as itemMatrix in sparse format with
##  60519 rows (elements/itemsets/transactions) and
##  47 columns (items) and a density of 0.08919232
##
## most frequent items:
##             Personal             CD Player Portable Music Centre
##                20731                 16191                 15661
##       Cassette Tapes            Videotapes               (Other)
##                13358                 12819                174938
##
## element (itemset/transaction) length distribution:
## sizes
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12
##  8837 14902 14455  9122  3124   938   537   566   558   772   903  1090
##    13    14    15    16    17    18    19    20    21    22
##  1173  1139   960   692   428   208    86    19     7     3
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.192   4.000  22.000
##
## includes extended item information - examples:
##          labels
## 1 32-Bit Systems
## 2 64-Bit Systems
## 3         Action
##
## includes extended transaction information - examples:
##   transactionID
## 1   C0100000111
## 2   C0100000199
## 3   C0100000343
```

```r
dim(basket)
```

```
## [1] 60519    47
```

```r
basket@itemInfo   # gives all the items of the basket
```

```
##                       labels
## 1             32-Bit Systems
## 2             64-Bit Systems
## 3                     Action
## 4                    Adaptor
## 5                  Amplifier
## 6            Arcade Classics
## 7                  Batteries
## 8                      Cable
## 9                  Camcorder
## 10 Cassette Player-Recorder
## 11            Cassette Tapes
## 12                 CD Player
## 13               Cleaner Kit
## 14           Combined TV/VCR
## 15          Desktop Computer
## 16                  Disk Box
## 17                DVD Player
## 18               Educational
## 19                  Fighting
## 20          Flight Simulator
## 21             Floppy Disks
## 22                 Hand held
## 23                   Headset
## 24             Home Software
## 25           Laptop Computer
## 26             Mini Speakers
## 27                     Mouse
## 28                 Mouse Pad
## 29              Music Centre
```

```
## 30          Office Software
## 31                  PC Game
## 32                 Personal
## 33    Portable Music Centre
## 34           Portable Radio
## 35                  Printer
## 36        Printer Cartridge
## 37                  Puzzles
## 38                   Racing
## 39           Screen Cleaner
## 40              Shoot-em-up
## 41                   Sports
## 42                    Stand
## 43           Television set
## 44                    Tuner
## 45                Turntable
## 46                      VCR
## 47               Videotapes
```

```r
#View the first five transactions
inspect(basket[1:5])
```

```
##     items                   transactionID
## [1] {Disk Box,
##      PC Game,
##      Personal,
##      Printer,
##      Tuner,
##      VCR}                     C0100000111
## [2] {Portable Music Centre,
##      VCR}                     C0100000199
## [3] {Personal,
##      Portable Music Centre,
##      Printer,
##      Shoot-em-up,
##      Turntable}               C0100000343
## [4] {Cable,
##      Home Software,
##      Mouse,
##      Portable Radio}          C0100000375
## [5] {Action,
##      Batteries,
##      Cable,
##      Cassette Tapes,
##      Cleaner Kit,
##      DVD Player,
##      Educational,
##      Mouse Pad,
##      Personal,
##      Portable Radio,
##      Sports,
##      Videotapes}              C0100000392
```

```r
# Occurrences of each item - Support
itemFreq <- itemFrequency(basket)
```

```r
sort(itemFreq, decreasing = T)[1:3]
```

```
##                Personal               CD Player Portable Music Centre
##               0.3425536               0.2675358               0.2587782
```

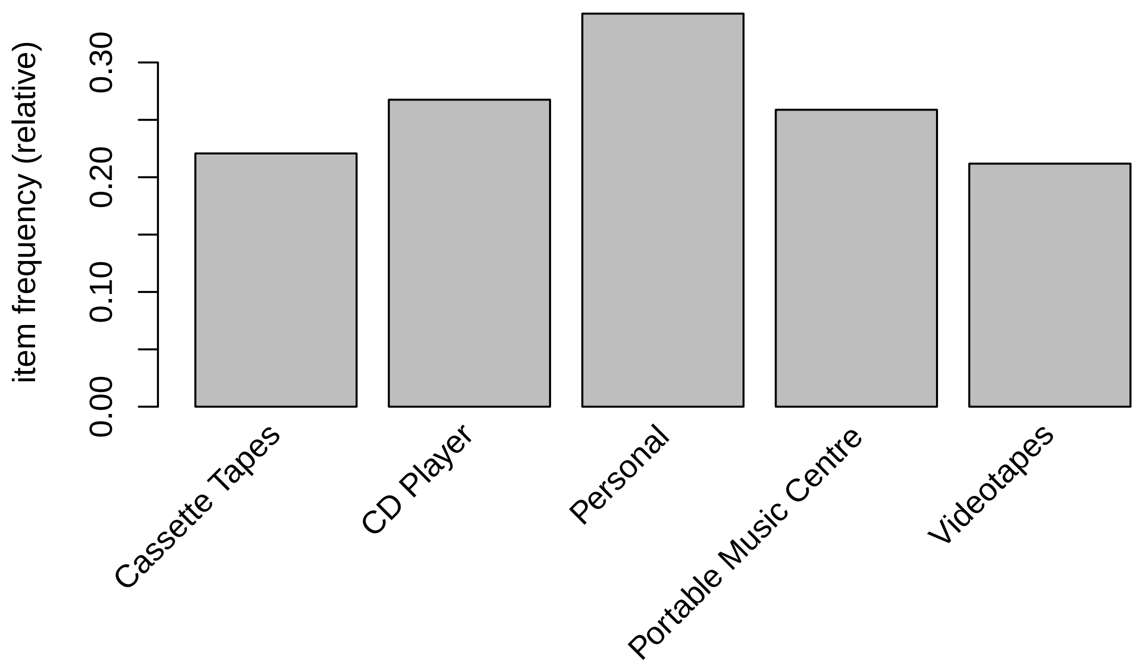
```r
summary(itemFreq)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003239 0.023786 0.050083 0.089192 0.148805 0.342554
```

```r
#View the frequency of the first three items
itemFrequency(basket[, 1:3])
```

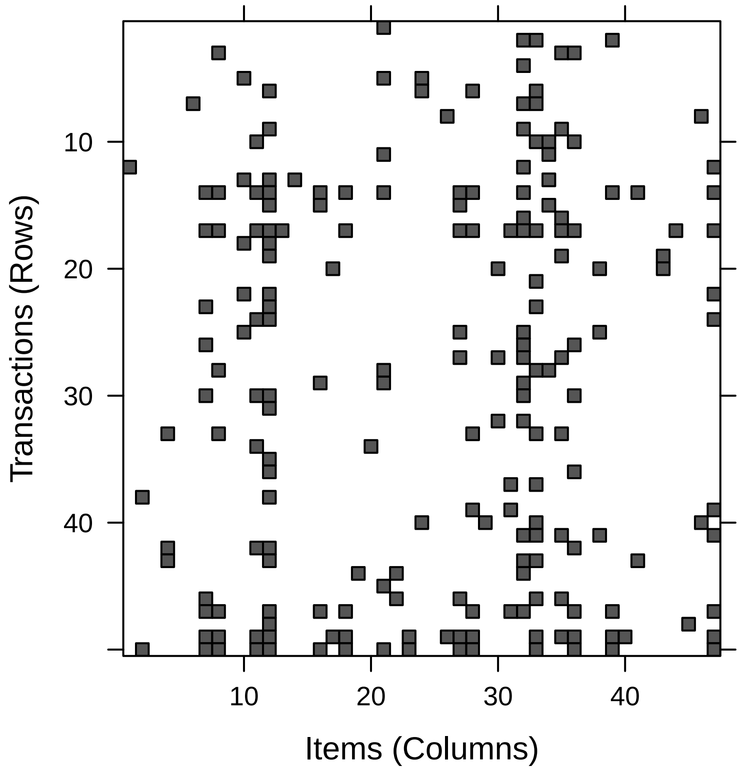
```
## 32-Bit Systems 64-Bit Systems         Action
##     0.04256514     0.05008344     0.02260447
```

```r
#Shows in a histogram plot items with at least s support
with(s <- 0.20,
  itemFrequencyPlot(basket, support = s)
)
```



Visualização da matriz de produtos comprados e respetiva dispersão.

```r
#image(basket[1:50])
image(sample(basket, 50)) # 50 linhas
```



Algoritmo Apriori para extração de Regras de Associação

Sup min = 5% e Conf min = 80%

```
sup.min = 0.05
conf.min = 0.80

basketRules <- apriori(basket, parameter = list(support = sup.min, confidence = conf.min, minlen = 2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5    0.05      2
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 3025
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[47 item(s), 60519 transaction(s)] done [0.02s].
## sorting and recoding items ... [24 item(s)] done [0.00s].
```

```
## creating transaction tree ... done [0.03s].
## checking subsets of size 1 2 3 4 5 6 7 done [0.03s].
## writing ... [919 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```r
summary(basketRules)
```

```
## set of 919 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4   5   6   7
##   6 214 360 241  86  12
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   4.000   4.000   4.243   5.000   7.000
##
## summary of quality measures:
##     support          confidence          lift           count
##  Min.   :0.05002   Min.   :0.8006   Min.   :3.631   Min.   :3027
##  1st Qu.:0.05479   1st Qu.:0.8717   1st Qu.:4.443   1st Qu.:3316
##  Median :0.06094   Median :0.9083   Median :4.647   Median :3688
##  Mean   :0.06586   Mean   :0.9111   Mean   :4.988   Mean   :3986
##  3rd Qu.:0.07072   3rd Qu.:0.9630   3rd Qu.:5.634   3rd Qu.:4280
##  Max.   :0.12221   Max.   :0.9885   Max.   :6.213   Max.   :7396
##
## mining info:
##    data ntransactions support confidence
##  basket         60519    0.05        0.8
```

```r
measures <- interestMeasure(basketRules, measure = c("coverage", "leverage", "conviction"), transaction

summary(measures)
```

```
##     coverage          leverage          conviction
##  Min.   :0.05076   Min.   :0.03859   Min.   : 3.925
##  1st Qu.:0.05985   1st Qu.:0.04346   1st Qu.: 6.371
##  Median :0.06682   Median :0.04867   Median : 9.174
##  Mean   :0.07266   Mean   :0.05241   Mean   :15.584
##  3rd Qu.:0.07909   3rd Qu.:0.05664   3rd Qu.:21.132
##  Max.   :0.14843   Max.   :0.09633   Max.   :68.572
```

```r
# Top rules by lift
inspect(head(basketRules, n = 5, by = "lift"))
```

```
##     lhs                  rhs        support confidence     lift count
## [1] {Batteries,
##      Cassette Tapes,
##      Disk Box,
##      Floppy Disks,
##      Mouse Pad,
##      Videotapes}     => {Cable} 0.05684165  0.9222520 6.213266  3440
## [2] {Batteries,
##      Disk Box,
##      Floppy Disks,
##      Mouse Pad,
##      Videotapes}     => {Cable} 0.05784960  0.9196217 6.195546  3501
```

```
## [3] {Batteries,
##      Cassette Tapes,
##      Disk Box,
##      Mouse,
##      Mouse Pad,
##      Videotapes}     => {Cable} 0.05109139  0.9194172 6.194168  3092
## [4] {Batteries,
##      Cassette Tapes,
##      Disk Box,
##      Floppy Disks,
##      Mouse Pad}      => {Cable} 0.05750260  0.9182058 6.186007  3480
## [5] {Batteries,
##      Cassette Tapes,
##      Disk Box,
##      Mouse,
##      Mouse Pad}      => {Cable} 0.05176887  0.9163498 6.173503  3133
```

```r
library(arulesViz)
```

```
## Loading required package: grid
```

```r
basketRules2 <- apriori(basket, parameter = list(support = 0.01, confidence = 0.05, minlen = 2, maxlen =
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.05    0.1    1 none FALSE            TRUE       5    0.01      2
##  maxlen target   ext
##      20  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 605
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[47 item(s), 60519 transaction(s)] done [0.02s].
## sorting and recoding items ... [45 item(s)] done [0.00s].
## creating transaction tree ... done [0.03s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.22s].
## writing ... [84749 rule(s)] done [0.01s].
## creating S4 object  ... done [0.03s].
```

```r
summary(basketRules2)
```

```
## set of 84749 rules
##
## rule length distribution (lhs + rhs):sizes
##     2     3     4     5     6     7     8     9    10
##   576  3174 10104 19445 23712 17815  7872  1881   170
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   5.000   6.000   5.863   7.000  10.000
##
```

```
## summary of quality measures:
##     support           confidence           lift             count      
##  Min.   :0.01001   Min.   :0.05021   Min.   :0.7267   Min.   : 606  
##  1st Qu.:0.01148   1st Qu.:0.51873   1st Qu.:4.3112   1st Qu.: 695  
##  Median :0.01413   Median :0.83846   Median :4.6837   Median : 855  
##  Mean   :0.01734   Mean   :0.73614   Mean   :4.6556   Mean   :1049  
##  3rd Qu.:0.01928   3rd Qu.:0.94124   3rd Qu.:5.8402   3rd Qu.:1167  
##  Max.   :0.14377   Max.   :1.00000   Max.   :8.0311   Max.   :8701  
## 
## mining info:
##    data ntransactions support confidence
##  basket         60519    0.01       0.05
```