# Format d´entrega

Aquest document s´ha realitzat mitjançant **Markdown**[[1]](#footnote-22) amb l´ajuda de l´entorn de desenvolupament **RStudio**[[2]](#footnote-24) utilitzant les característiques que aquest ofereix per a la creació de documents **R** reproduïbles.

La documentació generada en la realització de la pràctica es troba allotjada en **GitHub** al següent repositori:

* <https://github.com/rsanchezs/data-minig>

En aquest repositori es poden trobar els següents fitxers:

* Aquest document en formats **pdf** i **docx** amb el nom rsanchezs\_practica.
* Un document **R Markdown**[[3]](#footnote-27) que es pot utilitzar per a reproduir tots els exemples presentats a la PAC.
* El conjunt de dades utilitzades.

**Nota:** Propietat intel·lectual

Sovint és inevitable, al produir una obra multimèdia, fer ús de recursos creats per terceres persones. És per tant comprensible fer-lo en el marc d’una pràctica dels Estudis, sempre que això es documenti clarament i no suposi plagi en la pràctica.

Per tant, al presentar una pràctica que faci ús de recursos aliens, s’ha de presentar juntament amb ella un document en que es detallin tots ells, especificant el nom de cada recurs, el seu autor, el lloc on es va obtenir i el seu estatus legal: si l’obra està protegida pel copyright o s’acull a alguna altra llicència d’ús (Creative Commons, llicència GNU, GPL …). L’estudiant haurà d’assegurar-se que la llicència no impedeix específicament el seu ús en el marc de la pràctica. En cas de no trobar la informació corresponent haurà d’assumir que l’obra està protegida per copyright. Hauríeu a més, d’adjuntar els fitxers originals quan les obres utilitzades siguin digitals, i el seu codi font si correspon

Un altre punt a considerar és que qualsevol pràctica que faci ús de recursos protegits pel copyright no podrà en cap cas publicar-se en Mosaic, la revista del Graduat en Multimèdia de la UOC, llevat que els propietaris dels drets intel·lectuals donin la seva autorització explícita

# PART I Preparació de les Dades

## Definició de la tasca de mineria de dades

Aquesta pràctica tracta de plantejar com podria ser un projecte real de mineria de dades. Com a analistes de dades a partir de la presentació del client que exposa un problema de negoci difús i molt genèric haurem de reconduir-lo com a projecte de mineria de dades.

El client ens proporciona un conjunt de dades extretes del seu ERP, format per les següent taules: cabeceraticket, client, familia, lineasticket, pais, pedido, producto, promocion, proveedor, regiongeografica, seccion, subfamilia, tienda.

Els objectius principals del projecte de mineria de dades seran els següents:

En primer lloc, com què tenim poca informació del domini i volem començar a tenir-ne una idea més clara, intentarem **trobar similituds i agrupar objectes semblants**.

En segon lloc, a partir de la situació més informada obtinguda en el pas anterior, tractarem de **classificar els objectes**. El que es vol és estudiar millor les diferències entre grups i les seves característiques peculiars.

## **PRIMER OBJECTIU**

Trobar grups de clients semblants.

**SEGON OBJECTIU**

Un cop separats els clients en diversos grups, volem saber quin és l´atribut que distingueix millor un grup de clients o l´altre.

## Pre-processament de les Dades

### Carrega i exàmen preliminar del conjunt de dades

En primer lloc, instal.larem el paquet readr[[4]](#footnote-34) que forma part del ecosistema tidyverse[[5]](#footnote-36) i que ens permetrà llegir les dades:

# La forma més sencilla de instal.lar readr es instal.lar tidyverse  
install.packages("tidyverse")  
  
# Alternativament, podem instal.lar només readr  
install.packages("readr")

Un cop instal.lat el paquet el carregarem a la sessió R mitjançant la següent línia de codi:

# Carrega de readr  
library(readr)  
  
# Alternativament, com que forma part de tidyverse  
library(tidyverse)

Observem que, hem fet ús de la segona opció que carrega tots els paquets de tidyverse, ja que utilitzarem per a la realització de la pràctica altres paquets, com per exemple: dplyr (per a la transformació de dades),tibble (per a un tractament més refinat de data.frames), ggplot2 (per a la visualització de les dades), etc.

Un cop carregat el paquet a la sessió R, ja podem fer ús de les funcions. Per a importar les dades dels clients i els seus ticket de compra utilitzarem la funció read\_csv() de la següent manera:

# Carreguem la llibreria que ens permet importar arxius CSV  
if (!require("readr")) {  
 # Instal.lació de la llibreria  
install.packages("readr")  
# Carreguem la llibreria   
library(readr)  
}  
client <- read\_csv("data/gourmetdb/cliente.csv",   
 col\_names = FALSE)  
ticket <- library(readr)  
ticket <- read\_csv("data/gourmetdb/cabeceraticket.cvs",   
 col\_names = FALSE)

Convertim el conjunt de dades client que és del tipus data.frame a tibble:

# Convertim el dataframe a tibble  
as\_tibble(client)

## # A tibble: 4,069 x 12  
## X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12  
## <chr> <chr> <chr> <int> <chr> <chr> <chr> <int> <chr> <ch> <int> <int>  
## 1 0000~ Roca~ Homb~ 1.96e7 Solt~ Piaz~ Econ~ 0 Sur ~ Esp~ 4 7  
## 2 0065~ Fuen~ Mujer 1.94e7 Casa~ C/ N~ Inge~ 1 Sur ~ Esp~ 16 13  
## 3 0065~ Prat~ Homb~ 1.94e7 Casa~ cors~ Doct~ 2 Sur ~ Esp~ 14 10  
## 4 0000~ Jone~ Homb~ 1.91e7 Solt~ 1 Pl~ Inge~ 0 Nort~ Rei~ 2 9  
## 5 0000~ Burt~ Homb~ 1.94e7 Casa~ 46 S~ Doct~ 2 Nort~ Rei~ 13 9  
## 6 0065~ Sale~ Mujer 1.94e7 Casa~ Leop~ Econ~ 1 Nort~ Rei~ 7 11  
## 7 0065~ Crui~ Homb~ 1.96e7 Solt~ 2 Re~ Inge~ 0 Nort~ Rei~ 10 12  
## 8 0131~ Cole~ Homb~ 1.94e7 Casa~ 67 E~ Doct~ 0 Nort~ Est~ 2 6  
## 9 0131~ Shav~ Mujer 1.96e7 Casa~ 432 ~ Econ~ 3 Nort~ Est~ 21 15  
## 10 0196~ Mill~ Homb~ 1.94e7 Divo~ 68 A~ Econ~ 0 Nort~ Rei~ 5 9  
## # ... with 4,059 more rows

Podem adonar-nos que, el conjunt de dades està format per 4.069 observacions i 12 variables. A més, amb l´ajuda de tibble també podem observar el tipus per a cada columna.

Com que el nom de les columnes es poc descriptiu per alguns dels atributs, personalitzarem els noms mitjançant la següent línia de codi:

# Noms dels atributs  
names(client) <- c("CODCLIENT", "NOMCLIENT", "GENERE", "DATANAIXEMENT", "ESTATCIVIL",   
 "DIRECCIO", "PROFESSIO", "NROFILLS", "REGIO",  
 "NACIONALITAT","TOTALCOMPRES","PUNTSACUMULATS")  
names(ticket) <- c("CODVENTA", "NOMTENDA", "DATA", "HORA", "FORMAPAGAMENT",   
 "CODCLIENT", "TOTALIMPORT", "TOTALUNITATS", "PUNTSTICKET")

Podem comprovar el nom de les columnes mitjançant la funció colnames:

# Comprovem es nom de les columnes  
colnames(client)

## [1] "CODCLIENT" "NOMCLIENT" "GENERE" "DATANAIXEMENT"   
## [5] "ESTATCIVIL" "DIRECCIO" "PROFESSIO" "NROFILLS"   
## [9] "REGIO" "NACIONALITAT" "TOTALCOMPRES" "PUNTSACUMULATS"

# Comprovem es nom de les columnes  
colnames(ticket)

## [1] "CODVENTA" "NOMTENDA" "DATA" "HORA"   
## [5] "FORMAPAGAMENT" "CODCLIENT" "TOTALIMPORT" "TOTALUNITATS"   
## [9] "PUNTSTICKET"

### Exploració i tractament de valors desconeguts

Per altra banda, ens caldria comprovar que el nostre conjunt de dades no conté valors desconeguts. En primer lloc comprovem el conjunt de dades client:

# Estadístiques de valors buits client  
colSums(is.na(client))

## CODCLIENT NOMCLIENT GENERE DATANAIXEMENT ESTATCIVIL   
## 0 0 0 0 805   
## DIRECCIO PROFESSIO NROFILLS REGIO NACIONALITAT   
## 0 0 805 0 0   
## TOTALCOMPRES PUNTSACUMULATS   
## 0 0

Com es pot observar la variable estatcivil conté 805 observacions amb valors desconeguts. Amb l´objectiu de fer aquest grup més descriptiu podríem canviar aquests valors per la constant Desconegut:

# Amb l´ajuda de un test lògic descobrim els valors desconeguts  
missing\_values\_estat\_civil <- is.na(client$ESTATCIVIL)  
# Reemplacem els valors desconeguts amb la constant  
client$ESTATCIVIL[missing\_values\_estat\_civil] <- "Desconegut"

A més, fixe-mos que la variable nombrefills també conté 805 observacions amb valors desconeguts. En aquest cas, reemplaçarem els valor desconeguts amb un valor aleatori de la distribució de la variable. Em primer lloc, amb l´ajuda d´un test lógic descobrim els valors desconeguts:

# Amb l´ajuda de un test lògic descobrim els valors desconeguts  
missing\_values\_nombrefills <- is.na(client$NROFILLS)

A continuació, generem un valor aleatori de la distribució de la variable:

# Generem observacions aleàtories  
random\_nombrefills\_obs <- sample(na.omit(client$NROFILLS), 1)  
random\_nombrefills\_obs

## [1] 2

Finalment, reemplacem els valors desconeguts amb el valor aleatori calculat en en fragment de codi anterior:

# Reemplacem els valors desconeguts amb la observació aleàtoria  
  
client$NROFILLS[missing\_values\_nombrefills] <-random\_nombrefills\_obs

Pel que fa al conjunt de dades ticket realitzarem les mateixes tasques que hem realitzat amb els clients. Així començarem comprovant que no tinguem dades desconegudes:

# Estadístiques de valors buits tickets  
colSums(is.na(ticket))

## CODVENTA NOMTENDA DATA HORA FORMAPAGAMENT   
## 0 0 0 0 0   
## CODCLIENT TOTALIMPORT TOTALUNITATS PUNTSTICKET   
## 38810 0 0 0

Com es pot comprobar a l´anterior fragment de codi el conjunt de dades ticket no conté cap valor buit o desconegut per a cap de les seus variables.

### Transformació d´atributs

L´objectiu principal d´aquest apartat es realitzar tasques de transformació en les variables del nostre conjunt de dades.

Per a facilitar l´ànalisi sería convenient canviar els atributs de tipus character a factor, que és la manera que té R de tractar amb les variables de tipus categòric:

# Carreguem ecosistema tidyverse  
if (!require("tidyverse")) {  
 # Instal.lació de la llibreria  
install.packages("tidyverse")  
# Carreguem la llibreria   
library(tidyverse)  
}  
# Canviem les variables de tipus `character` a `factor`  
cols <- c('CODCLIENT', 'NOMCLIENT', 'GENERE', 'ESTATCIVIL', 'DIRECCIO',  
 'PROFESSIO', 'REGIO', 'NACIONALITAT')  
client <- mutate\_at(client, cols, as.factor)

De la mateixa manera, ens caldrà fer igual amb les variables de tipus charater en el conjunt de dades ticket:

# Canviem les variables de tipus `character` a `factor`  
cols <- c('CODVENTA', 'NOMTENDA', 'FORMAPAGAMENT', 'CODCLIENT')  
ticket <- mutate\_at(ticket, cols, as.factor)

Fixe-mos amb el codi anterior que amb l´ajuda de la funció dplyr::mutate\_at[[6]](#footnote-40) hem canviat les columnes de tipus character al tipus factor.

Amb el següent fragment de codi i amb l´ajuda de la funció lapply() verifiquem que s´han produït els canvis. Per exemple, comprovem-ho amb el conjunt de dades client:

# Retorna el tipus de cada variable  
lapply(client, class)

## $CODCLIENT  
## [1] "factor"  
##   
## $NOMCLIENT  
## [1] "factor"  
##   
## $GENERE  
## [1] "factor"  
##   
## $DATANAIXEMENT  
## [1] "integer"  
##   
## $ESTATCIVIL  
## [1] "factor"  
##   
## $DIRECCIO  
## [1] "factor"  
##   
## $PROFESSIO  
## [1] "factor"  
##   
## $NROFILLS  
## [1] "integer"  
##   
## $REGIO  
## [1] "factor"  
##   
## $NACIONALITAT  
## [1] "factor"  
##   
## $TOTALCOMPRES  
## [1] "integer"  
##   
## $PUNTSACUMULATS  
## [1] "integer"

Cal fer esment específic que sería pràctic convertir la variable naixement del tipus numeric a date:

# Carreguem lubridate per al tractament de dades de tipus date  
if (!require("lubridate")) {  
 # Instal.lació de la llibreria  
install.packages("lubridate")  
# Carreguem la llibreria   
library(lubridate)  
}  
# Convertim la variable naixement a tipus date  
client <- client %>% mutate\_at("DATANAIXEMENT", funs(ymd))

Gràcies a l´anterior canvi de tipus podem calcular l´edat del client de la següent manera:

# Carreguem ecosistema tidyverse  
if (!require("tidyverse")) {  
 # Instal.lació de la llibreria  
install.packages("tidyverse")  
# Carreguem la llibreria   
library(tidyverse)  
}  
client <- client %>%   
 mutate(EDAT = year(Sys.Date()) - year(client$DATANAIXEMENT))

Realitzem un resúm estadístic de la variable EDAT:

# Resúm estadístic  
summary(client$EDAT)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 40.00 52.00 68.00 67.37 80.00 109.00

Fixe-mos que el rang està comprés entre 40 i 109 anys, sembla que la edat estigui mal calculada. Si analitzem com hem calculat la edat anteriorment podem comprovar que els calculs són correctes, ja que hem realitzat la diferencia de la data actual amb la data de naixement.

En conseqüència, ens fa pensar que el conjunt de dades de la base de dades GourmetBD està format per valors pretèrits. Per aquest motiu, anem a calcular de nou la variable EDAT però en aquest cas utilitzamen com a any en curs el 1992:

# Calculem la edat client utilitzant com a any en curs 1992  
client <- client %>%   
 mutate(EDAT = 1992 - year(client$DATANAIXEMENT))

Concretament, després de realitzar els nous calculs podem comprovar que el rang està entre 13 i 82:

range(client$EDAT)

## [1] 13 82

En un altre ordre de coses, crearem un nou conjunt de dades que es tractara de la consulta **left-join** de les taules client i ticket que estan relacionades a la fase de dades per la clau forana CODCLIENT:

# Selecciona les observacions que apareixen almenys en una de les taules  
# cpnservant totes les observacions de ticket  
tickets\_client <- left\_join(client, ticket, by = "CODCLIENT")

A continuació, crearem una nova columna amb la suma dels imports totals dels tickets per a cada client. Es a dir, agruparem les observaciones per client i calcularem una nova variable amg la funció d´agregació sum():

# Agrupem per client i realitzem la suma del import de cada ticket  
import\_total\_tickets\_client <- tickets\_client %>%   
 group\_by(CODCLIENT) %>%   
 summarise(TOTAL = sum(TOTALIMPORT))

Al següent fragment de codi afegim la variable creada anteriorment al conjunt de dades tickets\_client:

tickets\_client <- tickets\_client %>% select(-CODVENTA, -DATA,   
 -HORA, -FORMAPAGAMENT,  
 -TOTALIMPORT, -TOTALUNITATS,  
 -PUNTSTICKET) %>%   
 group\_by(CODCLIENT) %>%   
 na.omit(.) %>%   
 distinct(.) %>%   
 left\_join(import\_total\_tickets\_client,  
 by = "CODCLIENT")

Com hem anteriorment sería convenient canviar els atributs de tipus character a factor, que és la manera que té R de tractar amb les variables de tipus categòric:

# Carreguem ecosistema tidyverse  
if (!require("tidyverse")) {  
 # Instal.lació de la llibreria  
install.packages("tidyverse")  
# Carreguem la llibreria   
library(tidyverse)  
}  
# Canviem les variables de tipus `character` a `factor`  
cols <- c('CODCLIENT', 'NOMCLIENT', 'GENERE', 'ESTATCIVIL', 'DIRECCIO',  
 'PROFESSIO', 'REGIO', 'NACIONALITAT')  
client <- mutate\_at(client, cols, as.factor)

### Reducció de la Dimensionalitat

Per a la simplificació de l´anàlisi les següents variables són descartades:

# Reducció de la dimensionalitat  
#tickets\_client$CODCLIENT <- NULL  
tickets\_client$NOMCLIENT <- NULL  
tickets\_client$DIRECCIO <- NULL  
tickets\_client$DATANAIXEMENT <- NULL

Els motius són els següents:

* La variable codi representa la clau primaria en la base de dades i no ens proporcionará informació per a estudiar millor les diferències entre grups.
* La variable nom pot ser el.liminada pel fet que, es tracta del nom del individu i que té funcions de particularització o individualització i no ens serveix per a les tasques d´agrupació i classificació.
* La variable direccio es tracta de la direcció del domicili del client i no ens proporciona informació rellevant per a la nostra anàlisi. En canvi, les variables regio i nacionalitat són atributs més adequats per al nostre proposit.
* La variable naixement pot ser el.liminada, ja que representa la data de naixement del client i no proporciona informació per al nostre model. No obstant, la edat del client es una característica peculiar i es pot estimar a partir de la data de naixement i la data actual. Aquest atribut ja ha sigut calculat i representat amb la variable edat.

Recollint tot el que s´ha realitzat, tenim 11 atributs per a la nostra anàlisi i 2923 observacions en el conjunt de dades client:

# Obtenim el nom de les variables  
colnames(tickets\_client)

## [1] "CODCLIENT" "GENERE" "ESTATCIVIL" "PROFESSIO"   
## [5] "NROFILLS" "REGIO" "NACIONALITAT" "TOTALCOMPRES"   
## [9] "PUNTSACUMULATS" "EDAT" "NOMTENDA" "TOTAL"

# Obtenim la dimensió  
dim(tickets\_client)

## [1] 3923 12

### Identificació de *outliers*

### Transformació de les Dades

Per a la *normalització* del conjunt de dades tickets\_client utilitzarem el **mètode d´estandardització de valors** assegurant-nos que s´obtenen valors dins el rang escollit que tenen la propietat que la seva mitjana és zero i la seva desviacióestàndard val 1.

Es a dir, l´estandardització consisteix en la diferència entre el valor de l´atribut i la seva mitjana, dividint aquesta diferència per la desviació estàndard dels valors de l´atribut. Es a dir:

En el següent fragment discretitzarem les variables NROFILLS, PUNTSACUMULATS, EDAT i TOTALestandaritzant-les amb l´ajuda de la funció scale():

tickets\_client <- tickets\_client %>% mutate\_at(vars(NROFILLS, TOTALCOMPRES  
 PUNTSACUMULATS, EDAT  
 TOTAL),   
 funs(scale(.)))

A banda d´això, per a facilitar el nostre anàlisi i en concret l´anàlisi exploratori de les dades, realitzarem una segmentació dels nostres clients respecte la freqüència de compra.

Així, realitzarem una discretització utilitzant l´atribut TOTALCOMPRES que pren els valors entre 1 i 48 dividint els valors entre els conjunts: **FREQUENTE**, **HABITUAL** i **OCASIONAL**.

En primer lloc, hem de calcular la distribució de freqüencies absolutes de la variable TOTALCOMPRES:

total\_compres <- tickets\_client$TOTALCOMPRES  
breaks <- seq(1, 48, by = 5)  
total\_compres\_cut <- cut(total\_compres, breaks, by = 5, right = FALSE)  
total\_compres\_freq <- table(total\_compres\_cut)

Amb la funció nrow() podem trobar la freqüencia total del conjunt de dades, així podem realitzar el cocient de les freqüencies absolutes per . De manera que, la **distribució de freqüencies relatives** es:

# Càlcul distribució freqüencies relatives total compres per client  
total\_compres\_relfreq <- total\_compres\_freq / nrow(tickets\_client)  
cbind(total\_compres\_freq, total\_compres\_relfreq)

## total\_compres\_freq total\_compres\_relfreq  
## [1,6) 1183 0.301554932  
## [6,11) 1470 0.374713230  
## [11,16) 786 0.200356870  
## [16,21) 300 0.076472088  
## [21,26) 105 0.026765231  
## [26,31) 49 0.012490441  
## [31,36) 20 0.005098139  
## [36,41) 7 0.001784349  
## [41,46) 1 0.000254907

Finalment, creem una nova variable que classificarà els nostres clients segons la freqüencia de compra:

tickets\_client <- tickets\_client %>%   
 mutate(FREQCOMPRA = case\_when(between(TOTALCOMPRES, 1, 5) ~ "OCASIONAL",  
 between(TOTALCOMPRES, 6, 21) ~ "FRECUENTE",  
 between(TOTALCOMPRES, 22, 48) ~ "HABITUAL" ))

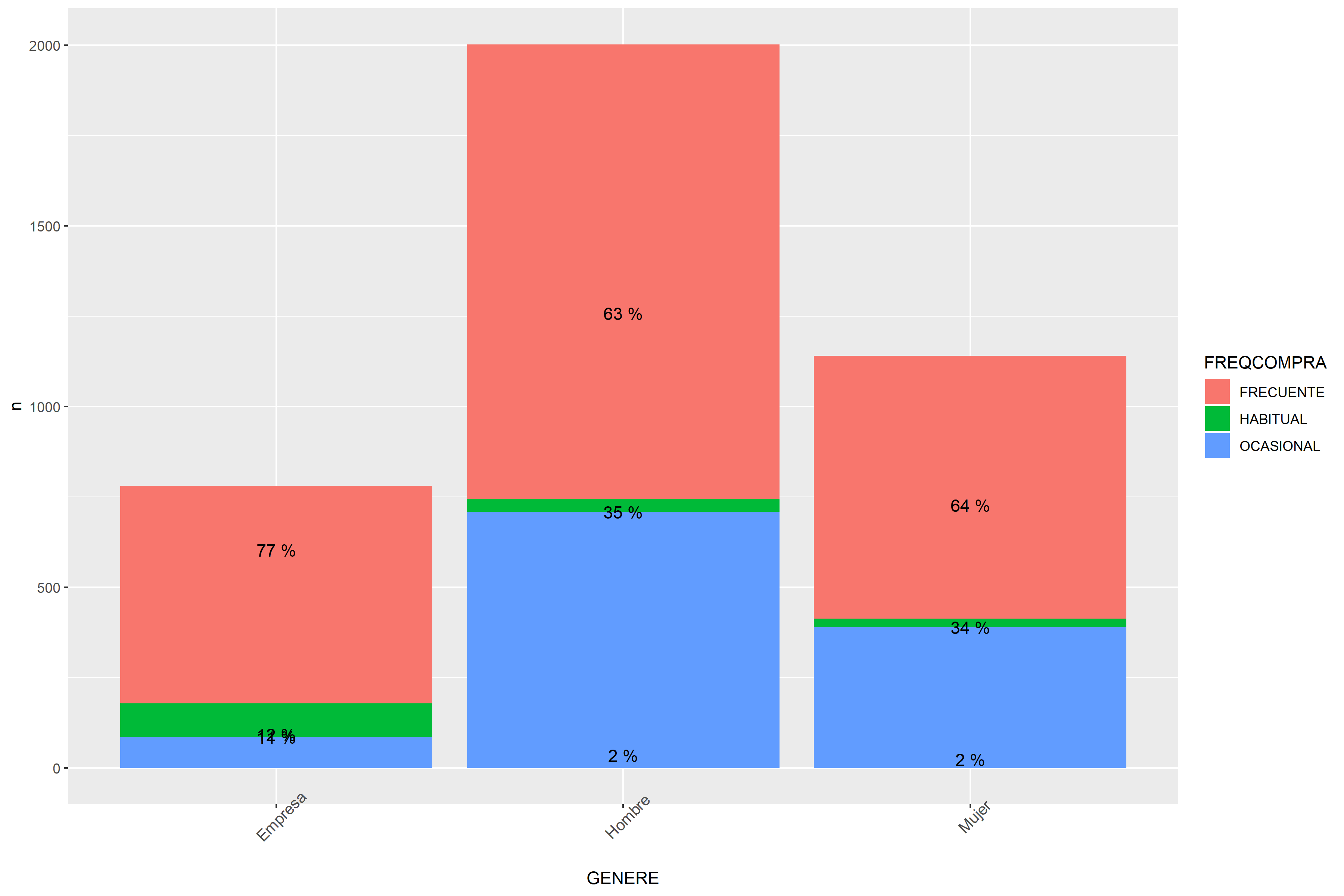
## Anàlisi Exploratori de les Dades

En primer lloc, passem a estudiar la variable GENERE respecte a FREQCOMPRA. Per tal d´explorar la relació entre aquestes variables realitzarem les següents operacions:

* Agrupar el conjunt de dades per la variable GENERE.
* Contar el nombre d´observacions de la variable FREQCOMPRA que apareixen en cada classe del atribut GENERE.
* Calcular el percentatge en cada classe segons la variable FREQCOMPRA.

Totes aquestes operacions queden simplificades amb l´ajuda de les funcions group\_by i count del paquet dplyr, i de l´operador %>%:

# Gràfic barra genere vs freqüència compra  
tickets\_client %>%   
 group\_by(GENERE) %>%   
 count(FREQCOMPRA) %>%   
 mutate(freq = round(n / sum(n) \* 100, 0)) %>%   
 ggplot(mapping = aes(x = GENERE, y = n, fill = FREQCOMPRA)) +  
 geom\_bar(stat = "identity") +  
 geom\_text(aes(label = paste(freq, "%"))) +  
 theme(axis.text.x = element\_text (size = 10, angle=45))



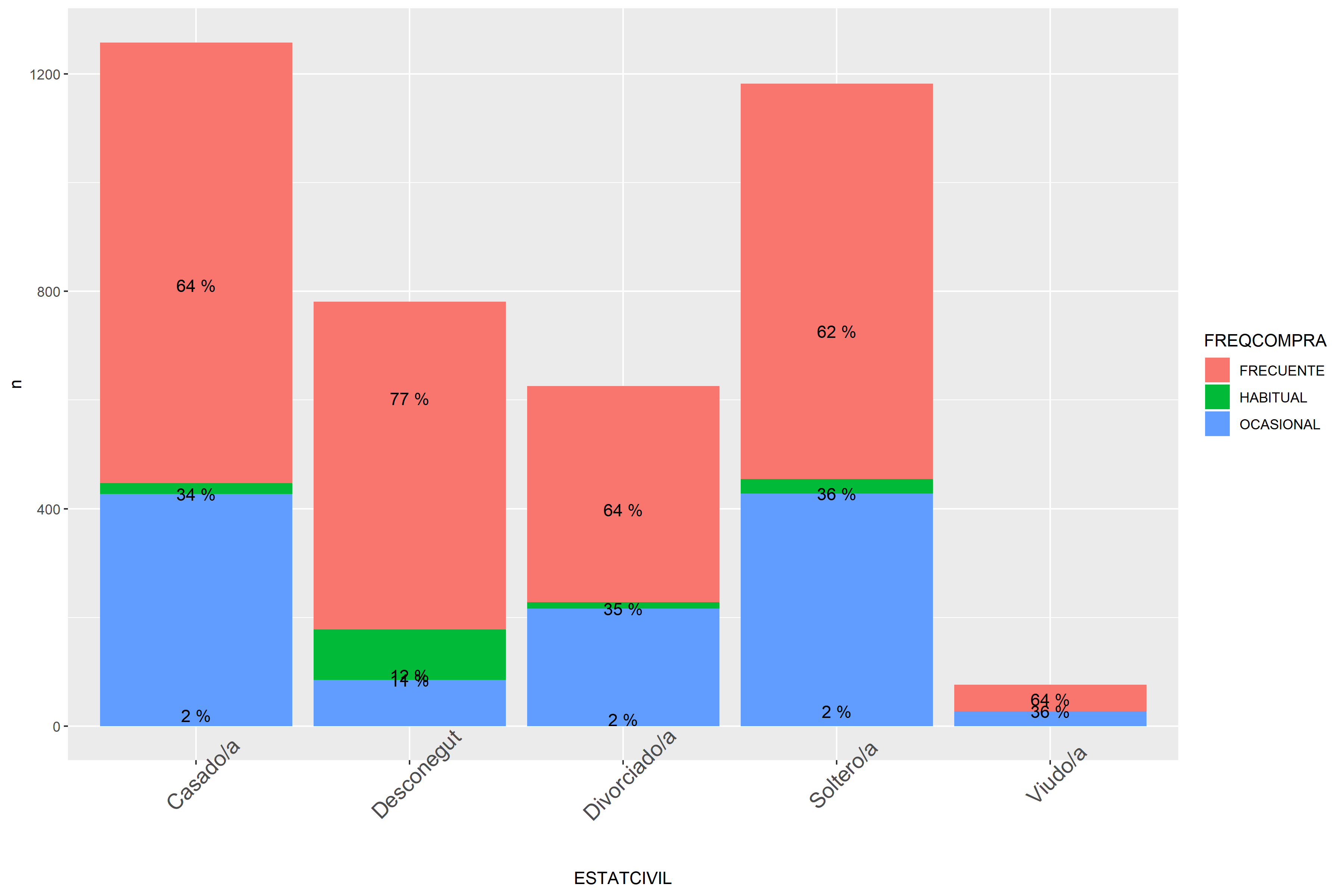
És a saber, que obtenim els mateixos resultats que al gràfic anterior però en format de resúm amb el següent codi:

# Resúm de les dades  
tickets\_client %>%   
 group\_by(GENERE) %>%   
 count(FREQCOMPRA) %>%   
 mutate(FREQ = round(n / sum(n) \* 100, 0))

## # A tibble: 9 x 4  
## # Groups: GENERE [3]  
## GENERE FREQCOMPRA n FREQ  
## <fct> <chr> <int> <dbl>  
## 1 Empresa FRECUENTE 603 77  
## 2 Empresa HABITUAL 93 12  
## 3 Empresa OCASIONAL 85 11  
## 4 Hombre FRECUENTE 1258 63  
## 5 Hombre HABITUAL 35 2  
## 6 Hombre OCASIONAL 709 35  
## 7 Mujer FRECUENTE 727 64  
## 8 Mujer HABITUAL 24 2  
## 9 Mujer OCASIONAL 389 34

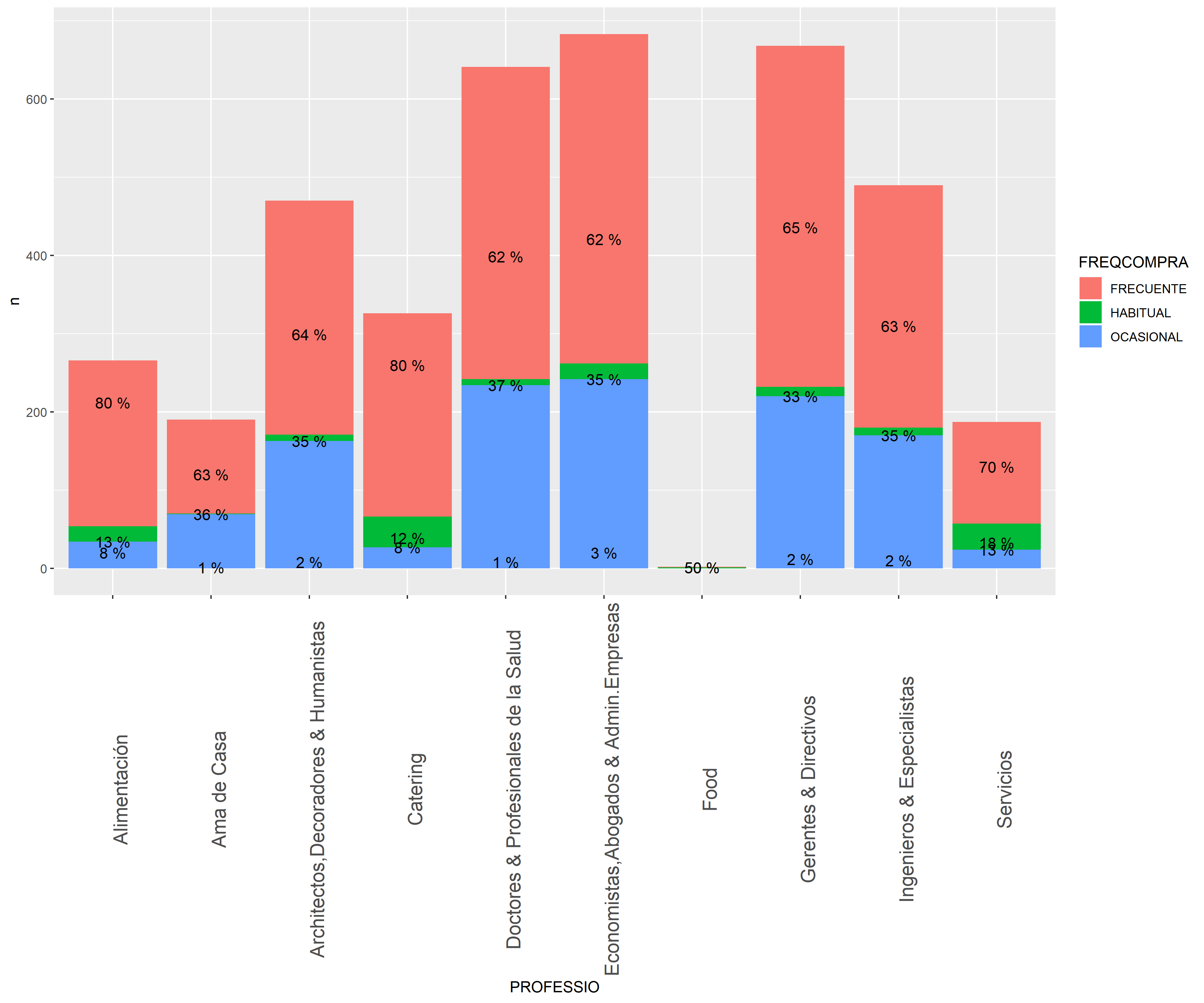
En segon lloc, estudiarem la variable ESTATCIVIL. Realitzant els mateixos passos que en el apartat anterior obtenim el següent diagrama de barres apilat:

# Gràfic barra estat civil vs freqüència de compra  
tickets\_client %>%   
 group\_by(ESTATCIVIL) %>%   
 count(FREQCOMPRA) %>%   
 mutate(freq = round(n / sum(n) \* 100, 0)) %>%   
 ggplot(mapping = aes(x = ESTATCIVIL, y = n, fill = FREQCOMPRA)) +  
 geom\_bar(stat = "identity") +  
 geom\_text(aes(label = paste(freq, "%"))) +  
 theme(axis.text.x = element\_text (size = 14, angle=45))



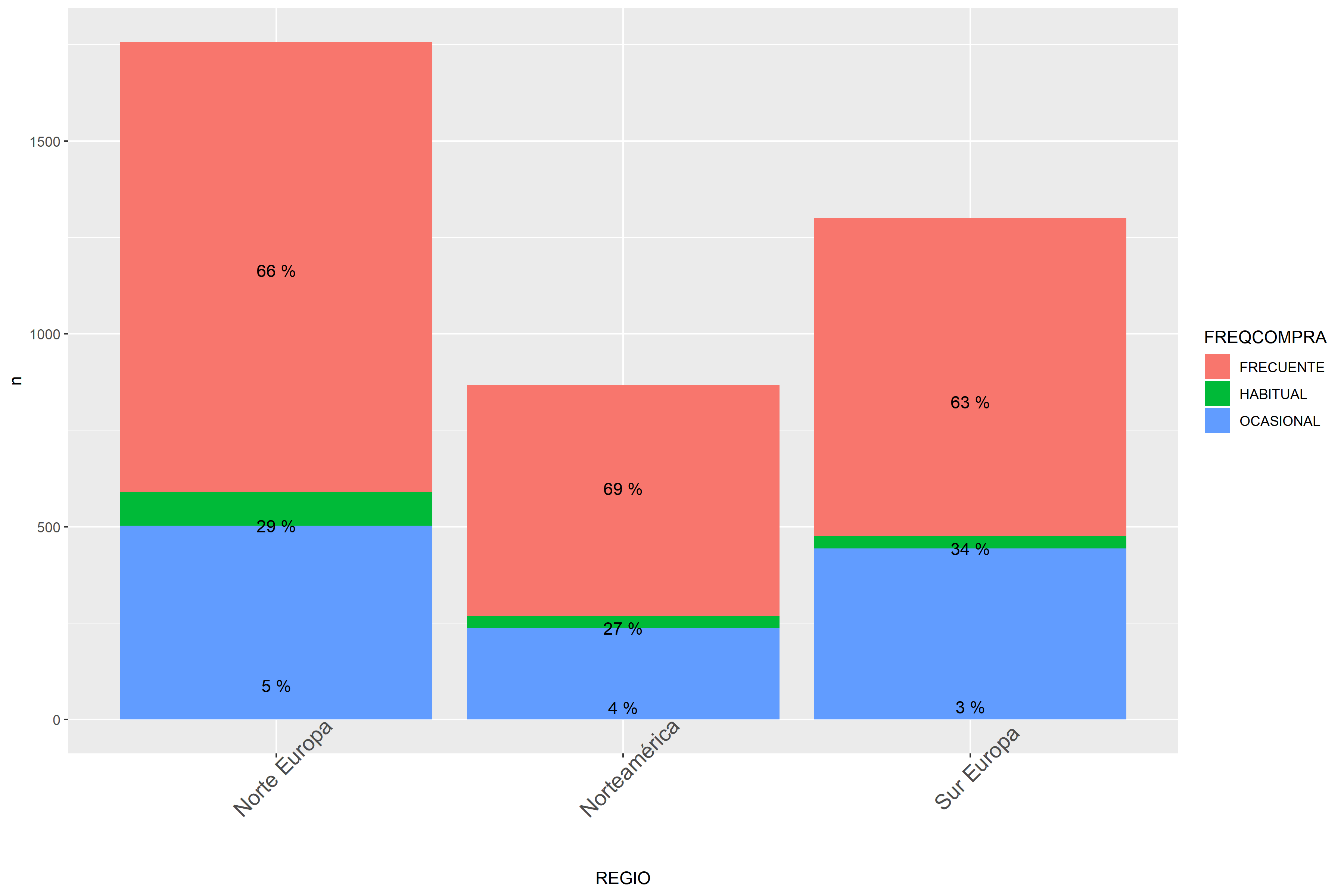
Prosseguim el nostre anàlisi amb la variable PROFESSIO. Igual com hem fet anteriorment obtenim el següent diagrama de barres:

# Gràfic barra professió vs freqüència de compra  
tickets\_client %>%   
 group\_by(PROFESSIO) %>%   
 count(FREQCOMPRA) %>%   
 mutate(freq = round(n / sum(n) \* 100, 0)) %>%   
 ggplot(mapping = aes(x = PROFESSIO, y = n, fill = FREQCOMPRA)) +  
 geom\_bar(stat = "identity") +  
 geom\_text(aes(label = paste(freq, "%"))) +  
 theme(axis.text.x = element\_text (size = 14, angle=90))



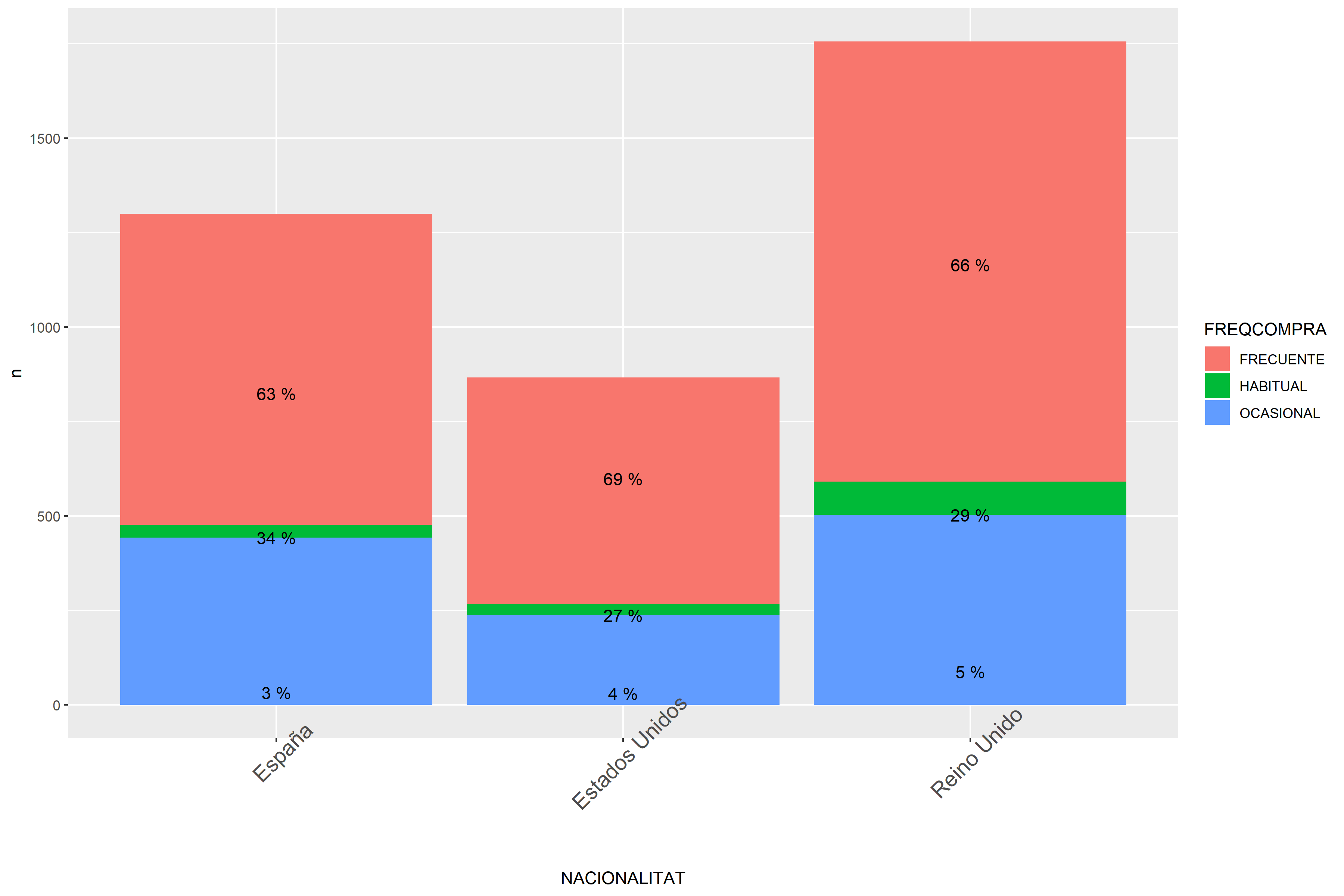
Avançant en el nostre anàlisi exploratori , passem a examinar la covariació entre les variables REGIO i FREQCOMPRA:

# Gràfic barra regio vs freqüència de compra  
tickets\_client %>%   
 group\_by(REGIO) %>%   
 count(FREQCOMPRA) %>%   
 mutate(freq = round(n / sum(n) \* 100, 0)) %>%   
 ggplot(mapping = aes(x = REGIO, y = n, fill = FREQCOMPRA)) +  
 geom\_bar(stat = "identity") +  
 geom\_text(aes(label = paste(freq, "%"))) +  
 theme(axis.text.x = element\_text (size = 14, angle=45))



Per acabar, analitzem la variable NACIONALITAT. El següent fragmet de codi ens mostra una representació gràfica en diagrama de barres apilat de la variable respecte a FREQCOMPRA:

# Gràfic barra regio vs freqüència de compra  
tickets\_client %>%   
 group\_by(NACIONALITAT) %>%   
 count(FREQCOMPRA) %>%   
 mutate(freq = round(n / sum(n) \* 100, 0)) %>%   
 ggplot(mapping = aes(x = NACIONALITAT, y = n, fill = FREQCOMPRA)) +  
 geom\_bar(stat = "identity") +  
 geom\_text(aes(label = paste(freq, "%"))) +  
 theme(axis.text.x = element\_text (size = 14, angle=45))



# PART II Clustering

## Requisits

Per començar, per a la realització del nostre anàlisi necessitarem els següents paquets:

* cluster per a la computació dels algoritmes d´agregació.
* factoextra per a la visualització de resultats d´agregació i que es fonamenta en el paquet ggplot2.[[7]](#footnote-52)
* clValid que s´utilitza per a comparar els mètodes d´agregació.
* clustertend per avaluar estadisticament de les tendencies d´agregació.

El paquet factoextra conté funcions per anàlisi de *clustering* i visualització dels resultats:

|  |  |
| --- | --- |
| Funció | Descripció |
| dist(fviz\_dist, get\_dist) | Visualització i computació de la matriu de distàncies |
| get\_clust\_tendency | Avaluació de la tendència d´agregació |
| fviz\_nbclust(fviz\_gap\_stat) | Determinació del nombre òptim de clústers |
| fviz\_dend | Visualització de dendrogrames |
| fviz\_cluster | Visualització dels resultats d´agrupament |
| fviz\_mclust | Visualització dels resultats del model d´agrupament |
| fviz\_silhouette | Visualització de la informació de la silueta |
| hkmeans | K-means jeràrquic |
| eclust | Visualització de l´anàlisi de agrupament |

Podem instal·lar els dos paquets com es mostra en la següent línia de codi:

# Instalació paquets clustering  
install.packages(c("cluster", "factoextra", "clValid", "clustertend"))

En acabat, ens caldrà carregar les llibreries a la sessió R:

# Carreguem les llibreries  
library(cluster)  
library(factoextra)  
library(clValid)

## Determinació del nombre de clústers

Per a determinar el nombre de clústers farem ús de la funció fviz\_nbclust() del paquet factoextra que calcula els mètodes **Elbow**, **Silhouhette** i **Gap**.

El prototip de la funció es el següent:

fviz\_nbclust(x, FUNcluster, method = c("silhouette", "wss", "gap\_stat"))

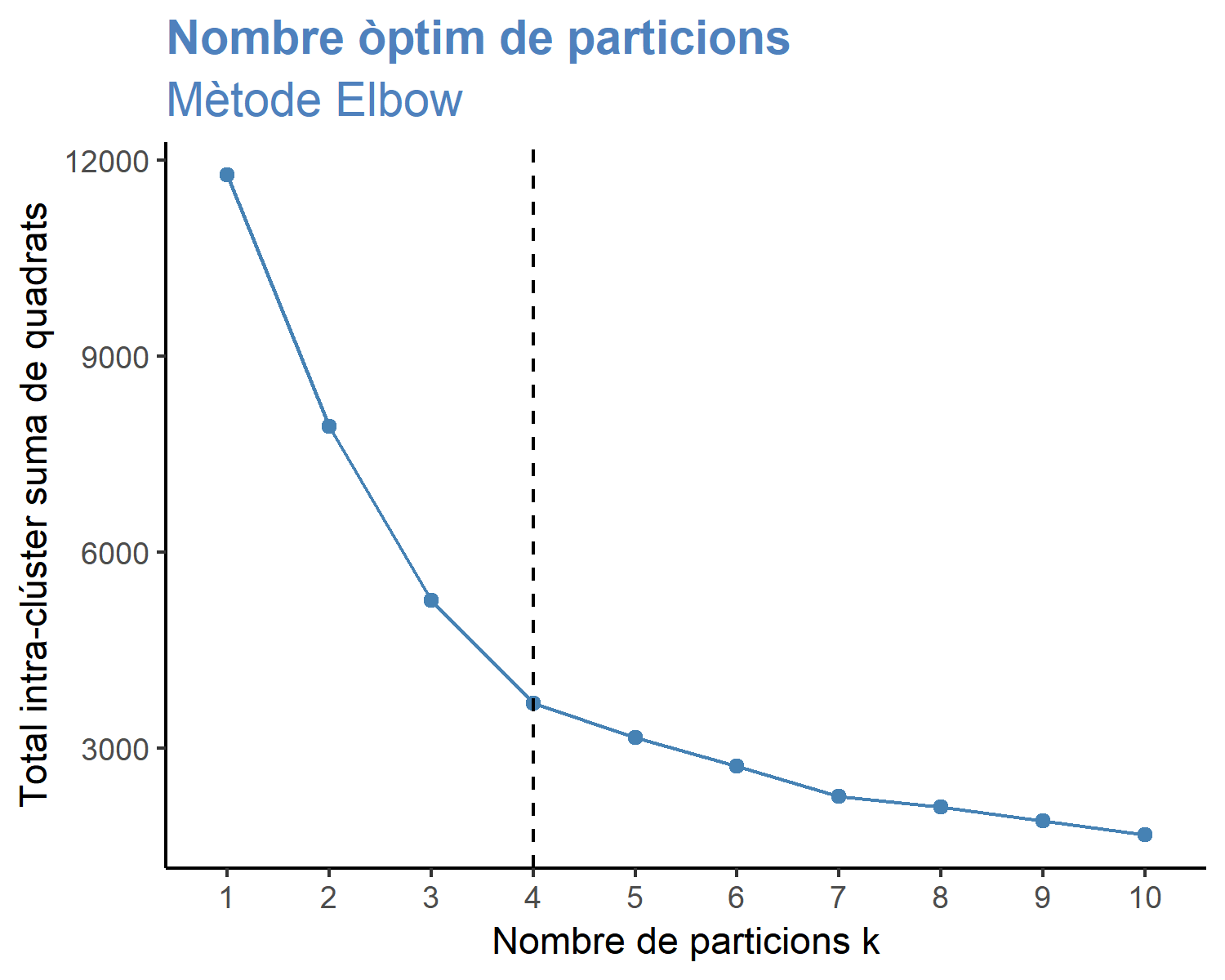
on els arguments són els següents:

* **x:** matriu o data frame.
* **FUNcluster:** una funció d´agregació. Valors possibles: kmeans, pam, clara i hcut.
* **method:** mètode per a determinar el nombre òptim de clústers. Valors possibles: **Elbow**, **Silhouhette** i **Gap**

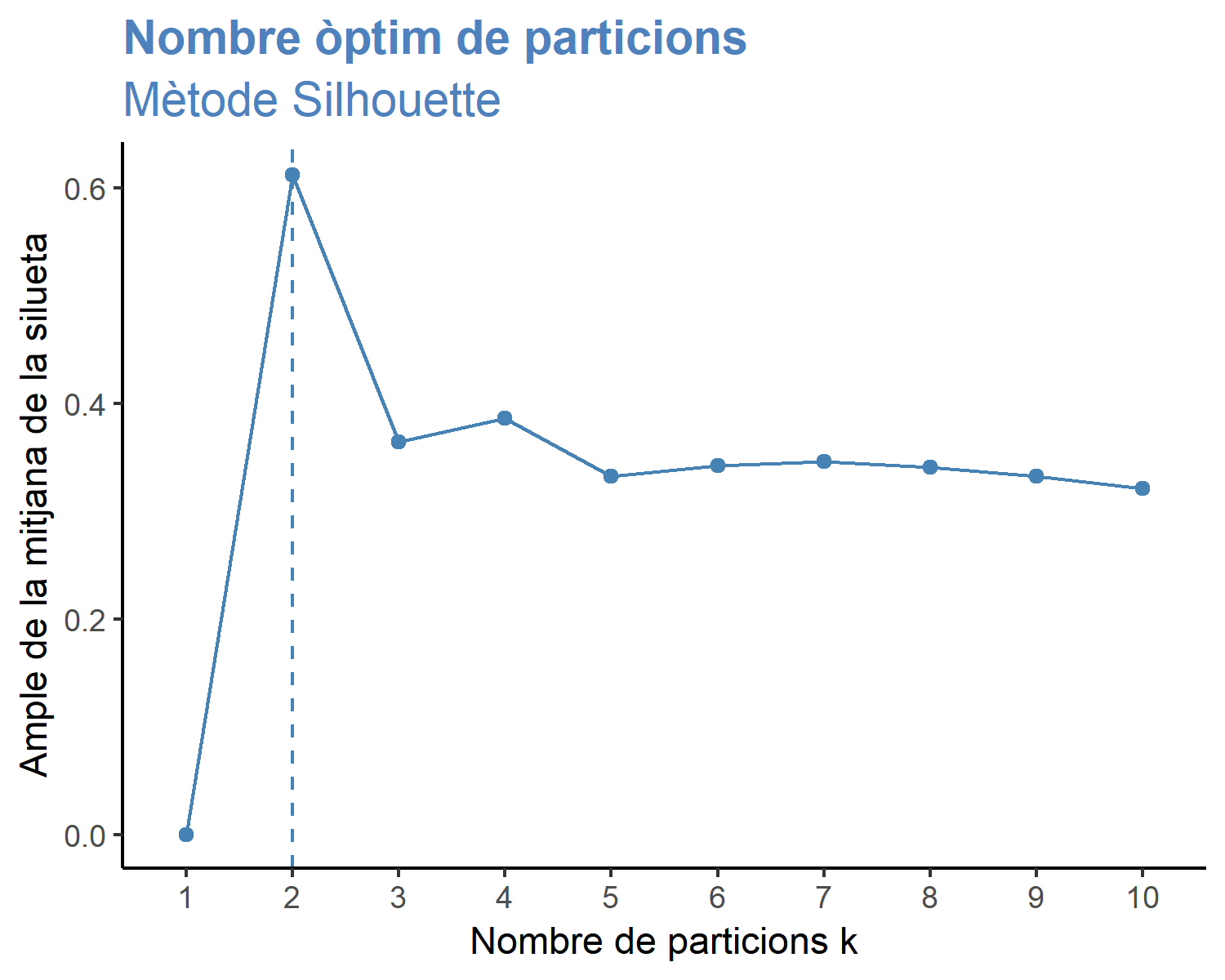
Per al nostre anàlisi de la segmentació dels nostres clients utilitzarem el següent conjunt de prova:

A continuació, es mostra com determinar el nombre òptim de particions per al mètode ***k-means***:

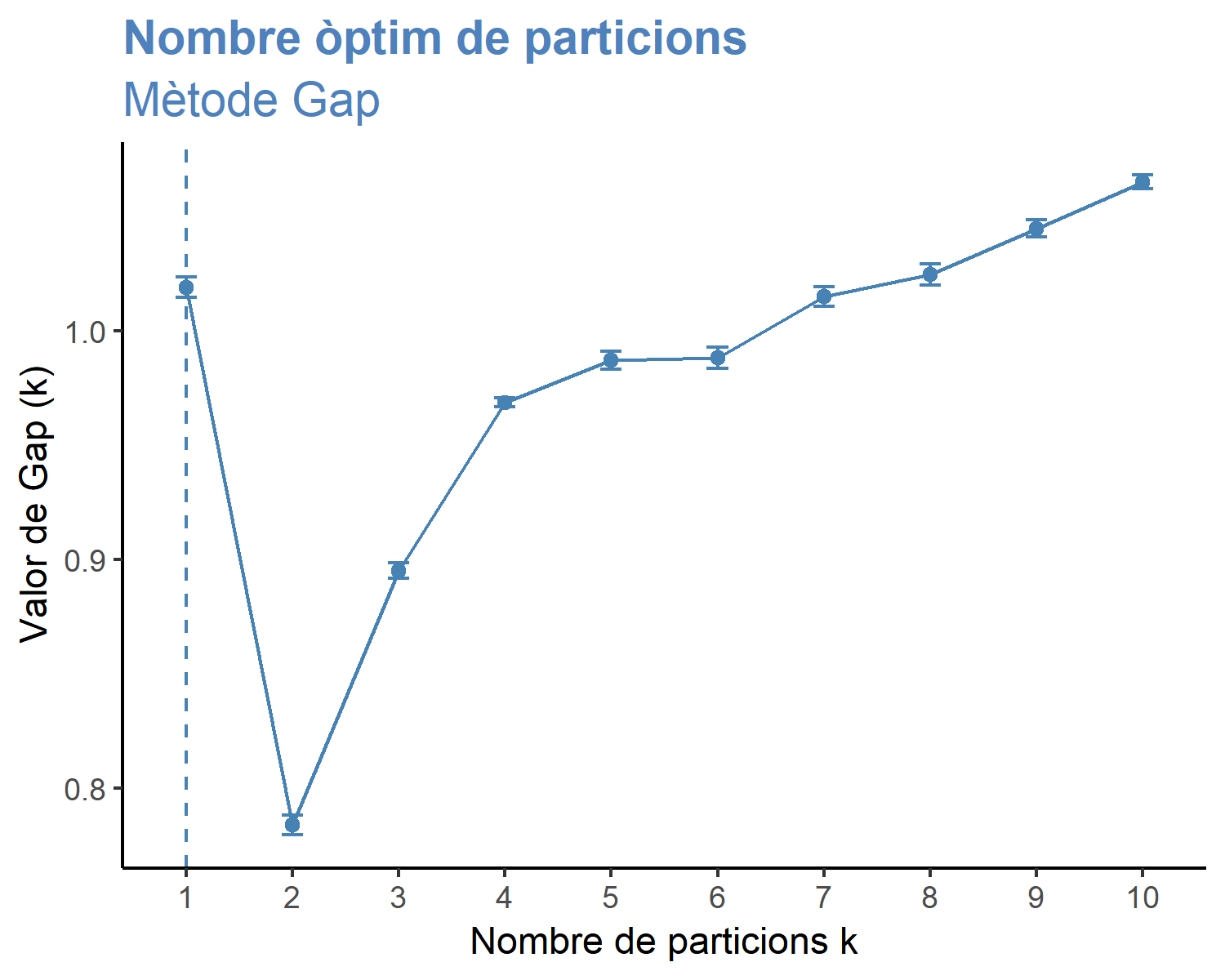
# Mètode elbow  
fviz\_nbclust(tickets, kmeans, method = "wss") +  
 geom\_vline(xintercept = 4, linetype = 2) +  
 labs(x = "Nombre de particions k",   
 y = "Total intra-clúster suma de quadrats",  
 title = "Nombre òptim de particions",  
 subtitle = "Mètode Elbow") +  
 theme\_classic() +  
 theme(plot.title = element\_text(color="#4F81BD", size=14, face="bold"),  
 plot.subtitle = element\_text(color="#4F81BD", size=14))



# Mètode Silhouette  
fviz\_nbclust(tickets, kmeans, method = "silhouette") +  
 labs(x = "Nombre de particions k",   
 y = "Ample de la mitjana de la silueta",  
 title = "Nombre òptim de particions",   
 subtitle = "Mètode Silhouette") +  
 theme\_classic() +  
 theme(plot.title = element\_text(color="#4F81BD", size=14, face="bold"),  
 plot.subtitle = element\_text(color="#4F81BD", size=14))



# Mètode Gap  
set.seed(123)  
fviz\_nbclust(tickets, kmeans, nstart = 25,   
 method = "gap\_stat", nboot = 5) +  
 labs(x = "Nombre de particions k",   
 y = "Valor de Gap (k)",  
 title = "Nombre òptim de particions",  
 subtitle = "Mètode Gap") +  
 theme\_classic() +  
 theme(plot.title = element\_text(color="#4F81BD", size=14, face="bold"),  
 plot.subtitle = element\_text(color="#4F81BD", size=14))



Com podem observar en els gràfics:

* El mètode Elbow ens suggereix 4 clústers.
* El mètode Silhoutte ens suggereix 2 clústers.
* El mètode Gap ens sugereix 4 clústers.

Així és que, segons aquestes observacions podem considerar *k* = 4 com el nombre òptim de clústers.

## Mètode d´agregació *k-means*

A causa de que, l´algoritme *k-means* comença seleccionant un centroide aleatòriament, es recomanable fer ús de la funció set.seed() a l´efecte de aconseguir resultats reproduïbles. Així el lector d´aquest document obtindrà els mateixos resultats que es presenten tot seguit.

A continuació es mostra com aplicar l´algorisme k-means amb k = 4:

# Execució k-means amb k = 4  
set.seed(123)  
kmeansFit <- kmeans(tickets, 4, nstart = 25)

Podem mostrar per pantalla els resultats amb la següent línia de codi:

# Mostrem els resultats  
print(kmeansFit)

## K-means clustering with 4 clusters of sizes 69, 74, 46, 7  
##   
## Cluster means:  
## TOTALCOMPRES EDAT TOTAL  
## 1 -0.343604 -1.0000575 -0.2771167  
## 2 -0.546290 0.8601257 -0.3153103  
## 3 0.873034 0.1985773 0.2374292  
## 4 3.424939 -0.5399843 4.5046100  
##   
## Clustering vector:  
## [1] 1 3 1 1 2 2 1 3 2 1 2 2 3 4 2 2 3 1 2 1 2 2 4 2 3 2 2 2 3 3 3 3 2 2 2  
## [36] 1 2 2 3 2 2 1 2 1 2 1 1 3 3 2 2 2 2 2 1 1 1 3 3 2 4 3 3 2 2 2 2 1 1 3  
## [71] 2 1 3 3 1 1 2 2 1 1 1 3 1 4 1 2 3 1 1 2 2 1 1 2 2 1 2 3 1 1 2 1 2 1 1  
## [106] 3 3 1 3 2 2 1 3 1 2 2 2 2 3 2 1 2 4 1 3 2 3 2 3 3 1 3 1 2 1 1 3 2 2 2  
## [141] 1 3 3 2 1 2 1 3 3 2 3 1 2 1 1 1 2 1 1 3 1 2 1 2 1 4 2 1 1 1 3 2 1 2 1  
## [176] 2 3 4 3 1 1 2 1 1 1 1 2 3 2 3 2 3 1 2 1 3  
##   
## Within cluster sum of squares by cluster:  
## [1] 33.40863 48.22872 40.68178 30.37939  
## (between\_SS / total\_SS = 73.9 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

Podem observar en la sortida el següent:

* La mitjana de clústers: una matriu, on les files són el nombre de clúster i les columnes són les variables.
* El vector de particions: un vector d´enters (de 1:k) que indica el clúster on cada observació ha sigut agrupada.

Així mateix, és recomanable realitzar un gràfic amb els resultats del model. Ja sigui, per a escollir el nombre de clústers, ja sigui per a comparar diferents anàlisis.

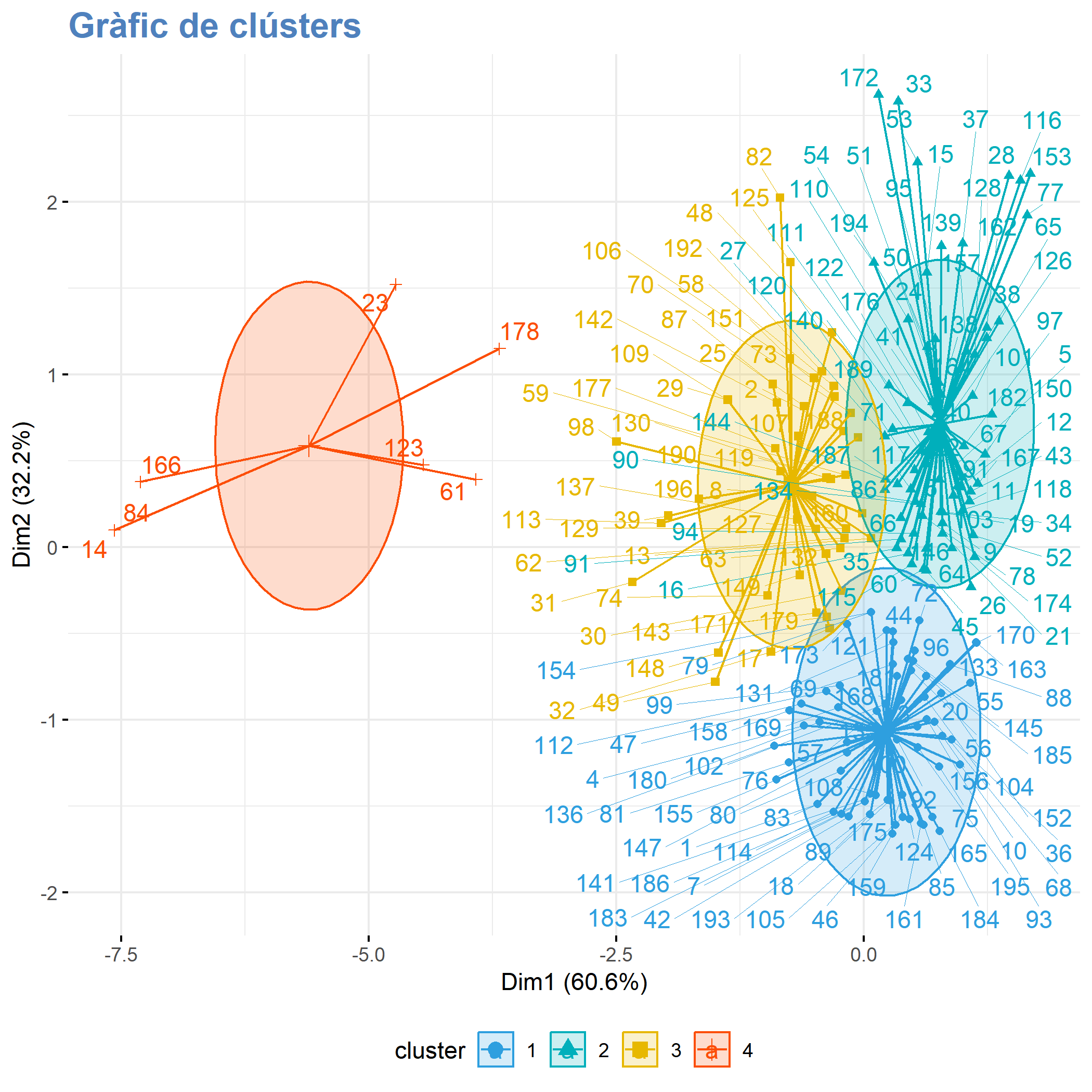
Una possible opció és visualitzar les dades en un diagrama de dispersió acolorint cada observació d’acord al grup assignat.

El problema és que el nostre conjunt de dades conté més de 2 variables i no és possible representar el model en dues dimensions.

Convé fer ressaltar que, una possible solució és reduir la dimensionalitat fent ús d´un algoritme de reducció del nombre d´atributs, com per exemple **Principal Component Analysis (PCA)**.

En aquest sentit, farem ús de la funció fviz\_cluster() que ens permetrà visualitzar els clústers i que utilitza PCA quan el nombre de variables és més gran de 2. Passarem com a arguments el resultat del model i el conjunt de dades original:

# Visualitzem els clústers  
fviz\_cluster(kmeansFit, data = tickets, stand = TRUE,  
 main = "Gràfic de clústers",  
 palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),  
 ellipse.type = "euclid",   
 star.plot = TRUE,  
 repel = TRUE,  
 ggtheme = theme\_minimal()) +   
 theme(legend.position = "bottom",  
 plot.title = element\_text(color="#4F81BD", size=16, face="bold"))



Podem observar en el gràfic que les observacions són representades mitjançant punts i que en el nostre cas s´ha usat PCA. A més, s´han dibuixat el.lipses per tal de diferenciar cada clúster.

# PART III Classificació

## El paquet Caret

Per a la realització del model predictiu utilitzarem el paquet caret[[8]](#footnote-62) (acrònim per a **C** lassification **A** nd **RE** gression **T** raining). Aquest paquet es un *framework* amb un conjunt de funcions que pretén optimitzar el procés de la creació de models predictius. Aquest paquet conté eines per a:

* Divisió del conjunt de dades.
* Pre-processament de dades.
* Selecció d´atributs.
* *Model tunning*[[9]](#footnote-64) mitjançant remostreig.
* Estimació de la importància de les variables.

En primer lloc, caldrà instal·lar el paquet des del repositori CRAN:

install.packages("caret", dependencies = TRUE)

A més, ens caldrà instal.lar els següents paquets:

* El paquet RWeka que implementa l´algoritme C4.5.
* El paquet C50 es tracta d´una implementació més moderna de l´algorisme ID3.
* El paquet rpart que implementa el mètode CART.
* El paquet randomForest que implementa l´agoritme de “Boscos aleatoris”[[10]](#footnote-65).

install.packages(c("RWeka", "C50", "rpart", "randomForest"))

## Separació de dades: Conjunt d´entrenament i prova:

En aquest apartat, dividiren el conjunt de dades en dos subconjunts:

* Conjunt d´entrenamet: Un subconjunt per a entrenar el model.
* Conjunt de prova: Un subconjunt per a provar el model entrenat.

L´objectiu en aquesta tasca es dividir el conjunt de dades de la següent manera:



Figura 1. Divisió d´un conjunt de dades en un conjunt d´entrenament i un de prova.

Cal que ens assegurem que el conjunt de prova reuneixi les següents dos condicions:

* Que sigui prou gran com per generar resultats significatius des del punt de vista estadístic.
* Que sigui representatiu de tot el conjunt de dades. En altres paraules, no triar un conjunt de prova amb característiques diferents al del conjunt d’entrenament.

Si suposem que el conjunt de prova reuneix aquestes dues condicions, el nostre objectiu és crear un model que generalitzi les dades noves de forma correcta. En altres paraules, hem de aconseguir un model que no sobreajusti (en àngles, *overfitting*) les dades d’entrenament.

La funció createDataPartition ens permetra crear els subconjunts de dades. Per exemple, per a crear una divisió de un 80/20% del conjunt de dades tickets\_client:

# Carreguem el paquet  
library(caret)  
set.seed(1234)  
# Creem el conjunt d´entrenament i el de prova  
inTrain <- createDataPartition(tickets\_client$FREQCOMPRA, p = 0.8, list = FALSE)  
training <- tickets\_client[inTrain, ]  
testing <- tickets\_client[-inTrain, ]

A continuació podem comprovar que hem obtingut un conjunt de entrenament de 2606 observacions:

dim(training)

## [1] 3140 13

## Classificació amb CART

Tot seguit clasificarem de nou les dades d´entrenament, però aquest cop utilitzarem l´algoritme CART:

# Carreguem el framework `caret`  
library(caret)  
# Execució model CART  
modelFitCART <- train(FREQCOMPRA ~ .,  
 data = training,   
 method = "rpart")

En aquet cas i amb l´ajuda del paquet rattle podem obtindre el gràfic de l´arbre de decisió:

# Carreguem la llibreria `rattle`  
library(rattle)  
# Obtenim un gràfic de l´arbre  
fancyRpartPlot(modelFitCART)

# PART IV Regles d´associació

# PART V Conclusions i Recomanacions al Client

# Bibliografia

[1] Daniel T. Larouse, Chantal D. Larouse: Data Mininig and Predictive Analytics.USA, John Wiley & Sons,2015,ISBN 978-1-118-11619-7

[2] Jordi Gironés Roig, Jordi Casas Roma, Julià Minguillón Alfonso, Ramon Caihuelas Quiles : Minería de Datos: Modelos y Algoritmos. Barcelona, Editorial UOC, 2017, ISBN: 978-84-9116-904-8.

[3] Jiawe Han, Michellie Chamber & Jian Pei: Data mining : concepts and techniques. 3º Edition. USA, Editorial Elsevier, 2012, ISBN 978-0-12-381479-1

1. <https://es.wikipedia.org/wiki/Markdown> [↑](#footnote-ref-22)
2. <https://www.rstudio.com/> [↑](#footnote-ref-24)
3. <https://rmarkdown.rstudio.com/> [↑](#footnote-ref-27)
4. Paquet per a la lectura de dades amb format rectangular: <https://readr.tidyverse.org/> [↑](#footnote-ref-34)
5. Conjunt de paquets R per a la Ciència de les Dades :<https://www.tidyverse.org/> [↑](#footnote-ref-36)
6. La notació paquet::nom\_funció s´utilitza per a indicar a R que es vol fer ús de la funció del paquet indicat, en el cas que existeixi ambigüitat amb el nom d´una funció en un altre paquet. [↑](#footnote-ref-40)
7. La documentació oficial es pot trobar a: <http://www.sthda.com/english/rpkgs/factoextra>. [↑](#footnote-ref-52)
8. Per a més informació: <https://topepo.github.io/caret/index.html> [↑](#footnote-ref-62)
9. Procés que consisteix en optimitzar els paràmetres del model amb l´objectiu que l´algoritme obtingue el millor rendiment. [↑](#footnote-ref-64)
10. De l`anglés *Random Forest*. Per a més informació :<https://es.wikipedia.org/wiki/Random_forest> [↑](#footnote-ref-65)