Drug consumption

Alex Ferrando de las Morenas, Carlos Hurtado Comín, Maria Ribot Vilà

Importació llibreries necessàries per la pràctica

```
library(naivebayes)   # Necessària per fer Naive Bayes
```

```
## naivebayes 0.9.7 loaded
```

```
library(rockchalk)    # Necessària per la funció combineLevels
library(MASS)         # Utils per funcions estadístiques
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:rockchalk':
##
##     mvrnorm
```

```
library(ca)           # Per fer l'anàlisi de correspondències
library(nnet)         # Per a implementar la neural network
library(caret)        # Per al trainControl
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(randomForest) # Random forest
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(ggplot2)      # Per fer plots
```

Lectura de la base de dades.

```r
dd = read.csv("drug_consumption.data", header = F)
colnames(dd) = c("ID","Age","Gender","Education","Country","Ethnicity","Nscore",
                 "Escore","Oscore","Ascore" ,"Cscore", "Impulsive", "SS", "Alcohol",
                 "Amphet","Amyl","Benzos","Caff","Cannabis","Choc","Coke","Crack",
                 "Ecstasy","Heroin","Ketamine","Legalh","LSD","Meth","Mushrooms",
                 "Nicotine","Semer" ,"VSA")
dd = dd[,-6]
```

Binarització de les variable resposta

```r
dd[, "Cannabis"] = combineLevels(dd[, "Cannabis"], levs = c("CL0", "CL1", "CL2"),
                                 newLabel = "NON-USER")
```

```
## The original levels CL0 CL1 CL2 CL3 CL4 CL5 CL6
## have been replaced by CL3 CL4 CL5 CL6 NON-USER
```

```r
dd[, "Cannabis"] = combineLevels(dd[, "Cannabis"], levs = c("CL3", "CL4", "CL5", "CL6"),
                                 newLabel = "USER")
```

```
## The original levels CL3 CL4 CL5 CL6 NON-USER
## have been replaced by NON-USER USER
```

Copiem la base de dades en *dd2* per poder fer un estudi previ coherent i sense la pèrdua d'interpretabilitat que genera el tractament previ que han tingut les dades i la conseqüent transformació de factors a valors reals.

```r
dd2 = dd

dd2$Age = as.factor(dd2$Age)
levels(dd2$Age) = c("18-24", "25-34", "35-44", "45-54", "55-64","65+")

dd2$Gender = as.factor(dd2$Gender)
levels(dd2$Gender) = c("Male", "Female")

dd2$Education = as.factor(dd2$Education)
levels(dd2$Education) = c("< 16", "16", "17", "18", "> 18",
                          "Professional Certificate", "University", "Master",
                          "Doctorate")

dd2$Country = as.factor(dd2$Country)
levels(dd2$Country) = c("USA", "New Zealand", "Other", "Australia", "Ireland", "Canada",
                        "UK")

dd2$Nscore = as.factor(dd2$Nscore)
levels(dd2$Nscore) = 12:60
```

```r
dd2$Nscore = as.integer(dd2$Nscore)

dd2$Escore = as.factor(dd2$Escore)
levels(dd2$Escore) = setdiff(16:59, 17)
dd2$Escore = as.integer(dd2$Escore)

dd2$Oscore = as.factor(dd2$Oscore)
levels(dd2$Oscore) = setdiff(24:60, c(25, 27))
dd2$Oscore = as.integer(dd2$Oscore)

dd2$Ascore = as.factor(dd2$Ascore)
levels(dd2$Ascore) = setdiff(12:60, c(13, 14, 15, 17, 19, 20, 21, 22))
dd2$Ascore = as.integer(dd2$Ascore)

dd2$Cscore = as.factor(dd2$Cscore)
levels(dd2$Cscore) = setdiff(17:59, c(18, 58))
dd2$Cscore = as.integer(dd2$Cscore)

dd2$Impulsive = as.factor(dd2$Impulsive)
levels(dd2$Impulsive) = 1:10
dd2$Impulsive = as.integer(dd2$Impulsive)

dd2$SS = as.factor(dd2$SS)
levels(dd2$SS) = 1:11
dd2$SS = as.integer(dd2$SS)
```

1. Preprocessament de les dades

1.1 Estudi de les dades

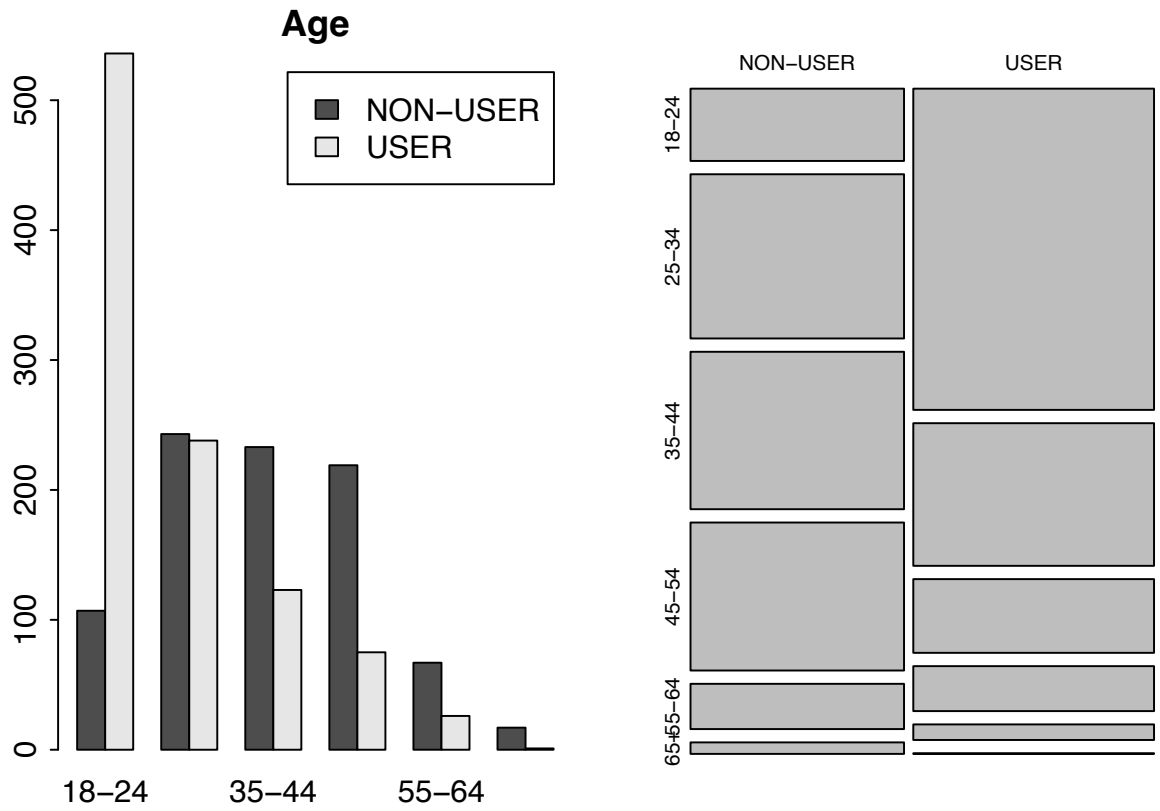
```r
summary(dd2[, c(2:13, 29, 18)])
```

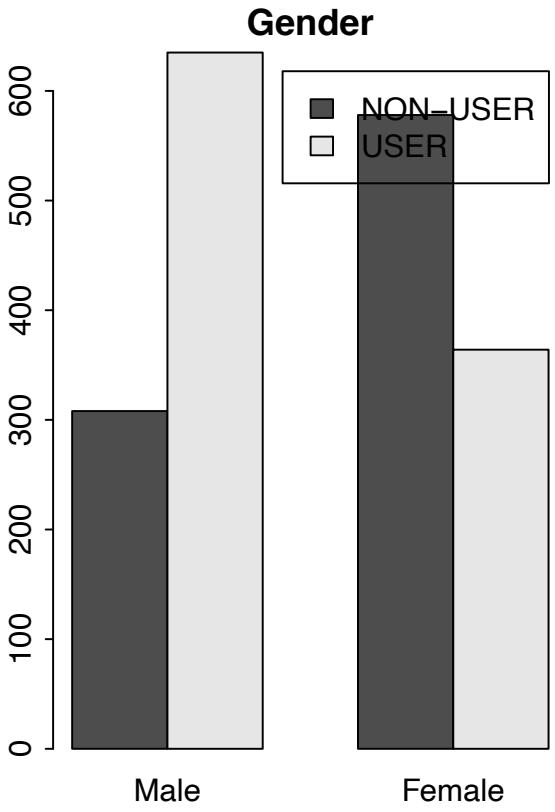
```
##     Age         Gender                         Education          Country
##  18-24:643   Male  :943   > 18                      :506   USA         : 557
##  25-34:481   Female:942   University                :480   New Zealand:   5
##  35-44:356                Master                    :283   Other       : 118
##  45-54:294                Professional Certificate:270   Australia   :  54
##  55-64: 93                18                        :100   Ireland     :  20
##  65+  : 18                16                        : 99   Canada      :  87
##                           (Other)                   :147   UK          :1044
##      Nscore          Escore          Oscore          Ascore
##  Min.   : 1.00   Min.   : 1.00   Min.   : 1.00   Min.   : 1.00
##  1st Qu.:18.00   1st Qu.:19.00   1st Qu.:16.00   1st Qu.:20.00
##  Median :25.00   Median :24.00   Median :21.00   Median :24.00
##  Mean   :24.92   Mean   :23.58   Mean   :20.77   Mean   :23.88
##  3rd Qu.:31.00   3rd Qu.:28.00   3rd Qu.:26.00   3rd Qu.:29.00
##  Max.   :49.00   Max.   :42.00   Max.   :35.00   Max.   :41.00
##
##      Cscore        Impulsive           SS         Alcohol   Nicotine
##  Min.   : 1.00   Min.   : 1.000   Min.   : 1.000   CL0: 34   CL0:428
```

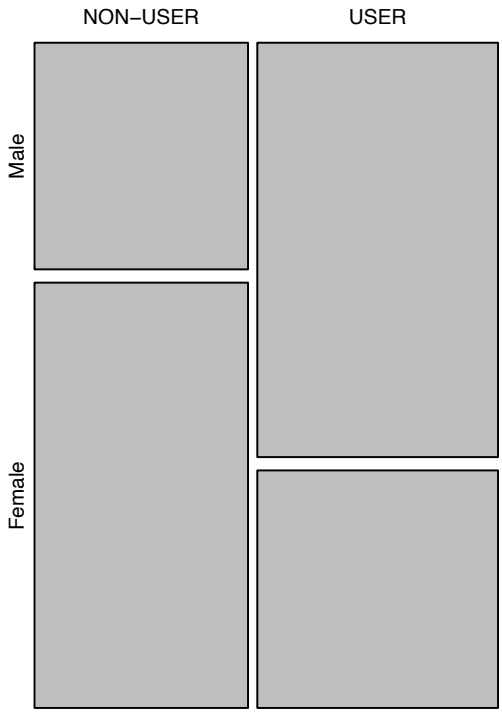
```
##  1st Qu.:20.00   1st Qu.: 3.000   1st Qu.: 5.000   CL1: 34   CL1:193
##  Median :25.00   Median : 4.000   Median : 7.000   CL2: 68   CL2:204
##  Mean   :24.44   Mean   : 4.801   Mean   : 6.561   CL3:198   CL3:185
##  3rd Qu.:29.00   3rd Qu.: 6.000   3rd Qu.: 9.000   CL4:287   CL4:108
##  Max.   :41.00   Max.   :10.000   Max.   :11.000   CL5:759   CL5:157
##                                                    CL6:505   CL6:610
##      Cannabis
##  NON-USER:886
##  USER    :999
##
##
##
##
##
```

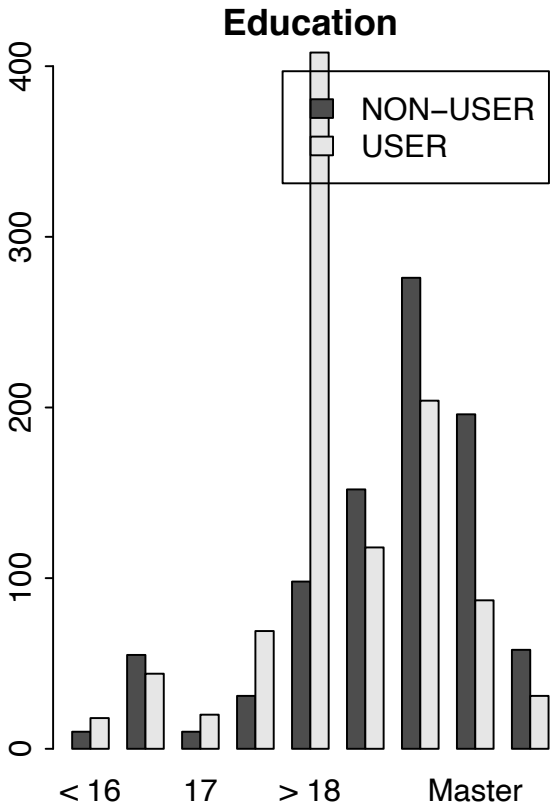
1.2 Barplots i graficació de taules de contingència

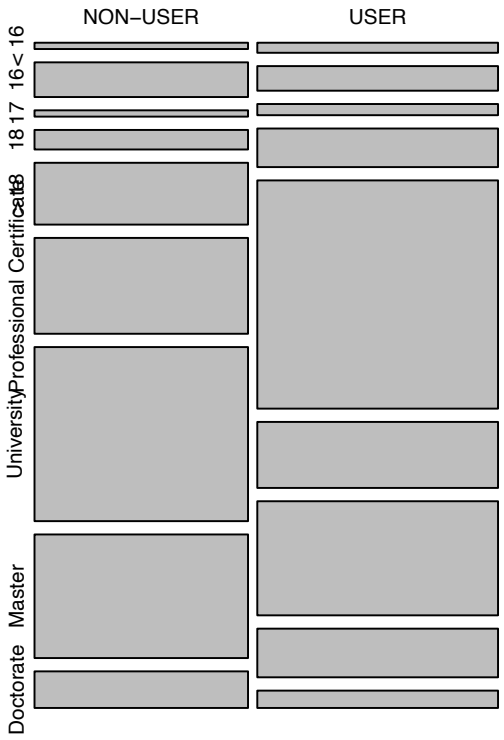
```
par(mfrow = c(1, 2), mex = 0.4)
for(i in c(2:5, 13, 29)) {
   tab =  table(dd2$Cannabis, dd2[,i])
   barplot(tab, beside = TRUE, legend = levels(dd$Cannabis), main = colnames(dd2)[i])
   plot(tab, main = "")
}
```

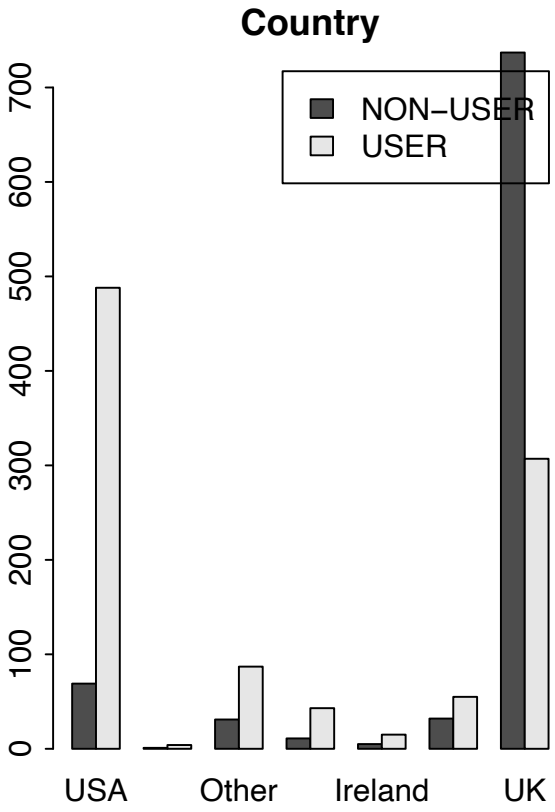


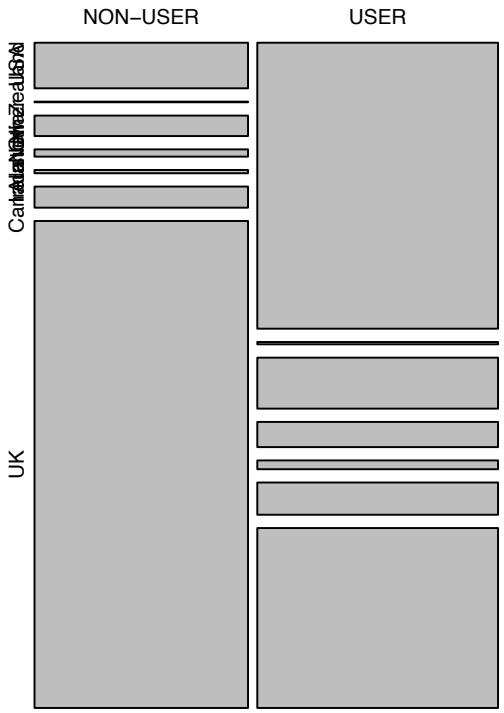

Gender
NON−USER
USER
0
100
200
300
400
500
600
Male
Female

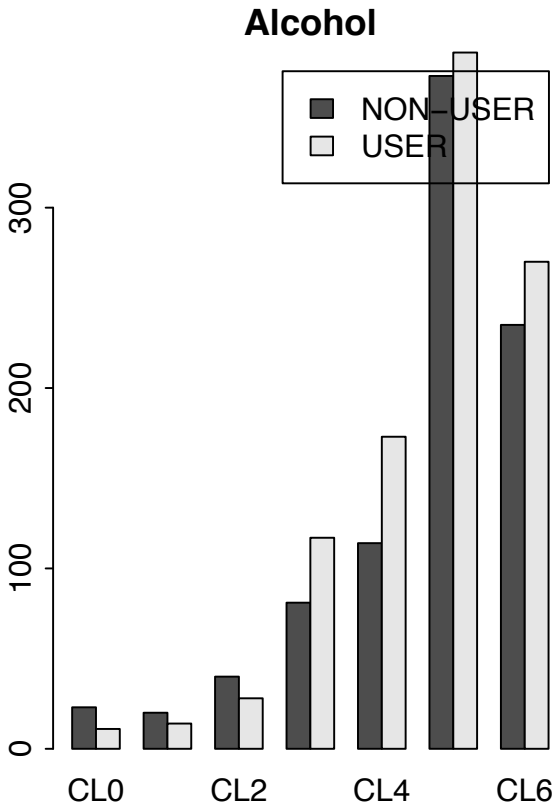

NON−USER
USER
Male
Female

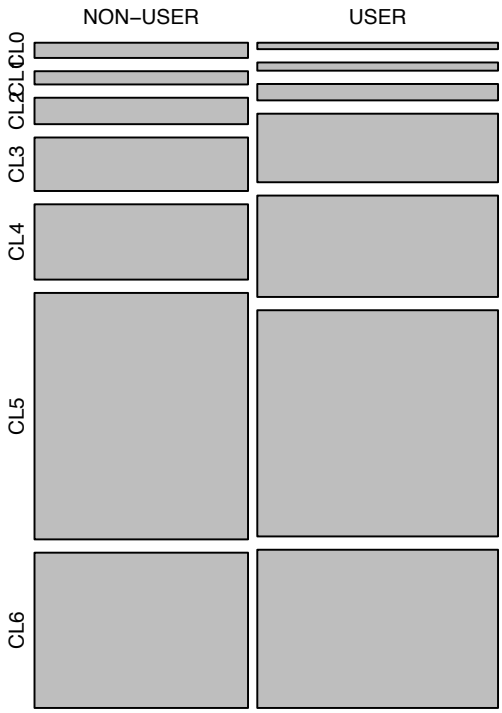

Education
NON−USER
USER
0
100
200
300
400
< 16
17
> 18
Master

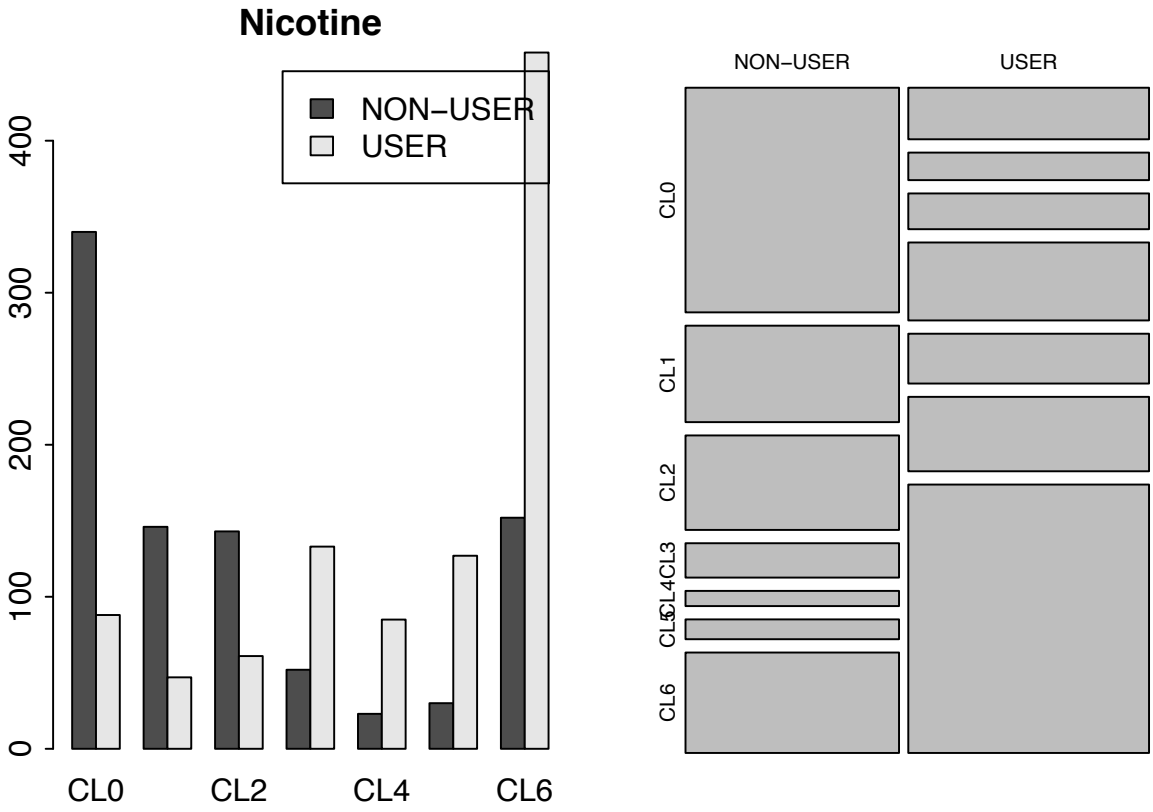

NON−USER
USER
< 16
16
17
18
> 18
Professional Certificate
University
Master
Doctorate


Country
NON−USER
USER
0
100
200
300
400
500
600
700
USA
Other
Ireland
UK

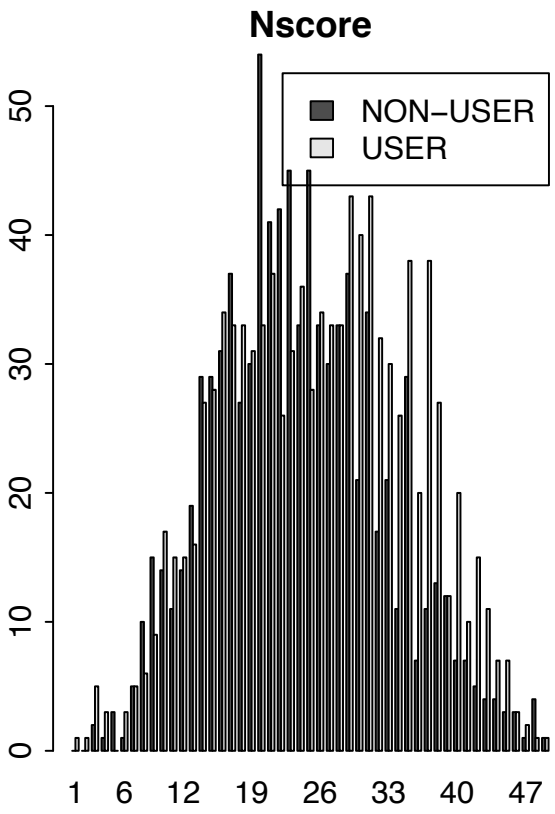

NON−USER
USER
UK

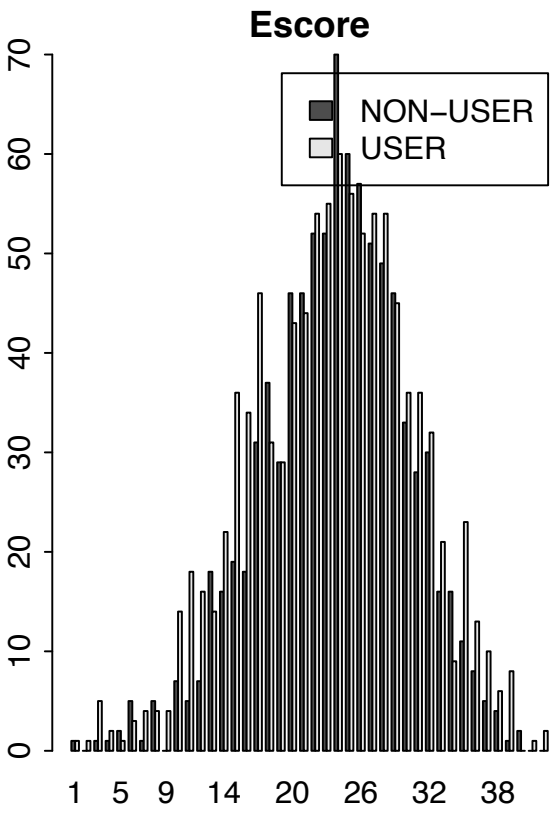

Alcohol
NON−USER
USER
0
100
200
300
CL0
CL2
CL4
CL6

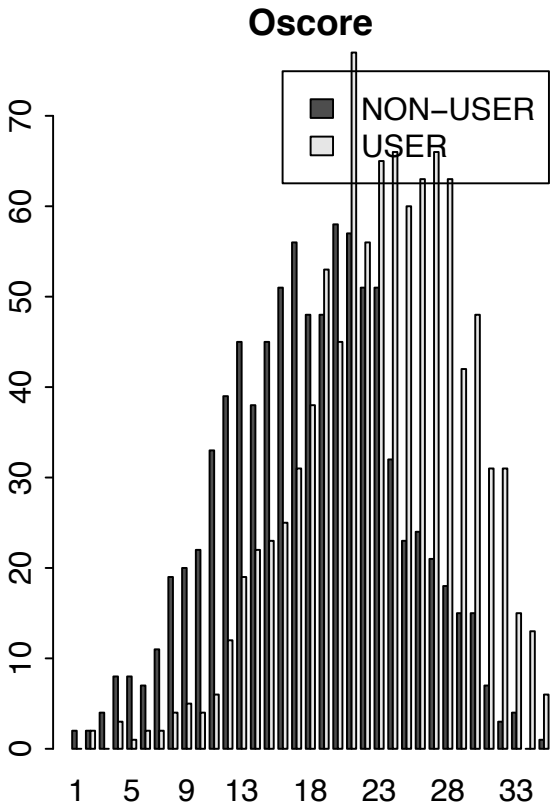

NON−USER
USER
CL0
CL1
CL2
CL3
CL4
CL5
CL6

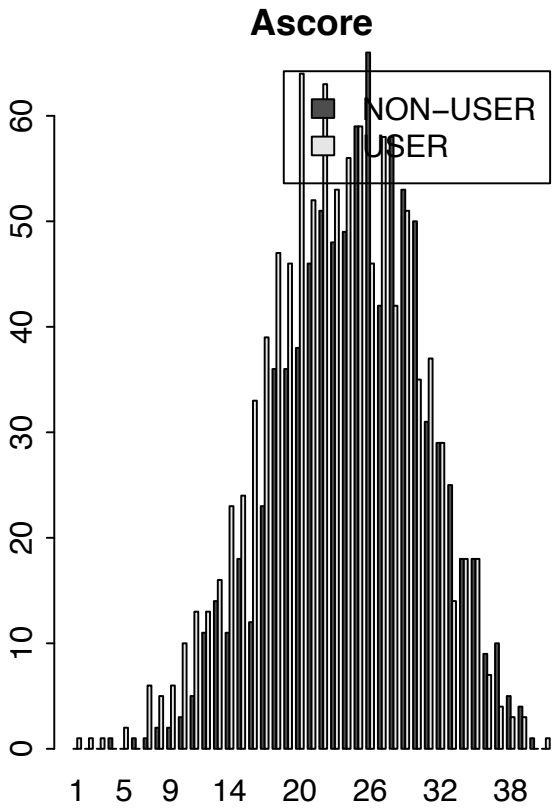


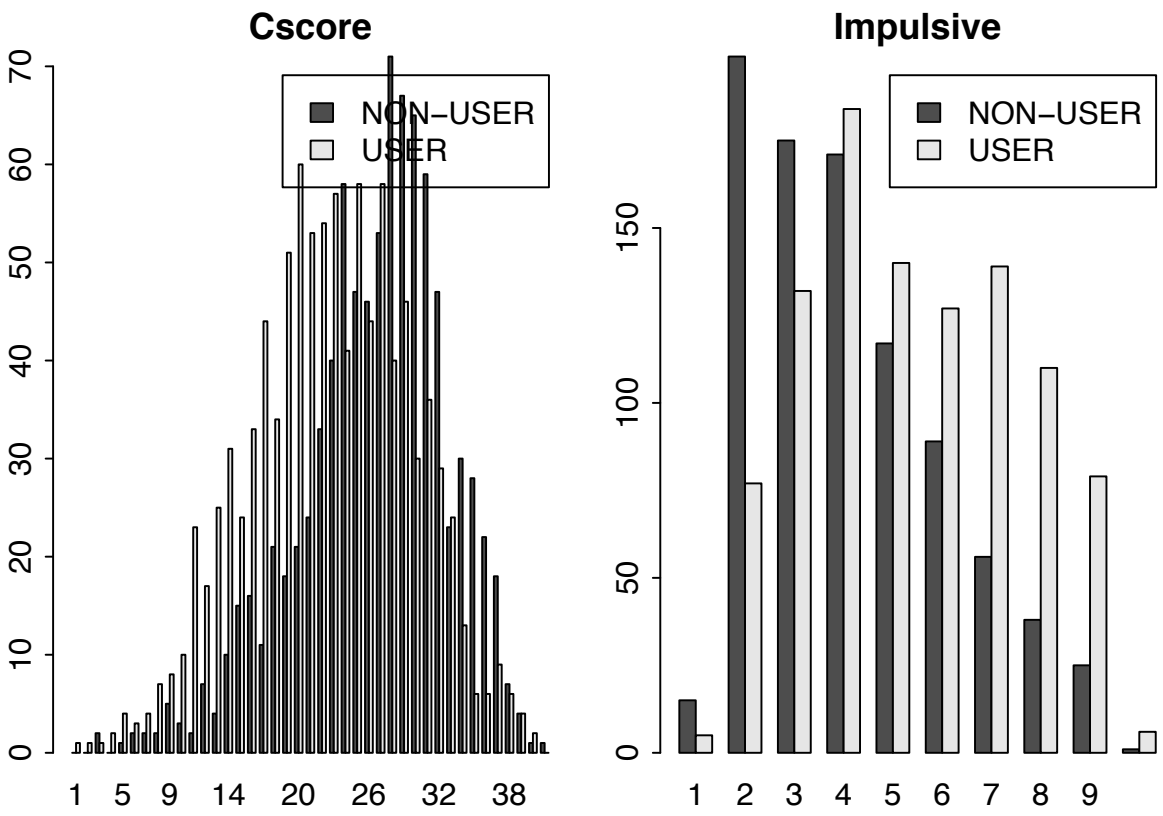

```
for(i in 6:12) {
   tab =  table(dd2$Cannabis, dd2[,i])
   barplot(tab, beside = TRUE, legend = levels(dd$Cannabis), main = colnames(dd2)[i])
}
```

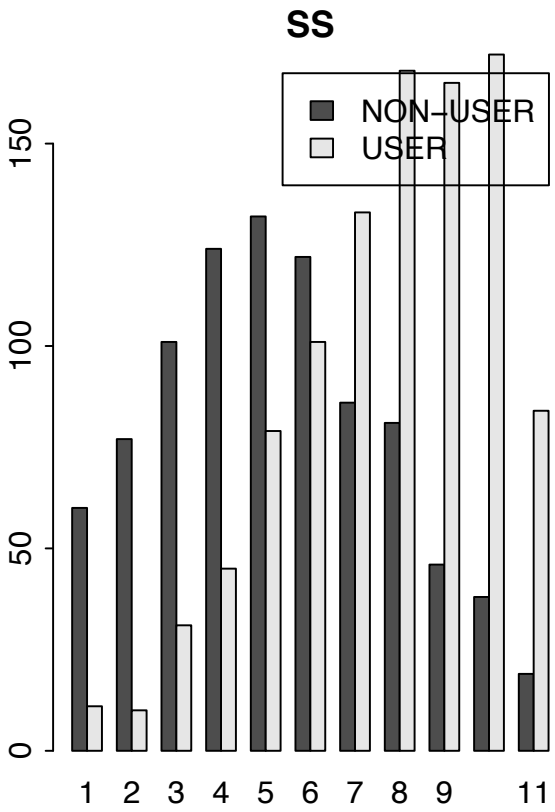

Nscore
NON−USER
USER
0
10
20
30
40
50
1
6
12
19
26
33
40
47


Escore
NON−USER
USER
0
10
20
30
40
50
60
70
1
5
9
14
20
26
32
38


Oscore
NON−USER
USER
0
10
20
30
40
50
60
70
1
5
9
13
18
23
28
33

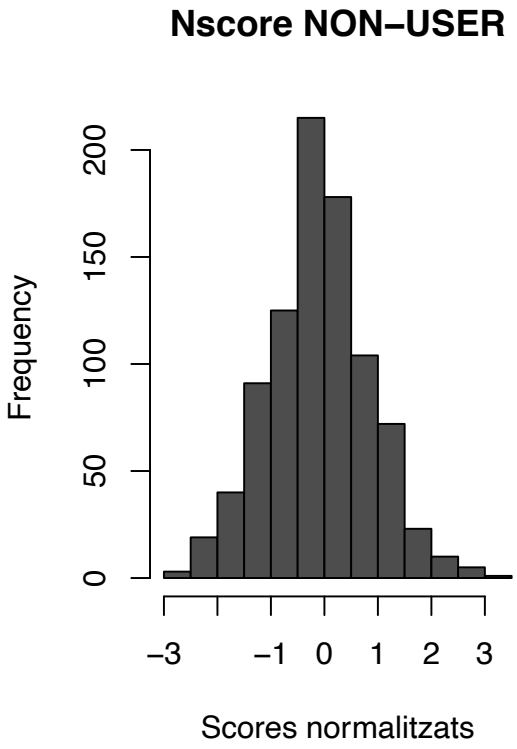

Ascore
NON−USER
USER
0
10
20
30
40
50
60
1
5
9
14
20
26
32
38

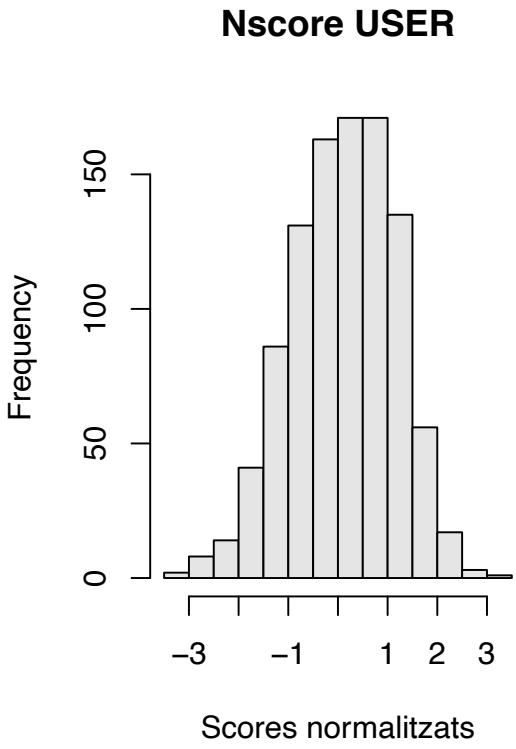

Cscore
NON−USER
USER
0
10
20
30
40
50
60
70
1
5
9
14
20
26
32
38
Impulsive
NON−USER
USER
0
50
100
150
1
2
3
4
5
6
7
8
9

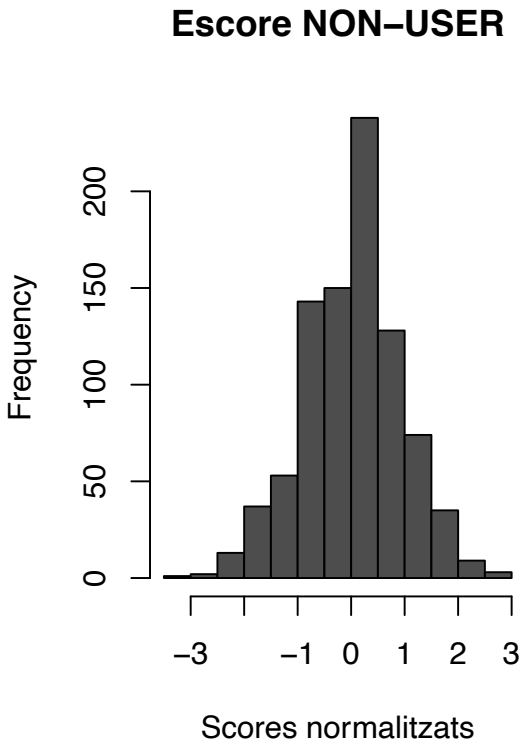


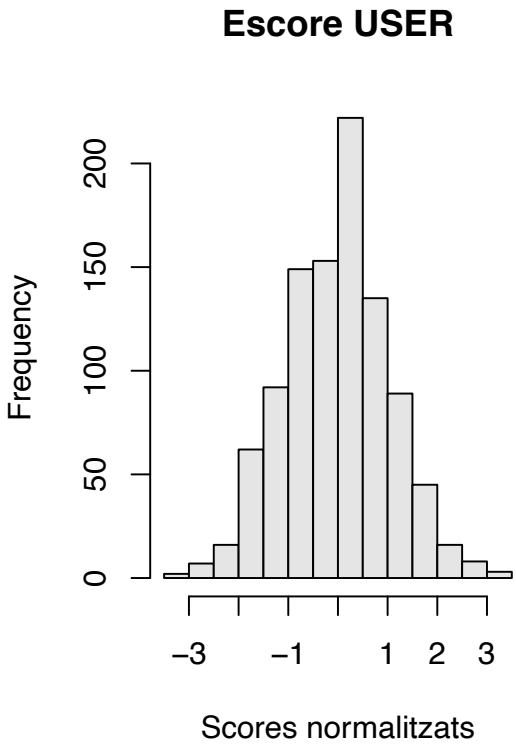
1.3 Histogrames scores

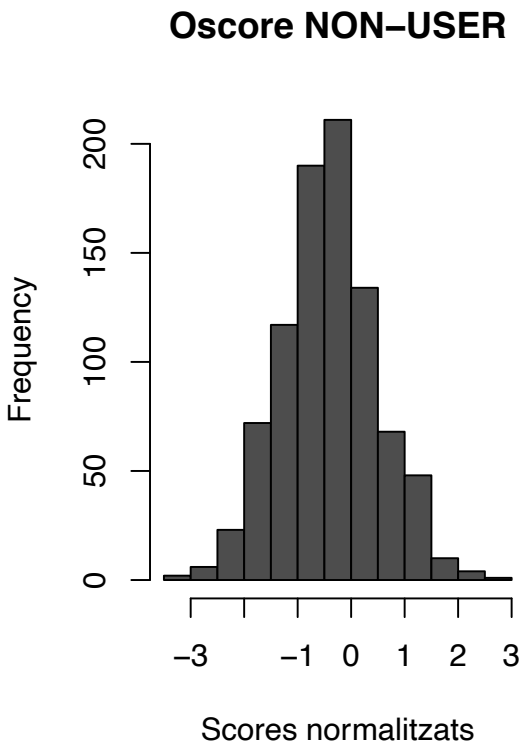
```
par(mfrow = c(1,2))
v = which(dd[, "Cannabis"] == "USER")
for (i in 6:10){
   hist(dd[-v, i], main = paste(colnames(dd)[i], "NON-USER"), col = "grey30",
        xlab = "Scores normalitzats")
   hist(dd[v, i], main = paste(colnames(dd)[i], "USER"), col = "grey90",
        xlab = "Scores normalitzats")
}
```

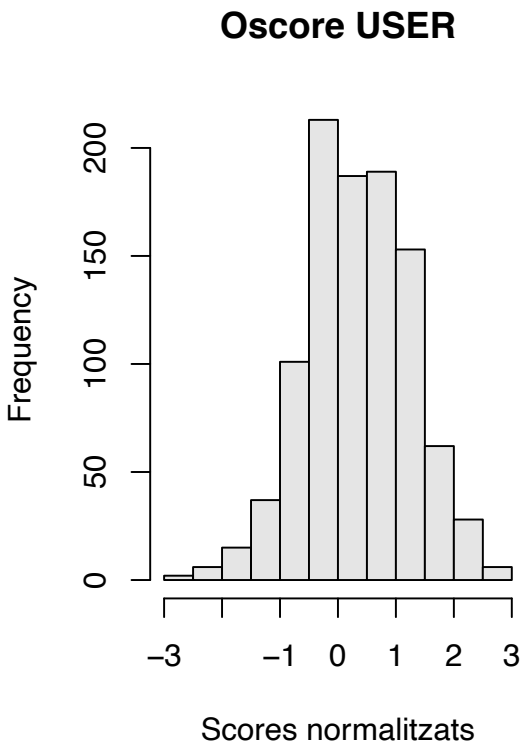

Nscore NON−USER
Frequency
0
50
100
150
200
−3
−1
0
1
2
3
Scores normalitzats

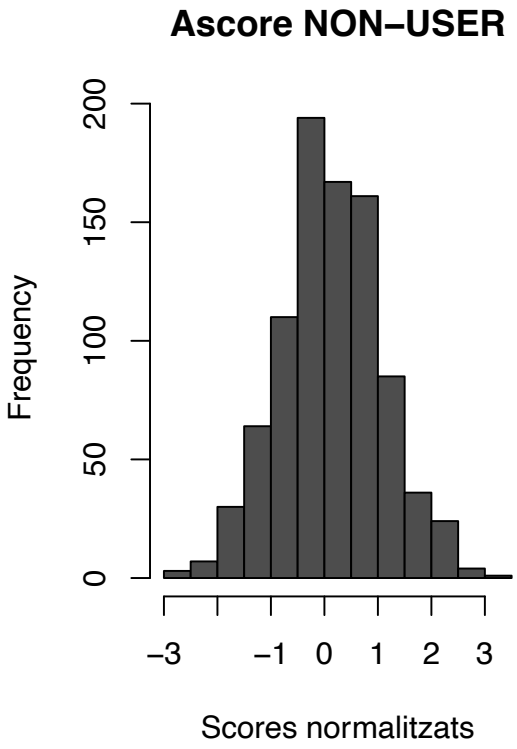

Nscore USER
Frequency
0
50
100
150
−3
−1
1
2
3
Scores normalitzats

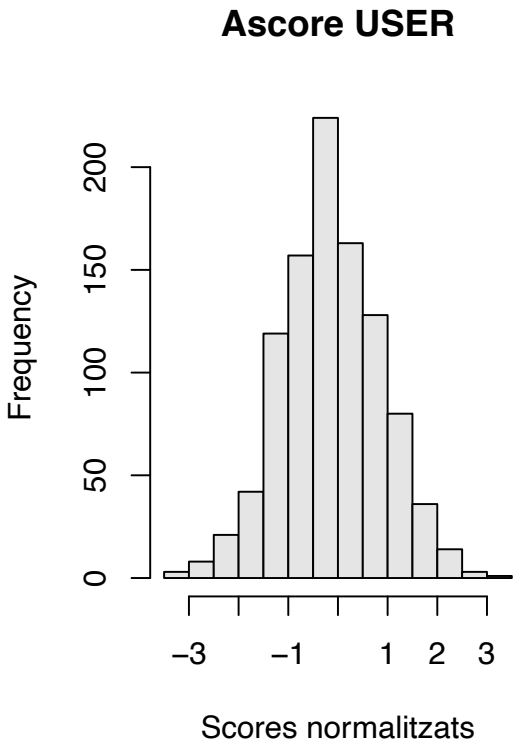

Escore NON−USER
Frequency
0
50
100
150
200
−3
−1
0
1
2
3
Scores normalitzats

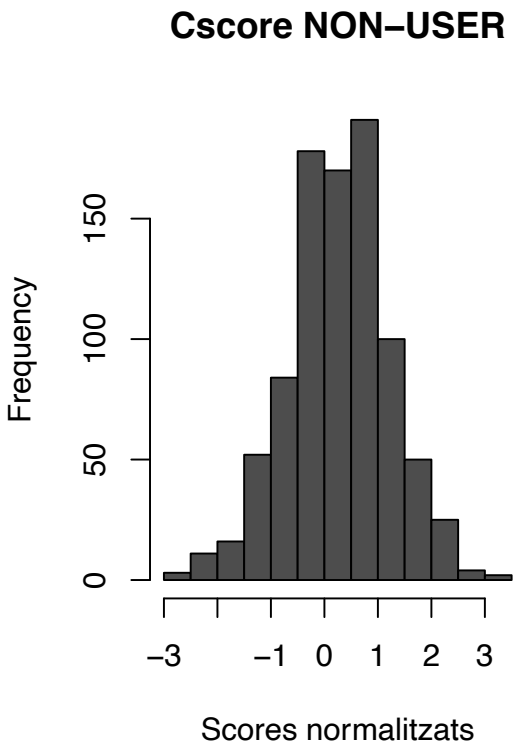

Escore USER
Frequency
0
50
100
150
200
−3
−1
1
2
3
Scores normalitzats

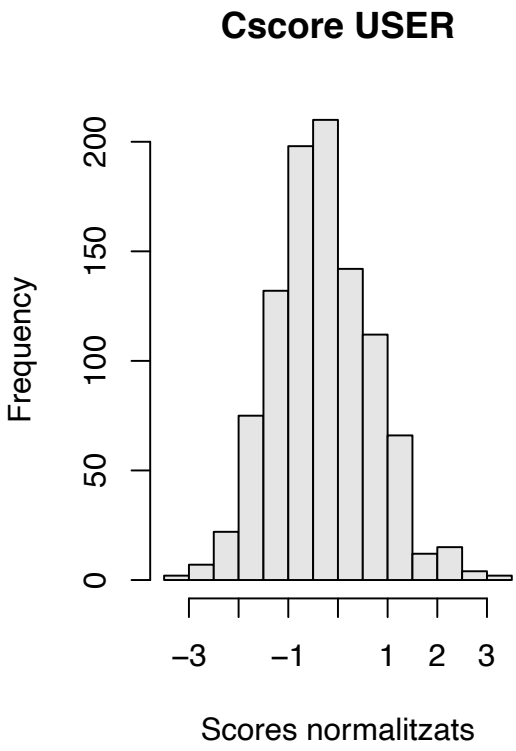

Oscore NON−USER
Frequency
0
50
100
150
200
−3
−1
0
1
2
3
Scores normalitzats


Oscore USER
Frequency
0
50
100
150
200
−3
−1
0
1
2
3
Scores normalitzats


Ascore NON−USER
Frequency
0
50
100
150
200
−3
−1
0
1
2
3
Scores normalitzats

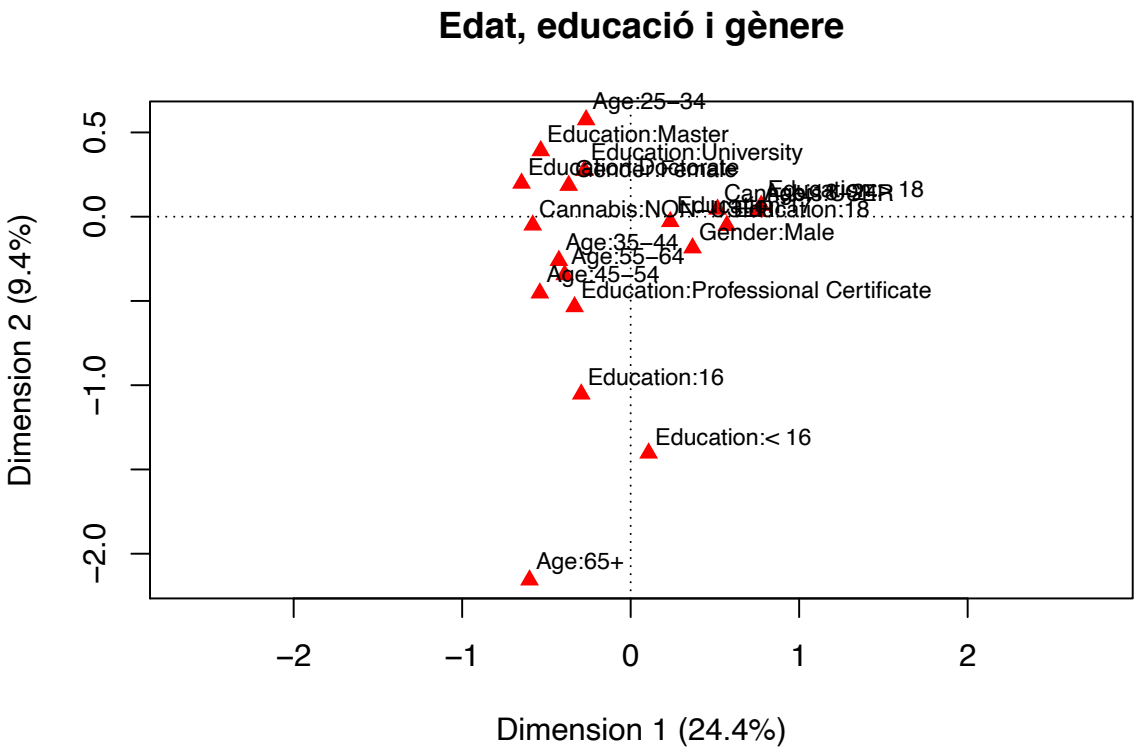

Ascore USER
Frequency
0
50
100
150
200
−3
−1
1
2
3
Scores normalitzats



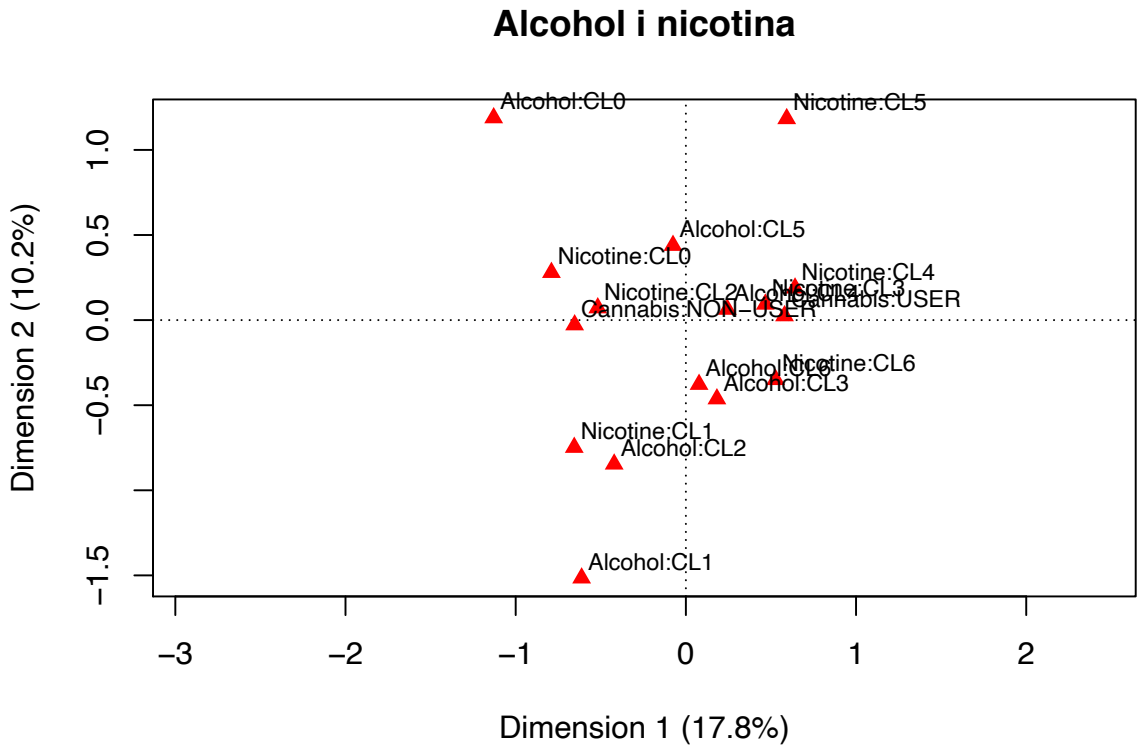


1.4 Anàlisi de correspondències

```
ca = mjca(dd2[,c(2:4, 18)], lambda = "Burt")
plot(ca, main = "Edat, educació i gènere")
```



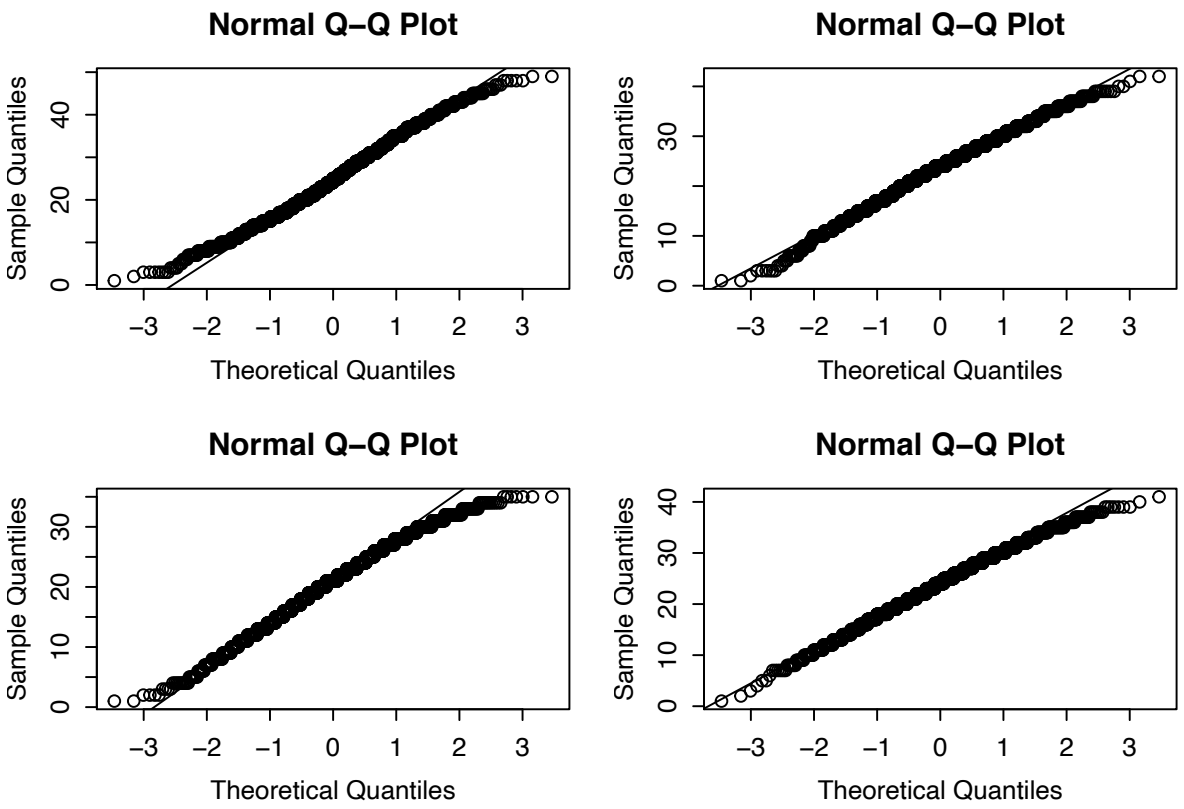

```
ca = mjca(dd2[,c(13, 29, 18)], lambda = "Burt")
plot(ca, main = "Alcohol i nicotina")
```

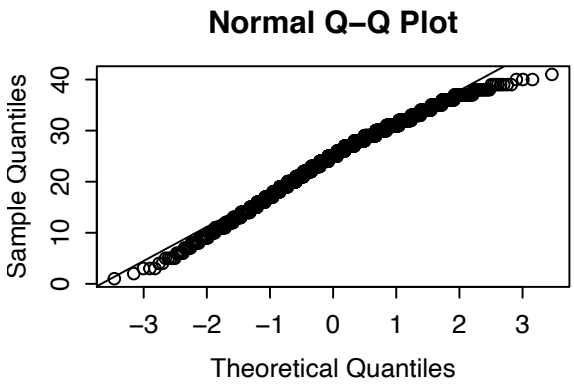


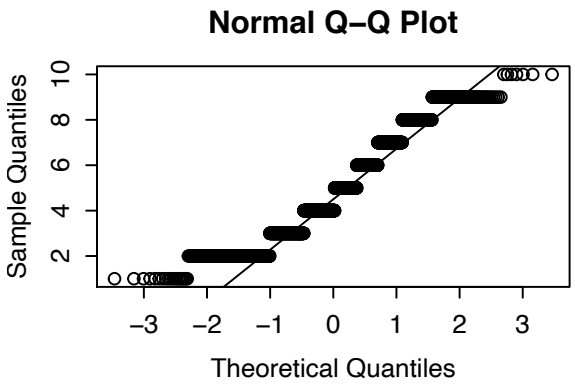
1.5 Test de Wilcox per les variables de personalitat

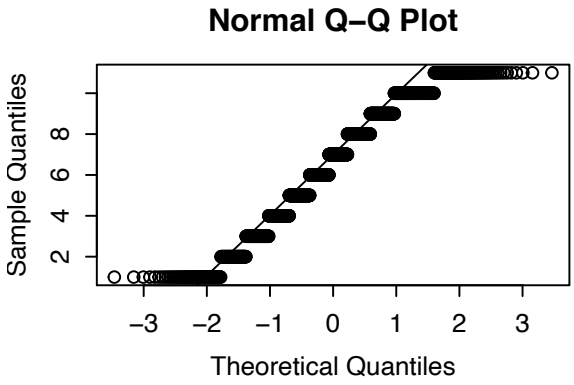
1.5.1 Normalitat variables de personalitat

```
par(mfrow = c(2, 2), mex = 0.7)

for (i in 6:12) {
   qqnorm(dd2[, i])
   qqline(dd2[, i])
}
```


Normal Q–Q Plot
Sample Quantiles
Theoretical Quantiles
Normal Q–Q Plot
Sample Quantiles
Theoretical Quantiles
Normal Q–Q Plot
Sample Quantiles
Theoretical Quantiles
Normal Q–Q Plot
Sample Quantiles
Theoretical Quantiles








```
for (i in 6:12){
   print(wilcox.test(dd[, i] ~ dd$Cannabis))
}
```

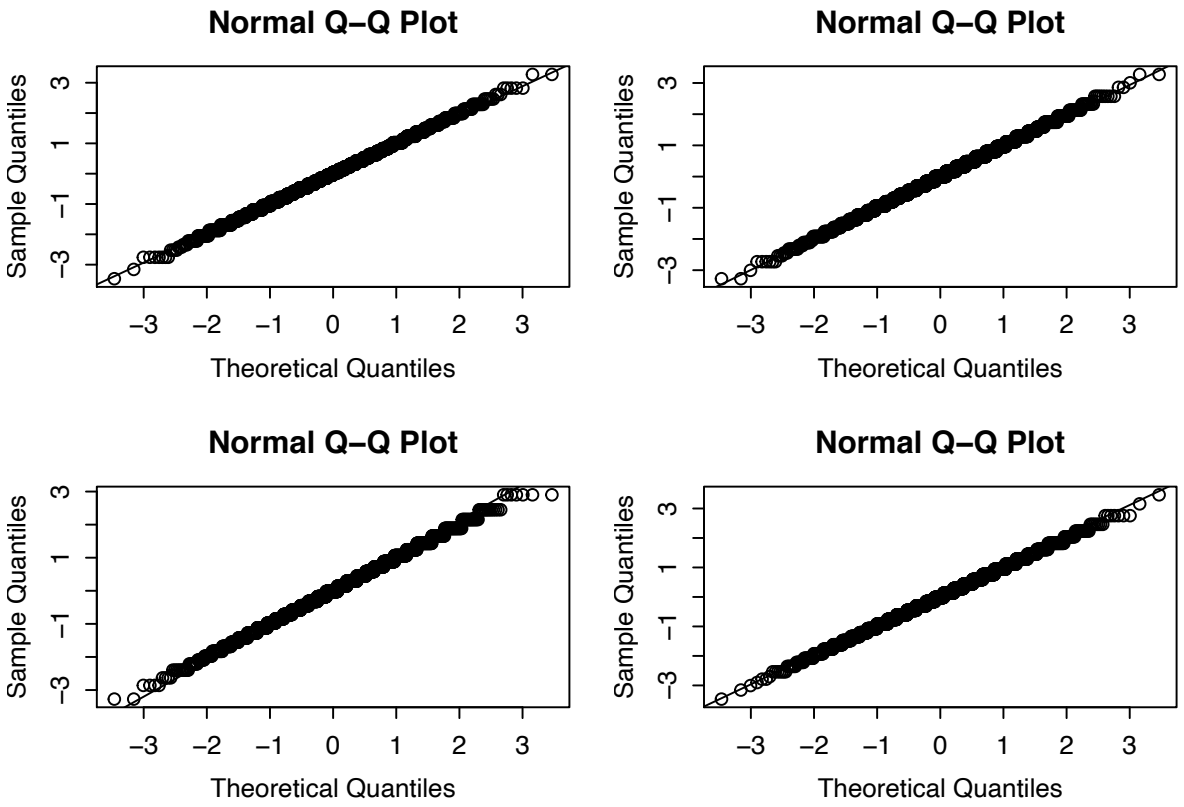
```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  dd[, i] by dd$Cannabis
## W = 380418, p-value = 1.355e-07
## alternative hypothesis: true location shift is not equal to 0
##
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  dd[, i] by dd$Cannabis
## W = 455765, p-value = 0.2623
## alternative hypothesis: true location shift is not equal to 0
##
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  dd[, i] by dd$Cannabis
## W = 240522, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
##
##
```

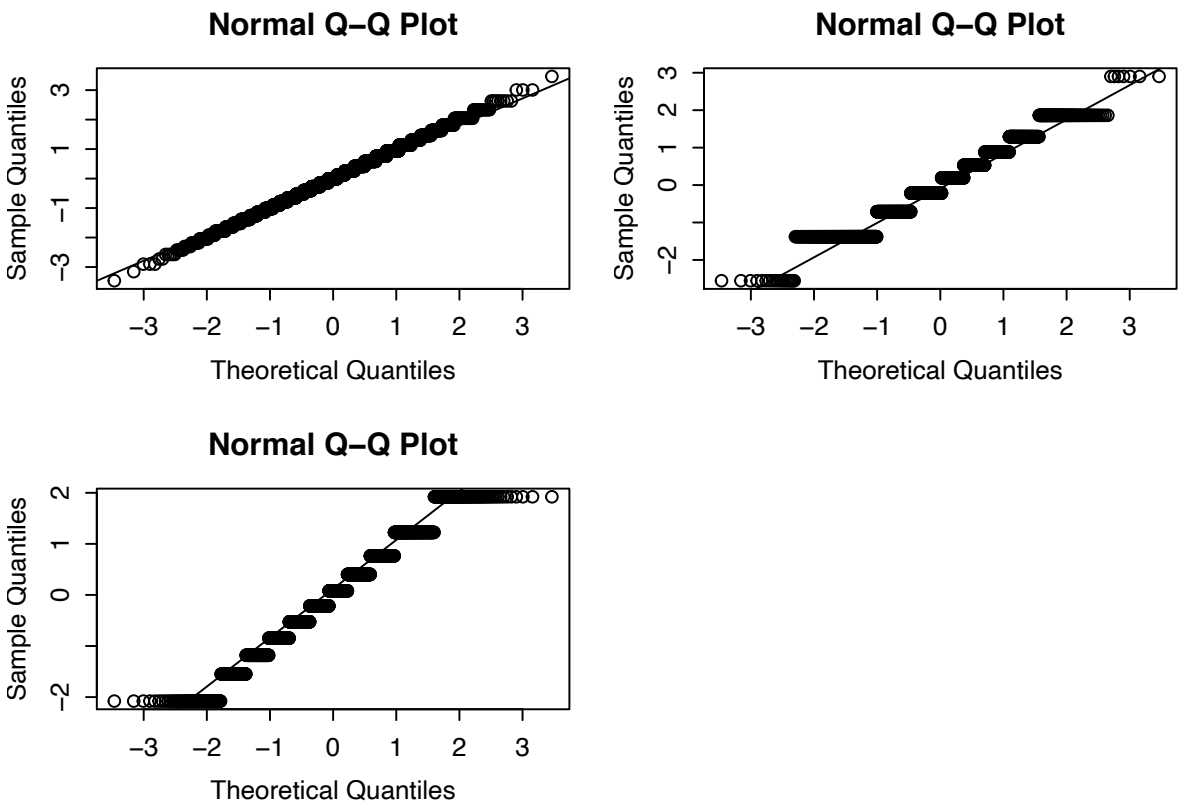
```
##  Wilcoxon rank sum test with continuity correction
##
## data:  dd[, i] by dd$Cannabis
## W = 507898, p-value = 2.923e-08
## alternative hypothesis: true location shift is not equal to 0
##
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  dd[, i] by dd$Cannabis
## W = 593386, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
##
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  dd[, i] by dd$Cannabis
## W = 293817, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
##
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  dd[, i] by dd$Cannabis
## W = 209760, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

1.6 Estudi normalitat per a l'ús de QDA i LDA

Normalitat variable per variable en la base de dades *dd* que és la base amb les transformacions a valors reals que serà la que s'utilitzarà per a la implementació de QDA i LDA.

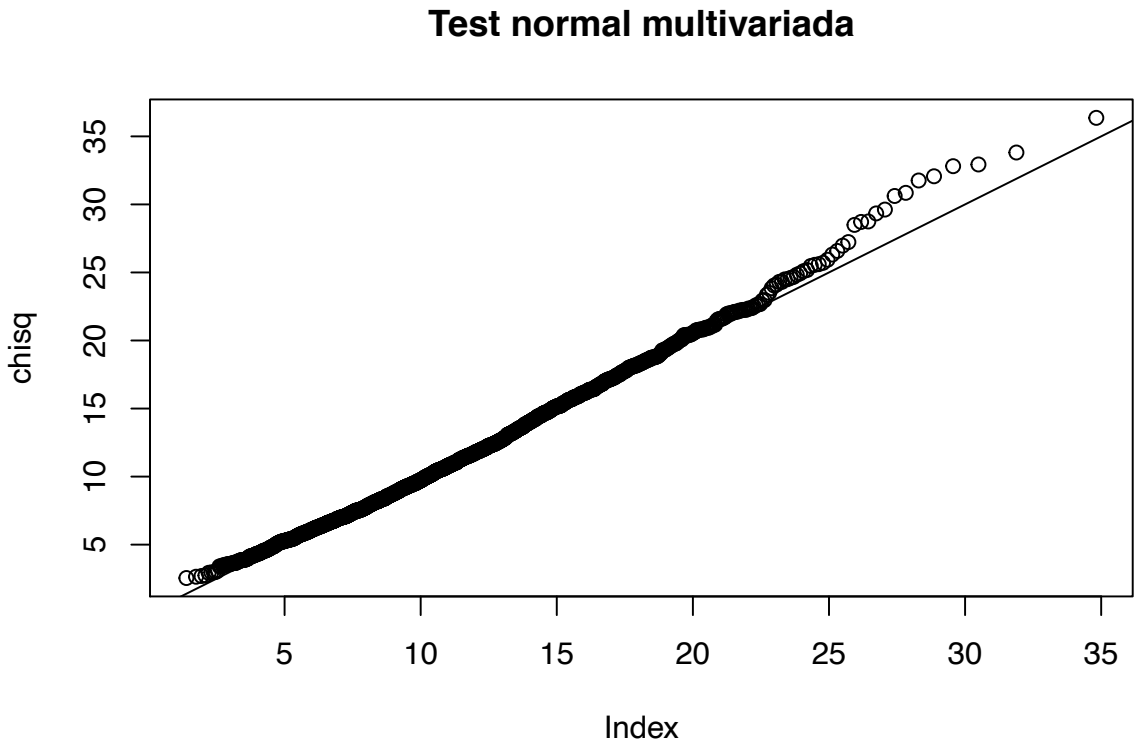
```
par(mfrow = c(2, 2), mex = 0.7)

for (i in 6:12) {
   qqnorm(dd[, i])
   qqline(dd[, i])
}
```


Normal Q–Q Plot
Sample Quantiles
Theoretical Quantiles
−3
−2
−1
0
1
2
3



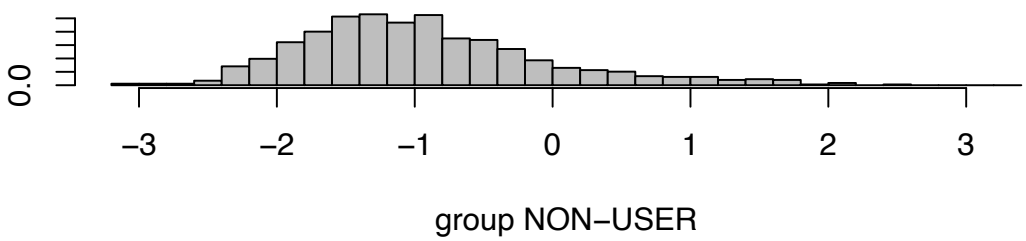
Normalitat multivariada de tot el conjunt de variables explicatives que conformen les dades d'entrenament.

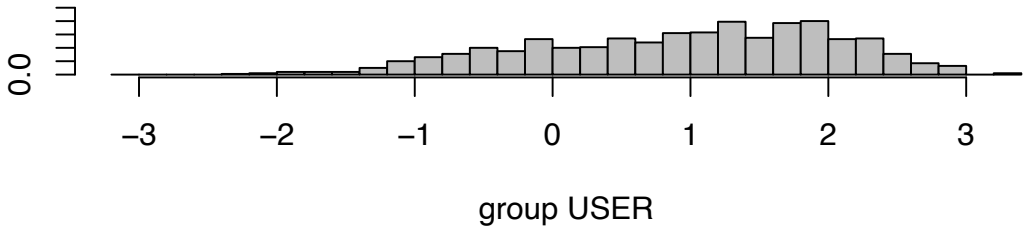
```
db = as.matrix(dd[,2:12])
S = cov(db) # Matriu de covariàncies
d = matrix(0, 1, nrow(db))
m = apply(db,2,mean)
# Mahanalobis
for (i in 1:nrow(db)){
   d[i] = t(db[i,] - m) %*% solve(S) %*% (db[i,] - m)
}
# Ordenem pel rank
d = sort(d);
# Correcció de ranks
aux = seq(1, nrow(db))
aux = (aux - 0.5) / nrow(db)
plot(qchisq(aux, ncol(db)), d,xlab = "Index", ylab = "chisq",
     main = "Test normal multivariada")
abline(0, 1)
```



1.6.1 Plot LDA

```
l = lda(dd[, c(2:12)], grouping = dd[, "Cannabis"])
plot(l, col = "grey")
```





2. Entrenament

2.1 Separació les dades en training y test

Traiem la variable *Escore* que hem detectat que és no significativa.

```
dd = dd[,-7]
dd2 = dd2[,-7]
```

```
n = nrow(dd)

set.seed(20)
samples = sample(n, round(0.75*n))  # Reservem el 25% de les dades per test
train = dd2[samples, c(2:12, 28, 17)]  # Dades de train amb categories
train.real = dd[samples, c(2:12, 28, 17)]  # Dades de training amb reals
test = dd2[-samples, c(2:12, 28, 17)]  # Dades de test amb categories
test.real = dd[-samples, c(2:12, 28, 17)]  # Dades de test amb reals

ntr = nrow(train)
nte = nrow(test)
```

Funcions auxiliars

2.2 Error QDA

Donat un conjunt de dades d'entrenament i de validació i les seves respectives respostes retorna l'error sobre el conjunt de validació havent creat un model amb QDA (suposició de matrius de covariàncies diferents) amb les dades d'entrenament.

```
error_qda = function (Xl, yl, Xv, yv, flag) {
   out = qda(yl ~ ., data = Xl)
   pred = predict(out, newdata = Xv)$class
   prediction = table(Truth = yv, Pred = pred)
   if (flag) print(prediction)
   err = 1 - sum(diag(prediction))/length(yv)
}
```

2.3 Error LDA

Donat un conjunt de dades d'entrenament i de validació i les seves respectives respostes retorna l'error sobre el conjunt de validació havent creat un model amb LDA (suposició de matrius de covariàncies iguals) amb les dades d'entrenament.

```
error_lda = function (Xl, yl, Xv, yv, flag) {
   out = lda(yl ~ ., data = Xl)
   pred = predict(out, newdata = Xv)$class
   prediction = table(Truth = yv, Pred = pred)
   if (flag) print(prediction)
   err = 1 - sum(diag(prediction))/length(yv)
}
```

2.4 Error GLM

Aquesta funció genera un model lineal generaltizat binomial amb link inidicat al paràmetre d'entrada *linkage* amb les dades d'entrenament i retorna l'error comés sobre el conjunt de dades de validació. De nou incloem un *flag* que ens indica si volem fer el print de taula de confusió o no.

```
error_glm = function (Xl, yl, Xv, yv, linkage, flag) {
   # Generació del model amb les dades de learn
   mod = glm(yl ~ ., family = binomial(link = linkage), data = as.data.frame(Xl))

   # Predicció sobre les dades de validació
   PI = predict(mod, newdata = as.data.frame(Xv), type = "response")
   PI = as.factor(round(PI))
   levels(PI) = c("USER", "NON-USER")
   prediction = table(Truth = yv, Pred = PI)
   if (flag) print(prediction)
   err = 1 - sum(diag(prediction))/length(yv)
}
```

2.5 Error Naive-Bayes

Retorna l'error de predicció sobre el conjunt de dades de validació entrenant un model Naive-Bayes per classificar amb les dades d'entrenament passades com a entrada.

```
error_naivebayes = function (Xl, yl, Xv, yv, flag) {
   # Generació del model amb les dades de learn
   mod = naive_bayes(Xl, yl, laplace = 0.1)

   # Predicció sobre les dades de validació
   PI = predict(mod, newdata = Xv, type = "class")
   prediction = table(Truth = yv, Pred = PI)
   if (flag) print(prediction)
   err = 1 - sum(diag(prediction))/length(yv)
}
```

2.6 Multilayer perceptron

Donat un conjunt de dades de train i de test retorna l'error sobre el conjunt de test en implementar un *Multilayer perceptron* amb una única capa oculta amb el nombre de neurones i *decay* indicats a l'entrada.

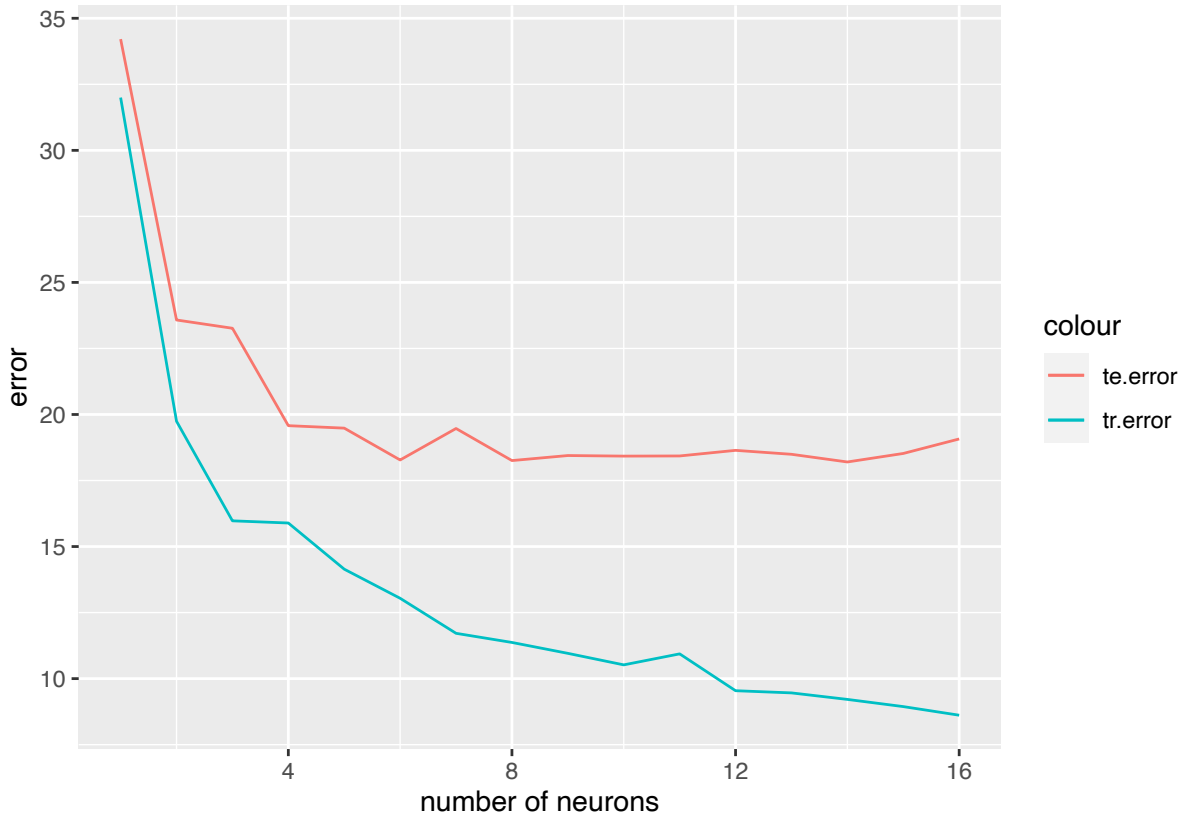
```
error_nnet = function (Xl, yl, Xv, yv, size, decay, flag) {
   out = nnet(yl ~ ., data = Xl, size = size, decay = decay, maxit = 10000, trace = F, MaxNWts = 10000)
   pred = predict(out, newdata = Xv, type = "class")
   prediction = table(Truth = yv, Pred = pred)
   if (flag) print(prediction)
   err = 1 - sum(diag(prediction))/length(yv)
}
```

2.6.1 Elecció hiperparàmetres nnet

Fem un plot de l'error de training i de validació en fer un 10-times 10-fold cross validation modificant el número de neurones de la capa oculta de la xarxa.

```
sizes = seq(1, 16, 1)

tr.error = rep(0, 16)
te.error = rep(0, 16)
cont = 1

for (k in sizes) {
   err = c("training" = 0, "test" = 0)
   for (i in 1:10) {
      samples = sample(ntr)  # Barrejem aleatòriament les dades de training
      for (j in 1:10) {
         start = round((j - 1)*ntr/10 + 1) # Índex inici
         end = round(j * ntr/10)  # Índex final

         # Separem en variables explicatives i resposta / learn i validation
         Xl = train[samples[-(start:end)], -13]
         yl = train[samples[-(start:end)], 13]
         Xv = train[samples[start:end], -13]
```

```
        yv = train[samples[start:end], 13]

        err["test"] = err["test"] + error_nnet(Xl, yl, Xv, yv, k, 0, F)
        err["training"] = err["training"] + error_nnet(Xl, yl, Xl, yl, k, 0, F)
      }
    }
    tr.error[cont] = err["training"]
    te.error[cont] = err["test"]
    cont = cont + 1
}
ggplot(as.data.frame(tr.error), aes(sizes)) +
  geom_line(aes(y = tr.error, colour = "tr.error")) +
  geom_line(aes(y = te.error, colour = "te.error")) +
  labs(x = "number of neurons", y = "error")
```




```
best.size = 10
```

Agafem 10 neurones ja que observem que a partir d'aquest punt sobreparametritzem. Fixem ara el nombre de neurones i busquem el decay òptim de nou amb un 10-times 10-fold cross validation.

```
set.seed(41)
decays = seq(1.5, 2.5, by = 0.1)
trc = trainControl(method = "repeatedcv", number = 10, repeats = 10)

model.10x10CV = train(Cannabis ~ ., data = train,
```

```
                        method = 'nnet', maxit = 1000, trace = FALSE,
                       tuneGrid = expand.grid(.size = best.size, .decay = decays), trControl = trc)
(decay = model.10x10CV$bestTune$decay)
```

```
## [1] 2.5
```

```
model.10x10CV
```

```
## Neural Network
##
## 1414 samples
##   12 predictor
##    2 classes: 'NON-USER', 'USER'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 1272, 1272, 1273, 1272, 1273, 1273, ...
## Resampling results across tuning parameters:
##
##   decay  Accuracy   Kappa
##   1.5    0.8385296  0.6765912
##   1.6    0.8379607  0.6754303
##   1.7    0.8390196  0.6775250
##   1.8    0.8394521  0.6784082
##   1.9    0.8381795  0.6758508
##   2.0    0.8381810  0.6758625
##   2.1    0.8379682  0.6754402
##   2.2    0.8388153  0.6770783
##   2.3    0.8386735  0.6768058
##   2.4    0.8380367  0.6755508
##   2.5    0.8395914  0.6786145
##
## Tuning parameter 'size' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 10 and decay = 2.5.
```

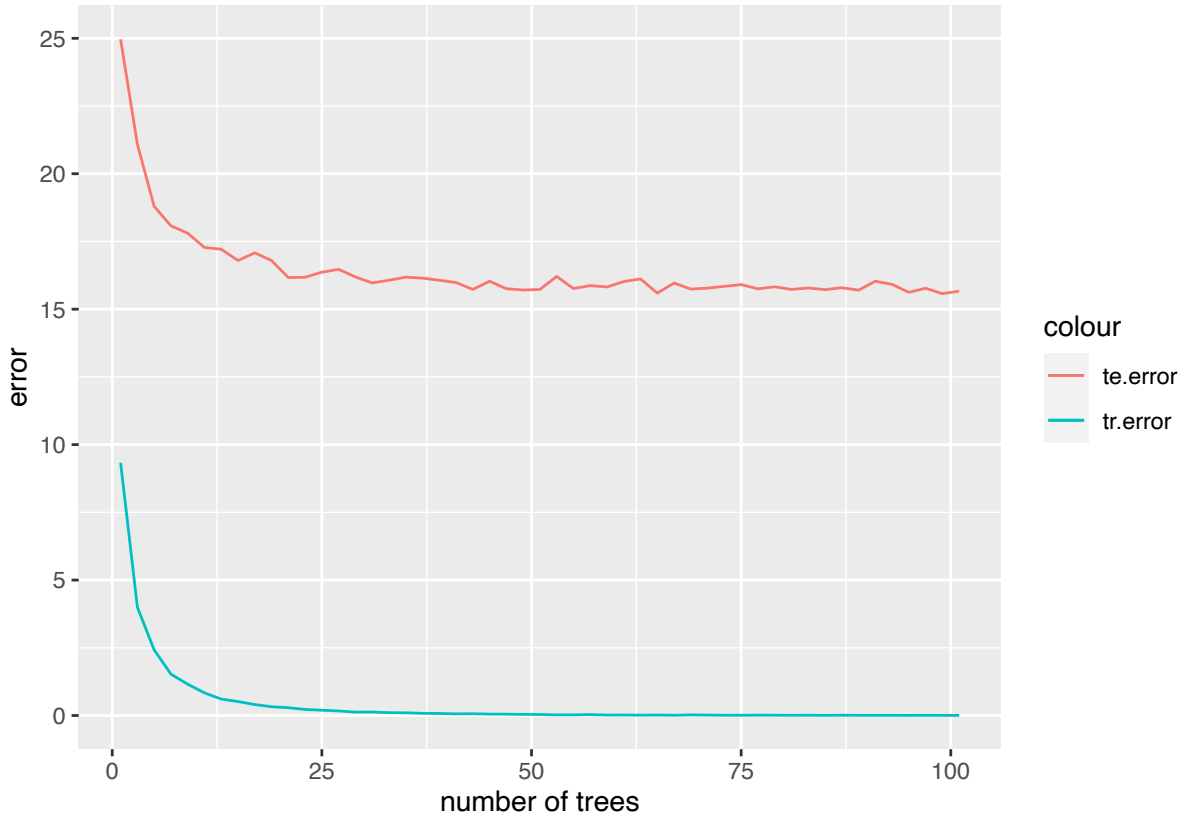
2.7 Random forest

Aquest funció retorna l'error en ajustar un model amb un Random Forest de mida k.

```
error_randomforest = function (Xl, yl, Xv, yv, k, flag) {
   rf = randomForest (yl ~ ., data = as.data.frame(Xl), ntree = k, proximity = FALSE)
   pred = predict(rf, newdata = Xv, type = "class")
   prediction = table(Truth = yv, Pred = pred)
   if (flag) print(prediction)
   err = 1 - sum(diag(prediction))/length(yv)
}
```

Fem un plot de l'error de training i de validació en fer 10-times 10-fold cross validation en canviar el nombre d'arbres d'un Random Forest.

```
set.seed(12)
ntrees = seq(1, 101, 2)
min_err = 100

tr.error = rep(0, 51)
te.error = rep(0, 51)
cont = 1

for (k in ntrees) {
   err = c("training" = 0, "test" = 0)
   for (i in 1:10) {
      samples = sample(ntr)  # Barrejem aleatòriament les dades de training
      for (j in 1:10) {
         start = round((j - 1)*ntr/10 + 1)  # Índex inici
         end = round(j * ntr/10)  # Índex final

         # Separem en variables explicatives i resposta / learn i validation
         Xl = train[samples[-(start:end)], -13]
         yl = train[samples[-(start:end)], 13]
         Xv = train[samples[start:end], -13]
         yv = train[samples[start:end], 13]

         err["test"] = err["test"] + error_randomforest(Xl, yl, Xv, yv, k, F)
         err["training"] = err["training"] + error_randomforest(Xl, yl, Xl, yl, k, F)
      }
   }
   tr.error[cont] = err["training"]
   te.error[cont] = err["test"]
   cont = cont + 1
}

ggplot(as.data.frame(tr.error), aes(ntrees)) +
  geom_line(aes(y = tr.error, colour = "tr.error")) +
  geom_line(aes(y = te.error, colour = "te.error")) +
  labs(x = "number of trees", y = "error")
```




```
best.ntree = 50
```

2.8 Cross-validation

Fem un 10 times 10-fold cross validation per avaluar quin dels models proposats és el millor. En aquest cas hem usat *QDA*, *LDA*, *GLM link logit*, *GLM link probit*, *GLM link loglog*, *Naive Bayes*, *MLP* i *Random Forest*.

```
set.seed(5)
error = c("QDA" = 0, "LDA" = 0, "GLM-logit" = 0, "GLM-probit" = 0, "GLM-loglog" = 0,
          "Naive-Bayes" = 0, "Nnet" = 0, "Random Forest"  = 0)

for (i in 1:10) {
   samples = sample(ntr)  # Barrejem aleatòriament les dades de training
   for (j in 1:10) {
      start = round((j - 1)*ntr/10 + 1)  # Índex d'inici del j-èssim-fold validation test
      end = round(j * ntr/10)  # Índex final

      # Separem en variables explicatives i resposta / learn i validation
      # 1. Training i validation amb la base de dades amb categòriques
      Xl = train[samples[-(start:end)], -13]
      yl = train[samples[-(start:end)], 13]
      Xv = train[samples[start:end], -13]
      yv = train[samples[start:end], 13]
```

```
      # 2. Raining i validation amb la base de dades amb reals
      Xl.r = train.real[samples[-(start:end)], -(11:13)]
      yl.r = train.real[samples[-(start:end)], 13]
      Xv.r = train.real[samples[start:end], -(11:13)]
      yv.r = train.real[samples[start:end], 13]

      # Actualització errors
      error["QDA"] = error["QDA"] + error_qda(Xl.r, yl.r, Xv.r, yv.r, F)
      error["LDA"] = error["LDA"] + error_lda(Xl.r, yl.r, Xv.r, yv.r, F)
      error["GLM-logit"] = error["GLM-logit"] + error_glm(Xl, yl, Xv, yv, "logit", F)
      error["GLM-probit"] = error["GLM-probit"] + error_glm(Xl, yl, Xv, yv, "probit", F)
      error["GLM-loglog"] = error["GLM-loglog"] + error_glm(Xl, yl, Xv, yv, "cloglog", F)
      error["Naive-Bayes"] = error["Naive-Bayes"] + error_naivebayes(Xl, yl, Xv, yv, F)
      error["Nnet"] = error["Nnet"] + error_nnet(Xl, yl, Xv, yv, best.size, decay, F)
      error["Random Forest"] = error["Random Forest"] + error_randomforest(Xl, yl, Xv, yv,
                                                                           best.ntree, F)
   }
}
error
```

```
##           QDA           LDA     GLM-logit    GLM-probit    GLM-loglog
##      19.57712      18.45964      15.95485      15.96883      17.12187
##   Naive-Bayes          Nnet Random Forest
##      17.30591      16.04041      15.91295
```

2.9 Error sobre les dades de test

```
set.seed(0415)
droga = "Cannabis"

print("GLM")
```

```
## [1] "GLM"
```

```
a = cbind("te. error", error_glm(train[, -13], train[, droga], test[, -13],
                                 test[, droga], "logit", T))
```

```
##           Pred
## Truth     NON-USER  USER
##   NON-USER  174       32
##   USER       41      224
```

```
a = rbind(a, cbind("tr. error", error_glm(train[, -13], train[, droga], train[, -13],
                                          train[, droga], "logit", T)))
```

```
##           Pred
## Truth     NON-USER  USER
##   NON-USER  587       93
##   USER      104      630
```

```
a
```

```
##      [,1]        [,2]
## [1,] "te. error" "0.154989384288747"
## [2,] "tr. error" "0.139321074964639"
```

```
print("Naive Bayes")
```

```
## [1] "Naive Bayes"
```

```
a = cbind("te. error", error_naivebayes(train[, -13], train[, droga], test[, -13],
                                        test[, droga], T))
```

```
##           Pred
## Truth      NON-USER USER
##   NON-USER      171   35
##   USER           48  217
```

```
a = rbind(cbind(a, error_naivebayes(train[, -13], train[, droga], train[, -13],
                                    train[, droga], T)))
```

```
##           Pred
## Truth      NON-USER USER
##   NON-USER      574  106
##   USER          131  603
```

```
a
```

```
##      [,1]        [,2]                [,3]
## [1,] "te. error" "0.176220806794055" "0.167609618104668"
```

```
print("nnet")
```

```
## [1] "nnet"
```

```
a = cbind("te. error", error_nnet(train[, -13], train[, droga], test[, -13],
                                  test[, droga], best.size, decay, T))
```

```
##           Pred
## Truth      NON-USER USER
##   NON-USER      170   36
##   USER           41  224
```

```
a = rbind(a, cbind("tr. error", error_nnet(train[, -13], train[, droga], train[, -13],
                                           train[, droga], best.size, decay, T)))
```

```
##           Pred
## Truth      NON-USER USER
##   NON-USER      583   97
##   USER           99  635
```

```
a
```

```
##      [,1]        [,2]
## [1,] "te. error" "0.16348195329087"
## [2,] "tr. error" "0.138613861386139"
```

```
print("Random Forest")
```

```
## [1] "Random Forest"
```

```
a = cbind("te. error", error_randomforest(train[, -13], train[, droga], test[, -13],
                                          test[, droga], best.ntree, T))
```

```
##           Pred
## Truth      NON-USER USER
##   NON-USER      167   39
##   USER           42  223
```

```
a = rbind(a, cbind("tr. error", error_randomforest(train[, -13], train[, droga],
                                                   train[, -13], train[, droga],
                                                   best.ntree, T)))
```

```
##           Pred
## Truth      NON-USER USER
##   NON-USER      680    0
##   USER            1  733
```

```
a
```

```
##      [,1]        [,2]
## [1,] "te. error" "0.171974522292994"
## [2,] "tr. error" "0.000707213578500676"
```