Machine Learning with R

Fernanda Eustáquio

Apresentação

- 2012.1 - 2017.1 Graduação em Ciência da Computação na Universidade Federal da Bahia (UFBA);
    - 2015.1 Iniciação Científica na área de Machine Learning;
    - 2016.2 Iniciação em R na disciplina de Laboratório em Inteligência Artificial;
- 2017.2 Mestrado em Ciência da Computação na área de Inteligência Computacional no Programa de Pós-graduação em Ciência da Computação (PGCOMP) da UFBA.

Machine Learning

- Pré-processamento;
- Extração de padrões dos dados;
  - Tarefas:
    - Supervisionadas:
      - Classificação;
      - Regressão.
    - Não-supervisionadas:
      - Regras de Associação;
      - Agrupamento.
- Pós-processamento.

Machine Learning
Pré-processamento

- Preparação dos dados:
  - Amostragem:
  - Balanceamento;
  - Limpeza:
    - Valores ausentes;
    - Ruído;
    - Outliers;
  - Transformação dos dados;
  - Redução de Dimensionalidade:
    - PCA;
    - MDS.

Pré-processamento
Amostragem

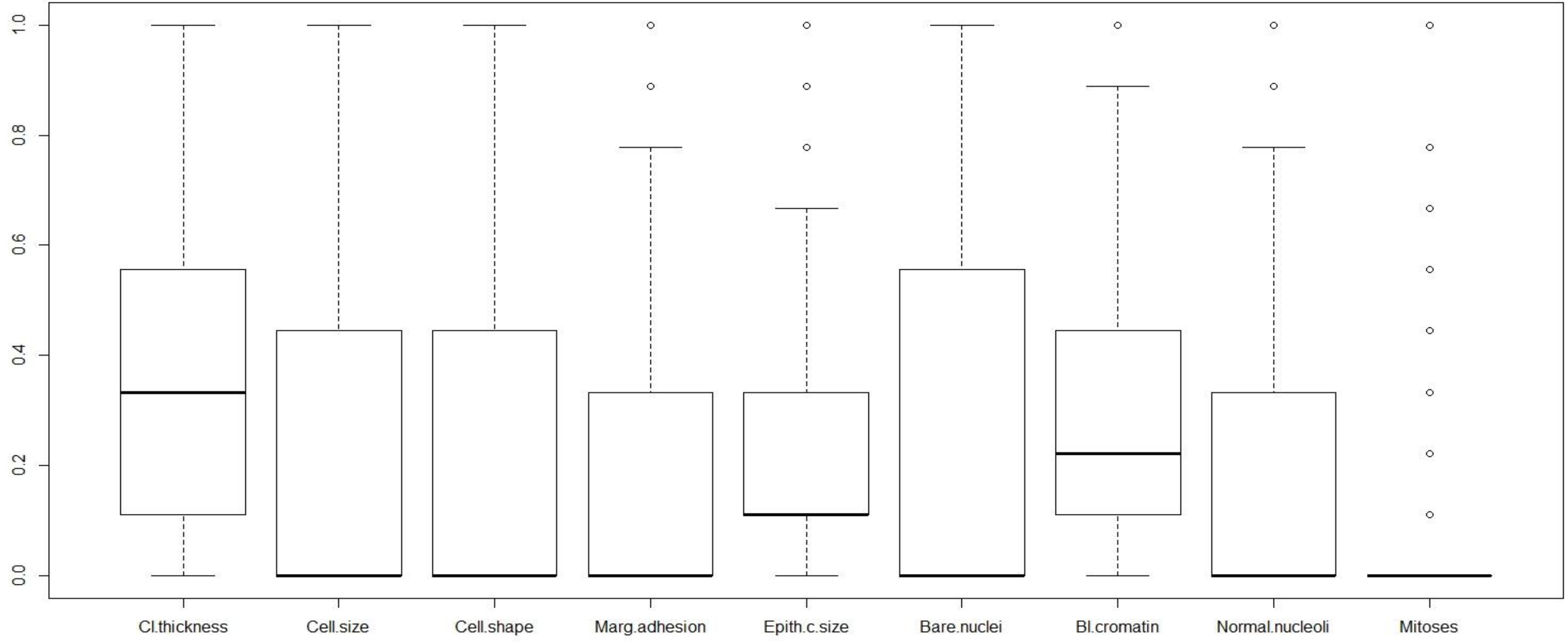
```
treino = sample(nrow(data), ceiling(nrow(data)*0.7));

data_treino = data[treino,];

data_teste = data[-treino,]
```

Pré-processamento
Limpeza - Valores ausentes

```
data = data[-which(is.na(data[,6]))]
```

Pré-processamento
Limpeza - Outliers

boxplot(data)



Pré-processamento
Transformação dos Dados

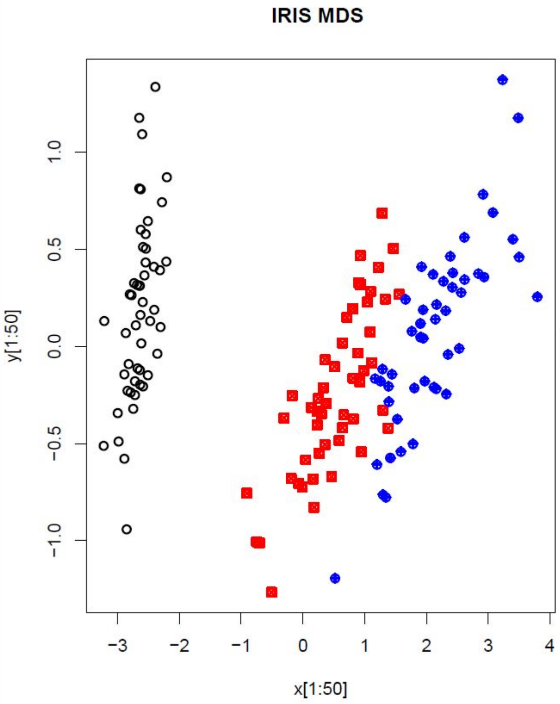
```r
levels(cstr[,ncol(cstr)])

[1] "ArtificialIntelligence" "Robotics"
[3] "Systems"                "Theory"

classnames = levels(cstr[,ncol(cstr)])

indices_AI = which(cstr[,ncol(cstr)]==classnames[1])
indices_Robotics = which(cstr[,ncol(cstr)]==classnames[2])
indices_Systems = which(cstr[,ncol(cstr)]==classnames[3])
indices_Theory = which(cstr[,ncol(cstr)]==classnames[4])

classes_numericas = rep(0, nrow(cstr))

classes_numericas[indices_AI] = 1
classes_numericas[indices_Robotics] = 2
classes_numericas[indices_Systems] = 3
classes_numericas[indices_Theory] = 4
```

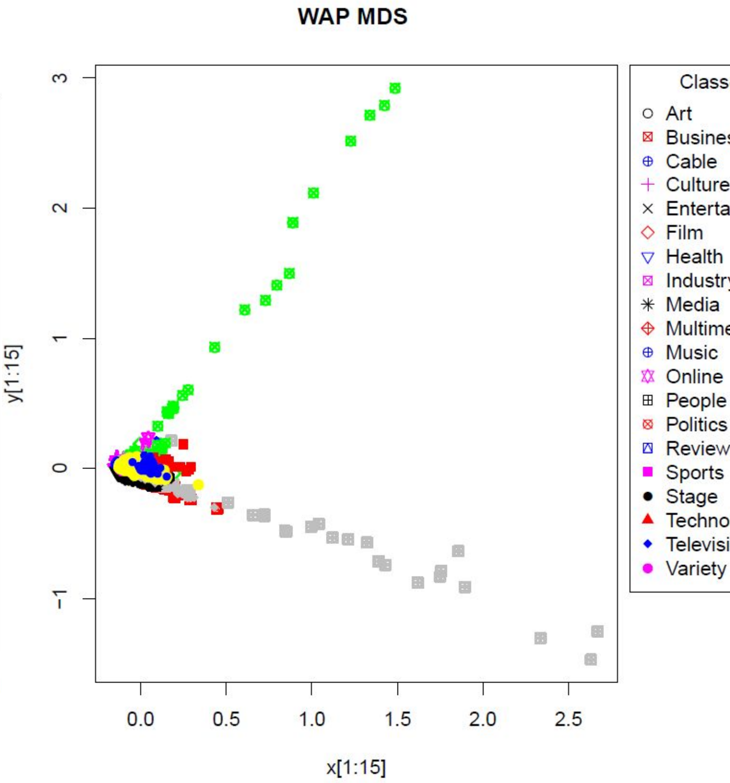
Pré-processamento

Redução de Dimensionalidade - PCA

Pré-processamento
Redução de Dimensionalidade - MDS

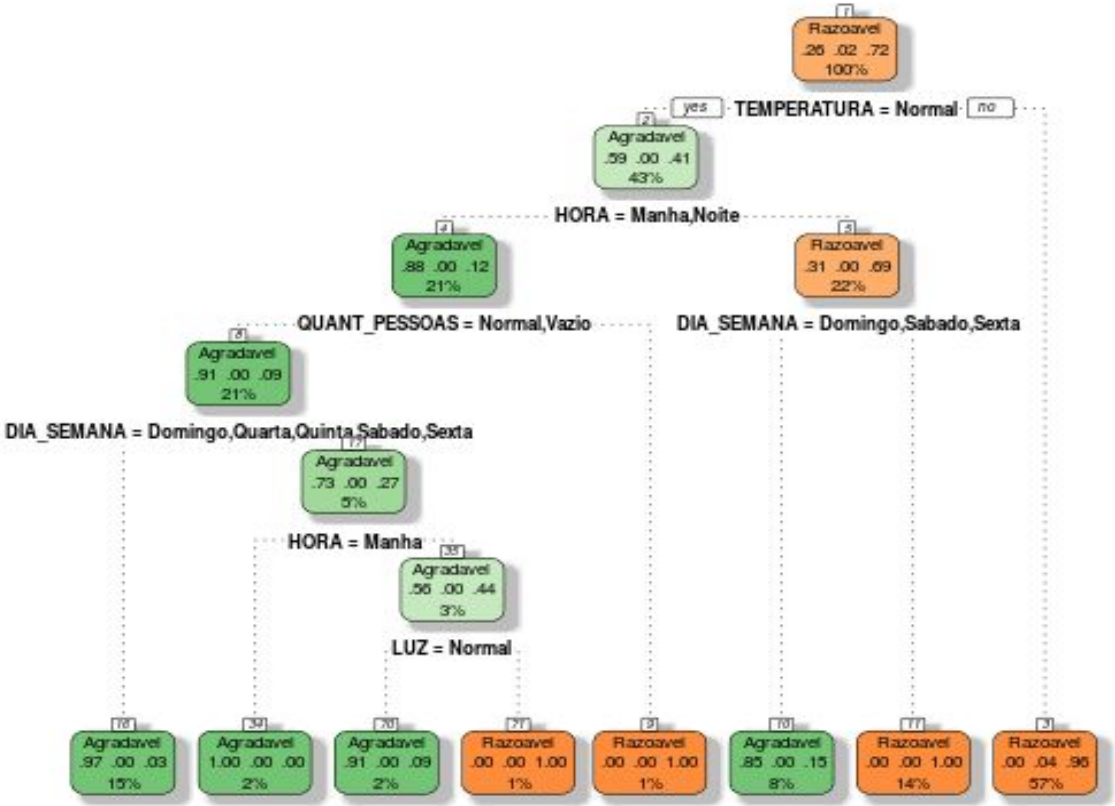
```
library(scatterplot3d)

d = dist(data)
mds = cmdscale(d, eig = TRUE, k = 3)
x = mds$points[,1]
y = mds$points[,2]
z = mds$points[,3]

scatterplot3d(x, y, z, box = FALSE)
```





Extração de padrões dos dados
Classificação

- Árvores de decisão:
  - Pacote *rpart*;



Extração de padrões dos dados
Regras de Associação

- Pacote *arules*:
  - Algoritmos Apriori e Eclat:

```
library(arules)

dados = read.csv("Base-15min-Rosana-Categorizada.csv",sep=",", header=T)

transactions_objects = as(dados[,-ncol(dados)], "transactions")

rules <- apriori(transactions_objects, parameter = list(supp = 0.2, conf = 0.5, minlen = 2, target = "rules"))

summary(rules)

inspect(rules)

write.table(inspect(rules), file = "~\\Rules-rosana.csv", row.names = FALSE, col.names = TRUE, sep = ",");
```

Extração de padrões dos dados
Regras de Associação

Regra	Supp	Conf	Lift
{TEMPERATURA=Quente} ⟶ {QUANT_PESSOAS=Vazio}	**0,47**	**0,82**	1,24

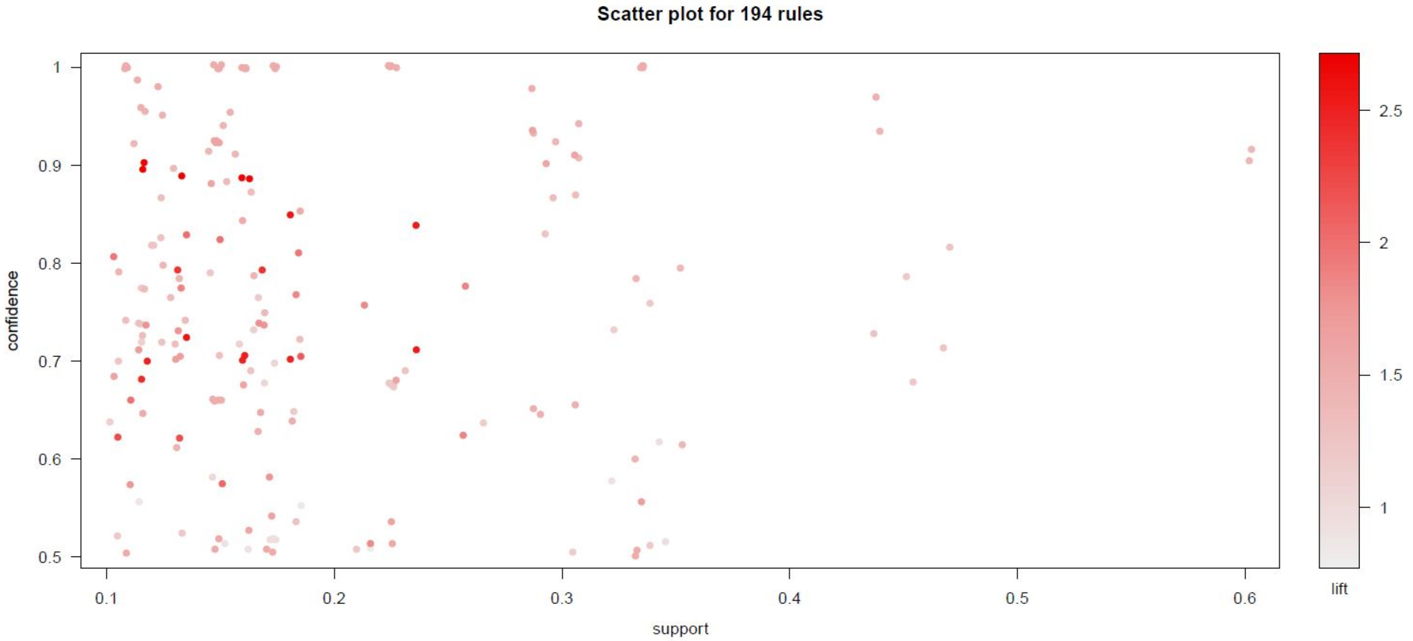
Em 47% das observações, o laboratório estava vazio e quente.
A probabilidade do laboratório estar vazio dado que está quente é de 82%.

Regra	Supp	Conf	Lift
{HORA=Comercial} ⟶ {QUANT_PESSOAS=Normal}	0,25	**0,61**	1,81

A probabilidade do laboratório estar com a quantidade normal de pessoas dados que o horário é comercial é de 61%.

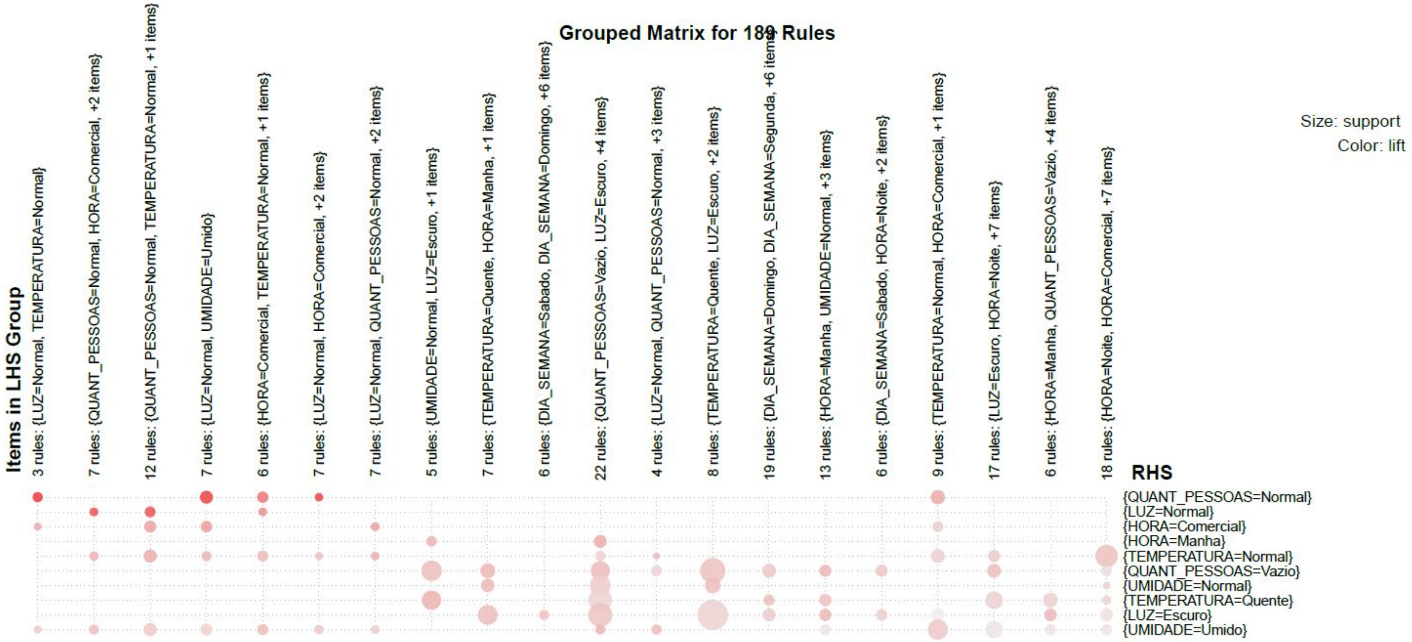
Extração de padrões dos dados
Regras de Associação - Pacote *arulesViz*

```
plot(rules)
```



Extração de padrões dos dados
Regras de Associação - Pacote *arulesViz*

```r
plot(rules, method="grouped matrix")
```



Extração de padrões dos dados

Regras de Associação - Pacote *arulesViz*

```
ruleExplorer(rules)
```

Cálculo de Distâncias

```
distEuclidiana = function (v1, v2){
    sqrt(sum((v1-v2)^2));
}
```

```
cor(v1, v2, method = c("pearson", "kendall", "spearman"))

dist(data, method = "canberra", diag = FALSE, upper = FALSE)

"euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski"
```

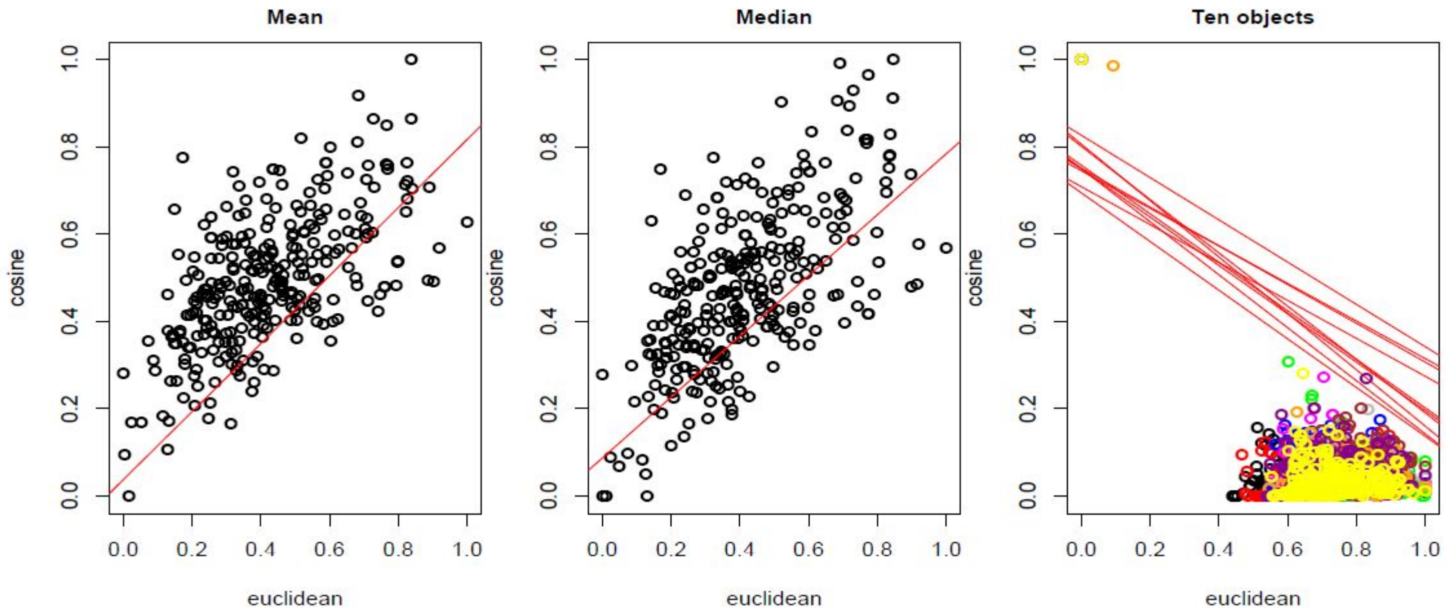
```
library(lsa)

cosine(v1, v2)
```

Cálculo de Distâncias

```
library(philentropy)

distance(data, method = "cosine")

[1] "euclidean"          "manhattan"          "minkowski"          "chebyshev"
[5] "sorensen"           "gower"              "soergel"            "kulczynski_d"
[9] "canberra"           "lorentzian"         "intersection"       "non-intersection"
[13] "wavehedges"         "czekanowski"        "motyka"             "kulczynski_s"
[17] "tanimoto"           "ruzicka"            "inner_product"      "harmonic_mean"
[21] "cosine"             "hassebrook"         "jaccard"            "dice"
[25] "fidelity"           "bhattacharyya"      "hellinger"          "matusita"
[29] "squared_chord"      "squared_euclidean" "pearson"            "neyman"
[33] "squared_chi"        "prob_symm"          "divergence"         "clark"
[37] "additive_symm"      "kullback-leibler"   "jeffreys"           "k_divergence"
[41] "topsoe"             "jensen-shannon"     "jensen_difference" "taneja"
[45] "kumar-johnson"      "avg"
```

Cálculo de Distâncias

```r
plot(matriz_dist_media, xlab = "euclidean", ylab = "cosine", col = "black", lwd = 3, main = "Mean")
abline(lm(matriz_dist_media[,1]~matriz_dist_media[,2], data.frame(matriz_dist_media)), col="red")
```



Extração de padrões dos dados
Agrupamento

- Pacote *ppclust*;
- Pacote *fclust*;
- Pacote *factoextra* for visualizing clusters.

Extração de padrões dos dados
Agrupamento - Pacote *ppclust*

```r
library(ppclust)
```

- Fuzzy c-Means

```r
fcm(x, centers, memberships, m=2, dmetric="sqeuclidean", pw = 2,
alginitv="kmpp", alginitu="imembrand",
nstart=1, iter.max=1000, con.val=1e-09,
fixcent=FALSE, fixmemb=FALSE, stand=FALSE, numseed)
```

Extração de padrões dos dados
Agrupamento - Pacote *ppclust*

- Fuzzy c-Means

```r
clust = fcm(wbcd, 3)

clust$x
clust$u
clust$cluster
```

Value

an object of class ‘ppclust’, which is a list consists of the following items:

`x`	a numeric matrix containing the processed data set.
`v`	a numeric matrix containing the final cluster prototypes (centers of clusters).
`u`	a numeric matrix containing the fuzzy memberships degrees of the data objects.
`d`	a numeric matrix containing the distances of objects to the final cluster prototypes.
`k`	an integer for the number of clusters.
`m`	a number for the fuzzifier.
`cluster`	a numeric vector containing the cluster labels found by defuzzying the fuzzy membership degrees of the objects.
`csize`	a numeric vector containing the number of objects in the clusters.
`iter`	an integer vector for the number of iterations in each start of the algorithm.
`best.start`	an integer for the index of start that produced the minimum objective functional.
`func.val`	a numeric vector for the objective function values in each start of the algorithm.
`comp.time`	a numeric vector for the execution time in each start of the algorithm.
`stand`	a logical value, `TRUE` shows that data set `x` contains the standardized values of raw data.

Extração de padrões dos dados
Agrupamento - Pacote *ppclust*

```
plotcluster(clust)
```

Extração de padrões dos dados
Agrupamento - Pacote *fclust*

```r
library(fclust)
```

- Fuzzy c-Means

```r
FKM (X, k, m, RS, stand, startU, conv, maxit)
```

Extração de padrões dos dados
Agrupamento - Pacote *fclust*

- Fuzzy c-Means

```
clust = FKM(wbcd, 3)

clust$U
clust$H
clust$Xca
```

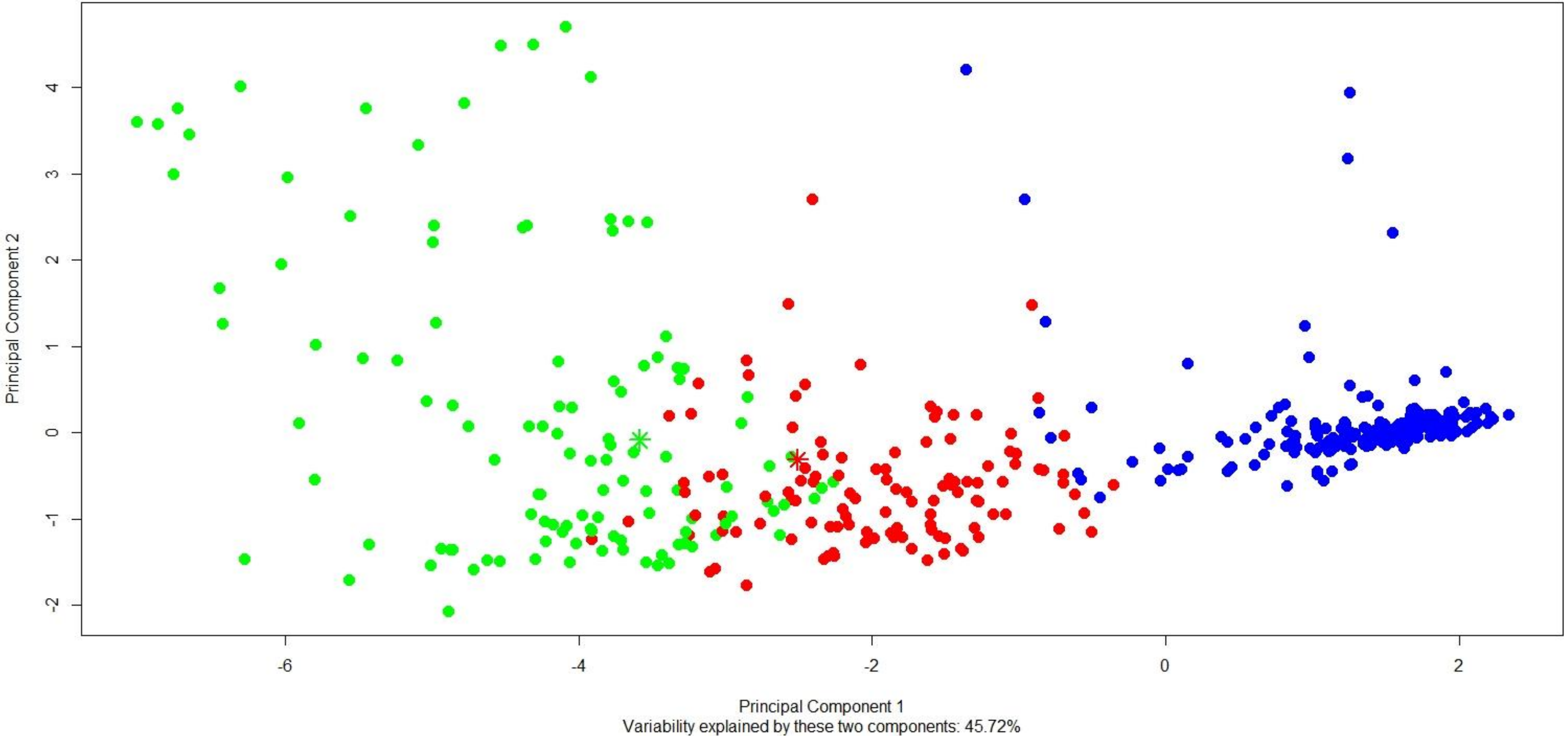
Value

Object of class `fclust`, which is a list with the following components:

`U`	Membership degree matrix
`H`	Prototype matrix
`F`	Array containing the covariance matrices of all the clusters (`NULL` for `FKM`)
`clus`	Matrix containing the indices of the clusters where the objects are assigned (column 1) and the associated membership degrees (column 2)
`medoid`	Vector containing the indices of the medoid objects (`NULL` for `FKM`)
`value`	Vector containing the loss function values for the `RS` starts
`cput`	Vector containing the computational times (user times) for the `RS` starts
`iter`	Vector containing the numbers of iterations for the `RS` starts
`k`	Number of clusters
`m`	Parameter of fuzziness
`ent`	Degree of fuzzy entropy (`NULL` for `FKM`)
`b`	Parameter of the polynomial fuzzifier (`NULL` for `FKM`)
`vp`	Volume parameter (`NULL` for `FKM`)
`delta`	Noise distance (`NULL` for `FKM`)
`stand`	Standardization (Yes if `stand=1`, No if `stand=0`)
`Xca`	Data used in the clustering algorithm (standardized data if `stand=1`)
`X`	Raw data
`call`	Matched call

Extração de padrões dos dados
Agrupamento - Pacote *fclust*

```
plot(clust, pca = TRUE)
```



Extração de padrões dos dados
Agrupamento - Pacote *fclust*

- A função FKM está implementada no pacote fclust com a função Euclidiana;
- Quer usar outra medida de similaridade ou dissimilaridade?
- Nome da função -> código

```
> FKM
function (X, k, m, RS, stand, startU, conv, maxit, seed)
{
    if (missing(X))
        stop("The data set must be given")
    if (is.null(X))
        stop("The data set X is empty")
    n = nrow(X)
    p = ncol(X)
    if (is.null(rownames(X)))
        rn = paste("Obj", 1:n, sep = " ")
    else rn = rownames(X)
```

Extração de padrões dos dados
Agrupamento - Pacote *fclust*

```r
while ((sum(abs(U.old - U)) > conv) && (iter < maxit)) {
    iter = iter + 1
    U.old = U
    for (c in 1:k) H[c, ] = (t(U[, c]^m) %*% X)/sum(U[,c]^m)
    for (i in 1:n) {
      for (c in 1:k) {
        D[i, c] = sum((X[i, ] - H[c, ])^2)
      }
    }
    for (i in 1:n) {
      if (min(D[i, ]) == 0) {
        U[i, ] = rep(0, k)
        U[i, which.min(D[i, ])] = 1
      }
      else {
        for (c in 1:k) {
          U[i, c] = ((1/D[i, c])^(1/(m - 1)))/sum(((1/D[i,])^(1/(m - 1))))
        }
      }
```

Extração de padrões dos dados
Agrupamento - Pacote *fclust*

```
while ((sum(abs(U.old - U)) > conv)){ #&& (iter < maxit)) {
    iter = iter + 1
    U.old = U
    for (c in 1:k) H[c, ] = (t(U[, c]^m) %*% X)/sum(U[, c]^m) #calculo do prototipo
    for (i in 1:n) {
      for (c in 1:k) {
        D[i, c] = 1 - cosine(X[i,], H[c,])^2
      }
    }
    for (i in 1:n) {
      if (min(D[i, ]) == 0) {
        U[i, ] = rep(0, k)
        U[i, which.min(D[i, ])] = 1
      }
      else {
        for (c in 1:k) {
          U[i, c] = ((1/D[i, c])^(1/(m - 1)))/sum(((1/D[i, ])^(1/(m - 1))))
        }
      }
```

Extração de padrões dos dados Agrupamento - Validação

- Após o agrupamento, a partição obtida pelo algoritmo deve ser validada por um índice de validação;
- Pacote *fclust*:
  - Coeficiente de Partição (PC)
  - Entropia da Partição (PE)
  - Coeficiente da Partição Modificada (MPC)
  - Índice de Xie -Beni (XB)
  - Silhueta (SIL)
  - Silhueta Fuzzy (SIL.F)

Extração de padrões dos dados
Agrupamento - Validação

```r
for (c in 2:cmax) {
    cat("cluster", c)
    cat("\n")
    clust <- FKM(data, k = c, m = mfuzz, RS = rs, conv = error);
    indicesresults[c-1, ] = c(KYIfunc(clust$Xca, clust$U), SCfunc(clust$Xca, clust$U, clust$H, mfuzz),
                            Pfunc(clust$Xca, clust$U), Kfunc(clust$Xca, clust$U, clust$H),
                            Tfunc(clust$Xca, clust$U, clust$H), PCAESfunc(clust$Xca, clust$U, clust$H),
                            PBMFfunc(clust$Xca, clust$U, clust$H, mfuzz), MPOfunc(clust$U),
                            GDfunc(clust$Xca, clust$U), XBfunc(clust$Xca, clust$U, clust$H, mfuzz),
                            FSfunc(clust$Xca, clust$U, clust$H, mfuzz), SILFfunc(clust$Xca, clust$U, alpha = 1),
                            PC(clust$U), PE(clust$U, 10), MPC(clust$U), WLIfunc(clust$Xca, clust$U, clust$H),
                            NPEfunc(clust$Xca, clust$U), PEBfunc(clust$U));
}
```

Extração de padrões dos dados
Agrupamento - Validação

- Coeficiente de Partição (PC)

$$PC(U) = \frac{1}{n} \sum_{i=1}^{c} \sum_{k=1}^{n} A_i^2(x_k)$$

Extração de padrões dos dados
Agrupamento - Validação

- Java

```java
public static double coeficienteParticao(List<Membership> membership){

        double soma=0D;

        for (Membership members : membership) {
            for (int j=0; j < members.getDegrees().size(); j++)
                soma += Math.pow(members.getDegrees().get(j), 2);
        }

        return (soma/membership.size());
}
```

- R

```r
> PC
function (U)
{
    part.coeff = sum(U^2)/nrow(U)
    return(part.coeff)
}
```

Extração de padrões dos dados
Agrupamento - Validação

- Índice de Xie -Beni (XB)

$$XB(U,V;X) = \frac{\sum_{i=1}^{c}\sum_{k=1}^{n} A_i^m(x_k)\|x_k - v_i\|^2}{n \times min_{k \neq i}\|v_i - v_j\|^2}.$$

- Índice de Zahid et al. (SC)

$$SC = SC_1(U,V;X) - SC_2(U),$$

$$SC_1(U,V;X) = \frac{\sum_{i=1}^{c}\|v_i - \bar{v}\|^2/c}{\sum_{i=1}^{c}\left(\sum_{k=1}^{n} A_i^m(x_k)\|x_k - v_i\|^2 / \sum_{k=1}^{n} A_i(x_k)\right)},$$

$$SC_2(U) = \frac{\sum_{i=1}^{c-1}\sum_{l=i+1}^{c}\left(\frac{\sum_{k=1}^{n}(min(A_i(x_k),A_l(x_k)))^2}{\sum_{k=1}^{n} min(A_i(x_k),A_l(x_k))}\right)}{\sum_{k=1}^{n}(max_{1 \leq i \leq c}A_i(x_k))^2 / \sum_{k=1}^{n} max_{1 \leq i \leq c}A_i(x_k)}.$$

Extração de padrões dos dados
Agrupamento - Validação *fclust*

ELSEVIER

CrossMark

Available online at www.sciencedirect.com

ScienceDirect

Fuzzy Sets and Systems 279 (2015) 1–16

FUZZY sets and systems

www.elsevier.com/locate/fss

A toolbox for fuzzy clustering using the R programming language

Maria Brigida Ferraro *, Paolo Giordani

Dipartimento di Scienze Statistiche, Sapienza Università di Roma, Italy

Received 20 May 2014; received in revised form 4 February 2015; accepted 4 May 2015

Available online 8 May 2015

Abstract

Fuzzy clustering is used extensively in several domains of research. In the literature, starting from the well-known fuzzy k-means (fkm) clustering algorithm, an increasing number of papers devoted to fkm and its extensions can be found. Nevertheless, a lack of the related software for implementing these algorithms can be observed preventing their use in practice. Even the standard fkm is not necessarily available in the most common software. For this purpose, a new toolbox for fuzzy clustering using the R programming language is presented by examples. The toolbox, called `fclust`, contains a suit of fuzzy clustering algorithms, fuzzy cluster validity indices and visualization tools for fuzzy clustering results.

Extração de padrões dos dados
Agrupamento - Validação *fclust*

- Funções para visualização dos resultados do agrupamento fuzzy

VIFCR:	three plots proposed in [22]. The input arguments are `fclust.obj` and `which` (1 for the chart diagram, 2 for the scatter plot of the highest membership degrees and 3 for the scatter plot of the membership degrees over distances).
VAT:	visual assessing of (cluster) tendency. The input argument is the data matrix used in the analysis, `Xca`.
VCV:	visual cluster validity. The input arguments are `Xca`, `U`, `H` and `which` (1 for VAT and 2 for VCV).
VCV2:	new type of visual cluster validity. The input arguments are `Xca`, `U` and `which` (1 for VAT and 2 for VCV2).

Extração de padrões dos dados
Agrupamento - Validação *fclust*

```
VAT(clust$Xca)

VCV2(clust$Xca, clust$U, 2)
```

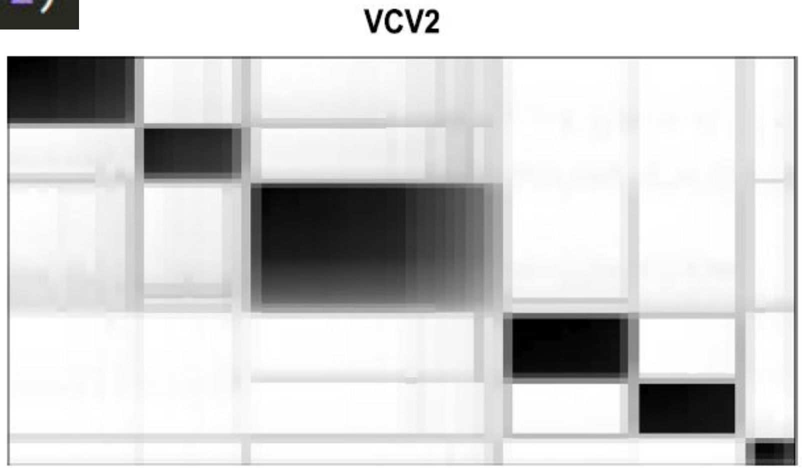
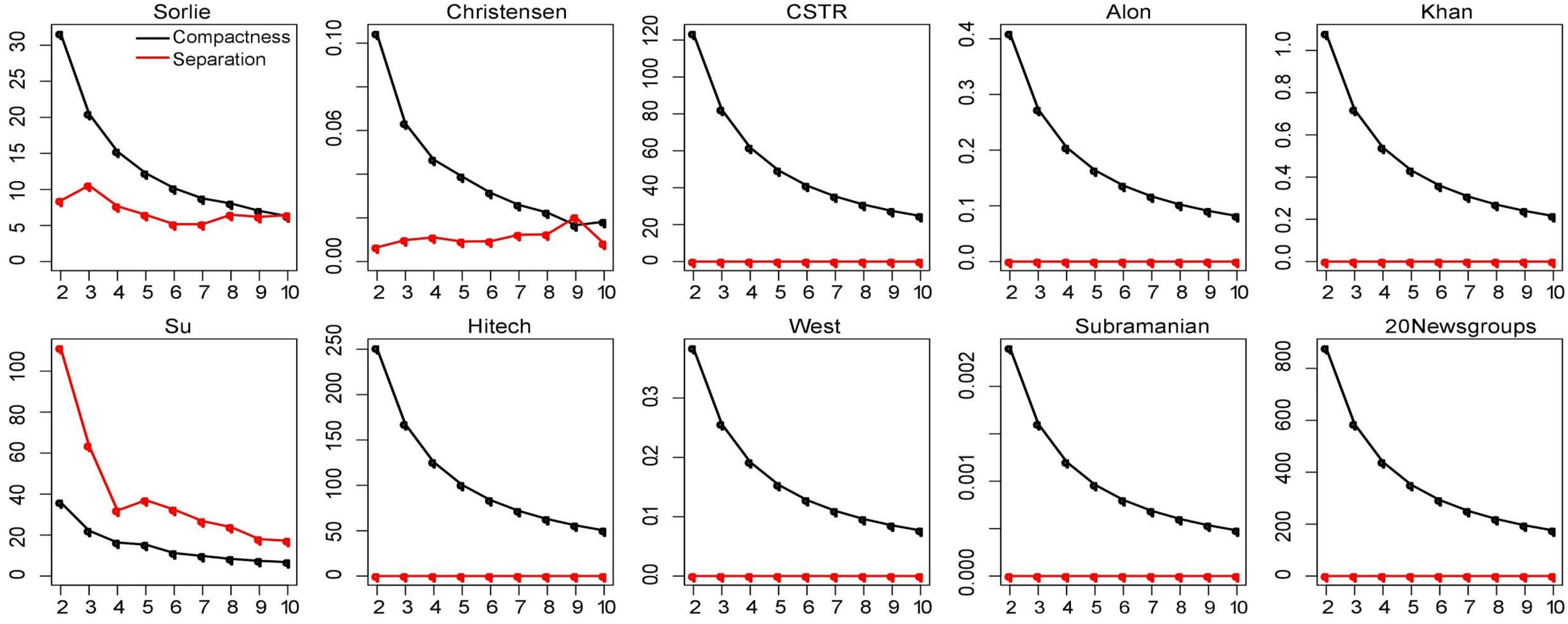


Fig. 2. Visual cluster validity for the 6 clusters obtained by means of fkm on McDonald's data.

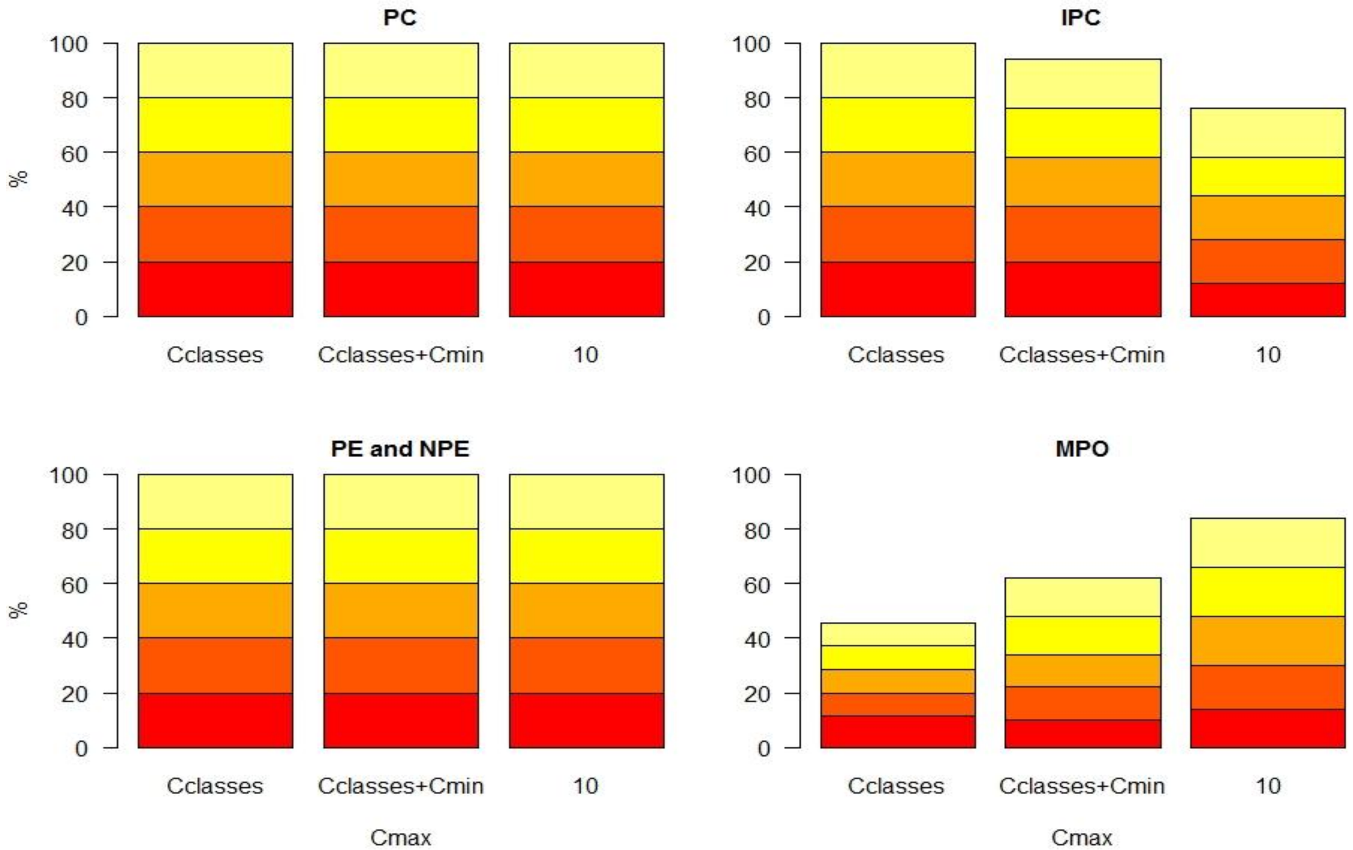
Gráficos

- Plot resultado de um índice de validação
- Plot t=*o*, Points



Gráficos

- Barplot



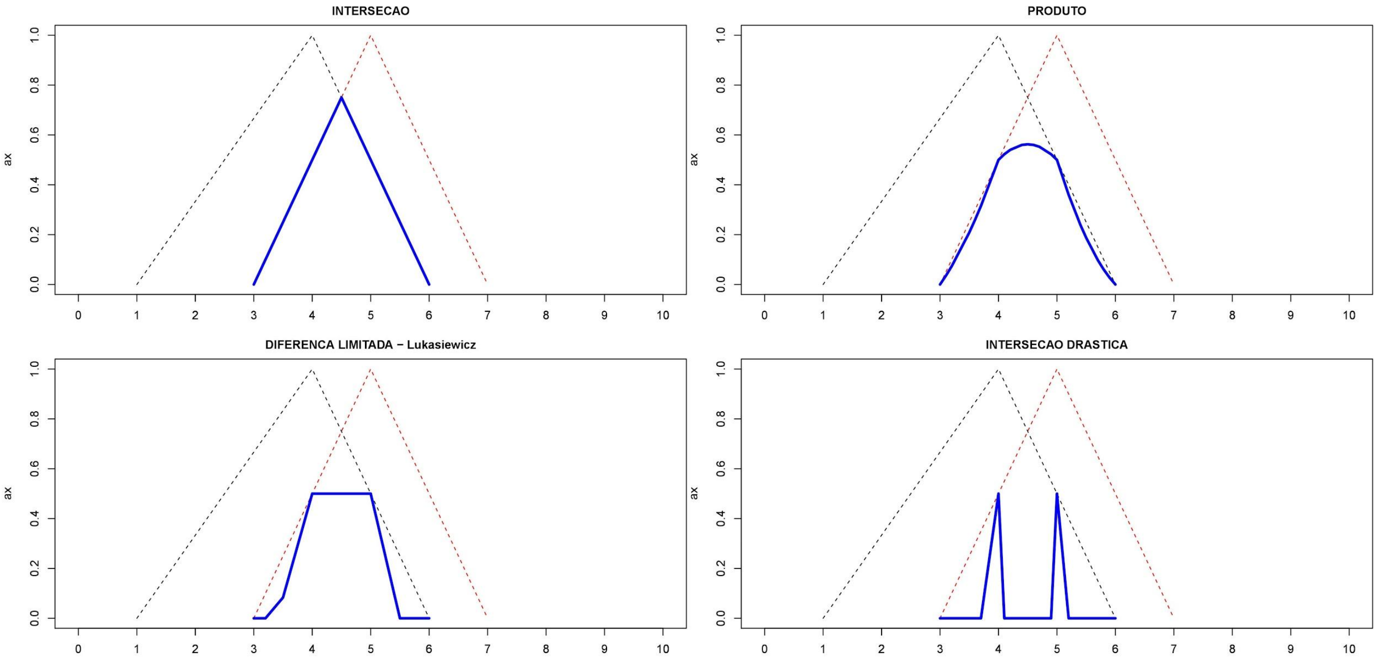
Gráficos

- Barplot

```r
par(mfrow=c(2,2), mar=c(4,3.8,2,0) + 0.1)
par(ps = 12, cex = 1, cex.main = 1, cex.sub  = 1.5)

barplot(pc, ylab = "%", yaxt = "n", col = heat.colors(5), ylim = c(0, 100));
axis(2, las = 2)
title("PC")
barplot(ipc, yaxt = "n", col = heat.colors(5), ylim = c(0, 100));
axis(2, las = 2)
title("IPC")
barplot(npe, ylab = "%", yaxt = "n", xlab = "Cmax", col = heat.colors(5), ylim = c(0, 100));
axis(2, las = 2)
title("PE and NPE")
barplot(mpo, yaxt = "n", xlab = "Cmax", col = heat.colors(5), ylim = c(0, 100));
axis(2, las = 2)
title("MPO")
```

Gráficos

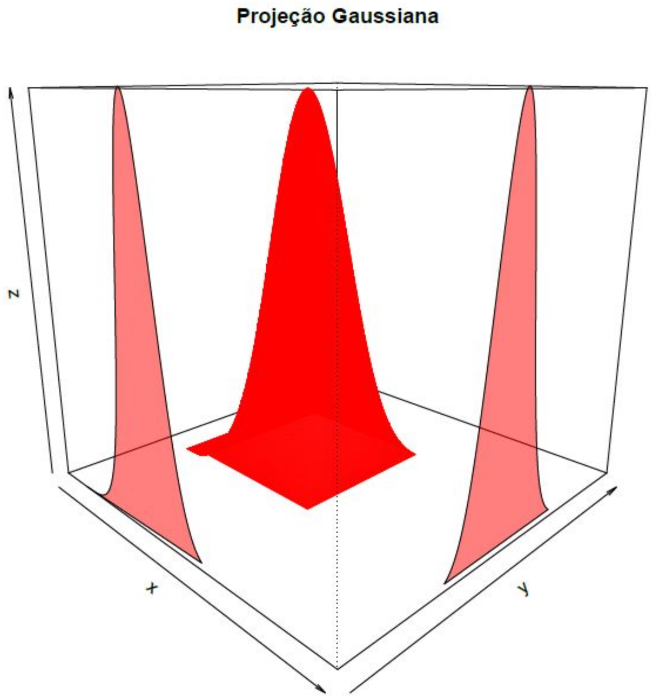
- ❑ Plot type *l*
- ❑ Points

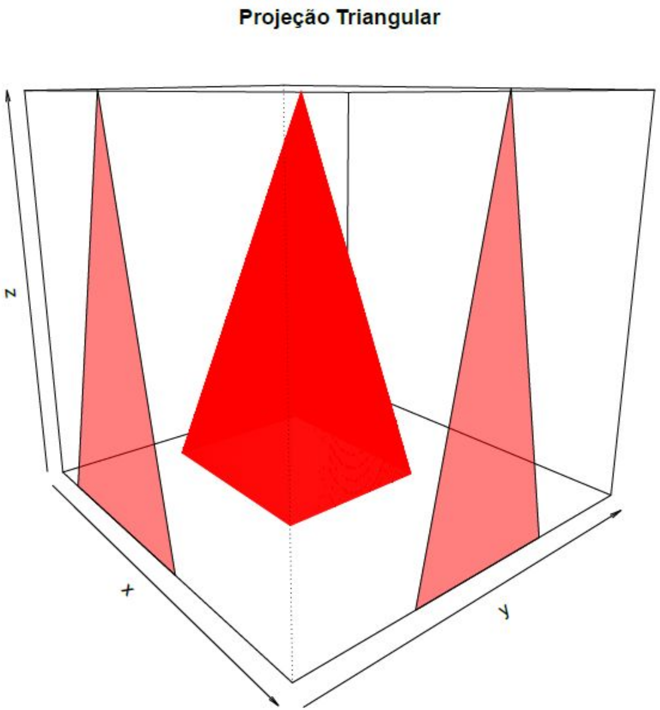



```r
par(mfrow=c(2,2), mar=c(2.5,3.8,2,0) + 0.1) #c(bottom, left, top, right)
par(ps = 12, cex = 0.8, cex.main = 1)

plot(a, ax, type = "l", xlim = c(0, 10), ylim = c(0,1), lty = 2, main = "INTERSECAO")
axis(1, at=1:10, labels=seq(1:10), las = 1)
points(b, bx, type = "l", xlim = c(0, 10), ylim = c(0,1), col = "red", xaxt = "n", lty = 2)
points(intersect(a, b), inter1(axinter, bxinter), type = "l", col="blue", lwd = 3)
```

Gráficos

- persp

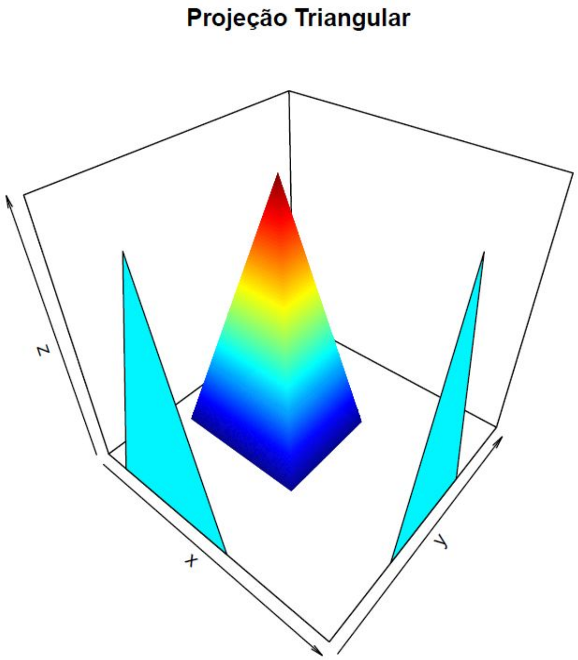


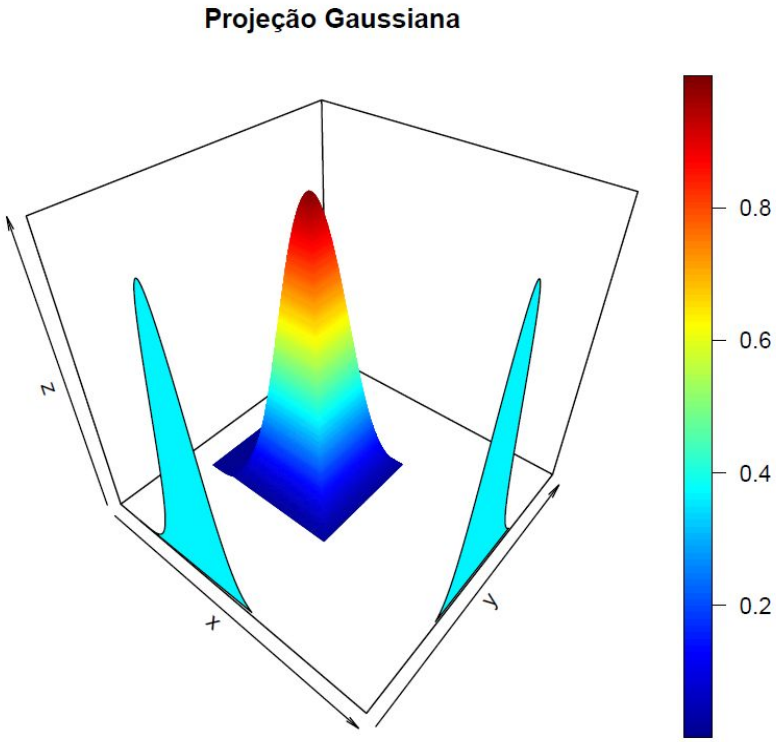



```r
pmat = persp(x, y, z, xlim = c(0, 10), ylim = c(0, 10), zlim = range(z), theta=50,  col = "red", border = NA,
    main = "Projeção Triangular")
mypoints = trans3D(x, rep(0, length(x)), rx(z), pmat=pmat)
polygon(mypoints, col=rgb(1,0,0, alpha = 0.5))
mypoints = trans3D(rep(10, length(y)), y, ry(z), pmat=pmat)
polygon(mypoints, col=rgb(1,0,0, alpha = 0.5))
```

Gráficos

- persp3D






```
pmat = persp3D(x, y, z, xlim = c(0, 10), ylim = c(0, 10), zlim = range(z), theta=40,  main = "Projeção - Triangular")
mypoints = trans3D(x, rep(0, length(x)), rx(z), pmat=pmat)
polygon(mypoints, col = "turquoise1")
mypoints = trans3D(rep(10, length(y)), y, ry(z), pmat=pmat)
polygon(mypoints, col = "turquoise1")
```

Obrigada!

Dúvidas?

E-mail: nandaeustaquio7@gmail.com