Introduzione alla chemiometria e disegno sperimentale

Modulo 4: PCA e clustering in R software

Docente: Dr. Sabina Licen (slicen@units.it)

Preparare le specifiche dell'area del grafico

la funzione par()

Questa funzione serve per indicare una serie di specifiche desiderate per la finestra in cui poi verranno disegnati i grafici.

par(argomento1=, argomento2=,...)

Argomenti possibili:

```
 [1] "xlog"     "ylog"     "adj"      "ann"      "ask"      "bg"
 [7] "bty"      "cex"      "cex.axis" "cex.lab"  "cex.main" "cex.sub"
[13] "cin"      "col"      "col.axis" "col.lab"  "col.main" "col.sub"
[19] "cra"      "crt"      "csi"      "cxy"      "din"      "err"
[25] "family"   "fg"       "fig"      "fin"      "font"     "font.axis"
[31] "font.lab" "font.main" "font.sub" "lab"     "las"      "lend"
[37] "lheight"  "ljoin"    "lmitre"   "lty"      "lwd"      "mai"
[43] "mar"      "mex"      "mfcol"    "mfg"      "mfrow"    "mgp"
[49] "mkh"      "new"      "oma"      "omd"      "omi"      "page"
[55] "pch"      "pin"      "plt"      "ps"       "pty"      "smo"
[61] "srt"      "tck"      "tcl"      "usr"      "xaxp"     "xaxs"
[67] "xaxt"     "xpd"      "yaxp"     "yaxs"     "yaxt"     "ylbias"
```

la funzione par()

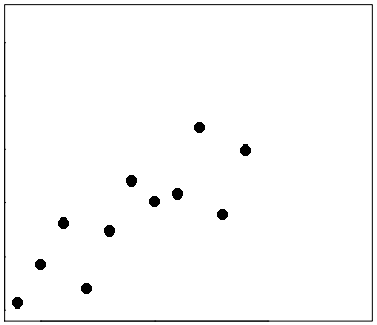
Per visualizzare i parametri di default basta eseguire:

```
> ParDef<-par()
> ParDef
$xlog
[1] FALSE

$ylog
[1] FALSE

$adj
[1] 0.5

$ann
[1] TRUE

$ask
[1] FALSE

$bg
[1] "transparent"

$bty
[1] "o"

$cex
[1] 1
```

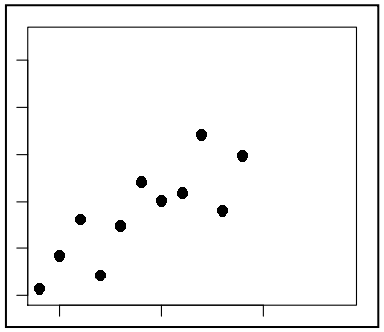
etc...

la funzione par()

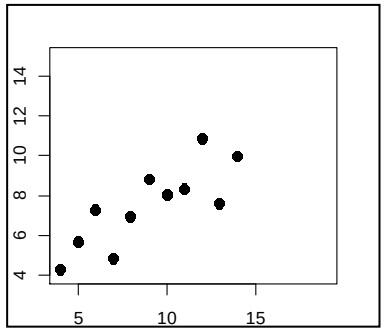
par(oma=c(0,0,0,0),mar=c(0,0,0,0))



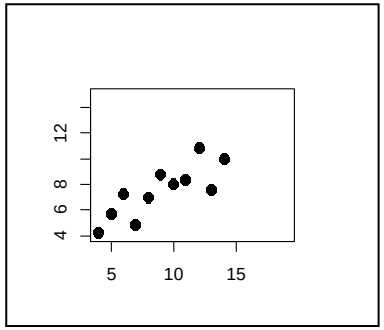
par(oma=c(0,0,0,0),mar=c(1,1,1,1))



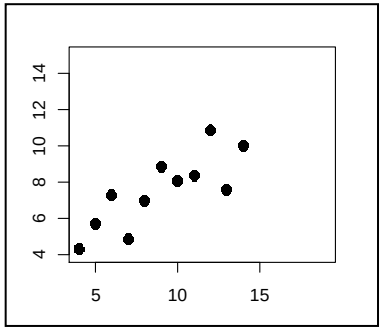
par(oma=c(0,0,0,0),mar=c(2,2,2,2))



par(oma=c(2,2,2,2),mar=c(2,2,2,2))

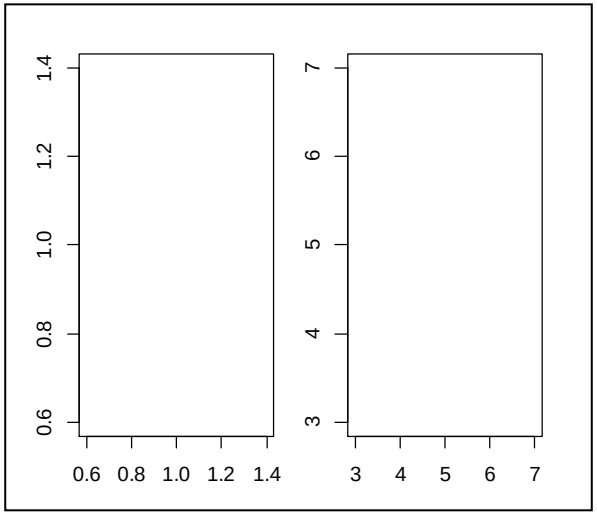


par(oma=c(1,1,1,1),mar=c(2,2,1,1))

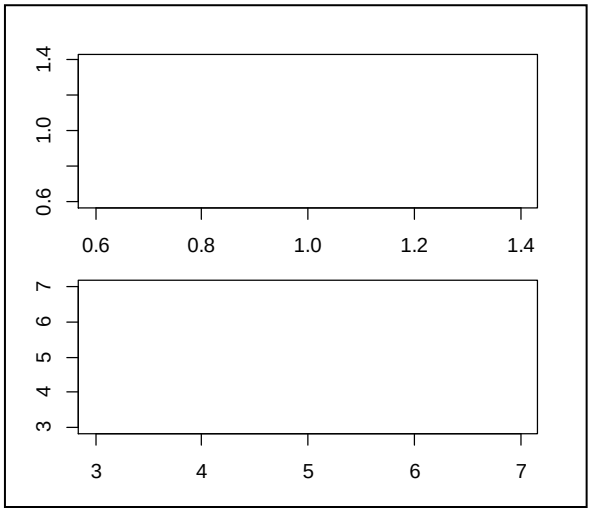


la funzione par()

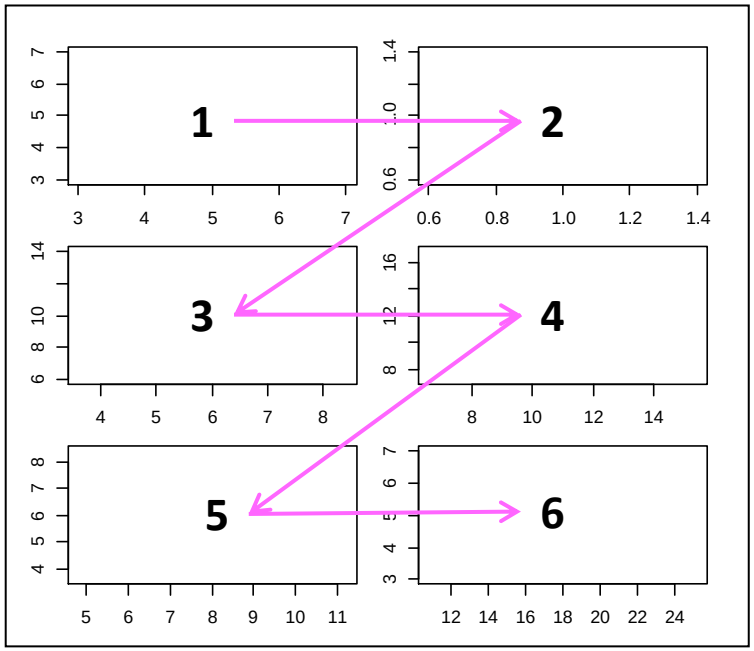
par(mfrow=c(1,2))



par(mfrow=c(2,1))



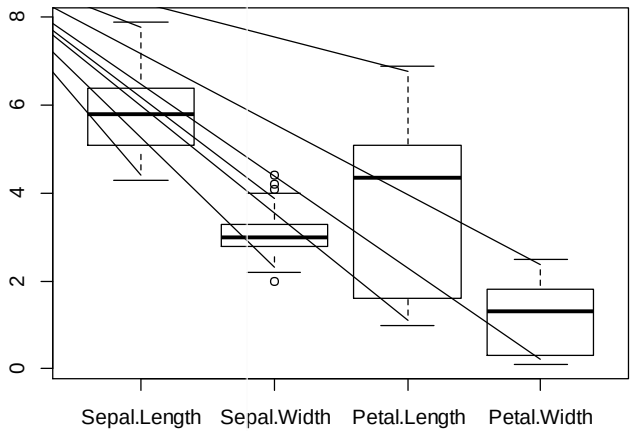
par(mfrow=c(3,2))



Valori restituiti da alcune funzioni grafiche

Alcune funzioni grafiche se assegnate ad un nome restituiscono anche dei valori, es.:

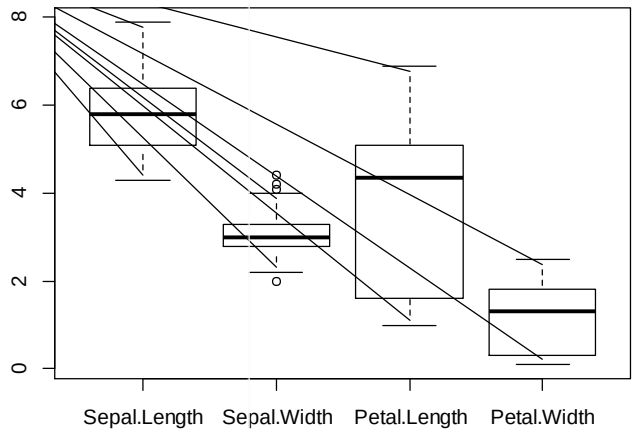
```
> Bx<-boxplot(iris[,1:4])
> Bx
$stats
     [,1] [,2] [,3] [,4]
[1,]  4.3  2.2 1.00  0.1
[2,]  5.1  2.8 1.60  0.3
[3,]  5.8  3.0 4.35  1.3
[4,]  6.4  3.3 5.10  1.8
[5,]  7.9  4.0 6.90  2.5

$n
[1] 150 150 150 150

$conf
         [,1]     [,2]     [,3]    [,4]
[1,] 5.632292 2.935497 3.898477 1.10649
[2,] 5.967708 3.064503 4.801523 1.49351

$out
[1] 4.4 4.1 4.2 2.0

$group
[1] 2 2 2 2

$names
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
```



Valori restituiti da alcune funzioni grafiche

Alcune funzioni grafiche se assegnate ad un nome restituiscono anche dei valori, es.:

```
> Bx<-boxplot(iris[,1:4])
> Bx
$stats
     [,1] [,2] [,3] [,4]
[1,]  4.3  2.2 1.00  0.1
[2,]  5.1  2.8 1.60  0.3
[3,]  5.8  3.0 4.35  1.3
[4,]  6.4  3.3 5.10  1.8
[5,]  7.9  4.0 6.90  2.5

$n
[1] 150 150 150 150

$conf
         [,1]     [,2]     [,3]    [,4]
[1,] 5.632292 2.935497 3.898477 1.10649
[2,] 5.967708 3.064503 4.801523 1.49351

$out
[1] 4.4 4.1 4.2 2.0

$group
[1] 2 2 2 2

$names
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
```




```
> Bx$stats
     [,1] [,2] [,3] [,4]
[1,]  4.3  2.2 1.00  0.1
[2,]  5.1  2.8 1.60  0.3
[3,]  5.8  3.0 4.35  1.3
[4,]  6.4  3.3 5.10  1.8
[5,]  7.9  4.0 6.90  2.5
> class(Bx$stats)
[1] "matrix"
> Bx$out
[1] 4.4 4.1 4.2 2.0
> class(Bx$out)
[1] "numeric"
```

attributes(nomegrafico)

Le diverse informazioni si richiamano così e l'elenco così

Grafici 3d

library(rgl) Bisogna installare e caricare il pacchetto rgl.

plot3d(vettoreX,vettoreY,vettoreZ, argomenti= ...)

points3d(vettoreX,vettoreY,vettoreZ, argomenti= ...)

text3d(vettoreX,vettoreY,vettoreZ, argomenti= ...)

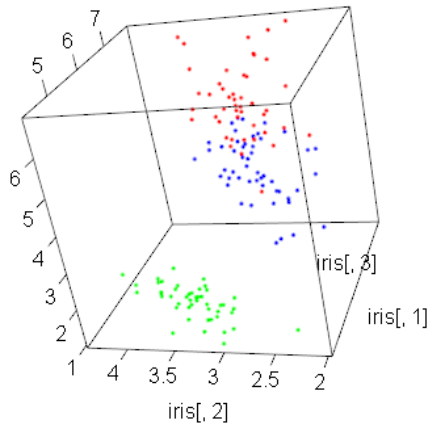
Funzionano come per i grafici bidimensionali

decorate3d(argomenti= ...) Per aggiungere assi, etichette, etc...

I grafici di questo tipo si aprono in una nuova finestra e possono essere fatti "girare" e ingrandire o rimpicciolire con la rotella del mouse.

Per salvare uno snapshot del grafico si usa:

snapshot3d("Nome grafico.emf")
[o anche .png]



Importare un set di dati in R

Premessa: si possono importare files con estensione .txt, .dat, .data (ovvero qualsiasi file in codice ASCII) oppure .csv ("comma delimited files")

Importare un set di dati in R

Premessa: si possono importare files con estensione .txt, .dat, .data (ovvero qualsiasi file in codice ASCII) oppure .csv ("comma delimited files")

(1) Importare un set di dati come dataframe

Nota: il set di dati contiene numeri ma può contenere anche più di una colonna di testo oltre alla prima

oppure **read.csv**

```
nomedataframe<-read.table("Nomefile.txt", header = TRUE, sep = "\t", dec=".")
nomedataframe <-data.frame(nomedataframe)
```

questa funzione serve a comunicare al software che l'oggetto importato è un dataframe

indicare il separatore di elenco presente nel file (es. ";" o "," o "\t" per tabulare)

indicare il separatore decimale presente nel file (es. "." o ",")

Importare un set di dati in R

Premessa: si possono importare files con estensione .txt, .dat, .data (ovvero qualsiasi file in codice ASCII) oppure .csv ("comma delimited files")

(1) Importare un set di dati come dataframe

Nota: il set di dati contiene numeri ma può contenere anche più di una colonna di testo oltre alla prima

oppure **read.csv**

```
nomedataframe<-read.table("Nomefile.txt", header = TRUE, sep = "\t", dec=".")
nomedataframe <-data.frame(nomedataframe)
```

questa funzione serve a comunicare al software che l'oggetto importato è un dataframe

indicare il separatore di elenco presente nel file (es. ";" o "," o "\t" per tabulare)

indicare il separatore decimale presente nel file (es. "." o ",")

E' utile anche porre l'argomento **stringsAsFactors** =**FALSE**, in modo che le colonne con caratteri non vengano automaticamente convertite in variabili *factor* (meglio farlo dopo, solo se serve).

SEGUE

(2) Importare un set di dati come matrice

Nota: il set di dati deve contenere solo numeri tranne che nella prima riga e nella prima colonna

```
nomematrice<-read.table("Nomefile.txt", header = TRUE, row.names=1, sep = "\t", dec=".")
nomematrice<-as.matrix(nomematrice)
```

questa funzione serve a comunicare al software che l'oggetto importato è una matrice

indicare il separatore di elenco presente nel file (es. ";" o "," o "\t" per tabulare)

indicare il separatore decimale presente nel file (es. "." o ",")

Notare la presenza dell'argomento row.names=1 !!!

(2) Importare un set di dati come matrice

Nota: il set di dati deve contenere solo numeri tranne che nella prima riga e nella prima colonna

```
nomematrice<-read.table("Nomefile.txt", header = TRUE, row.names=1, sep = "\t", dec=".")
nomematrice<-as.matrix(nomematrice)
```

questa funzione serve a comunicare al software che l'oggetto importato è una matrice

indicare il separatore di elenco presente nel file (es. ";" o "," o "\t" per tabulare)

indicare il separatore decimale presente nel file (es. "." o ",")

Notare la presenza dell'argomento row.names=1 !!!

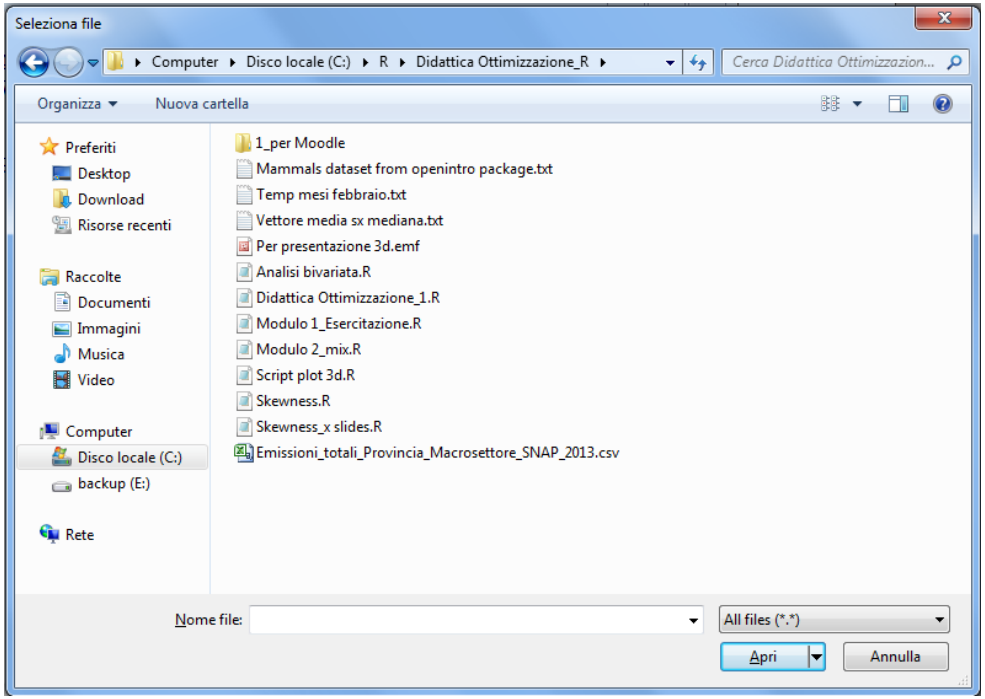
Altri argomenti che possono essere utili nella funzione read.table (o csv):

skip= ; na.strings =; nrow =

(3) La funzione file.choose()

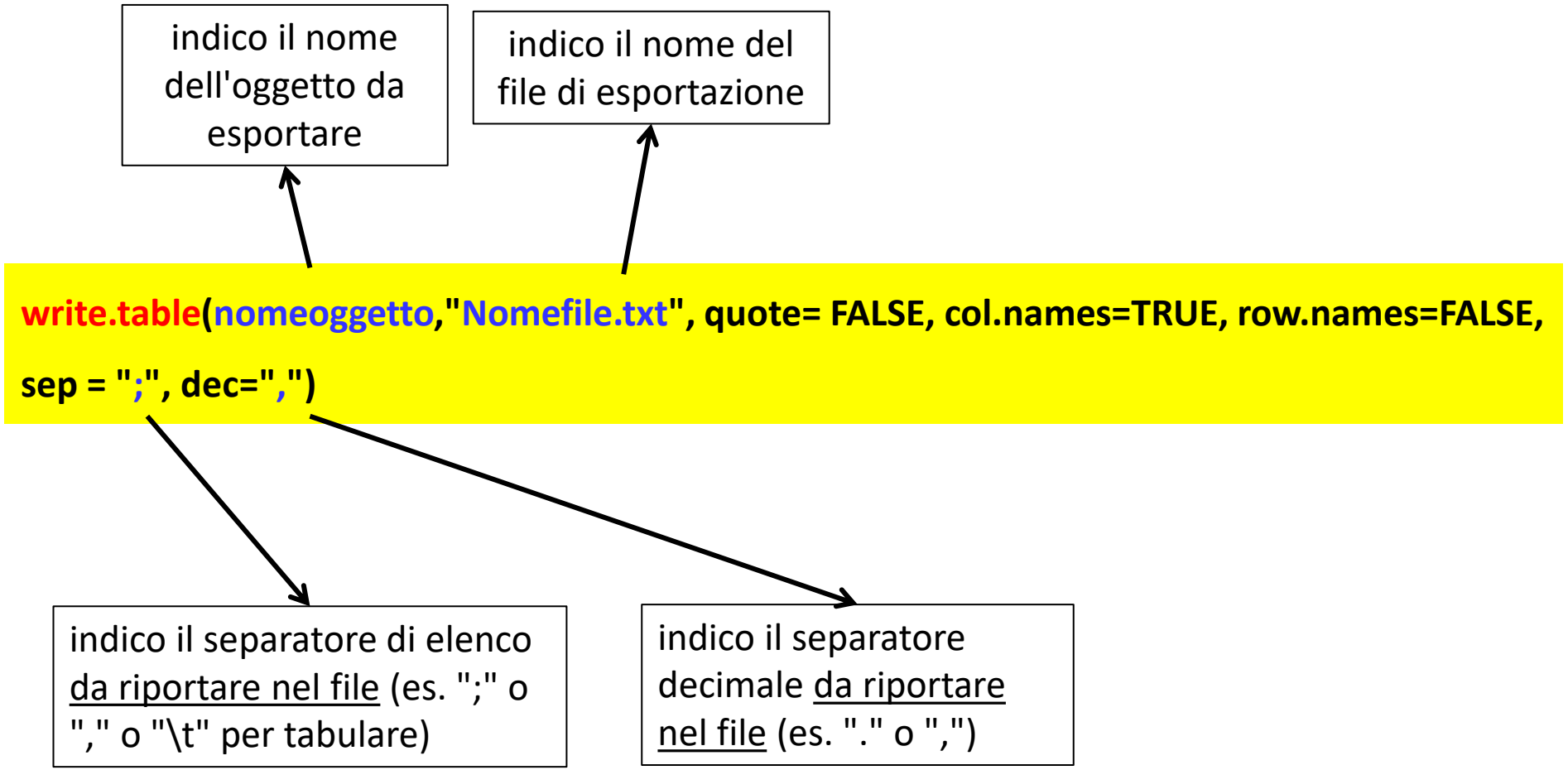
E' una funzione che consente di selezionare il file da una finestra di dialogo senza doverlo scrivere nella funzione read.table (o read.csv)

```
nomematrice<-read.table(file.choose(), header = TRUE, row.names=1, sep = "\t", dec=".")
```



Esportare un set di dati in R

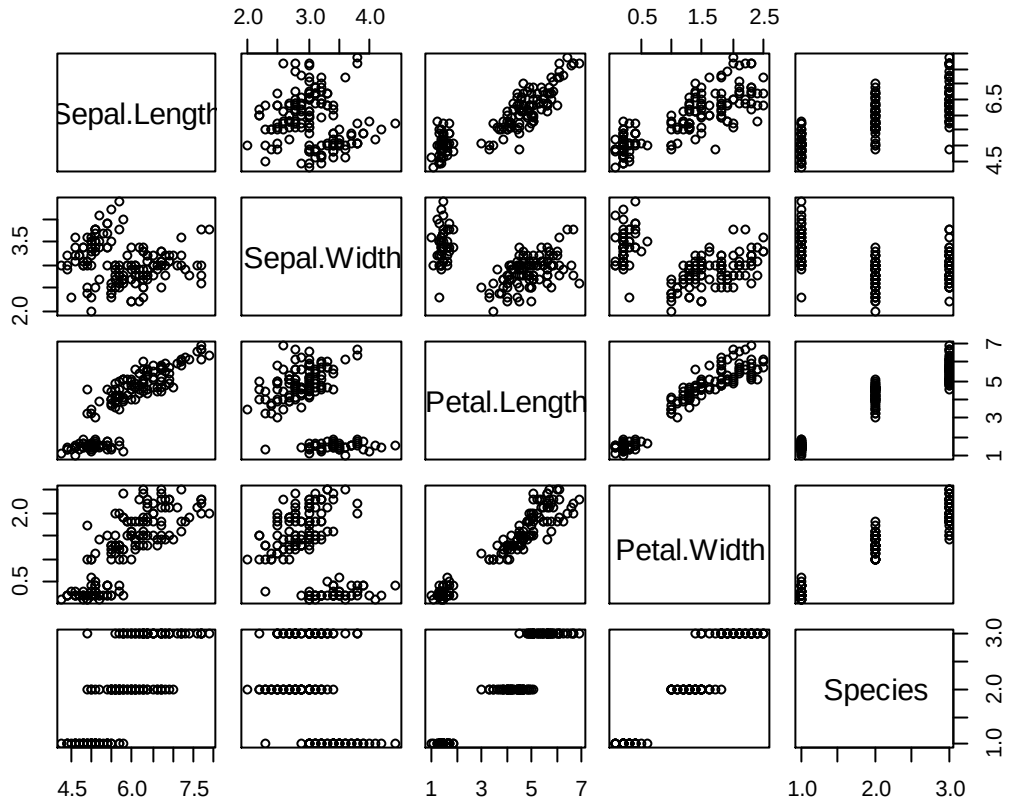
Qualsiasi oggetto creato in R può essere esportato come file di testo e poi essere importato ad es. in Excel



Una proprietà della funzione plot()

plot(nomeoggetto)

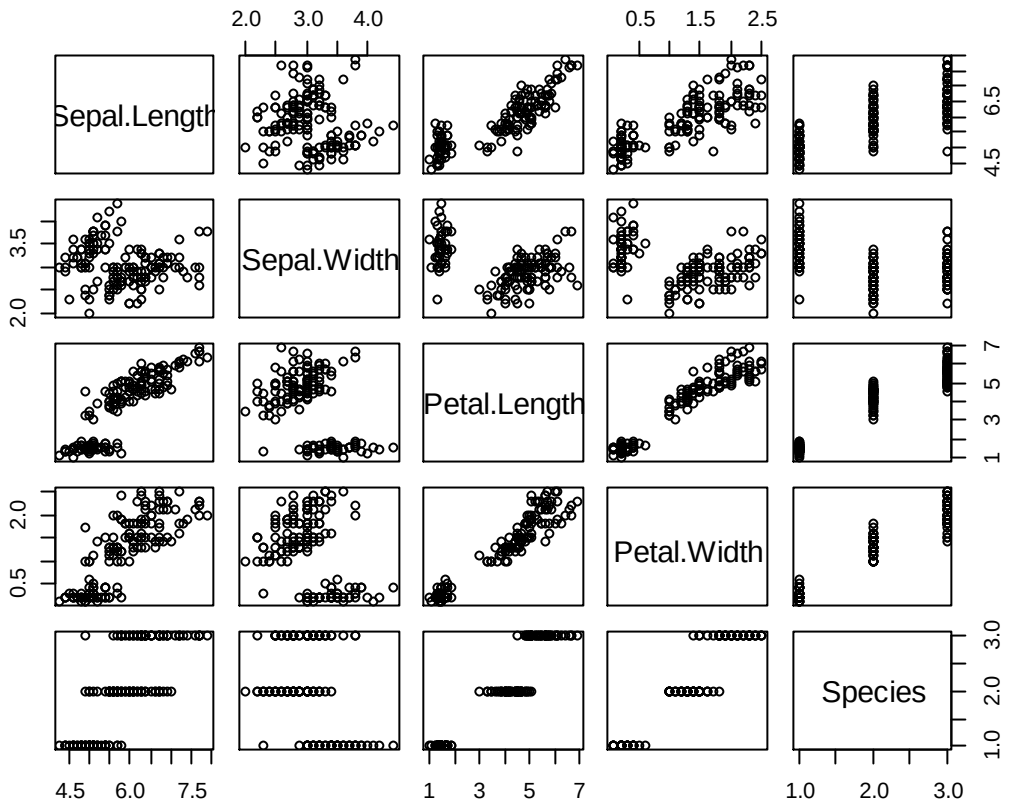
Utilizzando la funzione su una matrice o un dataframe si ottiene un grafico di questo tipo:



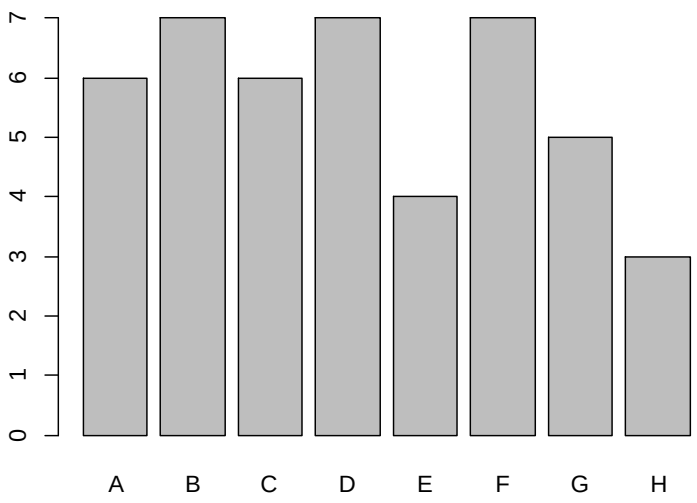
Una proprietà della funzione plot()

plot(nomeoggetto)

Utilizzando la funzione su una matrice o un dataframe si ottiene un grafico di questo tipo:



Utilizzando la funzione su un vettore di tipo factor, si ottiene un grafico a barre che "conta" quante volte compare nel vettore lo stesso livello:



La funzione cor()

cor(nomeoggetto)

Calcola il coefficiente di correlazione di Pearson per coppie di variabili e restituisce la "matrice di correlazione"

```
> cor(iris[,1:4])
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

La funzione cor()

cor(nomeoggetto)

Calcola il coefficiente di correlazione di Pearson per coppie di variabili e restituisce la "matrice di correlazione"

```
> cor(iris[,1:4])
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

---

round(nomeoggetto, digit=...)

Utile funzione che arrotonda al numero di cifre decimali desiderato

```
> round(cor(iris[,1:4]),digit=2)
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length         1.00       -0.12         0.87        0.82
Sepal.Width         -0.12        1.00        -0.43       -0.37
Petal.Length         0.87       -0.43         1.00        0.96
Petal.Width          0.82       -0.37         0.96        1.00
> round(cor(iris[,1:4]),digit=1)
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length          1.0        -0.1          0.9         0.8
Sepal.Width          -0.1         1.0         -0.4        -0.4
Petal.Length          0.9        -0.4          1.0         1.0
Petal.Width           0.8        -0.4          1.0         1.0
> round(cor(iris[,1:4]))
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length            1           0            1           1
Sepal.Width             0           1            0           0
Petal.Length            1           0            1           1
Petal.Width             1           0            1           1
>
```

La funzione corrplot()

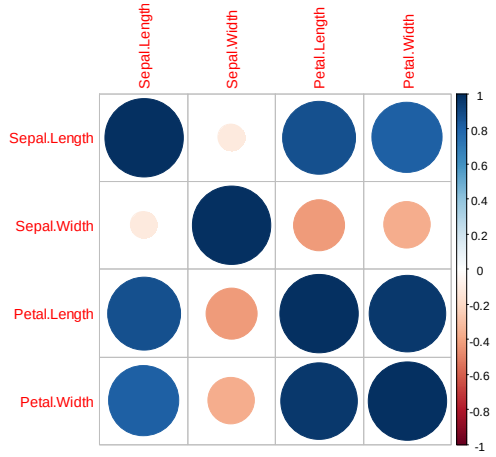
```
> cor(iris[,1:4])
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

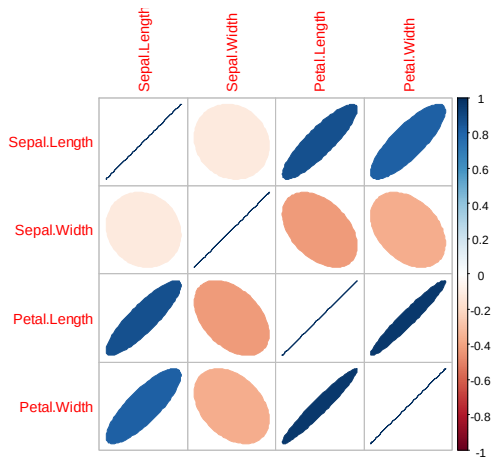
library(corrplot)

corrplot(nomematricecorrelazione)

Bisogna installare e caricare il pacchetto corrplot.

La funzione serve per visualizzare la matrice di correlazione






```
> CorM<-cor(iris[,1:4])
> corrplot(CorM)
```

```
> corrplot(CorM,method="ellipse")
```

```
> corrplot(CorM,method="number",type="upper")
```

PCA in R

1) Normalizzare i dati:

Centratura dei dati intorno alla media (per ogni variabile): necessario per poi ottenere la matrice di covarianza

```
DatasetN<-scale(Dataset, center= TRUE, scale=TRUE)
```

Scalatura dei dati rispetto alla deviazione standard (per ogni variabile): consigliato

2) Calcolare la matrice di covarianza:

```
B<-t(DatasetN)%*%DatasetN/(1-nrow(DatasetN))
```

3) Calcolare gli *eigenpair* della matrice di covarianza:

```
Eigen<-eigen(B)
```

Eigen$values → Autovalori

Eigen$vectors → Autovettori

PCA in R - tutto in una unica funzione

PCA<-prcomp (Dataset, center= TRUE, scale=TRUE)

```
> PCA<-prcomp(dataset,scale=TRUE,center=TRUE)
> attributes(PCA)
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

PCA$sdev → **PCA$sdev^2** Autovalori

PCA$rotation → Autovettori

PCA$center → Parametri di centratura per variabile (media)

PCA$scale → Parametri di scalatura per variabile (dev.std)

PCA$x → Nuove coordinate (score) per i dati sperimentali

PCA in R - costruzione dei grafici

Scree Plot →

```
varianze<- PCA$sdev^2
plot (varianze)
```

Varianza cumulata % →

```
varianzecum<- cumsum(varianze/sum(varianze) *100)
plot (varianzecum)
```

Loading →

```
plot (PCA$rotation[,a], PCA$rotation[,b])
```

Score →

```
plot (PCA$x [,a], PCA$x[,b])
```

Biplot →

```
biplot (PCA, choices =c(a,b))
```

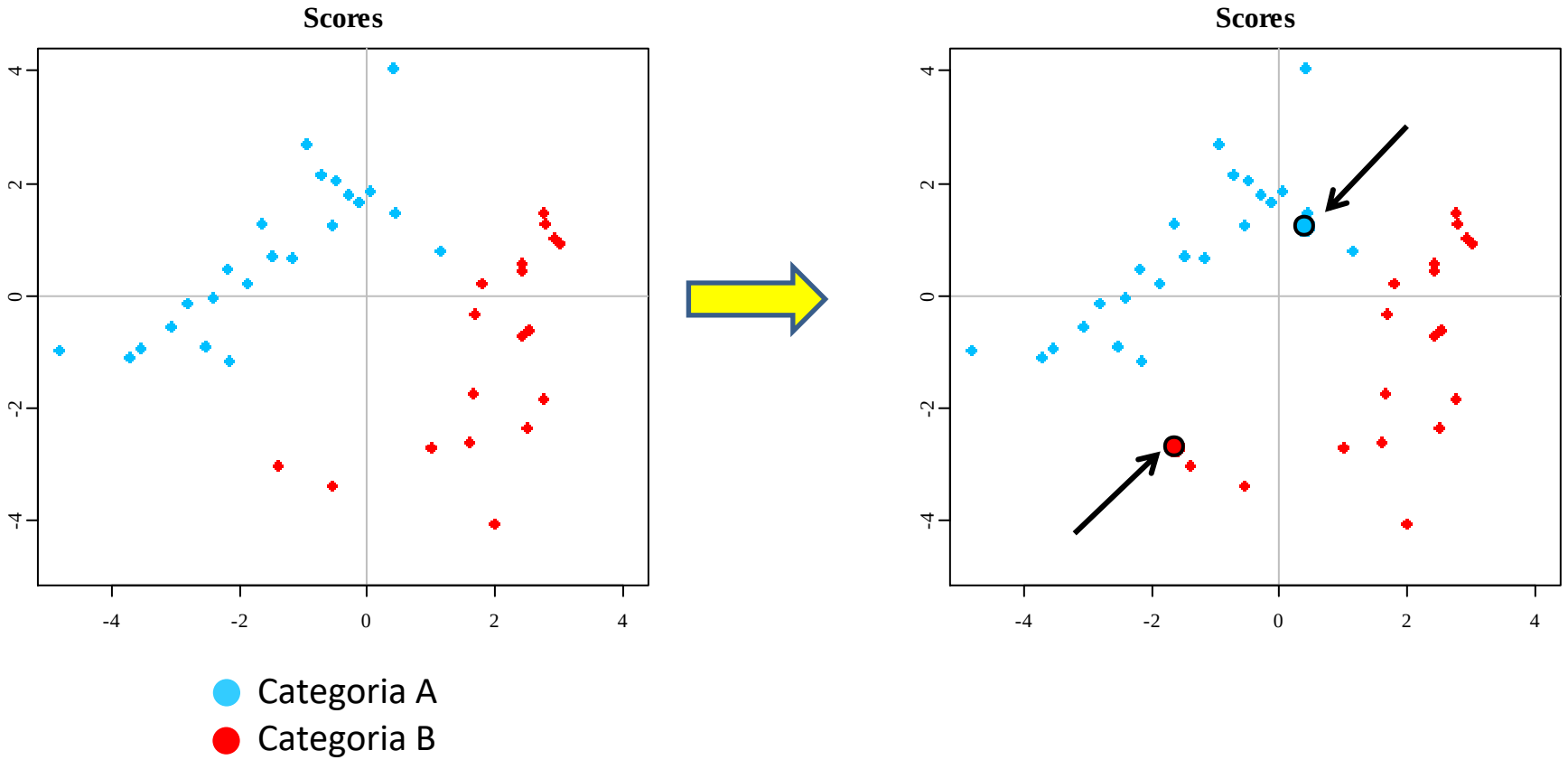
Proiezione di dati non utilizzati per costruire la PCA

```
New<-predict(PCA,newdata)
```

"newdata" deve avere lo stesso numero di colonne e nomi di colonna del Dataset!

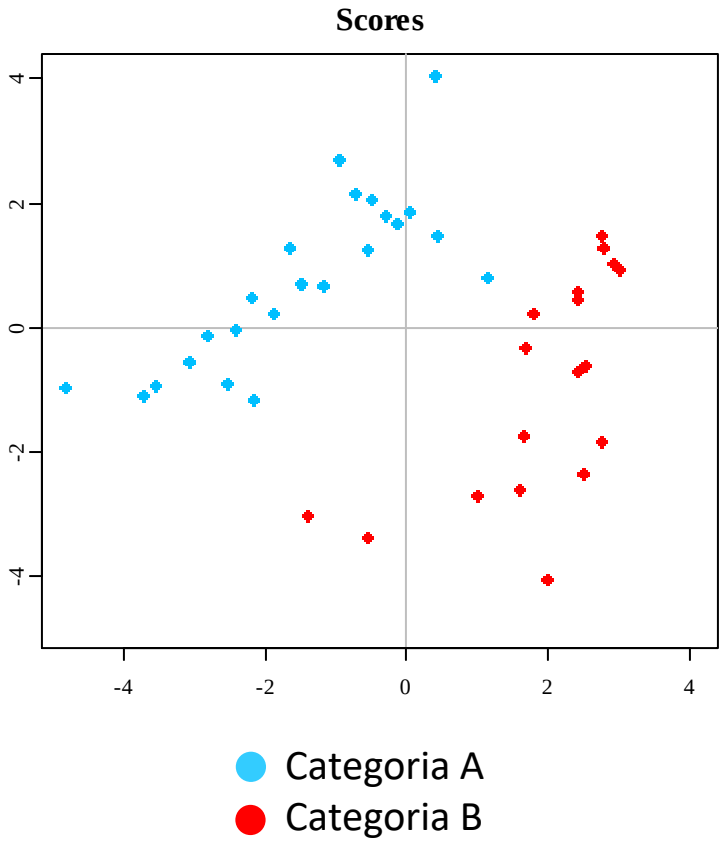
Utile se si conosce qualche categoria/classe a cui le osservazioni appartengono, es.

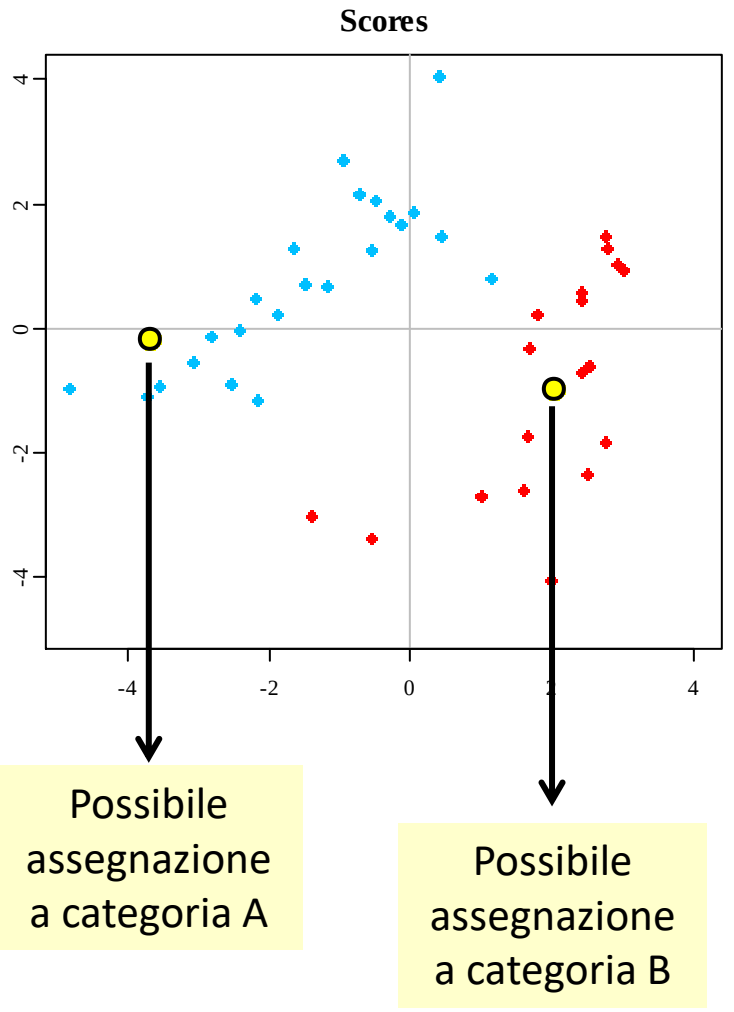
1) per conferma del modello:



Proiezione di dati non utilizzati per costruire la PCA

2) per attribuzione di una categoria/classe a nuovi dati:





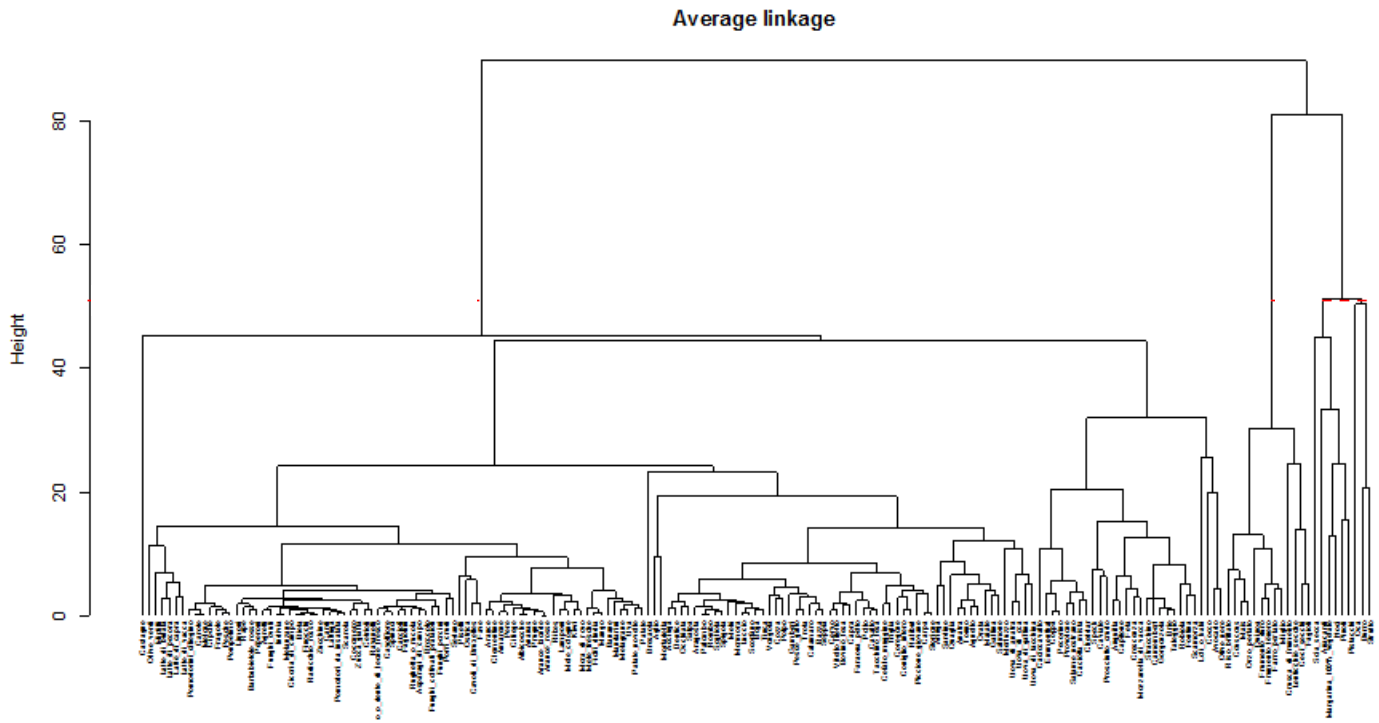
Il raggruppamento gerarchico in R

```
Dist<-dist(nomematrice, method="euclidean")
Hc<- hclust(Dist, method="complete")
Dend<-as.dendrogram(Hc)
```

→ Qua si decide il tipo di distanza

→ Qua si decide il tipo di linkage

```
plot(Dend)
```



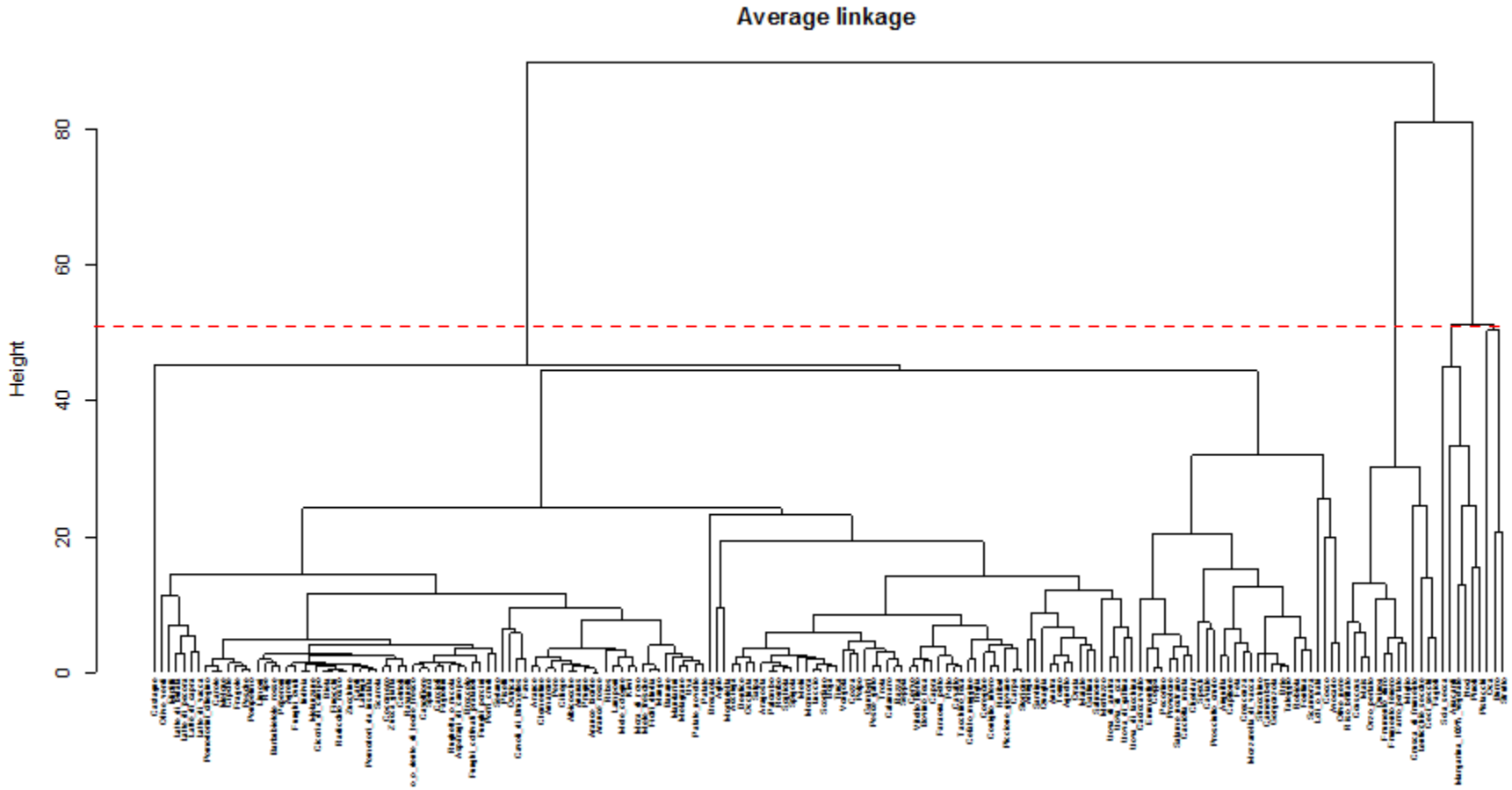
Fonte: http://www.sthda.com/english/wiki/beautiful-dendrogram-visualizations-in-r-5-must-known-methods-unsupervised-machine-learning#plot.dendrogram-function

Scegliere il numero di gruppi

```
Cut<-cutree(Hc, h= 51)
```

↓

Scelgo la distanza a cui "tagliare" i rami



Visualizzare boxplot per gruppo

Newdata<-data.frame(originale, H51= Cut)

```
> head(Newdata)
    Alimento Categoria Acqua Proteine Lipidi Carboidrati H51
1    Acciuga     Pesce  76.5     16.8    2.6       1.500   1
2      Aglio   Verdura  62.8      8.4    0.8       1.000   1
3    Agnello     Carne  70.1     20.0    8.8       0.001   1
4    Agretti   Verdura  92.3      1.8    0.2       2.200   1
5 Albicocche    Frutta  86.3      0.4    0.1       9.800   1
6    Amarene    Frutta  85.2      0.8    0.1      10.200   1
>
```

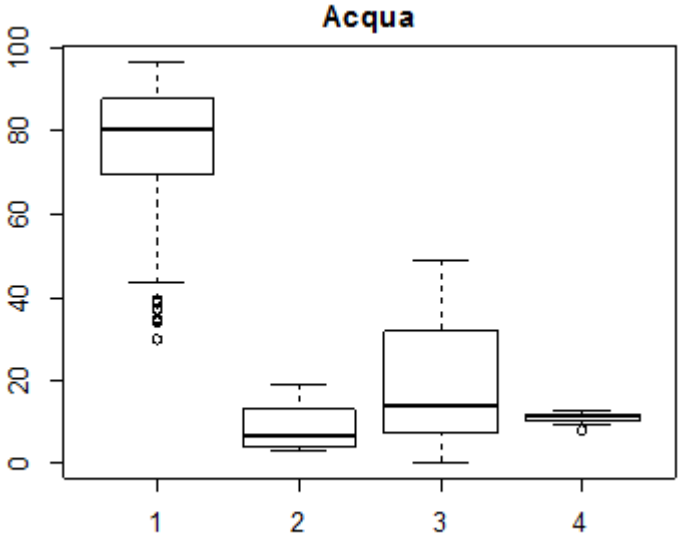
Visualizzare boxplot per gruppo

Newdata<-data.frame(originale, H51= Cut)

```
> head(Newdata)
    Alimento Categoria Acqua Proteine Lipidi Carboidrati H51
1    Acciuga     Pesce  76.5     16.8    2.6       1.500   1
2      Aglio   Verdura  62.8      8.4    0.8       1.000   1
3    Agnello     Carne  70.1     20.0    8.8       0.001   1
4    Agretti   Verdura  92.3      1.8    0.2       2.200   1
5 Albicocche    Frutta  86.3      0.4    0.1       9.800   1
6    Amarene    Frutta  85.2      0.8    0.1      10.200   1
>
```

Box<-boxplot(Newdata$Acqua~Newdata$H51)

(y~x)

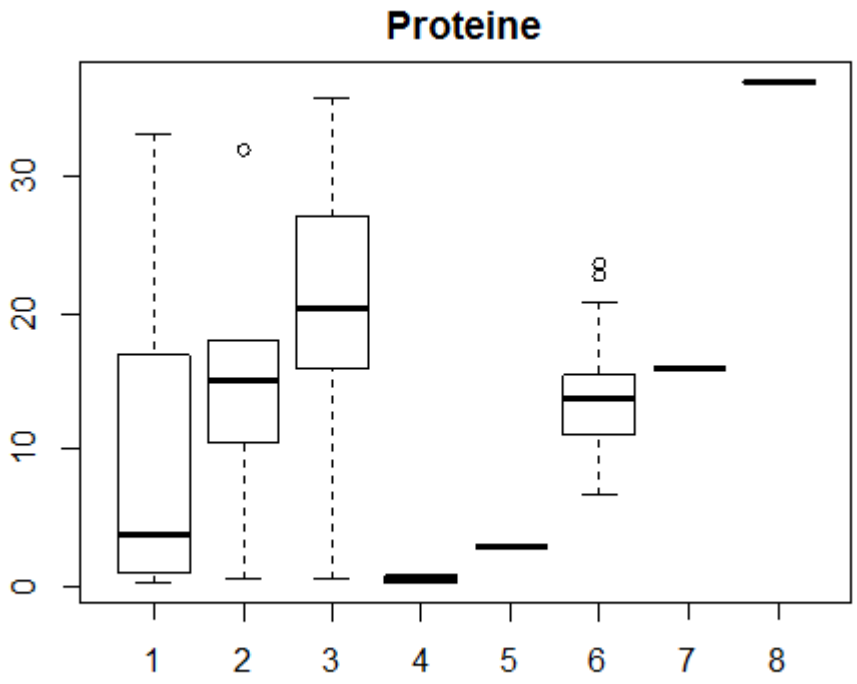


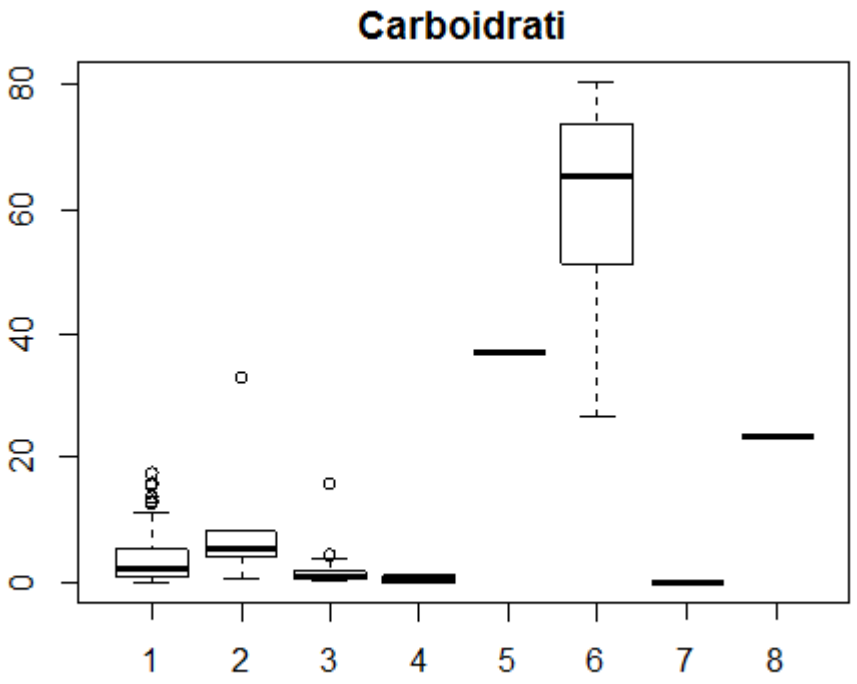

```
> Box$n
[1] 161   6   3  13
>
> Box$stats
     [,1] [,2] [,3] [,4]
[1,] 43.6  3.0  0.5  9.1
[2,] 69.7  3.9  7.3 10.5
[3,] 80.5  6.9 14.1 11.3
[4,] 87.5 13.0 31.6 12.0
[5,] 96.5 19.2 49.1 12.6
>
>
> Box$out
[1] 30.0 39.5 38.5 34.7 34.6 34.0 39.0 37.3  8.2
> Box$group
[1] 1 1 1 1 1 1 1 1 4
```

Scegliere il numero di gruppi

Cut<-cutree(Hc, k= 8) → Scelgo il numero presunto di cluster

Newdata2<-data.frame(Newdata, K8= Cut)





Esplorare i risultati

```
Cluster6<-Newdata2[which(Newdata2$K8==6),]
```

```
> Cluster6
              Alimento Categoria Acqua Proteine Lipidi Carboidrati H51 K8
42         Ceci_secchi    Legumi  10.3     20.9    6.3         46.9   4  6
56            Couscous   Cereali   9.1     13.7    1.1         76.5   4  6
59  Crusca_di_frumento   Cereali   8.2     14.1    5.5         26.6   4  6
62             Fagioli    Legumi  10.5     23.6    2.0         47.5   4  6
65       Farro_perlato   Cereali  11.3     14.6    2.4         69.3   4  6
72       Frumento_duro   Cereali  11.5     13.0    2.9         62.5   4  6
73     Frumento_tenero   Cereali  12.0     12.3    2.6         65.2   4  6
89   Lenticchie_secche    Legumi  11.2     22.7   17.5         51.1   4  6
94                Mais   Cereali  12.5      9.2    3.2         75.1   4  6
105             Miglio   Cereali  12.6     11.0    0.6         67.8   4  6
117       Orzo_perlato   Cereali  12.0      9.4    1.5         73.7   4  6
141             Quinoa   Cereali  10.8     15.4    8.1         57.8   4  6
147      Riso_brillato   Cereali  12.0      6.7    0.4         80.4   4  6
```

```
> Newdata2[which(Newdata2$K8==4),]
    Alimento Categoria Acqua Proteine Lipidi Carboidrati H51 K8
26     Burro Latticini  14.1      0.8   83.4       1.100   3  4
167  Strutto     Carne   0.5      0.3   99.0       0.001   3  4
>
```