DEA Analyse et Modélisation des Systèmes Biologiques

Introduction au logiciel S-PLUS©

D. Chessel

1 - Langage orienté objet

Résumé

Une fonction s'écrit f(objets, paramètres). Une partie d'un objet s'écrit objet[...]. Une fonction crée un objet objet<-f(objets[...]) ou un graphique. Les jeux du type f(objet[f(objet)]) ou objet[f(objet[f(objet)])] et le nombre de fonctions disponibles (2000) donne un langage. Les fonctions, les données, les formules, les résultats, les graphiques sont des objets.

Plan

1 - Introduction

Cette fiche suppose que l'on accède au logiciel S-PLUS©.

La fiche 1 de Jean Thioulouse permet la connexion au serveur, la configuration de la session, le lancement du logiciel, l'édition des commandes, la lecture et l'écriture des fichiers de données, l'ouverture et la fermeture d'une fenêtre graphique, la création de fonctions en langage S, la fermeture de la session. Elle s'adresse aux utilisateurs d'une station Unix ayant un compte sur les serveurs du laboratoire de Biométrie (licence pour 50 utilisateurs).

La fiche 2 de Jean Thioulouse permet la même chose à partir d'un micro-ordinateur connecté sur Internet. Elle permet aux étudiants du DEA de poursuivre leur prise de contact à partir des ordinateurs de leur laboratoire d'accueil.

On s'intéresse maintenant au contenu statistique du logiciel. Les illustrations sont choisies en rapport avec les trois cours de base du tronc commun de biostatistique, celui de Christian Gautier sur les statistiques non paramétriques, celui de Jean-Dominique Lebreton sur le modèle linéaire et celui de Jacques Estève sur le modèle linéaire généralisé.

Les illustrations sont préparées avec la version 4.3 de S-PLUS pour Windows 95. Elles n'utilisent pas l'interface graphique et sont reproductibles, pour l'essentiel avec la version 5 pour Unix. Le parti pris de la documentation d'origine — c'est un langage qu'on apprend en l'utilisant — a été respecté :

Statistical Sciences. (1995a) S-PLUS, User's manual, Version 3.3 for Windows. StatSci, a division of MathSoft, Seattle. 470 p.

Statistical Sciences. (1995b) S-PLUS, Programmer's manual, Version 3.2. StatSci, a division of MathSoft, Seattle. 425 p.

Statistical Sciences. (1995c) S-PLUS Guide to Statistical and Mathematical analysis, Version 3.3. StatSci, a division of MathSoft, Seattle. 650 p.

MathSoft. (1997a) S-PLUS User's Guide. Data Analysis Products Division, MathSoft, Seattle. 620 p.

MathSoft. (1997b) S-PLUS 4 Guide to Statistics. Data Analysis Products Division, MathSoft, Seattle. 877 p.

MathSoft. (1997c) S-PLUS Programmer's Guide. Data Analysis Products Division, MathSoft, Seattle. 582 p.

On peut trouver une très belle introduction en français au logiciel S-PLUS écrite par Marcel Baumgartner à Lausanne. En html :

http://dmawww.epfl.ch/~cerutti/intro/intro.html

et en fichier postscript :

http://dmawww.epfl.ch/Stat.mosaic/intro.ps

Conventions typographiques:

scan *Désigne un nom de fonction ou d'opérateur rencontré pour la première fois*

`> objects()` *Chaîne de caractères tapés au clavier après le prompt (>)*

`Residual standard error: 1.16 on 8 degrees of freedom` *Listing des réponses*

Creates a plot on the current graphics device. *Extrait de la documentation*

2 - Vecteurs

L'objet de base est le vecteur :

<- *Affectation*

```
> w0<-7 affecte à l'objet de nom w0 la valeur 7
> is.vector(w0)
[1] T   T pour True Wo est un vecteur
> w0
[1] 7   La première composante de Wo est 7
```

: *Série d'entiers*

```
> w0<-1:12
> w0
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

c *Combinaison*

```
> w0<-c(2,5,-3,8,"a")
> w0    w0  est un vecteur de chaînes de caractères
[1] "2"  "5"  "-3" "8"  "a"
> mode(w0)
[1] "character"

> w0<-c(1,5,-36,3.66)
> w0    w0  est un vecteur numérique
[1]   1.00   5.00 -36.00   3.66
> mode(w0)
[1] "numeric"

> w0<-c(T,T,T,F,F)
> w0    w0  est un vecteur logique
[1] T T T F F
> mode(w0)
[1] "logical"
```

scan *Saisie au clavier*

```
> w0<-scan()
1: 145
2: -1
3: 28.88
4: 2e-02
5: 3.45e1
6: Le premier retour-charriot après une chaîne vide met fin à la saisie
> w0
[1]  145.00   -1.00   28.88    0.02   34.50
```

[ ] *Éléments des vecteurs*

```
> w0[2:3] du second au troisième
[1]  -1.00  28.88
> w0[4] le quatrième
[1] 0.02
> w0[-4] tous sauf le quatrième
[1]  145.00   -1.00   28.88   34.50
```

3 - Tableaux

read.table *Saisie dans un fichier texte*

Dans l'éditeur de texte :

```
STA    SEM  HEU  RIC  Séparateur tabulation entre les en-têtes de
colonnes
```

```
3 2    1    5         Séparateur tabulation entre les nombres
3 2    2    3         idem
3 3    1    5         idem
3 3    2    3         idem
...
```

Dans S-PLUS :

DESCRIPTION

Reads in a file in table format and creates a data frame with the same number of rows as there are lines in the file, and the same number of variables as there are fields in the file.

USAGE

```
read.table(file, header=<<see below>>, sep, row.names, col.names,
       as.is=F, na.strings="NA", skip=0)
```

```
> read.table("D:\\Dea\\Dea2\\Ecrin.txt", header=T, sep="\t")
     STA SEM HEU RIC
   1   3   2   1   5
   2   3   2   2   3
   3   3   3   1   5
...

> ecrin<-read.table("D:\\Dea\\Dea2\\Ecrin.txt", header=T, sep="\t")
```

`ecrin` contient le nombre d'espèces d'Oiseaux (variable RIC) compté dans 1315 relevés ornithologiques exécutés dans le parc National des Ecrins en 1991. Chaque relevé est exécuté dans une station (1 à 14, variable STA), au cours d'une semaine de l'année (variable SEM de 1 à 52), soit le matin, soit le soir (variable HEU, 1-matin, 2-soir). Ces données sont extraites de la convention d'étude n° 228/92 du Parc National des Ecrins.

```
> is.vector(ecrin) ecrin n'est pas un vecteur
[1] F
> is.matrix(ecrin) ecrin est une matrice
[1] T
> is.data.frame(ecrin) ecrin est un data.frame
[1] T
> class(ecrin)
[1] "data.frame"
```

ecrin est un data.frame (tableau dont les colonnes ont des propriétés variées comme nom, type, ...). On peut mélanger dans un data.frame les variables quantitatives, qualitatives, logiques et textuelles. Une matrice ne contient qu'un type d'enregistrements. `ecrin` appartient à la classe des data.frame laquelle contient la classe des matrices. Une matrice n'est pas un data.frame mais peut le devenir :

```
> w0<-matrix("a",nrow=3,ncol=2)
> w0
     [,1] [,2]
[1,] "a"  "a"
[2,] "a"  "a"
[3,] "a"  "a"
> is.data.frame(w0)
[1] F
> as.data.frame(w0)
  V1 V2
1  a  a
2  a  a
3  a  a
```

[ ] *Éléments des matrices*

```
> ecrin[1:4,]
  STA SEM HEU RIC
1   3   2   1   5
2   3   2   2   3
3   3   3   1   5
4   3   3   2   3
> dim(ecrin)
[1] 1315    4
> ecrin[1312:1315,]
     STA SEM HEU RIC
1312  12  51   1   5
1313  12  51   2   4
1314  12  52   2   4
1315  12  52   1   7
```

Les variables d'un data.frame sont directement accessibles par leur nom :

```
> ric<-ecrin$RIC
> is.vector(ric)
[1] T
```

object *Liste des objets du dossier de travail*

```
> objects()
[1] ".Last.value" "ecrin"       "last.dump"   "ric"
```

ecrin$RIC ou ric sont strictement équivalents.

rm *Détruire un objet*

```
> objects()
[1] ".Last.value"  "ecrin"        "last.dump"    "last.warning"
[5] "ric"
> rm(ric)
> objects()
[1] ".Last.value"  "ecrin"        "last.dump"    "last.warning"
> is.vector(ric)
Error: Object "ric" not found
Dumped
> ric<-ecrin$RIC
```

4 - Fonctions

summary/plot *Fonctions génériques*

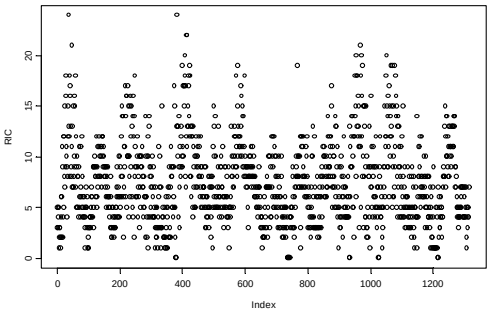
```
> ?summary

DESCRIPTION

Provides a synopsis of an object.

This function is generic (see Methods); method functions can be written
to handle specific classes of data. Classes which already have methods
for this function include:
aov, aovlist, data.frame, factor, gam, glm, lm, loess, mlm, ms, nls,
ordered, terms, tree.

USAGE

summary(object, ...)

> ?plot

DESCRIPTION
```

```
Creates a plot on the current graphics device.

This function is generic (see Methods); method functions can be written
to handle specific classes of data. Classes which already have methods
for this function include:
data.frame, design, factor, formula, gam, glm, lm, loess, preplot.gam,
preplot.loess, profile, stl, surv.fit, times, tree, tree.sequence.

USAGE

plot(x, ...)
```

Les fonctions génériques s'appliquent à plusieurs types d'objets qui peuvent être **très** différents entre eux. Une fonction générique *func* fonctionne sur une classe d'objets *clas* si il existe une fonction *func.clas*. Par exemple, plot et summary travaille sur des data.frames, des facteurs, des modèles linéaires généralisés, ...

```
plot.data.frame(data, labels = dimnames(data)[[2]], ...)
summary.data.frame(object, maxsum, ...)
plot.factor(x, y=NULL, style="box", rotate=<<see below>>, ...)
summary.factor(object, maxsum)
plot.glm(glm.obj, residuals = NULL, smooths = F, rugplot = F, , ...)
summary.glm(object, dispersion=NULL, correlation=T)
...
```

```
> summary(ecrin$RIC)
 Min. 1st Qu. Median Mean 3rd Qu. Max.
    0       4      7 7.48      10   24

> plot(ecrin$RIC)
```




```
> ric[2] la seconde composante de ric
[1] 3
> ric[c(1,10,100)] les composantes de rang 1, 10 et 100
[1] 5 3 2
> ric[ric<1] les composantes de ric pour lesquelles ric est < 1 !
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> length(ric)
[1] 1315
```

append/replace

Ajouter des valeurs, remplacer des valeurs

```
> ric[ric>20]
[1] 24 21 24 22 22 21
> (1:1315)[ric>20]
[1]  38  48 382 414 415 968
> ric[(1:1315)[ric>20]]
[1] 24 21 24 22 22 21
```

ric>20, 1:1315, (1:1315)[ric>20], ric, ric[ric>20] *sont des vecteurs*

unique/sort/rank/order

```
> x<-ric[1:10]
> x
 [1] 5 3 5 3 4 1 5 3 2 3 Le vecteur x
> unique(x)
[1] 5 3 4 1 2 les valeurs que peuvent prendre les composantes du vecteur
> sort(x)
 [1] 1 2 3 3 3 3 4 5 5 5 le vecteur des composantes rangées par ordre
croissant
> rank(x)
 [1] 9.0 4.5 9.0 4.5 7.0 1.0 9.0 4.5 2.0 4.5 le vecteur des rangs des
composantes
> order(x)
 [1]  6  9  2  4  8 10  5  1  3  7 la permutation qui donne le rangement
par ordre croissant
> x[order(x)]
 [1] 1 2 3 3 3 3 4 5 5 5
> sort(x)
 [1] 1 2 3 3 3 3 4 5 5 5
> x[order(x)]==sort(x)
 [1] T T T T T T T T T T le vecteur qui donne la comparaison élément par
élément des deux vecteurs
> x[order(x)]>sort(x)
 [1] F F F F F F F F F F
```

sum/prod/cumsum/cumprod/diff

```
> sum(x)
[1] 34
> prod(x)
[1] 81000
> cumsum(x)
 [1]  5  8 13 16 20 21 26 29 31 34
> cumprod(x)
 [1]     5    15    75   225   900   900  4500 13500 27000 81000
> diff(x)
[1] -2  2 -2  1 -3  4 -2 -1  1
> diff(x,lag=2)
[1]  0  0 -1 -2  1  2 -3  0
```

Calcul sur les vecteurs

```
> 2*x
 [1] 10  6 10  6  8  2 10  6  4  6
> x+10
 [1] 15 13 15 13 14 11 15 13 12 13
> abs(2*x-10)
 [1] 0 4 0 4 2 8 0 4 6 4
> log(x)
 [1] 1.6094 1.0986 1.6094 1.0986 1.3863 0.0000 1.6094 1.0986 0.6931
[10] 1.0986
> sqrt(x)
 [1] 2.236 1.732 2.236 1.732 2.000 1.000 2.236 1.732 1.414 1.732
```

rep/seq/rev

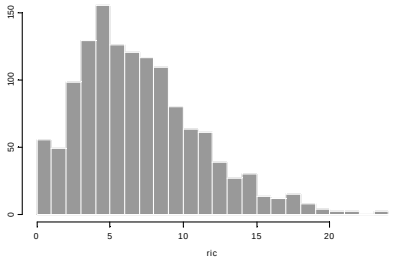
rep pour faire des répétitions, seq pour des suites, rev pour inverser :

```
> rep("a",3)
[1] "a" "a" "a"
> rep(1,3)
[1] 1 1 1
```

```
> rep(c(1,2),3)
[1] 1 2 1 2 1 2
> rep(c(1,2),c(3,3))
[1] 1 1 1 2 2 2
> seq(from=1,to=10,by=0.5)
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5
[13]  7.0  7.5  8.0  8.5  9.0  9.5 10.0
> seq(from=1,to=10,le=3)
[1]  1.0  5.5 10.0
> seq(from=1,to=10,le=5.5)
[1]  1  3  5  7 10
> seq(from=1,to=10,by=4)
[1] 1 5 9
> rep(1:3,1:3)
[1] 1 2 2 3 3 3
> rev(6:12)
[1] 12 11 10  9  8  7  6
```

Statistiques élémentaires

```
> hist(ric,nclass=30)
```




```
> min(ric)
[1] 0

> max(ric)
[1] 24
> range(ric)
[1]  0 24
> mean(ric)
[1] 7.477
> median(ric)
[1] 7
> quantile(ric, probs=seq(from=0, to=1,by=0.1))
  0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
   0    3    4    5    6    7    8    9   11   13   24
```

Les fonctions sont des objets

```
> mean
function(x, trim = 0, na.rm = F)
{
   if(na.rm) {
        wnas <- which.na(x)
        if(length(wnas))
             x <- x[ - wnas]
   }
   if(mode(x) == "complex") {
        if(trim > 0)
             stop("trimming not allowed for complex data"
                  )
        return(sum(x)/length(x))
   }
   x <- as.double(x)
   if(trim > 0) {
        if(trim >= 0.5)
```

```
            return(median(x, na.rm = F))
        if(!na.rm && length(which.na(x)))
            return(NA)
        n <- length(x)
        i1 <- floor(trim * n) + 1
        i2 <- n - i1 + 1
        x <- sort(x, unique(c(i1, i2)))[i1:i2]
    }
    sum(x)/length(x)
}
```

Que signifie trim ? Voir aussi :

ceiling/floor/trunc

5 - Facteurs

```
> sem<-ecrin$SEM
> is.numeric(sem)
[1] T
> summary(sem)
 Min. 1st Qu. Median Mean 3rd Qu. Max.
    1      13     26 26.2      39   52
```

sem est un variable quantitative

```
> sem.fac<-factor(sem)
> summary(sem.fac)
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
 18 25 26 28 28 25 28 28 27 26 26 28 28 23 26 26 27 27 26 24 24 27
 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
 24 27 27 24 24 28 24 23 22 25 23 21 26 27 25 28 21 27 24 26 24 24
 45 46 47 48 49 50 51 52
 26 25 25 23 25 28 25 23

> is.numeric(sem.fac)
[1] F
```

sem.fac est un variable qualitative. Ses modalités sont :

```
> levels(sem.fac)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12"
...
[49] "49" "50" "51" "52"

> heu<-ecrin$HEU
> heu.fac<-factor(heu)
> levels(heu.fac)
[1] "1" "2"
> levels(heu.fac)<-c("Ma","So")
> summary(heu.fac)
  Ma  So
 672 643
> heu.fac[1:10]
 [1] Ma So Ma So Ma So Ma So So Ma
```

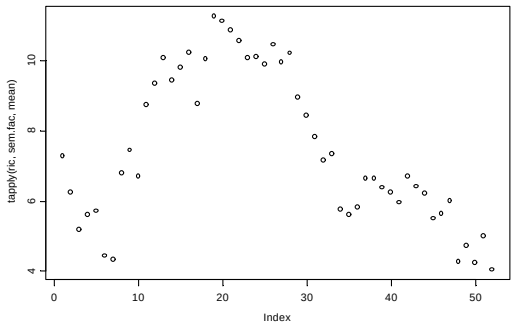
table ***tables de contingence***

```
> table(heu.fac) # Tabuler un facteur
  Ma  So
 672 643
> table(heu.fac,sem.fac) # Tabuler deux facteurs
   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
```

```
Ma 9 12 13 14 14 12 14 14 13 13 13 14 14 11 14 13 14 14 13 12 12
So 9 13 13 14 14 13 14 14 14 13 13 14 14 12 12 13 13 13 13 12 12
...
   43 44 45 46 47 48 49 50 51 52
Ma 12 12 12 13 14 12 13 14 14 12
So 12 12 14 12 11 11 12 14 11 11

> sta.fac<-factor(ecrin$STA)

> table(sta.fac,heu.fac,sem.fac) # Tabuler trois facteurs

, , 1 par semaine
   Ma So
 1  0  0
 2  0  0
 3  0  0
 4  0  0
 5  1  1
...
14  1  1

, , 2
   Ma So
 1  1  1
...
 5  0  0
 6  0  1
 7  1  1
...
14  1  1

, , 3
   Ma So
 1  1  1
 2  1  1
...
```

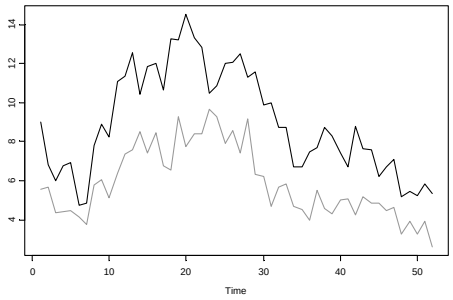
tapply *Appliquer une fonction à un vecteur par blocs*

```
> tapply(ric,heu.fac,mean)
    Ma    So
 8.972 5.914
> plot(tapply(ric,sem.fac,mean))
```




```
> tapply(ric,list(sem.fac,heu.fac),mean)
       Ma    So
 1  9.000 5.556
 2  6.833 5.692
 3  6.000 4.385
...
50  5.214 3.286
51  5.857 3.909
52  5.333 2.636
```

```
> ric.sem.heu<-tapply(ric,list(sem.fac,heu.fac),mean)
> tsplot(ric.sem.heu)
```




```
> ts.points(ric.sem.heu)
```

DESCRIPTION

Adds a legend to the current plot. The location and contents of the legend can be specified.

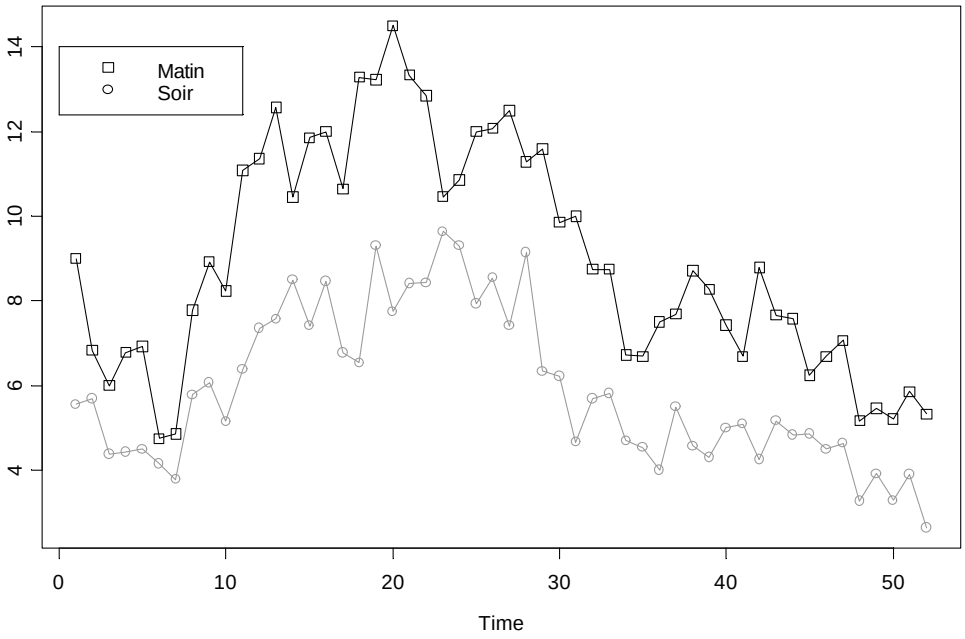
USAGE

```
legend(x, y, legend, angle = <<see below>>, density = <<see below>>,
       fill = <<see below>>, col = <<see below>>, lty = <<see below>>,
       lwd = <<see below>>, marks = <<see below>>, pch = <<see below>>,
       ncol = 1, background = 0)
```

REQUIRED ARGUMENTS

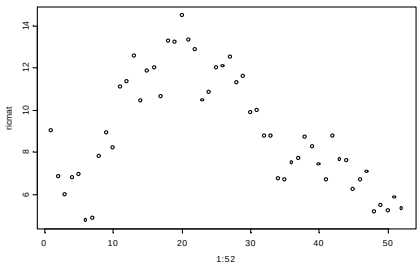
x,y location of the rectangle in which to put the legend. If x and y are length 1, they determine the top left corner of the rectangle; if they are length 2 vectors, they give opposite corners of the rectangular area. A list containing x and y values may be supplied.
legend vector of character strings to be associated with shading patterns, line types, plotting characters or marks.

```
> legend(0,14,c("Matin","Soir"),lty=c(1,2))
```



6 - Lisseurs

```
> ricmat<-ric.sem.heu[,1]
> ricmat
 1     2 3     4     5    6     7     8     9    10    11    12
 9 6.833 6 6.786 6.929 4.75 4.857 7.786 8.923 8.231 11.08 11.36
...
    47    48    49    50    51    52
 7.071 5.167 5.462 5.214 5.857 5.333
> plot(1:52,ricmat)
```

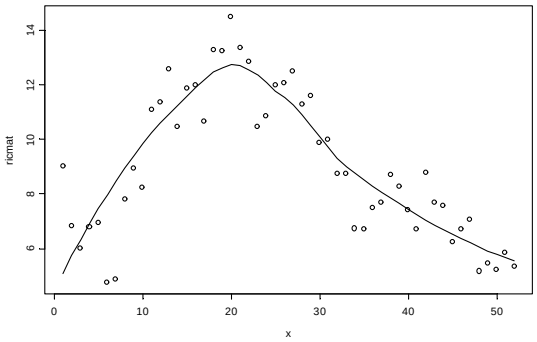



```
> x<-1:52
```

loess

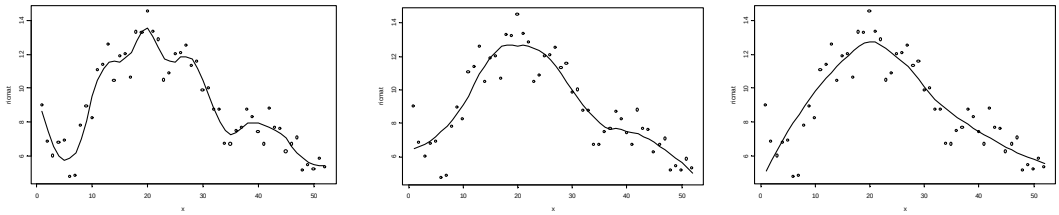
```
> loess(ricmat~x)
Call:
loess(formula = ricmat ~ x)

 Number of Observations:          52
 Equivalent Number of Parameters: 4.4
 Residual Standard Error:         1.317
 Multiple R-squared:              0.78
 Residuals:
    min   1st Q  median  3rd Q   max
 -3.622 -0.5629 0.03622 0.7652 3.894
> loess1<-loess(ricmat~x) loess1 est une liste
```

```
> lines(x,loess1[[1]])
```



pas mal ! Oui, mais :

```
> plot(x,ricmat)
> lines(x,predict.loess(loess(ricmat~x,span=0.25)))
> plot(x,ricmat)
> lines(x,predict.loess(loess(ricmat~x,span=0.50)))
> plot(x,ricmat)
> lines(x,predict.loess(loess(ricmat~x,span=0.75)))
```

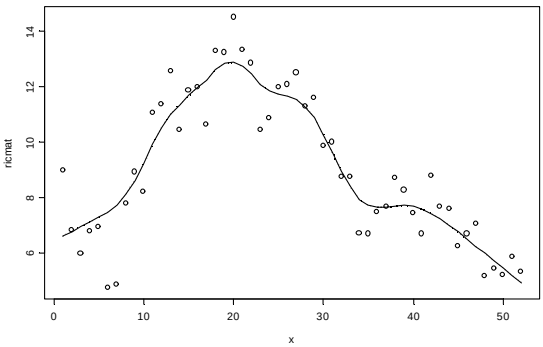


In medio stat virtus ?

```
> plot(x,ricmat)
```

supsmu

```
> lines(supsmu(x,ricmat))
```



BACKGROUND

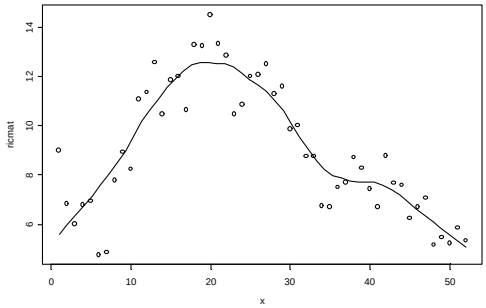
The supsmu function serves a purpose similar to that of the function lowess. supsmu is much faster although it does not have the robustness properties of lowess. For small samples (n<40), or if there are substantial serial correlations between observations close in x-value, a prespecified fixed span smoother (span > 0) should be used. Reasonable span values are from 0.3 to 0.5.

Moralité : S-PLUS contient un niveaud'expertise statistique considérable. A utiliser avec une citation, c'est la moindre des choses :

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74, 829-836.

Chambers, J. M., Cleveland, W. S., Kleiner, B. and Tukey, P. A. (1983). *Graphical Methods for Data Analysis*. Wadsworth, Belmont, California.

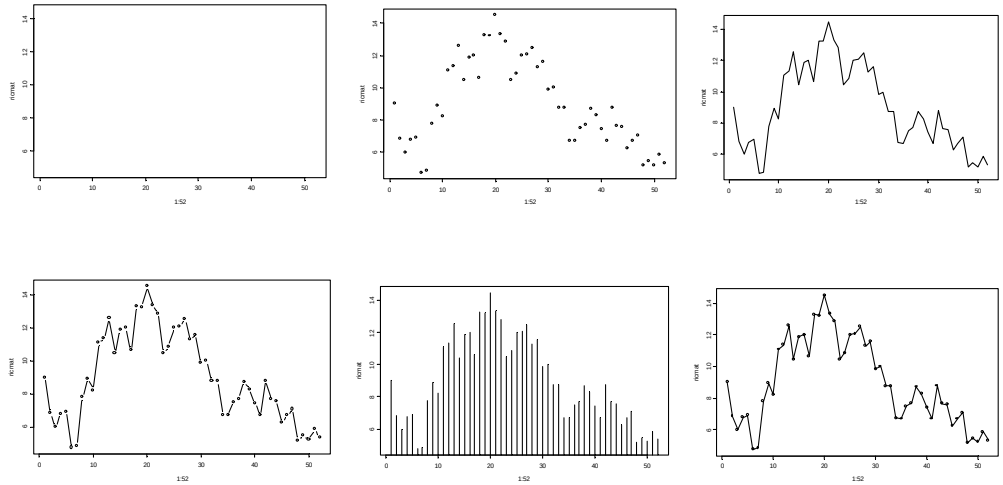
```
> scatter.smooth(x,ricmat,span=0.30)
```



7 - Graphiques

7.1 - plot

```
> plot(1:52,ricmat,type="n")
> plot(1:52,ricmat,type="p")
> plot(1:52,ricmat,type="l")
> plot(1:52,ricmat,type="b")
> plot(1:52,ricmat,type="h")
> plot(1:52,ricmat,type="o")
```



Pour gérer les superpositions :

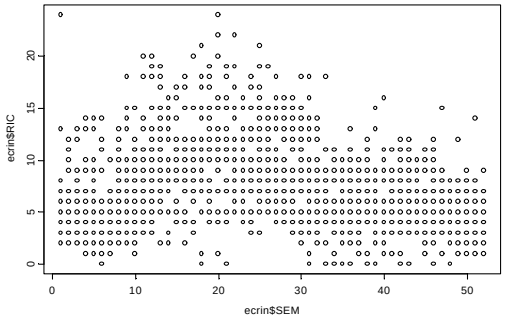
DESCRIPTION

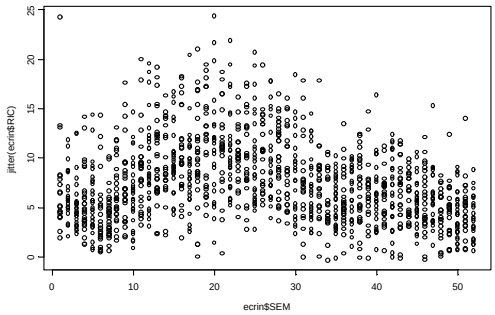
Returns an object like the input but with values "jittered" by the addition of random noise.

USAGE

jitter(x, factor=1)

```
> plot(ecrin$SEM,ecrin$RIC)
> plot(ecrin$SEM,jitter(ecrin$RIC))
```





7.2 - boxplot

DESCRIPTION

Produces side by side boxplots from a number of vectors. The boxplots can be made to display the variability of the median, and can have variable widths to represent differences in sample size.

USAGE

```
boxplot(..., range=1.0, width=<<see below>>, varwidth=F,
      names=<<see below>>, plot=T, notch=F, style.bxp=list(),
      boxwex=.5, boxcol=3, medchar=F, medpch=NA, medline=T, medlwd=5,
      medcol=0, confint=F, confcol=2, confangle=45, confdensity=25,
      confnotch=F, whisklty=2, staplelty=1, staplewex=1, staplehex=1,
      outchar=F, outpch=NA, outline=T, outwex=1)
```

A utiliser avec :

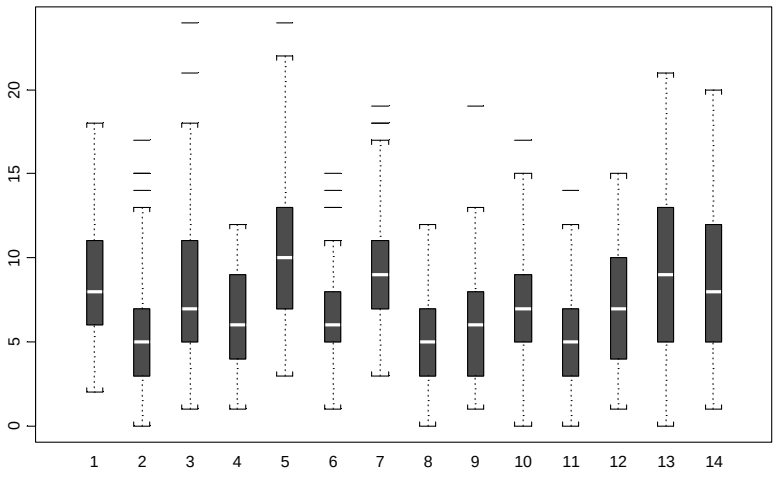
DESCRIPTION

Returns a list in which each component is a vector of values from data that correspond to a unique value in group.

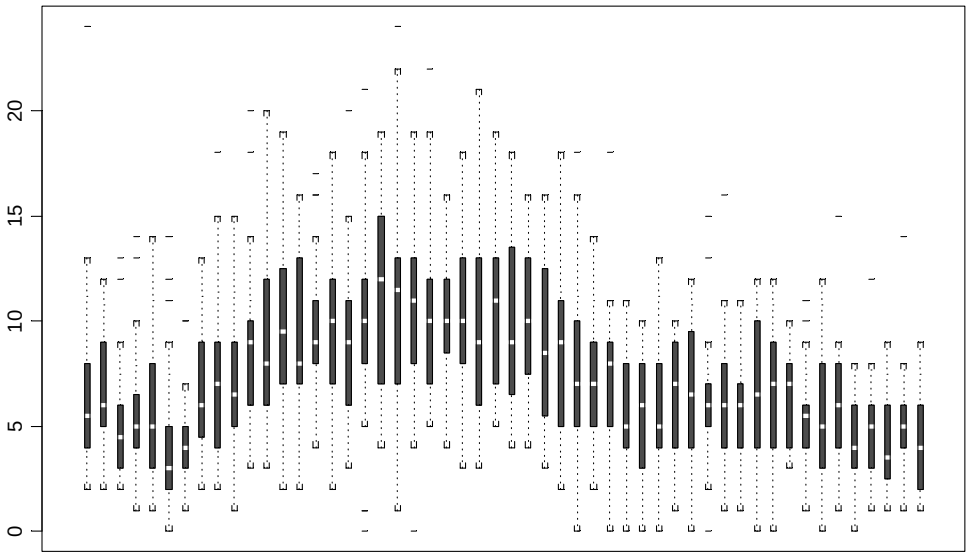
USAGE

```
split(data, group)
```

```
> boxplot(split(ric,sta.fac))
```

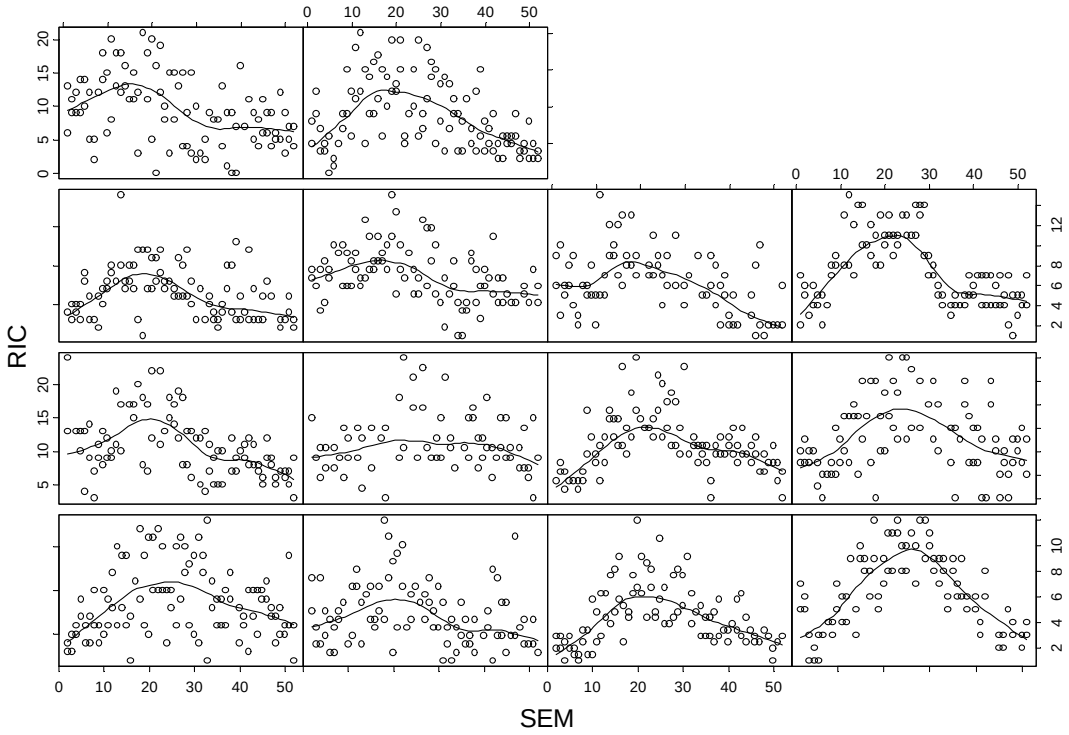



```
> boxplot(split(ric,sem),names=rep("",52))
```



7.3 - coplot

```
> coplot(RIC~SEM|factor(STA),data=ecrin,show.given=F,panel = function(x,
y) scatter.smooth(x,y, span = 0.4))
```



DESCRIPTION

The coplot function produces a Conditioning Plot in the current graphics device which shows how a response depends on a predictor given another predictor.

USAGE

```
coplot(formula, data, given.values, panel = points, rows, columns,
     column.row, show.given = T, add = F, xlab, ylab, xlim, ylim, ...)
```

REQUIRED ARGUMENTS

formula formula defining the response and the predictors involved in the plotting.

OPTIONAL ARGUMENTS

data data frame in which the formula will be evaluated. If missing, evaluation will take place as if the formula were evaluated in the frame of the function calling coplot.
given.values a numeric vector, character vector, or two-column matrix that specifies the given values when there is one given predictor, or a list of two such objects when there are two. If missing, reasonable things happen.
panel a user-supplied function of x and y that determines the method of plotting on the dependence panels.
rows for the case of one given predictor, the number of rows of the matrix of dependence panels. If missing, the following is the default: let k be the number of given values; if columns is missing, then it is ceiling(sqrt(k)) and otherwise it is ceiling(k/columns). This argument is not used if there are two given predictors.
columns for the case of one given predictor, the number of columns of the matrix of dependence panels. If missing, the following is the default: let k be the number of given values; if rows is missing, it is ceiling(k/ceiling(sqrt(k))) and otherwise it is ceiling(k/rows). This argument is not used if there are two given predictors.

column.row when conditioning on only one predictor, a vector of length 2 describing the arrangement of panels in the plot, i.e. a vector of the form c(n,m) where n and m are integers indicating the number of columns and rows, respectively for the arrangement of the dependence panels. The default is given by the arguments columns and rows which are overriden by this argument, if specified.
show.given if FALSE, given panels are not included.
add if TRUE, add to the current plot.

```
> coplot(RIC~SEM|factor(STA),data=ecrin,show.given=F,panel = function(x,
y) scatter.smooth(x,y, span = 0.3))
```

7.4 - xyplot

USAGE

xyplot(formula, ...)

The following arguments have special meaning within this function.
The common meanings for these and all other arguments are listed separately
under trellis.args.

formula a formula in the form

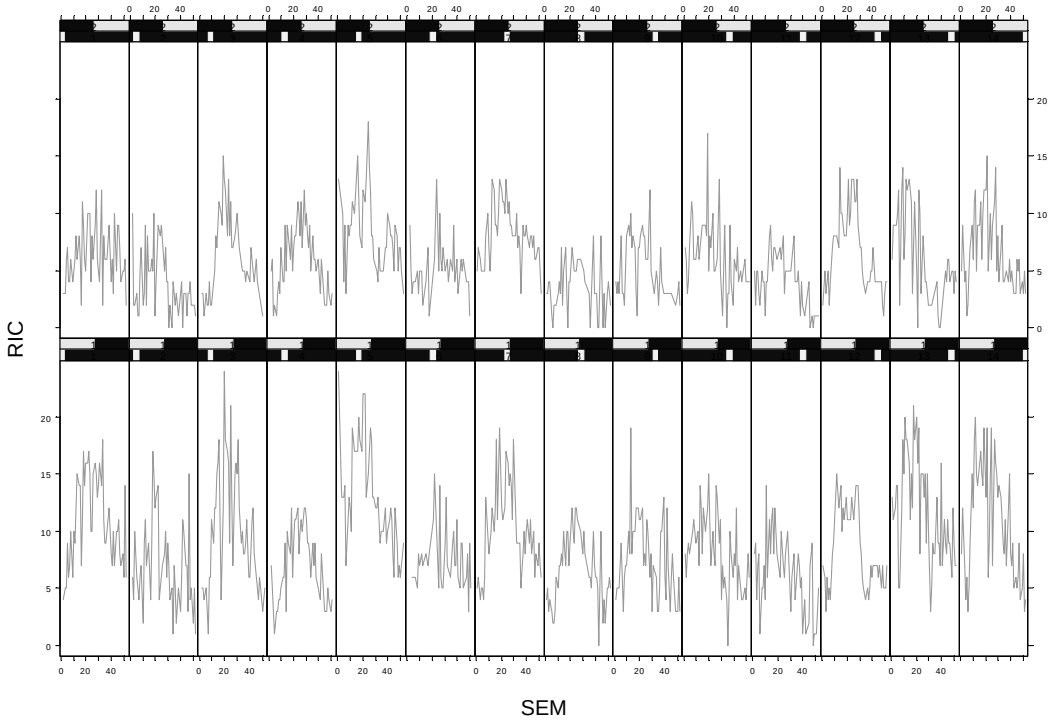
y ~ x | g1 * g2 * ...

however the given variables g1, g2, ... may be omitted. The x and y values in formula should both be numeric.

VALUE

an object of class trellis, which is automatically plotted by print.trellis.

```
> xyplot(RIC~SEM|factor(STA)*factor(HEU),
   data=ecrin[order(ecrin$SEM),],type="l")
```



Ces premiers exemples peuvent vous inviter à consulter un ouvrage fabuleux :

Cleveland, W.S. (1994) *The elements of graphing data*. AT&T Bell Laboratories, Murray Hill, New Jersey. 297 p.