Les graphiques dans R
par
Odile Wolber

R propose de nombreux outils graphiques pour l'analyse et la visualisation des données. Des fonctions graphiques sont fournies par les packages graphics, grid et lattice.

Au § 2, nous présentons les principales fonctions disponibles du package graphics. Nous n'aborderons pas les courbes de niveau (fonctions contour, filled.contour, image et persp), ni la fonction qui permet d'obtenir le graphe des effets partiels d'un modèle de régression (fonction termplot, qui sera traitée dans la prochaine session).

Au § 5, nous présentons des exemples d'utilisation de quelques fonctions du package lattice.

Nous ne traiterons pas les graphiques du package grid. Il s'agit d'un nouveau mode graphique avec son propre système de paramètres graphiques qui sont distincts de ceux vus ci-dessus. Les deux distinctions principales entre grid et le mode graphique de base sont :

- plus de flexibilité pour diviser les périphériques graphiques à l'aide des vues (viewports) qui peuvent être chevauchantes ;
- les objets graphiques (grob) peuvent être modifiés ou effacés d'un graphe sans avoir à le redessiner (comme doit êetre fait avec le mode graphique de base).

Les graphiques obtenus avec grid ne peuvent pas être combinés ou mélangés avec ceux produits par le mode graphique de base. Les deux modes graphiques peuvent cependant être utilisés dans la même session sur le même périphérique graphique.

On peut avoir un aperçu des possibilités graphiques de grâce à la commande demo(graphics).

1. Gestion des fenêtres graphiques

Lorsqu'une fonction graphique est exécutée, R ouvre une fenêtre graphique et y affiche le graphe. On peut spécifier le dispositif de gestion des fenêtres. La liste des dispositifs graphiques disponibles dépend du système d'exploitation.

Sous Mac, pour créer une nouvelle fenêtre graphique, on utilise la commande get("quartz")(). Sous PC, pour créer une nouvelle fenêtre graphique, on utilise la commande X11().

Tant sur Mac que sur PC, pour sélectionner une des fenêtres, on utilise la commande dev.set(). Par exemple, pour sélectionner la fenêtre graphique numéro *i*, on tape la commande dev.set(i). Si on souhaite connaître le numéro de la fenêtre active, on tape la commande dev.cur().

R propose deux manières de partitionner les graphiques :

① La commande split.screen(c(1, 2)) divise le graphique en deux parties qu'on sélectionne avec les commandes screen(1) et screen(2).

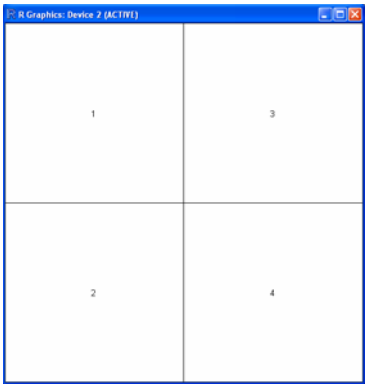
② La fonction layout() partitionne le graphique actif en plusieurs parties sur lesquelles sont affichés les graphes successivement. Cette fonction a pour argument une matrice de valeurs entières qui indiquent le numéro des sous-fenêtres.

Pour visualiser la partition crée, on utilise la fonction layout.show avec, en argument, le nombre de sous-fenêtres.

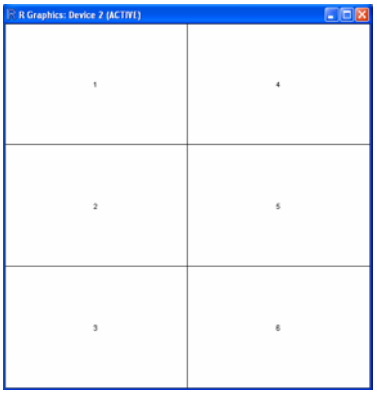
Par exemple, si on veut diviser la fenêtre en quatre parties égales, on tape la commande suivante :

layout(matrix(1:4, 2, 2)) ;
et pour visualiser la partition créée, on utilise la commande layout.show(4).

layout(matrix(1 :4, 2, 2))
layout.show(4)



layout(matrix(1 :6, 3, 2))
layout.show(6)



2. Les principales fonctions graphiques du package graphics

Liste des principales fonctions graphiques du package graphics
(extrait de R pour les débutants d'Emmanuel Paradis)
Les fonctions qui figurent en gris et italique ne sont pas traitées dans ce document

hist(x)	Histogramme des fréquences de x
barplot(x)	Histogramme des valeurs de x
mosaicplot(x)	Si x est une matrice ou un data.frame, graphe en mosaïque des résidus d'une régression log-linéaire sur une table de contingence
pie(x)	Graphe en « camembert »
boxplot(x)	Graphe « boites à moustaches »
plot(x)	Graphe des valeurs de x (sur l'axe des y) ordonnées sur l'axe des x
plot(x, y)	Graphe bivarié de x (sur l'axe des x) et y (sur l'axe des y)
pairs(x)	Dessine tous les graphes bivariés entre les colonnes de x
symbols(x, y, ...)	Dessine aux coordonnées données par x et y des symboles (cercles, carrés, rectangles, étoiles, thermomètres ou "boxplots") dont les tailles, couleurs, … sont spécifiées par des arguments supplémentaires
sunflowerplot(x, y)	Idem que plot() mais les points superposées sont dessinées sous forme de fleurs dont le nombre de pétales représente le nombre de points

coplot(x~y | z)	Graphe bivarié de x et y pour chaque valeur ou intervalle de valeurs de z
matplot(x,y)	Graphe bivarié de la 1ère colonne de x contre la 1ère de y, la 2ème de x contre la 2ème de y, etc.
stars(x)	Si x est une matrice ou un data.frame, dessine un graphe en segments ou en étoile où chaque ligne de x est représentée par une étoile et les colonnes par les longueurs des branches
assocplot(x)	Graphe de Cohen–Friendly indiquant les déviations de l'hypothèse d'indépendance des lignes et des colonnes dans un tableau de contingence à deux dimensions
fourfoldplot(x)	Visualise, avec des quarts de cercles, l'association entre deux variables dichotomiques pour différentes populations (x doit être un array avec dim = c(2, 2, k) ou une matrice avec dim = c(2, 2) si k = 1)
qqnorm(x)	Quantiles de x en fonction des valeurs attendues selon une loi normale
qqplot(x, y)	Quantiles de y en fonction des quantiles de x
interaction.plot(f1, f2, y)	Dans le cadre d'une analyse de la variance, si f1 et f2 sont des facteurs, graphe des moyennes de y (sur l'axe des y) en fonction des valeurs de f1 (sur l'axe des x) et de f2 (différentes courbes) ; l'option fun permet de choisir la statistique résumée de y (par défaut fun = mean)
contour(x, y, z)	*Courbes de niveau (les données sont interpolées pour tracer les courbes), x et y doivent être des vecteurs et z une matrice telle que dim(z)=c(length(x), length(y)) (x et y peuvent être omis)*
filled.contour(x, y, z)	*Idem mais les aires entre les contours sont colorées, et une légende des couleurs est également dessinée*
image(x, y, z)	*Idem mais en couleur (les données sont tracées)*
persp(x, y, z)	*Idem mais en 3-D (les données sont tracées)*
termplot(mod.obj)	*graphe des effets (partiels) d'un modèle de régression (mod.obj)*

2.1. Histogrammes des fréquences : fonction hist

hist(x) trace l'histogramme des fréquences de x.

Arguments :

hist(	x, breaks = "Sturges", freq = NULL, include.lowest = TRUE, right = TRUE, density = NULL, angle = 45, col = NULL, border = NULL, main = paste("Histogram of" , xname), xlim = range(breaks), ylim = NULL, xlab = xname, ylab, axes = TRUE, plot = TRUE, labels = FALSE,…)
x	vecteur des valeurs dont on souhaite tracer l'histogramme
breaks	il s'agit : - soit d'un vecteur donnant les tranches de l'histogramme - soit le nombre de barres de l'histogramme - soit une chaîne de caractère donnant l'algorithme pour déterminer le nombre de cellules (consulter l'aide en ligne pour plus de détails) - soit une fonction pour déterminer le nombre de cellules.
freq	si 'TRUE' l'histogramme représente les fréquences. Si 'FALSE', l'histogramme représente la densité de probabilité.

include.lowest : si ‘TRUE’, le x[i] égal aux seuils de chaque tranche est inclus dans la première tranche. Si ‘FALSE’, il est inclus dans la deuxième. Cet argument n’est pas pris en compte si breaks n’est pas un vecteur.

right : si ‘TRUE’, les tranches de l’histogramme sont des intervalles fermés à droite, ouverts à gauche.

density= : densité des hachures pour la couleur des tranches. Par défaut =NULL, i.e. aucune hachure n’est tracée. Si on indique une valeur négative, aucune hachure n’est tracée.

angle= : pente des hachures, donnée comme angle en degré (dans le sens inverse des aiguilles d’une montre). Par défaut = 45.

border : couleur de la bordure autour des tranches. Par défaut, la couleur standard du fond du graphique.

axes : si ‘TRUE’ (valeur par défaut), les axes sont dessinés si l’histogramme est tracé.

plot : si ‘TRUE’ (valeur par défaut), l’histogramme est tracé. Sinon, la liste de slimites des tranches et les fréquences de chaque tranche est éditée.

labels : logique ou caractère. Ajoute un label au-dessus de chaque tranche, cf. aide en ligne sur ‘plot.histogram’.

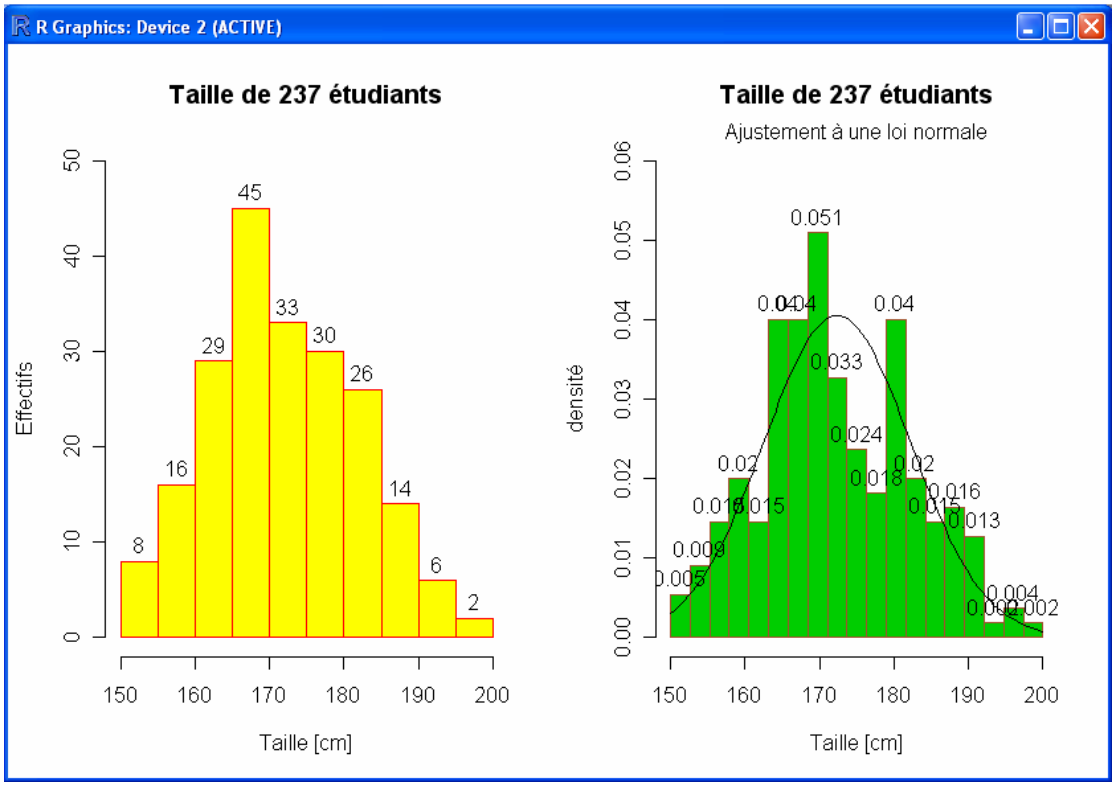
Exemple :

Nous reprenons l’exemple d’Arthur Tenenhaus dans son polycopié de cours. On s’intéresse à la taille de 237 étudiants de DEUG MASS. Les données sont disponibles dans le package MASS que l’on charge préalablement avec la commande library(MASS).

```
hist(survey$Height,col="yellow",border="red",
  main=paste("Taille de",nrow(survey),"étudiants"),
  xlab="Taille [cm]", ylab="Effectifs", ylim=c(0,50),
  labels=TRUE)

hist(survey$Height,breaks=seq(from=150, to=200, length=20),
  col="green3",border="sienna",
  main=paste("Taille de", nrow(survey), "étudiants"),
  xlab="Taille [cm]", ylab="densité",
  proba=TRUE, labels=TRUE, ylim=c(0, 0.06))

x=seq(from=150, to=200, length=100)
lines(x,dnorm(x, mean(survey$Height, na.rm=TRUE),
  sd(survey$Height, na.rm=TRUE),
   )
   )
mtext("Ajustement à une loi normale")
```

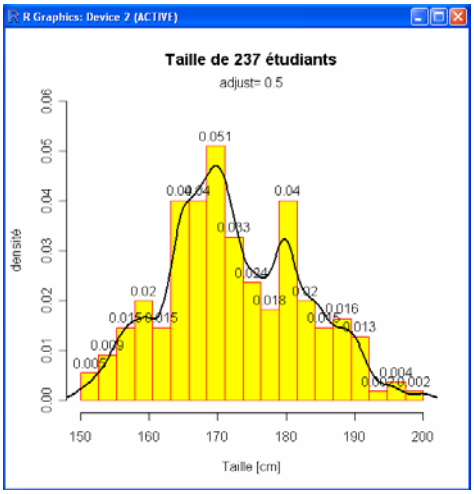


L'option labels=TRUE affiche les fréquences absolues au sommet de chaque barre. Pour le deuxième graphique, on utilise les fréquences relatives (proba=TRUE), ceci facilitant la superposition de distributions de référence (lines()).

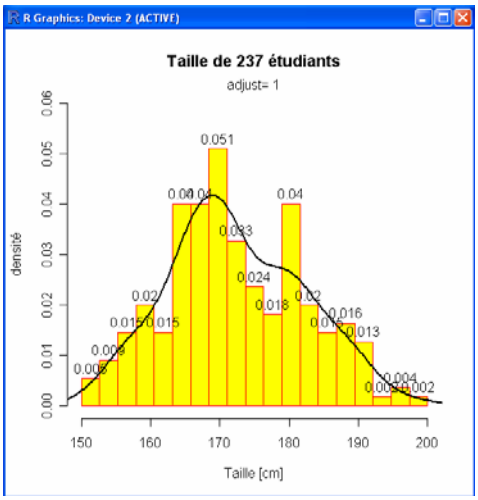
Le problème des histogrammes est que le choix du découpage en intervalles est arbitraire. On peut le contrôler avec le paramètre breaks comme pour le 2ème histogramme.

Le choix du découpage en intervalles est assez délicat et peut biaiser la perception des données. On peut compléter cette approche en utilisant des estimateurs locaux de la densité des points et explorer différentes échelles comme l'illustrent les 4 graphiques suivants.

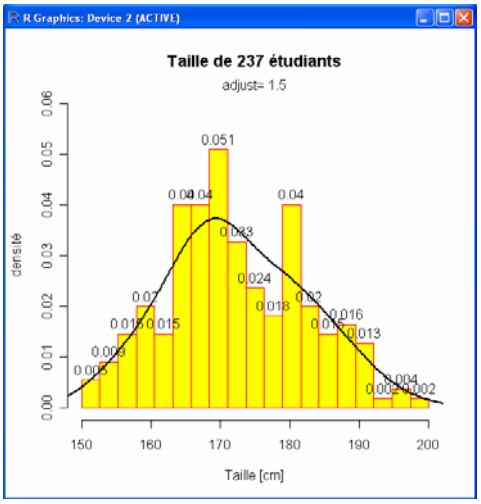
```
adj=0.5
dst=density(survey$Height,na.rm=TRUE,adjust=adj)
hist(survey$Height,
  breaks=seq(from=150, to=200, length=20),
  col="yellow",border="red",
  main=paste("Taille de",nrow(survey),"étudiants"),
  xlab="Taille [cm]", ylab="densité",
  proba=TRUE, labels=TRUE, ylim=c(0, 0.06))
lines(dst$x,dst$y,lwd=2)
mtext(paste("adjust=",adj))
```



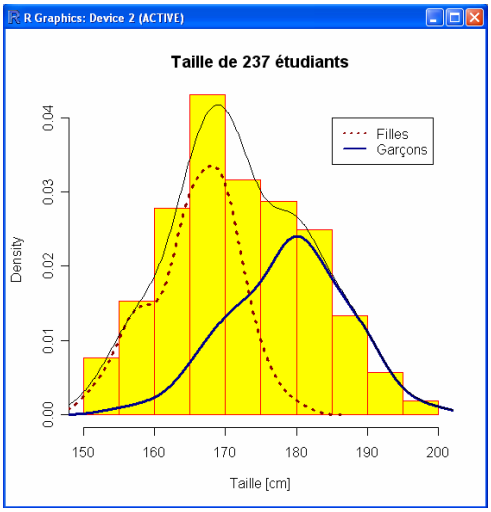
```
adj=1
dst=density(survey$Height,na.rm=TRUE,adjust=adj)
hist(survey$Height,
  breaks=seq(from=150, to=200, length=20),
  col="yellow",border="red",
  main=paste("Taille de",nrow(survey),"étudiants"),
  xlab="Taille [cm]", ylab="densité",
  proba=TRUE, labels=TRUE, ylim=c(0, 0.06))
lines(dst$x,dst$y,lwd=2)
mtext(paste("adjust=",adj))
```




```
adj=1.5
dst=density(survey$Height,na.rm=TRUE,adjust=adj)
hist(survey$Height,
  breaks=seq(from=150, to=200, length=20),
  col="yellow",border="red",
  main=paste("Taille de",nrow(survey),"étudiants"),
  xlab="Taille [cm]", ylab="densité",
  proba=TRUE, labels=TRUE, ylim=c(0, 0.06))
lines(dst$x,dst$y,lwd=2)
mtext(paste("adjust=",adj))
```




```
ng=sum(survey$Sex=="Male", na.rm=TRUE)
nf=sum(survey$Sex=="Female", na.rm=TRUE)
n <- ng + nf
dst=density(survey$Height, na.rm=TRUE)
dstg=density(survey$Height[survey$Sex=="Male"],
na.rm=TRUE)
dstf=density(survey$Height[survey$Sex=="Female"],
na.rm=TRUE)
hist(survey$Height, col="yellow",
border="red",
main=paste("Taille de",nrow(survey),"étudiants"),
xlab="Taille [cm]", proba=TRUE,
ylim=c(0,max(dst$y))
)
lines(dstg$x, ng/n * dstg$y, lwd=3, col="darkblue")
lines(dstf$x, nf/n * dstf$y, lwd=3, lty=3,
col="darkred")
lines(dst$x, dst$y)
legend(185, 0.04, legend=c("Filles", "Garçons"),
col=c("darkred", "darkblue"),
lty=c(3,1), lwd=2, pt.cex=2)
```



Le paramètre important de la fonction density() est le paramètre adjust. La valeur par défaut du paramètre adjust est 1. Si adjust est inférieur à 1, on s'intéresse aux petites variations et à la nature discrète de la variable. Si adjust est supérieur à 1, on s'intéresse au comportement global de la variable et aux variations importantes.

Un autre avantage des estimateurs locaux de densité est qu'ils permettent de superposer facilement plusieurs distributions. Ainsi, on a pu distinguer la distribution des filles de celles des garçons.

2.2. Histogramme des valeurs de x : fonction barplot(x)

barplot(x) trace l'histogramme des valeurs de x, où x est une variable qualitative (un facteur d'une data.frame).

Arguments :

barplot(height, width = 1, space = NULL, names.arg = NULL, legend.text = NULL, beside = FALSE, horiz = FALSE, density = NULL, angle = 45, col = NULL, border = par("fg"), main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, xpd = TRUE, log = "", axisnames = TRUE, axes = TRUE, cex.axis = par("cex.axis"), cex.names = par("cex.axis"), inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0, add = FALSE, ...)

height	vecteur ou matrice de valeurs décrivant les barres qui font l'objet d'un tracé.
width	vecteur donnant la largeur des barres. Recyclé si le nombre de barres dessinées est supérieur à la longueur du vecteur.
space	espace avant chaque barre (comme pourcentage de la hauteur moyenne des barres). On peut spécifier un seul nombre ou un nombre par barre.
names.arg	vecteur de noms inscrits sous les barres.
legend.text	vecteur du texte de légende ou argument logique indiquant si une légende doit être ajoutée. Cet argument ne sert que lorsque height est une matrice. Dans ce cas, les labels de la légende doivent correspondre aux lignes de height. SI legend.text=TRUE, les noms des colonnes de height sont utilisés comme label s'ils sont non nuls.
beside	argument logique. Si FALSE, les colonnes de height sont représentées par des barres empilées. Si TRUE, les barres sont juxtaposées.
horiz	argument logique. Si FALSE, les barres sont dessinées verticalement. Si TRUE, elles sont représentées horizontalement.
density=	densité des hachures pour la couleur des tranches. Par défaut =NULL, i.e. aucune hachure n'est tracée. Si on indique une valeur négative, aucune hachure n'est tracée.
angle=	pente des hachures, donnée comme angle en degré (dans le sens inverse des aiguilles d'une montre). Par défaut = 45.
col	vecteur des couleurs des barres.
border	couleur des bordures des barres.
xpd	argument logique. Si TRUE, les barres peuvent dépasser les limites fixées pour les axes.
log	chaîne de caractères spécifiant si les échelles des axes doivent être logarithmique.
axes	argument logique. Si TRUE, un axe vertical (horizontal si horiz=TRUE) est tracé.
axisnames	argument logique. Si TRUE et si on a précisé l'option names.arg, un autre axe est tracé (avec lty=0) et nommé.
inside	argument logique. Si TRUE, les lignes qui divisent des barres adjacentes sont tracées. S'applique uniquement lorsque space=0 (ce qui est en partie le cas lorsque beside=TRUE).
axis.lty	paramètres graphiques 'lty' (cf. § 4) appliqués à l'axe et aux graduations de l'axe.

offset vecteur indiquant de combien les barres doivent dépasser de part et d'autre de l'axe des x.

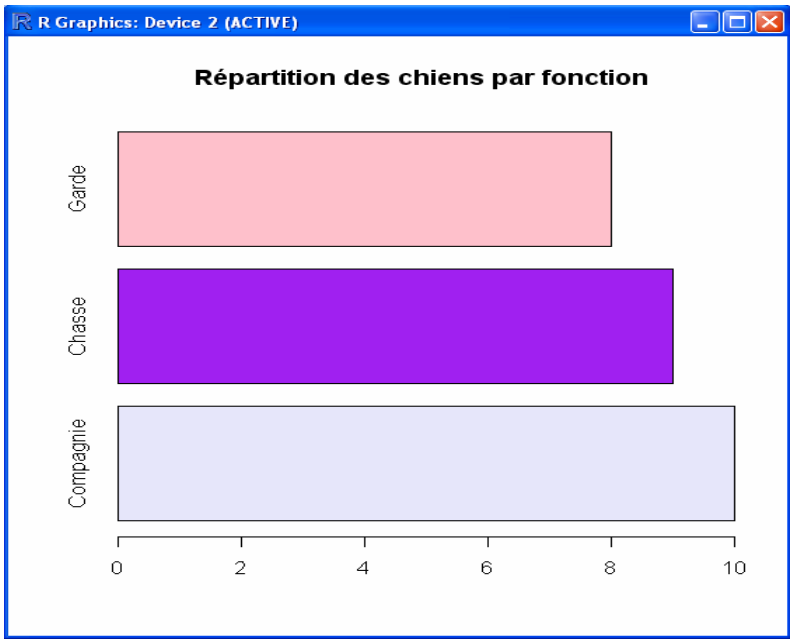
Pour les autres arguments, cf. § 4.

Exemple : *répartition des chiens par fonction*

```
X=table(chien$Fonction)

X
 1  2  3
10  9  8

barplot(height=X,names.arg=c("Compagnie","Chasse","Garde"),horiz=TRUE,
col=c("lavender","purple","pink"),main="Répartition des chiens par fonction")
```



2.3. Diagrammes en mosaïque : mosaicplot(x)

mosaicplot(x) permet de tracer le diagramme en mosaïque d'une variable nominale. On peut aussi utiliser cette fonction pour tracer la graphe en mosaïque des résidus d'une régression log-linéaire sur une table de contingence.

Arguments :

```
mosaicplot( x, main = deparse(substitute(x)), sub = NULL, xlab = NULL, ylab = NULL,
sort = NULL, off = NULL, dir = NULL, color = NULL, margin = NULL,
cex.axis = 0.66, las = par("las"), type = c("pearson", "deviance", "FT"), ...)
```

x variable nominale ou table de contingence

sort vecteur indiquant la manière dont sont ordonnées les données.

off vecteur indiquant l'espace entre chaque niveau de la mosaïque (indiqué en pourcentage, les valeurs doivent être idéalement comprises entre 0 et 20, le maximum est de 50).

dir vecteur des directions de séparations pour chaque dimension de la table de contingence ("v" pour vertical et 'h" pour horizontal). Par défaut, les directions sont alternées, en commençant par une direction verticale.

color argument logique ou vecteur de couleur. Par défaut, le graphique est dessiné en gris. 'color'=TRUE dessine la mosaïque en gris. color= 'FALSE' trace les contours des barres du diagramme mais ne colorie pas l'intérieur.

margin une liste de vecteurs pour le totaux des marges utilisés pour ajuster le modèle log-linéaire . Par défaut, on ajuste un modèle d'indépendance. Argument utilisé seulement si on trace le graphe en mosaïque des résidus d'une régression log-linéaire sur une table de contingence.

type chaîne de caractères indiquant le type de résidus réprésenté. Utilisé seulement si on trace le graphe en mosaïque des résidus d'une régression log-linéaire sur une table de contingence. Les valeurs possibles sont "pearson" (donne les composantes du Chi-2 de pearson), ''deviance'' (composantes du ratio de vraisemblance du Chi-2) ou ''FT'' pour les résidus de Freeman-Tukey.

Pour les autres arguments, cf. § 4.

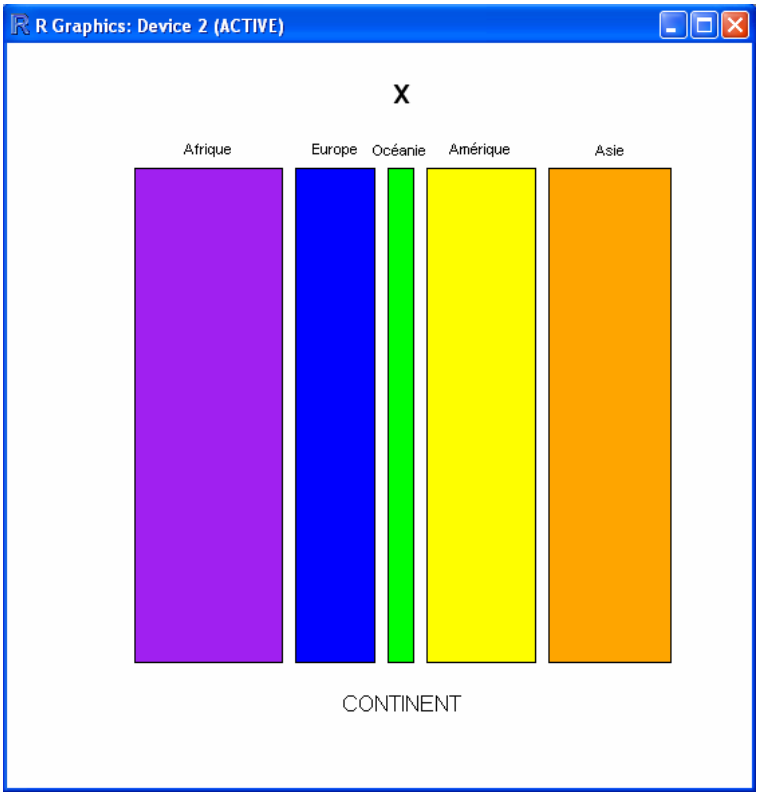
Exemples :

Les commandes suivantes permettent d'obtenir le diagramme en mosaïque de la variable continent du fichier paysniv3 :

```
paysniv3$CONTINENT=factor("Afrique",levels=c("Afrique", "Europe", "Océanie", "Amérique", "Asie"))
paysniv3$CONTINENT[paysniv3$CONTI==2]="Asie"
paysniv3$CONTINENT[paysniv3$CONTI==3]="Amérique"
paysniv3$CONTINENT[paysniv3$CONTI==4]="Europe"
paysniv3$CONTINENT[paysniv3$CONTI==5]="Océanie"

X=table(paysniv3$CONTINENT, dnn="CONTINENT")
CONTINENT
 Afrique  Europe  Océanie  Amérique  Asie
 53       28      9        39        44

mosaicplot(X,color=c("purple","blue","green","yellow","orange"))
```



Les commandes suivantes permettent de visualiser sous la forme d'un diagramme en mosaïque le croisement des variables origine et finition du fichier voitures.

On recode d'abord les variables Origine et Finition pour regrouper les modalités rares :

```
voitures$Origin2=factor("Italie",levels=c("Italie", "Allemagne", "France", "Japon", "Autres"))
```

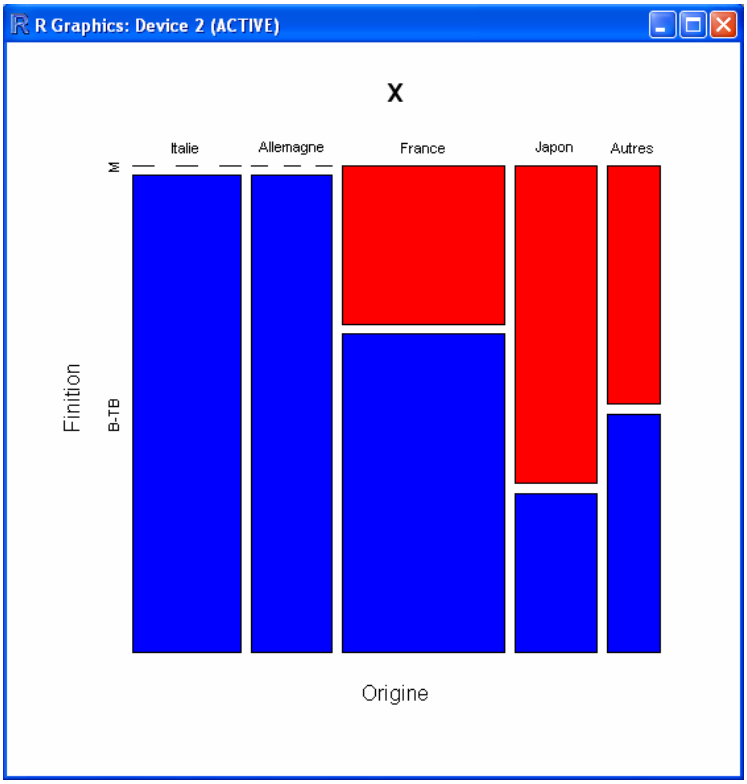
```
voitures$Origin2[voitures$Origine=="D"]="Allemagne"
voitures$Origin2[voitures$Origine=="F"]="France"
voitures$Origin2[voitures$Origine=="J"]="Japon"
voitures$Origin2[voitures$Origine=="GB"]="Autres"
voitures$Origin2[voitures$Origine=="U"]="Autres"
voitures$Finition2=factor(voitures$Finition,levels=c("M", "B-TB"))
voitures$Finition2[voitures$Finition=="B"]="B-TB"
voitures$Finition2[voitures$Finition=="TB"]="B-TB"
```

On crée ensuite la table de contingence des variables Origin2 et Finition2 :

```
X=table(voitures$Origin2,voitures$Finition2,dnn=c("Origine","Finition"))
```

Origine	Finition M	B-TB
Italie	0	4
Allemagne	0	3
France	2	4
Japon	2	1
Autres	1	1

```
mosaicplot(X,color=c("red","blue"))
```



A partir de 3 variables, le graphique obtenu n'est plus très lisible. Ainsi, dans l'exemple ci-dessous, on croise la taille, le poids et la fonction des chiens. On recode préalablement les variables afin d'obtenir un intitulé explicite des modalités :

```
chien$Taille2=factor("Gros",levels=c("Petit","Moyen","Gros"))
chien$Taille2[chien$Taille=="1"]="Petit"
chien$Taille2[chien$Taille=="2"]="Moyen"

chien$Poids2=factor("Lourd",levels=c("Lourd","Moyen","Léger"))
chien$Poids2[chien$Poids=="1"]="Léger"
chien$Poids2[chien$Poids=="2"]="Moyen"

chien$Fonction2=factor("Compagnie",levels=c("Compagnie","Chasse","Garde"))
chien$Fonction2[chien$Fonction=="2"]="Chasse"
chien$Fonction2[chien$Fonction=="3"]="Garde"
```

On construit la table de contingence :

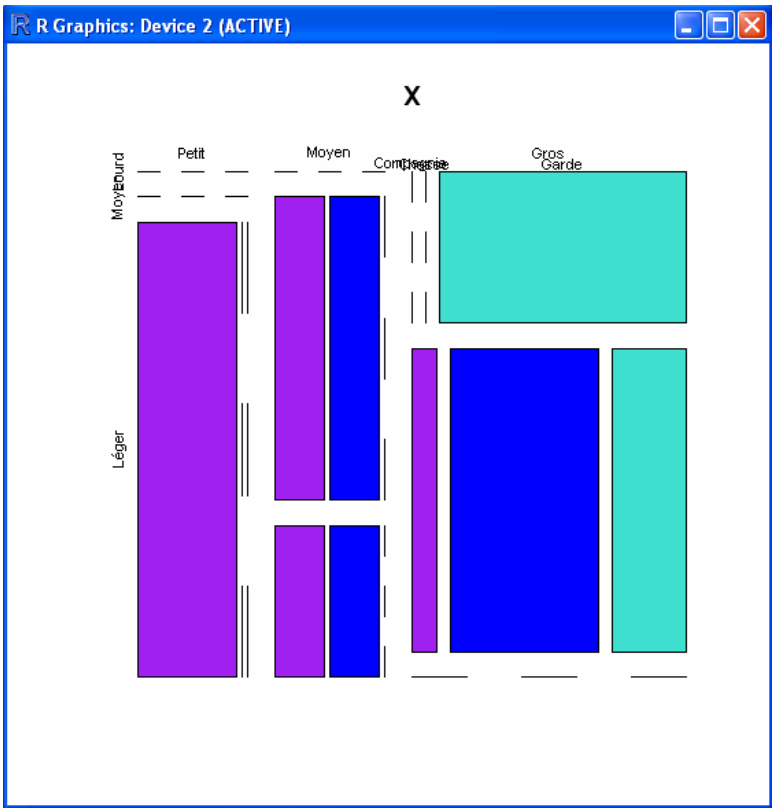
```
X=table(chien$Taille2,chien$Poids2,chien$Fonction2)
X
, ,  = Compagnie
        Lourd  Moyen  Léger
 Petit      0      0      6
 Moyen      0      2      1
 Gros       0      1      0

, ,  = Chasse
        Lourd  Moyen  Léger
 Petit      0      0      0
 Moyen      0      2      1
 Gros       0      6      0

, ,  = Garde
        Lourd  Moyen  Léger
 Petit      0      0      0
 Moyen      0      0      0
 Gros       5      3      0
```

```
mosaicplot(X,color=c("purple","blue","turquoise"))
```

Les chiens de compagnie sont représentés en violet, les chiens de chasse en bleu et les chiens de garde en turquoise.



2.4. Diagrammes en camembert : fonction pie

pie(x) permet d'obtenir un diagramme en camembert.

Arguments :

pie(x, …)

x vecteur de quantités positives

Arguments optionnels :

labels=	vecteur de chaînes de caractères correspondant aux noms des tranches du camembert
edges=	le contour du camembert est approché par un polygone dont le nombre de sommets est indiqués par edges (par défaut 200)
radius=	le camembert est dessiné dans un carré dont les côtés s'étendent des coordonnées -1 à +1. Si les labels sont longs, il peut être nécessaire de préciser une plus petite valeur pour radius (0.8 par défaut).
clockwise=	indique si les tranches sont dessinées dans le sens des aiguilles d'une montre (=TRUE) ou en sens inverse des aiguilles d'une montre (=FALSE, valeur par défaut).
init.angle=	angle de départ. Par défaut =0 (i.e. 3 heures), sauf si clockwise=TRUE, auquel cas =90 (i.e. 12 heures).
density=	densité des hachures pour la couleur des tranches. Par défaut =NULL, i.e. aucune hachure n'est tracée. Si on indique une valeur négative, aucune hachure n'est tracée.
angle=	pente des hachures, donnée comme angle en degré (dans le sens inverse des aiguilles d'une montre). Par défaut = 45.
col=	vecteur des couleurs utilisées pour les tranches. Si ce vecteur n'est pas indiqué, un ensemble de couleurs par défaut est utilisé sauf si on a précisé l'option density.
border, lty	arguments appliqués aux polygones qui tracent les contours des tranches
main=,…	paramètres graphiques courants

Exemples

Nous reprenons l'exemple du polycopié de cours d'Arthur Tenenhaus « ventes de tartes ».

```
vente.tarte=c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)

pie(vente.tarte,
labels=c("Cerise","Framboise","Pomme","Abricot","Pêche","Fraise"),
col=c("red2","magenta1","yellowgreen","orange1","orange2","magenta3"),
border=NA, main="Ventes de tartes")
```

2.5. Boîtes à moustaches : fonction boxplot

boxplot(x) trace la boîte à moustaches de la variable x. La boîte à moustaches est une invention de TUKEY (1977) pour représenter schématiquement une distribution. Sur la boîte à moustaches sont schématisés :

- le 1er quartile Q1, partie inférieure de la boîte,
- le 2ème quartile Q2, ou médiane,
- le 3ème quartile Q3, partie inférieure de la boîte,
- les valeurs adjacentes « 2 moustaches » représentées par des rectangles verticaux de part et d'autre de la boîte. Ces valeurs sont situées à une distance maximum, fonction de l'écart inter-quartile (Q3-Q1).
  (Q3-Q1) correspond à la hauteur de la boîte.

Arguments :

boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE, notch = FALSE, outline = TRUE, names, plot = TRUE, border = par("fg"), col = NULL, log = "", pars = list(boxwex = 0.8, staplewex = 0.5), horizontal = FALSE, add = FALSE, at = NULL)

x	vecteur numérique ou liste contenant des vecteurs numériques.
range	longueur des moustaches, par défaut *1,5*. La valeur maximale de la moustache supérieure est à la distance : Q3+*1,5**(Q3-Q1). La valeur minimale de la moustache inférieure est à la distance : Q1-*1.5**(Q3-Q1). range est donc la ligne joignant les quartiles Q1 et Q3 à l'observation la plus éloignée dans la limite de *1.5* fois l'écart interquartile (Q1-Q3).
width	un vecteur donnant la largeur relative des boîtes.
varwidth	si TRUE, les boîtes ont une largeur proportionnelle au nombre d'observations du groupe.
notch	si notch=TRUE, une encoche est dessinée de chaque côté de la boîte au niveau de la médiane. Si les encoches ne se recoupent pas, il y a une probabilité élevée que les deux médianes ne soient pas égales.
outline	si FALSE, les points aberrants ne sont pas représentés.
names	label des groupes qui figure sous chaque boîte.
boxwex	un facteur d'échelle appliqué à chaque boîte. Lorsqu'il y seulement quelques groupes, on peut améliorer la présentation en rétrécissant les boîtes en largeur.
staplewex	largeur de la ligne tracée aux niveaux des valeurs minimale et maximale de la moustache.
plot	si FALSE, on obtient uniquement un résumé des données à partir desquelles la boîte est tracée.
border	vecteur de couleurs pour les contours de la boîte.
log	variable caractère indiquant si x et y doivent être représentés sur des échelles logarithmiques.
horizontal	si TRUE, les boîtes sont tracées à l'horizontale.
add	vecteur numérique indiquant l'endroit où les boîtes à moustaches doivent être tracées., en particulier lorsque add=TRUE. Par défaut, '1 :n', où n est le nombre de boîtes à moustaches à tracer.

Exemple :

Nous reprenons l'exemple d'Arthur Tenenhaus sur la taille des étudiants de DEUG MASS. Le fichier utilisé est le même que celui traité en exemple pour les histogrammes construits avec la fonction hist.

On sépare la fenêtre en deux sous-fenêtres horizontales :

```
layout(matrix(2:1,2,1))
layout.show(2)
```

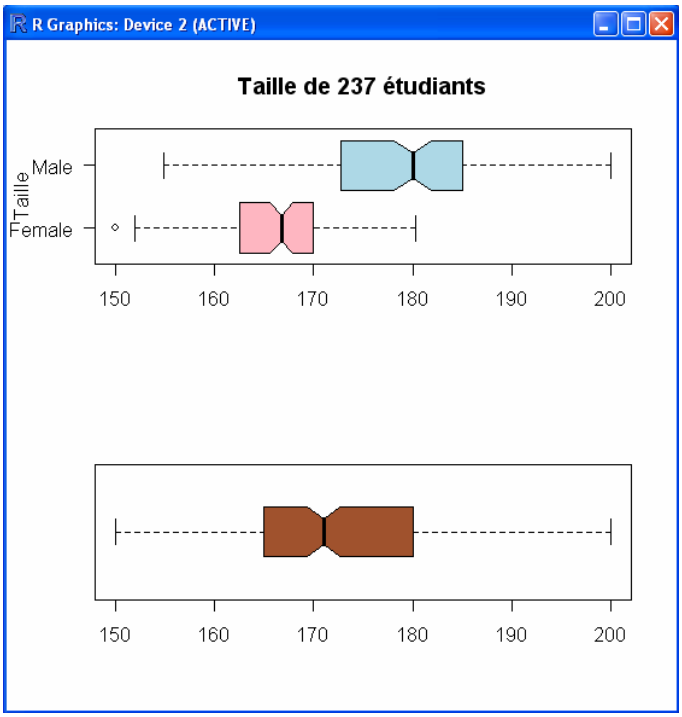
On trace d'abord la boîte à moustaches pour l'ensemble des étudiants :

```
boxplot(survey$Height,
col="sienna",
horizontal=TRUE,
notch=TRUE,
las=1)
```

On la trace ensuite la boîte à moustaches en fonction du sexe.

```
boxplot(survey$Height~survey$Sex,
col=c("lightpink","lightblue"),
horizontal=TRUE,
notch=TRUE,
main=paste("Taille de",nrow(survey),"étudiants"),
ylab="Taille",
las=1)
```

Pour toutes les boîtes, on trace une encoche autour de la médiane (option notch=TRUE).



2.6. Diagrammes de dispersion : fonction plot

plot(x) permet d'obtenir le graphe des valeurs de x (sur l'axe des ordonnées) selon l'ordre des observations (sur l'axe des abscisses).

plot(x,y) permet d'obtenir le diagramme de dispersion de y en fonction de x.

Arguments :

plot(x, y, ...)

x coordonnées des points (sur l'axe des ordonnées pour plot(x) ou sur l'axe des abscisses pour plot(x,y))

y coordonnées des points sur l'axe des ordonnées (y est facultatif)

types type de graphe

main titre principal, qui doit être une chaîne de caractères

sub sous-titre

xlab annotation de l'axe des abscisses, qui doit être une chaîne de caractère

ylab annotation de l'axe des ordonnéess, qui doit être une chaîne de caractère

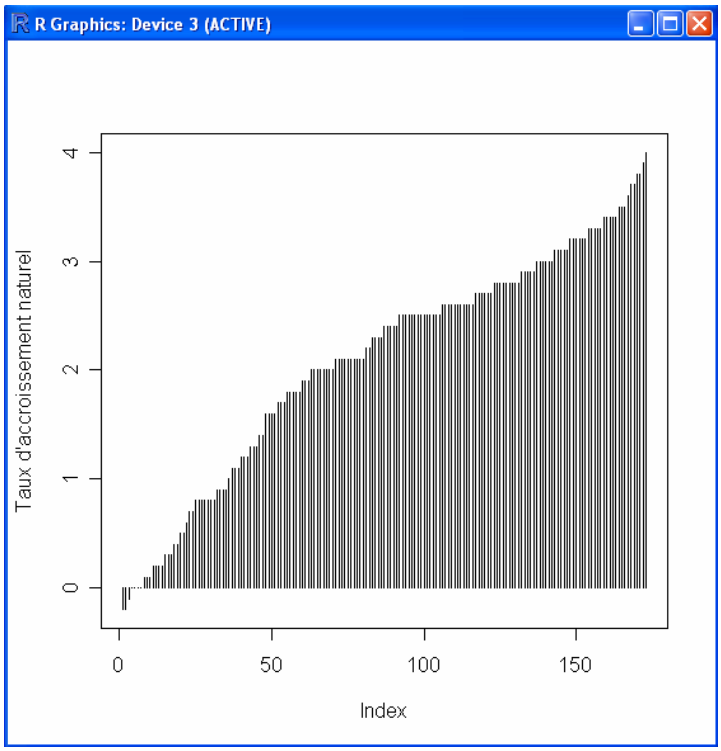
Exemples :

On veut visualiser la distribution des taux d'accroissement naturels. Une manière de procéder est d'utiliser la fonction plot. On crée d'abord le vecteur x, qui correspond aux taux d'accroissement naturels des 173 pays du fichier paysniv3, classés par ordre croissant :

```
x=paysniv3[order(paysniv3[,5]),][,5]
```

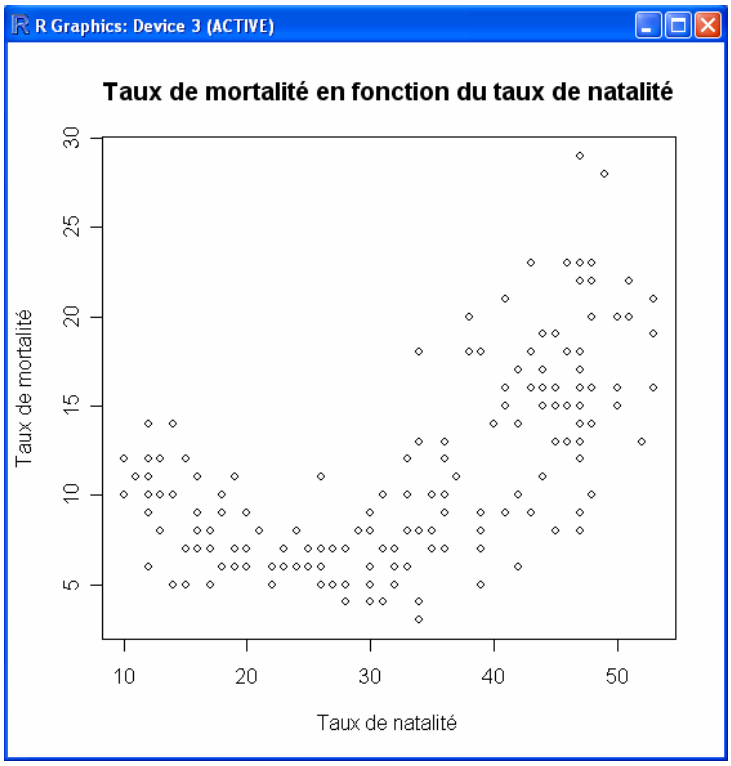
On trace ensuite des barres verticales avec l'option type="h" :

```
plot(x,type="h",ylab="Taux d'accroissement naturel")
```



Les commandes suivantes tracent le diagramme de dispersion de la variable MORT en fonction de la variable NAT :

```
plot(paysniv3$NAT,paysniv3$MORT,type="p",
xlab="Taux de natalité",ylab="Taux de mortalité",
main="Taux de mortalité en fonction du taux de natalité")
```



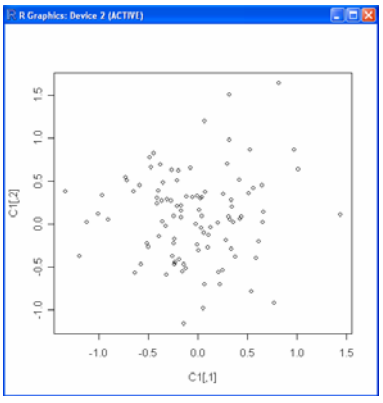
Le graphique précédent peut également être obtenu à partir des commandes suivantes :

```
X=paysniv3[3:4]
ls.str(pattern="X")
X : `data.frame':       173 obs. of  2 variables:
 $ NAT : num  42 37 39 36 45 32 51 48 35 46 ...
 $ MORT: num  10 11 9 10 16 7 20 20 8 15 ...
plot(X,type="p",
+ xlab="Taux de natalité",ylab="Taux de mortalité",
+ main="Taux de mortalité en fonction du taux de natalité")
```

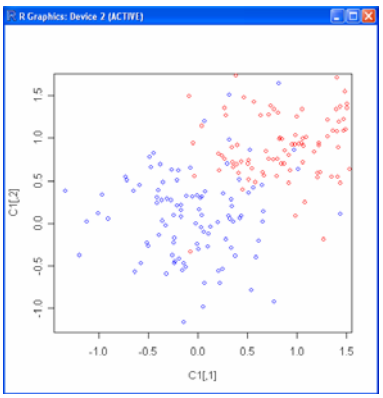
On reprend ci-dessous l'exemple du polycopié de cours d'Arthur Tenenhaus sur la fonction plot.

```
C1=matrix(rnorm(200,sd=0.5),ncol=2)
C2=matrix(rnorm(200,mean=1,sd=0.5),ncol=2)
mat=rbind(C1,C2)
```

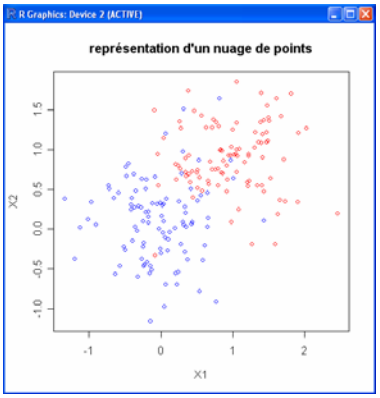
```
plot(C1)
```



```
plot(C1,col="blue")
points(C2,col="red")
```

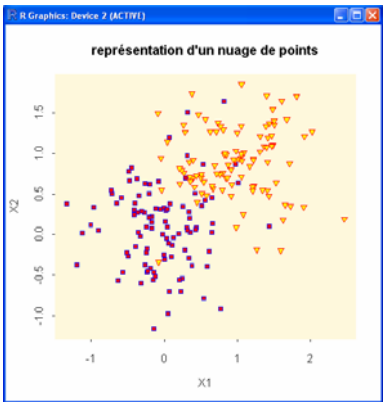



```
plot(C1, col="blue",
xlim=range(mat[,1]),
ylim=range(mat[,2]),
main="représentation d'un nuage de points",
xlab="X1", ylab="X2"
)
points(C2, col="red")
```




```
plot(1,
xlim=range(mat[,1]),
ylim=range(mat[,2]),
main="représentation d'un nuage de points",
xlab="X1", ylab="X2",
bty="l", tcl= -.25
)
rect(-3,-3,3,3,col="cornsilk")

points(C1, col="blue", pch=22, bg="red")
points(C2, col="red", pch=25, bg="yellow")
```



Le 1er graphique présente le graphe de base de la fonction plot(). C1 est une matrice composée de deux vecteurs. La fonction plot() trace le graphe bivarié de la première colonne de C1 sur la deuxième colonne de C1. On aurait également pu taper la commande suivante : plot(C1[,1], C1[,2])

On peut spécifier la couleur des points grâce à l'option col. La liste des couleurs disponibles (657 couleurs) est fournit par la fonction colors(). Le 2ème graphique présente, en plus de l'utilisation de l'option col, l'utilisation de la fonction points(). Cette fonction permet de rajouter des points à un graphique existant. Un problème subsiste : La longueur des axes ne s'adapte pas au nuage de points complet, mais seulement au nuage de points du premier plot.

Le 3ème graphique remédie à ce problème grâce aux options xlim et ylim. Notons que la fonction range()retourne le minimum et le maximum des arguments rentrés en paramètre. On peut par ailleurs donner des noms aux axes grâce aux options xlab et ylab et un titre au graphique grâce à l'option main.

Pour le 4ème graphique, on définit dans un premier temps l'architecture du graphique. L'option bty contrôle la forme du cadre tandis que l'option tcl spécifie la longueur des graduations. La fonction rect() trace un rectangle délimité par les quatre premiers arguments rentrés en paramètres. Une fois l'architecture du graphique achevé, on place les points via la fonction points(). Notons l'utilisation des options pch et bg qui spécifient respectivement la forme et la couleur de fond des points.

2.7. Matrices de dispersion : fonctions plot et pairs

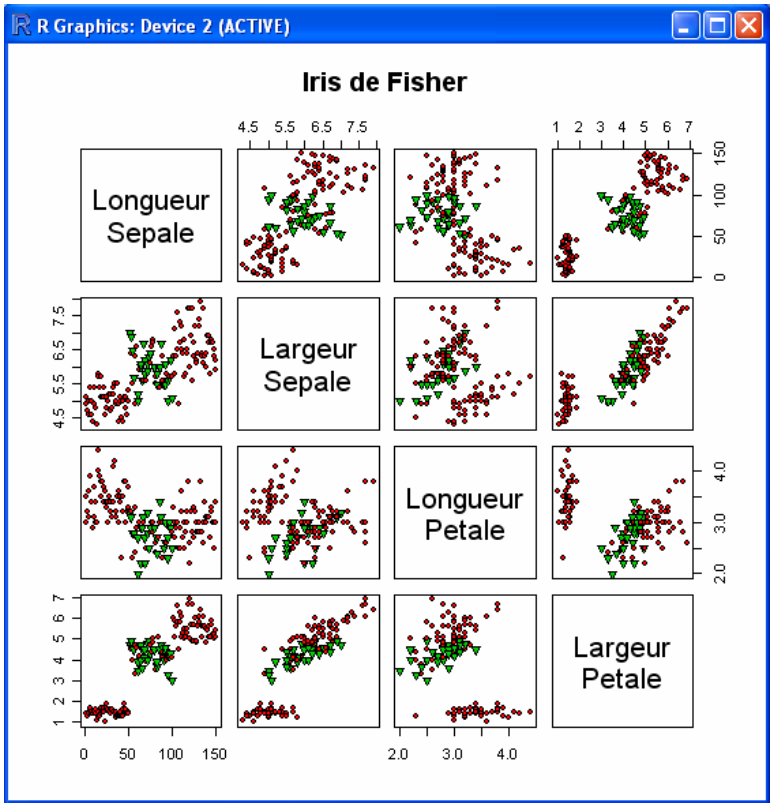
Nous avons déjà examiné la fonction plot.

Pour obtenir une matrice de dispersion, on utilisera plot(X) ou pairs(X) où X est une matrice ou une data.frame et on obtiendra alors tous les graphes bivariés entre les colonnes de X.

Exemples :

Nous reprenons l'exemple du polycopié de cours d'Arthur Tenenhaus sur les iris de Fisher, que nous allons représenter en utilisant les fonctions plot ou pairs.

```
plot(iris[1:4],        (ou pairs(iris[1:4],)
bg=c("red","green3","blue")[iris[,5]],
pch=c(21,25,24)[iris[,5]],
main="Iris de Fisher",
labels=
c("Longueur\nSepale",
"Largeur\nSepale",
"Longueur\nPetale",
"Largeur\nPetale") )
```



Le graphique précédent peut également être obtenu à partir des commandes suivantes :

```
pairs(iris[1:4],
bg=c("red","green3","blue")[iris[,5]],
pch=c(21,25,24)[iris[,5]],
main="Iris de Fisher",
labels=
c("Longueur\nSepale",
"Largeur\nSepale",
"Longueur\nPetale",
"Largeur\nPetale") )
```

2.8. Diagrammes de dispersion avec représentation des points en fonction d'une 3ème variable : fonction symbols

symbols(x,y,…) dessine aux coordonnées données par x et y des symboles (cercles, carrés, rectangle, étoiles, thermomètres, boxplots) dont les tailles, couleurs etc. sont spécifiées par des arguments supplémentaires.

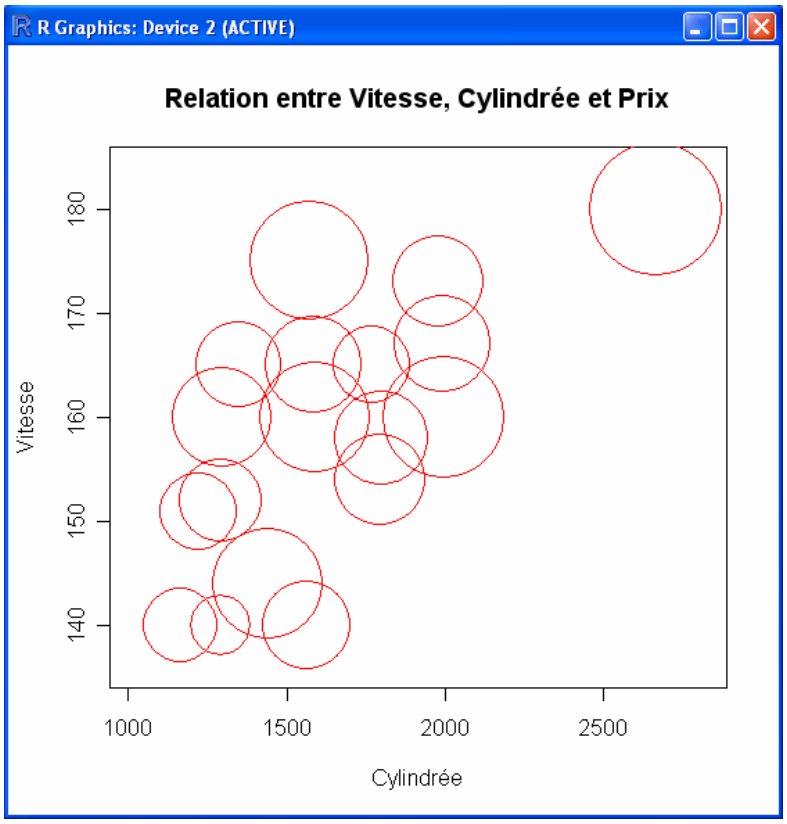
Arguments :

symbols(	x, y = NULL, circles, squares, rectangles, stars, thermometers, boxplots, inches = TRUE, add = FALSE, fg = par("col"), bg = NA, xlab = NULL, ylab = NULL, main = NULL, xlim = NULL, ylim = NULL, ...)
x, y	coordonnées des points
circles	un vecteur donnant le rayon des circles
squares	vecteur donnant la longueur des côtés de cercles
rectangles	matrice avec deux colonnes. La 1ère donne la largeur et la deuxième la hauteur des rectangles
stars	matrice avec trois colonnes ou plus donnant la longueur des rayons depuis le centre des étoiles. Les valeurs manquantes 'NA' sont remplacées par des 0. (cf. également fonction stars).
thermometers	matrice de 3 ou 4 colonnes. Les deux premières donnent la largeur et la hauteur des symboles. S'il y a 3 colonnes, la 3ème est prise comme proportion : les thermomètres sont remplis de leur base jusqu'à cette proportion de leur hauteur. S'il y a 4 colonnes, les 3ème et 4ème colonne sont prises comme proportion : les thermomètres sont remplis entre ces eux proportions de leur hauteur.
boxplots	une matrice avec 5 colonnes. Les deux premières donnent la largeur et la hauteur des boîtes à moustaches, les deux colonnes suivantes donnent la longueur de l'écart inter-quartile et la 5ème la proportion de la boîte où la ligne médiane est tracée.
inches	si 'FALSE', les unités utilisées sont celles de l'axe des abscisses. Si 'TRUE', la hauteur des symboles est ajustée de sorte que le plus grand fasse 1 pouce (1 inch). Si un chiffre est précisé, la hauteur du symbole le plus grand fait cette hauteur en pouces.
fg	couleur utilisée pour le contour des symboles
bg	couleur utilisée pour colorier l'intérieur des symboles. Si on ne précise pas cette option, les symboles ne sont pas coloriés.
…	autres paramètres graphiques courants

Exemple :

On veut visualiser la relation entre la cylindrée d'une voiture, sa puissance et son prix. On trace le diagramme de dispersion des variables Cylindrée et Vitesse et on utilise un symbole proprotionnel au prix.

```
symbols(voitures$Cylindrée,voitures$Vitesse,circles=voitures$Prix,
inches=0.5,
fg="red",
main='Relation entre Vitesse, Cylindrée et Prix',
xlab='Cylindrée',ylab='Vitesse')
```



2.9. *Diagrammes de dispersion en « tournesol » : fonction sunflowerplot*

sunflowerplot(x,y) permet comme plot(x,y) d'obtenir des diagrammes de dispersion, mais les points superposés sont dessinés sous forme de fleurs dont le nombre de pétales représente le nombre de points.

Arguments :

sunflowerplot(x, y, ...)

x	coordonnées des points sur l'axe des abscisses. Vecteur de longueur n (nombre d'observations).
y	coordonnées des points sur l'axe des ordonnées. Vecteur de longueur n.

Arguments optionnels :

number	vecteur de longueur n. number[i] donne le nombre de fois où le point de coordonnées (x[i],y[i]) doit être représenté. number[i] peut être égal à 0. Si number n'est pas précisé, chaque point sera représenté autant de fois qu'il existe dans la base de données.
log =	pour obtenir une échelle logarithmique
digits =	nombre de chiffres significatifs conservés pour les données avant de représenter chaque point (par défaut 6)
rotate =	si TRUE effectue une rotation aléatoire des tournesols (par défaut FALSE)
size =	taille des pétales (par défaut 1/8)
seg.col =	couleur des segments qui forment les pétales (par défaut 2)
seg.lwd =	largeur des segments qui forment les pétales (par défaut 1.5)

(cf. § 4) :

xlab = , ylab =	annotation des axes (par défaut NULL)
xlim = , ylim =	limites inférieures et supérieures des axes (par défaut NULL)
add =	si TRUE superpose le graphe au graphe existent (s'il y en a un) (par défaut FALSE)
pch =	type de symbole (par défaut 16, soit des diamants)

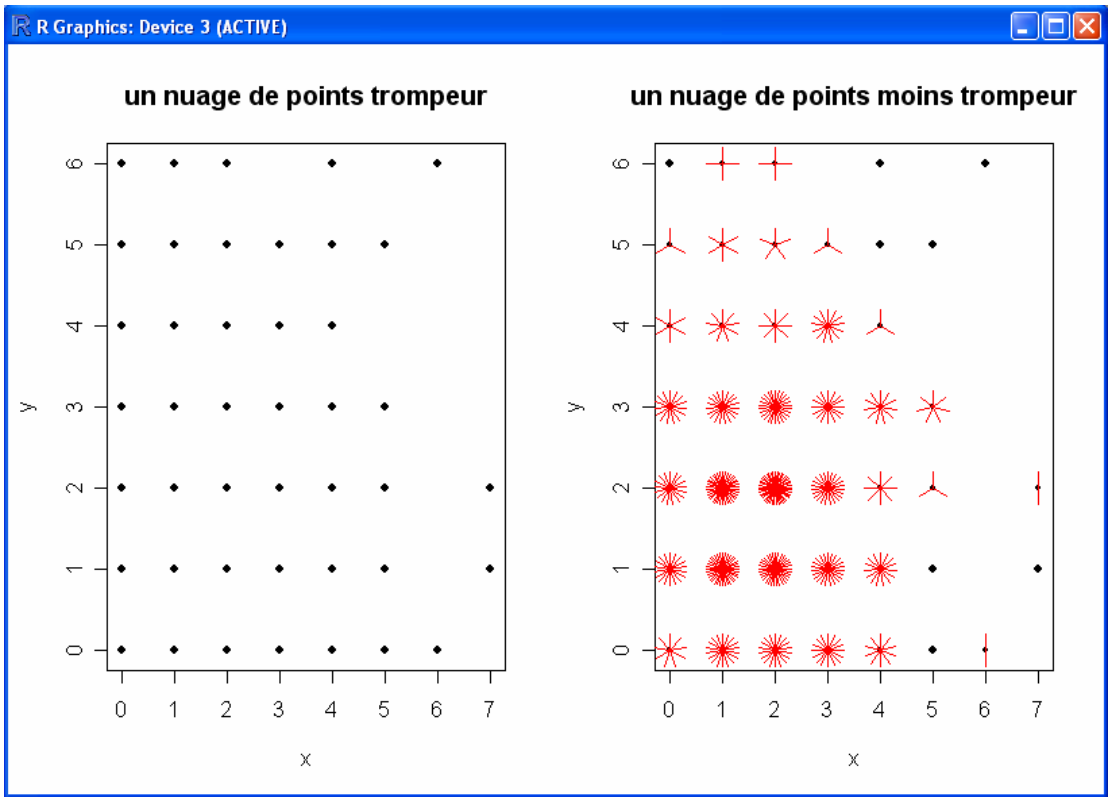
cex =	taille des caractères et des symboles (par défaut 0.8)
col =	couleur des symboles
bg =	couleur de l'arrière plan (par défaut NA)

Exemple

Nous reprenons l'exemple du polycopié de cours d'Arthur Tenenhaus « un exemple fleuri ».

A l'aide de la fonction rpois(), on génère pour x et y 500 données aléatoires suivant une loi de Poisson. Puis on trace le diagramme de dispersion de y en fonction de x, avec les fonctions plot et sunflowerplot. Avec la fonction sunflowerplot, le nombre de pétales indique le nombre de points superposés.

```
n=500
x=rpois(n,lambda=2)
y=rpois(n,lambda=2)
plot(x,y,pch=19,main="un nuage de points trompeur")
sunflowerplot(x,y,pch=19,main="un nuage de points moins trompeur")
```



2.10. Diagrammes de dispersion conditionnelle : fonction coplot

coplot(x~y|z) fournit le diagramme de dispersion de x et y pour chaque valeur ou intervalle de valeurs de z.

Arguments :

```
coplot(     formula, data, given.values, panel = points, rows, columns, show.given = TRUE,
            col = par("fg"), pch = par("pch"), bar.bg = c(num = gray(0.8), fac = gray(0.95)),
            xlab = c(x.name, paste("Given :", a.name)),
            ylab = c(y.name, paste("Given :", b.name)),
            subscripts = FALSE, axlabels = function(f) abbreviate(levels(f)), number = 6,
            overlap = 0.5, xlim, ylim, ...,
            co.intervals(x, number = 6, overlap = 0.5)
```

formula	formule décrivant la relation conditionnelle. La formule y~x|z indique que le diagramme de dispersion doit être tracé conditionnellement aux valeurs de la variable z. La formule y~x|a*b trace le diagramme de dispersion de x et y en fonction des valeurs des deux variables a et b. Les variables doivent être numériques ou facteurs.
data	data.frame contenant les variables indiquée dans la formule. On peut également indiquer directement dans la formule la data.frame dans laquelle se trouvent les variables analysées. Par exemple : paysniv3$NAT ~ paysniv3$MORT | paysniv3$CONT permet d'obtenir le diagramme de dispersion des taux de natalité et de mortalité par continent et il n'est pas nécessaire de repréciser « paysniv3 » après la formule précédente.
given.values	une valeur ou une liste de deux valeurs qui détermine le conditionnement par rapport à a et b.
panel	une fonction(x,y,col,pch…) qui précise les paramètres appliqués pour chaque sous-fenêtre du graphique. Par défaut, on obtient des points.
rows	les sous-fenêtres sont organisées dans un tableau comportant des lignes (rows) et des colonnes (columns). Avec l'option rows, on précise combien de lignes on souhaite.
columns	nombre de colonnes du tableau. Par exemple, si on doit tracer 6 graphiques (6 valeurs pour les variables conditionnelles), on peut choisir de les répartir de plusieurs manières : - rows=2, columns=3 on a 3 graphiques par ligne sur 2 lignes - rows=3, columns=2 on a 2 graphiques par ligne sur 3 lignes…
show.given	argument logique (éventuellement de longueur 2 si la formule comporte 2 variables de conditionnement) : si "TRUE" un diagramme des variables de conditionnement est tracé au-dessus des diagrammes de dispersion.
col	vecteur de couleurs utilisées pour tracer les points. S'il est trop court, les valeurs sont recyclées.
pch	vecteur de symboles ou de caractères utilisés pour représenter les points. S'il est trop court, les valeurs sont recyclées.
bar.bg	vecteur avec une composante "num" pour les variables conditionnelles numériques et "fac" pour les variables conditionnelles qualitatives (facteurs dans une data.frame), donnant la couleurs de fond des barres du diagramme des variables de conditionnement.
xlab, ylab	label pour l'axe des abscisses et l'axe des ordonnées
axlabels	fonction pour créer des graduations sur les axes si x ou y sont des variables qualitatives.
number	nombre d'intervalles pour les variables conditionnelles, il peut y en avoir deux s'il y a deux variables conditionnelles a et b. Cet argument est utilisé uniquement si les variables conditionnelles sont numériques.
xlim, ylim	limites inférieures et supérieures des axes

Lorsqu'on a plusieurs lignes, les sous-fenêtres sont organisées en partant du bas et de la gauche.

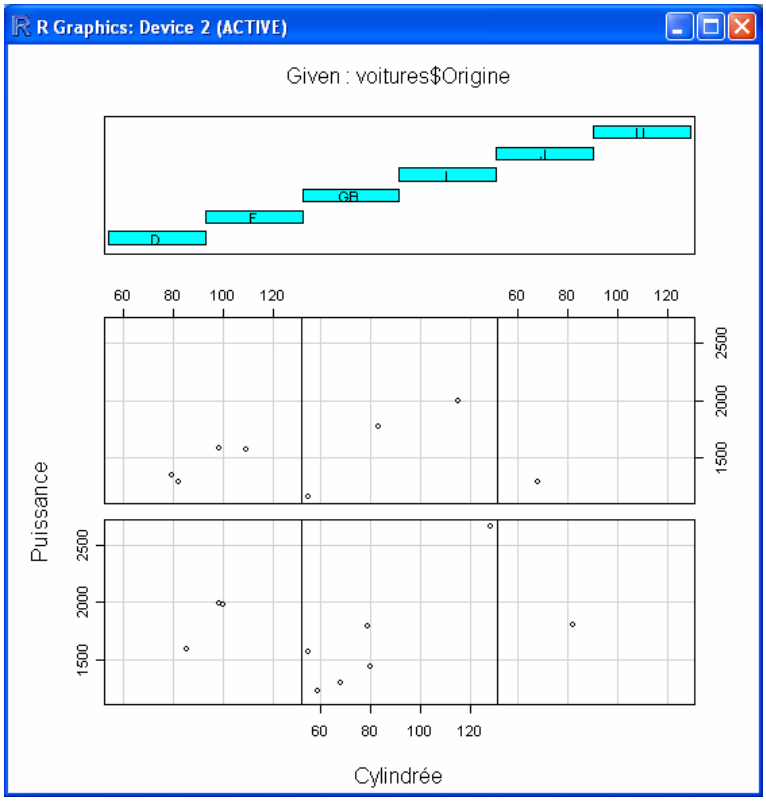
Exemple :

```
coplot(voitures$Cylindrée~voitures$Puissance|voitures$Origine,
rows=2,
xlab="Cylindrée",ylab="Puissance",
bar.bg=c(fac="cyan"))
```

Avec l'option bar.bg, on a précisé la couleur du diagramme de la variable conditionnelle.

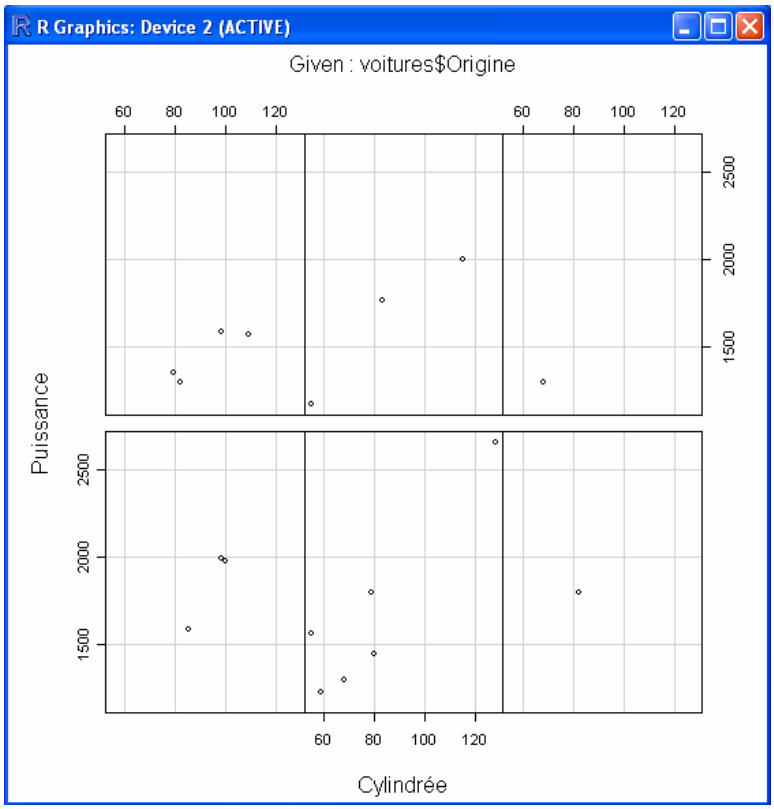
On a spécifié avec rows=2 que les données doivent être réparties sur deux lignes. Dans le tableau ci-dessous, les données sont organisées comme suit :

I : voitures italiennes	J : voitures japonaises	U : voiture russe
D : voitures allemandes	F : voitures françaises	GB : voiture britannique



Si on precise l'option show.given=FALSE, on supprime la representation de la variable conditionnelle :

```
coplot(voitures$Cylindrée~voitures$Puissance|voitures$Origine,
+ show.given=FALSE,
+ rows=2,
+ xlab="Cylindrée",ylab="Puissance")
```



2.11. Superposition de plusieurs diagrammes de dispersion : fonction matplot

matplot(x,y) trace le diagramme de dispersion de la 1ère colonne de x contre la 1ère de y, la 2ème de x contre la 2ème de y, etc.

Arguments :

matplot(x, y, type = "p", lty = 1:5, lwd = 1, pch = NULL, col = 1:6, cex = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, add = FALSE, verbose = getOption("verbose"))

x, y vecteurs ou matrices. S'il s'agit de matrices, elles doivent comporter le même nombre de colonnes.

Pour les autres arguments, cf. § 4 sur les paramètres graphiques.

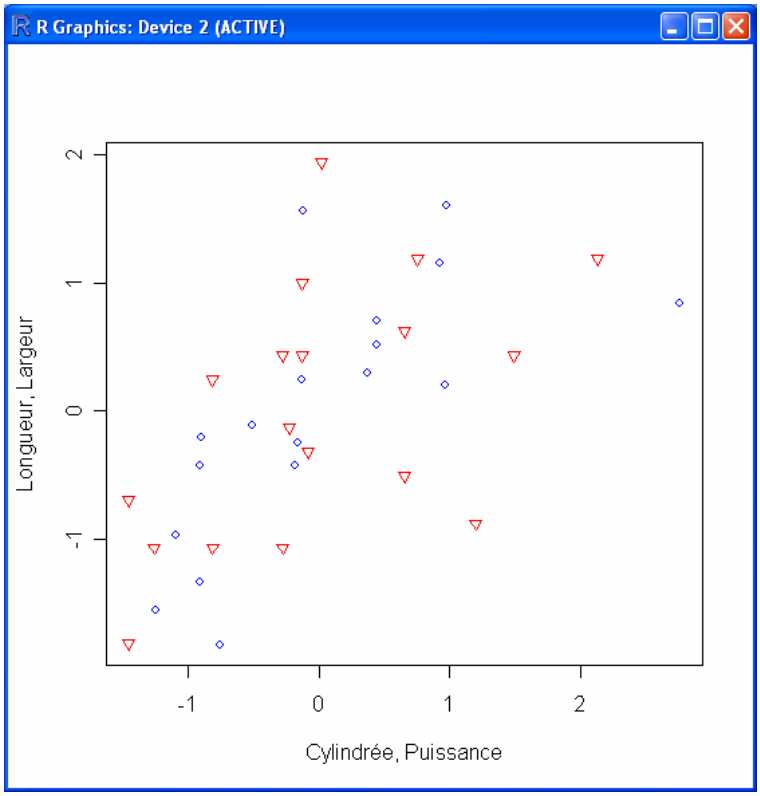
Exemple :

On trace la longueur en fonction de la cylindrée et la largeur en fonction de la puissance. On centre et on réduit les variables, pour qu'elles soient d'échelle comparable.

```
voitures$Cylindrée2=(voitures$Cylindrée-mean(voitures$Cylindrée)) / sd(voitures$Cylindrée)
voitures$Puissance2=(voitures$Puissance-mean(voitures$Puissance)) /
sd(voitures$Puissance)
voitures$Longueur2=(voitures$Longueur-mean(voitures$Longueur)) / sd(voitures$Longueur)
voitures$Largeur2=(voitures$Largeur-mean(voitures$Largeur)) / sd(voitures$Largeur)

X=voitures[11:12]
Y=voitures[13:14]

matplot(X,Y,pch=c(21,25),col=c("blue","red"),
xlab="Cylindrée, Puissance", ylab="Longueur, Largeur")
```



2.12. Diagramme en étoile : stars

Si x est une matrice ou une data.frame, stars(x) dessine un diagramme en étoiles où chaque ligne de x est représentée par une étoile et les colonnes par les longueurs des branches.

Arguments :

stars(	x, full = TRUE, scale = TRUE, radius = TRUE, labels = dimnames(x)[[1]], locations = NULL, nrow = NULL, ncol = NULL, len = 1, xlim = NULL, ylim = NULL, flip.labels = NULL, draw.segments = FALSE, col.segments = 1:n.seg, col.stars = NA, axes = FALSE, frame.plot = axes, main = NULL, sub = NULL, xlab = "", ylab = "", cex = 0.8, lwd = 0.25, lty = par("lty"), add = FALSE, plot = TRUE, ...)
x	matrice ou data.frame. Une étoile sera tracée pour chaque ligne de x. Les valeurs manquantes NA sont autorisées, elles sont traitées de la même manière que les 0.
full	argument logique. SI TRUE, les étoiles sont tracées à l'intérieur d'un cercle. Si FALSE, elles sont tracées à l'intérieur du demi-cercle supérieur.
scale	argument logique. Si TRUE, les valeurs de chaque colonne sont recalculées de manière à ramener le maximum à 1 et le minimum à 0, et ce calcul se fait indépendamment pour chaque colonne. SI FALSE, on part du principe que l'échelle des valeurs de chaque colonne est déjà [0,1].
radius	argument logique. Si TRUE, le rayon correspondant à chaque variable est tracé.
labels	vecteur de chaînes de caractères pour nommer les étoiles.
locations	matrice de 2 colonnes indiquant les coordonnées x et y pour placer chaque étoile, ou un nombre de logeur 2 si toutes les étoiles doivent être superposées (« toile d'araignée »). Par défaut, 'locations=NULL' et les étoiles sont placées dans une grille rectangulaire.

nrow,ncol	entiers renseignant le nombre de lignes et de colonnes à utiliser quand 'locations=NULL'. Par défaut, nrow=ncol.
len	longueur des rayons
flip.labels	argument logique. Si FALSE, les labels sont tous sur la même ligne, sinon ils sont alternativement décalés en haut et en bas.
draw.segments	argument logique. Si TRUE, colorie l'intérieur des segments.
col.segments	vecteur de couleurs pour les segments. Ignoré si draw.segments=FALSE.
col.stars	vecteur de couleurs pour chaque étoile. Ignoré si draw.segments=TRUE.
axes	argument logique. Si TRUE, des axes sont tracés.
frame.plot	Si TRUE, un cadre est tracé autour des graphiques.

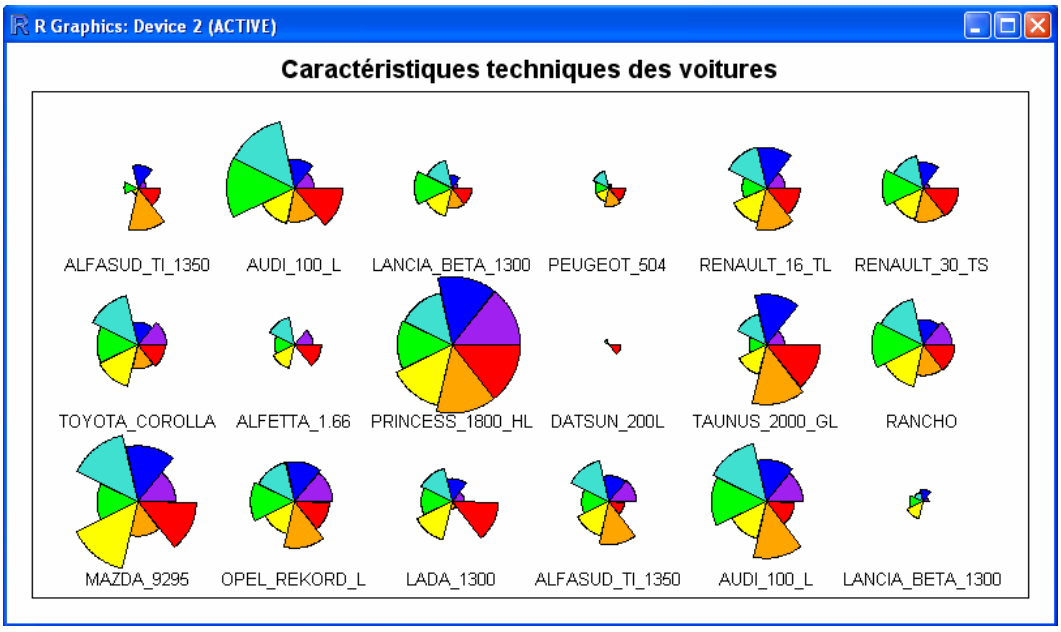
Pour les autres arguments, cf. § 4.

Exemple : *on visualise les performances techniques des 18 voitures du fichier voitures. On crée d'abord la data.frame voiture qui contient uniquement les variables numériques du fichier voitures.*

```
voiture
   Cylindrée Puissance Longueur Largeur      Poids    Vitesse    Prix
1       1350       79       393       161       870       165       30570
2       1588       85       468       177       1110      160       39990
3       1294       68       424       168       1050      152       29600
……
18     1294       68       404       161       955       140       22100
```

Afin de mieux visualiser les différences techniques entre les voitres, on colorie l'intérieur des segments (options draw.segments et col.segments). On répartit les voitures sur 3 lignes (option ncol).

```
stars(voiture,
labels=c("ALFASUD_TI_1350", "AUDI_100_L", "LANCIA_BETA_1300", "PEUGEOT_504",
"RENAULT_16_TL" ,"RENAULT_30_TS", "TOYOTA_COROLLA", "ALFETTA_1.66",
"PRINCESS_1800_HL", "DATSUN_200L", "TAUNUS_2000_GL", "RANCHO",
"MAZDA_9295", "OPEL_REKORD_L", "LADA_1300"),
ncol=6,flip.labels=FALSE,draw.segments=TRUE,
col.segments=c("purple","blue","turquoise","green","yellow","orange","red"),
frame.plot=TRUE, main="Caractéristiques techniques des voitures")
```



2.13. Visualisation des déviations de l'hypothèse d'indépendance entre deux variables nominales : fonction assocplot

assocplot(x) trace le graphe de Cohen-Friendly qui indique les déviations de l'hypothèse d'indépendance des lignes et des colonnes dans un tableau de contingence à deux dimensions.

Arguments :

assocplot(x, col = c("black", "red"), space = 0.3, main = NULL, xlab = NULL, ylab = NULL)

x	table de contingence sous forme de matrice
col	un vecteur caractère de dimension 2 donnant les couleurs utilisées pour dessiner les résidus de Pearson respectivement positifs et négatifs.
space	espace entre les rectangles (exprimé en fraction de la moyenne de la largeur et de la hauteur du rectangle).

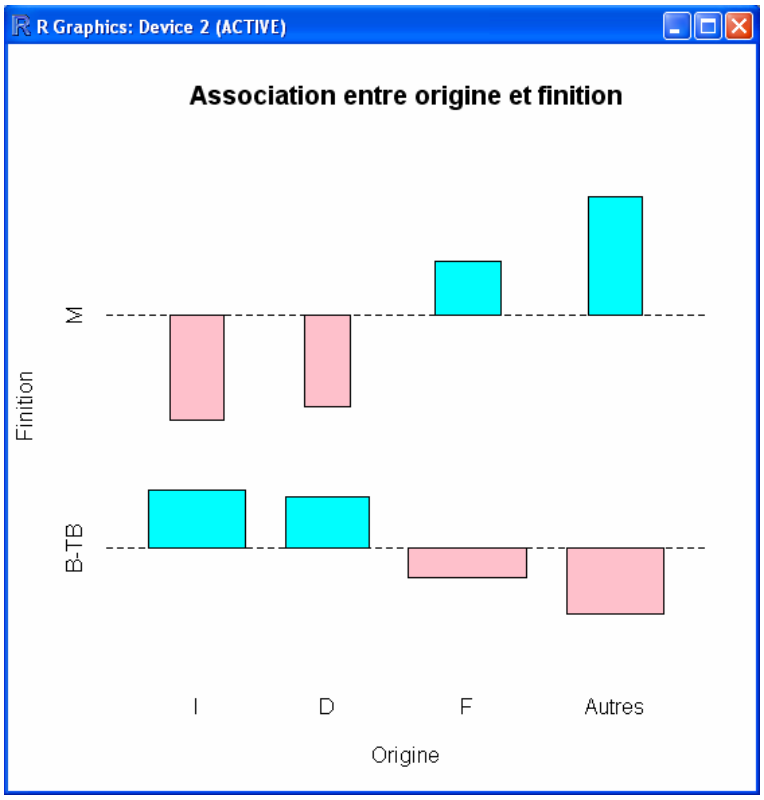
Pour les autres arguments, cf. § 4.

Exemple :

Avec la fonction assocplot, on va regarder s'il existe une liaison entre l'origine et la finition de la voiture. On constitue d'abord la table de contingence X :

```
voitures$Origin2=factor(voitures$Origine,levels=c("I", "D", "F", "Autres"))
voitures$Origin2[voitures$Origine=="GB"]="Autres"
voitures$Origin2[voitures$Origine=="J"]="Autres"
voitures$Finition2=factor(voitures$Finition,levels=c("M", "B-TB"))
voitures$Finition2[voitures$Finition=="B"]="B-TB"
voitures$Finition2[voitures$Finition=="TB"]="B-TB"

X=table(voitures$Origin2,voitures$Finition2)
X
        M B-TB
  I     0    4
  D     0    3
  F     2    4
  Autres 2   2
```



2.14. Visualisation de l'association entre deux variables dichotomiques : fonction fourfoldplot

fourfoldplot(x) visualise, avec des quarts de cercle, l'association entre deux variables dichotomiques pour différentes populations (x doit être un array avec dim=c(2,2,k) ou une matrice avec dim=c(2,2) si k=1).

fourfoldplot(x, color = c("#99CCFF", "#6699CC"), conf.level = 0.95, std = c("margins", "ind.max", "all.max"), margin = c(1, 2), space = 0.2, main = NULL, mfrow = NULL, mfcol = NULL)

x	un tableau (array) de dimension (2,2,k) où k est le nombre de sous-populations ou une matrice (2,2) si on étudie une seule population.
color	un vecteur de longueur 2 spécifiant les couleurs à utiliser pour la plus petite et la plus grande diagonale de chacune des tables (2,2).
conf.level	intervalle de confiance utilisé pour les cercles de confiance des odds ratios. Ce doit être un nombre positif ou nul inférieur à 1. Si conf.level=0, les cercles de confiance sont supprimés.
std	chaîne de caractère spécifiant comment standardiser la table. Cette chaîne de caractère peut prendre 3 valeurs : « margins », « ind.max » ou « all.max ». Si std=margins, chacune des table est standardisée pour égaliser les marges tout en conservant les mêmes odds ratios. Si std=ind.max ou al.max, les tables (2,2) sont individuellement ou simultanément standardisées de sorte que la fréquence de la cellule maximum soit égale à 1.
margin	vecteur numérique, égal à '1', '2' ou c(1,2) (par défaut), qui correspond à la standardisation de la ligne, de la colonne ou des deux marges pour chacune des tables. Utilisé uniquement si std=margins.
space	espace utilisé pour le labels des lignes et des colonnes (en pourcentage du rayon maximal des quarts de cercle).
main	titre du graphique

mfrow vecteur numérique de la forme c(nr,nc) indiquant l'organisation des tables dans la fenêtre graphique selon comportant nr lignes et nc colonnes.

mfcol vecteur numérique de la forme c(nr,nc) indiquant l'organisation des tables dans la fenêtre graphique selon comportant nr lignes et nc colonnes. Ce paramètre a le même rôle que mfrow.

Exemple :

A partir du fichier chien, on visualise la relation entre les variables affection et agressivité par fonction.

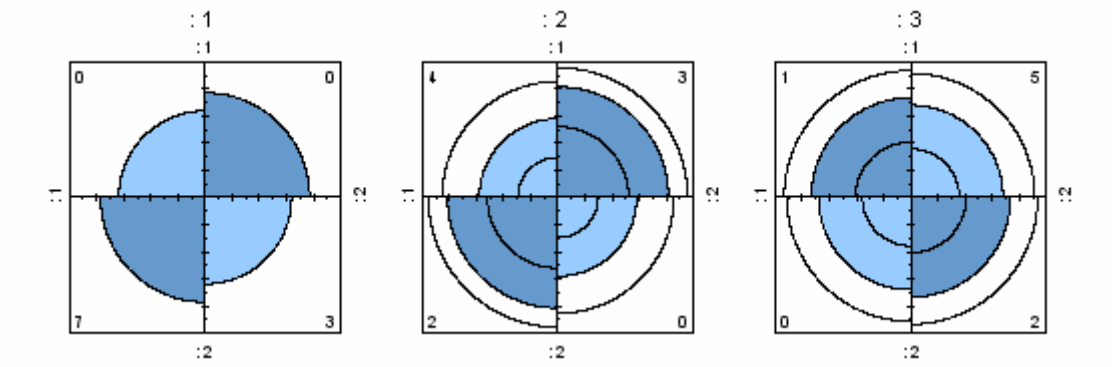
```
X=table(chien$Affection,chien$Agressivite,chien$Fonction)
X
, ,  = 1
   1 2
 1 0 0
 2 7 3

, ,  = 2
   1 2
 1 4 3
 2 2 0

, ,  = 3
   1 2
 1 1 5
 2 0 2

ls.str(pattern="X")
X :  int [1:2, 1:2, 1:3] 0 7 0 3 4 2 3 0 1 0 ...

fourfoldplot(X,mfrow=c(1,3))
```



On obtient le même résultat avec la commande fourfoldplot(X,mfcol=c(1,3)).

2.15. Graphiques Quantiles-Quantiles : fonctions qqnorm(x) et qqplot(x,y)

qqnorm(x) permet d'obtenir les quantiles de x en fonction des valeurs attendues selon une loi normale.

```
qqnorm(  y, ylim, main = "Normal Q-Q Plot",
         xlab = "Theoretical Quantiles",
         ylab = "Sample Quantiles", datax = FALSE,
         ...)
```

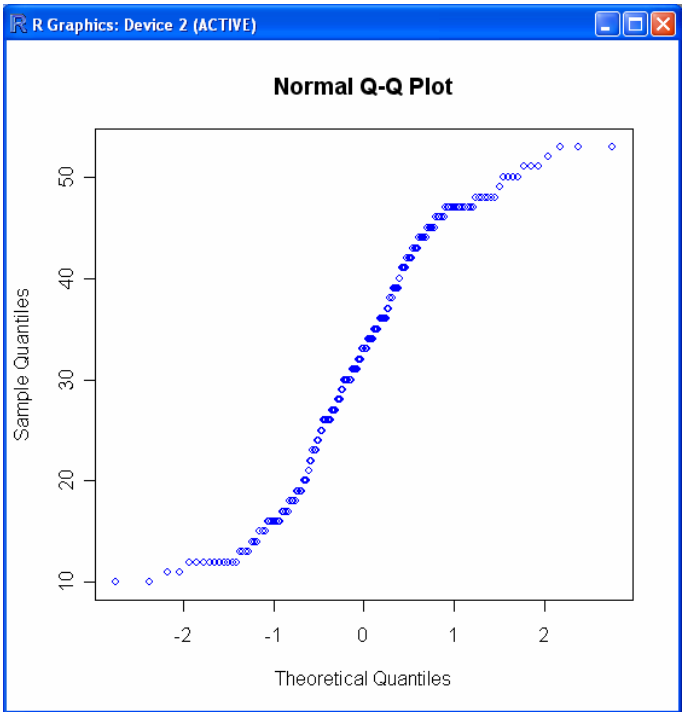
y vecteur de données à analyser

datax argument logique. Si FALSE, les données observées sont représentées sur l'axe des ordonnées. Si TRUE, elles sont représentées sur l'axe des abscisses.

Pour les autres arguments, cf. § 4.

Exemple :

qqnorm(paysniv3$NAT,col=c("blue"))



qqplot(x,y) trace les quantiles de y en function des quantiles de x.

Arguments :

qqplot(x, y, xlab = deparse(substitute(x)),
ylab = deparse(substitute(y)), ...)

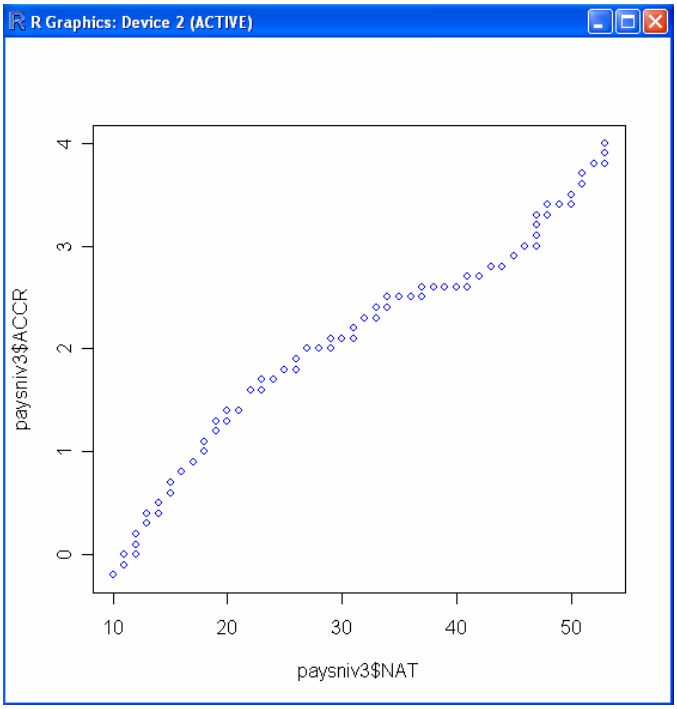
x vecteur des valeurs de l'axe des absisses

y vecteur des valeurs de l'axe des ordonnées

Pour les autres arguments, cf. § 4.

Exemple :

qqplot(paysniv3$NAT,paysniv3$ACCR,col=c("blue"))



2.16. Graphique des effets d'interaction : fonction interaction.plot(f1,f2,y)

Cette fonction graphique s'applique lorsqu'on effectue de l'analyse de la variance. Dans le cas où il a deux facteurs (entendus ici comme variables explicatives qualitatives), on peut visualiser leurs effets sur la variable à expliquer et plus particulièrement regarder si les effets des 2 facteurs sont indépendants ou s'il existe un terme du second ordre (correspondant à l'interaction entre les 2 facteurs) significatif.

En général, on trace pour ce faire le graphe des moyennes de la variable réponse y en fonction des valeurs du facteur f1 (sur l'axe des x) et du facteur f2 (différentes courbes). Avec interaction.plot, il est possible de choisir la statistique résumée de x (par défaut fun=mean).

Arguments :

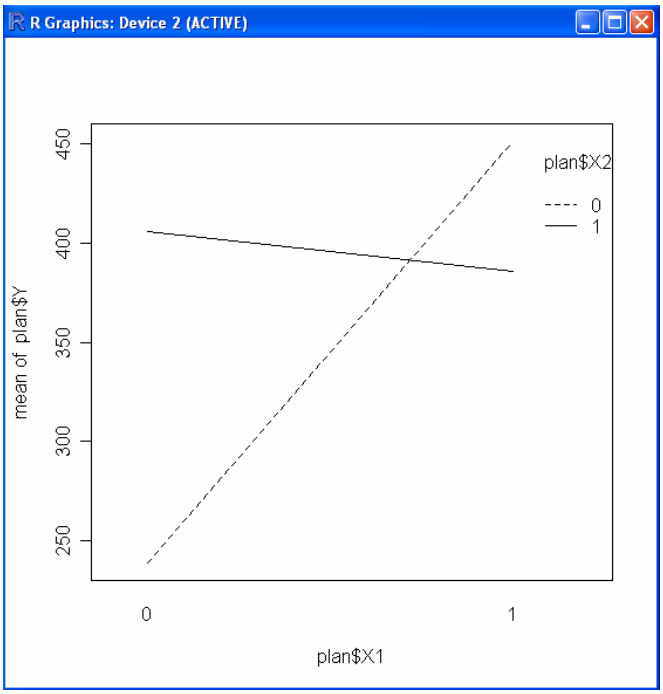
interaction.plot(x.factor, trace.factor, response, fun = mean, type = c("l", "p", "b"), legend = TRUE, trace.label = deparse(substitute(trace.factor)), fixed = FALSE, xlab = deparse(substitute(x.factor)), ylab = ylabel, ylim = range(cells, na.rm=TRUE), lty = nc:1, col = 1, pch = c(1:9, 0, letters), xpd = NULL, leg.bg = par("bg"), leg.bty = "n", xtick = FALSE, xaxt = par(« xaxt »), axes = TRUE, …)

x.factor	un facteur dont les niveaux seront représentés sur l'axe des abscisses.
trace.factor	deuxième facteur dont les niveaux seront représentés sur différentes courbes.
response	variable réponse mesurée lors de l'expérimentation (variable à expliquer)
fun	statistique résumée de y (par défaut la moyenne). Doit retourner une valeur numérique unique.
legend	argument logique. Si TRUE, une légende figure sur le graphe.
trace.label	label de la légende
fixed	argument logique. Si TRUE, la légende est dans l'ordre des niveaux du 2ème facteur.

xpd	détermine l'endroit où la légende est placée. Voir dans l'aide en ligne 'par(xpd').
leg.bg, leg.bty	arguments appliqués à la légende.
xtick	argument logique. Si TRUE, l'axe des x est gradué.

Exemple : *on visualise les résultats d'une expérience où X1 et X2 sont les facteurs que l'on fait varier et Y est la réponse.*

```
plan
  Y   X1  X2
1 493 1   0
2 410 0   1
3 402 0   1
4 410 1   0
5 434 1   1
6 261 0   0
7 216 0   0
8 338 1   1
interaction.plot(plan$X1,plan$X2,plan$Y)
```



Dans cet exemple, les 3 effets sont présents : X1, X2 et interaction entre X1 et X2 (l'influence de X1 n'est pas la même selon le niveau de X2). S'il n'y avait pas d'interaction, les courbes seraient parallèles.

3. Les fonctions graphiques secondaires

R propose un ensemble de fonctions graphiques dites « secondaires » qui ont une action sur un graphe déjà existant. Nous présentons dans le tableau suivant les fonctions graphiques secondaires principales *(extrait de R pour les débutants d'Emmanuel Paradis)*.

points(x, y)	Ajoute des points (l'option type= peut être utilisée)
lines(x, y)	Idem mais avec des lignes
text(x, y, labels, ...)	Ajoute le texte spécifié par labels au coordonnées (x,y). Un usage typique sera: plot(x, y, type="n"); text(x, y, names)
mtext(text, side=3, line=0, ...)	Ajoute le texte spécifié par text dans la marge spécifiée par side (cf. axis() plus bas) ; line spécifie la ligne à partir du cadre de traçage
segments(x0, y0, x1, y1)	Trace des lignes des points (x0,y0) aux points (x1,y1)
arrows(x0, y0, x1, y1, angle=30, code=2)	Idem avec des flèches aux points (x0, y0) si code=2, aux points (x1,y1) si code=1, ou aux deux si code=3; angle contrôle l'angle de la pointe par rapport à l'axe
abline(a,b)	Trace une ligne de pente b et ordonnée à l'origine a
abline(h=y)	Trace une ligne horizontale sur l'ordonnée y
abline(v=x)	Trace une ligne verticale sur l'abscisse x
abline(lm.obj)	Trace la droite de régression donnée par lm.obj
rect(x1, y1, x2, y2)	Trace un rectangle délimité à gauche par x1, à droite par x2, en bas par y1 et en haut par y2
polygon(x, y)	Trace un polygone reliant les points dont les coordonnées sont données par x et y
legend(x, y, legend)	Ajoute la légende au point de coordonnées (x,y) avec les symboles données par legend
title()	Ajoute un titre et optionnellement un sous-titre
axis(side, vect)	Ajoute un axe en bas (side=1), à gauche (2), en haut (3) ou `a droite (4) ; vect (optionnel) indique les abscisses (ou ordonnées) où les graduations seront tracées
rug(x)	Dessine les données x sur l'axe des x sous forme de petits traits verticaux
locator(n, type="n", ...)	retourne les coordonn´ees (x y) après que l'utilisateur ait cliqué n fois sur le graphe avec la souris ; également trace des symboles (type="p") ou des lignes (type="l") en fonction de paramètres graphiques optionnels (...) ; par défaut ne trace rien (type="n")

4. Paramètres graphiques

À chaque fonction graphique principale sont associés des paramètres consultables via l'aide en ligne (taper help(''nom_de_la_fonction''), par exemple help(''mosaicplot''). Certains de ces paramètres sont identiques pour plusieurs fonctions graphiques, les principaux d'entre eux sont récapitulés ci-dessous (avec leurs valeurs par défaut) :

add=FALSE	si TRUE superpose le graphe au graphe existant
axes=TRUE	si FALSE ne trace pas les axes ni le cadre
type=''p''	type de graphe qui sera tracé : type="p" points type="l" lignes type="b" points et lignes type="c" lignes sauf à l'endroit des coordonnées des points type="o" comme "b" mais les lignes recouvrent les points type="h" lignes verticales type="s" lignes en escalier, les données sont représentées par le sommet des lignes verticales type="S" lignes en escaliers, les données sont représentées par le bas des lignes verticales type="n" pour aucune représentation des données
xlim=, ylim=	fixe les limites inférieures et supérieures des axes
xlab=, ylab=	annotations des axes, doivent être des variables de mode caractère
main=	titre principal, doit être une variable de mode caractère
sub=	sous-titre (écrit dans une police plus petite)

Il existe 68 autres paramètres graphiques dont la liste détaillée peut être obtenue grâce à la commande ?par. Voici une liste de paramètres graphiques couramment utilisés *(tableau extrait de R pour les débutants d'Emmanuel Paradis)*.

adj	Contrôle la justification du texte (0 à gauche, 0.5 centré, 1 à droite)
bg	Spécifie la couleur de l'arrière-plan (ex : bg="red", bg="blue", …). La liste des 657 couleurs disponibles est affichée avec colors().
bty	Contrôle comment le cadre est tracé, valeurs permises : "o", "l", "7", "c", "u" ou "]" (le cadre ressemblant au caractère correspondant) ; bty="n" supprime le cadre
cex	Une valeur qui contrôle la taille des caractères et des symboles par rapport au défaut ; les paramètres suivants ont le même contrôle pour les nombres sur les axes, cex.axis, les annotations des axes, cex.lab, le titre, cex.main, le sous-titre, cex.sub
col	Contrôle la couleur des symboles ; comme pour cex il y a : col.axis, col.lab, col.main, col.sub Pour avoir la liste des couleurs disponibles, on tape les commandes suivantes : library(grDevices) colors()
font	Un entier qui contrôle le style du texte (1 : normal, 2 : italique, 3 : gras, 4 : gras italique) ; comme pour cex il y a : font.axis, font.lab, font.main, font.sub
las	Un entier qui contrôle comment sont disposées les annotations des axes (0 : parallèles aux axes, 1 : horizontales, 2 : perpendiculaires aux axes, 3 : verticales)
lty	Contrôle le type de ligne tracée, peut être un entier (1 : continue, 2 : tirets, 3 : points, 4 : points et tirets alternés, 5 : tirets longs, 6 : tirets courts et longs alternés), ou ensemble de 8 caractères maximum (entre "0" et "9") qui spécifie alternativement la longueur, en points ou pixels, des éléments tracés et des blancs, par exemple lty="44" aura le même effet que lty=2

lwd	Une valeur numérique qui contrôle la largeur des lignes
mar	Un vecteur de 4 valeurs numériques qui contrôle l'espace entre les axes et le bord de la figure de la forme c(bas, gauche, haut, droit), les valeurs par défaut sont c(5.1, 4.1, 4.1, 2.1)
mfcol	Un vecteur de forme c(nr,nc) qui partitionne la fenêtre graphique en une matrice de nr lignes et nc colonnes, les graphes sont ensuite dessinés en colonne
mfrow	Idem mais les graphes sont ensuite dessinés en ligne
pch	Contrôle le type de symbole, soit un entier entre 1 et 25, soit n'importe quel caractère entre guillemets
ps	Un entier qui contrôle la taille en points du texte et des symboles
pty	Un caractère qui spécifie la forme du graphe, "s" : carrée, "m" : maximale
tck	Une valeur qui spécifie la longueur des graduations sur les axes en fraction du plus petit de la largeur ou de la hauteur du graphe ; si tck=1 une grille est tracée
tcl	Une valeur qui spécifie la longueur des graduations sur les axes en fraction de la hauteur d'une ligne de texte (défaut tcl=-0.5)
xaxt	si xaxt="n" l'axe des x est défini mais pas tracé (utile avec axis(side=1, ...))
yaxt	Si yaxt="n" l'axe des y est défini mais pas tracé (utile avec axis(side=2, ...))

5. Le package lattice

Nous reprenons dans ce dernier chapitre les exemples donnés dans le polycopié de cours d'Arthur Tenenhaus.

L'idée principale derrière le package lattice est celle des graphes multiples conditionnés : un graphe bivarié entre deux variables sera découpé en plusieurs graphes en fonction des valeurs d'une troisième variable. La plupart des fonctions de lattice prennent pour argument principal une formule, par exemple ~x| y . y~x | z signifie que le graphe de y en fonction de x sera dessiné en plusieurs sous graphes en fonction des valeurs de z.

Le tableau ci-dessous indique les principales fonctions du package lattice.

barchart(˜y | x)	Histogramme des valeurs de y en fonction de celles de x
bwplot((˜y | x)	Graphe « boites à moustaches »
densityplot(˜ x)	Graphe de fonctions de densité
dotplot(y ˜ x)	Graphe de Cleveland (graphes superposés ligne par ligne et colonne par colonne)
histogram(˜ x)	Histogrammes des fréquences de x
qqmath(˜ x)	Quantiles de x en fonction des valeurs attendues selon une distribution théorique
stripplot(y ˜ x)	Graphe unidimensionnel, x doit être numérique, y peut être un facteur
qq(y ˜ x)	Quantiles pour comparer deux distributions, x doit être numérique, y peut être numérique, caractère ou facteur mais doit avoir deux « niveaux »
xyplot(y ˜ x)	Graphes bivariés (avec de nombreuses fonctionnalités)
levelplot(z˜x*y)	Graphe en couleur des valeurs de z aux coordonnées fournies par x et y (x, y et z sont tous de même longueur)
splom(˜ x)	Matrice de graphes bivariés
parallel(˜ x)	Graphe de coordonnées parallèles

Pour accéder aux fonctions du package lattice, il faut charger au préalable le package en mémoire via la commande library(lattice).

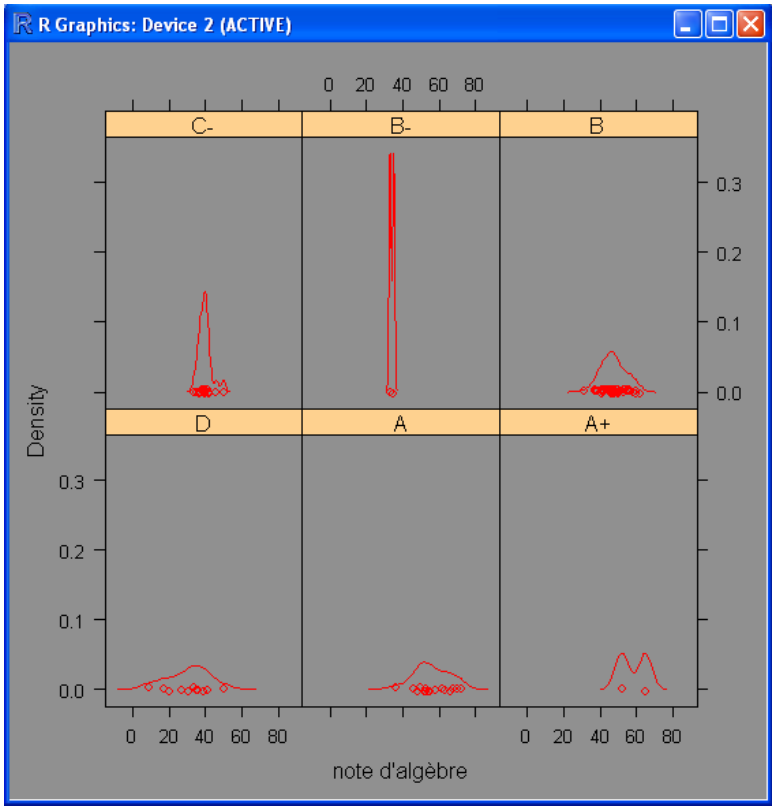
Pour illustrer l'utilisation des fonctions densityplot, bwplot, histogram, xyplot et splom, nous allons utiliser le jeu de données deug disponible dans le package ade4. Deug fournit les résultats de 104 étudiants sur 9 matières (Algebra, Analysis, Proba, Informatic, Economy, Option1, Option2, English, Sport). Deug fournit également les résultats globaux pour chacun des étudiants (A, B, C, D). On souhaite évaluer l'impact de la note d'algèbre sur le résultat global.

```
data(deug)
x=deug$tab$Algebra
y=deug$result
```

L'objectif de cette étude est de découper les valeurs des notes d'analyse suivant les résultats finaux des 104 étudiants. On souhaite donc découper l'ensemble des valeurs de x en tranches et, pour chacune de ces tranches, construire une courbe de densité (densityplot), une boîte à moustaches (bwplot), un histogramme (histogram).

5.1. Fonction densityplot

```
densityplot(~x|y,
+ xlab="note d'algèbre",
+ col="red")
```

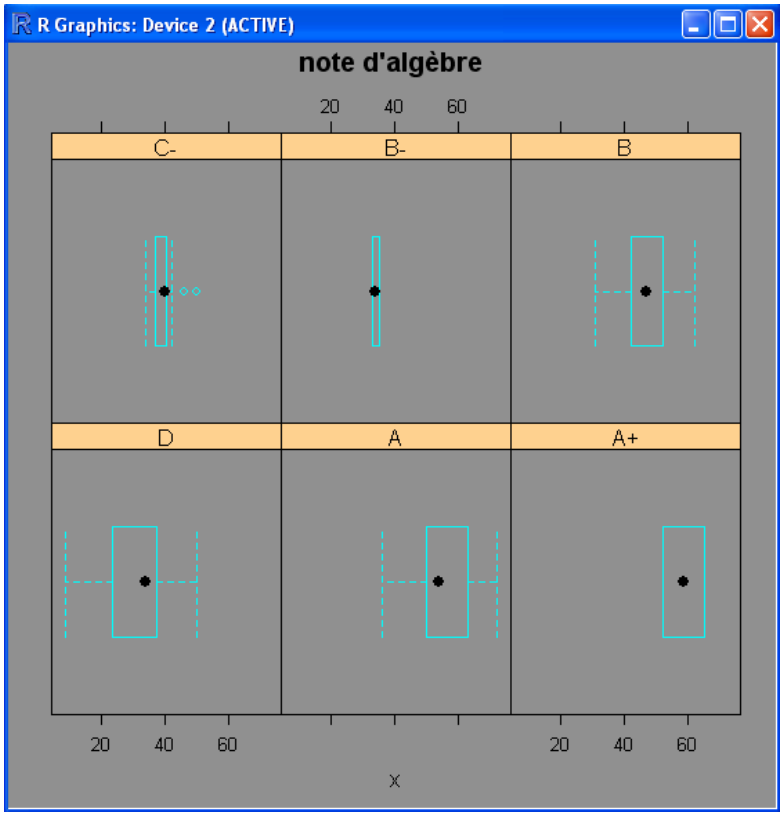


Comme on pouvait s'y attendre, les meilleurs étudiants en analyse ont également tendance à obtenir les meilleurs résultats globaux.

5.2. Fonction bwplot

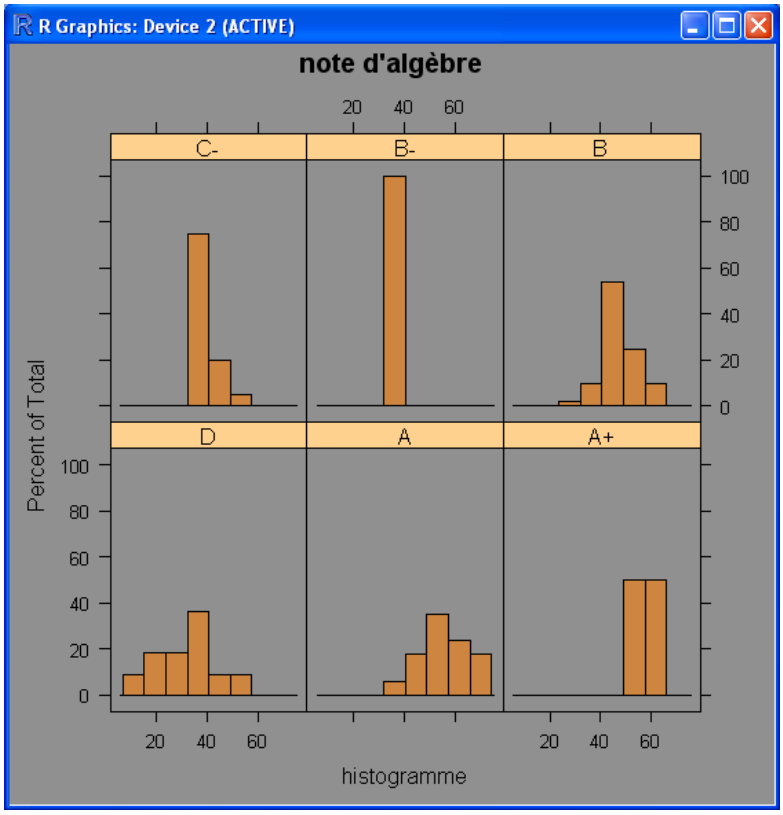
La fonction bwplot fournit des informations globalement identiques au graphique précédent, mais sous forme de boîtes à moustaches.

```
bwplot(~x|y,
+ main="note d'algèbre")
```



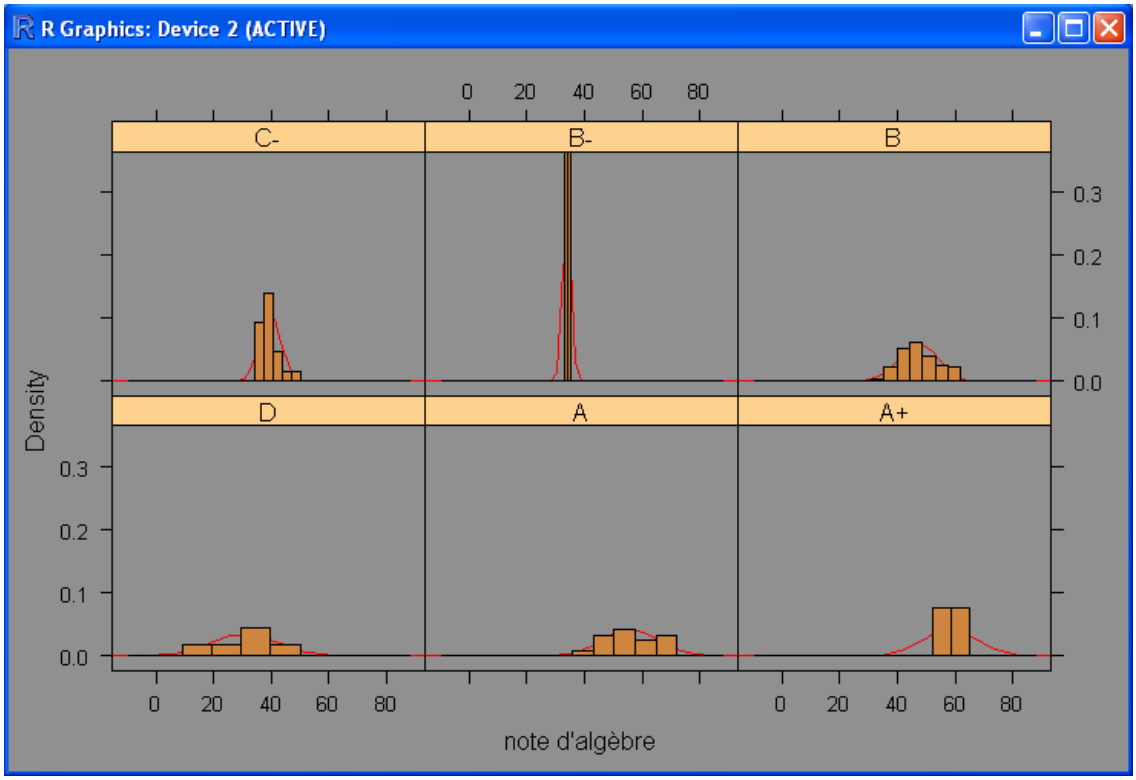
5.3. Fonction histogram

```
histogram(~x|y,
col="peru",
main="note d'algèbre",
xlab="histogramme")
```



5.4. Utilisation de l'argument panel

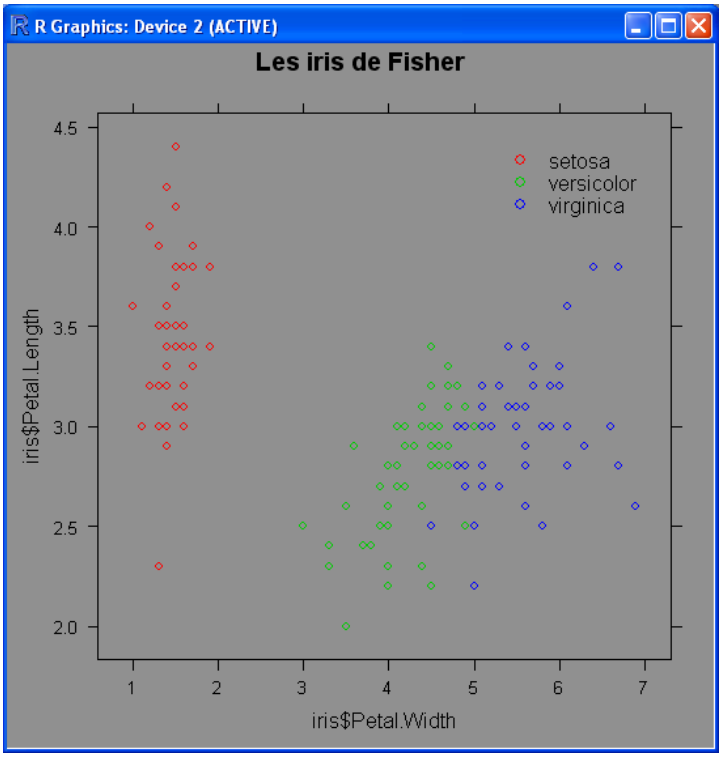
```
densityplot(~x|y,xlab="note d'algèbre",
panel=function(x,...)
{panel.mathdensity(dmath=dnorm, args=list(mean=mean(x), sd=sd(x)), col="red")
panel.histogram(x, breaks=NULL, col="peru")
})
```



La fonction densityplot produit un graphe par sous-échantillon. L'argument panel prend cette fonction pour argument. Panel définit les analyses qui doivent être ffectuées pour chaque sous-population.

5.5. Fonction xyplot

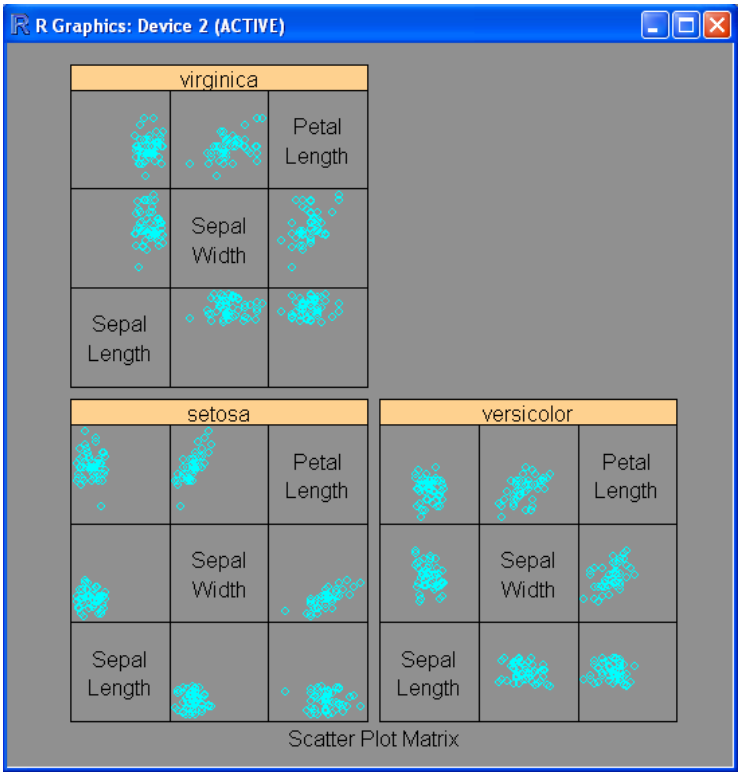
```
xyplot(iris$Petal.Length ~
iris$Petal.Width,
groups=iris$X,
col=2:4,
main="Les iris de Fisher",
key = list(x = 0.7, y = 0.85,
points=list(col=2:4,pch=1),
text=list(levels(iris$X)))
)
```



5.6. Fonction splom

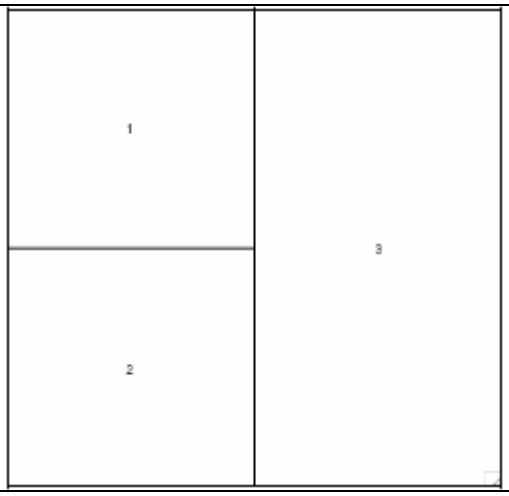
La fonction splom permet de tracer tous les graphes bivariés d'une matricen en partitionnant les graphes selon la valeur d'une autre variable.

```
splom(~iris[1:3]|iris$X,
pscales=0,
varnames=c("Sepal\nLength",
"Sepal\nWidth",
"Petal\nLength"
)
)
```

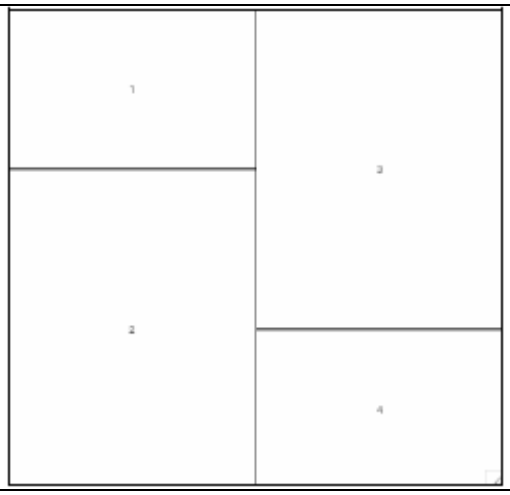


Exercices corrigés

1. Gestion des fenêtres graphiques

Créez la partition ci-contre :	

```
mat1 = matrix(c(1, 2, 3, 3), 2, 2)
layout(mat1)
layout.show(3)
```

Créez la partition ci-contre :	

```
mat2 = matrix(c(1, 2, 2,  3, 3, 4), 3, 2)
layout(mat2)
layout.show(4)
```

2. Fonctions graphiques

2.1. Utilisation de la fonction plot()

Exercice 1

Créez une matrice, mat1, composée de 100 lignes et deux colonnes ; chacune des colonnes est un vecteur aléatoire de loi normale centrée et de variance 1.

```
mat1 = matrix(rnorm(200, 0, 1), 100, 2)
```

Créez une matrice, mat2, composée de 100 lignes et deux colonnes ; chacune des colonnes est un vecteur aléatoire de loi normale de moyenne 1 et de variance 1.

```
mat2 = matrix(rnorm(200, mean = 1, sd = 0.5), 100, 2)
```

Tracez le graphe bivarié de la première colonne de mat1 sur la deuxième colonne de mat1

```
plot(mat1)
```

Ajoutez au graphe précédent le graphe bivarié de la première colonne de mat2 sur la deuxième colonne de mat2 . Que constatez vous ?
Remarque : on utilisera des couleurs différentes pour distinguer les points de mat1 des points de mat2. Pour connaître la liste des couleurs disponibles dans R, tapez la commande colors().

```
plot(mat1, col ="blue")
points(mat2, col = "red")
```

On constate que les points de mat2 ne sont affichés que dans le graphique défini par les bornes de mat1.

Utilisez les arguments xlim et ylim pour contourner ce problème.

```
mat = rbind(mat1, mat2)
plot(    mat1, col = "blue",
         xlim = range(mat[ ,1]), ylim = range(mat[ ,2]),
         main = "représentation d'un nuage de points",
         xlab = "X1", ylab = "X2"
        )
points(mat2, col = "red")
```

Utilisez les options de la fonction plot pour améliorer la présentation du graphique.

Par exemple :

```
plot(  1,
       xlim = range(mat[ ,1]),
       ylim = range(mat[ , 2]),
       main = "représentation d'un nuage de points",
       xlab = "X1",
       ylab = "X2",
       bty = "l", tcl = -.25
       )
rect(-3, -3, 3, 3, col = "cornsilk")

points(mat1, col = "blue", pch = 22, bg = "red")
points(mat2, col = "red", pch = 25, bg = "yellow")
```

Exercice 2

Dans le package datasets, on dispose du jeu de données iris. Chargez ce jeu de données en mémoire.

```
data(iris)
```

Quel est le mode de ce jeu de données ?

Ce jeu de données est une data.frame

Convertissez le jeu de données iris en une matrice nommé matrice_iris

```
matrice_iris = as.matrix(iris[ , 1:4])
```

Tapez la commande plot(matrice_iris). Que construit la fonction plot() dans le cadre des matrices ?

Dans le cadre de matrices, la fonction plot() trace le graphe de la première colonne de la matrice sur la deuxième.

Tracez le graphique composé de tous les graphiques bivariés du jeu de données Iris (à partir de matrice_iris).
Ajoutez une forme et une couleur aux points en fonction de l'espèce.
Ajoutez un titre et un label.

```
pairs(  matrice_iris[ ,1:4],
        bg = c("red", "green3","blue")[as.numeric(iris[ , 5])],
        pch = c(21, 25, 24)[iris[ ,5]],
        main = "Iris de Fisher",
        labels = c("Longueur\nSepale",
        "Largeur\nSepale",
        "Longueur\nPetale",
        "Largeur\nPetale"
                )
        )
```

Exercice 3

La fonction plot()dans le cadre d'une data.frame

Chargez en mémoire le jeu de données iris disponible dans le package datasets. Ce jeu de données est une data.frame. Tapez la commande plot(iris). Que constatez-vous ?

On constate que dans le cadre des data.frame la fonction plot trace toutes les combinaisons possibles de graphes bivariés.

Par exemple :

```
plot(  iris[ ,1:4],
       bg = c("red", "green3","blue")[iris[ ,5]],
       pch = c(21, 25, 24)[iris[ ,5]],
       main = "Iris de Fisher",
       labels = c("Longueur\nSepale",
       "Largeur\nSepale",
       "Longueur\nPetale",
       "Largeur\nPetale"
               )
       )
```

2.2. Utilisation de la fonction hist()

On s'intéresse aux liens entre le temps d'attente entre deux éruptions et la durée des éruptions pour le « old faithful geyser » du parc national du Yellowstone (Wyoming , Etats-Unis). Ce jeu de données nommé faithful est disponible dans le package datasets.

Lorsque l'on tape la commande ?faithful, on obtient un descriptif du jeu de données.

Visualisez le jeu de données faithful par l'intermédiaire de la fonction plot.

Définissez un seuil critique d'attente au delà duquel la probabilité (la prochaine éruption est longue) est forte.

Posons le seuil critique arbitrairement à 63.

Construisez un histogramme de la durée d'éruption. Ajoutez un titre, labellisez l'axe des abscisses, colorez les barres de l'histogramme en vert, colorez les traits de l'histogramme en rouge et représentez l'histogramme en terme de fréquence plutôt qu'en termes d'effectifs.

```
hist(faithful$eruptions, col = "green", border = "red", proba = TRUE,
main = "histogramme du temps des éruptions",
xlab = "le old faithful geyser"
)
```

Augmentez la taille du pas de l'histogramme à 20 et ajustez une loi normale à cet histogramme. Que remarque t'on ?

```
hist(faithful$eruptions, col = "green", border = "red", proba = TRUE,
main = "histogramme du temps des eruptions",
xlab = "le old faithful geyser", breaks = 20)

x = seq(from = 1.5, to = 5, length = 500)

y = dnorm(x, mean = mean(faithful$eruptions),
sd=sd(faithful$eruptions)
)

lines(x, y, col = "brown")
mtext("Ajustement à une loi normale")
```

L'ajustement à une loi normale n'est pas adéquat.

2.3. Utilisation des estimateurs locaux de densité

Superposez à l'histogramme les estimateurs locaux de la densité associés aux paramètres d'ajustement 0.1, 0.3, 0.8 et 1.

```
hist(faithful$eruptions, col = "yellow", border = "green", proba = TRUE,
main = "histogramme du temps des éruptions",
xlab = "le old faithful geyser", breaks = 20)

lines(density(faithful$eruptions, adj = .1), type='l', col='red', lwd=1)
lines(density(faithful$eruptions, adj = .3), type='l', col='green', lwd=1)
lines(density(faithful$eruptions, adj = .8), type='l', col='blue', lwd=1)
lines(density(faithful$eruptions, adj = 1), type='l', col='black', lwd=1)
```

On s'aperçoit que l'estimateur local de la densité associé à un coefficient d'ajustement de 0.1 dépasse de la fenêtre graphique.

Imposez des contraintes à la taille de l'axe des ordonnées.

```
hist(faithful$eruptions, col = "yellow", border = "green", proba = TRUE,
main = "histogramme du temps des éruptions",
xlab = "le old faithful geyser", breaks = 20, ylim = c(0, 1))

lines(density(faithful$eruptions, adj = .1), type='l', col='red', lwd=1)
lines(density(faithful$eruptions, adj = .3), type='l', col='green', lwd=1)
lines(density(faithful$eruptions, adj = .8), type='l', col='blue', lwd=1)
lines(density(faithful$eruptions, adj = 1), type='l', col='black', lwd=1)
```

2.4. Utilisation de la fonction boxplot()

Partitionnez la fenêtre graphique en deux dans le sens horizontal.

```
vecteur = t(1:2)
layout(vecteur)
```

Dans le premier cadre de la partition, tracez les boîtes à moustache associées aux durées d'éruption conditionnellement aux temps d'attente entre deux éruptions

```
boxplot(faithful$eruption ~ faithful$waiting)
```

Dans le second cadre de la partition, utilisez le seuil calculé précédemment pour construire deux boites à moustache associées aux durées d'éruption conditionnellement aux temps d'attente entre deux éruptions, l'une codant les séismes courts et l'autre les séismes longs. Pour ce faire créez une variable égale à court si waiting est inférieur au seuil et à long si waiting est supérieur au seuil.

```
Y = character(272)
Y[faithful$waiting > 63] = "long"
Y[faithful$waiting <= 63] = "court"
Y = as.factor(Y)
boxplot(faithful$eruption ~ Y)
```

Améliorez la présentation du graphique (Couleur, label, titre) et ajoutez un indicateur de densité.

```
boxplot(faithful$eruption ~ Y, col = c("yellow", "red"),
main = "boxplot en fonction de la duree du seisme",
xlab = "court vs long" )
rug(faithful$eruption, side = 2)
```

2.5. Utilisation de la fonction pie()

On souhaite visualiser, via les boîtes à moustache, la proportion de séismes dont la durée est supérieur à 3 minutes.

```
Y = numeric(272)
Y[faithful$eruption > 3] = 1
Y[faithful$eruption <= 3] = 0

Pourcentage_court = length(Y[Y == 0])/nrow(faithful)
Pourcentage_long = length(Y[Y == 1])/nrow(faithful)
Seisme = c(Pourcentage_court, Pourcentage_long)
names(Seisme) = c("seisme long", "seisme court")
pie(Seisme, col = c("yellow", "red"), main = "duree des seismes", border =NA)
```

2.6. Utilisation du package lattice

Chargez le package lattice en mémoire

```
library(lattice)
```

Nous allons illustrer l'utilisation les fonctions du package lattice à l'aide du jeu de données bordeaux_R.txt.

Il s'agit d'un jeu de données qui fournit la qualité de vin de bordeaux (QUALITE) en fonction de 4 facteurs (TEMPERAT, SOLEIL, CHALEUR, PLUIE).

Chargez en mémoire ce jeu de données sur R

```
A = read.table('/chemin/bordeaux_R.txt', header = TRUE, sep = '\t')

TEMPERAT = somme des températures moyennes journalières (°C)
SOLEIL = Durée d'insolation (h)
CHALEUR = Nombre de jours de grande chaleur
PLUIE = hauteur de pluie en (mm)
QUALITÉ : 1 = BON, 2 = MOYEN et 3 = MÉDIOCRE
```

On souhaite évaluer l'impact des différents variables sur la qualité des vins de bordeaux.

Tracez les estimateurs de la densité de chacune des variables conditionnellement à la qualité. Qu'en concluez-vous ?

```
densityplot(~ A$TEMPERAT | A$QUALITE, xlab = "Temperature", col = "red")
densityplot(~ A$SOLEIL | A$QUALITE, xlab = "Soleil", col = "Yellow")
densityplot(~ A$CHALEUR | A$QUALITE, xlab = "Chaleur" , col = "green")
densityplot(~ A$PLUIE | A$QUALITE, xlab = "Pluie", col = "blue")
```

On peut interpréter l'influence de chaque variable sur la qualité du produit.

Par exemple : à l'aide du premier graphique sur la température, on peut remarquer que les meilleurs vins résultent d'années de forte température.

Effectuez la même analyse à l'aide des boîtes à moustaches conditionnées

```
bwplot(~ A$TEMPERAT | A$QUALITE, xlab = "Temperature", col = "red")
bwplot (~ A$SOLEIL | A$QUALITE, xlab = "Soleil", col = "Yellow")
bwplot (~ A$CHALEUR | A$QUALITE, xlab = "Chaleur" , col = "green")
bwplot (~ A$PLUIE | A$QUALITE, xlab = "Pluie", col = "blue")
```

On peut interpréter l'influence de chaque variable sur la qualité du produit.

Par exemple : à l'aide du graphique sur la pluie, on peut remarquer que les meilleurs vins résultent d'années plutôt sèches.

Visualisez les relations entre couple de variables conditionnellement à la qualité. Conclure.

Par exemple : on remarque qu'une année peu pluvieuse jumelée à de forte chaleur fournira des vins de bonne qualité.