Quelques commandes R

R Version 1.9.0

Lancement de R

`R`	Lancement d'une session interactive (ou menu démarrer... sous windows)
`R --vanilla <` *file*	Lancement de R et execution des commandes contenues dans *file*
`R --help`	description des options de commande

Arrêt de R

`q()`	sortie de R
`INTERRUPT`	(*e.g.* `C-c` ou `esc`) arrêt de la commande en cours et retour au niveau principal

Aide

`help.start()`	démarrage de l'aide html
`help(`*command*`)`	aide en ligne de *command*
`apropos(`*command*`)`	recherche d'aide à propos de *command*
`help.search("`*command*`")`	recherche de *command* dans les packages R

Assignement

var `<-` *expr*	Assigne l'expression à la variable
var `=` *expr*	Assigne l'expression à la variable
expr `->` *var*	Assigne l'expression à la variable

Types élémentaires

`NA`	valeur manquante
`TRUE FALSE`	booleen (logique)
`numeric`	reel ou entier
`complex`	complexe
`character`	caractère

Caractères spéciaux

`\\`	backslash
`\n`	newline, ASCII code 10
`\t`	tabulation, ASCII code 9
`#`	commentaires (jusqu'à la fin de ligne)

Vecteurs

Les vecteurs ne sont pas des matrices et n'ont qu'1 dimension. Ils sont constitués d'un seul type avec ou non des NA (objet homogène)

`c(` x`,` y`, ... )`	saisie d'un vecteur
`rep(c(0,1),10)`	répétition d'un motif
`10:(-1)`	séquence de 10 à -1
`seq(1,20,by=0.5)`	séquence par pas
`seq(1,20,length=13)`	séquence par longueur finale
x`[c(6,1,9)]`	affichage des coordonnées 6, 1 et 9
x`[c(-6,-2)]`	affichage des coordonnées *autres que* 6 et 2
x`[c(TRUE,FALSE)]`	affichage de la coordonnée 1 et pas de la 2
x`[`x`>0]`	affichage des coordonnées positives

Arithmetique

x `+` y	addition (elt par elt)
x `-` y	soustraction (elt par elt)
x `%*%` y	multiplication matricielle
x `%o%` y	produit extrieur
x `*` y	multiplication (elt par elt)
x`^` y	élévation à la puissance (elt par elt)
x `/` y	division (elt par elt)
`-` x	moins unaire
x `%/%` y	partie entière de la division (elt par elt)
x `%%` y	reste de la division (elt par elt)

Autres opérations (numérique)

`c(`x`,`y`)`	concaténation de x et de y
`length(`x`)`	longueur de x
`min(`x`)`	minimum de x
`max(`x`)`	maximum de x
`range(`x`)`	étendue de x
`mean(`x`)`	moyenne de x
`median(`x`)`	médiane de x
`IQR(`x`)`	écart inter-quartile de x
`quantile(`x`)`	quantile de x
`var(`x`)`	variance de x
`sd(`x`)`	écart type de x
`rank(`x`)`	rang des éléments de x
`sort(`x`)`	éléments de x ordonnés
`order(`x`)`	coordonnées du plus petit élément de x, puis du 2eme plus petit...
`which.min(`x`)`	indice du minimum x
`which(`x`==min(`x`))`	indices des TRUE (indice du minimum)
`sum(`x`)`	somme (des éléments) de x
`prod(`x`)`	produit de x
`diff(`x`)`	différence des éléments consécutifs de x
`cumsum(`x`)`	somme cumulée de x

Autres opérations (charactères)

`paste(`x`,`y`)`	concaténation de x et y (elt par elt)
`substring(`x`,2,3)`	extraction du 2e et 3e caractère (elt par elt)

Facteurs

Les facteurs sont les mesures de variables qualitatives. Ils ont une longueur (le nombre de mesures) et des modalités (levels). Ce sont des vecteurs de caractères "spéciaux".

`factor(`x`)`	transforme x en facteur
`ordered(`x`)`	transforme x en facteur ordonné
`levels(`fa`)`	modalités de fa
`as.integer(`fa`)`	transforme fa en numérique
`as.character(`fa`)`	transforme fa en caractre

Opérations combinées avec des facteurs

`ave(`x`,`fa`)`	moyenne du vecteur x par modalité de fa
`split(`df`,`fa`)`	sépare le vecteur df par modalité de fa
`by(`x`,`fa`,median)`	applique la fonction `median` par modalité de fa au data-frame x.

Matrices

Les matrices possèdent 2 dimensions. Elles sont constituées d'un seul type avec ou non des NA (objet homogène).

`matrix(1:2,1,2)`	matrice à 1 ligne 2 colonnes
`matrix(0,10,20)`	matrice de 0 à 10 ligne 20 colonnes
`as.matrix(1:10)`	matrice unicolonne du vecteur 1:10
`diag(1:10)`	matrice diagonale d'ordre 10 dont les valeurs sont 1:10
`diag(10)`	identité d'ordre 10

Sélection et matrices

x`[c(6,1,9),]`	affichage des lignes 6, 1 et 9
x`[,c(6,1,9)]`	affichage des colonnes 6, 1 et 9
x`[1,-2]`	affichage de la ligne 1 sans la colonne 2
x`[`x`[,3]<2,]`	affichage des lignes dont la colonne 3 a une valeur inférieure à 2

Autres opérations (matrices)

`length(`x`)`	longueur de x, i.e. son nombre d'éléments
`diag(`x`)`	extraction de la diagonale de x
`nrow(`x`)`	nombre de lignes de x
`ncol(`x`)`	nombre de colonnes de x
`dim(`x`)`	dimension de x
`cbind(`x`,`y`)`	concaténation par colonne de x et$y$
`rbind(`x`,`y`)`	concaténation par ligne de x et$y$
`qr(`x`)`	décomposition QR de x
`chol(`x`)`	décomposition de Cholesky de x
`svd(`x`)`	décomposition en valeur singulière$x$
`eigen(`x`)`	valeurs propres et vecteurs propres de x
`det(`x`)`	déterminant de x
`solve(`x`)`	inversion de x
`apply(`x`,1,sum)`	somme par ligne de x
`apply(`x`,2,var)`	variance par colonne de x
`colMeans(`x`)`	moyenne par colonne de x
`rowSums(`x`)`	somme par ligne de x

Listes

Les listes sont des objets hétérogènes composés d'une collection d'objets homogènes de longueur pas obligatoirement identiques.

`list(`*nomx*`=`x`,`y`)`	listes à 2 éléments (ou composantes) x de nom *nomx* et y sans nom.
`as.list(1:10)`	liste à 10 éléments de longueur 1

Divers sur les listes

li`[[2]]`	extraction de la comp. 2 de la liste *li*
li`$`*nomx*	extraction de la comp. *nomx* de la liste *li*
`dimnames(`x`) <-` `list(`*nomli*`,`*nomco*`)`	noms des lignes et des colonnes de la matrice x.
`lapply(`*li*`,`f`)`	application de la fonction f à chacune des composantes de la liste
`unlist(`*li*`)`	transformation en vecteur de la liste

Data-frame

Les data-frames sont des objets hétérogènes composés d'une collection d'objets homogènes de longueur identique.

`data.frame(`x`)`	Converti x en data-frame.
`cbind.data.frame(`*df*`,`*df2*`)`	Concatène par colonne.
`names(`*df*`)`	nom des colonnes
df`[,1]`	extraction de la colonne 1
df`[[1]]`	extraction de la composante 1 (colonne 1 en général)
df`[,"`*nomcol*`"]`	extraction de la colonne *nomcol*
`summary(`*df*`)`	résumé colone par colonne (min,max,moyenne etc...)

Opérations sur les data-frame

`lapply(`*df*`,`*f*`)` application de la fonction *f* à chacune des composantes de la liste

`apply(`*df*`,1,`*f*`)` application de la fonction *f* à chacune des lignes

`aggregate(`*df*`,`*li*`,`*f*`)` application de la fonction *f* à chacune des modalités de l'intéraction des facteurs contenus dans la liste *li*

Objets : question et conversion

`is.factor(`*x*`)` renvoie un booleen; vrai si *x* est un facteur

`is.matrix(`*x*`)` de même avec une matrice

`is.vector(`*x*`)` de même avec vecteur

`as.factor(`*x*`)` conversion explicite en facteur

`as.matrix(`*x*`)` conversion explicite en matrice

`as.vector(`*x*`)` conversion explicite en vecteur

Entrée sortie

df `<- read.table("c:/doc.txt",header=T,sep=" ")`
chargement d'un fichier (nombreuses options)

`write.table(`*x*`,"c:/sortie.txt",row.names=F)`
exportation de l'objet *x* dans le fichier

Voir aussi `scan(), write()`

`source("c:/fichiercommandes.txt")`
importe et execute les commandes contenues dans le fichier

Programmation

`for (i in vecteur) { listecommandes }`
boucle sur tous les éléments du vecteur

`while (condition) { listecommandes }`
boucle tant que la condition est vrai

`repeat { listecommandes }`
boucle infinie (couplée avec l'ordre `break` pour sortir de la boucle)

```
if (condition) {
    listecommandes
 } else {
    listalternative }
```
Structure conditionnelle

Fonction

```
fonction <- function(a,b=1) {
    listecommandes
    return(nom1=resultat1,nom2=resultat2) }
```
Fonction avec 2 arguments en entrée `a` et `b`. `b` a comme valeur par défaut 1 renvoyant une liste à deux composantes (nommées nom1 et nom2).

`fonction(3,1)` ou `fonction(3)`

`res <- fonction(3,2)`
appel de la fonction

Lois de probilités et nombres aléatoires

Pour avoir un quantile utiliser le préfixe `q`. Pour avoir une probabilité utiliser le préfixe `p`, pour avoir une densité utiliser le préfixe `d` et pour générer des nombres aléatoire utiliser le préfixe `r`. Lois diponibles : norm, poiss, binom, logis,...

`qt(0.95,12)` quantile à 95% d'une loi de student (t) (préfixe `q`) à 12 ddl

`ppoiss(3,2)` probabilité cumulée jusqu'à 3 d'une loi de poisson (poiss) (préfixe `p`)

`df(3,6,30)` densité au point 3 d'une loi de fisher (f) (préfixe `d`) à (6,30) ddl

`rnorm(10,1,2)` génération de 10 nombres aléatoires (préfixe `r`) selon une loi normale (norm) de moyenne 1 d'écart type 2

`sample(10,1:30)` tirage sans remise de 10 nombres parmi les coordonnées du vecteur `1:30`

Formules

`y~x1+x2` `y` expliqué par `x1` et `x2`

`y~x1+x2+x1:x2` `y` expliqué par `x1` et `x2` et leur intéraction

`y~x1*x2` raccourci de la précédente

`y~x1%in%x2` `y` expliqué par `x1` nidé dans `x2`

`y~x1|x2` `y` expliqué par `x1` conditionnellement à `x2`

`y~-1+x1` `y` expliqué par `x1` sans moyenne générale (intercept)

`y~I(x1*x2)` `y` expliqué par la variable résultant du produit `x1x2` ; `I()` protège pour éviter l'interprétation comme `x1+x2+x1:x2`

`y~I(x1>0)` `y` expliqué par la variable binaire 0 quand `x1` est négatif ou nul, 1 sinon

`y~exp(x1*1932)` `y` expliqué par la variable `x1` multiplié par 1932 et dont on en prend l'exponentielle

Graphiques

`plot(`*x*`,`*y*`-options-)` dessin des points aux abscisses contenues dans *x* et aux ordonnées dans *y* (voir les options ci-dessous et l'aide)

`matplot(`*x*`-options-)` dessin des colonnes de la matrice (voir les options ci-dessous et l'aide)

`coplot(y~x1|x2, data=`*x*`-options-)` dessin de `y` fonction de `x1` conditionnellement à `x2` (voir les options ci-dessous et l'aide)

`,lty=1 2 ...` type de ligne (pointillé, continue...) -option-

`,pch=1 2 ...` ou `pch="o"` type de point (rond...) -option-

`,col=0 1 2 ...` numéro de couleur-option-

`,type="l" "p" "b" "n"` tracé de ligne ou point ou les deux (both) ou rien (none)-option-

`abline(v=0)` ajoute une ligne verticale en 0

`abline(h=0)` aoute une ligne horizontale en 0

`abline(0,1)` ajoute une droite d'ordonnée à l'origine 0 et de coef. directeur 1

`text(1,1,"blabla")` ajoute le texte "blabla" en (1,1)

`lines(`*x*`,`*y*`,col=2)` ajout d'une ligne (brisée) de coordonnées (x,y) de couleur 2

Voir aussi `interaction.plot, stars, pairs, piechart, mtext, symbols,...`

Analyses statistiques

`lm() aov() anova() glm() rpart() nlme() nls()`
modèle linéaire, analyse de variance, modèle linéaire généralisé, arbres régression/segmentation, modèles mixtes, régression non-linéaire selon les librairies

```
resmodele <- lm(y~x1+x2+x1:x2,data=df)
summary(resmodele)
residuals(resmodele)
predict(resmodele)
plot(resmodele)
```
exemple d'une régression linéaire, explication de `y` par `x1` `x2` et leur intéraction. Les données sont contenues dans le data-frame *df*. Vecteurs des résidus, de l'estimation de `y` par le modèle et dessin du modèle.

`chisq.test(), fisher.test(), friedman.test(), mantelhaen.test(), t.test(),...`

Librairies - packages

De nombreux programmes sont disponibles sur `http://cran.r-project.org` et sur ses sites mirroirs.

`install.packages(tree)` Installe le package `tree` ; attention aux droits d'écriture sur votre disque...

`library()` liste des packages disponibles sur votre plateforme

`library(splines)` Chargement du package `splines`

Chemin

R cherche tout objet dans des répertoires donnés dans un ordre donné. Cette liste ordonnée est appelée un chemin (path).

`search()` Liste du chemin

`attach(`*df*`,pos=2)` attache le data frame *df* en position 2 sur le chemin.

`detach(`*df*`)` détache le data frame *df*.

Liste non exhaustive pour la programmation

`assign(`*x*`,pos=1)` assigne l'objet *x* dans l'environnement 1 du chemin

`parse(`*texte*`)` transforme le texte en expression (non évaluée)

`eval(`*expr*`)` évalue l'expression ; par exemple `eval(parse(text=`*texte*`))`

`deparse(`*expr*`)` transforme une expression en texte

`substitute(`*expr*`)` susbtitue dans une expression les valeurs des variables ; par exemple `deparse(substitute(`*x*`))`

`browser()` stoppe l'execution et permet de fureter, utile pour debogguer

---

Edition 0.3 for R Version 1.9.0. 2003, P.A. Cornillon (pac@uhb.fr). Comme d'habitude, "the author assumes no responsibility for any errors on this card" et l'aide en ligne *a toujours raison*.

Roland Pesch (pesch@cygnus.com) a codé les macros TEX, originellement pour la carte de reference GDB et John W. Eaton (jwe@che.utexas.edu) les a modifiées pour Octave.