

Carte de référence pour R

Document original de Tom Short, disponible sur www.Rpad.org. Traduction et modifications par Vincent Chouraki. Document sous GNU GPL. Pour la dernière version et le code source, voir www.santepub-lille.fr. Version du 18 avril 2008.

“Fonctions vitales”

La plupart des fonctions R ont une documentation en ligne. `help(fonction)` documentation sur `fonction`
`?fonction` idem
`help.search("thème")` effectue une recherche dans l'aide
`apropos("mot.clé")` retourne les fonctions dont le nom contient "mot.clé"
`help.start()` démarre l'aide au format HTML
`getwd()` retourne le chemin du répertoire de travail ((wd) pour *w*orking *d*irectory)
`setwd("chemin")` permet de modifier le chemin du répertoire de travail
`list.files()` retourne les noms des fichiers du répertoire de travail
en début de ligne pour faire des commentaires (c.-à-d. que le code n'est pas interprété)
`q()` pour quitter R
http://blogs.nolife-tv.com/temps_perdu en cas de surchauffe du cerveau, mieux vaut faire une petite pause ☺

Entrées et sorties

`load()` charge les données écrites avec la commande `save`
`data(x)` charge le jeu de données `x`; `data()` affiche la liste des jeux de données disponibles
`library(x)` charge le package `x`
`read.table("fichier")` crée une data frame à partir des données d'une table; le séparateur de champ par défaut est `sep=" "` c.-à-d. n'importe quel espace; `header=TRUE` précise que la première ligne de la base contient les noms des variables; `as.is=TRUE` empêche la conversion automatique des vecteurs caractères en facteurs; `comment.char=""` empêche "#" d'être interprété comme un commentaire; `skip=n` permet de sauter `n` lignes avant de commencer la lecture des données; pour le reste, ?`data.frame` ☺
`read.csv("fichier",header=TRUE)` idem mais réglé par défaut pour les fichiers .csv (`sep=","`); voir ?`read.csv2` (`sep=";"`)
`read.delim("fichier",header=TRUE)` idem mais réglé par défaut pour les fichiers où le séparateur est une tabulation (`sep="\t"`)
`save(...,file)` sauvegarde les objets spécifiés (...) au format R
`save.image(file)` sauvegarde tous les objets
`cat(...,file=" ",sep=" ")` concatène les arguments après conversion en chaîne de caractères; `sep` précise le caractère séparant les arguments
`print(a, ...)` affiche les arguments; fonction générique pouvant avoir plusieurs méthodes en fonction de la classe des objets
`format(x,...)` formate un objet R pour un affichage plus sympa
`write.table(x,file=" ",row.names=TRUE,col.names=TRUE,sep=" ")` sauvegarde `x` sous forme de table dans un fichier; voir

les options pour spécifier le séparateur de champs, le codage des données manquantes, etc.
`sink("fichier")` envoie les sorties dans `fichier`; `sink()` pour arrêter la procédure
La plupart des fonctions E/S ont un argument `file`. Celui-ci est souvent une chaîne de caractères indiquant le chemin d'un fichier ou d'une connexion (url, pipes, zip,...). Voir ?`connection` pour plus d'informations.
Sous windows™, on peut aussi utiliser `file = "clipboard"` pour lire et écrire dans le presse-papier. Par exemple, pour lire une table copiée depuis Excel :
`x <- read.delim("clipboard")`
Pour envoyer une table vers le presse-papier pour copie dans Excel :
`write.table(x,"clipboard",sep="\t",col.names=NA)`
Pour utiliser d'autres types de base de données, voir les packages RODBC, DBI, RMySQL, RPostgreSQL, and ROracle. Voir les packages XML, hdf5, netCDF pour lire d'autres formats de fichiers.

Création de données

`c(...)` fonction générique pour combiner des éléments, par défaut sous forme de vecteur; avec `recursive=TRUE`, permet d'assembler tous les éléments de listes en un vecteur
`from:to` génère une séquence; ":" est prioritaire dans les calculs; `1:4 + 1` donne "2,3,4,5"
`seq(from,to)` génère une séquence; `by=` précise le pas; `length=` précise la longueur de la séquence
`seq(along=x)` génère 1, 2, ..., `length(along)`; utile dans les boucles `for`; identique à `1:length(x)`
`rep(x,times)` répète `x` "times" fois; `each=` permet de répéter chaque élément de `x` "each" fois; `rep(c(1,2,3),2)` donne 1 2 3 1 2 3; `rep(c(1,2,3),each=2)` donne 1 1 2 2 3 3
`factor(x,levels=)` transforme un vecteur `x` en facteur (syn. variable catégorielle)
`data.frame(...)` crée une data frame à partir des arguments; p.ex. `data.frame(v=1:4,ch=c("a","B","c","d"),n=10)`; les vecteurs plus petit sont réutilisés ("recyclés") pour correspondre à la longueur du plus grand vecteur
`list(...)` crée une liste à partir des arguments; `list(a=c(1,2),b="hi",c=3i)`
`array(x,dim=)` crée un array à partir de `x`; `dim=` pour préciser les dimensions, p.ex. `dim=c(3,4,2)`; les éléments de `x` sont recyclés si `x` n'est pas assez long is not long enough
`matrix(x,nrow=,ncol=)` matrice, voir `array`
`rbind(...)` combine les arguments (matrices, data frames,...) selon les lignes (r pour "row", ligne en anglais)
`cbind(...)` idem par *c*olonnes

Conversion de variables

`as.array(x)`, `as.data.frame(x)`, `as.numeric(x)`, `as.logical(x)`, `as.complex(x)`, `as.character(x)`, ... conversion selon le type précisé; `methods(as)` pour obtenir la liste complète des conversions possibles

Informations sur les variables

`str(a)` affiche la *str*ucture interne d'un objet R
`summary(a)` retourne un sommaire de `a`, généralement constitué de quelques indicateurs statistiques (moyenne, quantiles, range et nombre de données manquantes) mais cela peut varier en

fonction de la classe de `a`
`ls()` affiche les objets présents dans l'environnement de travail; précisez `pat="pat"` pour affiner la recherche
`ls.str()` `str()` pour chaque variable de l'environnement de travail
`is.na(x)`, `is.null(x)`, `is.array(x)`, `is.data.frame(x)`, `is.numeric(x)`, `is.complex(x)`, `is.character(x)`, ... teste si l'objet est du type précisé; voir `methods(is)` pour la liste complète des tests possibles
`length(x)` nombre d'éléments de `x`
`dim(x)` retourne ou permet de définir les dimensions d'un objet; `dim(x) <- c(3,2)`
`dimnames(x)` retourne ou permet de définir les noms des dimensions d'un objet
`nrow(x)` nombre de lignes; `ncol(x)` nombre de colonnes
`class(x)` retourne ou permet de définir la classe de `x`; `class(x) <- "myclass"`
`unclass(x)` efface l'attribut "classe" de `x`
`attr(x,which)` retourne ou permet de définir l'attribut `which` de `x`
`attributes(objet)` retourne ou permet de régler l'ensemble des attributs de objet

Indexation des données

Indexation des vecteurs
`x[n]` n^e élément
`x[-n]` tout *sauf* le n^e élément
`x[1:n]` n 1^{ers} éléments
`x[-(1:n)]` n+1^e élément jusqu'au dernier
`x[c(1,4,2)]` éléments spécifiques
`x["name"]` élément nommé "name"
`x[x > 3]` éléments supérieurs à 3
`x[x > 3 & x < 5]` éléments compris entre 3 et 5
`x[x %in% c("a","and","the")]` éléments appartenant à la liste

Indexation des listes
`x[[n]]` n^e élément de la liste
`x[["name"]]` élément de la liste nommé "name"
`x$name` idem

Indexation des matrices
`x[i,j]` élément de la i^e ligne, j^e colonne
`x[i,]` i^e ligne
`x[,j]` j^e colonne
`x[,c(1,3)]` colonnes 1 et 3
`x["name",]` ligne nommée "name"

Indexation des data frames (idem ci-dessus plus ci-dessous)
`x[["name"]]` colonne nommée "name"
`x$name` idem

Manipulation et sélection des données

`which.max(x)` renvoie la position du maximum de `x`
`which.min(x)` renvoie la position du minimum de `x`
`rev(x)` inverse l'ordre des éléments de `x`
`sort(x)` trie les éléments de `x` par ordre croissant; `rev(sort(x))` pour un tri décroissant

cut(x,breaks) divise **x** en intervalles précisés par **breaks** (nombre d'intervalles ou valeurs des points de coupe); **x** est converti en facteur

match(x, y) retourne un vecteur de même longueur que **x** contenant les indices des éléments de **y** retrouvés dans **x**, **NA** sinon

which(x == a) retourne un vecteur des positions de **x** pour lesquelles la condition précisée est remplie (**TRUE**); l'argument de cette fonction est toujours de type logique

na.omit(x) supprime les observations manquantes (**NA**) d'un objet (supprime toute la ligne si **x** est une matrice ou une data frame)

na.fail(x) retourne un message d'erreur si **x** contient au moins un **NA**

unique(x) si **x** est un vecteur ou une data frame, retourne un objet similaire sans les éléments dupliqués

table(x) retourne un tableau d'effectifs des différentes valeurs de (**x**) (adapté aux variables **integer** ou **factor**)

subset(x, ...) retourne un sousgroupe de **x** choisi selon les critères précisés par les options **subset=** et **select=**

sample(x, size) réalise un échantillonnage aléatoire de "size" éléments de **x**, avec ou sans remise (option **replace=TRUE** ou **replace=FALSE** respectivement)

prop.table(x,margin=) retourne un tableau de proportion de **x**; **x** doit être un objet de classe **table**; **margin=** permet de choisir le type de pourcentages –lignes, colonnes ou cellules

Mathématiques

sin, cos, tan, asin, acos, atan, atan2, log, log10, exp

max(x) maximum de **x**

min(x) minimum de **x**

range(x) identique à **c(min(x), max(x))**

sum(x) somme des éléments de **x**

diff(x) différences entre les éléments de **x**

prod(x) produit des éléments de **x**

mean(x) moyenne arithmétique de **x**

median(x) médiane de **x**

quantile(x,probs=) quantiles de l'échantillon (par défaut 0,.25,.5,.75,1)

weighted.mean(x, w) moyenne pondérée (**w**) de **x**

rank(x) rangs des éléments de **x**

var(x) ou **cov(x)** variance de **x**, calculée sur $n - 1$; si **x** est une matrice ou une data frame, la fonction retourne la matrice de variance-covariance

sd(x) déviation standard de **x**

cor(x) matrice de corrélation de **x** (matrice ou data frame); 1 si **x** est un vecteur

var(x, y) ou **cov(x, y)** covariance entre **x** et **y**, ou entre les colonnes de **x** et de **y** si ce sont des matrices ou des data frames

cor(x, y) corrélation linéaire entre **x** et **y**, ou matrice de corrélation en cas de matrices ou data frames

round(x, n) arrondit les éléments de **x** à **n** décimales

log(x, base) calcule le logarithme de base "base" de **x**

scale(x) si **x** est une matrice, centre et réduit les données

pmin(x,y,...) retourne un vecteur dont le i^{e} élément est le minimum de **x[i]**, **y[i]**, ...

pmax(x,y,...) idem pour le maximum

cumsum(x) retourne un vecteur dont le i^{e} élément est la somme de **x[1]** à **x[i]**

cumprod(x) idem pour le produit

cummin(x) idem pour le minimum

cummax(x) idem pour le maximum

union(x,y), **intersect(x,y)**, **setdiff(x,y)**, **setequal(x,y)**, **is.element(el,set)** fonctions "set"

Beaucoup de fonctions mathématique ont un paramètre logique permettant de retirer les données manquantes du calcul (**na.rm=TRUE**)

Chaines de caractères

paste(...) concatène les vecteurs après conversion en caractères; **sep=** permet de préciser le caractère séparant les arguments (espace simple par défaut); voir aussi l'option **collapse**

substr(x,start,stop) extrait une partie d'un vecteur "chaîne de caractères"; permet aussi de faire des remplacements, p.ex. **substr(x, start, stop) <- "blabla"**

strsplit(x,split) sépare **x** selon **split**

grep(pattern,x) recherche la séquence **pattern** dans **x**; voir **?regex**

gsub(pattern,replacement,x) permet de remplacer des séquences obtenues par expressions rationnelles; voir également **sub()**

tolower(x) conversion en minuscules

toupper(x) conversion en majuscules

match(x,table) retourne un vecteur de même longueur que **x**, contenant les indices des éléments de **table** retrouvés dans **x**

x %in% table idem mais retourne un vecteur logique

nchar(x) nombre de caractères

Graphiques

plot(x) graphe des valeurs de **x** (sur l'axe **y**) ordonnées sur l'axe **x**

plot(x, y) graphe "classique" de **x** (abscisse) et **y** (ordonnée)

hist(x) histogramme de **x** (effectifs par défaut)

barplot(x) diagramme en barres de **x**

piechart(x) diagramme "camembert"

boxplot(x) boxplot...

coplot(x~y | z) graphes de **x** et **y** pour chaque valeur de **z**

pairs(x) si **x** est une matrice ou une data frame, affiche tous les graphes bivariés possibles entre les colonnes de **x**

qqnorm(x) diagramme quantiles-quantiles de **x** et des valeurs attendues sous une loi Normale

Voir **library(help=graphics)** et **library(help=lattice)** pour plus de fonctions graphiques

Les paramètres suivants se retrouvent dans beaucoup de fonctions graphiques :

add=FALSE si **TRUE** superpose le nouveau graphe sur le précédent (s'il existe)

axes=TRUE si **FALSE** les axes et le cadre ne sont pas représentés

type="p" précise le type de représentation des données, "p" : points, "l" : lignes, "b" : points connectés par des lignes, "o" : idem avec superposition des lignes et des points, "h" : lignes verticales (comme un *h*istogramme), "s" : marches (*s*teps), etc.

xlim=, ylim= précise les limites basses et hautes des axes, p.ex. avec **xlim=c(1, 10)** ou **xlim=range(x)**

xlab=, ylab= légende pour les axes, sous forme de chaîne de caractères; p.ex. "blabla"

main= titre principal

sub= sous-titre

voir également **?par** pour plus de paramètres graphiques

Un peu de statistiques

Quelques tests

chisq.test() test du χ^2

fisher.test() test exact de Fisher

t.test() test t de comparaison de moyennes (appariées ou non)

bartlett.test() test d'homogénéité des variances

Voir également **apropos("test")** et **help.search("test")**

Distributions

rnorm(n, mean=0, sd=1) génère **n** valeurs suivant une distribution Normale; voir **?Normal** pour plus de fonctions utilisant la loi Normale

Voir **?Chisquare**, **?Uniform**, etc. pour d'autres distributions

Modélisation

lm(formula) modèle linéaire; **formula** suit une syntaxe standardisée, de la forme **var.explicitee ~ var.explicatives + var.ajustement + ...**; voir **?formula** pour une explication plus détaillée

glm(formula,family=) modèle linéaire généralisé; **family** précise la distribution des résidus et la fonction de lien utilisée dans le modèle; voir **?family** pour plus de détails

Essayer également **summary()** et **names()** sur ces objets ☺

Manipulation avancée des données

apply(X,INDEX,FUN=) retourne un vecteur, matrice ou array de valeurs obtenues par l'application de la fonction **FUN** sur les colonnes ou lignes (**INDEX**) de **X**

lapply(X,FUN) idem pour les éléments des data frame ou listes

Voir **apropos("apply")** pour plus de fonctions

by(data,INDEX,FUN) applique **FUN** à la data frame **data** selon chaque modalité de **INDEX**

merge(a,b) fusionne deux data frames selon les noms de colonnes ou lignes communes (pratique pour les jointures de tables!!!)

aggregate(x,by,FUN) applique la fonction **FUN** à des sous-groupes de **x**, précisés par l'argument **by**, p.ex. une liste de variables

Voir également les fonctions **stack()** et **reshape()**

Programmation

function(arglist) expr permet de définir de nouvelles fonctions

return(valeur)

if(condition) expressions

if(condition) expression else expression alternative

for(var in seq) expression

Les expressions doivent être encadrées par **{}**

ifelse(test, yes, no) retourne le résultat de l'expression **yes** si le test renvoie **TRUE**, **no** si le test renvoie **FALSE**

do.call(funname, args) exécute un appel à la fonction à partir de son nom **funname** et d'une liste d'arguments qui lui est transmise **args**