Commandes usuelles de R

J. Chiquet, octobre 2015

Cette liste de commandes (non exhaustive) est une adaptation de la ref-card R de Tom Short. Elle est un relais entre ce que cherche l'utilisateur et l'aide de R : les nombreuses options disponibles ne sont que rarement spécifiées ici.

Aide et fonctions de base

`help(topic)`,`?topic` affiche l'aide relative à `topic`
`apropos("topic")`,`??topic` recherche par mot-clé (version courte)
`help.search("topic")`,`???topic` recherche par mot-clé (version longue)
`help.start()` lance la version HTML de l'aide
`str(a)` affiche la structure de l'objet `a`
`head(a)` affiche les premiers éléments de l'objet `a` (selon son type: vecteur, matrice, tableau, etc.)
`summary(a)` propose un «résumé» de `a`, la plupart du temps un résumé statistique
`search()` affiche l'itinéraire de recherche
`ls()` affiche tous les objets présents dans l'itinéraire de recherche
`ls.str()` applique `str()` à variable présente dans l'itinéraire de recherche
`dir()` affiche les fichiers présents dans le répertoire courant
`library(x)` charge la bibliothèque `x`
`attach(x)` place l'objet `x` dans l'itinéraire de recherche; `x` peut être une liste, un tableau de données ou un objet créé à l'aide de la fonction `save`
`detach(x)` ôte l'objet `x` de l'itinéraire de recherche
`with(x, expr)` évalue la commande `expr` en ayant placé l'objet `x` dans l'itinéraire de recherche.
`rm(x)`, `remove(x)` détruit l'objet `x`
`setwd(dir)`, `getwd(dir)` affecte ou récupère le chemin du répertoire de travail courant
`function( arglist ) { expr return(result)}` définition de fonction
`if, while, repeat, etc.` voir `help(if")`

Entrées / Sorties

`save(file,x,y)` enregistre les objets `x,y` dans le fichier binaire `file`
`save.image(file)` enregistre tous les objets de la session
`load(file)` charge un objet précédemment enregistré à l'aide de `save`
`data(x)` charge le jeu de données `x`
`read.table(file)`, `read.csv`, `read.delim` lit un fichier stocké sous la forme d'un tableau et crée un objet `data.frame`; le séparateur par défaut est le caractère espace pour `read.table`, la virgule ou le point virgule pour `read.csv`, la tabulation pour `read.delim`; utiliser l'option `header=TRUE` pour que la première ligne soit considérée comme définissant le nom des colonnes
`cat(...)` fonction d'impression bas niveau
`print(a)` fonction d'affichage d'un objet `a` s'adaptant au type de l'objet
`format(x)` formatage d'un objet
`write.table(x)` imprime `x` après conversion en type `data.frame`

Variables réservées

`NULL` l'objet nul (objet réservé)
`NA` absense de données/valeur manquante
`TRUE/FALSE` vrai et faux logiques
`Inf` valeur infinie

Création de données

`vector(mode, size)` initialise un vecteur de mode `mode` de taille `size`
`logical(size)`, `numeric(size)`, `double(size)`, `character(size)` spécialisation de `code` aux modes élémentaires.
`c(nom1=, nom2=, ...)` fonction générique combinant une suite d'éléments en un vecteur (possibilité d'attribuer des noms)
`from:to` génère une séquence; priorité de l'opérateur «:» `1:4 + 1` vaut `2,3,4,5`
`seq(from,to)` génère une séquence; `by=` et `length=` spécifient l'incrément et/ou la longueur.
`seq_along`, `seq.int`seq variante de `seq`
`rep(x,times)` répète `x` un nombre `times` de fois; utiliser `each=` pour répéter chaque élément `each` fois; `each` peut être un vecteur
`rep.int`, `rep_len` variantes de `rep`
`data.frame(...)` crée un tableau de données; les vecteurs courts sont répétés jusqu'à correspondre à la taille des vecteurs les plus longs
`list(...)` crée une liste
`vector(''list'', size)` crée une liste de taille `size`
`array(x,dim=)` crée un tableau multidimensionnel `x`; les éléments de `x` sont répétés si la taille ne correspond pas aux dimensions spécifiées
`matrix(x,nrow=,ncol=)` crée une matrice; les éléments de `x` sont répétés si la taille ne convient pas
`factor(x,levels=)` crée un vecteur de facteurs
`expand.grid()` génère un tableau de données contenant les combinaisons des vecteurs spécifiés en arguments
`rbind()`,`cbind()` pour combiner les éléments d'un objet par ligne et par colonne

Extraction de données

Indexation des listes

`x[n]`	une liste avec les éléments de `n`
`x[[n]]`	le n^e élément de la liste
`x$name`,`x[["name"]]`	l'élément `"name"`

Indexation des vecteurs

`x[n]`	n^e élément du vecteur
`x[-n]`	tous les éléments sauf le n^e
`x[1:n]`	`n` premiers éléments
`x[-(1:n)]`	tous les éléments sauf les `n` premiers
`x[c(1,4,2)]`	éléments 1,4 et 2
`x["name"]`	élément(s) de nom `"name"`
`x[x > 3]`	tous les éléments plus grands que 3
`x[x > 3 & x < 5]`	tous les éléments compris entre 3 et 5

Indexation des matrices

`x[i,j]`	élément de la i^e ligne et j^e colonne
`x[i,]`	i^e ligne
`x[,j]`	j^e colonne
`x[,c(1,3)]`	colonnes 1 et 3
`x["name",]`	lignes intitulées `"name"`
`x[rowSums(x)>10,]`	lignes dont la somme est supérieure à 10

Variables et attributs

`as.array(x)`, `as.data.frame(x)`, `as.numeric(x)`, `as.logical(x)`, `as.character(x)`, ... conversion de type
`is.na(x)`, `is.null(x)`, `is.array(x)`, `is.data.frame(x)`, `is.numeric(x)`, `is.character(x)`, ... teste le type d'un objet

Les fonctions suivantes s'utilisent pour récupérer ou spécifier un attribut.

`length(x)` nombre d'éléments de `x`
`dim(x)` nombre de dimensions d'un objet
`dimnames(x)` noms des dimensions d'un objet
`names(x)` manipulation de l'attribut `names` de l'objet `x`
`setNames(noms, x)` attribue le vecteurs de noms `noms` au vecteur `x`
`nrow(x), ncol(x)` nombre de lignes et de colonnes
`class(x)` classe de l'objet `x`
`unclass(x)` supprime l'attribut `class` de la variable `x`
`attr(x,which)` récupère ou spécifie les attributs de `x` décrits par `which`
`attributes(x)` récupère ou spécifie tous les attributs de `x`

Manipulation et sélection de données

`which.max(x)`,`which.min(x)` retourne l'indice du plus grand (resp. plus petit) élément de `x`
`rev(x)` inverse l'ordre des éléments `x`
`sort(x)` ordonne les éléments de `x` par ordre croissant
`cut(x,breaks)` découpe `x` en intervalles définis par `breaks`
`split(x,index)` renvoie une liste découpant `x` selon le facteur `index`
`unlist(l, recursive)` mise à plat de la liste `l` (récursivement par défaut)
`match(x, y)` renvoie un vecteur de la même taille que `x` dont l'élément i vaut `x[i]` si `x[i]` appartient à `y` et `NA` sinon
`which(x==a)` renvoie les indices des éléments de `x` vérifiant `x==a`
`choose(n, k)` calcule les combinaisons de k éléments parmi n
`tabulate(x,nbin=length(x)` compte les occurennces de tous les entiers jusqu'à `nbin` de `x`
`table(x` généralisation de `tabulate` à des facteurs et tableaux de données
`na.omit(x)` supprime les observations manquantes (notées `NA`)
`na.fail(x)` renvoie une erreur si `x` contient au moins un `NA`
`any(x)` teste si `x` contient au moins un élément `TRUE`
`anyNA(x)` teste si `x` contientau moins un élément `NA`
`unique(x)` supprime les doublons d'un vecteur ou d'un tableau
`table(x)` renvoie un tableau avec le nombre des différentes valeurs
`subset(x, ...)` renvoie un sous ensemble de `x` défini par `...`
`levels(f)`, `nlevels(f)`, `is.ordered(f)` manipulations des niveaux du facteur `f`
`sample(x, size)` crée un échantillon aléatoire de taille `size` parmi les éléments de `x`

Mathématiques

`abs`, `sqrt`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, `log`, `log10`, `exp`, `%%`, `%\%`, `exp`... fonctions mathématiques élémentaires
`max(x)`, `min(x)`, `range(x)`, `sum(x)`, `diff(x)`, `prod(x)`, `mean(x)`, `median(x)`, `sd(x)` maximum, minimum, amplitude, somme, différences, produit, moyenne, médiane, écart-type
`quantile(x,probs=)` fractiles des éléments de `x`
`weighted.mean(x, w)` moyenne de `x` pondérée par `w`
`var(x), cov(x)` variance empirique corrigée; si `x` est une matrice, renvoie la matrice de variance-covariance
`cor(x)` matrice de corrélations de `x`
`var(x, y), cov(x, y)` covariance entre `x` et `y`, ou entre les colonnes de `x` et de `y` si ce sont des matrices ou des tableaux
`cor(x, y)` idem pour la corrélation linéaire
`round(x, n)` arrondit les éléments de `x` à `n` décimales
`floor(x)`, `ceiling(x)` arrondissent à l'entier relatif supérieur ou inférieur
`scale(x)` centre et réduit les données `x`; pour centrer et/ou réduire uniquement, utiliser les `scale` et/ou `center`
`pmin(x,y,...), pmax(x,y,...)` un vecteur dont le i^{e} élément est le minimum (resp. maximum) entre `x[i]` et `y[i]`
`cumsum(x)` un vecteur dont le i^{e} élément est la somme des i premiers éléments de `x`
`cumprod(x), cummin(x), cummax(x)` idem pour le produit, le min, le max
`union(x,y), intersect(x,y), setdiff(x,y), setequal(x,y), all.equal(x,y)` et `is.element(el,set)` fonctions de définition d'ensembles
`Re(x), Im(x), Mod(x),Arg(x), Conj(x)` partie réelle, partie imaginaire, module, argument et conjugué d'un nombre complexe
`convolve(x,y)` calcule de convolution entre deux séquences
`fft(x)`, `mvfft(x)` transformation de Fourier d'une matrice, resp des colonnes d'une matrice
`filter(x,filter)` application d'un filtre linéaire à chaque élément d'une suite `x`

Matrices

`rowSums(x), colSums(x), rowMeans(x), colMeans(x)` somme et moyenne de chaque ligne, resp. chaque colonne de `x`
`t(x)` transposée de `x`
`diag(x)` renvoie ou spécifie la diagonale de `x`
`upper.tri(A), lower.tri(A)` selection du triangle supérieur/inférieur de `A`
`%*%` multiplication matricielle
`crossprod(x,y), t(x)%*% y` produit scalaire de `x` par `y`
`det(x)` déterminant de `x`
`svd(x)` décomposition en valeurs singulières
`eigen(x)` diagonalisation d'une matrice
`chol(x)` décomposition de Cholesky
`qr(x)` décomposition QR
`solve(a,b)` résout `a %*% x = b`
`solve(a)` calcule l'inverse de `a`
`chol2inv(x)` Inversion à partir d'une décomposition de Cholesky

Quelques fonctions du package `Matrix`
`Matrix(x, sparse=)` définition d'un objet de class matrice
`sparseMatrix(i, j, p, x=)` définition d'un matrice creuse
`bdiag(...)` création d'une matrice diagonal par blocs à partir d'une série de matrices
`bandSparse(...)` matrice creuse définie par ses termes super/sous diagonales.
`Diagonal(n, x=)` création d'une matrice diagonale creuse

Les objets de type `Matrix` possèdent les méthodes associées aux factorisations et décomposition usuelles (SVD, Cholesky, QR)

Traitements avancés

`apply(x,INDEX,FUN=)` renvoie un vecteur ou une liste de valeurs obtenues en appliquant la fonction `FUN` aux éléments de la dimension `INDEX` de `x`
`lapply(x,FUN)` applique `FUN` aux éléments d'une liste
`sapply(x,FUN)` applique `FUN` aux éléments d'une liste et simplifie la sortie
`tapply(x,INDEX,FUN=)` applique `FUN` à chaque groupe du tableau `X` défini par les indices `INDEX`
`rowsum(x,INDEX)` spécialisation de tapply pour la fonction `sum` (très performant)
`by(data,INDEX,FUN)` applique `FUN` au tableau de données `data` découpé via `INDEX`
`ave(x,...,FUN)` applique `FUN` à chaque sous-ensemble de `x` définis par des facteurs
`merge(a,b)` fusion de deux tableaux de données portant les mêmes noms de ligne ou de colonne

`aggregate(x,by,FUN)` découpe le tableau `x` en sous-ensembles auxquels sont appliqués la fonction `FUN` et renvoie le résultat; `by` est une liste définissant les sous-ensembles de `x`

`stack(x, ...)`, `unstack(x, ...)` transforme un tableau ou une liste `x` en un vecteur colonne, et réciproquement

`combn(x, m, func)` applique la fonction `func` à toutes les combinaisons de `m` éléments parmi les élément de `x`

`replicate(n, expr, ...` repète une opération faisant intervenir de l'aléa et renvoie un tableau multidimensionnel résultant de ces opérations

`do.call(func, list)` appelle la fonction `func` qu'elle applique aux arguments défini par les élements de `list`.

Chaînes de caractères

`paste(...)` concaténation de vecteurs après conversion en caractères

`substr(x,start,stop)` extraction ou spécification d'une sous-chaîne de `x`

`strsplit(x,split)` découpe `x` selon la sous-chaîne `split`

`grep(pattern,x)` recherche le motif `pattern` dans la chaîne `x`

`tolower(x)`, `toupper(x)` conversion en minuscules, resp. en majuscules

`match(x,table)` un vecteur renvoyant les positions où les éléments de `x` ont été pour la première fois rencontrés dans `table`

`x %in% table` identique, mais renvoie un vecteur de booléens

`pmatch(x,table)` appariement partiel des éléments de `x` parmi `table`

`nchar(x)` nombre de caractères de `x`

Graphiques et figures

`x11()`, `windows()` ouvre une nouvelle fenêtre graphique

`pdf()`, `png()`, `jpeg()`, `bitmap()`, `xfig()`, `pictex()`, `postscript()` pilote graphique produisant des sorties dans des fichiers plutôt qu'à l'écran

`dev.off()` ferme le pilote de sortie graphique pour clore le fichier de sortie

Commandes graphiques haut niveau

`plot(x)` trace les valeurs contenues dans `x` sur l'axe des *y*; s'adapte à la classe de l'objet `x`

`plot(x, y)` graphe bivarié (`x` sur l'axe des *x*, `y` sur l'axe des *y*)

`hist(x)` histogramme des fréquences de `x`

`curve(expr)` trace la fonction définie par l'expression `expr`

`barplot(x)` histogramme des valeurs de `x`

`dotchart(x)` si `x` est un tableau de données, trace les données par nuages de points groupés par ligne en ordonnées puis par colonne en abscisses

`pie(x)` graphe en camembert

`boxplot(x)` boîte à moustaches

`interaction.plot (f1, f2, y)` si `f1` et `f2` sont des facteurs, trace les moyennes de `y` en fonction des valeurs de `f1` et `f2` sur deux courbes différentes

`matplot(x,y)` graphe bivarié traçant la première colonne de `x` *vs.* la première colonne de `y`, puis la deuxième colonne de `x` *vs.* la deuxième colonne de `y`, etc.

`assocplot(x)` graphe d'association indiquant à quelle point les colonnes et lignes du tableau de contingence `x` dévient de l'hypothèse d'indépendance

`mosaicplot` graphe mosaïque des résidus d'un modèle log-linéaire

`pairs(x)` trace tous les graphes bivariés possibles entre les colonnes du tableau de données `x`

`plot.ts(x)` si `x` est de classe `"ts"` (time-serie), trace `x` en fonction du temps

`qqnorm(x)` fractiles de `x` en fonction des valeurs attendues sous l'hypothèse gaussienne

`qqplot(x, y)` fractiles de `y` en fonction des fractiles de `x`

`contour(x, y, z)`, `image(x, y, z)`, `persp(x, y, z)` variantes pour tracer les données de la matrice `z` en fonction des vecteurs `x` et `y`

`symbols(x, y, ...)` trace aux coordonnées spécifiées par `x` et `y` des cercles, carrés, rectangles, étoiles, boîtés à moustaches, etc.

`termplot(mod.obj)` trace les termes d'un modèle de régression en fonction des prédicteurs

Paramètres récurrents des fonctions graphiques

`add=FALSE` si `TRUE` superpose le graphe au précédent

`axes=TRUE` si `FALSE` ne trace pas d'axes

`type="p"` spécifie le type de tracé: `"p"` pour points, `"l"` pour lignes, `"b"` pour points liés par des lignes, `"o"` pour lignes superposées aux points, `"h"` pour lignes verticales, `"s"` ou `"S"` pour fonction en escaliers

`xlim=`, `ylim=` spécifie les limites des axes *x* et *y*

`xlab=`, `ylab=` annotation des axes *x* et *y*

`main=` titre du graphe en cours

`sub=` sous-titre du graphe en cours

Commandes graphiques bas-niveau

`points(x, y)` ajoute des points aux coordonnées `x` et `y`

`lines(x, y)` trace `y` en fonction de `x`

`text(x, y, labels, ...)` ajoute le texte `labels` aux coordonnées `(x,y)`

`rug(x)` ajoute les occurrences des points en abscisses)

`mtext(text, side=3, ...)` ajoute le texte `text` dans la marge `side`

`segments(x0, y0, x1, y1)` trace des lignes des points `(x0,y0)` aux points `(x1,y1)`

`arrows(x0, y0, x1, y1)` identique mais avec des flèches

`abline(a,b)` trace une droite de pente `b` et de décalage `a` par rapport à l'axe des *x*

`abline(h=y)` trace une ligne horizontale à l'ordonnée `y`

`abline(v=x)` trace une ligne verticale à l'abscisse `x`

`rect(x1, y1, x2, y2)` trace un rectangle défini par `x1`, `x2`, `y1` et `y2`

`polygon(x, y)` trace un polygone en liant les points de coordonnées définies dans les vecteurs `x` et `y`

`legend(x, y, legend)` ajoute une légende au point `(x,y)` spécifié par `legend`

`title()` ajoute un titre et éventuellement un sous-titre

`axis(side)` fonction de bas niveau pour gérer les axes de la figure

`box()` trace un cadre autour de la figure courante

Paramètres graphiques de bas de niveau

Ces paramètres sont définis à l'aide de la commande `par(...)` ou directement par passage à la fonction graphique d'appel

`adj` contrôle la justification du texte

`bg` spécifie la couleur de fond

`bty` contrôle le type de cadre tracé autour de la figure

`cex` contrôle la taille du texte et des symboles

`col` contrôle la couleur des symboles et des courbes (entier ou chaîne de caractères)

`font` un entier contrôlant le style de la police

`las` un entier contrôlant l'orientation des annotations des axes

`lty` contrôle le type de ligne (entier ou chaîne de caractère)

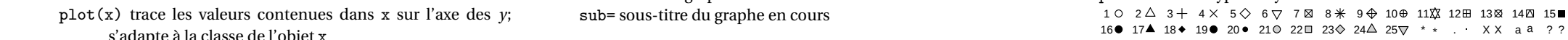
`lwd` contrôle l'épaisseur des lignes

`mar` contrôle l'espace entre le tracé et les bordures de la fenêtre

`mfc` un vecteur de la forme `c(nr,nc)` qui partitionne la fenêtre graphique en `nr` lignes et `nc` colonnes, les graphes étant tracés par colonne.

`mfrow` identique mais les graphes sont tracés par ligne

`pch` contrôle le type de symbole:



`ps` un entier contrôlant la taille du texte et des symboles

Optimisation

`optimize(fn,interval)` méthode d'optimisation pour les fonctions unidimensionnelles
`optim(par, fn)` méthode d'optimisation générique minimisant la fonction `fn` en partant de la valeur `par` des coefficients
`nlm(f,p)` minimise la fonction `f` à l'aide d'un algorithme type Newton, partant de la valeur `p` pour les coefficients
`approx(x,y=)` interpolation linéaire
`spline(x,y=)` interpolation par splines cubiques

Statistiques

Distributions

Toutes les fonctions suivantes peuvent s'utiliser en remplaçant la lettre `r` avec `d`, `p` ou `q` pour obtenir, respectivement, un tirage de n réalisations d'une variable aléatoire, la densité de probabilité, la fonction de répartition, et la valeur des fractiles.

`rnorm(n, mean=0, sd=1)` gaussienne
`rexp(n, rate=1)` exponentielle
`rgamma(n, shape, scale=1)` Gamma
`rpois(n, lambda)` Poisson
`rweibull(n, shape, scale=1)` Weibull
`rcauchy(n, location=0, scale=1)` Cauchy
`rbeta(n, shape1, shape2)` Beta
`rt(n, df)` Student
`rf(n, df1, df2)` Fisher–Snedecor (F)
`rchisq(n, df)` Pearson (χ^2)
`rbinom(n, size, prob)` binomiale
`rgeom(n, prob)` geometrique
`rhyper(nn, m, n, k)` hypergeometrique
`rlogis(n, location=0, scale=1)` logistique
`rlnorm(n, meanlog=0, sdlog=1)` lognormale
`rnbinom(n, size, prob)` binomiale negative
`runif(n, min=0, max=1)` uniforme
`rwilcox(nn, m, n)`, `rsignrank(nn, n)` Statistique de Wilcoxon

Modèles

`density(x)` estimateur à noyaux de la densité de `x`
`lm(formula)` ajuste un modèle linéaire; `formula` est typiquement de la forme `response ~ termA + termB + ...`
`glm(formula,family=)` ajuste un modèle linéaire généralisé
`nls(formula)` estimateur non-linéaire des moindres carrés des paramètres d'un modèle non-linéaire

Les fonctions ci-dessus renvoient un objet modèle dont l'ajustement dépend de la méthode utilisée. Certains des attributs de cet objet sont évalués à l'aide des commandes suivantes.

`aov, anova` fonction d'analyse de la variance
`df.residual(fit)` renvoie le nombre de degrés de liberté résiduels de `fit`
`coef(fit)` renvoie les coefficients estimés de `fit`
`residuals(fit)` renvoie les résidus
`predict(fit, ...)` prédiction à partir d'un modèle ajusté. Calcule également les intervalles de confiance et de prédiction.
`fitted(fit)` renvoie les valeurs ajustées
`logLik(fit)` calcule la log-vraisemblance du modèle et le nombre de paramètres
`AIC(fit)` calcule le critère AIC (Akaike information criterion)

Quelques fonctions liées au modèle linéaire.

`step` régression stepwise sur critère AIC/BIC
`regsubsets` du package `leaps`, régression exhaustive
`rstandard(fit)`, `rstudent(fit)` résidus standardisé ou studentisé associés à un modèle
`cooks.distance(fit)` calcul de la distance de cook
`lm.influence(fit)` diverse fonctions d'influence

Tests

`t.test(x,y=)` test de Student pour une ou deux population
`pairwise.t.test` test de Student apparié
`power.t.test` calculs de puissance associée à un test de Student
`chisq.test` test du χ^2 de contingence ou d'adéquation
`var.test` test de Fisher d'égalité des variance
`fisher.test` test exact de Fisher d'indépendance
`ks.test` test de Kolmogorov-Smirnov d'adéquation, une ou deux populations
`shapiro.test` test de normalité de Shapiro-Wilk
`binom.test` test du paramètre d'une loi binomiale
`prop.test` test d'égalité de proportion

Utiliser `help.search("test")` pour voir l'ensemble des tests statistiques disponibles