Aide mémoire des principales commandes R

Obtenir de l'aide. Lorsque l'on connaît le nom de la commande R (e.g., `cmd`), on peut taper `help(cmd)` ou `?cmd` (sauf dans le cas de certains opérateurs). Sinon, on peut rechercher à partir de mots-clés en tapant `help.search(cmd)`. Une alternative pour la recherche par motif consiste à utiliser `apropos(cmd)`. Pour connaître toutes les commandes fournies par un package (e.g., `pkg`), il suffit de taper `help(package=pkg)`.

An Introduction to R contient les informations essentielles sur les *éléments de base* du langage, l'écriture de *fonctions*, les procédures *graphiques*, et les principaux *modèles statistiques* disponibles sous R.

Dans les exemples qui suivent, `x` est supposé être un vecteur, `f` un facteur, `m` une matrice, `d` un data.frame.

Créer un vecteur simple ou avec motif

```
c(1,2,3)                # séquence (1,2,3)
1:3                     # séquence (1,2,3)
letters[1:3]            # les 3 premières lettres de l'alphabet (en minuscules)
LETTERS[1:3]            # les 3 premières lettres de l'alphabet (en majuscules)
seq(1, 6, by=2)         # séquence (1,3,6)
seq(1, 10, length=100)  # séquence de 100 nombres équirépartis entre 1 et 10
seq(3)                  # les 3 premiers entiers
seq_along(letters[1:3]) # les élements passés à la fonction sous forme d'entiers
sample(1:10, 5)         # 5 éléments tirés sans remise parmi la séquence (1..10)
sample(1:10, 5, rep=T)  # 5 éléments tirés avec remise parmi la séquence (1..10)
rep(c(1,2), 2)          # répète la séquence (1,2) 2 fois
rep(c(1,3), c(2,3))     # répète les éléments (1,3) 2 et 3 fois respectivement
rep(c(1,2), each=2)     # répète chaque élément de (1,2) 2 fois
```

Créer une matrice

```
matrix(c(1,2,3,4), nrow=2, ncol=2)       # crée une matrice 2x2 arrangée par colonnes
matrix(c(1,2,3,4), nr=2, nc=2, byrow=T)  # crée une matrice 2x2 arrangée par lignes
cbind(c(1,2), c(3,4))                    # assemble deux vecteurs en colonnes
rbind(c(1,2), c(3,4))                    # assemble deux vecteurs en lignes
replicate(2, c(1,2))                     # duplique les éléments (1,2) en colonnes
```

Créer un facteur

```
factor(c(1,2), labels=c("a","b"))  # crée un facteur dont les labels sont a et b
gl(2, 2, 8, labels=c("a","b"))     # crée un facteur à 2 niveaux répétés 2 fois, de taille 8
as.factor(x)                       # convertit x en facteur
factor(c(1,2), ordered=TRUE)       # créé un facteur ordonné
```

Créer un data.frame

```
data.frame(x, y, f)                 # assemble les vecteurs x, y (numériques) et f (facteur) en colonnes
as.data.frame(m)                    # convertit la matrice m en data.frame
as.data.frame(replicate(2, c(1,2))) # id.
data.frame(x, m)                    # assemble le vecteur x et la matrice m en colonnes
data.frame(x, y, row.names=z)       # assemble les vecteurs x et y en colonnes, en utilisant les valeurs de z comme row.n
```

Nommer les dimensions d'un objet R

```
names(x)      # nomme les éléments d'un vecteur
colnames(m)   # nomme les colonnes d'une matrice (ou d'un data.frame)
names(d)      # nomme les colonnes d'un data.frame
rownames(d)   # nomme les lignes d'un data.frame
dimnames(d)   # noms des lignes et colonnes d'un data.frame (ou d'une matrice)
```

Importer des données

```
scan("fichier.txt")                      # lit une suit de nombres
scan("fichier.txt", what="character")    # lit une suite de caractères
read.table("fichier.txt", header=TRUE)   # lit un fichier text délimité par des tabulations
read.csv("fichier.csv")                  # lit un fichier csv délimité par des ','
read.csv2("fichier.csv")                 # lit un fichier csv délimité par des ';'
read.spss("fichier.sav", to.data.frame=T)  # {foreign} lit un fichier SPSS
```

Manipuler un vecteur

```
length(x)        # nombre d'éléments dans x
head(x)          # 6 premiers éléments de x
tail(x)          # 6 derniers éléments de x
rev(x)           # éléments de x dans l'ordre inverse
t(x)             # x transposé (ligne -> colonne)
sort(x)          # trie x en ordre croissant
sort(x, decr=T)  # trie x en ordre décroissant
order(x)         # renvoit le rang des éléments de x
unique(x)        # renvoit les éléments uniques contenus dans x
```

Indexer les éléments d'un vecteur, d'une matrice, ou d'un data.frame

```
x[1]            # 1er élément de x
x[c(1,4)]       # 1er et 4ème élément de x
x["a"]          # si x est nommé, aème valeur de x
x[y[3]]         # ième élément de x où i = 3ème élément de y

m[1,2]          # élément de x situé en 1ère ligne, 2ème colonne
m[c(1,2), 2]    # éléments de x situé sur les lignes 1 et 2, 2ème colonne
m[1, "a"]       # si les colonnes de x sont nommées, 1ère ligne, colonne a
m[,1]           # tous les éléments de m dans la 1ère colonne
m[1,]           # tous les éléments de m dans la 1ère ligne
m[1:3,]         # les trois premières lignes, toutes les colonnes

d[1,]           # 1ère ligne de d
d[,1]           # 1ère colonne de d
d[,"a"]         # colonne nommée a dans d
d[,c("a","b")]  # colonnes nommées a et b dans d
d$a             # idem

d[d$f=="a",]    # toutes les valeurs de d pour lesquelles d$f="a"
d[d$f=="a",1:2] # toutes les valeurs des colonnes 1 et 2 de d pour lesquelles d$f="a"
```

Manipuler un facteur

```
levels(f)                   # niveaux de f
nlevels(f)                  # nombre de niveaux de f
relevel(f, ref="a")         # change la catégorie de référence de f en a
levels(f)[1:2] <- "a"       # regroupe les deux premiers niveaux sous le niveau a
as.numeric(as.character(f)) # convertit les niveaux de f en nombre en préservant leur cardinalité
table(f)                    # distributions des niveaux de f
ordered(f)                  # convertit f en facteur ordonné
```

Filtrer les valeurs d'un data.frame

```
subset(d, f == "a")             # tous les éléments de d tel que d$f="a"
subset(d, f %in% c("a", "c"))   # tous les éléments de d tel que d$f="a" ou d$f="c"
subset(d, select=V1:V4)         # les colonnes V1 à V4 de d
subset(d, select=c(V1:V3, V5))  # les colonnes V1 à V3, et V5 de d
subset(d, f == "a", V1:V3)      # les colonnes V1 à V3 de d, tel que d$f="a"
complete.cases(d)               # indique quelles lignes sont sans valeurs manquantes
na.omit(d)                      # d sans valeurs manquantes
```

Accéder aux éléments d'un data.frame

```
with(d, mean(x))            # calcule la moyenne de x contenu dans d (lecture seule)
within(d, x <- factor(x))   # convertit la colonne nommée x dans d en facteur (écriture)
within(d, {                 # id. pour plusieurs variables
         x <- factor(x)
         y <- factor(y)
       })
transform(d, x2 = log10(x)) # crée une nouvelle variable x2 dans d à partir de d$x
```

Opérer par lignes/colonnes

```
apply(m, 1, mean)                       # moyenne de chaque ligne de m
apply(m, 2, mean)                       # moyenne de chaque colonne de m
apply(d, 1, function(x) sum(is.na(x)))  # nombre de valeurs manquantes par ligne
```

Résumé numérique selon les niveaux d'un facteur

```
tapply(x, f, mean)             # moyenne de x par niveaux de f
tapply(x, list(f1, f2), mean)  # moyenne de x pour chaque croisement des niveaux de f1 et f2
by(d, f, colMeans)             # moyenne de chaque colonne de d par niveaux de f
aggregate(x, list(f=f), mean)  # moyenne de x par niveaux de f, sous forme de data.frame
```

Générateurs de nombres aléatoires et combinatoire

```
rnorm(10, mean=2, sd=1)       # 10 aléas gaussiens (moyenne 2, ds 1)
runif(10, min=0, max=1)       # 10 aléas uniformes compris entre 0 et 1
rbinom(10, 1, 1/2)            # 10 aléas binaires avec probabilité de survenue 1/2
sample(1:10, 10)              # permutation de la séquence (1..10)
sample(1:10, 5, replace=TRUE) # tire 5 éléments parmi la séquence (1..10) avec remise
combn(4, 2)                   # combinaisons de 2 éléments choisis parmi 4
choose(4, 2)                  # nombre de combinaisons de 2 éléments choisis parmi 4
```

Mise à jour : Sun Mar 25 12:35:26 CEST 2012