Formulaire de R

Pour obtenir l'aide de R sur une commande, il faut taper `help("nom de la commande")`. L'aide est en anglais.

1 Vecteurs, matrices, tableaux, listes

- Génération
  - d'un vecteur `c(nombre1,nombre2,...)` ou `c(vecteur1,vecteur2,...)`
  - d'un vecteur séquence : `seq(from,to,by ou length.out)`. Crée une séquence allant de *from* à *to*, soit par pas de *by*, soit de longueur totale *length.out*.
  - d'un vecteur : `rep(vecteur ou nombre, nombre de fois)`.
  - d'une matrice : `matrix(dvecteur, nblignes, nbcol, byrow=TRUE ou FALSE,dimnames)`
  - d'une matrice diagonale : `diag(vecteur)`
  - d'une matrice triangulaire : `lower.tri(vecteur,diag=FALSE)` ou `upper.tri`.
  - d'un tableau de données : `array(données(vecteur),dim,dimnames)`.
  - d'une liste : `list(nomdonnée1=donnée1,nomdonnée2=donnée2,...)`
  - d'une série temporelle : `ts(vecteur,début,fin,frequency(nb obs/unité tps) ou deltat(int entre 2 obs))`
- Accès aux éléments
  - d'un vecteur `v[numéro élément]`
  - d'une matrice/d'un tableau de données de dimension p : `M[ligne,colonne]` ou `T[dim1,dim2,dim3,...]`
  - d'une liste : soit `L[[numéro de la donnée]]` soit `L$nomdonnée`.
  - d'une série temporelle : les données sont traitées comme un vecteur par R. Pour accéder aux autres caractéristiques de la série temporelle, on utilise les commandes `start(t)`, `end(t)`, `frequency(t)`.
- Manipulation
  - dimension : `length` pour un vecteur, `nrow`, `ncol`, `dim`.
  - manipulation terme à terme : toutes les fonctions usuelles.
  - arrondi : Les fonctions `round`, `ceiling` et `floor` permettent de donner des arrondis : `round` arrondi à l'entier le plus proche, `ceiling` à l'entier supérieur, `trunc` à l'entier inférieur. La fonction `round(x,nb)` permet de garder nb chiffres après la virgule.
  - comparaison : les comparaisons `<`, `<=` (...) se font composante par composante.
  - algèbre : `a%%b` donne le reste de la division de a par b, `factorial(k)` calcule $k!$, `choose(n,k)` calcule C_n^k.
  - manipulation de matrices : `%*%` pour la multiplication, `solve` pour l'inverse, `t` pour la transposée.
  - tri des données : `sort`.
  - opérations sur les vecteurs : `max`, `which.max`, `sum`,`prod`.
  - opérations sur les vecteurs : `cumsum` (sommes cumulées), `cumprod`, `cummin`, `cummax`.

– concaténation : `c`, `cbind`,`rbind`.
– opération sur les lignes ou les colonnes : `rowSums`, `rowMeans`, `colMeans`, `colSums`.

– La fonction `table(v)` crée la table de contingence du vecteur v.
– La fonction `apply(matrice, 1 pour lignes ou 2 pour colonnes, f)` permet d'appliquer une fonction aux colonnes ou aux lignes d'une matrice. Typiquemente, $f = mean$ ou toute autre fonction du même type.
– La fonction `(outer(u,v,'f')` permet à partir de deux vecteur u de longueur m et v de longueur n de créer la matrice

$$\begin{pmatrix} f(u(1),v(1)) & f(u(1),v(2)) & \dots & f(u(1),v(n)) \\ f(u(2),v(1)) & f(u(2),v(2)) & \dots & f(u(2),v(n)) \\ \dots & \dots & \dots & \dots \\ f(u(m),v(1)) & f(u(m),v(2)) & \dots & f(u(m),v(n)) \end{pmatrix}$$

f est une fonction de deux variables : $f = +$ ou $*$, par exemple.

Le logiciel `R` est capable de gérer des séries temporelles ou des fichiers de données. Il comporte aussi beaucoup de jeux données déjà implémentés. Si on veut utiliser un fichier excel, il faut d'abord le convertir en .csv puis utiliser la fonction `read.table("adresse du fichier")`, `read.csv` (pour les fichiers codés à la manière anglaise) et `read.csv2` (pour les fichiers codés à la manière française).
On peut stocker des tableaux dans un fichier grâce à la commande `write.table`.

2 Probabilités

R connaît la plupart des lois de probabilités usuelles. On y accède en chargeand le package stats (`library(stats)`). Il peut :
– Simuler une variable suivant cette loi grâce à la commande `rloi`
– Donner la densité (pour une variable continue) ou les probabilités marginales (pour une loi discrète) : `dloi`
– Donner la fonction de répartition : `ploi`
– Donner la fonction quantile : `qloi`.

Loi usuelles :
– Loi normale : `norm`
– loi exponentielle : `exp`
– loi uniforme : `unif`
– loi Gamma : `gamma`
– loi binomiale : `binom`
– loi de Poisson : `pois`
– loi du χ^2 : `chisq`
– loi de Student : `t`

Par exemple, la commande `dexp(2,1)` donne la densité d'une loi exponentielle de paramètre 1 au point 2. La fonction `sample` permet de générer aléatoirement des entiers :
– `sample(v)` génère une permutation aléatoire du vecteur v.
– `sample(v,n)` choisit au hasard n nombres parmis ceux du vecteur v (tirage sans remise).
– `sample(m,n)` choisit au hasard n nombres parmis les m premiers entiers (tirage sans remise), (c'est un raccourci pour `sample(seq(1,m),n)`).

Le logiciel `R` a de plus beaucoup de fonctions qui permettent d'étudier les données : par exemple `mean`, `var` (qui calcule la variance corrigée, ce qui est légèrement différent de la variance empirique), `sd` (écart-type), `cov`, `cor`, `median`, `mad` :

3 Graphiques

Il existe différents type de graphiques sous `R` : les graphes classiques, les histogrammes, les boîtes à moustaches . . . .

Pour les graphes les plus simples, on utilise les trois commandes `plot(x,y,options)`, `points(x,y,options)` et `lines(x,y,options)`. Ces trois commandes permettent de tracer y en fonction de x.
On utilise toujours `plot` pour commencer un graphique. Les options les plus utilisées sont :

- `type='p'` par défaut (ne trace que des points), ='l' si on veut tracer une ligne, ='s' pour une fonction en escalier.
- `col ='black'` par défaut (couleurs possibles : red, blue, green, yellow, orange, magenta, pink, ...)
- `pch` (c'est le style de point utilisé, il peut être égal à 1,2,3,...)
- `lty` (c'est le style de ligne utilisé, il peut être égal à 1,2,3,...)
- `ylim=c(a,b)` (option très utile si on veut superposer plusieurs graphiques. Par défaut, `R` affiche les y entre min(y) et max(y), on peut forcer l'affichage d'autres valeurs). Ne fonctionne qu'avec la commande `plot`.
- `xlim=c(a,b)` (même chose que ylim, mais pour x. Peut-être moins utile).
- `main=` donne un titre au graphe.
- `xlab=`, `ylab=` change le nom des axes.

Pour superposer un deuxième graphique, on utilise soit `points` (pour tracer des points) soit `lines` (pour tracer des lignes), soit utiliser l'option `add=TRUE` (utilisable avec tous les types de graphes). La fonction `abline(a=,b=,h=,v=)` permet de tracer une droite, soit en précisant l'abscisse à l'origine (a) et la pente (b), soit de tracer une ligne horizontale d'ordonnée h, soit de tracer une ligne verticale d'abscisse v.

La fonction `hist` trace l'histogramme d'un vecteur. La commande `hist(x,breaks,freq=TRUE)` donne le nombre de variables dans chaque intervalle, et la commande `hist(x,breaks,freq=FALSE)` donne la proportion de variables dans chaque intervalle. Pour changer la couleur, on peut utiliser `border=`.

Attention : sauf dans le cas d'une série temporelle, il faut toujours préciser les deux vecteurs !.

La fonction `boxplot` trace la boîte à moustache d'un vecteur. Elle permet de tracer soit la boîte à moustache d'un seul vecteur, soit la boîte à moustache de plusieurs vecteurs côte à côte. La fonction `qqplot` permet de tracer le graphe quantile/quantile de deux vecteurs (X, Y). Les deux vecteurs n'ont pas besoin d'être de même longueur. Avec `qqline`, on peut ajouter la droite qui passe par les quartiles $Q1$ et $Q3$. La fonction `qqnorm` trace les quantiles empiriques d'un vecteur X par rapport aux quantiles théoriques de la loi normale $\mathcal{N}(0, 1)$.

On peut aussi tracer plusieurs graphiques côte à côte : pour cela on commence par partager la fenêtre des graphes en plusieurs cases grâce à la commande `split.screen(c(i,j))` (partage en i lignes et j colonnes). Pour tracer un graphe dans une case, on peut donner le numéro de la case : `screen(k)` et on trace ensuite le graphique grâce aux commandes habituelles. Il faut fermer les graphes à la fin, sinon tout se superpose. Pour cela, on utilise la commande `close.screen(all=TRUE)`.

4 Un peu de programmation

Le logiciel `R` n'aime pas du tout les boucles. À chaque fois que cela est possible, il faut passer par les matrices et les opérations sur les vecteurs. (Une boucle pour 5 ou 10 opérations n'est bien sûr pas génante, une boucle for de 1000 itérations peut énormément ralentir le programme).

- Structure d'une boucle for :

```
for (k in 1:n)
{
lignes codes
}
```
Les accolades ne sont pas nécessaires s'il n'y a qu'une seule ligne de code.

– Conditions
```
if (condition)
{
lignes codes
}
else
{
lignes codes
}
```
On n'est pas obligé de mettre un `else`. `R` ne sait comparer pas utiliser la condition `if` sur un vecteur. La structure `if/then/else` s'utilise quand on a qu'une seule variable à tester. Pour faire des opérations conditionnelles sur un vecteur, on peut simplement utiliser les fonctions `<` ou `==`.

– Boucles while
```
while (condition)
{
lignes commandes
}
```

– Fonctions
```
nomfonction <- function (nomvariables)
 {
une ligne de code
}
```
ou si la fonction est plus compliquée :
```
nomfonction<- function (nomvariables)
{
lignes code
return(expression)
}
```
Dans ce dernier cas, le résultat de la fonction ne s'affichera pas automatiquement.

5 Tests du chi2

La fonction `chisq.test` permet de faire les tests du χ^2. Il faut faire très attention, car suivant les valeurs qu'on rentre, elle fera soit le test du χ^2 d'adéquation, soit le test du χ^2 d'indépendance.

– `chisq.test(M)` avec M une matrice d'au moins deux lignes et deux colonnes ou `chisq.test(x,y)` avec x et y de même longueur fait un test du χ^2 d'indépendance.

– `chisq.test(x,p=v)` avec p un vecteur de même longueur que x et de somme 1 réalise un test du χ^2 d'adéquation à la loi donnée par p. Si p n'est pas spécifiée, les probabilités sont toutes considérées égales par `R`.

6 Régression linéaire

La commande `lm` permet de faire une régression linéaire multiple. Pour faire une régression linéaire de y en fonction de x, il faut taper : `lm(y~x)`. Le logiciel `R` estimera a et b.

Attention :

- Si on pense que $b = 0$, il faut alors faire `lm(y~0+x)`.
- Si $Y = a_0 + a_1X + a_2X^2$, il faudra taper : `lm(y~x+I(x^2))`.

La commande `lm` calcule entre autre :

- les coefficients (`coefficients`)
- les résidus (`residuals`)
- le critère de détermination et le critère de détermination ajustée.

On peut utiliser `summary` pour obtenir toutes les informations sur la régression. Cette commande affiche

- les quartiles des résidus.
- la valeur des coefficients, leur écart-type estimé, et le résultat du test "Ce coefficient est nul".
- les coefficients de détermination et de détermination ajustée.

7 Modèles ARIMA

- `acf(nomvecteur,lag=nombre de coefficients à tracer)` permet de visualiser les coefficients d'autocorrélation. De plus, `R` représente l'intervalle $\left[-\frac{1,96}{n}, \frac{1,96}{n}\right]$ où doivent se trouver les coefficents à 95% si on a un bruit blanc.
- La commande `Box.test(nomvecteur,lag=nb de coef pris en compte, fitdf=nb de coef estimés)` réalise le test du porte manteau.
- La commande `arima(nomvecteur, order=c(p,0,q)` estime les coefficients d'un processus arma d'ordre (p, q). Elle renvoie entre autres :
  - les coefficents du modèle (coef)
  - la variance estimée des coefficents (var.coef)
  - la variance estimée des ε_i (sigma2)
  - la vraisemblance du modèle
  - le critère aic : $aic = 2 \times$ nombre de paramètres estimés $- 2 \times$ log-vraisemblance.
  - les résidus $\hat{ep}_i$ (residuals)
- La commande `auto.arima(nomvecteur,ic="aic" ou "bic")` du package `forecast` sélectionne le meilleur modèle arima (par rapport au modèle sélectionné) et renvoie les coefficients.
- La commande `lm` permet de faire une régression linéaire multiple. On peut utiliser `summary` pour obtenir toutes les informations sur la régression.