Sur le logiciel R

Mouhamadou DOSSO[1]

[1]Université Félix Houphouët-Boigny de Cocody-Abidjan, Côte d'Ivoire

UFR de Mathématiques et Informatique

Outline

1. Interface d'utilisation sous windows
2. Les objets : Vecteurs, Matrices, Matrices à plus de deux dime
3. Quelques fonctions usuelles
4. Quelques fonctions de statistique exploratoire
5. Construction d'une nouvelle fonction
6. Quelques éléments de programmation
7. Les entrées/Sorties et gestion des objets crées

Chapitre 3

LOGICIEL R

Fonction des menus

Ce logiciel R forme un interface utilisateur simple. Elle est structurée autour d'une barre de menu et de diverses fenêtres.

- Le menu **file** (ou fichier) : contient les outils nécessaires à la gestion de l'espace de travail ;
- Le menu **Edit** (ou Edition): contient les habituelles commandes de copier-coller et la boîte de dialogue pour la personnalisation de l'apparence de l'interface ;
- Le menu **Misc** traite de la gestion des objets en mémoire et permet d'arrêter une procédure en cours de traitement.
- Le menu **Package** : automatise la gestion et le suivi des librairies de fonctions ;
- Les menus **windows** (ou fenêtre) et **Help** (ou Aide) : assurent des fonctions similaires à celles qu'ils occupent dans les autres applications Windows (les fénêtres, l'aide en ligne et manuels de références de R)

D’autres fenêtres

Parmi les fenêtres, on a la console qui est la fenêtre principale où on réalise par défaut les entrées de commandes et sorties de résultats en mode texte. On y ajoute également un certain nombre de fenêtres telles que les fenêtre graphiques et les fenêtres d’informations (historique des commandes, aide, visualisation de fichier, etc...).
Ces dernières sont toutes appelées à partir de la console par des commandes spécifiques.

Objets

Les éléments de base du language R sont des objets qui peuvent être des données (vecteurs, matrices,...), des fonctions, des graphiques,...
Les objets se différencient en mode (qui sont :**null (objet vide), logical, numeric, complex, character**) qui décrivent leur contenu et leur classe.
Les principales classes d'objets sont : vector, matrix, array, factor, time-series, data.frame, list
N.B.: On peut avoir des vecteurs, matrices, tableaux, variables,... de mode **null (objet vide)**, **logical**, **numeric**, **complex**, **character**.

Les vecteurs

C'est l'objet de base dans R.

$>$	$a = c(5, 5.6, 1, 4, -5)$	Création de l'objet a recevant un vecteur numérique de dimension 5 et de cordonnées $5, 5.6, 1, 4, -5$.
$>$	a	Affichage du vecteur a
$>$	$a[1]$	Affichage de la première coordonnée du vecteur a
$>$	$b = a[2:4]$	création du vecteur numérique b de dimension 3 et de coordonnées $5.6, 1, 4$

Vecteurs

$>$	$d = a[c(1,3,5)]$	Création d'un vecteur numérique d de dimension 3 et de coordonnées $5, 1, -5$
$>$	$sum(d)$	Calcul de la somme de d
$>$	$length(d)$	Affichage de la dimension de d
$>$	$t(d)$	Transposition du vecteur d
$>$	$t(d)\%*\%e$	Produit scalaire entre les vecteurs d et e

Les matrices

Les matrices comme les vecteurs, sont de modes quelconque, mais elles ne peuvent pas contenir des éléments de nature différente. La syntaxe de création d'une matrice est : **matrix(vec,nrow=n,ncol=p,byrow=T)** où vec est le vecteur contenant, qui seront rangés en colonne (sauf si "byrow=T" est choisie) les éléments de la matrice.

$> \quad a = 1:20$

$> \quad b = sample(1:10, 10)$

$> \quad x1 = matrix(a, nrow = 5)$

Création d'une matrice numérique $x1$ de dimesinsion 5×4 ayant pour prémière ligne $1, 6, 11, 16$

Les matrices

$>$	$x2 = matrix(a, nrow, byrow = T)$	Création d'une matrice numérique $x2$ de dimension 5×4 ayant pour prémière ligne $1, 2, 3, 4$
$>$	$x3 = t(x2)$	Tranposition de la matrice $x2$
$>$	$b = x3\% * \%x2$	produit de matriciel entre $x3$ et $x2$
$>$	$dim(x1)$	Affichage de la dimension de $x1$

Les matrices à plus de deux dimensions

Les matrices à plus de deux dimensions sont crées à l'aide de la commande suivante :

array(vec,c(n,p,q,...)) où

- **vec** est le vecteur contenant les éléments de la matrice
- **c(n,p,q,...)** désigne les dimensions :
  - **n** est le nombre de lignes,
  - **p** le nombre de colonnes,
  - **q** le nombre de lignes,

  ⋮

$>x = array(1:50, c(2,5,5))$

$>x$

$>x[1,2,2]$

$>dim(x)$

$>aperm(x)$ Transposition généralisée de **x, x[i,j,k]** devient **x[k,j,i]**

Les listes

La construction d'une liste passe par la fonction **list(nom1=el1,nom2=el2,...)** , l'utilisation des noms étant facultative.

$>li = list(num = 1:5, y = "couleur", a = T)$

$>li$

$>li\$num$

$>li\$a$

$>li[[1]]$

$>li[[3]]$

$>a = matrix(c(6,2,0,2,6,0,0,0,36), nrow = 3)$

$>res = eigen(a, symmetric = T)$ Diagonalisation de a

$>res\$values$

$>res\$vectors$

Les structures de données

Les tableaux de données (**data.frame**) constituent une classe partitulière de listes consacrée au stockage des données destinées à l'analyse. Pour créer un tableau de données, on peut régrouper les tableaux de même longueur à l'aide de la commande

data.frame(nom1=var1,nom2=var2,...).

On peut ainsi transformer une matrice en tableau de données en utilisant la commande **as.data.frame(mat)**.

Les structures de données

Exemple

$> v1 = sample(1:12, 30, rep = T)$
Echatillonnage avec remise
dans les entiers de 1 et 12
$> v2 = sample(LETTERS[1:10], 30, rep = T)$
$> v3 = runif(30)$
30 réalisations indépendantes d'une loi
uniforme sur $[0, 1]$
$> v4 = rnorm(30)$
30 réalisations indépendentes d'une loi
normale de moyenne 0 et variance 1
$> v1; v2; v3; v4$
$> xx = data.frame(v1, v2, v3, v4)$
$> xx$

Les distributions usuelles

Loi	Nom (nomdist)	Paramètres	Val. par défaut
Beta	beta	shape1,shape2	
Binomiale	binom	size,prob	
Cauchy	cauchy	location,scale	0,1
Khi-Deux	chisq	df	
Explonentielle	exp	1/mean	1
Fisher	f	df1,df2	
Gamma	gamma	shape,1/scale	-1
Géométrique	geom	prob	
Hypergéométrique	hyper	m,n,k	
Log-Normale	lnom	mean,sd	0,1
Logistique	logis	location,scale	0,1
Normale	norm	mean,sd	0,1
Poisson	pois	lambda	

Les distributions usuelles

Student	t	dt	
Uniforme	unif	min,max	0,1
Weibull	weibull	shape	

Pour chacune de ces distributions, on dispose de quatre commande préfixées par une des lettres **d,p,q,r** et suivi du nom de distribution :

- **dnomdist** : fonction de densité pour une distribution probabilité continue et de la fonct. de probabilité ($\mathbb{P}(X = k)$);
- **pnomdist** :fonction de répartition ($\mathbb{P}(X \leq x)$);
- **qnomdist** : fonction de quantité ;
- **rnomdist** :génère des réalisations aléatoires indépendantes de la distribution **nomdist**.

Quelques fonctions de statistique exploratoire

$>$ *data*(*women*)	
$>$ *names*(*women*)	
$>$ *attach*(*women*)	
$>$ *mean*(*height*)	Calcul de la moyenne empirique de variable quantitative **height**
$>$ *var*(*height*)	Calcul de la varance empirique de **height** estimateur non biaisé (diviseur $n-1$)
$>$ *sd*(*height*)	Calcul de l'écart-type de **height**
$>$ *median*(*height*)	Calcul de la médiane empirique de **height**
$>$ *quantile*(*height*)	Calcul des quantiles empiriques de **height**
$>$ *summary*(*weight*)	Résumé de **weight**
$>$ *summary*(*women*)	Résumé de **women**

Quelques fonctions de statistique exploratoire

> $hist(weight, nclass = 15)$
Histogramme de **weight** constitué de 15 classes
> $cor(height, weight)$
Calcul du coefficient de corrélation linéaire
> empirique entre **weight** et **height**
> $v1 = rnorm(100)$
> $hist(v1)$
> $v2 = factor(saple(letters[1:4], 100, rep = T))$
> $table(v2)$
Résumé de la variable qualitative **v2**
> $barplot(table(v2))$
Diagramme en barre de **v2**
> $sd(height)$
Diagramme en secteur **v2**

Construction d'une nouvelle fonction

On peut définir une nouvelle fonction soit directement à partir de la console, soit via un éditeur de texte externe grâce à la commande *fic*(*nom_fonction*). La seconde possibilité permet la correction du code en cours d'édition, tandis que la prémière s'éffectue ligne par ligne, sans retour en arrière.
La syntaxe générale de la définition d'une nouvelle fonction par l'expression

nom_fonction-function($arg1[-expr1], arg2[-expr2], ...$){
bloc d'intructions
}

Les accolades définissent le début et la fin du code source de la fonction ; les crochets ne font pas partie de l'expression mais indiquent le caractère facultatif des valeurs par défaut des arguments.

Construction d'une nouvelle fonction

On peut créer également une fonction personnalisée partir d'une fonction existente grâce

nom_fonction2=edit(nom_function1); *fix*(nom_fontion2)

Exemple

$>x = 2$

$>carre = function(x)\ \{\ x - x * x;\ x\ \}$

$>carre(2)$

$>x$

$>fix(carre)$

On peut ajouter des commentaire au code en les faisant précéder du symbole #

Quelques éléments de programmation

Cette partie concerne les commandes **if**, **while**, **for**

Exemple

$>bool = T$

$>i = 0$

$>while(bool == T)\ \{i = i + 1;\ if\ (i > 10)\ \{bool = F\}\}$

$>i$

$>s = 0\ \ ;\ \ x = runif(10000)$

$>for\ (i\ in\ 1 : 1000)\ \{s = s + x[i]\}$

$>s$

$>un = rep(1, 10000)$

$>t(un)\% * \%x$

$>s = 0$

Quelques éléments de programmation

Exemple (suite)

$$>system.time(for\ (i\ in\ 1:10000)\ \{s = s + x[i]\})[3]$$
$$>system.time(t(un)\%*\%x)[3]$$

N.B.: Le logiciel **R** peut avoir des problème de mémoire si on fait appel à un nombre élévé de boucles ; de ce fait, il est préférable de les remplacer par les outils de calcul matriciel.

Les entrées/Sorties

Les données que l'on souhaite analyser proviennent de source externes sous forme de fichiers. Les objets crées doivent pouvoir être sauvegardés dans des fichiers afin d'être transportables.

- **Formats propriétaire**: **save()** autorise la sauvegarde de n'importe quelle liste d'objet en mémoire (sous un chemin quelconque) ; ces objets peuvent être rechargés en mémoire grâce à la fonction **load()**
- **Fichiers textes ASCII**: ils sont prisent en charge par la commande **scan()** ; Les argurments de cette fonction permettent de décrirent la structure du fichier texte.
- **Logiciels statistiques** : La librairie **foreign** offre des outils pour une sélection des logiciels statistiques plus courants, à savoir MINITAB, S-PLUS,SAS, SPSS et STATA.

gestion des objets crées

Lors de l'exécution de **R**, les fichiers **.RData** et **.Rhistory** sont automatiquement crées dans le répertoire de travail. Losqu'on quitte **R** à l'aide de la commande **q()**, les nouveaux objets crées peuvent être sauvegardé dans les fichier : **.RData** ou **.Rhistory** en mode texte.

- **history()**: permet de visualiser la suite de commandes que l'on a tapée
- **getwd()**: permet de conaitre le répertoire de travail courant au cours d'une session ;
- **ls()** : permet de visualiser la liste des objets crées
- **rm()**: permet de détruire des objets