

Notes de cours R

Thibaut Marmey

October 17, 2018

Contents

1	Programmation R	1
1.1	Généralité	1
1.2	Fonctions	2
1.3	Manipulation de chaînes de caractères	2
1.4	Gestion de données complexes	3
1.5	Opérations sur des vecteurs	3
1.6	Analyses statistiques sur des vecteurs	4
1.7	Manipuler et comparer plusieurs vecteurs	5
1.8	Corrélations et régression linéaire	6

1 Programmation R

1.1 Généralité

- *lien internet* : reprise cours OCR
- *lien internet* : doc d'installation de R et Rstudio
- Utilisation de la documentation "aide" : `help()` ou `help("nom de la fonction")` ou `?log`
- Documentation plus détaillée sur internet : `help.start()`
- Interaction avec l'environnement R :
 - liste toutes les variables créées : `ls()`
 - liste des variables avec une succession de lettre particulière : `ls(pattern="mot")`
 - supprimer des variables : `rm(var)`
 - quitter le travail : `q()` ou `quit()`
- Ecrire des commentaires avec "`#`"
- Utiliser la fonction `print()` pour afficher les informations, textes sur la console
- Enregistrer les variables que l'on veut :
`save(var1, var2, ..., file = "nameFile.extension")`.
Enregistrement de ces fichiers dans le répertoires associés *Tools/Global Options*, ici c'est Documents/Notes-de-cours/Cours R/Test

1.2 Fonctions

- Tester le type d'une variable :
 - *is.character(var)*
 - *is.numeric(var)*
 - *is.logical(var)*
- Spécifier le typage de la variable :
 - *as.numeric(var)*
 - *as.logical(var)*
 - *as.character(var)*
- Renvoyer l'entier inférieur : *floor(nb)*
- Renvoyer l'entier supérieur : *ceiling(nb)*
- Arrondir à l'entier le plus proche : *round(nb)*
- Racine carré : *sqrt()*
- Fonction trigo : *cos(angleRad)*, *sin*, ...
- Fonction somme : *sum(vect)*
- Longueur vecteur : *length(vect)*

1.3 Manipulation de chaînes de caractères

- Saisir des données depuis le clavier : *scan()*. Pour valider l'input appuyer sur *entrée*, et un nouvel input apparaîtra. Pour arrêter la prise d'input appuyer directement sur *entrée*.
- Rentrer le nombre d'input avec l'appel de *scan(nmax=n)*
- La concaténation de texte et nombre avec *paste(texte, nombres, ...)*
- Combinaison de *scan()* et *paste()* : *paste('Bonjour j'ai ', scan(nmax=1), ' ans')*
- Permet de compter le nombre de lettres présentes dans la chaîne de caractères : *nchar("chaîne")*
- Mettre les caractères en majuscules ou minuscules : *toupper()*, *tolower()*
- Extraire une sous chaîne de caractères avec point de départ et d'arrivée inclus : *substr("chaîne", start, stop)*. La chaîne de caractères commence à 1.

1.4 Gestion de données complexes

- Le vecteur : élément de base du langage, c'est une liste d'éléments étant tous du même type.
 - Spécifier deux attributs : le type de ces éléments et la longueur du vecteur (nb d'éléments)
 - Les fonctions prennent toutes des vecteurs en paramètre. Elles renverront le résultat sous la forme d'un vecteur de longueur (généralement) égale à la longueur du vecteur d'entrée.
 - Différentes méthodes pour créer un vecteur :
 - * la fonction *vector(type, length)*
ex : vector("numeric", 10) crée un vecteur de 10 éléments tous égaux à 0. Les valeurs par défaut sont 0 (numeric), "" (character), FALSE (logical).
 - * les fonctions *numeric(length)*, *character(length)*, *logical(length)*
 - Générer des séries de nombres
 - * Suite d'un nombre à un autre : *nb1:nb2*
 - * Suite d'un nombre répété : *rep(element, length)*
 - * Séquence de nombres : *seq(start, stop, step)*
- Concaténer plusieurs vecteurs (ils doivent contenir les mêmes types de variables) : *c(vect1, vect2, ...)* (bonne méthode pour créer rapidement un vecteur avec des valeurs directement attribuées *ex: c(70, 50, 10, 0, 5)*)
- Nommer les éléments du vecteur :
names("nom1", "nom2", rep(NA,2), "nom3", ...) (ici permet de nommer des éléments et d'en laisser 2 sans nom)
- Indexation numériques : travailler seulement sur une sous partie d'un vecteur. Les éléments des vecteurs sont indexés de 1 jusqu'au dernier.
 - Accéder à un élément du vecteur via son index : *vect1[index]*
 - Accéder à plusieurs éléments du vecteur :
vect1[nb1:nb2], *vect1[c(index1, index2, ...)]* il est en fait possible d'utiliser toutes les fonctions de séquences ou listes de nombres comme index.
 - Utiliser des booléens pour sélectionner les éléments à récupérer :
vect1[vect1 > 7] (on récupère ici tous les éléments de vect1 qui sont supérieur à 7).
 - Utiliser les noms des éléments : *vect1[c("nom1", "nom2")]*
- Modifier un élément du vecteur : *vect1[index] = valeur* (sachant que index peut être des fonctions telles que *c()*, *nb1:nb2*, etc...

1.5 Opérations sur des vecteurs

- Addition sur vecteur : *vect1 = vect1 + 1* (+1 à tous les éléments)
Dans ce cas là, R prend le plus grand vecteur pour y effectuer l'opération à partir

du vecteur plus court.

`vect1 = vect1 + c(nb1, nb2,...)` (revoie d'un avertissement si le grand vecteur n'est pas un multiple du petit vecteur)

- Connaître la longueur d'un objet : `length()`
- Sélectionner seulement certains éléments :
 - Les premiers éléments : `head(nb d'éléments)`
 - Les derniers éléments : `tail(nb d'éléments)`
- Triez les vecteurs :
 - `sort()` renvoie un nouveau vecteur contenant les mêmes éléments mais triés dans l'ordre.
Dans l'ordre croissant : `sort(vect1)`
Dans l'ordre décroissant : `sort(vect1, decreasing=True)`
Dans le cas des caractères, la priorité est donnée aux majuscules, ainsi un "a" sera placé après un "Z". Pour contourner ce problème, utiliser les fonctions `toupper()` ou `tolower()`.
 - `order()` renvoie l'ordre des éléments via leur indice.
> [1] 2 5 4 1 3 (ce résultat indique que l'élément d'index 2 est le plus petit, suivi de l'index 5, 4, 1 et 3)
 - `rank()` renvoie le rang de l'élément au sein de la distribution. Par défaut le `ties.method` est `average` ce qui implique que si plusieurs éléments sont de même rang la fonction attribuera ainsi la moyenne de ses rangs.

1.6 Analyses statistiques sur des vecteurs

- Analyser la distribution d'un vecteur consiste à calculer différentes valeurs sur celui-ci : la moyenne, la médiane ou la variance par exemple.
- Calculer la moyenne : `mean(vect1)`
- Calculer la médiane : `median(vect1)`
La médiane permet de répartir deux parties en un même nombre d'éléments. C'est en fait la valeur centrale de la distribution.
- En présence de valeurs NA dans la distribution, utiliser l'argument facultatif : `na.rm = TRUE`.
- Mesurer la dispersion d'une distribution :
 - les fonctions `min()` et `max()`. Utiliser `na.rm` si présence de NA.
 - Les quantiles sont les valeurs permettant de séparer une distribution ordonnée de valeurs en q sous-distributions.
Généralement on calcule les quartiles c'est à dire on sépare la distribution en 4-quantiles.
Utiliser la fonction `quantile(vect1)`
 - La fonction tout en un : `summary(vect1)` renvoie :

- * la valeur min
- * le premier quartile (25%)
- * la médiane (ou second quartile (50%))
- * la moyenne
- * le troisième quartile (75%)
- * la valeur max
- La variance mesure la dispersion par rapport à la moyenne. Elle prend une valeur grande si les éléments de la distribution sont généralement éloignés de la moyenne ou une valeur petite si ces éléments sont au contraire resserrés près de la moyenne.
Calcul de la variance : *var(vect1)*
Calcul de l'écart-type : *sd(vect1)*

1.7 Manipuler et comparer plusieurs vecteurs

- Faire l'union de deux ensembles : *union(vect1, vect2)*
union(names(vect1), names(vect2)) (cette fonction retourne l'ensemble des éléments se trouvant dans au moins un de ces vecteurs)
- Faire l'intersection des ensembles pour savoir quels sont les éléments présents dans les tous les vecteurs en argument : *intersect(vect1, vect2)*
- Isoler les éléments qui ne sont pas communs aux vecteurs placés en argument : *setdiff(x, y)* (renvoie les éléments de x qui ne sont pas dans y)
Cette fonction est asymétrique c'est à dire que l'ordre des arguments modifie le résultat en sortie.
- L'opérateur *%in%* renvoie un vecteur d'éléments logiques de la même taille que le premier vecteur référencé à son appel : *x %in% y*
Utilisation : *vect1[names(vect1) %in% vect2]*
- Ordonner les vecteurs de manière réciproque : certaines fonctions ont besoin de vecteurs qui sont ordonnés dans le même ordre.
 - Commencer par avoir un vecteur de nom obtenu par :
commun = intersect(names(vect1), names(vect2))
 - Utiliser ce vecteur de nom commun pour récupérer dans le même ordre les éléments de *vect1* et *vect2*
vect1-2 = vect1[commun]
vect2-2 = vect2[commun]
Ainsi les deux vecteurs *vect1-2* et *vect2-2* sont ordonnés de la même manière avec le même nombre d'éléments.
- Trier ses données. Lorsque l'on veut trier un des vecteurs, on veut garder la réciprocité de l'ordre des éléments avec l'autre vecteur. Pour cela on veut récupérer les indices des éléments qui sont triés.
 - Utiliser la fonction *order()*. Utiliser le même vecteur avec *order* :
vect1-tri = vect1-2[order(vect1-2)]

$vect2-tri = vect2-2[order(vect1-2)]$

On obtient alors deux vecteur dont les éléments sont classés et dans même ordre.

1.8 Corrélations et régression linéaire

- Calculer la corrélation entre deux variables revient à calculer l'intensité de leur liaison. Ce coefficient est compris entre -1 et 1. Plus le coefficient est proche de ces valeurs extrêmes, plus la liaison est forte. Si à l'inverse le coefficient est proche de 0, la liaison est faible ce qui veut aussi dire que les données sont linéairement indépendantes.
- Il est possible que ces données soient corrélées de manière non linéaire.
 - Un coefficient entre 0 et 1 indique que les deux séries de données évoluent dans le même sens.
 - Inversement, si le coefficient est compris entre -1 et 0.
- Calculer un **coefficient de corrélation linéaire** : $cor(x,y)$
Attention : x et y doivent être de la même taille et contenir des éléments numériques
Les deux données doivent être réciproques (premier élément de x correspond au premier élément de y et ainsi de suite)
- Calcul des coefficients de la **régrétion linéaire** $y = a*x + b$
Utiliser la fonction : $lm()$
Cette fonction prend comme premier argument une formule : $formula = y \sim x$
(étudier le comportement de y en fonction de x)