Niveau : M1 IS
Cours : Introduction au logiciel R
Enseignant : Aude Illig

Université Versailles Saint-Quentin
Année 2014/2015 - 1er semestre

# TP 1 : Introduction au logiciel R

**** Manipulation des objets* `vector` *et* `factor` *****

## Table des matières

# 1 A propos de R

R est un logiciel *libre* dédié à l'analyse statistique qui offre des facilités graphiques considérables. R est à la fois un logiciel et un language de programmation. Il peut être vu comme une implémentation du language S développé par les laboratoires Bell et à la base du logiciel S-Plus vendu par TIBCO Software Inc. Contrairement à S-Plus, R est un logiciel *gratuit* qui est téléchargeable directement sur le site internet

`http://cran.r-project.org/`

où se trouvent également les instuctions à suivre lors de l'installation pour chaque système d'exploitation (Mac OS X, Windows, GNU/Linux,...). Le logiciel R a initialement été développé dans les années 1990 par deux statisticiens : Robert Gentelman et Ross Ihaka de l'Université d'Auckland en Nouvelle Zélande. Depuis, de nombreux mathématiciens ont rejoint la "R Developpement Core Team" qui tient à jour un site internet

`http://www.r-project.org/`

où sont notamment rassemblées de nombreuses informations bibliographiques sur le logiciel R. Ainsi, R est un logiciel en constante évolution comme l'atteste le nombre croissant de nouvelles librairies ou extensions (aussi appelées *packages* en anglais) développées parallèlement aux avancées statistiques (et mathématiques en général) et mises à la disposition de tous sur le site du projet.

*Remarque.* Vous pouvez aussi consultez la page `Wikipédia` de R pour savoir ce que l'on y dit de R...

Il n'est pas nécessaire d'acheter d'ouvrages sur R car de nombreux tutoriels et polycopiés sont disponibles en ligne. A la fin de ce document, sont listés certains de ces polycopiés dont l'orientation sur le type d'usage de R dans le document est précisé après la flêche ondulée.

# 2 Editeur de texte

## 2.1 Editeur par défaut

Sous Mac OS X et Windows, R intègre un éditeur de texte basique. Ainsi, la commande

`Ctrl R`

permet d'exécuter la ligne de commandes R où se trouve le curseur ou bien le bloc sélectionné du fichier *script* utilisé. Si le fichier *script* se nomme `fichier.R`, on peut également taper directement dans R la commande

```
source("fichier.R")
```

afin d'éxécuter les lignes de commandes du `fichier.R`. Assurez vous que vous connaissez l'endroit où R enregistre les fichiers source du type `fichier.R`. Par exemple sous Windows, sélectionnez l'onglet *file*, puis *Change dir....* Vous pouvez également spécifier le répertoire source à l'aide de la commande `setwd()`.

## 2.2 Installation d'un autre éditeur

Il est également possible de choisir d'installer un autre éditeur plus efficace notamment lorsque vous vous lancerez dans la programmation. De nombreux éditeurs intelligents comme emacs, Rstudio ou encore TINN-R sont disponibles gratuitement. Ils vous permettront de gagner du temps

tout en en vous facilitant l'écriture du programme par l'utilisation de couleurs, de l'indentation automatique et de détection de `)` ou accolades `}` manquantes. Il vous faudra installer R avant de pouvoir l'utiliser depuis votre nouvel éditeur de texte.

Pour avoir un aperçu de l'utilisation de R avec EMACS, vous pouvez consultez la référence [1]. Si vous souhaitez plutôt installer TINN-R, le document [4] est très complet et met l'accent sur les problèmes (et solutions) que l'on peut rencontrer avec TINN-R.

# 3 Session introductive

Lorsque vous lancez R, vous constatez que le prompt en début de ligne est
- `>` lorsque R est en attente d'une commande,
- `+` lorsque la ligne tapée précedemment est incomplète (à dessein ou parce que vous avez oublié une `)` ou une accolade `}` par exemple).

*Remarque.* A ce stade du TP, vous pouvez :
- soit vous reporter directement aux paragraphes 3.4 et 3.5 de ce document pour taper quelques commandes en ligne afin d'obtenir une vision plus intuitive de R puis revenir ensuite aux paragraphes précédents afin de structurer votre connaissance,
- soit continuer linéairement la lecture de ce document.

## 3.1 Obtenir de l'aide

Pour obtenir des renseignements sur une fonction R donnée, par exemple la fonction `plot`, on peut faire appel à l'aide en ligne de R

```
> help(plot)
```

ou bien

```
>?plot
```

La commande

```
> help.search("plot")
```

ou alternativement

```
>??plot
```

va rechercher toutes les fonctions R pour lesquelles le mot-clef plot est utilisé dans le descriptif de cette fonction. Vous remarquerez qu'à la fin de l'aide apparaît presque toujours un exemple illustrant l'usage de la fonction d'intérêt. L'exemple de la fonction `plot` peut être lancé par la commande

```
> example(plot)
```

Enfin, la plupart des installations de R offrent une aide au format HTML disponible à l'aide de la commande

```
> help.start()
```

## 3.2 Utilisation de librairies

Pour des études (statistiques) spécifiques, il peut être nécessaire de faire appel à une ou plusieurs librairies (aussi appelées *packages*) de R bien précise(s). La commande

```
> library()
```

affiche ainsi les *packages* actuellement installés mais non nécessairement chargés au démarrage de R. Pour faire de l'analyse de survie, il peut être utile de charger le *package* `survival` :

```
> library(survival)       # Charge le package survival
> library(help=survival)  # Affiche la liste des fonctions disponibles du package
> search()                # Affiche la liste des packages chargés
```

*Remarque.* Il est également possible de télécharger de nouveaux *packages* et de les charger ensuite dans R.

## 3.3 Permanence des objets

R est un language *objet*. Les objets de R peuvent être de nature très différentes : vecteurs, tableaux, matrices, fonctions... Au cours d'une session R, vous pouvez à tout moment lister les noms des objets créés et enregistrés dans l'espace de travail en tapant

```
> objects() # Ou alternativement ls()
```

*Remarque.* Vous remarquerez que le dièze (ou *hashmark* en anglais) permet d'écrire des commentaires.

Les objets s'effacent en utilisant la commande `rm`. Ainsi,

```
> x=3           # Crée un objet x de mode numérique de longueur 1 contenant le nombre 3
> rm(x)         # Efface l'objet du nom x
> rm(list=ls()) # Efface tous les objets en mémoire
```

Enfin, à la fin de chaque session, il vous est proposé de sauver une image de la session. Les objets créés sont sauvergardés dans le fichier `.RData` et les lignes de commandes exécutées dans le fichier `.Rhistory`.

## 3.4 Opérations élémentaires

Par exemple, vous pouvez utiliser R comme une calculatrice en tapant

```
> 5+26
```

Lorsque vous tapez sur la touche *Entrée*, le résultat apparaît :

```
> 5+26
[1] 31
```

Le résultat est précédé de `[1]` pour indiquer à l'utilisateur que le résultat qui suit les crochets est le premier (ici le seul) des résultats attendus. Certaines commandes renvoient plusieurs résultats :

```
> options(width=40)
> 1:20
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
[13] 13 14 15 16 17 18 19 20
```

Le résultat est la suite des entiers de 1 à 20 écrite sur deux lignes : le premier resultat (le chiffre 1) suit le crochet `[1]` et le treizième (le nombre 13) suit le crochet `[13]`. Quelques exemples supplémentaires à propos de la calculatrice de R :

```
> # Les opérations élémentaires sur les scalaires
> # sont *, +, -, / et ^
> 5 *3
[1] 15
> 3-8
[1] -5
> 12/2
[1] 6
> 2^{-3}
[1] 0.125
```

## 3.5 Illustration graphique

Terminons cette introduction par un exemple graphique :

```
> search()              # Le package graphics est déjà chargé mais
> attach(cars)          # les données cars ne sont pas encore attachées
> colors()              # Affiche la liste des couleurs disponibles
> x11()                 # Ouvre un dispositif graphique
> par(bg="turquoise1") # Permet de choisir la couleur du fond d'écran
>                       # Tracé du graphique :
> plot(speed,dist,type="p",pch=25,
+ col="violetered2",col.main="blue",
+ col.axis="blue",xlab="Vitesse",
+ ylab="Distance de freinage",
+ main="Joli graphique")
>                       # Enregistrement du graphique au format pdf :
> dev.copy2pdf(file="JoliGraph.pdf")
> detach(cars)          # Détache les données cars
> search()
```

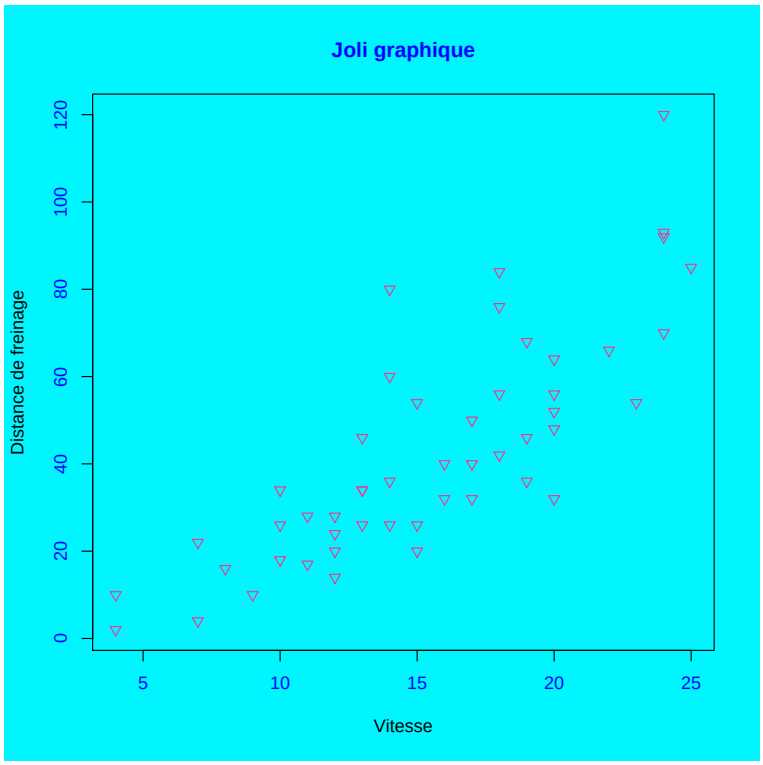


FIGURE 1 – Le joli graphique

# 4 Les objets de R

## 4.1 Objets, modes et longueurs

Le logiciel R manipule plusieurs *objets* : `vector, factor, array, matrix, data.frame, list, time-series` décrits dans le tableau TAB 1. Chaque objet dispose de deux attributs intrinsèques : *mode*, *longueur*. Le *mode* correspond au type des éléments d'un objet. Il existe quatre *modes* principaux : `numeric, complex, logical, character`.

Les commandes `mode(x)` et `length(x)` affichent le *mode* et la *longueur* de l'objet `x` :

```
> x=c(0,1,-9,7)          # Affectation classique pour un vecteur
> x + 1                  # Réplication du scalaire 1 à la longueur de x
> mode(x)                # Affiche le mode de x, ici numeric
> length(x)              # Affiche la longueur de x
> class(x)               # Si non spécifiée, class est égal au mode
> dim(x)=c(2,2)
> class(x)               # On a changé la class de x
```

Afin de savoir si `x` est par exemple un vecteur ou une matrice, on utilise les fonctions `is.vector(x)` ou `is.matrix(x)` :

```
> bool=logical(0)        # Crée un vecteur "vide" de type booléen
> bool[2]=is.matrix(x)   # R ajuste la taille du vecteur bool
> print(bool)            # NA indique une case non affectée
> bool[1]=is.vector(x)
> bool
> mode(bool)
> length(bool)
```

On peut également contraindre `y` à être d'un type `xxx` à l'aide de la commande `as.xxx(y)` :

```
> y=3.4
> is.complex(y)
> as.complex(y)
> as.character(y)
```

La commande `ls.str()` permet d'afficher les différents objets en mémoire et leurs caractéristiques. Par exemple, après avoir enregistré les différents objets du tableau TAB 1., la commande `ls.str()` affiche :

```
> ls.str()
A :  int [1:5, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
bool :  logi [1:2]  TRUE FALSE
B :  num [1, 1:4] 1 -1 8 3
C :  num [1:8, 1:2] 0 0 0 0 0 0 0 0 0 0 ...
Donnees : 'data.frame': 4 obs. of  2 variables:
 $ D1: num  0 1 -9 7
 $ D2: int  4 5 6 7
f :  Factor w/ 4 levels "1","2","3","5": 1 2 3
L : List of 3
 $ : num [1:4] 0 1 -9 7
 $ : num 3.4
```

| Objet | Description | Exemples |
|---|---|---|
| `vector` | Vecteur au sens classique (éléments de même *mode*) | `> x=c(0,1,-9,7)`<br>`> y=3.4`<br>`> text=c("petit","grand")` |
| `factor` | Variable catégorique (éléments de *modes* identiques : `numeric` ou `character`) | `> f=factor(1:3,levels=1:5, exclude=4)` |
| `array` | Tableau $k$-dimensionnel (éléments de même *mode*) | `> A=array(1:20,dim=c(5,4))` |
| `matrix` | Cas particulier de `array` avec $k = 2$ | `> B=matrix(c(1,-1,8,3), nrow=1,ncol=4)`<br>`> C=matrix(0,nrow=8,ncol=2)` |
| `data.frame` | Tableau de données composé d'un ou plusieurs vecteur(s) et/ou facteur(s) ayant même longueur mais des *modes* pouvant être différents | `> Donnees=data.frame(D1=x, D2=y)`<br>`> Donnees$D1` |
| `ts` | Données de type série temporelle | `> Serie1=ts(z,start=1942)`<br>`> Serie2=ts(z,freq=12, start=c(1942,3))` |
| `list` | Liste d'éléments de tout *mode* | `> L=list(x,y,text)` |

TABLE 1 – Les objets de R

```
 $ : chr [1:2] "grand" "petit"
Serie1 :  Time-Series [1:26] from 1942 to 1967: -3.0 -2.8 -2.6 -2.4 -2.2 ...
Serie2 :  Time-Series [1:26] from 1942 to 1944: -3.0 -2.8 -2.6 -2.4 -2.2 ...
t :  num [1:8] 0 1 -9 7 0 1 -9 7
text :  chr [1:2] "grand" "petit"
x :  num [1:4] 0 1 -9 7
y :  num 3.4
```

## 4.2 Manipulations simples sur les objets `vector` et `factor`

La structure la plus simple utilisée par R est celle des vecteurs numériques. Pour assigner une valeur (*attribute* en anglais) à un vecteur, il existe plusieurs méthodes :

```
> x=c(0,1,-9,7)
> x<-c(0,1,-9,7)
> c(0,1,-9,7)-> x
> assign("x", c(0,1,-9,7))
```

Les commandes `seq()` ou `rep()` sont dédiées à la génération de suites régulières :

```
> seq(from=1,to=10,by=1)
> seq(2,-2,0.1)
> rep(1,times=5)
> rep(x,each=3)
```

Les opérations élémentaires `+`,`-`,`*`,`/` et `^` ainsi que les opérations classiques du type `log` ou `exp` s'éffectuent élément par élément :

```
> u<- x/3+2*y
```

```
> v<-log(x)
> y<-3.4
> sqrt(-y)
> as.complex(y)
> sqrt(-y)
```

D'autres manipulations peuvent être effectuées sur les vecteurs :

```
> range(x)   # Identique à c(min(x),max(x))
> sum(x)     # Somme des éléments de x
> prod(x)    # Produit des éléments de x
> mean(x)    # Identique à sum(x)/length(x)
> sd(x)      # Identique à sum((x-mean(x))^2)/(length(x)-1)
> summary(x) # Pour plus d'informations statistiques ...
> sort(x)    # Ordonne les éléments de x de manière croissante
```

L'extraction de sous-vecteurs est très souple comme le montre les exemples suivants :

```
> x[1:2]     # Vecteur constitué des 2 premières composantes de x
> x[-(1:2)]  # Vecteur formé des composantes de x exeptées les 2 premières
> x[x>3]     # Vecteur constitué des composantes d'indice >3
> x[c(1,4)]  # Vecteur composé de la 1ère et de la dernière compostante de x
```

R manipule également les vecteurs de mode `logical`

```
> x>0             # Les autres opérateurs logiques sont <, <=, >=, ==, !=
> as.numeric(x>0) # Transformation d'un vecteur logique en vecteur numérique
> (x>=0)&(x<=0)   # & pour "et"
> (x>0)|(x<0)     # | pour "ou"
```

Les vecteurs de mode `character` sont également très utilisé en R, notamment pour donner un nom à des objets ou aux axes d'un graphique.

```
> nom.taille<-c("petit","moyen","grand")
> taille<-c(10,3,15)
> names(taille)=nom.taille
> effectif.petit=taille["petit"]
```

Le type `factor` de R permet également de manipuler aisément les variables qualitatives.

```
> v=c(1, 2, 2, 1, 2)           # Vecteur de données catégorielles
> sexe=factor(v)               # 1 pour masculin
> print(sexe)                  # 2 pour féminin
> is.factor(sexe)
> print(levels(sexe))
> levels(sexe)=c("masc","fem") # On renome les niveaux et R comprend
> print(sexe)                  # que 1="masc" et 2="fem"
```

# 5 Exercice d'application

La valeur actuelle $P$ d'une série de $n$ mensualités $M_1, M_2, \ldots, M_n$ est

$$P = \sum_{j=1}^{n} \left\{ \prod_{k=1}^{j} (1+i_k)^{-1} \right\} M_j$$

où $i_k$ désigne le taux d'intérêt appliqué pour le mois $k$.

1. Quel est le montant d'un prêt remboursable sur une année dans les cas listés ci-dessous ?
   (a) Mensualités de 1000 euros et taux d'intérêt constant $i = 5\%$.
   (b) Mensualités de 1500 euros les 6 premiers mois, mensualités de 500 euros les 6 derniers mois et taux d'intérêt constant $i = 5\%$.
   (c) Mensualités de 500 euros les 6 premiers mois, mensualités de 1500 euros les 6 derniers mois et taux d'intérêt constant $i = 5\%$.
   (d) Mensualités de 1000 euros, taux d'intérêt de $4,5\%$ les 6 premiers mois et de $5.5\%$ les 6 derniers mois.
2. Un personne souhaite emprunter 10000 euros remboursable sur une année avec des paiements mensuels constants. Si le taux d'intérêt mensuel est constant égal à $i = 5\%$, quel est le montant des mensualités ?

# Références


[1] Christophe Genolini, *R, Bonnes pratiques* (français)
⇝ Document qui propose des règles simples pour écrire proprement un programme et éviter les erreurs classiques

[2] Vincent Goulet, *Introduction à la programmation en R* (français)
⇝ Document orienté vers la programmation en R

[3] Emmanuel Paradis, *R pour les débutants* (français)
⇝Document qui détaille entre autre les fonctionnalités graphiques de R

[4] Ricco Rakotomalala, *TINN-R* (français)
⇝ Document destiné à l'installation de l'éditeur TINN-R mettant l'accent sur les principaux écueils

[5] W. N. Venables, D. M. Smith et the R Core Team, *An introduction to R* (english)
⇝ Document fournissant une vue d'ensemble des diverses fonctionnalités du logiciel R