TP1 et 2 : le logiciel R

1 Introduction

R est un logiciel de calcul statistique, libre et gratuit. Il comporte un langage de programmation et un environnement mathématique utilisés pour le traitement des données. Il permet de faire des analyses statistiques aussi bien simples que complexes, comme l'étude de modèles linéaires ou non-linéaires, des tests d'hypothèse, de la modélisation de séries chronologiques, de la classification, etc ... Il dispose également de nombreuses fonctions graphiques très utiles et de qualité professionnelle.

On trouve de nombreuses FAQ consacrées à R sur internet.

Comment installer R chez soi ?

R fonctionne avec plusieurs systèmes d'exploitation, en particulier Microsoft Windows, MacOS X et Linux. Il existe plusieurs interfaces graphiques, RGUI est l'interface graphique installée par défaut sous Windows (Graphical User Interface). C'est celle que nous utiliserons ici en TP.

Pour installer R :
- allez sur le site `https://cran.r-project.org/mirrors.html`, puis choisissez un site miroir français
- téléchargez la version correspondant à votre système d'exploitation

Par la suite, selon les traitements statistiques souhaités, il sera peut-être nécessaire d'installer des packages supplémentaires (ce sont des bibliothèques de fonctions déjà écrites), comme nous le verrons en TP.

Pour lancer le logiciel, cliquez sur le raccourci R que vous devez maintenant avoir sur votre bureau.

Comment utiliser R en salle de TP ?

R est déjà installé sur les machines. Pour le lancer :
Cliquez dans **Démarrer →Tous les programmes →Mathématiques →R →R 2.9.1**

Une fenêtre s'ouvre en attente d'instructions. La ligne commence toujours par >.

R est un langage orienté-objet qui est interprété (non compilé), sans déclaration obligatoire des variables, et basé sur la notion de vecteurs. C'est à la fois une calculatrice, un puissant logiciel de statistiques, et un éditeur graphique.

2 Premiers pas

- Attention : R fait la différence entre majuscules et minuscules (`Rennes` est différent de `rennes`)

- Pour ajouter un commentaire (non interprété) aux instructions, on utilise le `#`
- Pour affecter une valeur à un objet, on utilise l'un des opérateurs `<-` ou `=`
  Le nom d'un objet doit commencer par une lettre, et ne peut comporter que des lettres, des chiffres des _ ou des points.

Exemples (testez) :

```
> a<-2.5   # crée l'objet a en lui donnant la valeur 2.5
> a        # affiche la valeur de a
> a_bis<-a # l'objet a_bis reçoit la valeur de a
> b=a      # l'objet b reçoit la valeur de a
> b=7      # modifie la valeur de b
> a+b      # calcule la somme de a et b
> 2+pi     # essayez ...
> x=T      # x est un objet de type booléen qui vaut "vrai"
> y=F      # y est un objet de type booléen qui vaut "faux"
> x&y      # "et" logique
> x|y      # "ou" logique
```

L'opérateur logique `==` permet de tester l'égalité entre deux objets et renvoie une valeur booléenne. L'opérateur "est différent de" est `!=`

Attention : comme toutes les calculatrices, R fait des calculs *approchés* sur les réels. Testez les instructions suivantes :

```
> a<-5/10-3/10
> b<-3/10-1/10
> a
> b
> a==b
> a-b
```

Les vecteurs

Un vecteur est un objet composé d'une ou de plusieurs valeurs (appelées *éléments*, *coordonnées* ou *composantes*) de même type.

Il y a plusieurs façons de définir un vecteur, la plus générale étant d'utiliser la commande `c` (comme "combinaison"). Observez les vecteurs obtenus par les commandes suivantes :

```
> (v1<-c(7,-2.5,4.2))  # v1 est un vecteur numérique à 3 composantes.
> v2<-c(v1,3,c(12,-8))  # v2 est un vecteur numérique à 6 éléments.
> v2
> 12:76             # vecteur de composantes les entiers de 12 à 76
> seq(1,4,by=0.5)   # séquence de 1 à 4 avec un pas de 0.5
> seq(1,6,length=5) # vecteur à 5 composantes équiréparties entre 1 et 6
> rep(1,50)         # répétition de la même valeur
> c(T,T,F,F,T)      # vecteur booléen
> !c(T,T,F,F,T)
> c("bleu","vert","marron")  # vecteur chaîne de caractères
```

Manipulations sur les coordonnées des vecteurs (testez) :

```
> v2[3]       # la 3ème coordonnée de v2
> v2[-3]      # les coordonnées de v2 sauf la n°3
> v2[3:5]     # les coordonnées n°3, 4 et 5 de v2
> v2[c(3,6)]  # les coordonnées n°3 et 6 de v2
> v2[v2>6]    # les coordonnées > 6 de v2
> v2[3]<-20   # la coordonnée n°3 de v2 prend la valeur 20
> v2[v2<0]<-0 # remplace toutes les coordonnées négatives de v2 par 0
```

Les matrices

Une matrice est un tableau à deux dimensions constitué d'éléments de même type. Chaque élément est repéré par son numéro de ligne et son numéro de colonne : `m[i,j]` est l'élément situé sur la i^e ligne et la j^e colonne de la matrice `m`.

On définit une matrice avec la fonction `matrix`. Le nombre d'éléments doit être multiple du nombre de colonnes (ou de lignes).

Exemples (testez) :

```
> m<-matrix(c(5,0,4,6,1,12),ncol=2)
> m
> m<-matrix(c(5,0,4,6,1,12),nrow=2)
> m
> m<-matrix(c(5,0,4,6,1,12),byrow=T,nrow=2) # pour ranger les éléments par ligne
> m
        # analysez bien les résultats suivants :
> m[1,3]
> m[1,]
> m[,3]        # observez que le vecteur colonne s'affiche en ligne
> m[1:2,-1]
> m[,m[1,]>=4]
```

3 Les fonctions

R dispose de toutes les fonctions mathématiques usuelles.

Sur le vecteur `x=c(1,-1,4.7,2)`, essayez les fonctions suivantes :

```
2*x, x+2:5, x^2, abs(x), exp(x), floor(x), sort(x), rev(x), order(x), x>3, x==2
sum(x), prod(x), min(x), max(x), which.min(x), length(x), mean(x), min(x)==4
```

La fonction racine carrée est `sqrt()`. Que se passe-t-il si on l'applique à `x` ? (NaN signifie "not a number").

Essayez `sqrt(abs(x))`.

R contient de nombreuses fonctions prédéfinies, mais on peut aussi écrire soi-même des fonctions avec R. La syntaxe générale est la suivante :

```
nom_fonction<-function(arg1[=expr1],arg2[=expr2],...){bloc d'instructions}
```

Exemple : somme des n premiers entiers

```
> som1n <-function(n){ sum(1:n) }
> som1n(5)
[1] 15
```

☞ Pour avoir l'aide sur une fonction prédéfinie, en particulier pour connaître les arguments possibles, tapez `help(<nom de la fonction>)`.

Exemple de la fonction `rnorm` :

```
> rnorm(10)   # génère 10 variables aléatoires normales centrées réduites
              # par défaut la moyenne vaut 0 et l'écart-type 1
> rnorm(10,mean=2,sd=0.5)   # modification de la moyenne et de l'écart-type
```

Exercice

Ecrire une fonction `variance` qui calcule la variance d'une série statistique s donnée en paramètre.

Comparez votre résultat avec celui donné par la fonction `var()` de R (prenez par exemple pour `s` les entiers de 1 à 10).

☞ Pour pouvoir sauvegarder et modifier facilement votre travail, il est conseillé d'écrire vos commandes dans un script (Fichier →Nouveau script). Créez un dossier AD **dans votre dossier personnel**, vous y sauvegarderez ce script régulièrement.

On peut exécuter tout le script d'un coup (Edition →Exécuter tout), ou seulement une sélection ou la ligne courante par Ctrl+R.

4 Manipuler les données

Nous allons utiliser un premier jeu de données, sur lequel nous ferons des statistiques descriptives : importez dans votre dossier AD le fichier `N:l2ie/ad/mesures.txt` .
Si vous travaillez sur votre machine personnelle, vous trouverez ce fichier dans le dossier suivant : `https://share-etu.istic.univ-rennes1.fr/l2ie/ad`.

Ensuite, dans R placez-vous dans votre dossier AD (avec Fichier →Changer le répertoire courant) pour pouvoir utiliser ce fichier.

Pour vérifier le nom du répertoire courant : `getwd()`.

Tapez ensuite `tab=read.table("mesures.txt", header=T)` dans votre script pour charger le fichier dans R (l'argument `header` signifie que la première ligne contient les noms des variables), puis `tab` pour l'afficher.

Cet objet est un *data frame* : c'est un tableau de données, dont les lignes représentent les individus, et les colonnes donnent les valeurs des variables (quantitatives ou qualitatives). Ici nous avons 66 individus et 3 variables (sexe, poids, et taille).

Chaque variable peut être accessible soit par son nom (avec "$"), soit par sa place dans le data frame. Pour la variable poids, tapez par exemple `tab$poi` ou `tab[,2]`.

Une fonction intéressante est la fonction "summary" (tapez `summary(tab)`) qui donne les statistiques sommaires sur les variables. Observez la différence entre variable qualitative et variable quantitative sur la forme des résultats.

Exercice

- Faites afficher l'écart-type des variables quantitatives (fonction `sd`).
- Il est parfois utile de sélectionner certains individus :
  - faites afficher les caractéristiques des individus n° 5, 10, et 20
  - faites afficher tous les individus de sexe féminin (pour connaître leur nombre, appliquer `nrow`)
  - faites afficher tous les individus qui mesurent plus de 175 cm
  - faites afficher le poids et la taille de tous les individus de sexe féminin qui mesurent plus de 175 cm
  - faites afficher les trois individus les plus lourds
- Calculer le poids moyen des femmes et celui des hommes.

5 Représenter les données

Les graphiques sont importants en statistiques et constituent souvent le point de départ d'une analyse de données. R permet de créer très facilement des graphiques très propres. Vous pourrez en avoir un aperçu en tapant `demo(graphics)`, qui donne le code et le résultat.

Une seule fenêtre graphique est ouverte *a priori* par R (tout nouveau graphique écrase le précédent). Pour ouvrir une nouvelle fenêtre graphique, tapez la commande `x11()`.

5.1 Histogramme

La fonction qui permet de tracer un histogramme est `hist`. Il y a de nombreux paramètres possibles pour améliorer la présentation. Essayez les commandes suivantes :

```
histo=hist(tab$tai)   # le basique
# et plus agréable à lire :
hist(tab$tai, main="taille du panel",col="blue",xlab="taille en cm", ylab="effectif")
histo         # pour avoir le détail des valeurs numériques
colors()      # pour avoir les couleurs prédéfinies
```

Tracez l'histogramme des poids. Que remarquez-vous ?

Tracez maintenant un histogramme des poids des femmes, et un autre des poids des hommes. Vous pouvez les faire apparaitre sur la même fenêtre graphique en tapant au préalable `par(mfrow=c(1,2))`, ce qui signifie qu'on veut afficher les graphiques suivants sur 1 ligne et 2 colonnes. Variez les titres et les couleurs. Pour comparer les deux histogrammes, on peut imposer la même échelle en ordonnée. Ajoutez par exemple l'argument `ylim=c(0,18)` dans les fonctions `hist`.

On peut aussi utiliser l'argument `prob=T` pour obtenir l'histograme des fréquences (alors la surface totale des barres vaut 1).

5.2 Camembert

Le camembert est utilisé pour représenter une variable qualitative. La fonction qui permet de tracer un camembert est `pie`, et on peut ajuster plusieurs paramètres :

```
eff=table(tab$sexe)        # détermine l'effectif de chaque modalité
eff
pie(eff)
pie(eff, main="panel",col=c("pink","lightblue"),labels=c("femmes","hommes"))
pie(eff, main="nature du panel",col=c("pink","lightblue"),
         labels=paste(c("femmes","hommes"),"  ",eff*100/66, "%"))
pie(eff, main="nature du panel",col=c("pink","lightblue"),
         labels=paste(c("femmes","hommes"),"  ",round(eff*100/66,1), "%"))
```

5.3 Boxplot

Appelé aussi "boite à moustaches", le boxplot permet de visualiser la dispersion d'une variable quantitative, et il permet aussi (et surtout) de comparer un même caractère observé dans des populations différentes. La fonction R réalisant le tracé est `boxplot` :

```
par(mfrow=c(2,2)) # pour avoir les 4 graphiques dans la même fenêtre
b=boxplot(tab$poi,main="poids tous")
b$stats   # donne le détail des valeurs caractéristiques du boxplot
boxplot(tab$tai,main="taille tous")
boxplot(tab$poi~tab$sexe,main="poids selon le sexe")
boxplot(tab$tai~tab$sexe,main="taille selon le sexe",
                         col=c("pink","lightblue"),names=c("femmes","hommes"))
```

Observez les graphiques obtenus. Que constate-t-on ?

5.4 Nuage de points

On utilise la fonction `plot` de R. Elle possède de nombreux arguments possibles, comme nous le verrons par la suite. Pour tracer un nuage de points, on exécute `plot(x,y)`, où `x` est le vecteur abscisse et `y` le vecteur ordonnée. Essayez par exemple :

```
plot(0:50,sqrt(0:50))
plot(0:50,sqrt(0:50), main="fonction racine carrée", col="red", ylab="",
                      xlab="x entier de 0 à 50")       # avec quelques options
```

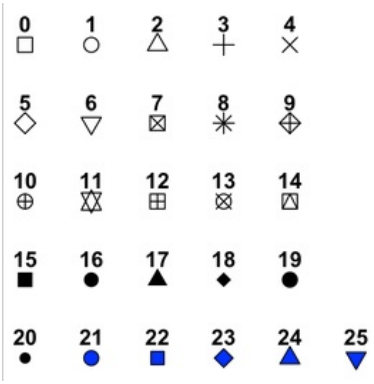
Tracez le nuage de points de la taille en fonction du poids.

Vous pouvez ajouter l'option `col=as.numeric(tab$sexe)` pour séparer les hommes des femmes, puis exécuter `legend("topleft",legend=c("femmes","hommes"),fill=c(1,2))` pour ajouter une légende.

Autre présentation plus sophistiquée :

```
plot(tab$poi,tab$tai, main = paste("Poids et taille de",nrow(tab), "individus"),
      xlab = "Poids", ylab = "Taille",col = c("pink", "blue")[as.numeric(tab$sexe)],
      pch = c(17,20)[as.numeric(tab$sexe)], cex = 1.5, las = 1)
legend("topleft", c("Femmes", "Hommes"), pch = c(17, 20), col = c("pink","blue"))
```

Examinez l'effet des différentes options : affichage du titre comprenant une variable, `cex` correspond à la taille des points, `pch` à leur forme, et `las` permet de changer le sens de la graduation sur l'axe (0y). Les valeurs de `pch` (et les formes associées) sont données ci-dessous :



5.5 Courbes

La fonction `plot` de R permet également de tracer des fonctions continues. Tapez les commandes suivantes :

```
x=seq(-pi,pi,length=8)
plot(x,sin(x),type="l")  # fonction sinus entre -pi et +pi
x=seq(-pi,pi,length=100)
plot(x,sin(x),type="l",main="fonction Sinus entre -pi et +pi",lwd=5) #avec plus d'options
lines(c(-3.2,3.2),c(0,0))
```

`lwd` permet de modifier la largeur du trait.

Quelques lois de probabilités :

- Loi normale
  C'est une loi fondamentale pour faire des statistiques, comme nous le verrons plus tard. Sa densité est donnée par $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$, où μ est son espérance et σ son écart-type.

  ```
  # densité de la loi normale centrée réduite
  x=seq(-5,5,length=100)
  plot(x,dnorm(x),type="l", main="densité de la loi normale N(0,1)")
  ```

  Faites varier l'écart-type (de 1 à 5) et ajoutez les courbes sur le même graphique (utilisez `lines` au lieu de `plot` pour les suivantes). Ajoutez une légende. Notez que l'aire sous la courbe vaut toujours 1. Combien vaut le maximum pour chaque courbe ?

☞ Un résultat fondamental en statistiques est qu'environ 95% des valeurs se trouvent dans l'intervalle $[\mu - 2\sigma; \mu + 2\sigma]$.

- Loi exponentielle

  La densité de la loi exponentielle de paramètre $\lambda > 0$ est : $f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}$

```
  # densité de la loi exponentielle de paramètre 3
x=seq(0,3,length=100)
plot(x,dexp(x,rate=3),type="l", main="densité de la loi exp de paramètre lambda=3")
```

  Faites varier le paramètre. L'aire sous la courbe vaut toujours 1.

6 Les séries temporelles (ou chronologiques)

Une série temporelle est une collection de données où une variable quantitative évolue en fonction du temps. Des phénomènes saisonniers ou ponctuels apparaissent. Il y a de nombreuses applications dans le domaine météorologique, financier, ou dans le suivi de populations.

Exemple avec l'évolution du nombre de passagers d'une compagnie aérienne de 1949 à 1960 :

```
data(AirPassengers)  # chargement des données
AirPassengers        # visualisation des données
plot(AirPassengers,xlim=c(1950,1962))  # graphique
p=HoltWinters(AirPassengers)
lines(predict(p,n.ahead=12),col=5,lwd=2) # prédiction pour l'année suivante
```

L'objectif est de faire des prévisions fiables, nous verrons comment dans la suite.

==========

Vous connaissez maintenant l'essentiel sur le logiciel R. N'oubliez pas de sauvegarder votre script avant de quitter.