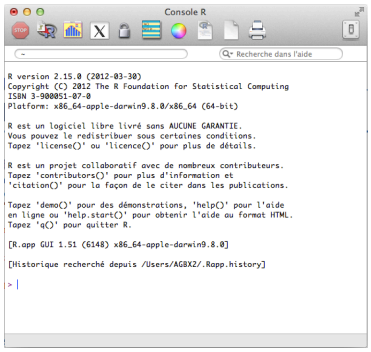
Sommaire

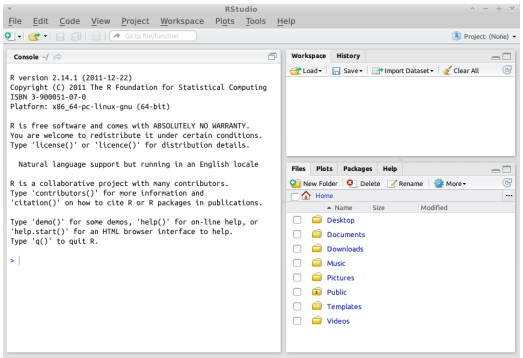
Comment installer R ?

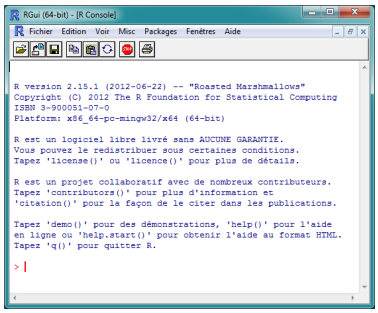
Allez sur le site http ://www.r-project.org/ et cliquez sur download R. Il vous est ensuite demandé de choisir un site miroir du CRAN[1] (http://cran.univ-lyon1.fr/ par exemple) puis de sélectionner le téléchargement correspondant à votre système d'exploitation (Mac, Linux ou Windows).
Pour une installation simplifiée, sélectionnez ensuite le "*base package*" qui contient les "*binaries*". Les liens directs vers les dernières versions sont ici pour Mac OS et ici pour Windows. Pour Linux il faut d'abord choisir sa distribution puis la version de sa distribution mais pour Ubuntu par exemple il est plus simple de passer par un gestionnaire de paquets type *synaptic* et de chercher *r-base* dans la base de données. L'installation en est grandement simplifiée et se fait en quelques clics.

Comment lancer R ?

Pour ce document, nous utilisons le GUI[2] de R. Pour le lancer il suffit d'aller dans les applications (pour Mac) dans le menu démarrer (pour Windows) ou de taper R dans une console (pour Linux) En ce qui concerne Linux il existe au moins 2 modes de consoles graphiques : RKward et RStudio qui rendent son utilisation beaucoup plus conviviale que la ligne de commande standard.







Console Mac | Console RStudio | Console Windows

Comment installer des packages ?

En utilisant le GUI aller dans Package & données // Installateur de package. Par défaut la sélection CRAN (binaires) est active et fonctionne la plupart du temps. Il faut ensuite cliquer sur Acquérir liste. R propose alors de choisir un site miroir, par exemple : France (Lyon 1). Vous pouvez ensuite entrer le nom du package directement pour une recherche ou parcourir la liste proposée. Une fois le package choisi il est conseillé de cocher la case Installer les dépendances. Puis il faut valider par Installer/Mettre à jour. Le package est alors installé de façon résidente et peut être appelé directement.

Comment activer des packages ?

Lors d'une session tous les packages ne sont pas chargés par défaut. Pour en utiliser un il faut donc l'activer (l'importer). Pour ce faire il faut aller dans Package & données // Installateur de package et choisir Gestionnaire de packages. Il suffit alors de cocher le ou les packages que l'on désire utiliser dans la session pour qu'ils se chargent automatiquement.

[1] Comprehensive R Archive Network
[2] Graphical user interface

Comment avoir de l'aide dans R ?

Aide générale : help.start(). S'ouvre dans le navigateur par défaut sous la forme de pages HTML
Aide sur une fonction connue : help(nom#de#fonction) ou bien ?nom#de#fonction. Affiche dans une fenêtre une documentation détaillée sur cette fonction. Exemple : ?cos
Aide sur une fonction non connue : si la commande précédente (?) ne trouve pas la fonction il est possible d'invoquer une recherche globale sur cette fonction avec la commande ??. Par exemple ?Cos ne fonctionne pas mais ??Cos renvoie les pointeurs de l'aide sur la fonction cos indépendamment de la casse.

Comment saisir des données ?

En ligne de commande ou à l'aide de scripts, R permet de créer des variables de plusieurs classes et des méthodes pour travailler sur les données. Pour créer une variable on utilise l'opérateur d'affectation "<-" (mais le signe"=" marche aussi).
La création d'une série quelconque se fait avec la fonction c() et les accès à l'aide des [] avec un numéro d'indice allant de 1 – N pour une série de longueur N.
Dans les lignes suivantes on crée une variable nommée arbitrairement set_1 en lui attribuant un scalaire, on l'affiche et on la transforme ensuite en vecteur que l'on modifie également.

```
> set_1<-1
> set_1
[1] 1
> set_1<-c(12,3,2,11,23,3,21)
> set_1
[1] 12  3  2 11 23  3 21
> set_1[2]<-15
> set_1
[1] 12 15  2 11 23  3 21
```

Comment générer des données automatiquement ?

Parmi les nombreuses commandes R disponibles, les plus simples utilisent l'opérateur " :" ou l'instruction seq()

- seq(length=, from=, by=) avec length : nombre d'éléments et from/by : valeurs de départ et d'incrément
- seq(from=,to=,by=) avec from/by : valeurs de départ et d'incrément et to valeur de fin

Les lignes suivantes permettent de créer des vecteurs

```
> set_1<-1:6
> set_1
[1] 1 2 3 4 5 6
> set_1<-6:1
> set_1
[1] 6 5 4 3 2 1
> set_1<-seq(from=1,to=10)
> set_1
 [1]  1  2  3  4  5  6  7  8  9 10
> set_1<-seq(from=1,to=10,by=2)
> set_1
[1] 1 3 5 7 9
```

```
> set_1<-seq(length=101,from=0,to=1)
> set_1
  [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07
  [9] 0.08 0.09 0.10 0.11 0.12 0.13 0.14 0.15
 [17] 0.16 0.17 0.18 0.19 0.20 0.21 0.22 0.23
 [25] 0.24 0.25 0.26 0.27 0.28 0.29 0.30 0.31
 [33] 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39
 [41] 0.40 0.41 0.42 0.43 0.44 0.45 0.46 0.47
 [49] 0.48 0.49 0.50 0.51 0.52 0.53 0.54 0.55
 [57] 0.56 0.57 0.58 0.59 0.60 0.61 0.62 0.63
 [65] 0.64 0.65 0.66 0.67 0.68 0.69 0.70 0.71
 [73] 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79
 [81] 0.80 0.81 0.82 0.83 0.84 0.85 0.86 0.87
 [89] 0.88 0.89 0.90 0.91 0.92 0.93 0.94 0.95
 [97] 0.96 0.97 0.98 0.99 1.00
```

Comment générer des séries aléatoires ?

Il est possible de générer directement des vecteurs contenant des valeurs qui suivent des distributions statistiques connues.
Loi normale : rnorm(N, moyenne, écart type)
Loi de Poisson : rpois(N,lambda)
Loi uniforme : runif(N, min, max)

```
> rnorm(8,0,1)
[1] -0.7698952  1.1057276 -0.2021818 -0.7903316 -0.3532141  2.0713235  1.0430033  0.3637724
> rpois(8,1)
[1] 1 0 1 2 1 0 1 1
> runif(8,0,3)
[1] 0.2498875 1.0620919 1.2637117 2.5192826 2.8494318 2.2646496 2.6704337 1.7846283
```

Comment charger des données ?

Sélection du répertoire courant

Mode ligne de commande

Obtenir le répertoire courant : getwd()
Définir le répertoire courant : setwd("path")

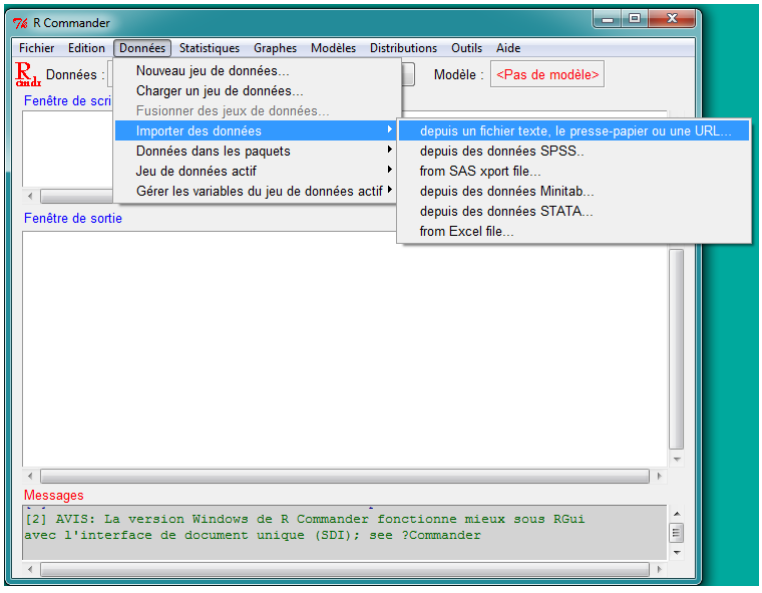
```
> getwd()
[1] "C:/Users/sam/Documents"
> setwd("C:/Users/sam/Desktop/")
> getwd()
[1] "C:/Users/sam/Desktop"
```

Mode menu du GUI

Aller dans Divers // Changer le répertoire de travail et le sélectionner ce qui est un peu plus rapide.

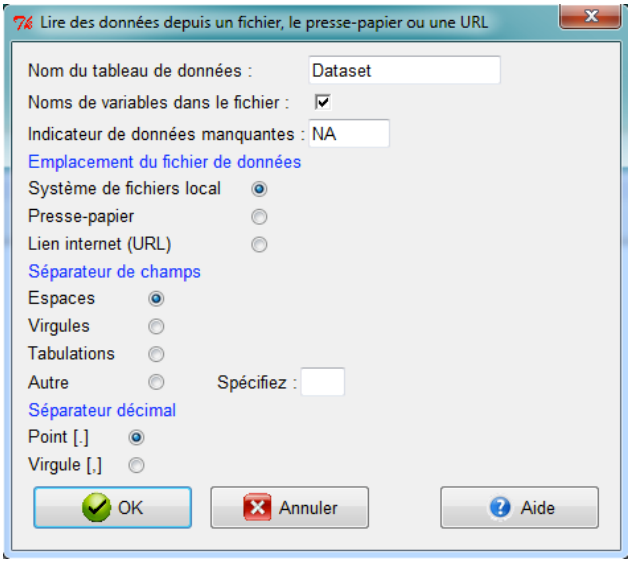
Une alternative pour importer des données: Rcmdr

Le package Rcmdr offre une alternative élégante à la ligne de commande pour un traitement statistique. Une présentation rapide de son utilisation pour l'importation de données est apportée ici mais pour des raisons de portabilité et de souplesse d'utilisation le reste du document détaille l'utilisation de commandes en ligne uniquement.
Une fois le package R Comander installé et exécuté il faut cliquer sur Données // Importer des données // Depuis un fichier texte, le presse-papier … comme le montre la figure suivante.



R Comander ouvre alors une fenêtre de dialogue (voir figure suivante) permettant de faciliter l'importation des données en spécifiant les paramètres du format de fichier. De haut en bas les champs à remplir comprennent:

- le nom qui sera donné à la variable dans l'espace de noms
- le fait que les noms de variables soient déjà inclus dans le fichier ou pas
- ce qui doit figurer si le tableau rencontre une case vide (ici NA par exemple)
- dans la rubrique "Séparateur de champs" l'utilisateur coche comment sont séparées les données entre les cellules (équivalent de l'argument sep de la commande read.table cf. paragraphe suivant)
- enfin il faut préciser si les données ont comme séparateur décimal un "." ou une ","



Commandes en ligne pour l'importation de données

Il existe plusieurs façons de procéder. Afin de simplifier et d'unifier la démarche nous présentons ici l'emploi d'une instruction générale en ligne de commande

read.table("nom du fichier", header=TRUE/FALSE)

header=TRUE si la première ligne contient les noms des colonnes.

```
> data_01<-read.table("DataTD2Ex1.txt",header=TRUE)
```

Remarque sur le format des fichiers de données

Nous travaillerons ici avec des données au format texte. Cela signifie qu'elles peuvent être facilement ouvertes avec n'importe quel éditeur de texte (Notepad, Vi ou Textedit par exemple) et être ainsi lues et modifiées. Même avec cette simplification le problème des formats de données et leur importation dans R n'est pas une question triviale. En règle générale les données que nous utiliserons seront bidimensionnelles et donc représentable sous la forme de matrices. La commande read.table est très souple et paramétrable et lira spontanément les fichiers dont les éléments sont par exemple séparés par des espaces ou des tabulations entre les colonnes. Par contre il faudra spécifier le cas échéant d'autres séparateurs type ";" ou "," par exemple il faudra ajouter l'argument sep=";" ou sep="," lors de l'appel de la fonction.

Editer graphiquement des données

La commande edit(nom_de_variable) ouvre une fenêtre permettant de modifier le contenu d'une variable

```
> edit(data_01)
```

Comment extraire des données des variables complexes ?

Organisation des variables complexes

La méthode read.table() retourne un objet de type data.frame. L'architecture de ce type d'objet serait trop longue pour ce type de résumé et le lecteur peut utiliser la commande ?data.frame pour obtenir quelques informations de base.

Sélection simple

Dans le cas des matrices on accède aux éléments par les indices [ligne, colonne]. Un "" signifie tous les éléments d'une ligne ou d'une colonne.

```
> data_01<-read.table("DataTD2Ex1.txt",header=TRUE)
> data_01[,1]
 [1] Placebo Placebo Placebo Placebo Placebo Placebo Placebo
 [8] Placebo Placebo Placebo Placebo Placebo Placebo Placebo
[15] Placebo Traite  Traite  Traite  Traite  Traite  Traite
[22] Traite  Traite  Traite  Traite  Traite  Traite  Traite
[29] Traite  Traite
Levels: Placebo Traite
> data_01[1,]
  TRAITEMENT. SCORE
1     Placebo    12
> data_01[1,1]
[1] Placebo
Levels: Placebo Traite
```

Sélection conditionnelle

Pour extraire une colonne il suffit d'appeler le nom de la variable $ le nom de la colonne. Par exemple avec les données précédentes: data_01$TRAITEMENT envoie la colonne TRAITEMENT.

Pour extraire les données sous critère dans une liste de données on peut appliquer un test à tous les éléments d'une colonne.

```
> data_01$TRAITEMENT
 [1] Placebo Placebo Placebo Placebo Placebo Placebo Placebo
 [8] Placebo Placebo Placebo Placebo Placebo Placebo Placebo
[15] Placebo Traite  Traite  Traite  Traite  Traite  Traite
[22] Traite  Traite  Traite  Traite  Traite  Traite  Traite
[29] Traite  Traite
Levels: Placebo Traite
> data_01$TRAITEMENT=="Placebo"
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[10]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
[19] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[28] FALSE FALSE FALSE
```

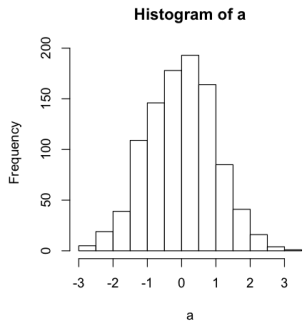
```
> traitement<-data_01[data_01$TRAITEMENT=="Placebo",]
> traitement
   TRAITEMENT. SCORE
1      Placebo    12
2      Placebo    10
3      Placebo    10
4      Placebo    13
5      Placebo    11
6      Placebo    11
7      Placebo    13
8      Placebo     8
9      Placebo     8
10     Placebo    12
11     Placebo     9
12     Placebo     8
13     Placebo     9
14     Placebo     9
15     Placebo    11
```

Comment visualiser graphiquement les données ?

Création d'un histogramme

Utilisation de la commande hist() et tracé d'une distribution gaussienne de 1000 éléments, de moyenne 0 et d'écart-type 1, exemple:

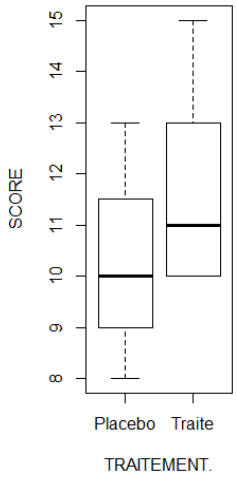
```
> a<-rnorm(1000,0,1)
> hist(a)
```



Création d'un whisker-box

Utilisation des données du TD 2 par exemple:

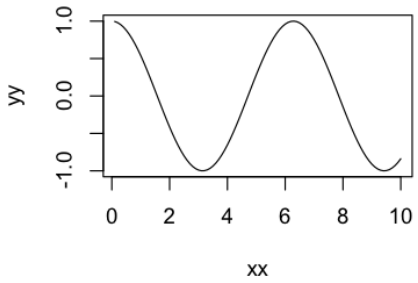
```
> data_01<-read.table("DataTD2Ex1et4.txt",header=TRUE)
> plot(data_01)
```



Remarque: dans ce cas R est capable d'interpréter le format de la série et de choisir un whisker-box de lui-même. Mais la commande plot() est complexe et peut servir à plusieurs types de tracés suivant le format et le type des données d'entrée et aussi des nombreux paramètres de la fonction. Avec la même instruction on peut par exemple tracer une série de valeurs en mode ligne (type="l").

```
> xx<-0.1*(1:100)
> yy<-cos(xx)
> plot(xx,yy,type="l")
```

Le mieux est de se reporter à ?plot pour une explication plus générale.



Tests paramétriques

Test de Student (ou Welch)

La première possibilité consiste à extraire les 2 échantillons puis à réaliser le test avec la syntaxe : t.test(ech1,ech2,var.equal=TRUE). Un exemple est illustré ici à l'aide des données du TD2 du fichier DataTD2Ex1et4.txt.

```
> data_01<-read.table("DataTD2Ex1et4.txt",header=TRUE)
> placebo<-data_01[data_01$TRAITEMENT=="Placebo",2]
> traite<-data_01[data_01$TRAITEMENT=="Traité",2]
> result<-t.test(placebo,traite,var.equal=TRUE)
> result

    Two Sample t-test

data:  placebo and traite
t = -2.1843, df = 28, p-value = 0.03747
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.7128894 -0.0871106
sample estimates:
mean of x mean of y
 10.26667  11.66667
```

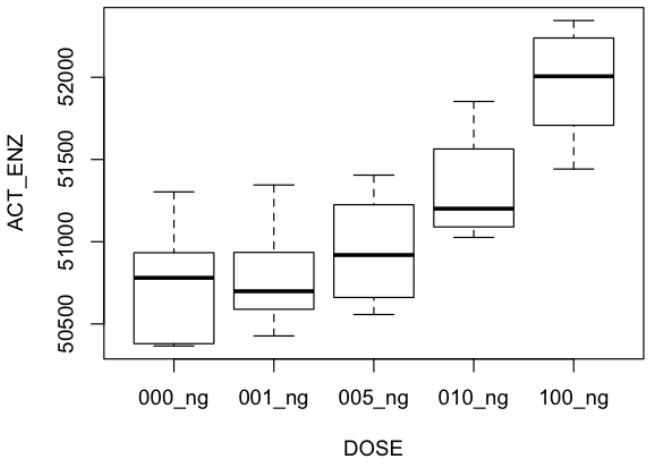
La second possibilité consiste à utiliser l'opérateur "~" (A~B signifie grosso modo formule A est une fonction ou un mapping de B mais sans que l'écriture fasse une évaluation, simplement cette commande crée un objet de la classe formule)

```
> result<-t.test(data_01$SCORE~data_01$TRAITEMENT,var.equal=TRUE)
> result

    Two Sample t-test

data:  data_01$SCORE by data_01$TRAITEMENT
t = -2.1843, df = 28, p-value = 0.03747
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.7128894 -0.0871106
sample estimates:
mean in group Placebo  mean in group Traité
             10.26667              11.66667
```

ANOVA à 1 voie

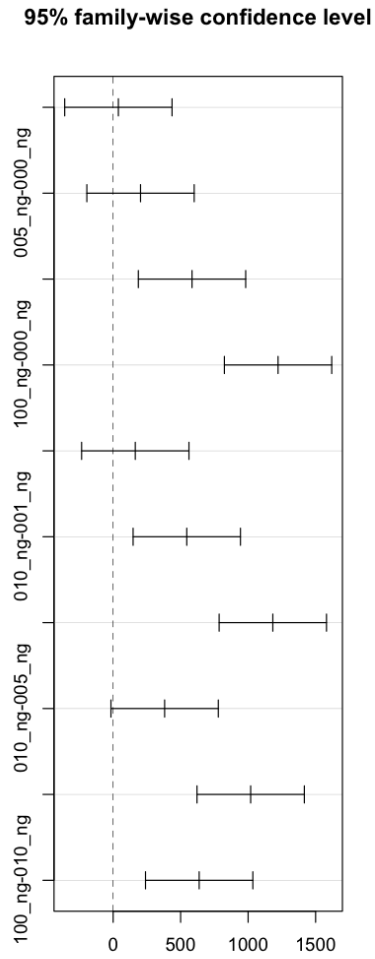
La commande aov() de R permet de réaliser une analyse de variance, ici à 1 voie et en utilisant les données du TD2 (DataTD2Ex3.txt).

```
> data_02<-read.table("DataTD2Ex3.txt",header=TRUE)
> result<-aov(ACT_ENZ~DOSE,data_02)
> summary(result)
             Df   Sum Sq Mean Sq F value   Pr(>F)
DOSE          4 10382136 2595534   26.51 2.47e-11 ***
Residuals    45  4406199   97916
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> plot(data_02)
```



L'anova retourne ici un résultat positif (Pr 2.47e-11) et on peut donc réaliser un post-hoc (ici TukeyHSD) pour mettre en évidence les différences:

```
> thsd<-TukeyHSD(result)
> thsd
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = ACT_ENZ ~ DOSE, data = data_02)

$DOSE
                  diff        lwr       upr     p adj
001_ng-000_ng    38.8 -358.83154  436.4315 0.9986550
005_ng-000_ng   202.9 -194.73154  600.5315 0.5994613
010_ng-000_ng   584.4  186.76846  982.0315 0.0012131
100_ng-000_ng  1221.8  824.16846 1619.4315 0.0000000
005_ng-001_ng   164.1 -233.53154  561.7315 0.7666844
010_ng-001_ng   545.6  147.96846  943.2315 0.0028142
100_ng-001_ng  1183.0  785.36846 1580.6315 0.0000000
010_ng-005_ng   381.5  -16.13154  779.1315 0.0656438
100_ng-005_ng  1018.9  621.26846 1416.5315 0.0000000
100_ng-010_ng   637.4  239.76846 1035.0315 0.0003690
```



Il est également possible de représenter graphiquement les résultats du post-hoc:

```
> plot(thsd)
```

Dans ce cas et en raison de l'échelle, toutes les comparaisons n'apparaissent pas sur l'axe des ordonnées.

Sources

http ://www.duclert.org/ excellent guide et assez exhaustif