TP1 : Prise en main du logiciel R

1. R et RStudio

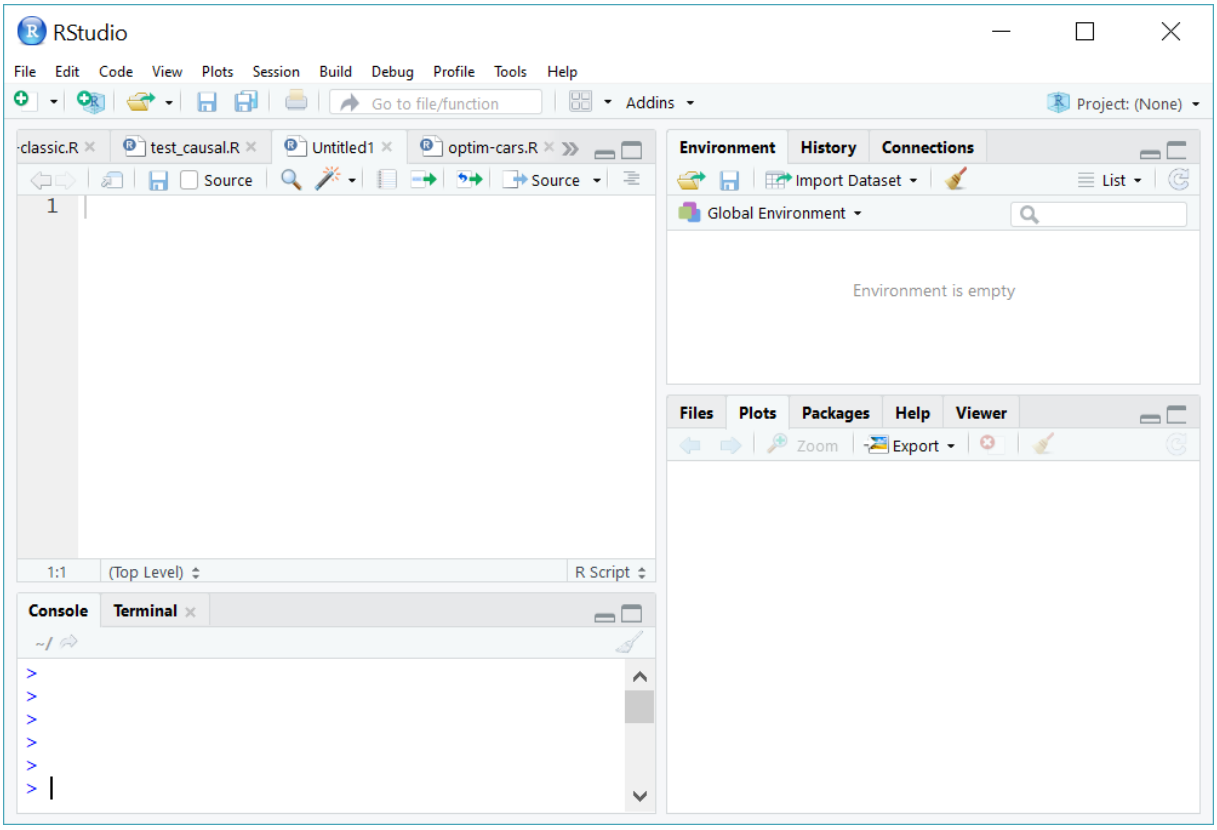
L'objectif de ce TP est de vous familiariser avec R, un environnement libre, conçu au départ pour le calcul statistique, et étendu à de nombreux domaines grâce à la disponibilité de packages, c'est-à-dire des extensions téléchargeables sur Internet. Vous allez apprendre aujourd'hui à manipuler des données simples et faire quelques calculs. La première étape consiste à installer R :

https://www.r-project.org/

Ensuite, on va utiliser une interface graphique gratuite délivrée par RStudio Desktop :

https://www.rstudio.com/products/rstudio/#Desktop

Attention à ne pas changer l'ordre d'installation. On installe d'abord R puis RStudio Desktop.



RStudio est séparé en 4 fenêtres graphiques :
— la console (en bas à gauche) : permet d'exécuter les instructions avec la touche Entrée
— le script (en haut à gauche) : permet d'écrire l'ensemble des commandes (ou instructions) que l'on veut exécuter et de pouvoir les sauvegarder dans un fichier. Une instruction s'exécute dans la fenêtre en bas à gauche en appuyant sur Ctrl+Entrée. Pour

ne pas perdre de temps, vous pouvez copier-coller les commandes qui sont données dans les énoncés de TP dans votre script.
— la fenêtre Files/Plots/Packages/Help (en bas à droite) : permet entre autres de visualiser les graphiques ou l'aide.
— la fenêtre Workspace/History (en haut à droite) : permet de voir l'ensemble des objets en mémoire et historique de toutes les instructions réalisées.

2. Utiliser R comme calculatrice

Dans la suite de ce TP, les codes informatiques à soumettre directement dans la console de Rstudio seront précédés du symbole « > ». Bien entendu ce chevron ne fait pas partie du code à taper...

L'addition des deux entiers 2 et 4 donne un seul résultat qui vaut 6. Vérifiez que vous obtenez bien :

```
> 2+4
[1] 6
```

Quels affichages provoquent les instructions suivantes :

```
> 36/5
> sqrt(4)
```

Le résultat d'un calcul doit être affecté à une variable, pour être conservé. Si l'on demande :

```
> var <- 3 * sqrt(4) + pi
```

R n'affiche alors plus le résultat... Pour accéder au contenu de la variable var, on pourra utiliser la fonction d'affichage à l'écran print() :

```
> print(var)
```

Que vaut var ?
Certaines variables telles que pi sont prédéfinies. Il en existe de plus surprenantes ; par exemple que vaut mystere après l'exécution de l'instruction suivante :

```
> mystere <- var / 0
```

3. Quelques commandes utiles

q() quitter R
help(topic) afficher l'aide relative à topic ou simplement utiliser le menu Help de Rstudio
source("fichier.r") charger un programme stocké dans un fichier nommé ici fichier.r
getwd() et *setwd("chemin")* afficher et modifier le répertoire de travail

4. LES OPERATEURS

Comme toute calculatrice, R permet l'usage direct de nombreux opérateurs de calcul :
+ et - addition et soustraction
* et / multiplication et division

%/% et %% division entière et modulo

Après l'examen par R des instructions ci-dessous, que valent les variables resultat et reste ?

```
> tmp <- 7 / 2
> resultat <- floor(tmp)
> reste <- 7 - 2*resultat
```

Essayez maintenant en écrivant :

```
> resultat <- 7 %/% 2
> reste <- 7 %% 2
```

D'autres opérations ne disposent pas d'un symbole réservé, mais sont accessibles grâce à des fonctions, par exemple le calcul de la valeur absolue, de la racine carrée ou de la partie entière :

```
> abs(-3)
[1] 3
> sqrt(49)
[1] 7
> floor(-3.7)
[1] -4
```

Que vaut la racine carrée de la partie entière de trois fois pi ? Posez le calcul en une seule ligne.

Les opérateurs logiques

Comparaisons
== teste l'égalité entre deux valeurs
!= teste la différence entre deux valeurs
<, <=, >, >= teste les relations d'ordre entre deux valeurs

Dans le code suivant, x est inférieur à y, l'affirmation x > y est donc fausse :

```
> x <- 10
> y <- 20
> x > y
[1] FALSE
```

Le résultat d'une opération logique peut être stocké dans une variable de type logique (dite parfois booléenne ou binaire : elle ne peut prendre que deux valeurs, TRUE ou FALSE)

```
> x <- 3.14
> y <- pi
> logiq <- x == y
```

Que contient la variable logiq ?

! négation
&& ET logique
|| OU logique inclusif
xor(,) OU logique exclusif (fonction à deux paramètres)

Si logiq est fausse, son contraire est vrai :

```
> !logiq == TRUE
[1] TRUE
```

Examinez les instructions suivantes :

```
> x <- T
> y <- F
> v1 <- x && y
> v2 <- x || y
> v3 <- v2 && !v1
```

Que vaut v3 ?
Il est souvent primordial d'utiliser des parenthèses pour fixer la priorité des opérations. En effet, après :

```
> x <- T
> y <- F
> z <- F
> v1 <- y && z || x
> v2 <- y && (z || x)
```

v1 et v2 valent-elles la même chose ?

5. Généralités sur les variables

5.1 Types de variables

type en français	Type R	Valeurs possibles
Logique	logical	TRUE, FALSE noté aussi T et F
Numérique	numeric	1, 1.14, pi
Chaîne de caractères	character	'un_mot', "L"
Non précisé	na	NA
Objet vide	null	NULL

Il est toujours possible de vérifier le type d'une variable quelconque, grâce à la fonction class() :

```
> txt <- "un texte"
> class(txt)
[1] "character"
```

On peut aussi faire des tests plus spécifiques :

```
> is.character(txt)
[1] TRUE
> is.numeric(txt)
[1] FALSE
```

Il est éventuellement possible de modifier le type d'une variable existante, grâce aux fonctions as.logical() as.numeric() ou as.character(), par exemple. Ainsi, que contiennent txt et nbr et qu'affiche R après l'exécution du code ci-dessous :

```
> txt <- '42'
> nbr <- as.integer(txt)
> is.numeric(nbr)
```

5.2 Manipulation de variables

5.2.1 Concaténation de texte

Afin de créer un texte à partir de plusieurs variables (de type character ou pouvant s'y ramener) on utilisera la fonction paste() :

```
> mot <- "petite"
> text1 <- paste("une", mot, "phrase")
> text2 <- paste(text1, "compte", nchar(text1), "lettres")
```

Que contiennent alors text1 et text2 ?

5.2.2 Quelques particularités

Les valeurs NA (non affecté) et INF (infini) existent explicitement et peuvent être manipulées :

```
> tmp <- 3 / 0
> nsp <- NA
> resultat <- paste(tmp, tmp+1, tmp+nsp)
```

Que contient résultat ?
Il est d'ailleurs possible de tester les variable à l'aide des fonctions : is.null(), is.nan(), is.na()...

```
> tmp <- NA
> tmp == NA
[1] NA
> is.na(tmp)
[1] TRUE
```

6. Les vecteurs de variables

6.1 Création d'un vecteur

Il existe de nombreux moyens de créer des vecteurs de variables : concaténation explicite de valeurs avec la fonction c(), génération automatique de séquence avec seq() ou l'opérateur « : », répétition d'une séquence avec rep()...
Créez et affichez chacun des 4 vecteurs ci-dessous :

```
> vecteur1 <- c(1, 3, 5, 7, 9)
> vecteur2 <- seq(from=0, to=10, by=2)
> vecteur3 <- 0:10
> vecteur4 <- rep(1:2, 5)
```

6.2 Manipulation d'un vecteur

On accède au ième élément du vecteur vect en écrivant : vect[i]. Vérifiez que R commence à numéroter à 1 les éléments du vecteur en demandant d'afficher le contenu de vecteur1, de vecteur1[1], et de vecteur1[0].
De nombreuses fonctions prédéfinies peuvent être appliquées aux vecteurs, parmi lesquelles :
– print() permet d'afficher le contenu du vecteur ;
– length() permet de récupérer le nombre d'éléments contenus ;
– summary() permet d'obtenir une description statistique sommaire du contenu.
Pour afficher une à une les valeurs du vecteur on pourra écrire :

```
> vecteur <- rnorm(10)
> for (i in 1:length(vecteur))
> print(vecteur[i])
```

6.3 Pour aller plus loin

Vous venez de générer des vecteurs suivant une loi statistique (rnorm pour « normale »). Vous pourriez aussi utiliser une loi uniforme runif(), de poisson rpois()... Ces fonctions prennent un nombre de paramètres variable, pour plus d'informations vous consulterez l'aide en ligne : help(rnorm) par exemple.
Vous pouvez générer des matrices à partir de vecteurs, en les concaténant par lignes ou par colonnes, ou bien en tronçonnant un vecteur en plusieurs morceaux...

```
> v1 <- runif(10)
> v2 <- rpois(10, 1)
> v3 <- rnorm(10)
> matrice <- rbind(v1, v2, v3)
```

L'exemple ci-dessus produit une matrice de 3 lignes et de 10 colonnes. Les opérations matricielles usuelles sont définies, vous pouvez ainsi transposer matrice et vérifier ses dimensions :

```
> matricet <- t(matrice)
> dim(matricet)
```

Vous accéderez alors aux données de la matrice par un double indiçage : nom_matrice[num_ligne, num_colonne].

7. Les data.frames

Les data.frames sont formés de plusieurs vecteurs colonnes nommés, de même longueur, mais pas nécessairement de même type. Il s'agit du format d'échange privilégié avec des fichiers de données. Vous pouvez créer un data.frames à partir d'une matrice par exemple, puis vérifier son type :

```
> converti <- as.data.frame(matricet)
> is.data.frame(converti)
```

Vous accéderez alors aux données par une des deux méthodes suivantes :
– par un double indiçage : nom_dataframe[num_ligne, num_colonne] ;

– en nommant la colonne (ce qui est impossible avec une matrice) : nom_dataframe$nom_colonne[num_ligne].

Essayez maintenant l'exemple suivant :

```
> v1 <- c(175, 182, 165, 187, 158)
> v2 <- c(19, 18, 21, 22, 20)
> tableau <- data.frame(taille=v1,age=v2)
> names(tableau)
> print(tableau$taille)
> summary(tableau)
> write.table(tableau, "sortie.csv", sep=";")
```

8. Les bibliothèques de fonctions

Les fonctionnalités de R peuvent être enrichies en chargeant des packages de library (c'est-à-dire des groupes de bibliothèques de fonctions) auprès de « sites miroirs » faisant partie du réseau CRAN, acronyme de Comprehensive R Archive Network 2.

Installez la base de données cartographique nommée maps et dessinez un fond de carte de la France, à l'aide du code
suivant (choisissez de préférence un site de téléchargement français) :

```
> install.packages("maps")
> library(maps)
> map("france")
```

Attention, alors que les packages sont installés une fois pour toute, une library doit être rechargée à chaque redémarrage de R. En effet les packages que vous installez sur la machine resteront, sauf indication contraire de votre part, sur le disque dans un répertoire nommé R. En revanche, pour ne pas alourdir inutilement le fonctionnement du logiciel R, les library ne sont chargées qu'à la demande.

9. Les graphiques

Vous pouvez produire avec R l'ensemble des graphiques descriptifs habituels, tels que diagrammes en batons, histogrammes... Essayez les exemples ci-dessous :

```
> datas <- rnorm(20)
> barplot(datas)
> hist(datas, nclass=4)
```

Pour créer vous même vos représentations graphiques, d'innombrable fonctions sont à votre disposition. La plus rustique est plot().

```
> plot(seq(0,2*pi,by=0.01), sin(seq(0,2*pi,by=0.01)),type="l")
```

10. Exercices

Exercice 1.

Charger le fichier iris avec la commande data(iris).

1. Quelle est la nature de iris ? (matrix, vector, data.frame....) data.frame

2. Les données. Combien d'individus ? de variables ? Pour chacune des variables donner leur nature (quantitative, qualitative, ...). 5 colonnes, de 150 individus

3. Calculer pour chaque variable lorsque cela a un sens, la moyenne et la variance empirique. Même question en divisant la population selon le niveau de la variable Species. cf screen,

4. On s'intéresse au 4 premières variables. Standardiser les données (retirer la moyenne et diviser par l'écart-type).

5. Tracer le graphique variable Petal.Length versus Petal.Width en utilisant des couleurs différentes selon le niveau de la variable Species. Essayer tous les "croisements" possibles et commentez ces graphiques. utiliser les variables standartdiser

Exercice 2. Etudiez par simulation et illustrez graphiquement la validité du théorème limite central (TLC) dans le cas d'un échantillon binomial et d'un échantillon exponentiel. Faites la simulation pour les échantillons de petite et de grande taille. Essayez la simulation "dynamique" suivante :

```
ms=numeric(10000);

p=0.75; x=seq(-4,4,0.025);

for (j in(1:50)){

k=j*j; for (i in (1:10000)){

sig=sqrt(p*(1-p)/k); mu=p; ms[i]=(mean(rbinom(k,1,p))-mu)/sig }

hist(ms, breaks=41, xlab="x-variable", xlim=c(-4,4), prob=TRUE, main=sprintf("normal curve over histogram, n = %d",k))

curve(dnorm(x), col="darkblue", lwd=2, add=TRUE, yaxt="n")}
```

Commentez le code R et analysez les résultats. Refaites la même expérience avec des échantillons de loi de Cauchy et analysez les résultats.