TP1 STA201
Prise en main du logiciel statistique R

christine.keribin@universite-paris-saclay.fr, Olivier Coudray, Henri Mermoz Kouye, Zacharie Naulet, Camille Palmier

10 septembre 2020

La statistique développe des problématiques théoriques, méthodologiques et appliquées, et le logiciel est un élément constitutif de la discipline. Le logiciel R joue un rôle clé dans la recherche scientifique statistique comme outil de partage des recherches, mais aussi dans le développement d'applications industrielles (avec le module Rshiny pour le développement web par exemple). D'où le choix de son utilisation dans le cours STA201.

L'objectif n'est pas de faire des étudiants des professionniels de ce langage, mais de l'utiliser pour les calculs permettant ensuite d'interpréter des résultats statistiques. Le logiciel R sera utilisé dans tous les autres modules de statistique du M1 Mathématiques Appliquées.

Objectifs de la séance

- (Re)-découvrir le logiciel R et l'interface Rstudio
- Mettre en place un environnement de travail efficace
- Repérer les commandes nécessaires pour créer des vecteurs, dataframe, calculer des statistiques usuelles, utiliser des fonctions, tracer des graphiques élémentaires
- Optionnel: créer des fonctions, créer des rapports incluant commandes R et texte Latex.
- La **partie 8 est à rendre par binôme** sous forme d'un compte-rendu rédigé (pdf) et d'un fichier de commande (R) pour **le vendredi 18 septembre 18h** via l'outil devoir de ecampus. Ne pas oublier de définir un titre, une introduction, une conclusion et de justifier vos résultats. Nommez-vos fichiers avec le NOM des membres du binôme.

Le logiciel R

Le logiciel R (disponible sur le site CRAN: `http://www.r-project.org/`) est un logiciel libre de Statistique avec de nombreux avantages[1] :

- il dispose de fonctions prédéfinies pour utiliser des méthodes statistiques classiques,
- il permet d'écrire ses propres fonctions,
- il permet d'utiliser des techniques statistiques innovantes et récentes à l'aide de "packages" développés par les chercheurs et mis à disposition sur le site du CRAN

Le logiciel R fonctionne en ligne de commande, mais des interfaces permettent une utilisation plus conviviale, comme `RStudio` (`https://www.rstudio.com/products/rstudio/`).

Il est aussi possible de faire un notebook sans installation directement sur le web `https://rnotebook.io/`.

Dans l'environnement ENSTA, `RStudio` se lance à partir d'une fenêtre de commande de la façon suivante

```
useensta rstudio
usediam r
rstudio &
```

L'interface `RStudio` est composée de quatre fenêtres:

[1] De nombreux tutoriels existent sur le web ; par exemple *R pour les débutants*, par E. Paradis. voir aussi http://beginr.u-bordeaux.fr/

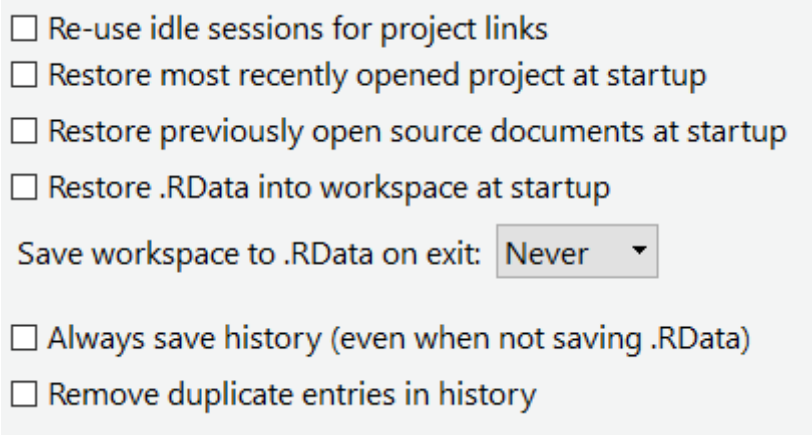

Figure 1: panel de configuration des options

1. Fenêtre de **commande** (à gauche) : contient une console dans laquelle les codes R sont saisis pour être exécutés.
2. Fenêtre **espace de travail** (en haut à droite) : contient les objets en mémoire, que l'on peut consulter en cliquant sur leur nom, ainsi que l'historique des commandes exécutées.
3. Fenêtre **explorateur** (en bas à droite) : permet de se déplacer dans les répertoires, contient les graphiques tracés, montre les packages installés et actuellement chargés, permet d'accéder à la documentation sur les fonctions et les packages.
4. Fenêtre d'**édition** (en haut à gauche) : cette fenêtre n'apparaît que s'il existe des fichiers contenant les scripts R ouverts.

R est organisé en `packages`: les packages de base sont chargés en mémoire au moment du lancement, certains seront à charger avant utilisation par la commande `library()`. D'autres peuvent ne pas être encore installés; leur installation pourra se faire grâce à l'onglet `Package/Install` si l'ordinateur possède une connexion internet.

La **documentation** et l'aide à l'utilisation d'une fonction ou d'un package sont accessibles en cliquant sur l'onglet help ou par ligne de commandes:

```
help("mean")
?mean
```

Commenter le résultat des commandes suivantes:

```
??mean
apropos("mean")
```

Il est conseillé de paramétrer le panel des options (menu `tools/options`) comme indiqué sur la figure 1, pour éviter la sauvegarde et le rechargement de la session à la fermeture/ouverture de session.

1 Calculette

R peut être utilisé pour réaliser des opérations élémentaires :

```
((1+sqrt(5))/2)^2
```

dont le résultat peut être stocké dans une variable

```r
a=((1+sqrt(5))/2)^2
```

gardée en mémoire (`a` apparaît alors dans l'espace de travail) pour être ré-utilisée ensuite :

```r
nombredor = sqrt(a)
```

La fonction `rm` supprime des objets en mémoire dans la session de travail :

```r
rm(a)  # suppression de l'objet a
# noter l'utilisation du dièze définir un commentaire dans du code R
```

Il est conseiller de nettoyer l'environnement de travail au début de chaque nouvelle étude:

```r
rm(list=objects()) # suppression de tous les objets en mémoire
```

2 Fichier de script

2.1 Scripts .R

Toutes les commandes peuvent être lancées à partir d'un fichier.

1. Créer un fichier de commandes (`File / New File / R Script`)
2. L'enregistrer avec l'extension `.R` permettant une coloration syntaxique adaptée à R
3. Y copier les quelques commandes précédemment exécutées depuis l'onglet `History` de la fenêtre `Environment` (icône `To Source`).
4. Enregistrer le fichier.
5. Pour exécuter le code depuis la fenêtre d'édition, sélectionner les commandes à exécuter, puis cliquer sur l'icône `run`, ou utiliser le raccourci clavier `CTL+enter`. Note: ce raccourci clavier permet également d'exécuter la ligne sur laquelle se trouve le curseur, sans avoir à la sélectionner en entier.
6. Il est possible d'exécuter la totalité fichier (même non ouvert) avec la commande `source()`

2.2 Répertoire de travail

Le **répertoire de travail** est par défaut celui à partir duquel l'interface `RStudio` est lancée. Il est pratique de se placer dans un répertoire de travail bien défini, celui qui contiendra les fichiers `.R` sur lesquels vous travaillez. Pour ce faire, vous pouvez soit utiliser la commande `setwd()`, soit utiliser le menu de l'interface : `Session / Set working Directory / To Source File Location` qui exécute cette commande.

```r
getwd() # indique le répertoire de travail en cours
setwd("MonRepertoire") # définit le répertoire en cours: remplacer "MonRepertoire" par le vôtre!
```

2.3 Scripts .Rmd

Pour les curieux, voir en fin de TP.

3 Vecteurs

Les objets de R se différencient par leur mode, qui décrit leur contenu (`logical`, `numeric`, `character`), et leur classe, qui décrit leur structure (`vector`, `matrix`, `array`, `factor`, `data.frame`, `list`, `function`, ...). Les listes ou les data.frame peuvent avoir un contenu hétérogène par sous-structure.

Les nombres décimaux sont encodés avec un point décimal (`3.14`), les chaînes de caractères par des guillemets simples ou doubles (`'BLEU'`), et les valeurs logiques par `TRUE` ou `FALSE`, ou leurs abréviations `T` et `F`. Les valeurs manquantes sont codées par la chaîne de caractères `NA`.

L'objet `a` défini ci-dessus est un scalaire. Pour définir un vecteur, la fonction de base est la fonction de concaténation `c()`, mais il existe des fonctions pour créer des suites régulières (`seq()` et `rep()` ou l'opérateur `:` par exemple).

1. Construire les vecteurs suivants avec la méthode la plus adaptée
   - $V1 = (-1, 3.2, -2, 8)$
   - $V2 = (-2, -1, 0, 1, 2, 3, 4, 5, 6)$
   - $V3 = (0.05, 0.1, 0.15, 0.2)$
   - $V4 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$
   - $V5$ =(OUI, NON)
2. Ordonner le vecteur $V1$
3. Quel est le résultat de l'opération $V6 = 2 * V2 - 3$, puis de l'opération $V3 + V2$? Qu'en concluez-vous sur les opérations arithmétiques utilisant des vecteurs ?
   Calculer le logarithme des coordonnées de $V3$.
   Quel est le résultat de V5+1?
4. Accéder à la deuxième composante de $V5$. Afficher la longueur de $V6$. Créer le vecteur $V7$ formé de 3 dernières composantes de $V6$
5. Calculer la somme des coefficients de $V6$.
6. Compter le nombre de composantes positives de $V2$
7. Quel est le rôle de la fonction `choose`? Calculer les poids $P(X = 2)$ de la densité de la loi binomiale $\mathcal{B}(10, 0.2)$ pour les valeurs successives de $k = 0, \ldots, 10$. Créer un vecteur `k` contenant les entiers de 0 à 10, et un vecteur `pk` contenant les poids. Calculer la moyenne et la variance de cette binomiale en utilisant les deux vecteurs, puis comparer avec la valeur attendue. Calculer $P(X \leq k)$ (aide: fonction `cumsum`).

4 Structure de données

La meilleure façon de décrire et analyser des données dans R est de les saisir dans des objets appelés `data.frame`. Un data.frame possède des lignes et des colonnes comme une matrice, mais les colonnes peuvent être de différente nature : numérique, caractères, logique. Les lignes contiennent différentes observations ou mesures et peuvent être vues comme des "individus". Les colonnes contiennent les valeurs de variables afférentes aux individus. Elles ont des noms, ce qui permet d'y accéder par l'opérateur `$`.

Exécuter, puis interpréter le résultat des commandes suivantes:

```
df= data.frame(k,proba=pk)
print(df)
df$proba
df["proba"]
df[1:4,2]
df$proba[1:4]
max(df$proba)
summary(df)
attach(df) # accès direct aux colonnes, mais attention !!
detach(df)
```

Nous étudions maintenant le jeu de données `cabbages` du package `MASS`, qu'il faut commencer par charger

```
library(MASS)
?cabbages
```

```r
summary(cabbages)
var(cabbages$HeadWt)
cabbages[ cabbages$Cult=="c39" | cabbages$Date=="d20" ,]
tapply(cabbages$HeadWt,cabbages$Date,mean)
?tapply
```

1. Commenter le résumé (`summary()`) de chaque variable. Citer les fonctions R qui permettent d'en calculer chaque valeur.
2. Quel est le calcul fait par la fonction `var()`?
3. Inspirez-vous d'une des commandes ci-dessus pour définir une instruction donnant le nombre de choux cultivés à la date d16 suivant le protocole de culture c52 et de poids supérieur au poids moyen de l'échantillon.
4. Quelle est la fonction de la commande `tapply()` ? Calculer le poids moyen suivant la date et le type de culture.
5. Le plan d'expérience est-il équilibré suivant les deux facteurs qualitatifs (fonction `table()`)?

5 Les fonctions de probabilité

R dispose d'un grand nombre de fonctions prédéfinies, utilisables en appelant la fonction par son nom suivi de ses arguments (il n'est pas nécessaire de mentionner les arguments optionnels utilisés avec les valeurs par défaut).

Les fonctions liées aux lois de probabilités sont dénommées avec un préfixe (parmi `p`, `d`, `r`, `q`) indiquant le type d'opération (probabilité, densité, simulation, quantile), et un suffixe indiquant le type de la loi (`norm`, `t`, `chisq`, `binom`, `exp`, ...)

1. Soit X une variable aléatoire suivant une loi de Poisson de paramètre 3. Que vaut $P(X = 1)$, $P(X \geq 3)$?
2. Soit Y une variable aléatoire suivant une loi du χ^2 à 5 ddl. Déterminer a tel que $P(X \leq a) = 0.95$.
3. Retrouver les résultats de la question 3.7 en utilisant une fonction de probabilité.
4. Le diamètre d'une pièce produite par une machine suit une loi gaussienne d'espérance $\mu = 10$ et de variance $\sigma^2 = 0.01$. On tire 9 pièces d'un lot de cette machine. Quelle est la probabilité que le diamètre moyen de ces 25 pièces soit compris entre 9.97 et 10.125?

6 Graphiques

Le logiciel R propose de nombreuses fonctionnalités graphiques, dont un aperçu peut-être visualisé en jouant la commande `demo(graphics)`.

Les fonctions graphiques sont séparées en fonctions **principales** (`plot`, `boxplot`, `hist`, ...) qui définissent le type de graphique, ses axes, etc, et les fonctions **secondaires** (`points`, `lines`, `text`, `legend`, ...) qui ajoutent des ordres graphiques sur un graphe existant.

```r
attach(cabbages)
par(mfrow=c(1,2),oma=c(0,0,3,0))
barplot(table(Date),main="diagramme en barres",
        col=rainbow(3))

plot(VitC,HeadWt,col=Cult,pch=as.numeric(Cult))
title("poids en fonction du taux de Vitc et de la culture",cex=.5)
legend("topright",levels(Cult),col=1:2,pch=1:2)
title(main="Quelques graphes",   outer=TRUE)
```

1. Interpréter le code précédent, indiquer le rôle de chaque argument.
2. Représenter la variable `HeadWt` sous forme d'un histogramme (`hist()`). Superposer (`curve()`) la densité de la loi gaussienne d'espérance et variance celles calculées sur l'échantillon. Ajouter un titre reprenant ces informations.
3. Afficher la boîte à moustaches (`boxplot()`) de la variable `HeadWt`. Superposer un point (`points()`) représentant la moyenne de l'échantillon. Commenter.
4. Représenter la boîte à moustaches du jeu de données `chem` du package `MASS`. Superposer un point représentant la moyenne de l'échantillon. Commenter.
5. Il est possible de générer des fichiers d'image sous forme de pdf ou de postscript par exemple. Il suffit d'encadrer le code concerné par les commandes de création (`pdf`, `postscript`) et de fermeture (`dev.off()`)

```
pdf("monpdf.pdf")  # pour créer un fichier pdf contenant le graphique
# insérer les commandes graphiques
dev.off()  # pour fermer le fichier pdf
```

7 Import et export

7.1 A partir d'un fichier texte

L'import et l'export de fichiers de données sous forme de fichier texte peuvent se faire en appelant les fonctions `read.table()` (import dans un data.frame d'un fichier texte) et `write.table()` (export dans un data.frame d'un fichier texte). Par exemple, la commande suivante enregistre le data.frame `df` dans un fichier texte, en omettant les guillemets pour les chaînes de caractères et en choisissant le point virgule comme séparateur.

```
write.table(df,"coeff_binom.txt",row.names=FALSE,sep=";",
            quote=FALSE)
```

Si le fichier contient des lignes que l'opération de lecture doit ignorer (par exemple, une explication du contenu du fichier), il faut indiquer de les sauter (`skip=`). L'argument `header=` indique si la ligne juste avant les données contient le nom des variables

1. Ajouter une ligne de commentaire en haut du fichier, puis importer le nouveau fichier texte dans un nouveau data.frame. Vérifier le résultat obtenu.

```
df2=read.table("coeff_binom.txt",sep=";",header=TRUE,skip=1)
```

7.2 A partir d'un fichier binaire

Il est également possible d'exporter `save()` et d'importer `load()` des objets suivant un format binaire propre à R. Dans ce cas, il est commode de définir le fichier enregistré avec une extension `.RData` pour permettre une identification aisée du type de fichier.

Le code suivant enregistre le data.frame sous forme binaire, le supprime de l'environnement R puis le recharge à partir son l'enregistrement.

```
save(df,file="coeff_binom.RData")
rm(df)
load("coeff_binom.RData")
```

En particulier, à l'issue de la session, RStudio propose d'enregistrer tous les objets en mémoire, qui seront alors rechargés au lancement de la session suivante. Il est conseillé de ne pas le faire, d'où le paramétrage indiqué sur la figure 1.

8 Une analyse statistique

Le fichier `crabs.data` enregistre en mm la longueur de la carapace (`CL`) et la profondeur du corps (`BD`) de 200 crabes[2].

1. Lire les données, calculer les statistiques pertinentes. Les représenter graphiquement sous forme de boîte à moustache (`boxplot`) et y superposer la valeur moyenne (`points`). Commenter.
2. Représenter l'histogramme (`hist`), superposer (`curve` avec l'argument `add=TRUE`) la densité gaussienne de paramètres l'espérance et la variance empirique. Faire de la bibliographie sur le test de Shapiro-Wilks. Expliquer sa construction. Peut-on considérer les données comme étant gaussiennes (`shapiro.test`) ?
3. Construire un intervalle de confiance de niveau 90%, puis 95%, puis comparer vos résultats avec ceux de la fonction `t.test`.
4. Un biologiste affirme que cette espèce est de longueur moyenne de carapace inférieure à $31mm$. Ce biologiste sous-estime-t-il la taille de cette espèce? préciser le risque.
5. Ce biologiste se demande si la profondeur moyenne du corps de cette espèce fait $14mm$. Que pouvez-vous lui répondre?

9 Programmer une fonction

Un des intérêts de R est de pouvoir créer ses propres fonctions, suivant le schéma suivant:

```
fun1 = function(arg1, arg2=1){
  w = arg1/arg2^2
  return(arg2 + w)
}
fun1(3)
```

Soit f la fonction définie par $f(x) = k/(1 + (k/n_0 - 1)e^{-rx})$, où k, n_0, r sont des paramètres.

1. Écrire le code de cette fonction, en définissant les paramètres k, n_0, et r par défaut à 2, 1 et 1 respectivement.
2. Soit `x=0:10`. Que retourne cette fonction si on l'appelle avec les paramètres par défaut?
3. L'ordre des arguments a-t-il de l'importance?
4. L'argument de sortie d'une fonction étant unique, il est fréquent d'utiliser une liste (`list()`) pour passer différentes informations qui peuvent être de mode et de classe différents. Modifier la fonction précédente, de façon à ce qu'elle retourne, en plus de la valeur $f(x)$, une chaîne de caractère indiquant `grand` si $f(x) > 1$ et `petit` sinon. Tester!
5. Tracer les points $(x, f(x))$ pour $x = -10 : 10$, où f est la fonction sigmoïde de la section précédente.
6. Superposer au graphe précedent la fonction sigmoïde sous forme de courbe (`lines`)

10 Tableaux et matrices

Les matrices ont deux dimensions, sont de mode quelconque mais homogène (ne contiennent que des éléments de même nature). La création d'une matrice se fait avec la syntaxe :

```
n=4 # nombre de lignes
p=3 # nombre de colonnes
 matrix(vec,nrow=n,ncol=p)
```

[2]Campbell, N.A. and Mahon, R.J. (1974): A multivariate study of variation in two species of rock crab of genus Leptograpsus. Australian Journal of Zoology 22, 417-425

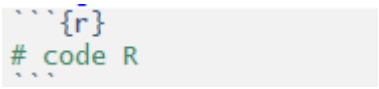

Figure 2: les marquages Rmarkdown

où `vec` est le vecteur contenant les éléments de la matrice, qui seront rangés en colonne (sauf si l'argument `byrow=TRUE` est choisi). On peut également créer des matrices en concaténant des vecteurs de même dimension en ligne (`rbind()`) ou en colonne (`cbind()`).

1. Créer les matrices A et B par concaténation de plusieurs vecteurs (**cbind(), rbind()**):

$$A = \begin{pmatrix} 4 & -1 & 1 \\ 3 & 10 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 4 \\ 3 & 3 \\ 4 & 2 \end{pmatrix}$$

2. Sélection de la deuxième colonne de A, et le coefficient $3,2$ de B?
3. Créer la matrice C en par suppression de la deuxième colonne de A et la matrice D en sélectionnant les deux premières lignes de B ?
4. Quel est le résultat des instructions `A*B`, `C*D` ? de l'instruction `M=A%*%B`?
5. Extraire les coefficients diagonaux de M, puis inverser M.
6. Calculer les moyennes des colonnes de A et la somme de chaque ligne de B (`apply()`)

11 R-Markdown

Les fichiers `.R` sont utilisés pour faire de la programmation mais sont peu pratiques pour mettre des commentaires ou enregistrer les résutlats des commandes. Afin de faciliter la prise de notes en TP, R-Markdown[3] offre une syntaxe simplifiée pour mettre en forme des documents contenant à la fois du texte, des instructions R et le résultat fourni par R lors de l'évaluation de ces instructions. Les fichiers R-Markdown sont suffixés par `.Rmd` tandis que les fichiers de commande R sont suffixés par `.R`. Le fichier `TP1-Intro-2018.Rmd` sur le site du cours est un fichier R-Markdown qui contient les commandes produisant l'énoncé du TP.

Dans un fichier R-Markdown, les commandes `R` sont incluses dans des paragraphes appelés *chunks* et encadrés par des marques spécifiques (figure 2) qu'il est possible de générer avec le bouton `Insert`.

Des options permettent de paramétrer la mise en forme du document

- `eval`: exécute le code dans les *chunks* (TRUE ou FALSE)
- `echo`: affiche le code dans le document (TRUE ou FALSE)
- `results`: 'markup', 'asis', 'hide', 'hold'

La génération du document peut se faire en cliquant sur le bouton `knit`, soit sous forme pdf, soit sous forme html. Il est possible d'extraire le code `R` d'un document R-Markdown en utilisant la fonction `purl()` du package `knitr`.

```
library(knitr)
purl("STA201-TP1-Intro-2018.Rmd")
```

Les options peuvent être définies de façon générale en début de document. Par exemple

```
knitr::opts_chunk$set(echo = TRUE,eval=FALSE)
```

[3] voir par exemple https://www.fun-mooc.fr/c4x/UPSUD/42001S02/asset/RMarkdown.pdf pour une introduction, https://rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf pour un résumé des commandes

empêche l'exécution du code R (`eval=FALSE`) lors du "tricottage", mais inclut les commandes dans le document généré (`echo = TRUE`). Elles peuvent aussi être redéfinies localement à chaque *chunk*.