Probabilités Appliquées - Langage R, Rstudio (et ChatGPT)

CM - séance 1

Olivier Francois

Introduction et objectifs

- Brève présentation du langage de programmation R
- Ouvrage en français (Vincent Goulet)
- Documentation en ligne, de ressources sur la toile
- Open AI ChatGPT/Codex ou Copilot

Langage R

- Langage de programmation interprété pour des applications en analyse de données
- Basé sur la notion de **vecteur** simplifiant le recours aux structures itératives
- Pas de typage ou de déclaration obligatoire des variables
- Programmes courts comportant quelques lignes de code

Utilisation de R en mode interactif

- Utilisation en ligne de commande
- Suite des commande enregistrée dans le fichier *.Rhistory*
- Peut être retrouvée par la commande `history()`

```
# x contient 100 variables aléatoires uniformément réparties entre 0 et 1
> x <- runif(100)
# la fonction mean() calcule la moyenne du vecteur
> mean(x)
   [1] 0.4897892
```

Utilisation de scripts et de packages

- Script : programme court ayant l'extension **.R** ou **.r** exécutés par la commande `source()`.
- Package R : un programme écrit en R (pouvant utiliser d'autres langages)
- Sauvegarde de l'espace de travail dans le fichier **.RData** (permanence des objets).

Exemple de données manipulées dans R

- Loi de probabilité discrète = suite de nombres positifs de somme égale à un et associés à un ensemble de valeurs numérotées
- Code R pour simuler une loi de probabilité discrète

```
# Définition des probabilités
> probabilites <- c(0.2, 0.3, 0.1, 0.4)
# Générer un échantillon aléatoire en utilisant les probabilités
> echantillon <- sample(1:length(probabilites), size = 100, replace = TRUE, prob = probabilites)
```

Afficher les valeurs de l’échantillon

il n’est pas nécessaire d’écrire print()

> print(echantillon[1:10])

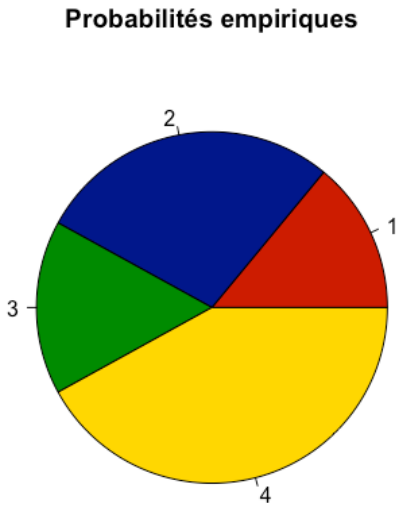
```
 [1] 4 2 2 3 2 4 1 3 2 3
```

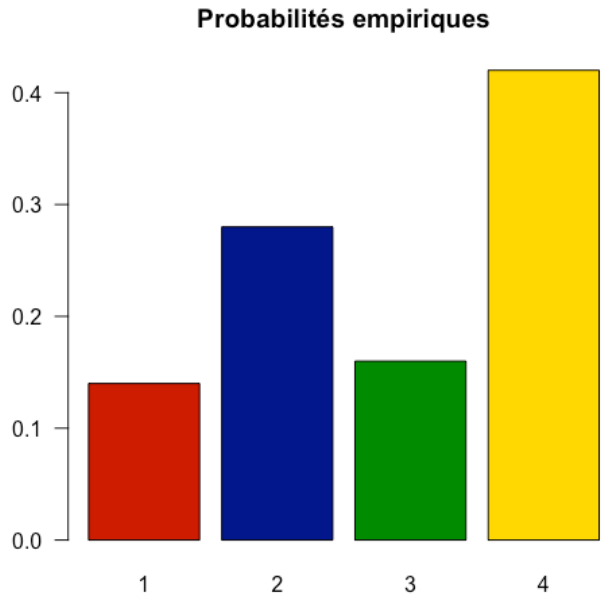
Afficher une table des valeurs de l'échantillon

table() donne d'effectif de chacune des valeurs simulées

```
> table(echantillon)
   echantillon
    1   2   3   4
   14  28  16  42
```

Camembert (pie) et diagramme en barre (barplot)





Principales classes d'objets

- `numeric` : un vecteur contenant des valeurs numériques
- `character` : un vecteur contenant des caractères ou des chaînes de caractères
- `logical` : un vecteur contenant des valeurs logiques ou binaires
- `matrix` : un vecteur contenant des valeurs numériques, recodé comme une matrice comportant n lignes et p colonnes

Principales classes d'objets

- `list` : une liste d'objets potentiellement hétérogène, contenant par exemple des vecteurs, des matrices, des caractères
- `data.frame` : une liste d'objets de même longueur organisée en tableau
- `function` : une fonction, par exemple, `mean`

Exemple d'objet

- Supposons que la variable `x` prenne la valeur 8
- `x` est un vecteur dont la première valeur est égale à 1

```
[1] 8
```

- le vecteur `x` est de classe `numeric` et de longueur 1

```
> class(x)
[1] "numeric"
> length(x)
[1] 1
```

Listes (fonction `list`)

Les attributs d'une liste sont accessibles par le symbole dollar ($) ou par un double crochet

```
# Creation d'une liste
> mes_betes <- list(animal = c("chat", "chien"), sex = c("M", "F"), age = c(2, 8), vaccine = c(TRUE, TRUE))

# Extraire l'attribut animal
> mes_betes$animal
   [1] "chat"  "chien"
```

Modifier un attribut de la liste

> mes_betes$animal[2] <- “raton laveur”
> mes_betes

```
$animal
[1] "chat"          "raton laveur"

$sex
[1] "M" "F"

$age
[1] 2 8

$vaccine
[1] TRUE TRUE
```

Tableau de données (data.frame)

Organiser les données sous la forme d'un tableau (`data.frame`)

> mes_betes <- data.frame(animal = c("chat", "raton laveur"), sex = c("M", "F"), age = c(2, 8), vaccine = c(TRUE, TRUE))

```
          animal sex age vaccine
1          chat   M   2    TRUE
2 raton laveur   F   8    TRUE
```

Exemple de tableau de données

Dans la suite, nous serons amenés à manipuler des tableaux de données contenant des statistiques sur le sport (football)

```
# install.packages("worldfootballR")
> library(worldfootballR)
> df <- fb_match_results(country = c("FRA"), gender = "M", season_end_year = 2022)
```

Exemple de tableau de données (football)

> head(df[,c(“Competition_Name”, “Home”, “HomeGoals”, “Away”, “AwayGoals”, “Referee”)], n = 3L)

	Competition_Name	Home	HomeGoals	Away	AwayGoals	Referee
1	Ligue 1	Monaco	1	Nantes	1	Antony Gautier
2	Ligue 1	Lyon	1	Brest	1	Mikael Lesage
3	Ligue 1	Troyes	1	Paris S-G	2	Amaury Delerue

Exemple d'analyse de données

Récupérer les scores à domicile pour la saison 2021-2022 (colonne `HomeGoals`)

```
# On peut obtenir la colonne "HomeGoals" de la manière suivante
> but_a_domicile <- df[, "HomeGoals"]
# Méthode alternative
> but_a_domicile <- df$HomeGoals
# Autre méthode : attach/detach
> attach(df) ; but_a_domicile <- HomeGoals ; detach(df)
```

Exemple d'analyse de données

La fonction `table` crée un histogramme des données de manière automatique

```
> table(but_a_domicile)
   but_a_domicile
     0   1   2   3   4   5   6
    84 128  92  41  22  10   5
```

Manipulations vectorielles élémentaires

Extraire les éléments d'un vecteur satisfaisant une condition particulière : valeurs supérieures à deux

```
> x <- c(1, 6, 7, 1, 0, 9, 2, 3) ; x
 [1] 1 6 7 1 0 9 2 3
```

Tester les valeurs plus grandes que 2

```
> (x > 2)
   [1] FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE
```

valeurs supérieures à deux

```
> x[x > 2]
   [1] 6 7 9 3
```

Manipulations vectorielles élémentaires

Remplacer les valeurs paires d'une matrice par des valeurs nulles

```
# Creation de la matrice M
> M <- matrix(1:24, nrow = 4, ncol = 6)

# dimensions de la matrice M
> dim(M)
    [1] 4 6
```

Solution

Le reste de la division (modulo) s'écrit "%%"

```
> M[M %% 2 == 0] <- 0
```

Vérifier le résultat

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    0    0    0    0    0    0
[3,]    3    7   11   15   19   23
[4,]    0    0    0    0    0    0
```

Fonctions de base

Structure d’une fonction

```
> f <- function(arg){
                    # body
                    }
```

Quelques fonctions clés

Lire des données locales ou distantes ou pour en sauvegarder

- `read.table()`,`write.table()`
- `scan()`,`write()`
- `read.csv2()`,`write.csv()`, etc.

La documentation de ces fonctions est directement disponible dans R.

Quelques fonctions clés

Effectuer des calculs statistiques de base

- `mean()`,`median()`,`sd()`
- `table()`,`hist()`,`pie()`, etc

Effectuer des tirages aléatoires

- `sample()` pour un tirage d'urne
- `rpois()` pour un tirage de loi de Poisson
- `rbinom()` pour un tirage de loi binomiale

Quelques fonctions clés

- Fonction `apply()` : Appliquer une fonction aux lignes ou aux colonnes d'une matrice ou d'un tableau

```
# Calculer la somme des valeurs de chaque colonne de la matrice M
> apply(M, MARGIN = 2, FUN = sum)
   [1]  4 12 20 28 36 44
# Calculer cette somme directement à l'aide de la fonction colSums()
> colSums(M)
   [1]  4 12 20 28 36 44
```

Bonnes pratiques

- Permettre à un autre programmeur ou un autre utilisateur de comprendre l'organisation d'un projet
- Faciliter la reproductibilité du travail effectué
- Organiser son espace de travail (avec Rstudio).

Bonnes pratiques

- Créer un répertoire pour le TP_n (n = 1,2,3)
- Le répertoire de travail d'un TP pourra contenir les sous-répertoires

  Data, Codes, Documents, Compte_Rendu

Rstudio

- **Rstudio** est une interface de développement dont nous nous servirons pour les TPs
- **Avantages** :
  - possibilité d'associer un projet à un espace de travail,
  - la possibilité de rédiger les TPs et les comptes rendus de TP de manière interactive en même temps que l'on effectue le TP.
  - utilisation de cahiers de travail, **Notebook**.

Projets sous Rstudio

- Rstudio permet d'associer un "projet" à un espace de travail.
- Le projet peut être créé dans l'onglet **File**, en selectionnant **File > New Project** et en suivant les instructions de l'interface
- **Avantages**:
  - Retrouver l'environnement de travail tel qu'on le laisse à la fermeture de l'application.
  - Changer de projet facilement.

Notebooks (Rmd)

- Les cahiers (**Notebooks**) sont le mode de travail interactif
- Un cahier est un programme écrit en langage **Rmarkdown**.
- **Rmarkdown** est un langage de balise permettant de mélanger du texte et du code, que l'on peut exécuter à la volée en mode interactif.
- Il est aussi possible d'intégrer des images, des liens web, des commandes html ou des commandes latex pour inclure des équations.

Règle pour les notebooks TP

- On peut “compiler” un fichier Rmarkdown pour une sortie en format PDF ou html.
- En TPs, il vous sera demandé de fournir obligatoirement un compte rendu au format html, pour que l’enseignant puisse le visualiser à partir de l’application de dépot (TEIDE). Attention, tout autre format sera ignoré (pas de tar.gz, please).
- Il est inutile de connaître le langage Rmarkdown à fond.