Programmation Statistique avec R

Servane Gey

DUT STID en Alternance

Coordonnées et références

Servane.Gey@parisdescartes.fr - Bureau B4-18

- **Documents** : Moodle, COMMUN
- **Références** (liste non-exhaustive) :
  - *Cours de F.-X. Jollois* : https://fxjollois.github.io/cours-2018-2019/stid-2afa--prog-r/
  - *R for Data Science*, H. Wickham & G. Grolemund, O'Reilly, (https://r4ds.had.co.nz/)
  - *The Grammar of Graphics*, L. Wilkinson, Springer (https://www.springer.com/us/book/9780387245447)
  - *Tidyverse* : https://tidyverse.tidyverse.org/
  - *Visualisation* : https://ggplot2.tidyverse.org/reference/
  - *Applications Web* : https://shiny.rstudio.com/
  - *R Markdown* : https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
  - *RStudio Cheat Sheets* : https://www.rstudio.com/resources/cheatsheets/

Organisation et validation

- **Organisation** :
  - 33h de cours et TP répartis sur 11 séances de 3h
- **Validation** :
  - 1 TP noté en milieu de semestre (50 %)
  - 1 DST sur machine en fin de semestre (50 %)

Plan du cours

1. **Manipulation et analyse de données**
   - Importation et exportation de données (package *readr*)
   - Manipulation de tables de données (package *dplyr*)
   - Statistiques : quelques exemples avec *dplyr*
2. **Visualisation de Données**
   - Représentations graphiques (package *ggplot2*)
   - Cartographie (package *leaflet*)
3. **Fonctions**
   - Ecriture de fonctions avec *dplyr*
   - Fonctions spécifiques à l'analyse de données
   - Gestion des sorties sous forme de listes
4. **Rapports et Présentations**
   - Rappels sur R Markdown
   - Rédaction de rapports et de présentations
5. **Applications Web**
   - Rappels sur R Shiny
   - Tableaux de bord (package *shinydashboard*)
   - Conception d'une application avec interface utilisateur

Installation de R et RStudio en local

- Télécharger la dernière version de R adaptée à votre système d'exploitation (Windows, Mac OS, Linux) via le CRAN de l'université de Lyon : https://pbil.univ-lyon1.fr/CRAN/
- Télécharger la dernière version de RStudio adaptée à votre système d'exploitation (Windows, Mac OS, Linux) via le site rstudio : https://rstudio.com/products/rstudio/download/
- Installer les 2 logiciels en éxécutant les fichiers téléchargés.
- **Attention** RStudio est une interface graphique très évoluée pour une utilisation simplifiée du logiciel R. L'installation de R au préalable est donc indispensable avant toute utilisation de RStudio.
- L'installation des librairies (ou *packages*) peut se faire directement via la console, ou via l'interface de RStudio.

Serveur RStudio

- Serveur dédié à l'utiliation de R, dont la dernière version y est installée, avec toutes les librairies régulièrement mises à jour.
- Connexion sur l'interface RStudio directement via la connexion sur le serveur.
- Serveur RStudio de l'IUT (disponible uniquement en local, i.e. sur les ordinateurs de l'IUT) :
  hhtp ://rstudio.iutparis.local :8787/
- Première connexion : login et mot de passe = login de l'IUT.
- Changer immédiatement votre mot de passe (voir le mail de M. Jollois).

Introduction

- **R** :
  - langage de programmation pour le traitement des données
  - fonctions propres intégrées au langage
  - nombreuses librairies
- **tidyverse** :
  - ensemble de librairies développées par l'équipe de R Studio
  - librairies principalement utilisées dans ce cours : *magrittr*, *tibble*, *dplyr*, *readr*, *readxls*, *ggplot2*
- **Data frames** :
  - format des données dans R
  - *observations* (=individus) en lignes et *variables* en colonnes
- **tibble** :
  - format *tbl_df* basé sur un *data.frame*
  - traitement identique aux *data.frame*
- **Conseils** :
  - préférer les librairies développées par des laboratoires ou des entreprises (mises à jour susceptibles d'être plus régulières)
  - écrire et sauvegarder ses codes dans un script R ou R Studio
  - se référencer à l'aide pour l'utilisation des fonctions

Les données avec *tibble*

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1
```

```
library(tibble)
as_tibble(mtcars)
```

```
## # A tibble: 32 x 11
##      mpg   cyl  disp    hp  drat    wt  qsec    vs    am
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  21       6  160    110  3.9   2.62  16.5     0     1
##  2  21       6  160    110  3.9   2.88  17.0     0     1
##  3  22.8     4  108     93  3.85  2.32  18.6     1     1
```

Manipulation et Analyse de Données

Importation de données à partir de fichiers externes

- Importation de données à partir de fihiers externes de divers formats
- Importation à partir d'un fichier .txt :

```
library(readr)
iris = read_delim("Donnees/iris.txt", delim="\t")
iris
```

```
## # A tibble: 150 x 5
##    `Sepal Length` `Sepal Width` `Petal Length` `Petal W
##             <dbl>         <dbl>          <dbl>      <
##  1            5.1           3.5            1.4
##  2            4.9           3              1.4
##  3            4.7           3.2            1.3
##  4            4.6           3.1            1.5
##  5            5             3.6            1.4
##  6            5.4           3.9            1.7
##  7            4.6           3.4            1.4
##  8            5             3.4            1.5
```

Importation de données à partir de fichiers externes (2)

- Importation à partir d'un fichier .xlsx :

```
library(readxl)
iris = read_excel("Donnees/iris.xlsx")
str(iris)
```

```
## tibble [150 x 5] (S3: tbl_df/tbl/data.frame)
##  $ Sepal Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5
##  $ Sepal Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3
##  $ Petal Length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4
##  $ Petal Width : num [1:150] 0.2 0.2 0.2 0.2 0.2 0.4 0.3
##  $ Species     : chr [1:150] "setosa" "setosa" "setosa"
```

Importation de données à partir de fichiers externes (3)

- Importation à partir d'un fichier SAS :

```
library(haven)
iris = read_sas("Donnees/iris.sas7bdat")
iris
```

```
## # A tibble: 150 x 5
##    Sepal_Length Sepal_Width Petal_Length Petal_Width Spe
##           <dbl>       <dbl>        <dbl>       <dbl> <ch
##  1          5.1         3.5          1.4         0.2 set
##  2          4.9         3            1.4         0.2 set
##  3          4.7         3.2          1.3         0.2 set
##  4          4.6         3.1          1.5         0.2 set
##  5          5           3.6          1.4         0.2 set
##  6          5.4         3.9          1.7         0.4 set
##  7          4.6         3.4          1.4         0.3 set
##  8          5           3.4          1.5         0.2 set
##  9          4.4         2.9          1.4         0.2 set
## 10          4.9         3.1          1.5         0.1 set
```

Importation de données issues du Web (2)

2. Lecture de pages Web
   - Exemple

```
library(rvest)
sw = read_html("https://www.imdb.com/title/tt0076759/")
```

Manipulation de données

- Le "pipe" %>% (package *magrittr*) : ctrl+shift+m + permet d'enchaîner les opérations + écriture plus naturelle et plus lisible

```
x = c(1,4,2,3,6,10,2,11)
mean(x)
```

```
## [1] 4.875
```

```
x %>% mean
```

```
## [1] 4.875
```

- Manipulation de données (package *dplyr*) : ajouts de variables, filtres, organisation, etc. . .

Manipulation de données (2)

- Ajouter à la table de données mtcars le nom des modèles de voiture contenu dans les noms de lignes de la table (nouvelle variable dénommée *model*) :

```
library(tibble)
cars = mtcars %>% rownames_to_column("model")
```

- Filtrer les observations suivant les valeurs de variables

```
library(dplyr)
cars %>% filter(mpg>=30)
cars %>% filter(cyl==4)
cars %>% filter(cyl==4 & qsec >17)
cars %>% filter(between(mpg, 30, 32))
```

Manipulation de données (3)

- Sélection de lignes par leurs indices (numéros)

```
cars %>% slice(1:2)
cars %>% slice(c(2,6))
cars %>% slice(seq(1,n(), by=4))
cars %>% slice(31:n())
```

où $n()$ est le nombre de lignes de la table

- Sélection de variables

```
cars %>% select(model)
cars %>% select(mpg,cyl,model)
cars %>% select(2,5,7)
cars %>% select(starts_with("m"))
```

Voir ?select_helpers pour la liste des possibilités

Manipulation de données (4)

- Tri selon les valeurs d'une ou plusieurs variables

```
cars %>% arrange(mpg)
cars %>% arrange(am,mpg)
cars %>% arrange(desc(mpg))
```

où desc() indique un ordre décroissant.

- Ajout de variables

```
cars %>% mutate(cyl_ratio=cyl/carb)
cars %>% mutate(cyl_ratio=cyl/carb, wt_ratio=wt/hp)
```

- Pour garder uniquement la variable créée

```
cars %>% transmute(cyl_ratio=cyl/carb)
```

Manipulation de données (5)

- Suppression des doublons

```
unique(cars$cyl)
```

```
## [1] 6 4 8
```

```
cars %>% select(cyl) %>% distinct
```

```
##   cyl
## 1   6
## 2   4
## 3   8
```

Manipulation de données (6)

- Enchaîner les opérations

```
cars %>% select(starts_with("c")) %>% summary
```

```
##       cyl             carb
##  Min.   :4.000   Min.   :1.000
##  1st Qu.:4.000   1st Qu.:2.000
##  Median :6.000   Median :2.000
##  Mean   :6.188   Mean   :2.812
##  3rd Qu.:8.000   3rd Qu.:4.000
##  Max.   :8.000   Max.   :8.000
```

```
cars %>% select(mpg,wt,hp) %>% as_tibble %>% head
```

```
## # A tibble: 6 x 3
##     mpg    wt    hp
##   <dbl> <dbl> <dbl>
## 1  21    2.62   110
## 2  21    2.88   110
## 3  22.8  2.32    93
```

Manipulation de données (7)

- Grouper les individus en fonction des modalités d'une variable

```
cars %>% group_by(cyl)
```

- Exemple : déterminer le nombre de modèles de la table *cars* dans chaque catégorie de cylindre

```
cars %>% group_by(cyl) %>% summarise(n=n())
```

```
## `summarise()` ungrouping output (override with `.groups`
```

```
## # A tibble: 3 x 2
##     cyl     n
##   <dbl> <int>
## 1     4    11
## 2     6     7
## 3     8    14
```

Manipulation de données (8)

- Joindre deux tables en fonction des valeurs de l'une des variables

```
engine = tibble(
  cyl = c(6,8,12),
  type = c("medium", "big", "very big")
)
```

Extraction des tables en jointure entre cars et engine :

```
cars %>% inner_join(engine)
cars %>% left_join(engine)
cars %>% right_join(engine)
cars %>% full_join(engine)
cars %>% semi_join(engine)
cars %>% anti_join(engine)
```

TP : Exercice 1 (avec *tibble*, *dplyr* et *readr*)

- Donner la liste des différentes espèces d'iris dans la table *iris*
- Sélectionner
  - les 60 premiers iris de la table
  - les 50 derniers iris de la table
  - 1 iris sur 10 dans la table
- Sélectionner les iris ayant les plus longues Sépales et les plus longues Pétales. En donner la répartition par espèce.
- Créer une nouvelle table en organisant la table *iris* par ordre alphabétique des espèces et par ordre croissant des longueurs de Sépale
- Créer une nouvelle table en ajoutant 2 colonnes à la table *iris* contenant respectivement les rapports longueur/largeur de Sépale et de Pétale
- Importer les 4 tables concernant les indicateurs de gouvernance fournis par la banque mondiale, *WGIValues.csv*, *WGICountry.csv*, *WGISerie.csv* et *WGIType.csv*
- Créer une nouvelle table contenant les noms des pays, séries et indicateurs à la place des codes correspondants

Données spécifiques

- Gestion de chaînes de caractères (package *stringr*) :

```
library(stringr)
str_length(cars$model)
```

```
##  [1]  9 13 10 14 17  7 10  9  8  8  9 10 10 11 18 19 17
## [26]  9 13 12 14 12 13 10
```

```
str_c(cars$model, collapse = ", ")
```

```
## [1] "Mazda RX4, Mazda RX4 Wag, Datsun 710, Hornet 4 Driv
```

```
str_sub(cars$model, 1, 5)
```

```
##  [1] "Mazda" "Mazda" "Datsu" "Horne" "Horne" "Valia" "Du
## [10] "Merc " "Merc " "Merc " "Merc " "Merc " "Cadil" "Li
## [19] "Honda" "Toyot" "Toyot" "Dodge" "AMC J" "Camar" "Po
## [28] "Lotus" "Ford " "Ferra" "Maser" "Volvo"
```

Données spécifiques (2)

- Gestion des expressions régulières (package *stringr*) :

```
str_subset(cars$model, "Merc")
str_subset(cars$model, "[0-9]")
str_detect(cars$model, "[0-9]")
str_match(cars$model, "(.+)[ ](.+)")
str_split(cars$model, " ")
```

TP : Exercice 2 (avec *stringr*)

- Créer le vecteur x composé des 5 chaînes de caractères suivantes : “Je” “suis” “en” “DUT” “STID”
- Ajouter la chaîne “alternance” à la fin du vecteur x
- Donner le nombre de caractères composant chaque élément de x
- Ajouter une majuscule à “alternance” (indication : str_to_title)
- Trouver tous les éléments de x contenant le caractère “e” et les extraire.
- Concaténer les éléments de x afin de ne former qu'une seule phrase. Combien de caractères contient ce nouvel objet ? Interpréter.
- Re-découper l'objet obtenu en chaînes de caractères distinctes

Dates

- Gestion des dates aisée (package *lubridate*)

```
library(lubridate)
now()
```

```
## [1] "2020-07-19 20:21:24 CEST"
```

```
today = today()
today
```

```
## [1] "2020-07-19"
```

```
year(today)
```

```
## [1] 2020
```

Dates (2)

```
month(today)
month(today, label = TRUE)
month(today, label = TRUE, abbr = FALSE)
day(today)
mday(today)
wday(today)
wday(today, label = TRUE)
wday(today, label = TRUE, week_start = 1)
wday(today, week_start = 1)
yday(today)
```

Dates (3)

```
today + period(week = 1, day = 3)
```

```
## [1] "2020-07-29"
```

```
today + period("1W 3D")
```

```
## [1] "2020-07-29"
```

```
today - years(1) - days(10)
```

```
## [1] "2019-07-09"
```

Dates (4)

- Calculer des périodes en secondes ou en années (par défaut en jours) :

```
bday = ymd("19771114")
diff = today - bday
diff
```

```
## Time difference of 15588 days
```

```
as.period(diff)
```

```
## [1] "15588d 0H 0M 0S"
```

```
as.duration(diff)
```

```
## [1] "1346803200s (~42.68 years)"
```

TP : Exercice 3 (avec *tibble*, *dplyr*, *stringr*)

- Afficher le jeu de données *starwars* (package *dplyr*)
- Donner le nombre total de personnages distincts apparaissant dans les trilogies “Star Wars” (indication : nrow)
- Donner le pourcentage de valeurs manquantes par variable (indication : is.na, colSums)
- Faire de même avec les personnages (indication : rowSums)
- Lister les différentes races des personnages de la table *starwars*. Combien de races apparaissent dans les trilogies “Star Wars” ?
- Donner le nombre de personnages dans chaque race, en triant par ordre décroissant de ce nombre
- Choisir 2 races pour lesquelles on dénombre plus de 2 personnages et donner le nom de ces personnages. Quels sont les personnages dont la race est indéfinie ? (indication : is.na)
- **Optionnel** : donner les noms des personnages ayant changé de couleur de cheveux (indication : regarder la variable hair_color et utiliser la librairie *stringr*)
- **Optionnel** : quels personnages apparaissent dans plus de 4 films ? (indication : bind_cols, add_column)

Statistiques sous R : quelques exemples

- Résumer une ou plusieurs variables dans un data.frame :

```
cars %>% summarise(n=n(), mpg_av=mean(mpg),
                   wt_med=median(wt))
```

```
##    n   mpg_av wt_med
## 1 32 20.09062  3.325
```

- Résumer toutes les variables dans un data.frame suivant le même indicateur statistique :

```
mtcars %>% summarise_all(mean)
```

```
##         mpg    cyl     disp        hp     drat       wt
## 1 20.09062 6.1875 230.7219 146.6875 3.596563 3.21725 17.
##      gear   carb
## 1 3.6875 2.8125
```

Statistiques sous R : quelques exemples (2)

- Résumer par groupes définis par les modalités d'une variable :

```
cars %>% group_by(cyl) %>%
  summarise(n=n(), mpg_av=mean(mpg), wt_med=median(wt))
```

```
## `summarise()` ungrouping output (override with `.groups`
```

```
## # A tibble: 3 x 4
##     cyl     n mpg_av wt_med
##   <dbl> <int>  <dbl>  <dbl>
## 1     4    11   26.7   2.2
## 2     6     7   19.7   3.22
## 3     8    14   15.1   3.76
```

Statistiques sous R : quelques exemples (2)

- Utilisation des fonctions descriptives classiques

```
cars %>% select(mpg,wt,qsec) %>% colMeans
```

```
##      mpg       wt     qsec
## 20.09062  3.21725 17.84875
```

```
cars %>% select(mpg,wt,qsec) %>% cor
```

```
##              mpg         wt        qsec
## mpg   1.0000000 -0.8676594  0.4186840
## wt   -0.8676594  1.0000000 -0.1747159
## qsec  0.4186840 -0.1747159  1.0000000
```

Statistiques sous R : quelques exemples (3)

- Calculs d'agrégats sur toutes les variables

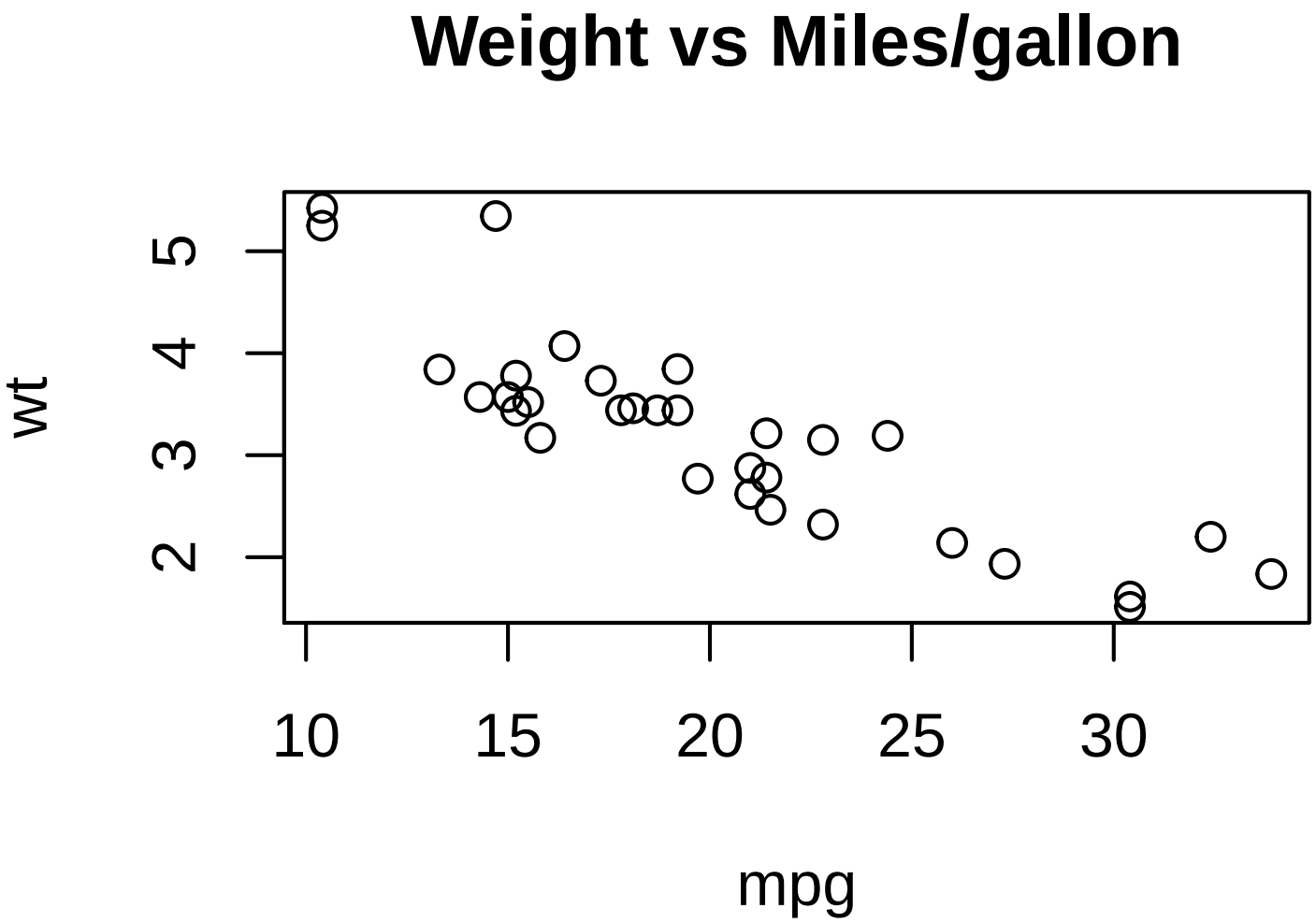
```
cars %>% group_by(cyl) %>%
  select(cyl,mpg,wt,qsec) %>%
  summarise_all(mean)
```

```
## # A tibble: 3 x 4
##     cyl   mpg    wt  qsec
##   <dbl> <dbl> <dbl> <dbl>
## 1     4  26.7  2.29  19.1
## 2     6  19.7  3.12  18.0
## 3     8  15.1  4.00  16.8
```

Statistiques sous R : quelques exemples (4)

- Production de graphiques avec les fonctions de base

```
cars %>% select(mpg,wt) %>%
  plot(main="Weight vs Miles/gallon")
```



TP : Exercice 4 (avec *tibble*, *dplyr*)

- Afficher les indicateurs statistiques résumant les données *iris*
- Créer deux tables résumant ces mêmes données dont les valeurs sont, par espèce, le nombre d'iris, les moyennes et les médianes de chaque variable, l'une concernant les Sépales, l'autre les Pétales
- Créer une seule table rassemblant toutes les informations précédentes
- Représenter la distribution des espèces sous forme de diagramme en barres et de diagramme circulaire (indication : table, barplot, pie)
- Représenter le lien entre la couleur des yeux et la couleur des cheveux des personnages de la base de données *starwars* (indication : table, barplot)