Session_2

Julian Maitra

Session 2

Répétition rapide de session 1

```
# Script de répétition

# Avez-vous compris comment R traite les différents types de variables ?
# Quelle est la différence entre a et b ?

a <- 5
b <- "5"

# Comment afficher a et b ?


# Alternative : avec la fonction print(), qui augmente la compréhension ultérieure du code


# Pourquoi le code suivant produit-il une erreur ?
a*b
```

```
Error in a * b: non-numeric argument to binary operator
```

```
# Le code suivant corrige à nouveau l'erreur, pourquoi ?
b <- 5
a*b

# Vécteurs
v1 <- c(1,2,3,4)
```

```r
v1

# Vous pouvez créez des nombres succesifs avec un colon :
v2 <- c(1:10)
v2

v3 <- c("salut", "le", "pote")
v4 <- c(TRUE, FALSE, FALSE)
```

> **Astuce**
>
> Vous pouvez utiliser la fonction `class()` pour vérifier le type d'un objet, p. ex. (exécutez vous-même !) :
>
> - `class(c(1,2,3))`
> - `class(c("Hello", "World"))`
> - `class(c(TRUE,TRUE,FALSE))`

Données en forme de tableau

Les données sous forme de tableaux, avec des lignes et des colonnes, sont au cœur des méthodes quantitatives dans les sciences de la communication.

Dans R, il existe plusieurs structures de données en format de tableau, notamment :

- Les **matrices**
- Les **data frames**
- D'autres types, tels que les **tibbles**, qui sont en fait des data frames adaptés à des paquets R spécifiques (nous traiterons ce sujet plus tard dans le cours).

> **Important**
>
> - Il est de convention qu'un tableau de données présente les **unités statistiques** par **ligne** (angl. *row*).
> - Les **variables** ou caractéristiques de ces unités statistiques sont indiquées en conséquence par **colonnes** (angl. *column*).

À noter

Selon le type de données, surtout en recherche empirique en sciences sociales dont font partie les sciences de la communication, les unités statistiques peuvent être :

- des **participants** à un questionnaire ou à une expérimentation
- des **documents** (p. ex. articles de presse, posts sur réseaux sociaux)
- des **observations/ cas** (p. ex. des profils sur réseaux sociaux, des entreprises médiatiques)

Dans ces jeux de données, par convention, chaque ligne représente une unité statistique (ou un participant, un document, une observation), tandis que chaque colonne représente les variables qui ont été collectées pour chaque unité statistique.

À noter

La taille d'un jeu de données est indiquée par **le nombre d'unités statistiques n**.

- Par exemple, si 49 personnes ont répondu à un questionnaire, nous parlons donc d'un échantillon de n = 49 participants.

Voici un modèle de la manière dont on devrait structurer les réponses à un questionnaire avec n participants et X questions :

```
          id  question_A  question_B  question_C etc.  question_X
1 personne_1 réponse_a_A réponse_a_B réponse_a_C  ... réponse_a_X
2 personne_2 réponse_a_A réponse_a_B réponse_a_C  ... réponse_a_X
3 personne_3 réponse_a_A réponse_a_B réponse_a_C  ... réponse_a_X
4       etc.         ...         ...         ...  ...         ...
5 personne_n réponse_a_A réponse_a_B réponse_a_C  ... réponse_a_X
```

Dans ce modèle, l'unité statistique de ce jeu de données est la personne qui a répondue au questionnaire (un participant). Chaque ligne correspond donc à une personne différente. Les variables, en revanche, sont les différentes questions qui ont été enregistrées pour chaque participant.

Regardons maintenant quelques objets R sous forme de tableau :

Les matrices

Les **matrices** sont des objets plutôt mathématique. Elles doivent notamment être composé de données de même type, c'est à dire numérique, textuel, logique, etc.

On peut créer des objets matriciels avec la commande **`matrix()`**, avec les arguments suivants :

- un vecteur de valeurs (p. ex. `c(1,2,3,4,5,6)`)
- le nombre de lignes (`nrow=`),
- le nombre de colonnes (`ncol=`) et
- la *direction* de remplissage des valeurs, c'est-à-dire par lignes (`byrow = TRUE`) ou par colonnes (`byrow = FALSE`).

Créons quelques matrices exemplaires avec du code R maintenant.

```r
m1 <- matrix(c(1,2,3,4,5,6), nrow = 2, ncol = 3, byrow = FALSE)

print(m1)


m2 <- matrix(c(1:6), nrow = 2, ncol = 3, byrow = TRUE)

print(m2)

# Quelle est la différence entre m1 et m2 au niveau du code ?
```

> **i** À noter
>
> Si vous ne spécifier pas tous les arguments d'une fonction, R va tenter d'utiliser des arguments par défauts.

```r
# On peut p. ex. créer une matrice sans spécifier tout les arguments de la fonction matrix()
m3 <- matrix(c(1:10), nrow = 5)

print(m3)
```

```
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
```

```
[4,]    4    9
[5,]    5   10
```

Vous pouvez extraire des éléments individuels ou des parties d'une matrice avec des **parenthèses carrées `[]`** en spécifiant les lignes et les colonnes concernées.

```r
# Créons d'abord un matrice m4 avec 100 éléments :
m4 <-  matrix(c(1:100), nrow = 10, byrow = TRUE)
print(m4)
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    2    3    4    5    6    7    8    9    10
 [2,]   11   12   13   14   15   16   17   18   19    20
 [3,]   21   22   23   24   25   26   27   28   29    30
 [4,]   31   32   33   34   35   36   37   38   39    40
 [5,]   41   42   43   44   45   46   47   48   49    50
 [6,]   51   52   53   54   55   56   57   58   59    60
 [7,]   61   62   63   64   65   66   67   68   69    70
 [8,]   71   72   73   74   75   76   77   78   79    80
 [9,]   81   82   83   84   85   86   87   88   89    90
[10,]   91   92   93   94   95   96   97   98   99   100
```

```r
# Extraire le deuxième élément de la troisième ligne :
m4[3,2]
```

```
[1] 22
```

```r
# Extraire les cinq premiers éléments de la dixième ligne :
m4[10, c(1:5)]
```

```
[1] 91 92 93 94 95
```

```r
# Extraire tous les éléments de la première ligne :
m4[1,]
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```r
# Extraire tous les éléments de la troisième colonne :
m4[,3]
```

```
 [1]  3 13 23 33 43 53 63 73 83 93
```

```r
# Extraire la moitié supérieure de la matrice :
m4[c(6:10),]
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]   51   52   53   54   55   56   57   58   59    60
[2,]   61   62   63   64   65   66   67   68   69    70
[3,]   71   72   73   74   75   76   77   78   79    80
[4,]   81   82   83   84   85   86   87   88   89    90
[5,]   91   92   93   94   95   96   97   98   99   100
```

Les data frames

> **i** À noter
>
> Les **data frames** (français : *cadre de données*) sont le type de donnée en forme tableau le plus commun dans R. Cette structure de donnée peut notamment être composé de données de différents types (numérique, textuel, logique).

Créons un data frame avec la fonction `data.frame()` maintenant. Nous appelons ce type d'objets désormais un **jeu de données**.

```r
# Créons d'abord un vecteur textuel avec les noms de certains animaux du film L'âge de glace
name <- c("Manny", "Sid", "Diego", "Ellie",
          "Peaches", "Scrat", "Rudy", "Buck")

#Un second vecteur textuel indique l'espèce de chaque animal.
species <- c("mammoth", "sloth", "sabertooth", "mammoth",
             "mammoth", "squirrel", "dinosaur", "weasel")

# Un troisième vecteur logique indique si chaque animal est une femelle
female <- c(FALSE, FALSE, FALSE, TRUE,
            TRUE, FALSE, FALSE, FALSE)

# Un quatrième vecteur numérique indique l'âge en années de chaque animal.
age <- c(30, 20, 35, 25,
```

```
         7, 4, 100, 28)

# Un cinquième tableau numérique indique le poids en kg de chaque animal.
weight <- c(5000, 80, 400, 4000,
            1000, 1, 10000, 3)

# La fonction data.frame() permet d'utiliser les vecteurs pour former un tableau de données.
ice_age_df <- data.frame(name, species, female, age, weight)
```

Nous pouvons afficher l'objet `ice_age_df` en exécutant son nom ou en utilisant la fonction `print()` :

```
print(ice_age_df)
```

```
     name    species female age weight
1   Manny    mammoth  FALSE  30   5000
2     Sid      sloth  FALSE  20     80
3   Diego sabertooth  FALSE  35    400
4   Ellie    mammoth   TRUE  25   4000
5 Peaches    mammoth   TRUE   7   1000
6   Scrat   squirrel  FALSE   4      1
7    Rudy   dinosaur  FALSE 100  10000
8    Buck     weasel  FALSE  28      3
```

Quelques fonctions utiles pour analyser les data frames

Il existe plusieurs **fonctions utiles** dans R qui vous permettent d'analyser les propriétés de jex de données de type data frame (et d'autres types d'objets R), comme p. ex. :

- `dim()` : Retourne les dimensions d'un jeu de données - le nombre de lignes et de colonnes.
- `head()` : Affiche les six premières lignes du jeu de données.
- `str()` : Fournit la structure du jeu de données, y compris
  - le type de variable de chaque colonne,
  - le nombre d'observations et
  - les premières entrées de chaque colonne.

    C'est un moyen rapide de se faire une idée du contenu de votre jeu de données.
- `summary()`: Donne un résumé du jeu de données, y compris des statistiques telles que

- la moyenne,
- la médiane,
- le minimum et le maximum pour les variables numériques, et
- les fréquences pour les variables factorielles.

  Cette fonction est utile pour obtenir rapidement une vue d'ensemble des statistiques.

```
# Utilisons ces fonctions pour analyser l'objet ice_age_df

dim(ice_age_df)
```

```
[1] 8 5
```

```
head(ice_age_df)
```

```
     name    species female age weight
1   Manny    mammoth  FALSE  30   5000
2     Sid      sloth  FALSE  20     80
3   Diego sabertooth  FALSE  35    400
4   Ellie    mammoth   TRUE  25   4000
5 Peaches    mammoth   TRUE   7   1000
6   Scrat   squirrel  FALSE   4      1
```

```
str(ice_age_df)
```

```
'data.frame':   8 obs. of  5 variables:
 $ name   : chr  "Manny" "Sid" "Diego" "Ellie" ...
 $ species: chr  "mammoth" "sloth" "sabertooth" "mammoth" ...
 $ female : logi  FALSE FALSE FALSE TRUE TRUE FALSE ...
 $ age    : num  30 20 35 25 7 4 100 28
 $ weight : num  5000 80 400 4000 1000 1 10000 3
```

```
summary(ice_age_df)
```

```
     name             species            female             age
 Length:8           Length:8           Mode :logical   Min.   :  4.00
 Class :character   Class :character   FALSE:6         1st Qu.: 16.75
 Mode  :character   Mode  :character   TRUE :2         Median : 26.50
```

```
                                                    Mean   : 31.12
                                                    3rd Qu.: 31.25
                                                    Max.   :100.00
    weight
 Min.   :    1.00
 1st Qu.:   60.75
 Median :  700.00
 Mean   : 2560.50
 3rd Qu.: 4250.00
 Max.   :10000.00
```

> **Astuce**
>
> Une autre astuce utile consiste à utiliser le `$` pour **extraire des colonnes spécifiques** d'un jeu de données.

```
# P. ex. :
ice_age_df$name
```

```
[1] "Manny"   "Sid"     "Diego"   "Ellie"   "Peaches" "Scrat"   "Rudy"
[8] "Buck"
```

```
# Essayez vous-même et extrayez d'autres colonnes de ice_age_df à l'aide du $ !
```

Fonctions statistiques de base

Il existe dans R toute une palette de fonctions statistiques qui vous permettent d'évaluer principalement des données de type numérique. Quelques fonctions utiles pour les statistiques descriptives de base :

- `mean()` : Calcule la moyenne arithmétique
- `median()` : Calcule la médiane
- `sd()` : Calcule l'écart-type (angl. *standard deviation*)
- `var()` : Calcule la variance

```r
a1 <- c(1,2,100)

mean(a1)

median(a1)

sd(a1)

var(a1)
```

```
[1] 34.33333
```

```
[1] 2
```

```
[1] 56.8712
```

```
[1] 3234.333
```

Les facteurs

Un autre type de variable clé dans R est ce que l'on appelle les **facteurs**, qui sont essentiellement des **variables catégorielles**.

> **i** À noter
>
> Les facteurs sont des variables qui prennent un nombre limité de valeurs différentes (appelées *levels* dans R).
> Il existe deux sous-type de facteurs :
>
> - Les **facteurs nominaux** (sans ordre) : toutes les catégories sont équivalentes les unes aux autres. Il n'y a pas de hiérarchisation.
>   - Exemples : le genre et l'appartenance à un parti
> - Les **facteurs ordinaux** (avec un ordre): il y a un ordre ou une hiérarchie spécifié parmi les catégories (p. ex. des niveaux tels que *faible*, *moyen*, *élevé*).
>   - Exemples : niveau d'éducation, cohortes d'âge

Vous pouvez créer un facteur nominal à l'aide de la fonction `factor()` et vérifier ses catégories avec `levels()`. Voici un exemple :

```r
# Créer un vecteur de données catégorielles, d'abord de type textuel
genre <- c("homme", "femme", "femme", "femme", "homme")

# Le convertir ensuite en facteur
facteur_genre <- factor(genre)

# Voir les niveaux du facteur genre
levels(facteur_genre)
```

```
[1] "femme" "homme"
```

```r
# Imprimer le facteur (montre aussi les niveaux)
print(facteur_genre)
```

```
[1] homme femme femme femme homme
Levels: femme homme
```

Et voici comment créer un facteur ordinal :

```r
# Voici un vecteur textuel avec des tailles de T-shirt
tailles <- c("Small", "Medium", "Large", "Medium", "Small")

# On le transforme en facteur ordinal en spécifiant les arguments suivants :
tailles_facteur <- factor(tailles, order = TRUE, levels = c("Small", "Medium", "Large"))

levels(tailles_facteur)
```

```
[1] "Small"  "Medium" "Large"
```

```r
print(tailles_facteur)
```

```
[1] Small  Medium Large  Medium Small
Levels: Small < Medium < Large
```

Transformons maintenant la colonne `species` de notre jeu de données `ice_age_df` en facteur avec la fonction `as.factor()`:

```r
# Vérifions d'abord la structure de l'objet ice_age_df :
str(ice_age_df)
```

```
'data.frame':   8 obs. of  5 variables:
 $ name   : chr  "Manny" "Sid" "Diego" "Ellie" ...
 $ species: chr  "mammoth" "sloth" "sabertooth" "mammoth" ...
 $ female : logi  FALSE FALSE FALSE TRUE TRUE FALSE ...
 $ age    : num  30 20 35 25 7 4 100 28
 $ weight : num  5000 80 400 4000 1000 1 10000 3
```

```r
# Transformation en facteur :
# ice_age_df$species <- as.factor(ice_age_df$species)

# Vérifions encore une fois la structure de l'objet :
# str(ice_age_df)
```

C'est maintenant à vous de transformer la colonne `name` de `ice_age_df`en facteur !

Le répértoire de travail

> **!** Important
>
> Dans R, le **répertoire de travail** (angl. *working directory*) fait référence au dossier de votre ordinateur dans lequel R lit et enregistre les fichiers par défaut. Il s'agit d'un concept crucial dans la programmation R, car il détermine l'endroit où R recherche les fichiers à lire et l'endroit où il place les fichiers lorsque vous les enregistrez.

Vous pouvez définir votre répertoire de travail avec la fonction **`setwd()`**.

Une possibilité, c'est de le faire manuellement dans le volet *output* en bas à droite :

1. Naviguez sur l'onglet **Files**
2. Cliquez sur les trois petits points **...** tout à droite
3. Choisissez un dossier sur votre ordinateur qui peut servir de répertoire de travail (vous devez peut-être en créer un d'abord et lui donner un nom utile, p.ex. "ex_meth_sp24"
4. Cliquez ensuite sur la petite **roue bleue** (More) et choisissez **Set as Working Directory**
5. Vous pouvez ensuite copier et coller le code R qui définit le répertoire de travail depuis l'onglet de la console et l'inclure dans votre script R

Sur l'ordinateur de l'enseignant, ce code est le suivant : `setwd("C:/Users/julim/switchdrive/001_Exercices`
Attention : Ce code ne marche naturellement pas sur votre machine. Il faut l'adapter et inclure un chemin de fichier de votre ordinateur.

Vous pouvez ensuite vérifier si le répertoire de travail a été correctement défini avec la fonction `getwd()`

Devoir pour Session 3 (Problem Set 1)

- **Problem Set 1**
  - Le Problem Set 1 est un QCM sur Moodle qui sera accessible pendant les 30 premières minutes de l'exercice.
  - Assurez-vous que R et RStudio fonctionnent sur votre ordinateur portable et qu'il a suffisamment de batterie. Il est de votre responsabilité de vous en assurer.
  - Dans le Problem Set 1, il y a à la fois des questions que vous devez résoudre avec du code R et des questions de compréhension qui peuvent être résolues sans code. Vous devez donc être capable non seulement d'exécuter le code, mais aussi d'en comprendre l'objectif.
  - Répétez le contenu des sessions 1 et 2 sur votre ordinateur afin de bien vous préparer pour le Problem Set 1.
  - Assurez-vous que toutes les parties du code fonctionnent sans message d'erreur.
- **Attention** :
  - Vous devez tous venir la semaine prochaine dans le groupe dans lequel vous êtes inscrits via my.unifr (A, B, C, D) si vous voulez participer au Problem Set 1.
  - Chaque étudiant.e n'aura accès qu'au Problem Set 1 de son groupe.
  - Les personnes qui ne sont inscrites dans aucun groupe ne peuvent pas valider le Problem Set 1.