Introduction à R

Crash course en moins de 10 minutes

Qu'est ce que R ?

- Un langage informatique créé en 1993, géré par le projet GNU (l'interpréteur qui permet de faire tourner le code est libre)
- C'est un langage interprété, par opposition aux langages compilés
- Vous pouvez lancer un terminal, et lancer l'interpréteur avec la commande « R », vous pouvez ensuite lui passer directement des commandes à exécuter
- Comme « une grosse calculette »

Qu'est ce qu'une base de données ?

- Dans sa forme la plus simple, une base de données est un fichier texte « pur ». Il présente une structure définie (et donc prévisible), qui permet un traitement programmatique du fichier.
- Les fichiers .csv se structurent de la manière suivante :

```
colonne1_ligne1,colonne2_ligne1,colonne3_ligne1
colonne1_ligne2,colonne2_ligne2,colonne3_ligne2
```

- Les « scripts » R sont aussi des fichiers texte purs , que l'on va passer à l'interpréteur.
- Le script est la recette, la base de données les ingrédients, l'interpréteur le cuisinier. Nous, on écrit la recette.

Pourquoi on utilise pas Excel?

- Avoir un script présente certains avantages :
  - On garde une trace de tout ce qui est fait sur la base, et on peut revenir modifier des bouts à loisir, vu qu'on ne modifie jamais « pour de vrai » la base.
  - On peut copier coller d'autres bouts de code.
  - On peut s'interfacer de manière programmatique avec le code des autres : packages R, voire librairies dans d'autres langages.
  - On peut automatiser facilement les opérations répétées

Interlude exemple

Avec le terminal

! Important : Bien qu'exécuté en temps réel, les résultats fournis par l'interpréteur ne sont stockés nulle part. Si on veut « garder » temporairement le résultat d'une opération, il faut le stocker dans une variable, si on veut le garder de manière plus longue, il faut le stocker dans un fichier !

Quelques notions de base

Les ***objets*** représentent les blocs primaire avec lesquels vous interagissez dans R, le symbole d'attribution est <- ou =, la déclaration est dynamique. On peut y **stocker**:

- Des variables, des suites de chiffres, des chaînes de caractères, en gros, « des suites de trucs qu'on stocke pour les avoir sous la main »:

```
Var_1 <- c(28,32,14,56)
```

- Des fonctions:

```
Fun_1 <- function(arg1, arg2){return(print(c(arg1,arg2)))}
```

- D'autres objets:

```
List_1 <- list(Var_1, Fun_1)
```

- Des résultats de fonctions:

```
Regression_results <- glm(var1 ~ var2 + var3, family = « binomial »)
```

Quelques notions de base

Ils sont:

- **Indexables, avec [ ]**:

`Var_1[1]` donne la première valeur de la variable Var_1

`Data_frame[1,2]` donne la première valeur de la seconde colonne de la table Data_frame.

`Data_frame$Var_1` donne la colonne « Var_1 » de la table Data_frame.

- **Appelables** par les fonctions:

`mean(Var_1)` donne la moyenne de la variable Var_1

- Et **interopérables**:

New_object <- num_1 + (num_2 – 2)

additionne les variables num 1 et num 2 + 2, puis les stocke dans un nouvel objet

Ou encore :

`New_object <- paste(carac_1,  num_1, sep = « _ »)`

Qui « colle » la chaine de caractères carac 1 aux nombres de num 1.

Quelques notions de base

Les objets R sont typés. L'usage du bon type est important: des contraintes sont appliquées sur le contenu des objets; + certaines fonctions attendent certains types d'objets. On distingue:

- Les objets **numériques**. Définis par: `numeric()` ou `as.numeric()`: suite de nombres entiers ou décimaux libres.

- Les objets **caractères**. Définis par `character()` ou `as.character()`: suite de chaînes de caractères libres.

- Les objets **facteur**. Définis par `factor()` ou `as.factor()`. Représentent conceptuellement les variables catégorielles.

- Les objets synthétiques comme les listes ou les data.frames, qui ne sont que des collections d'objets unitaires décrits plus haut, ont leurs propres types spécifiques.

Quelques notions de base

Les fonctions ne sont elles même que des objets contenant le code de la suite d'instructions à appliquer pour obtenir le résultat souhaité.

- Elle « s'appellent » (se convoquent) de la manière suivante :

```
Fonction ( Arguments )
# Les arguments permettent de spécifier : les données sur lesquelles ont veut faire tourner la fonction, et les options liées  à la fonction.
```

- Elles prennent des arguments spécifiques, dans un ordre spécifique:

```
glm(formula = (Ici entrer le modèle),
family = (Ici la famille de lois stat))
```

- **Le plus important à retenir:** ? Suivi du nom de la fonction donne la liste des arguments attendus par la fonction, ainsi qu'une brève description de celle-ci, des exemples concrets d'application, etc.