# L3 Eco-Fi - Informatique

Exercices

Ewen Gallic

2021-2022

## 1 Données

### 1.1 Structures de données

1. Que permet une liste par rapport à un vecteur ?
2. Quel sera le retour de `c("deux", 1, TRUE)` ? de `c(0, FALSE)` ? de `c(0, "FALSE")` ?
3. Créer un vecteur de facteurs, de taille 5, dont les modalités sont les suivantes : "Google", "Instagram", "Twitter". La modalité de référence doit être "Google". Toutes les modalités doivent être présentes au moins une fois ;
4. Créer la matrice suivante sous `R` :

$$\begin{bmatrix} 39 & 66 & 13 \\ 66 & 168 & 28 \\ 13 & 28 & 5 \end{bmatrix}$$

5. En utilisant la matrice de la question 4, la transformer en data frame ;
6. Créer la matrice de la fonction 4 à l'aide de la fonction `data.frame()`.

### 1.2 Importation de données

1. Télécharger le fichier phbirths.dat, et le placer dans le répertoire de travail ;
2. Lire le fichier téléchargé à l'aide de la fonction `read.table()`.
3. Idem avec la fonction `read_table()`.
4. Idem avec la fonction `scan()`.

### 1.3 Manipulation de données : opérateurs

Soient les observations `x <- c(2, 5, 6, 3, 2)`.

1. Multiplier `x` par 10 ;
2. Normaliser les données (utiliser les fonctions `min()` et `max()`) :

$$x_i = \frac{x_i - min(x)}{max(x) - min(x)};$$

3. Convertir les valeurs de `x` pour obtenir leur proportions (utiliser la fonction `sum()`) ;
4. Tester si les valeurs présentes appartiennent à l'ensemble

$$\{2, 3, 4\};$$

5. Tester si les valeurs sont supérieures ou égales à 2 ;
6. Tester si les valeurs sont inférieures ou égales à 5 ;
7. Tester si les valeurs sont comprises entre 2 et 5 inclus.

## 1.4 Manipulation de données : extractions

1. Extraire le troisième élément du vecteur `x <-c(3, 5, 6, 9)` ;
2. Extraire les éléments supérieurs à 5 ;
3. Extraire les éléments qui ne sont pas dans l'intervalle $[1, 3]$ ;
4. Déterminer le maximum du vecteur `x` et donner sa position ;
5. Remplacer ce maximum par le minimum ;
6. Remplacer par la valeur `NA` les éléments négatifs de la matrice suivante :

$$\begin{bmatrix} 1 & -1 & 5 \\ 5 & 7 & -9 \end{bmatrix};$$

7. Extraire la deuxième colonne de la matrice obtenue à l'issue de la question 6 ;
8. Extraire le 3e élément de la liste suivante :

   `L <- list(rnorm(10), LETTERS[1:26], month.abb)` ;
9. Extraire tous les éléments de la liste `L` sauf le second.
10. Extraire le troisième élément de la liste `L`, en utilisant le nom de l'élément.

    `L <- list(nombre = rnorm(10), lettre = LETTERS[1:26], mois = month.abb)` ;
11. Toujours en accédant par le nom, ajouter la valeur `10` aux valeurs du premier élément de `L`. Le résultat doit altérer `L` ;
12. Afficher les valeurs de la variable `y` du tibble `tb` :

    `tb <- tibble(x = 1:10, y = letters[1:10], z = rev(letters[1:10]))` ;
13. Afficher uniquement les colonnes `x` et `z` du tibble `tb`.

## 1.5 Manipulation de données : chaînes de caractères (1)

1. Créer le vecteur de chaînes commençant par `joueur_` et se terminant par un entier `i` allant de `1 à 10` ;
2. Créer la chaîne composée des deux sous-chaînes `Criquette Rockwell` et `Brett Montgomery`, séparées par un retour à la ligne. Afficher le résultat dans la console avec la fonction `cat()` ;
3. Créer la chaîne `J'aime les pommes - "Ryuk"`, et stocker le résultat dans l'objet `light` ;
4. Passer la chaîne de la question 3 (`light`) en majuscules ;
5. Afficher le contenu de `light` en faisant appel à la fonction `get()`.

## 1.6 Manipulation de données : chaînes de caractères (2)

1. Quelle fonction utiliser pour connaître le nombre de caractères d'une chaîne ?
2. Extraire `c("LON-1.6794", "LAT48.1147")` les valeurs numériques de longitue et latitude ;
3. Dans la chaîne `LON-1.6794`, remplacer `LON` par `LONG` ;
4. Repérer les éléments du vecteur `c("Finistere:I. d'Ouessant", "Finistere:I. de Batz", "Gard")` qui contiennent la sous-chaîne `finistere`, en ne tenant pas compte de la casse ;
5. Remplacer toutes les occurences de tirets (`-`) par une espace dans les éléments du vecteur `c("02-23-23-35-35", "02-23-23-35-45")` ;
6. Découper les numéros de téléphone `c("02-23-23-35-35", "02/23/23/35/45")` en fonction du tiret ou de la barre oblique.

## 1.7 Manipulation de données : dates

1. Avec la fonction appropriée du *package* {`lubridate`}, créer un objet contenant la date du jour (sans les heures, minutes et secondes) ;
2. Idem avec l'heure du jour et les minutes (mais pas les secondes) ;
3. Convertir la chaîne de caractères `x <- "17/05/2020"` en objet de type `Date` ;
4. Stocker sous forme de `POSIXct` la date initialement donnée dans la chaîne de caractères suivate : `"Thu Sep 18 23:40:54 +0000 2020"` ;
5. À partir de la date obtenue en question précédente, obtenir l'heure équivalente à New York City ;
6. Calculer le nombre de jours séparant votre date de naissance et aujourd'hui ;
7. Créer une séquence de dates allant de "1975-01-01" à "2021-01-01", par pas de 6 mois.
8. À partir de `dd`, trouver la date une semaine auparavant (en termes de durées, et en termes d'époques).

```
dd <- "2012-03-02 23:40:54 +0000 2020"
```

## 1.8 Tableaux de données

Soient deux tableaux de données `X` et `Y` :

```
(X <- tibble(mois = month.name[c(1,6,1,6)], annee = c(2014, 2014, 2015, 2015),
             val_1 = 1:4, val_2 = 5:8))
```

```
## # A tibble: 4 x 4
##   mois    annee val_1 val_2
##   <chr>   <dbl> <int> <int>
## 1 January  2014     1     5
## 2 June     2014     2     6
## 3 January  2015     3     7
## 4 June     2015     4     8
```

```
(Y <- tibble(mois = rev(month.name[c(1,6,1,6)]), annee = c(2014, 2014, 2015, 2015),
             val_3 = 9:12, val_2 = 13:16))
```

```
## # A tibble: 4 x 4
##   mois    annee val_3 val_2
##   <chr>   <dbl> <int> <int>
## 1 June     2014     9    13
## 2 January  2014    10    14
## 3 June     2015    11    15
## 4 January  2015    12    16
```

1. Créer un tableau `Z` qui contient `X` et `Y` côte-à-côte ;
2. Que se passe-t-il si on évalue l'instruction suivante : `rbind(X,Y)` ;
3. En utilisant une fonction appropriée, fusionner les tibbles `X` et `Y` par année et par mois ;
4. À l'aide de la fonction appropriée, ajouter à `X` les valeurs de la colonne `val_5` du tibble `Z`, en effectuant l'appariement sur la colonne des années :

```
(Z <- tibble(year = c(2013, 2014), val_5 = c("A", "B")))
```

```
## # A tibble: 2 x 2
##    year val_5
##   <dbl> <chr>
## 1  2013 A
## 2  2014 B
```

## 1.9 Tableaux de données : agrégations

1. À partir du tibble `chomage`, calculer la moyenne du nombre de chômeurs par région et par année ;
2. Même question en effectuant le calcul uniquement pour l'année 2010 ;
3. Ajouter une colonne indiquant le carré du nombre de chomeurs pour les ouvriers, puis calculer la moyenne de cette colonne par année et par région.

## 1.10 Tableaux de données : pivot

Soit le tableau de données suivant :

```
tb <- tibble(region = rep(c(rep("Bretage",4), rep("Corse", 2)), 2),
             dep = rep(c("Cotes-d'Armor", "Finistere",
                         "Ille-et-Vilaine", "Morbihan",
                         "Corse-du-Sud", "Haute-Corse"), 2),
             annee = rep(2012:2013, each = 6),
             ensoleillement = c(1586, 1545, 1762, 1792, 2933, 2735,
                                1639, 1661, 1729, 1770, 2715, 2606),
             precipitations  = c(733, 1311, 788, 920, 876, 676,
                                 679, 1166, 736, 788, 871, 729))
```

1. Creer les variables `ensoleillement_c` et `precipitations_c` dans le tibble `tb`, en utilisant la fonction `mutate()`. Ces deux variables doivent représenter l'écart à la moyenne de la variable `ensoleillement` et `precipitations` respectivement ;
2. Calculer la moyenne de précipitations pour chaque département et chaque année ;
3. Trier le résultat de la question précédente par année décroissante et ordre alphabétique des régions ;

4. À partir de `tb`, créer un tibble en long, où chaque ligne correspond à la durée d'ensoleillement ou les précipitations par région, département et année. Chaque ligne doit indiquer si la valeur concerne l'ensoleillement ou les précipitations.

# 2 Fonctions

## 2.1 Une première fonction

Créer une fonction qui, pour deux arguments `x` et `y` retourne le résultat suivant :

$$\frac{xy}{x^2+y^2}.$$

Tester cette fonction pour les valeurs $(x = 0, y = 0)$, puis $(x = 1, y = 2)$.

## 2.2 Une deuxième fonction

Soient deux points $x$ et $y$, de coordonnées $(x_1, x_2)$ et $(y_1, y_2)$, respectivement.

Créer une fonction à deux arguments, `x` et `y`, qui retourne la distance euclidienne entre `x` et `y`, c'est-à-dire qui calcule :

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Tester cette fonction dans les 2 situations suivantes :

1. `x=c(0,0)` et `y=c(1,1)`
2. `x=c(1,1)` et `y=c(1,1)`.

## 2.3 Calcul de la densité

1. Créer une fonction, qui étant donné un quantile $x$, une espérance $\mu$ et un écart-type $\sigma$, retourne la densité de la loi normale. Comparer avec la fonction `dnorm()`.
2. Reprendre la fonction de la question 1, et donner les valeurs par défaut `0` et 1 pour l'espérance et l'écart-type respectivement.

Rappel : la fonction de densité d'une loi normale est donnée par :

$$\varphi(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

## 2.4 Modèle exponentiel

Considérons un phénomène croissant que l'on modélise à l'aide d'un modèle logistique. La quantité que l'on modélise, $C$ dépend du temps $t$ et de trois paramètres :

- $K$ : la valeur de $C$ quand $t$ tend vers l'infini
- $\tau$ : la valeur du temps tel que $C(\tau) = K/2$ (valeur de $t$ au point milieu de la sigmoïde)
- $r$ : le taux de croissance

La fonction est la suivante :

$$C(t) = \frac{K}{1 + \exp\left(-r(t - \tau)\right)}$$

Créer une fonction en `R` qui retourne, en fonction du temps `t` et de valeurs pour les paramètres `K`, `r` et `t`, la valeur `C(t)`.

Pour tester la fonction, utiliser par exemple les valeurs suivantes : `K = 155074`, `tau = 76.847`, `r=0.094` pour une valeur de `t=10`.

## 2.5 Création de fonctions (1)

1. Créer une fonction nommée `somme_n_entiers` qui retourne la somme des $n$ premiers entiers. Son seul argument sera `n` ; 2.Utiliser la fonction `somme_n_entiers()` pour calculer la somme des 100 premiers entiers ;
2. Terminer la fonction par l'assignation du résultat dans un objet nommé `res`, puis évaluer l'expression suivante : `somme_n_entiers(100)`. Que peut-on constater ?
3. Charger les données `diamonds` du *package* {ggplot2} dans la session `R` à l'aide de l'expression suivante : `data(diamonds, package = "ggplot2")`

   Créer une fonction que l'on appellera `prix_diamant_coupe()`, qui, quand on lui fournit la valeur de la coupe du diamant sous forme de caractères (`Fair`, `Good`, `Very Good`, `Premium`, ou `Ideal`), filtre le tableau de données `diamonds` pour ne conserver que les observations pour lesquelles la coupe du diamant correspond à celle indiquée en argument, et retourne le prix moyen des observations de la base ainsi filtrée ;
4. Reprendre le code de la fonction précédente, et le modifier pour retourner à présent une liste de deux éléments : (i) la moyenne des prix et (ii) l'écart-type ;
5. Créer la fonction `resume_diamant_coupe_couleur()`, qui pour une coupe et une couleur de diamant données, retourne une liste de deux éléments : (i) la moyenne des prix et (ii) l'écart-type pour les diamants possédant cette coupe et cette couleur (la couleur du diamant est une lettre allant de `J` pour les pires, à `D` pour les meilleurs). Tester la fonction pour la coupe `Fair` et la couleur `D` ;
6. Reprendre la fonction précédente, et lui attribuer la valeur `D` (en chaîne de caractères) comme argument effectif pour la couleur. Tester alors l'appel à la fonction en précisant :

- la coupe `Fair` et la couleur `D`,
- la coupe `Fair`, mais pas d'argument pour la couleur,
- la coupe `Fair` et la couleur `E`,
- la coupe non précisée mais la couleur `E` ;

## 2.6 Création de fonctions (2)

Supposons que les adresses e-mails des étudiant•e•s d'Aix-Marseille Université sont constituées de la manière suivante : le prénom et le nom de famille séparés par un point, le symbole arobase et le enfin le nom de domaine. Supposons de plus que les étudiant•e•s ont un seul prénom, et aucune particule au nom de famille. La syntaxe des adresses e-mail est donc comme suit : `nom.prenom@etu.univ-amu.fr`.

```
emails <- c("marie.petit@etu.univ-amu.fr",                "jean.dupont@etu.univ-amu.fr",
"isabelle.martinez@etu.univ-amu.fr",                "pierre.moreau@etu.univ-amu.fr")
```

Créer une fonction, qui à partir d'**une seule** adresse e-mail d'un•e étudiant•e, retourne un tibble contenant trois variables : le prénom, le nom et l'adresse e-mail de cet•te étudiant•e.

# 3 Boucles

## 3.1 Boucle while

Considérons le vecteur suivant :

```
x <-c(2,1,4,5,2,1,6)
```

À partir de combien d'éléments de `x` la somme cumulée devient-elle supérieure ou égale à 10 ? Pour répondre à cette question, utilisez une boucle `while`:

- à chaque itération, la somme cumulée doit être mise à jour pour venir ajouter à la valeur précédente la valeur courante du ie élément de `x`
- un itérateur doit être créé
- la somme cumulée doit être initialisée à `0`

## 3.2 Pile ou face

1. Calculer, dans une expérience de "pile"ou "face", le nombre de tirages alétoires nécessaires avant d'obtenir le premier "pile".

Note : penser à utiliser la fonction `sample()`.

## 3.3 Boucle for

1. Utiliser une boucle `for` pour calculer, sur 50 tirages de "pile" ou "face" avec une pièce de monnaie combien de "pile" sont obtenus.

Note : penser à utiliser la fonction `sample()` pour effectuer le tirage.

## 3.4 Expressions conditionnelles

1. Créer une fonction qui retourne `TRUE` si un nombre `x` est divisible par une autre nombre `diviseur`, et `FALSE` sinon.

## 3.5 Chargement de fichiers à l'aide d'une boucle for

Neuf (9) fichiers CSV sont présents dans le dossier situé à l'adresse suivante : egallic.fr/www/Enseignement/R/Exercices/Loc

La structure de chacun de ces fichiers CSV est identique d'un fichier à l'autre :

- 3 colonnes (`x`, `y` et `row_number`)
- 1000 lignes.

Le nom des fihciers suit la syntaxe suivante : `loop_file_xxx.csv`, où `xxx`vaut `001`, `002`, ..., `009`.

L'objectif est de charger ces 9 fichiers à l'aide d'une boucle et d'empiler leur contenu dans un seul tableau de données.

1. Créez une liste vide nommée `df_l`, de longueur 9, à l'aide de la fonction `vector()`.
2. Créez une chaîne de caractères nommée `i` contenant la valeur `1`.
3. À l'aide de la fonction `str_pad()`, créez la chaîne de caractères `i_pad` formattant la valeur de `i` de manière à ajouter des "0" devant la valeur de `i` ; `i_pad` doit avoir une longueur de 3 caractères. Exemple : Si `i` vaut 1, alors `i_pad` doit valoir `001`.
4. Créez un lien vers le ie fichier, en vous appuyant sur la valeur de `i_pad` (concaténation).
5. Importez le ie fichier CSV dans R, en fournissant le lien créé.
6. Stockez le contenu importé dans le ie élément de la liste `df_l`.
7. À l'aide d'une boucle, importez les 9 fichiers CSV et stockez le résultat dans `df_l`.

### 3.6 Calculs sur des éléments de liste

1. Créer une liste de longueur 5 dans laquelle chaque élément doit être composé d'un échantillon d'observations issues d'une loi Normale centrée réduite ;
2. Sur chaque échantillon de l'objet créé, calculer la moyenne et l'écart-type. Le résultat doit être sous forme d'un tableau de données.

## 4 Graphiques

### 4.1 Nuage de points

1. Créer un nuage de points représentant la durée des films en fonction de leur budget :

- la couleur devra correspondre au budget estimé,
- tester plusieurs valeurs de taille pour les points,
- tester plusieurs valeurs de formes pour les points,
- tester différentes valeurs de transparence.

### 4.2 Paramètres esthétiques et géométries

À partir du jeu de données `diamonds` :

1. Créer le tableau de données `diamonds_s`, *i.e.*, un échantillon de `diamonds` de taille $n = 1000$
2. Représenter le nuage de points du prix (`price`) en fonction de la masse (`carat`), afficher les points en rouge ;
3. Reprendre le graph de la question 2 en colorant les points en fonction de la coupe (`cut`) ;
4. Afficher l'histogramme des masses des diamants (`carat`), avec une fenêtre de `0.05` ;
5. Afficher les boîtes à moustaches de la masse (`carat`) pour chaque coupe (`cut`).

### 4.3 Représentation de distributions

1. Afficher l'histogramme des masses des diamants (`carat` dans le `data.frame` `diamonds`) ;
2. Afficher une estimation de la densité de la masse des diamants ;
3. Faire figurer sur un même graphique une visualisation de la répartition des masses de diamants, ainsi qu'une estimation de la densité.

### 4.4 Échelles

## 5 Exercices

1. Afficher un nuage de points de la masse des diamants (`carat`) en fonction du prix associé (`price`) ;

2. Reprendre ce graphique et faire dépende la couleur des points de la qualité de la coupe (`cut`) ;
3. Changer le nom de la légende en `"Qualité de la coupe"` ;
4. Indiquer la traduction française dans la légende, plutôt que le terme anglais ;
5. Changer les couleurs par défaut des points pour 5 couleurs de votre choix.

## 5.1 Annotations

À partir du graphique suivant :

```
p <- ggplot() + geom_point(data = diamonds[sample(1:nrow(diamonds), 1000),],
                           aes(x = carat, y = price, colour = cut))
```

1. Ajouter une ligne horizontale à $y = 10,000$, en pointillés gris ;
2. Ajouter le texte “Echantillon de taille 1000” à n’importe quel endroit du graphique.

## 5.2 Étiquettes (titre du graphique et des axes)

À partir du graphique suivant :

```
p <- ggplot() + geom_point(data = diamonds[sample(1:nrow(diamonds), 1000),],
                           aes(x = carat, y = price, colour = cut))
```

1. Ajouter un titre cohérent ;
2. Changer les étiquettes des axes pour y mettre des étiquettes en français.

## 5.3 Modification des éléments du thème

À partir du graphique suivant :

```
p <- ggplot() + geom_point(data = diamonds[sample(1:nrow(diamonds), 1000),],
                           aes(x = carat, y = price, colour = cut))
```

1. Mettre le fond de la zone de graphique en blanc ;
2. Changer la taille des étiquettes des axes ;
3. Changer la couleur des lignes secondaires de la grille de la zone de graphique.