

Statistique descriptive avec R

TP 1

R possède des jeux de données intégrés. Nous allons travailler sur les données **iris**, une **data.frame** déjà présente dans R. On rappelle que pour accéder à l'aide d'une fonction, on peut utiliser **help(nom_fonction)**.

La fonction **str** appliquée à une **data.frame** permet de voir les noms des colonnes, leurs types, et quelques valeurs.

```
str(iris)
```

On peut avoir une représentation rapide des minimums, maximums, moyennes et quartiles de chaque colonne avec **summary**.

```
summary(iris)
```

Pour explorer au-delà d'un affichage dans la console, on peut utiliser la fonction **View**.

```
View(iris)
```

La fonction **tapply** est similaire à l'opération *Group By* de SQL. Elle a la forme **tapply(x, y, f)** où **x** est un vecteur de données quantitatives, **y** est un vecteur de modalités de même taille, et **f** est une fonction d'agrégation ($f : \mathbb{R}^n \rightarrow \mathbb{R}$). Ici, **tapply** calcule la fonction **f** sur **x** pour chaque modalité de **y**.

Par exemple, nous pouvons calculer la longueur moyenne de **Sepal.Length** dans **iris** pour chaque espèce.

```
tapply(iris$Sepal.Length, iris$Species, mean)
```

1. La fonction **f**, qui est ci-dessus **mean**, peut aussi être programmée. Calculez la moyenne des valeurs en excluant le minimum et le maximum, c'est-à-dire :

$$f(x) = \frac{1}{n-2} \left(\sum_{i=1}^n x_i - \min(x) - \max(x) \right).$$

2. Pour obtenir la matrice de corrélation des données quantitatives de **iris**, on exclut la dernière colonne et on utilise la fonction **cor**. Affichez la matrice de corrélation des données quantitatives de **iris**.

3. La fonction **plot(x, y)** permet d'afficher **y** en fonction de **x**. Affichez **Petal.Length** en fonction de **Petal.Width**.

4. Changez de style de point en utilisant l'argument `pch=16`, et choisir une couleur *moderne* en piochant dans la palette <https://flatuicolors.com/palette/defo> (e.g. le bleu sombre).
5. Mettez une étiquette pour les abscisses : "**Largeur des pétales**" et pour les ordonnées : "**Longueur des pétales**" avec les arguments `xlab` et `ylab` de `plot`.
6. Pour faire une régression linéaire, on utilise la syntaxe `lm(y ~ x, data=mes_donnees)` où `mes_donnees` est la `data.frame` qui contient les données : `x` est le nom de la colonne de la variable explicative, `y` est le nom de la colonne de la variable à expliquer. Faites la régression linéaire de `Petal.Width` en fonction de `Petal.Length` et enregistrez-la dans une variable nommée `reg`.
7. Vérifiez que c'est une liste et obtenez le nom des éléments avec `names`.
8. Ajoutez la droite de régression avec `abline` avec une épaisseur de 2 (argument `lwd`) et une couleur sombre.
9. Ajoutez une légende avec `legend` en haut à gauche. Indiquer le type de point (argument `pch`) dans la légende affiche un point, et indiquer l'épaisseur du trait (argument `lwd`) affiche un trait. `NA` permet de dire qu'il n'y a rien. La fonction `legend` est vectorielle.
10. Dans le graphique, les largeurs et longueurs des pétales sont affichées pour toutes les espèces. On souhaite mettre une couleur différente en fonction de l'espèce. On choisit la couleur "`#16a085`" pour `setosa`, "`#c0392b`" pour `versicolor` et "`#8e44ad`" pour `virginica`. Construisez un vecteur `couleurs` de la même longueur que le nombre de lignes des données, avec la valeur constante "`#16a085`".
11. Pour les indices où l'espèce est `versicolor`, remplacez la valeur de `couleurs` par "`#c0392b`". Faites de même pour `virginica`.
12. Affichez le graphique complet avec les couleurs par espèce, en adaptant la légende.
13. Ajoutez un maillage avec la fonction `grid` et mettez une couleur de fond à votre légende.
14. Enregistrez votre graphique au format PDF grâce à la fonction `pdf` avec une largeur de 10 et une hauteur de 7. On place l'appel à la fonction `pdf` avant de générer le graphique, et une fois le graphique terminé, on conclut avec `dev.off()` qui ferme et enregistre le PDF.