[Support] Formation : Prise en main des packages Dplyr et GGplots

Plateforme universitaire de données de Bretagne

Paul Pinard

Extensions / Packages :

- Packages à lancer / installer avant de commencer la suite de l'atelier

```
# install.packages("*****")

library(datasets)

library(dplyr)

library(ggplot2)

library(tidyr)

library(DT)

library(catdata)

library(car)
```

Jeux de données jouet / d'entrainement :

- Jeu de données Mtcars & Starwars seront utilisés pour des exemples concrets

```
data(mtcars)
starwars <- as.data.frame(starwars)[,1:11] %>%
                  mutate(gender = as.factor(gender)) %>%
                  mutate(across(where(is.character), as.factor))
```

- Pour la pratique, utilisation de « data_entrainement »
    - Jeu entièrement fictif
    - Réalisé à partir de bases existantes

```
data_entrainement <- read_excel("data_entrainement.xlsx") %>%
                                  mutate_if(is.character, as.factor)
```

Dplyr & la gestion de sa base de données :

- Pourquoi ?
  - Changer des valeurs dans une / plusieurs colonnes
  - Créer de nouvelles variables
  - Créer des résumés
  - Filtrer notre base
  - Etc…
- Comment ?
  - Nécessaire un jeu de données
  - Ajout d'un « **PIPE** » (%>%) à la fin de notre ligne
    - *Exception pour la dernière ligne où il ne doit rien avoir (cf. code importation des données !)*
- Quels types de commandes ?
  - Mutate
  - Filter
  - Select
  - Group_by
    - Souvent utiliser en lien avec summarise ou mutate
  - Left / Right / inner / Full_join
  - Arrange

➢ Fonctionnement pour la suite :

❖ Un exemple de question réponse SANS (première ligne) et AVEC (seconde ligne) dplyr

❖ Une nouvelle question à traiter avec le fichier data_entrainement

❖ Question à traiter UNIQUEMENT avec Dplyr

✓ ***Exemple 1*** *: Filtrer uniquement les personnages qui possèdent des yeux marrons et une couleur de peau claire.*

```
starwars[starwars$skin_color == "light" & starwars$eye_color == "brown"
, ]

starwars %>% filter(skin_color == "light", eye_color == "brown")
```

✓ ***Exercice 1*** *: Filtrer uniquement les individus qui font plus d'1m70 et qui habitent à la campagne*

```
data_entrainement %>% filter(Taille > 1.70, Lieu_de_Vie == "Campagne")
```

✓ ***Exemple 2*** *: Ordonner par taille (du plus grand au plus petit) puis, par la suite, par la masse.*

```
starwars[order(-starwars$height, starwars$mass), ]

starwars %>% arrange(desc(height), mass)
```

✓ ***Exercice 2 :*** *Ordonner par la marque de voiture, puis par la situation d'emploi (décroissant)*

```
data_entrainement %>% arrange(Marque_Voiture, desc(Situation_Emploi))
```

✓ ***Exemple 3*** *: Garder uniquement les colonnes « name », « height », « sex » et « skin_color »*

```
starwars[,c("name","height","sex","skin_color")]

starwars %>% dplyr::select(name,height,sex,skin_color)
```

✓ ***Exercice 3 :*** *Garder les colonnes correspondant à l'âge, le lieu de vie, le poids ainsi que la nationalité*

```
data_entrainement %>% dplyr::select(Age, Lieu_de_Vie,Poids,Nationalite)
```

✓ ***Exemple 4*** *: Renommer les colonnes actuellement en Anglais en Français*

```
colnames(starwars)[colnames(starwars) == "name"] <- "NOM_PERSONAGE"

data("starwars")
starwars %>% rename(NOM_PERSONAGE = name)
```

✓ ***Exercice 4 :*** *Mettre en minuscule le nom de 3 colonnes au choix*

✓ ***Bonus :*** *Passer en minuscule l'entièreté des colonnes (demander si problème)*

```
data_entrainement %>% rename(sexe = Sexe, taille = Taille, lieu_de_vie
                             = Lieu_de_Vie)

data_entrainement %>% rename_all(tolower)
```

✓ ***Exemple 5*** *: Créer 2 nouvelles variables Taille (en mètre) & IMC*

```
starwars$hight_m <- starwars$height/100
starwars$IMC <- starwars$mass / (starwars$hight_m^2)

starwars %>% mutate(height_m = height / 100,
                    IMC = mass / (height_m^2))
```

✓ ***Exercice 5 :***

- *Passer la variable « Taille » des mètres en Feet* ***SANS*** créer une nouvelle variable
- *Pour la variable Poids, la passer des Kg en Lbs* ***EN*** *créant une nouvelle variable*
- *Conversion (1m = 3.28 ft / 1kg = 2.20 Lbs)*

✓ ***Bonus :*** *Arrondir les calculs à 2 valeurs après la virgule (Indice : fonction round())*

```
data_entrainement %>% mutate(Taille = Taille * 3.28,
                             poids_lbs = Poids * 2.20)

data_entrainement %>% mutate(Taille = round(Taille * 3.28,2),
                             poids_lbs = round(Poids * 2.20,2))
```

✓ ***Exemple 6*** *: Calculer la moyenne de la variable « Taille » en fonction du Sexe de l'individu*

```
starwars_filtered <- starwars[complete.cases(starwars$height), ]
moyenne_taille_sexe <- tapply(starwars_filtered$height,
starwars_filtered$sex, mean)

starwars %>%
  group_by(sex) %>%
  summarise(Moyenne_taille_sexe = mean(height, na.rm = TRUE))
```

✓ ***Exercice 6 :*** *Calculer la moyenne et l'écart-type de l'âge en fonction de la situation familiale ET le lieu de vie (Indice : fonction écart-type = sd())*

```
data_entrainement %>%
  group_by(Situation_Familiale, Lieu_de_Vie) %>%
  summarise(Moy_Age = mean(Age, na.rm = TRUE),
            Ecart_Type_Age = sd(Age, na.rm = TRUE))
```

✓ ***Exercice Bonus 1 :*** *Changer l'ensemble des colonnes de type « Character » en « factor »*

```
data_entrainement %>% mutate_if(is.character, as.factor)
```

✓ ***Exercice Bonus 2 :*** *Trier les marques de voitures par ordre décroissant en fonction du Sexe et de la variable « Propriétaire »*

```
data_entrainement %>%
  group_by(Sexe, Proprietaire) %>%
  arrange(desc(Marque_Voiture), .by_group = TRUE)
```

✓ ***Exercice Bonus 3 :***

- *Obtenir le nombre de situations d'emplois totales pour l'ensemble des pays*
- *Comparer ce total en fonction de la nationalité*
- *Mettre ce chiffre final en pourcentage*

```
data_entrainement %>%
  group_by(Situation_Emploi) %>%
  summarise(Total_Sit_Emploi = n()) %>%
  left_join(data_entrainement, by = c("Situation_Emploi")) %>%
  group_by(Situation_Emploi, Nationalite) %>%
  summarise(count = n())%>%
  mutate(Pourcentage = count / total_situtation_emploi,
         Pourcentage = round(Pourcentage,2))
```

OU (sans jointure)

```
data_entrainement %>%
  group_by(Nationalite) %>%
  mutate(total_Nationalite = n()) %>%
  group_by(Nationalite, Situation_Emploi,total_Nationalite) %>%
  mutate(Total_Nat_Sit = n()) %>%
  mutate(Pourcentage = Total_Nat_Sit / total_Nationalite * 100,
         Pourcentage = round(Pourcentage,2)) %>%
  dplyr::select(Nationalite,Situation_Emploi,
            Total_Nat_Sit,total_Nationalite,Pourcentage) %>%
  unique() %>%
  arrange(Nationalite)
```

- Pour toujours accentuer la comparaison entre R Base et Dplyr, les deux morceaux de code suivant sont des exemples l'un AVEC et l'autre SANS dplyr pour montrer des utilisation pratiques complètes. Ils ont la même finalité. Chronologiquement, le code va :
  - *Suppression des lignes avec un NA dans la colonne « Height »*
  - *Sélection de certaines variables que l'on garde*
  - *Création d'une nouvelle variable en fonction d'une autre déjà existante (**ifelse**)*
  - *Récupération d'une moyenne de taille et de poids en fonction de l'espèce*
  - *Jointure entre notre table originale et celle des moyennes*
  - *Réarrangement des colonnes dans un autre sens*
  - *Changement des noms de colonnes (**uniquement pour le code sans dplyr**)*

```
# Sans Dplyr
starwars_sans_na <- subset(starwars, !is.na(height))
starwars_sans_na_select <- subset(starwars_sans_na_select, select = c(name, h
eight, mass, species))
starwars_sans_na_select$height_category <- ifelse(starwars_sans_na_select$hei
ght < 170, "Short", ifelse(starwars_sans_na_select$height < 180, "Medium", "T
all"))

Especes_resume <- aggregate(cbind(height, mass) ~ species, data = starwars_sa
ns_na_select, FUN = function(x) mean(x, na.rm = TRUE))

starwars_final <- merge(starwars_filtered, species_summary, by = "species")
starwars_final <- starwars_final[,c("species", "name", "height.x", "mass.x",
"height.y", "mass.y")]

colnames(starwars_final) <- c("species", "name", "height ", "mass", "Moyenne_
Espece_Taille", "Moyenne_Espece_Poids"))
```

```
# Avec Dplyr
starwars %>%
  filter(!is.na(height)) %>%
  dplyr::select(name, height, mass, species) %>%
  mutate(height_category = ifelse(height < 170, "Short",

                                  ifelse(height < 180, "Medium", "Tall"))) %>%
  group_by(species) %>%
```

```
  summarise(mean_height = mean(height, na.rm = TRUE),
            mean_mass = mean(mass, na.rm = TRUE)) %>%
  left_join(starwars, by = "species") %>%
  dplyr::select(rank, species, name, height, mass, mean_height, mean_mass)
```

GGplot et la création de graphiques :

- Avantages :
  - Graphiques très modulables
  - Possibilité de mettre un nombre de variables « illimitées »
  - Possibilité de mettre plusieurs types de graphiques en un unique
    - *Mettre un histogramme et une courbe ou plusieurs histogrammes*
- Restrictions :
  - Besoin obligatoire de deux variables
    - *Histogramme hist() de R de base peut créer un graphique avec une seule variable en entrée*
  - Peu compatible avec des titres de graphiques avec de nombreux caractères
    - *Obligé de passer par un léger détour dans ce cas*

➢ Procéder pour la suite :
- ❖ Création d'un graphique simple que l'on étoffera par la suite
- ❖ De la même manière que le « **Pipe** », besoin de rajouter un « **+** » à la fin de chaque ligne sauf la dernière
- ❖ Un exemple dans un premier temps, puis un exercice
- ❖ Le graphique montre ce que produit le code « exemple »

- Créer un graphique :
  - Pour l'exemple :
    - En abscisse, le nombre de cylindrées
    - En ordonnée, le nombre de chevaux
    - Représentation par point
  - En utilisant data_entrainement :
    - En abscisse, la taille
    - En ordonnée, le poids
    - Représentation par point
    - L'enregistrer dans un objet « Graph_Appr »

```
ggplot(data = mtcars, aes(x = cyl,y = hp)) +
  geom_point()

ggplot(data = mtcars) +
  geom_point(aes(x = cyl,y = hp))

ggplot() +
  geom_point(data = mtcars, aes(x = cyl,y = hp))
```

Correction:

```
Graph_Appr <- ggplot(data = data_entrainement) +
  geom_point(aes(x = Taille,y = Poids))
```

- Rajouter sur ce même graphique :
  - Pour l'exemple :
    - De la couleur en fonction du nombre de carburateur pour chaque voiture
  - En utilisant data_entrainement :
    - De la couleur pour le sexe des individus
    - Des formes différentes en fonction du nombre d'enfants à charge
      - ✓ *Indice : le paramètre à utiliser est 'shape'*

Nb : Que cela soit pour rajouter de la couleur ou bien des formes, bien mettre la variable souhaitée en facteur !

```
ggplot(data = mtcars) +
  geom_point(aes(x = cyl,y = hp, color = as.factor(carb)))
```

Correction:

```
Graph_Appr <- ggplot(data = data_entrainement) +
               geom_point(aes(x = Taille,y = Poids, color = as.factor(Sexe),
shape = as.factor(Nombre_Enfants)))
```

- Personnaliser le graphique :
  - Pour l'exemple :
    - Changer le titre du graphique, le nom de l'axe X et Y ainsi que le nom de la légende
    - Changer les couleurs mises automatiquement par des couleurs manuelles (bleue, cyan, magenta, orange, marron, et noir)
    - Rajouter un thème (fond du graphique) au choix
  - En utilisant data_entrainement :
    - Changer le nom des DEUX légendes
    - Rajouter un thème au choix
    - Changer manuellement les couleurs ET les formes
      - ✓ *Pour les formes, indice* : `values = c(3,4,15,17,19)`
      - ✓ *https://www.sthda.com/english/wiki/ggplot2-point-shapes*

```
ggplot(data = mtcars) +
  geom_point(aes(x = cyl,y = hp, color = as.factor(carb))) +
  theme_bw() +
  labs(title = "Nombre de cheveaux en fonction Cylindré / Nb Carburateurs",
       x = "Nombre de cylindré",
       y = "Nombre de cheveaux",
       color = "Nombre de carburateurs")+
  scale_color_manual(values = c("blue", "cyan", "magenta", "orange", "brown",
"black"))
```

Correction:

```
ggplot(data = data_entrainement) +
               geom_point(aes(x = Taille,y = Poids, color = as.factor(Sexe),
shape = as.factor(Nombre_Enfants))) +
  labs(color = " Sexe des Individus", shape = "Nombre d'enfants à charge")
  scale_color_manual(values = c("blue", "brown")) +
  scale_shape_manual(values = c(3,4,15,17,19))
```

OU

```
Graph_Appr +
  labs(color = " Sexe des Individus", shape = "Nombre d'enfants à charge")
```

```
  scale_color_manual(values = c("blue", "brown")) +
  scale_shape_manual(values = c(3,4,15,17,19))
```

- Touches Finales :
  - Pour l'exemple :
    - Utiliser l'argument theme() pour :
      - ✓ Changer la position de la légende en bas du graphique, augmenter la taille de sa police, la mettre en caractère gras
      - ✓ Changer les titres des axes en italique avec une taille 14
      - ✓ Mettre le titre principal en couleur rouge et en gras
      - ✓ Mettre le titre de la légende en gras italique et de couleur verte
  - En utilisant data_entrainement :
    - Place à l'imagination !
    - Pour plus d'exemples, voir plus bas sur le support (3 exemples complets)

```
ggplot(data = mtcars) +
  geom_point(aes(x = cyl,y = hp, color = as.factor(carb))) +
  theme_bw() +
  labs(title = "Nombre de cheveaux en fonction Cylindré / Nb Carburateurs",
       x = "Nombre de cylindré",
       y = "Nombre de cheveaux",
       color = "Nombre de carburateurs")+
  scale_color_manual(values = c("blue", "cyan", "magenta", "orange", "brown",
"black")) +
  theme(legend.text = element_text(size = 8, face = "bold"),
        legend.position = "bottom",
        axis.title = element_text(face = "italic", size = 14),
        plot.title = element_text(face = "bold", color = "red"),
        legend.title = element_text(face = "bold.italic", color = "green"))
```

Correction:

```
ggplot(data = data_entrainement) +
               geom_point(aes(x = Taille,y = Poids, color = as.factor(Sexe),
shape = as.factor(Nombre_Enfants))) +
  labs(color = " Sexe des Individus", shape = "Nombre d'enfants à charge")
  scale_color_manual(values = c("blue", "brown")) +
  scale_shape_manual(values = c(3,4,15,17,19)) +
```

```
  theme(legend.text = element_text(size = 8, face = "italic"),
        legend.position = "bottom",
        legend.box.background = element_rect(color = "red"),
        legend.box.margin = ggplot2::margin(t = 1, l = 1),
        axis.title = element_text(face = "bold", size = 11),
        legend.title = element_text(face = "bold.italic", family = "serif"))
```

- Quelques exemples un peu plus complets :
  - Premier graphique :
    - Résume les informations de 3 colonnes
    - Boxplots des poids d'individus sur 3 périodes de temps différents et avec 6 groupes par temps
  - Second Graphique :
    - Résume les informations de 5 colonnes
    - Pour chaque département de Bretagne, graphique des revenus gagnés par des individus fictifs en fonction de leurs âges, de leurs sexes et de leurs éducations
  - Troisième Graphique :
    - Plus utile pour des graphiques avec plusieurs Y dans de même valeurs de X (par exemple la comparaison du salaire entre homme et femme en fonction de leurs âges OU la quantité de pluie tombée dans 3-4 villes différentes sur plusieurs années)
    - Données n'ont pas d'applications fictives comme les deux derniers graphiques

Nb : Le data frame qui est utilisé pour chaque graphique est copiable dans ce document.

Nb 2 : Besoin d'installer un package supplémentaire pour faire fonctionner le graphique X

```
windowsFonts()
## [1] "TT Times New Roman"
## [1] "TT Arial"
## [1] "TT Courier New"

install.packages("extrafont")
```

```
library(extrafont)

font_import()

loadfonts(device = "win")
```

Data Frame Graphique 1 :

```
crea_df <- data.frame(Periode = rep(c("0-10", "11-20", "21-31"), each = 12),
                      Groupe = rep(1:6, each = 2, times = 3),
                      Poids_Pers = round(runif(36, min = 60, max = 88), 2),
                      Poids_Pers = round(runif(36, min = 175, max = 194), 2))
%>%
                                    mutate(Periode = as.factor(Periode)) %>%
                                     mutate(Groupe = as.factor(Groupe))
```

Data Frame Graphique 2 :

```
set.seed(1234)
n <- 150

age <- sample(18:80, n, replace = TRUE)
revenu <- rnorm(n, mean = 50000, sd = 20000)
genre <- sample(c("Homme", "Femme"), n, replace = TRUE, prob = c(0.5,0.5))
education <- sample(c("Lycée", "License", "Master", "PhD"), n, replace =
TRUE)
Dep <- sample(c("Finistère", "Côte D'Armor", "Ille et Vilaine", "West"), n,
replace = TRUE)

data_plus <- data.frame(Age = age, Revenus = revenu, Genre = genre, Education
= education, Dep = Dep)
```

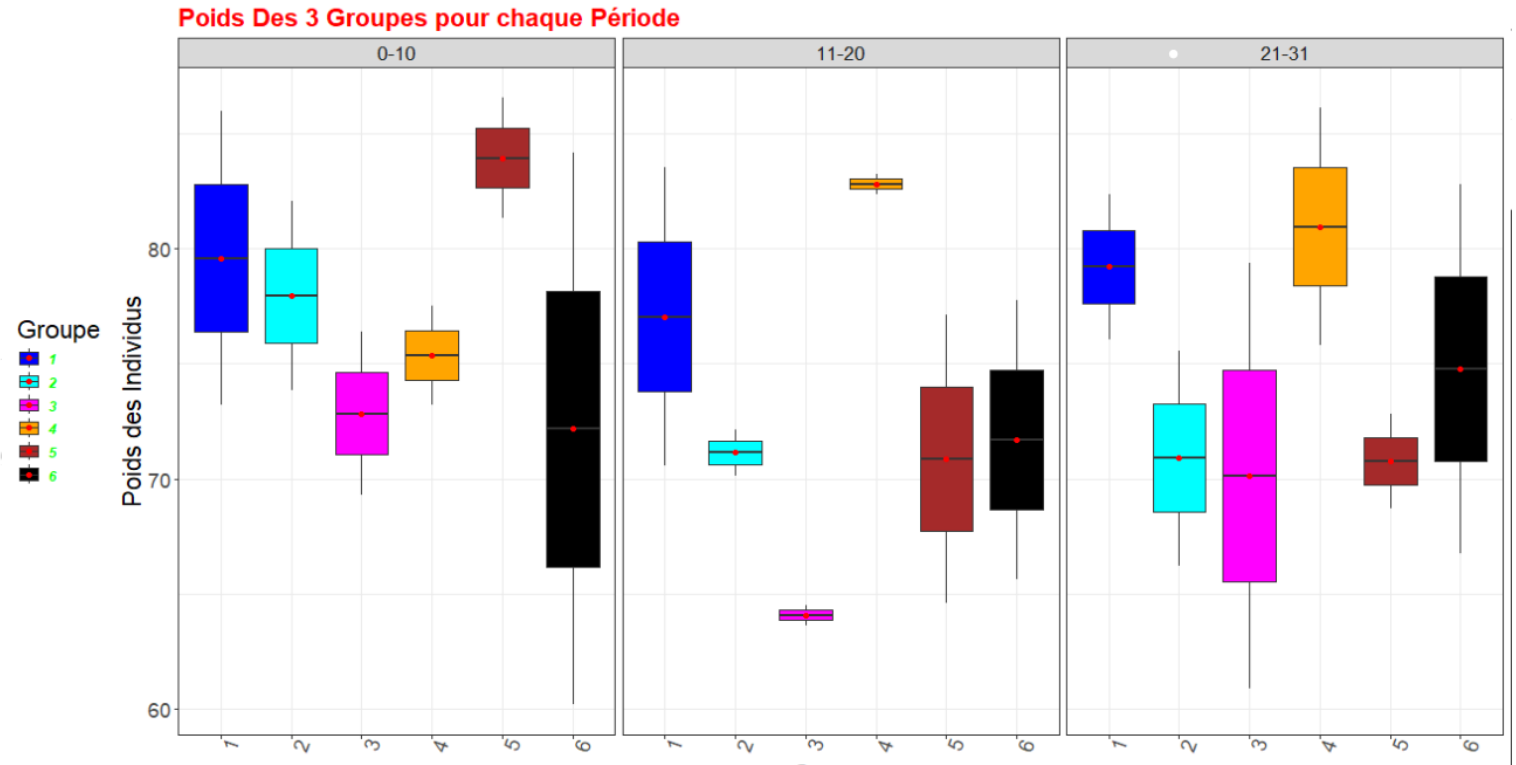
Data Frame Graphique 3 :

```
data <- data.frame(x = 1:20,
                   y1 = rnorm(20, 0, 1),
```

```
                    y2 = rnorm(20, 0, 1),
                    y3 = rnorm(20, 0, 1),
                    y4 = rnorm(20, 0, 1))
```

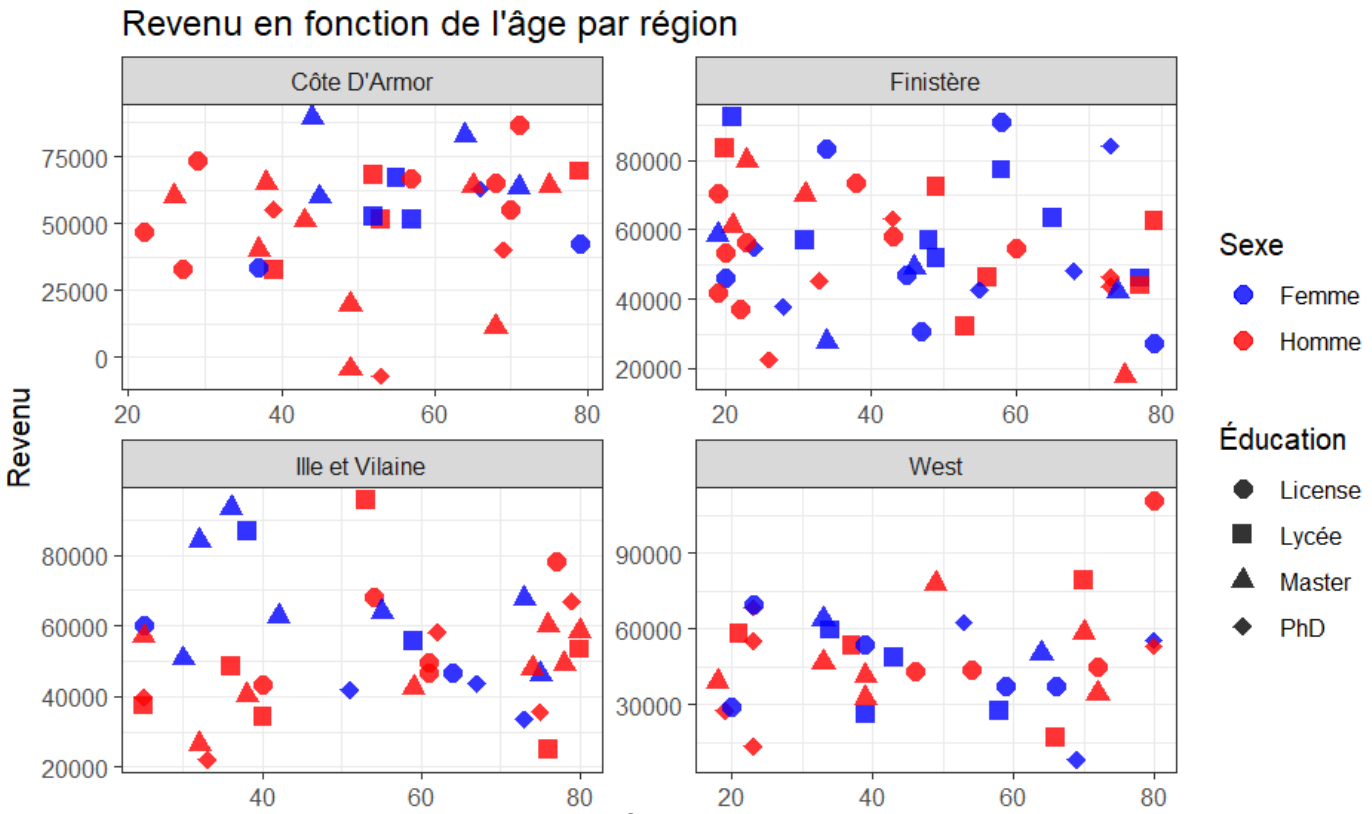
Graphique n°1:

```
ggplot(data = crea_df,aes(x=Groupe, y = Poids_Pers, fill=Groupe))+
          theme_bw()+
          geom_boxplot()+
          ggtitle("Poids Des 3 Groupes pour chaque Période")+
          stat_summary(geom = "point",col="red",fun = mean)+
          scale_fill_manual(values = c("blue", "cyan", "magenta", "orange", "
brown", "black"))+
          facet_wrap("Periode")+
          labs(x= "Groupes", y= "Poids des Individus")+
          theme(axis.text.x=element_text(angle=70),
                axis.text = element_text(size = 15),
                legend.text = element_text(size = 10, face = "bold.italic", c
olor = "green"),
                legend.position = "left",
                title = element_text(size = 18),
                plot.title = element_text(size = 18, face = "bold", color = "
red"),
                strip.text = element_text(size=14))
```



Graphique n°2:

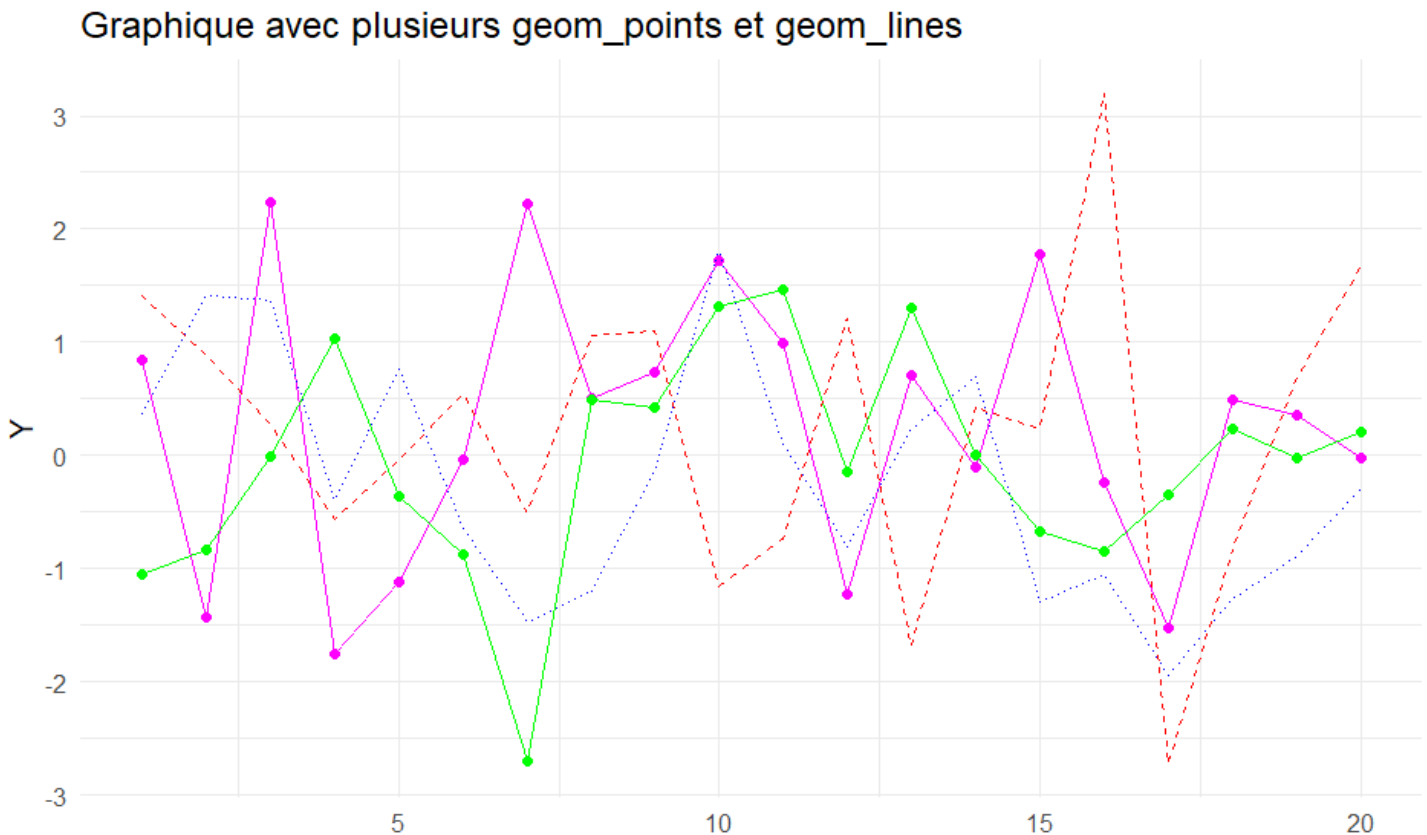
```
ggplot(data = data_plus, aes(x = Age, y = Revenus, color = Genre, shape = Education)) +
  geom_point(size = 3, alpha = 0.8) +
  facet_wrap(~Dep, scales = "free") +
  theme_bw() +
  scale_colour_manual(values = c("blue","red"))+
  scale_shape_manual(values = c(16,15,17,18))+
  labs(title = "Revenu en fonction de l'âge par région",
       x = "Âge",
       y = "Revenu",
       color = "Sexe",
       shape = "Éducation")
```



Graphique n°3:

Version "Simple":

```
ggplot(data, aes(x = x)) +
  geom_point(aes(y = y3), color = "magenta") +
  geom_line(aes(y = y3), color = "magenta") +
  geom_line(aes(y = y1), color = "blue", linetype = "dotted") +
  geom_point(aes(y = y4), color = "green") +
  geom_line(aes(y = y4), color = "green") +
  geom_line(aes(y = y2), color = "red", linetype = "dashed") +
  labs(title = "Graphique avec plusieurs geom_points et geom_lines",
       x = "X",
       y = "Y") +
  theme_minimal()
```



Version "Complexe":

```
data_long <- data %>%
  pivot_longer(cols = start_with("y"), names_to = "Variable", values_to = "Va
lue")


ggplot(data, aes(x = x, y = Value, color = Variable, linetype = Variable)) +
  geom_point()+
  geom_line() +
  labs(title = "Graphique avec légende pour chaque colonne",
       x = "X",
       y = "Y",
       color = "Variable",
       linetype = "Variable") +
  theme_minimal()
```