TP1 - Logiciel statistique R

Maty NDIONE

2024-04-08

Table des matières

1. Préparation des données

1.1. Description

Voir énoncé

1.2. Importation et mise en forme

Le point de départ de toute étude statique est la disposition de données. La donnée représente la matière première du statisticien et son importation est une étape indispensable pour une bonne analyse. Pour ce faire, nous utiliserons la fonction **readxlsx()** de la librairie **readxl**[1] que nous importons au préalable.

```
library(readxl)
projet = read_xlsx("..//Donnees//Base_Projet.xlsx")
```

Nous vérifions aisément que l'objet *projet* est de type *data.frame* grâce à la fonction **class()**

```
class(projet)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

Après avoir importé la base de données sur laquelle nous allons travailler, nous pouvons procéder à l'affichage de quelques inforations synthétiques afin de vérifier l'intégrité de l'importation. Pour ce faire, nous utilisons la fonction **str()**

```
str(projet)
```

La fonction précédente affiche également la dimension du dataframe. Mais il est également possible d'affiche uniquement les dimensions à l'aide de la fonction **dim()**. Elle affche 2 valeurs comme résultat: la première représente le nombre de lignes (**nombre de PME: 255**) et la seconde le nombre de colonnes (**variables: 33**)

```
dim(projet)
```

```
## [1] 250  33
```

Après avoir consulter globalement la base de données, nous pouvons procéder au traitement. Pour cela, il est primordial de checker les valeurs manquantes plus particulièrement pour la variable *key* qui représente l'identifiant de chque PME dans la base. La fonction utilisée pour détecter les valeurs manquantes est **is.na()** qui reçoit en argument la variable key. Elle donne

[1] Pour en savoir plus: [https://readxl.tidyverse.org/]

comme sortie un booléen : TRUE si la valeur est manquante et FALSE sinon. En aplliquant uen opération somme(**sum()**), nous obtenons le nombre exact de valeurs manquantes pour ladite variable.

```
sum(is.na(projet$key))
```

```
## [1] 0
```

Le résultat montre que la variable *key* ne contient pas de valeus manquantes. Nous nous permettrons dans cette partie d'ajouter les descriptions des variables pour une meilleure compréhension du dataframe. Pour cela, nous utilisons la fonction **apply_labels()** de la librairie **expss**

```
projet = projet|>expss::apply_labels(
  q1 = "Région",
  q2 = "Département",
  q23 = "Sexe du dirigeant/responsable de la PME",
  q24 = "Age du dirigeant/responsable de la PME",
  q24a_1 = "français", q24a_2 = "wolof",q24a_3 = "Diola",
  q24a_4 = "Serere", q24a_5 = "Peul",
  q24a_6 = "Mandingue", q24a_7 = "Balante",
  q24a_9 = "Bambara", q24a_10 = "Autres langues",
  q25 = "Niveau d'instruction",
  q26 = "Nombre d'années d'experience professionnelle dans l'entreprise",
  q12 = "Statut juridique",
  q14b = "Autorisation de fabrication et de mise en vente (FRA)",
  q16 = "L'entreprise est-elle désservie par une route bitumée ?",
  q17 = "Etat de la route bitumée",
  q19 = "Etat de la piste qui mène à l'entreprise",
  q20 = "Avez-vous des associés dans l'entreprise",filiere_1 = "arachide",
  filiere_2 = "anacarde", filiere_3 = "mangue", filiere_4 = "riz",
  q8 = "Activité principale", q81 = "Propriétaire ou locataire",
  gps_menlatitude = "Latitude", gps_menlongitude = "Longitude",
  start = "Date de début de l'enregistrement", today = "Date de l'enquête")
```

1.3. Création de variables

Nous allons dans un premier temps renommer les variables q1, q2 et q23 en utilisant la fonction **rename()**

```
library(dplyr)
projet = projet |> rename(region = q1,
                  departement = q2,
                  sexe = q23)
```

Dans la suite, nous créerons une nouvelle variable *sexe_2* qui dépend de la variable *sexe*. Ceci se réalise grâce aux fonctions **mutate()** et **case_when()**

```
projet = projet |> mutate(sexe_2 = case_when(sexe == "Femme" ~ 1,
                                             sexe != "Femme" ~ 0))
```

Nous pouvons recoder la variable à l'aide de la fonction **recode** de **dplyr** pour voir les correspondances de chaque valeur (0 ou 1).

```
as.factor(projet$sexe_2)
dplyr::recode_factor(projet$sexe_2,
                     '1' = "Femme", '0' = "Homme")
# table(projet$sexe_2)
```

Dans cette partie, nous allons créer un dataframe composé de variables de la base *projet*. Pour ce faire, nous utilisons la fonction **select()**. Tout d'abord, nous sélectionnons la colonne *key* et enseuite les colonnes concernant les langues parlées. Il est à noter que ces dernières commencent par q24a_, par conséquent nous utilisons la fonction **starts_with()** qui donne la possibilité de sélectionner l'ensemble des variables en même temps. Le dataframe obtenu est stocké dans l'objet *langues*.

```
langues = data.frame(projet |>
                       select("key"), projet |> select(starts_with("q24a_")))
```

Pour créer la variable *parle* qui égale au nombre de langues parlées, il suffit de faire une opération somme(**rowSums()**) sur toutes les variables langues ligne par ligne. En effet, ces dernières prennent 1 si le dirigeant ou propriétaire de la PME parle la langue et 0 sinon. Ainsi, faire la somme de toutes ces variables revient à compter le nombre de langues parlées par le dirigeant ou propriétaire. Nous associerons la fonction **rowSums()** à la fonction **mutate()** (pour la création de variable).

```
langues = langues |> mutate(parle =
                                rowSums(select(langues, starts_with("q24a_"))))
```

Nous allons reconstruire la base de sorte à garder deux variables: *key* et *parle*.

```
langues = langues |> select("key","parle")

# Exportation de la base langues
expss::xl_write_file(langues, "..//Sorties//langues.xlsx")
```

Pour merger les dataframe *projet* et *langues*, nous allons faire un INNER JOIN. Comme tous les *key* sont les mêmes dans les deux datframe, faire un INNER JOIN va permettre de faire correspondre à chaque PME les variables initiales et le nombre de langues parlées.

```
df = inner_join(x=projet,y=langues, join_by(x$key==y$key))

expss::xl_write_file(df, "..//Sorties//nv_projet.xlsx")
```

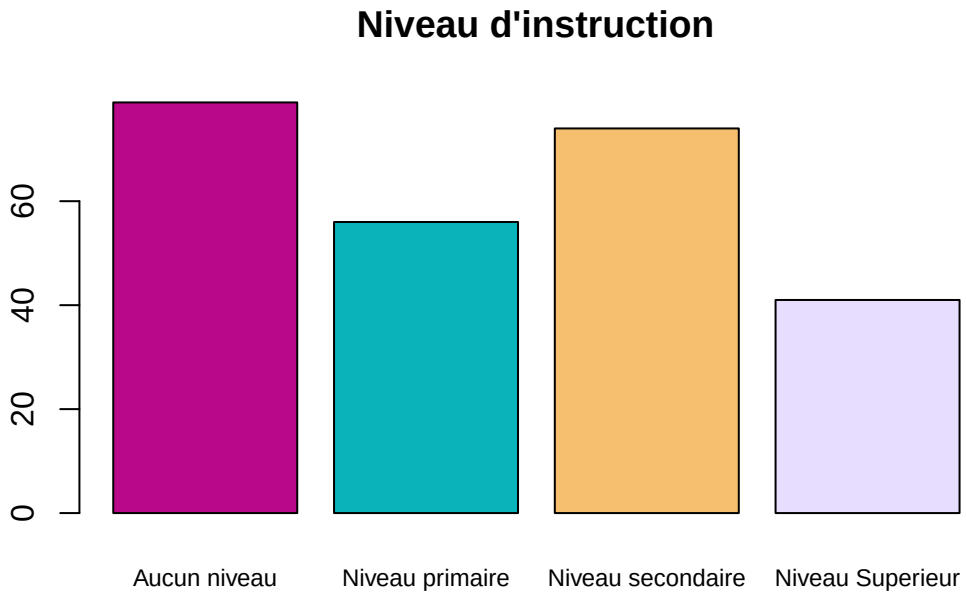
2. Analyse descriptive

La statistique descriptive constitue une étape fondamentale dans l'analyse de données. Elle permet de résumer et de présenter les caractéristiques principales d'un jeu de données. À l'aide d'outils statistiques simples, tels que les mesures de tendance centrale, de dispersion et de forme, la statistique descriptive offre un aperçu initial des données, facilitant ainsi leur interprétation et leur compréhension. Dans RStudio, un environnement de développement intégré (IDE) largement utilisé pour l'analyse de données en langage R, la statistique descriptive peut être réalisée de manière efficace à l'aide de différentes fonctions et packages disponibles. En ce qui nous concerne, nous utiliserons spécifiquement les packages graphics[2] et ggplot2 pour les graphiques et les packages gtsummry et GGally pour les tableaux.

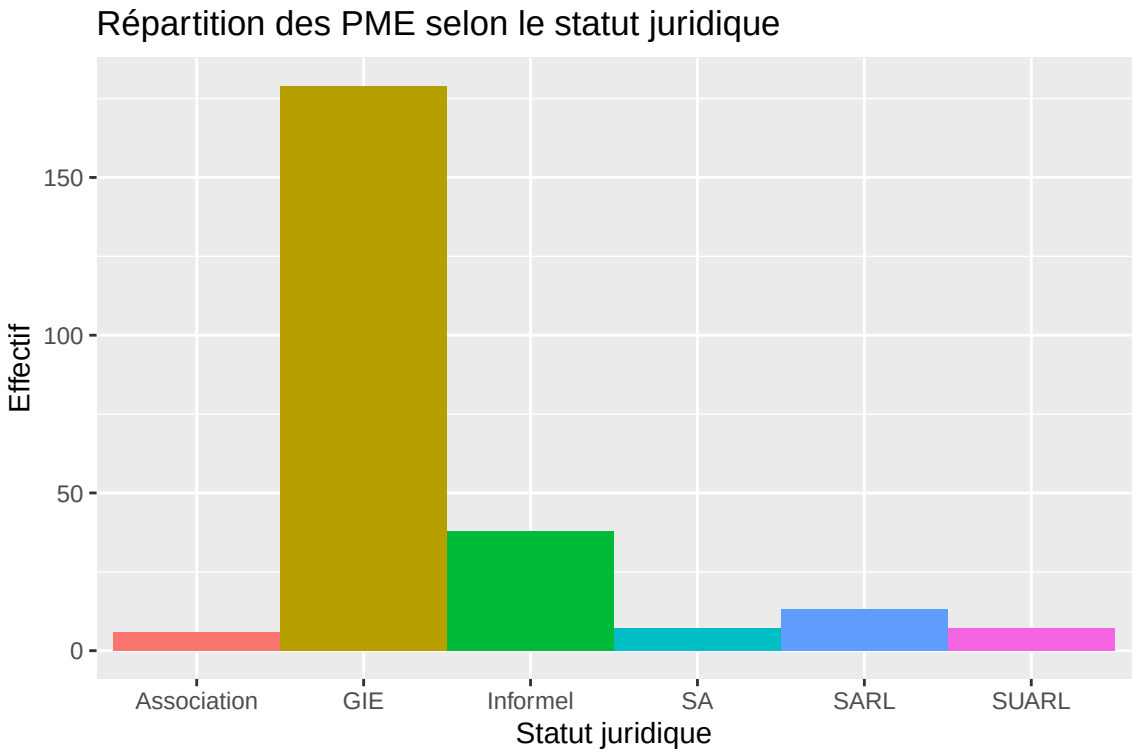
Pour créer des diagrammes en barres, la fonction **barplot()** peut être utilisée. Les arguments utilisés sont en grande partie similaires à ceux de la fonction pie.

```
plot3 = barplot(table(df$q25), col=c("#BA088A","#08B4BA","#F6BF6D","#E7DDFF"),
                main="Niveau d'instruction", cex.names = 0.75)
```

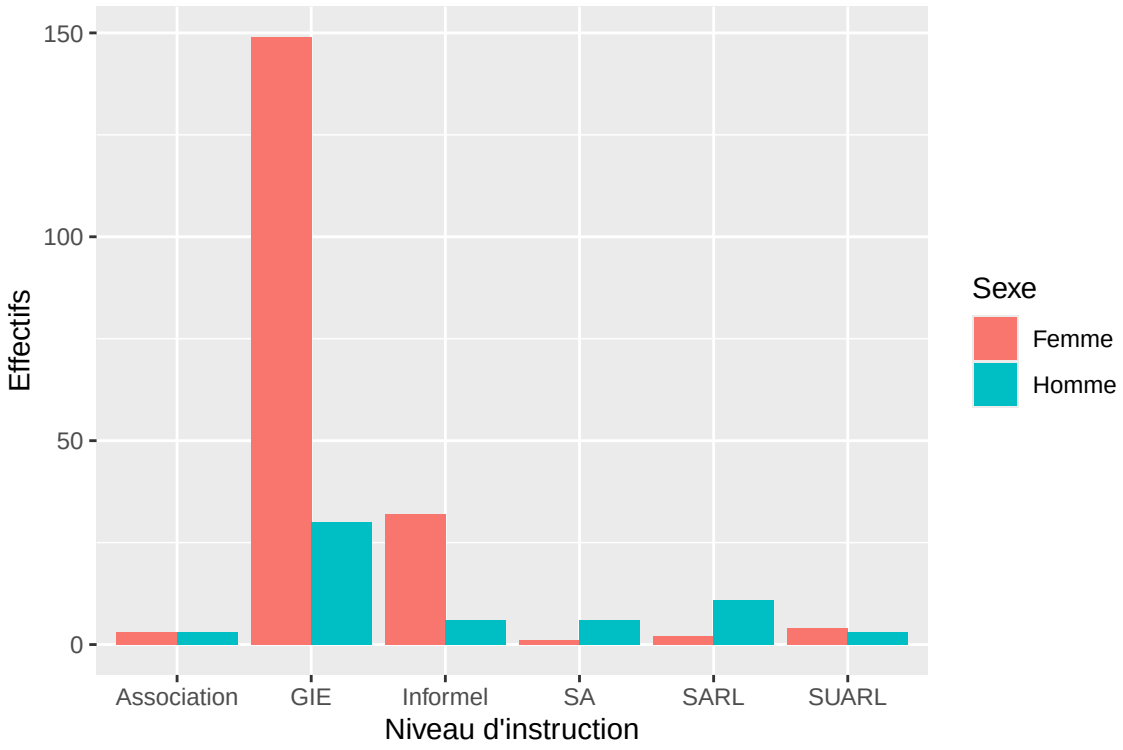
[2]Système graphique de base en R



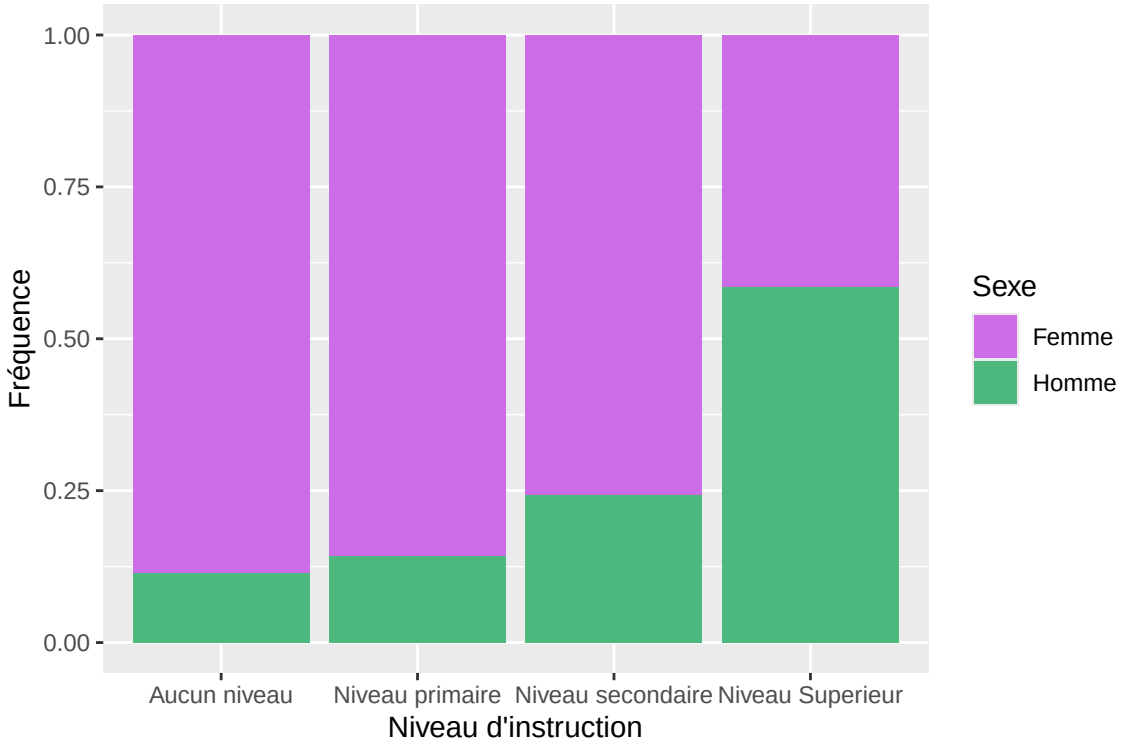

```
library(ggplot2)

plot4 = ggplot(data.frame(table(df$q12)),
               aes(x = Var1, y = Freq, fill = Var1)) +
  geom_bar(stat = "identity", width=1) + theme(legend.position = "none") +
  ggtitle("Répartition des PME selon le statut juridique") +
  xlab("Statut juridique") + ylab("Effectif")
plot4
```




```
plot6 = ggplot(data.frame(table(df$q12, df$sexe)),
             aes(x = Var1, y = Freq, fill = Var2)) +
        geom_bar(stat = "identity", position = "dodge") +
        xlab("Niveau d'instruction") + ylab("Effectifs") +
        labs(fill = "Sexe")
plot6
```

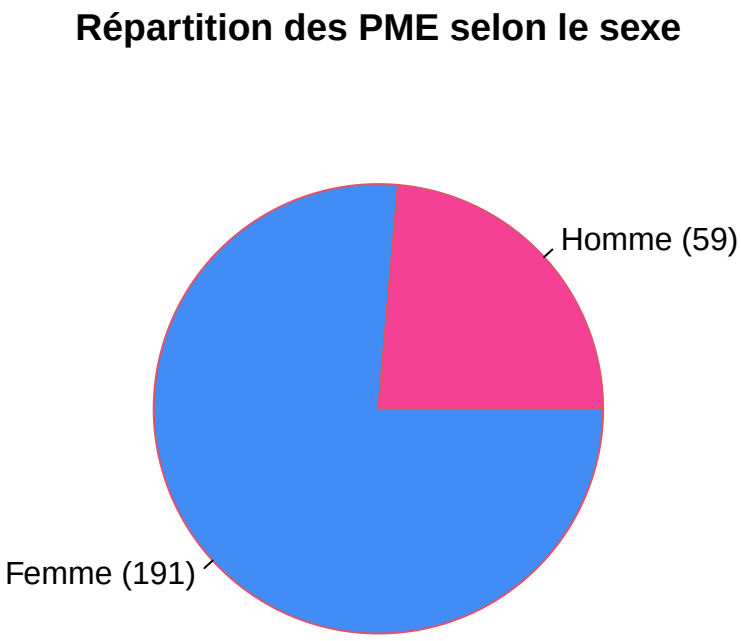



```
plot7 = ggplot(data.frame(proportions(table(df$q25, df$sexe))),
             aes(x = Var1, y = Freq, fill = Var2)) +
        geom_bar(stat = "identity", position = "fill") +
        xlab("Niveau d'instruction") + ylab("Fréquence") +
        labs(fill = "Sexe")  +
        scale_fill_manual(values = c("#CC6CE7","#4CB87D"))
plot7
```



Pour visualiser la distribution des variables qualitatives, une option est d'utiliser un diagramme circulaire à l'aide de la fonction **pie()**. Cette représentation graphique permet de voir la proportion de chaque catégorie de manière intuitive. de spécifier les étiquettes des modalités (labels), la couleur des parts de chaque modalité, ainsi que d'autres paramètres comme le titre du graphe, offrant ainsi une personnalisation complète de la visualisation.

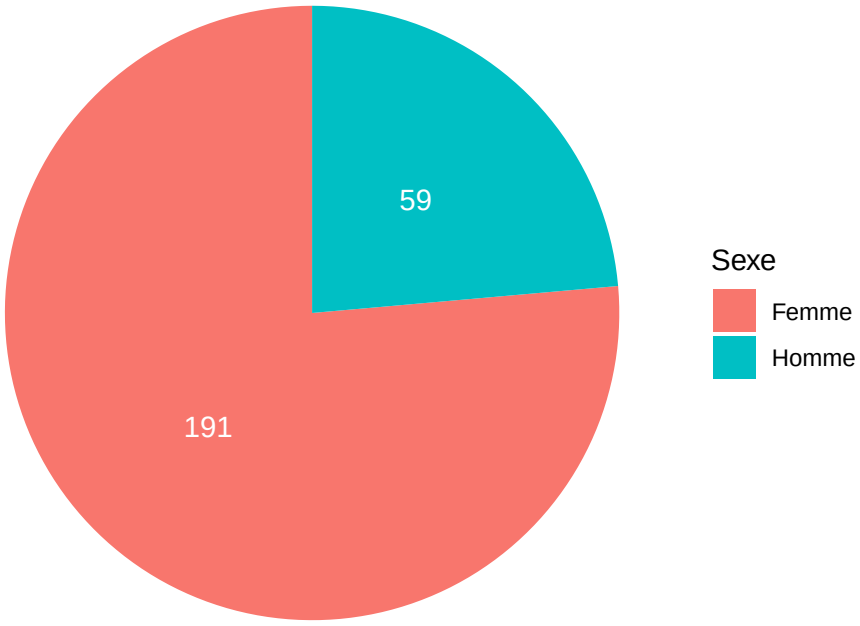
```
# Création du diagramme circulaire avec pie()
plot1 = pie(table(projet$sexe), border = 2,
            col =c("#418CF5", "#F54195"), init.angle = 0,
            angle = 90, clockwise = TRUE,
            labels = c(paste0(names(table(projet$sexe))[1]," (",table(projet$sexe)[1],")
                       paste0(names(table(projet$sexe))[2]," (",table(projet$sexe)[2],")")
            main = "Répartition des PME selon le sexe")
```



La transition vers ggplot2} permet d'exploiter des fonctionnalités plus avancées et esthétiquement plus attrayantes pour créer des graphiques visuellement percutants et personnalisables. Cette approche offre une affichage plus agréable et facilite la communication des résultats des analyses de données. Pour tracer les graphes sur ggplot2}, on utilise la fonction **ggplot()** à laquelle nous ajoutons des options selon notre besoin: - geom_bar : crée des barres dont la hauteur dépend de la variable spécifiée; - coord_polar : transforme le graphique en coordonnées polaires pour obtenir un diagramme circulaire; - geom_text : indique le texte ainsi que sa position dans le graphique.

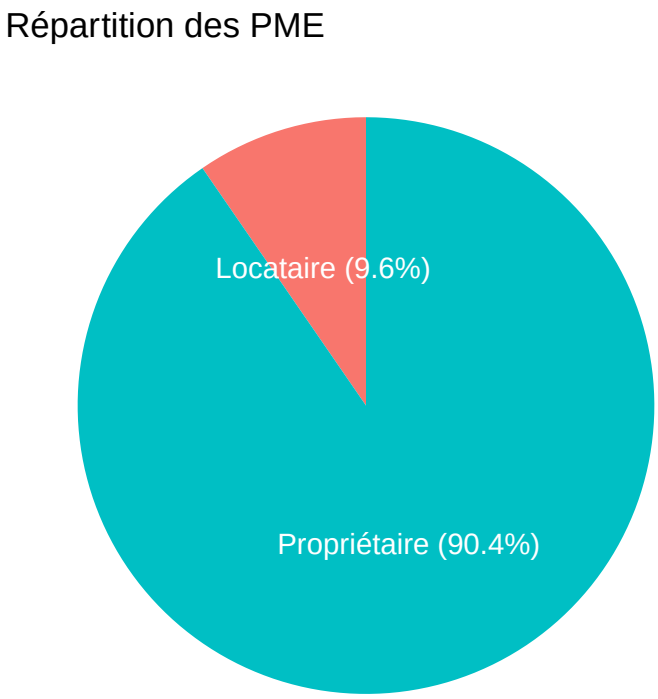
```
# Création du diagramme circulaire avec ggplot()
plot2 = ggplot(data.frame(table(df$sexe)),
               aes(x = "", y = Freq, fill = Var1)) +
  geom_bar(stat = "identity", width=1) +
  coord_polar(theta = "y", start = 0) +
  geom_text(aes(label = Freq),color="white",
            position = position_stack(vjust=0.5)) +
```

```
  labs(fill="Sexe") + theme_void()
plot2
```



Le résultat est un graphique clair et informatif qui illustre visuellement la répartition des sexes dans les données. Il met en évidence une proportion nettement plus élevée de femmes par rapport aux hommes au sein de l'ensemble des PME.

```
plot5 = ggplot(data.frame(round(proportions((table(df$q81)))*100, 2)),
                          aes(x = "", y = Freq, fill = Var1)) +
          geom_bar(stat="identity", width=1) + ggtitle("Répartition des PME") +
          coord_polar(theta = "y", start = 0) +
          geom_text(aes(label = paste0(Var1," (",Freq,"%)")),
                    color="white", position = position_stack(vjust=0.5)) +
          theme_void() + theme(legend.position = "none")
plot5
```



Quant à la création de tableaux résumant les statistiques descriptives, nous utilisons la bibliothèque gtsummary . Grâce à la fonction **tbl_summary()** , nous pouvons rapidement créer des tableaux récapitulatifs très esthétiques. Ces tableaux offrent une synthèse claire et concise des statistiques descriptives et génère des tableaux contenant des informations telles que les effectis, les proportions, etc. pour différentes variables.

```
library(gtsummary)
tabs = df |> tbl_summary(include = c("sexe","q25","q12","q81"))
tabs
```

Characteristic	**N = 250**
Sexe du dirigeant/responsable de la PME	
Femme	191 (76%)
Homme	59 (24%)
Niveau d'instruction	
Aucun niveau	79 (32%)
Niveau primaire	56 (22%)
Niveau secondaire	74 (30%)
Niveau Superieur	41 (16%)
Statut juridique	

Characteristic	**N = 250**
Association	6 (2.4%)
GIE	179 (72%)
Informel	38 (15%)
SA	7 (2.8%)
SARL	13 (5.2%)
SUARL	7 (2.8%)
Propriétaire ou locataire	
Locataire	24 (9.6%)
Propriétaire	226 (90%)

```
library(GGally)
plot8 = ggtable(data=df,
                columnsX = "q81",
                columnsY = "sexe",
                cells = "row.prop")
plot8
```

En utilisant les sorties graphiques et tabulaires ci-dessus, nous pouvons automatiser le

processus d'analyse descriptive dans une base de données en créant deux fonctions :
- **univarie()** pour les analyses univariées : elle prend en argument la dataframe (data), la variable à analyser (var). Selon le type de cette dernière quanti ou quali, elle permet soit de représenter un graphique (uniquement pour les variables aualitatives) ou un tableau. Dans le cas où la variable renseignée est de type discrète, elle donne en sortie un tableau des statistiques de tendance centrale (moyenne, médiane, quartiles...). Lorque que la variable est qualitative, il est possible d'avoir des sorties gaphique ou tableau selon la demande de l'utilisateur. Si vous souhaitez avoir les deux sorties, il faut renseigner all=TRUE (valeur par défaut). Il est également possible de voir uniquement la sortie graphique avec l'option all="plot" ou la sortie tableau avec l'otion all="table" . Par défaut, le graphe de sortie est un camembert mais vous pouvez tout de même spécifier type="bar" pour afficher le diagramme en barres. - **bivarie()** pour les analyses bivariées : elle prend en argument le jeu de données (data) et les deux variables (var1 , var2). Il est à noter que cette fonction ne prend en charge que des variables qualitatives. Avec son option par défaut all=TRUE , elle affiche le tableau croisé ainsi que le diagramme des deux variables. Tout comme la fonction **univarie()**, elle peut afficher uniquement le graphique ou le tableau avec les mêmes options. Ces fonctions nous permettront de généraliser les statistiques descriptives en fonction du type de variable.

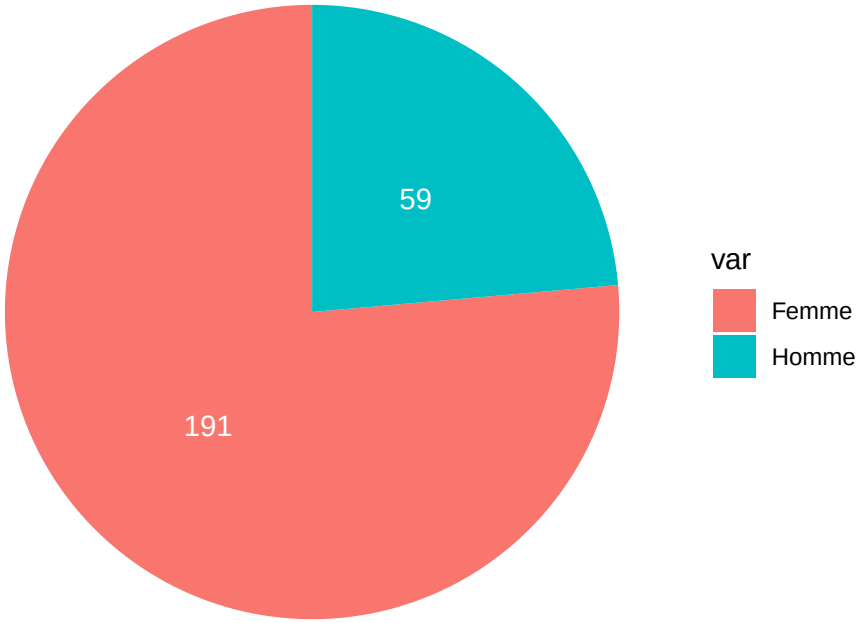
```
univarie <- function(data, var, all=TRUE, type=""){
  library(gtsummary)
  library(ggplot2)
  attach(data)

  ## Variable qualitative
    if (is.character(var)==TRUE){
      df = data.frame(table(var))
      names(df) = c("var","eff")
      table = data.frame(var) |> tbl_summary()

      plot1 = ggplot(df,
                aes(x = "", y = eff, fill = var)) +
              geom_bar(stat = "identity", width=1) +
              coord_polar(theta = "y", start = 0) +
              geom_text(aes(label = eff),color="white",
                        position = position_stack(vjust=0.5))+
              theme_void()
```

```
    plot2 = ggplot(df,
              aes(x = var, y = eff, fill = var)) +
            geom_bar(stat = "identity", width=1) +
            theme(legend.position = "none")
  }

  ## Variable discrète
  else if (is.numeric(var)==TRUE){
      table = as.data.frame(summary(data.frame(var)))['Freq']
  }

  ## Sorties : graphiques ou tableaux
  if (all==TRUE){res = list(table,plot1)}
  else if(all=="table"){res = table}
  else if(all=="plot"){
      if(type=="bar"){res = plot2}
      else{res = plot1}
      }
  else {res = print("Choisir l'option table ou plot")}

  return(res)
}
```

Nous pouvons à présent cette fonction à nos variables dans la base de données *projet*. Il convient de noter que cette fonction est également applicable sur une autre base de données.

```
univarie(projet, sexe, all=TRUE)[[1]];univarie(projet, sexe, all=TRUE)[[2]]
```

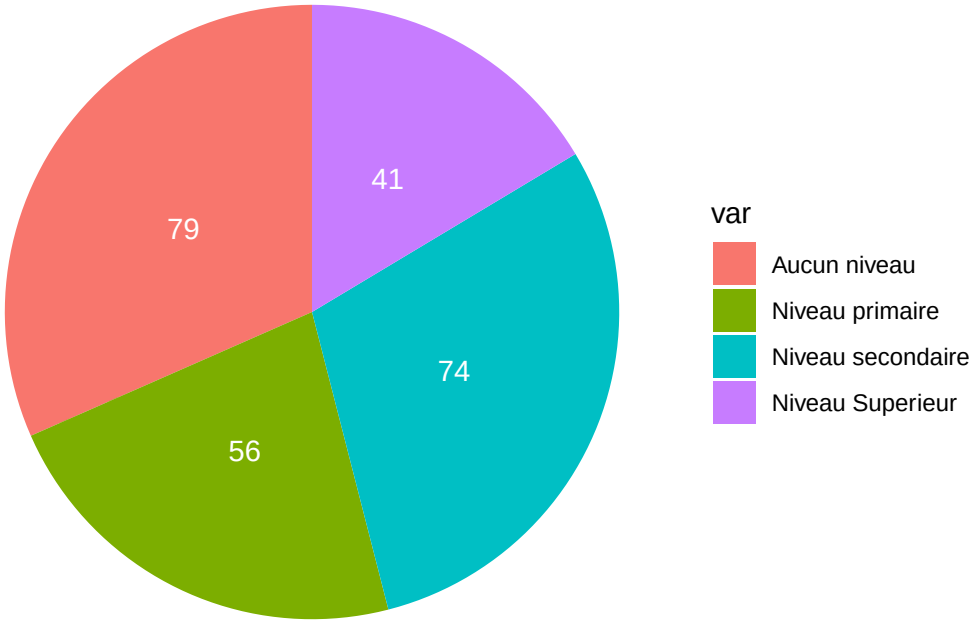
Characteristic	**N = 250**
Sexe du dirigeant/responsable de la PME	
Femme	191 (76%)
Homme	59 (24%)



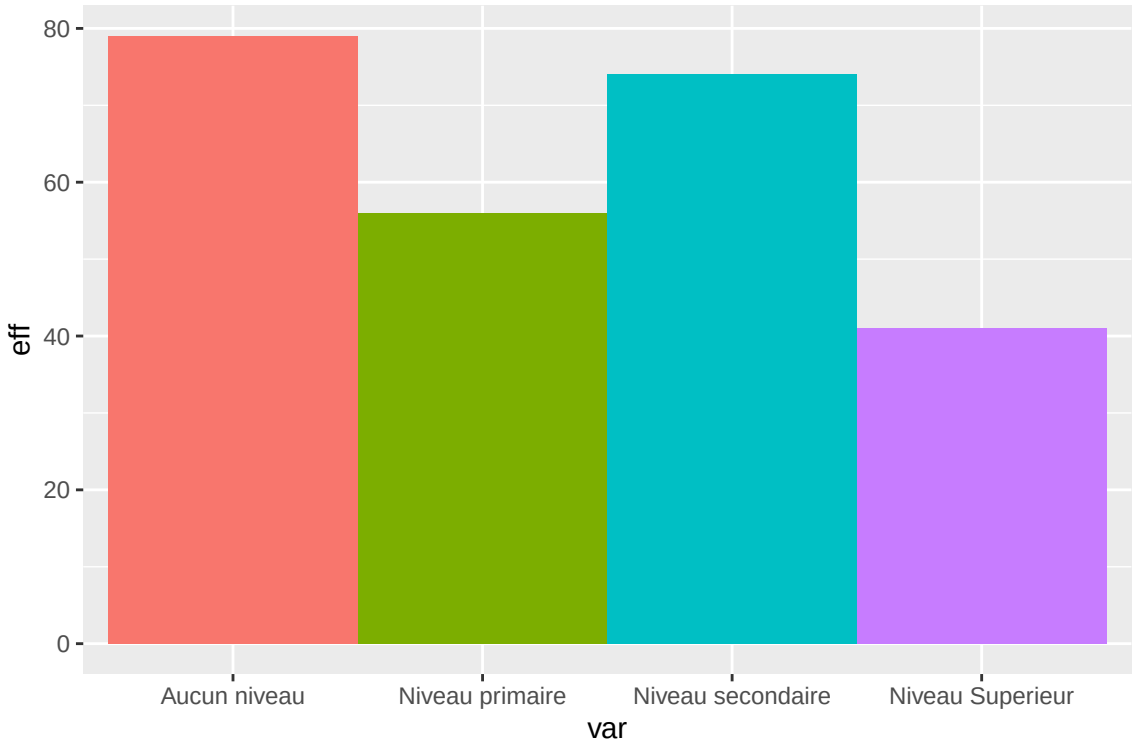

```
univarie(projet, region, all="table")
```

Characteristic	**N = 250**
Région	
Dakar	1 (0.4%)
Diourbel	34 (14%)
Fatick	30 (12%)
Kaffrine	8 (3.2%)
Kaolack	21 (8.4%)
Kolda	9 (3.6%)
Saint-Louis	42 (17%)
Sédhiou	4 (1.6%)
Thiès	51 (20%)
Ziguinchor	50 (20%)

```
univarie(projet, q25, all="plot")
```




```
univarie(projet, q25, all="plot", type="bar")
```



```r
univarie(projet, q24, all="table")
```

```
##                  Freq
## 1 Min.    :       18
## 2 1st Qu.:       45
## 3 Median  :       55
## 4 Mean    :  3106217
## 5 3rd Qu.:       62
## 6 Max.    :776530031
```

```r
bivarie <- function(df, var1, var2, all=TRUE){
  library(gtsummary)
  library(ggplot2)
  attach(df)

  ## Variable qualitative
  if (is.character(var1)==TRUE){
    df_plot = data.frame(table(var1, var2))
    names(df_plot) = c("Var1","Var2","Freq")

    plot = ggplot(df_plot,
                  aes(x = Var1, y = Freq, fill = Var2)) +
             geom_bar(stat = "identity", position = "fill")

    table = tbl_cross(df, df$var1, df$var2)
  }

  ## Sorties : graphiques ou tableaux
  if (all==TRUE){res = list(table,plot)}
  else if(all=="table"){res = table}
  else if(all=="plot"){res = plot}
  else {res = print("Choisir l'option table ou plot")}

  return(res)
}
```

```
bivarie(projet, sexe, q25, all="plot")
```