Projet Data

Margot Goudard

2023-2024

```r
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'

## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag

## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(lubridate)
```

```
##
## Attachement du package : 'lubridate'

## Les objets suivants sont masqués depuis 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(ggplot2)
library(tidyr)
library(purrr)

# Définir la fonction pour l'analyse descriptive des salaires
analyse_descriptive_salaire <- function(chemin_fichier) {


  # Charger les données depuis le fichier CSV
  data <- read.csv(chemin_fichier)

  # Calculer les statistiques descriptives pour le salaire actuel
  stats_salaire_actuel <- data %>%
    summarise(
      moyenne = mean(remuneration_prime, na.rm = TRUE),
      ecart_type = sd(remuneration_prime, na.rm = TRUE),
      minimum = min(remuneration_prime, na.rm = TRUE),
```

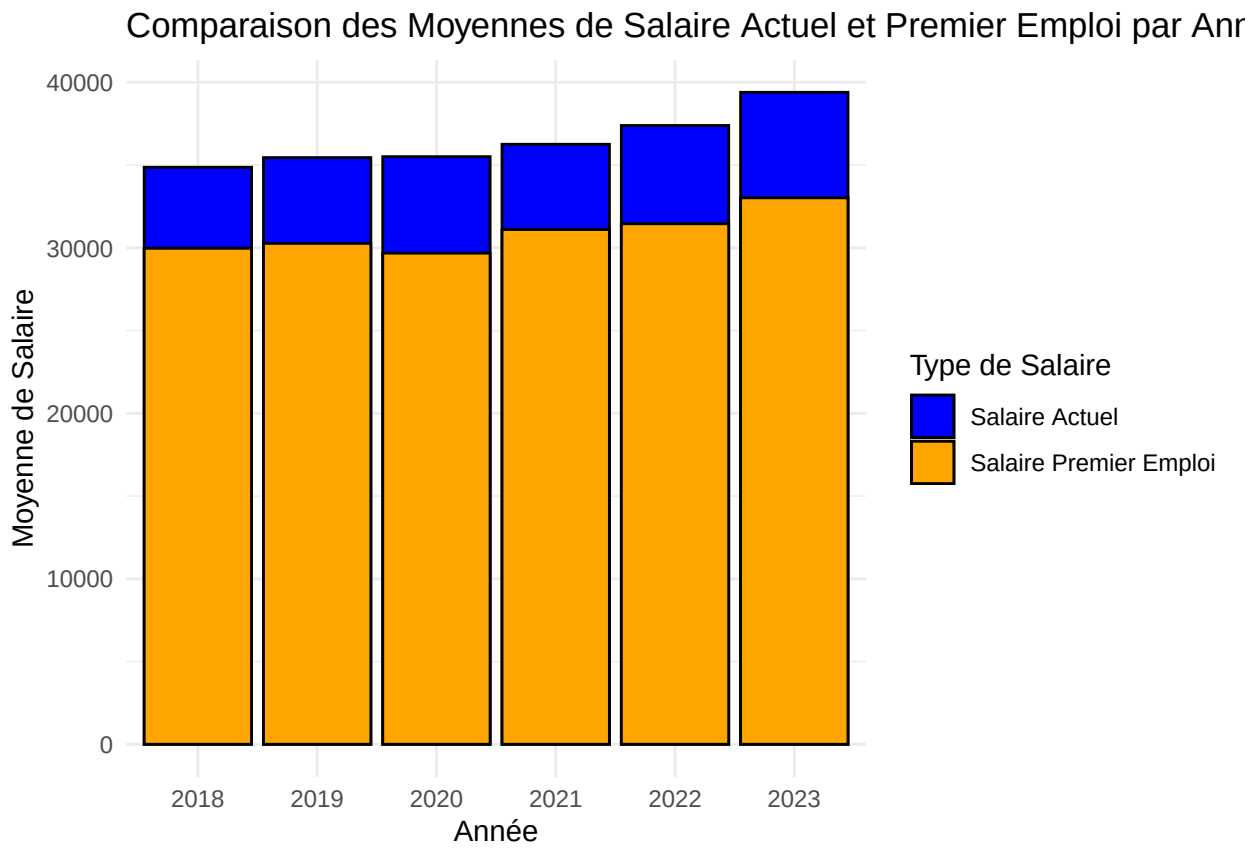
```
      maximum = max(remuneration_prime, na.rm = TRUE)
    )

  # Calculer les statistiques descriptives pour le salaire au premier emploi
  stats_salaire_premier_emploi <- data %>%
    summarise(
      moyenne = mean(remuneration_annuelle_brute_avec_prime_premier_emploi, na.rm = TRUE),
      ecart_type = sd(remuneration_annuelle_brute_avec_prime_premier_emploi, na.rm = TRUE),
      minimum = min(remuneration_annuelle_brute_avec_prime_premier_emploi, na.rm = TRUE),
      maximum = max(remuneration_annuelle_brute_avec_prime_premier_emploi, na.rm = TRUE)
    )

  # Affichage des statistiques descriptives pour le salaire actuel
  cat("Statistiques descriptives pour le salaire actuel en", annee, ":\n")
  print(stats_salaire_actuel)

  # Affichage des statistiques descriptives pour le salaire au premier emploi
  cat("\nStatistiques descriptives pour le salaire au premier emploi en", annee, ":\n")
  print(stats_salaire_premier_emploi)

  # Retourner les statistiques descriptives
  return(list(stats_salaire_actuel = stats_salaire_actuel, stats_salaire_premier_emploi = stats_salaire_
}

# Appliquer la fonction à chacun des cinq fichiers
chemins_fichiers <- c(
    "C:\\Users\\gouda\\OneDrive\\Documents\\INFO4A Montpellier\\projetdatascience\\Nettoyage_2018\\enque
    "C:\\Users\\gouda\\OneDrive\\Documents\\INFO4A Montpellier\\projetdatascience\\Nettoyage_2019\\2019
    "C:\\Users\\gouda\\OneDrive\\Documents\\INFO4A Montpellier\\projetdatascience\\Nettoyage_2020\\enque
    "C:\\Users\\gouda\\OneDrive\\Documents\\INFO4A Montpellier\\projetdatascience\\Nettoyage_2021\\enque
    "C:\\Users\\gouda\\OneDrive\\Documents\\INFO4A Montpellier\\projetdatascience\\Nettoyage_2022\\enque
    "C:\\Users\\gouda\\OneDrive\\Documents\\INFO4A Montpellier\\projetdatascience\\Nettoyage2023\\enquet
)

# Stocker les statistiques dans une liste
stats_list <- list()

# Création d'un dataframe pour stocker les années et les moyennes des salaires actuels et au premier em
df_moyennes_salaire <- data.frame(
  Annee = numeric(),
  Moyenne_Salaire_Actuel = numeric(),
  Moyenne_Salaire_Premier_Emploi = numeric()
)

# Définir l'année initiale
annee <- 2018

for (chemin in chemins_fichiers) {
  cat("\nAnalyse descriptive pour le fichier:", chemin, "\n")
  stats <- analyse_descriptive_salaire(chemin)
  stats_list[[chemin]] <- stats

  annee_fichier <- annee
```

```r
  # Ajouter les données au dataframe
  df_moyennes_salaire <- rbind(df_moyennes_salaire, data.frame(
    Annee = annee_fichier,
    Moyenne_Salaire_Actuel = stats$stats_salaire_actuel$moyenne,
    Moyenne_Salaire_Premier_Emploi = stats$stats_salaire_premier_emploi$moyenne
  ))

  # Incrémenter l'année
  annee = annee + 1
}
```

```
##
## Analyse descriptive pour le fichier: C:\Users\gouda\OneDrive\Documents\INFO4A Montpellier\projetdatas
## Statistiques descriptives pour le salaire actuel en 2018 :
##    moyenne ecart_type minimum maximum
## 1 34868.12   8709.454    6000   90000
##
## Statistiques descriptives pour le salaire au premier emploi en 2018 :
##    moyenne ecart_type minimum maximum
## 1 29973.48   7882.164    1500   70000
##
## Analyse descriptive pour le fichier: C:\Users\gouda\OneDrive\Documents\INFO4A Montpellier\projetdatas
## Statistiques descriptives pour le salaire actuel en 2019 :
##    moyenne ecart_type minimum maximum
## 1 35452.86   9558.979   12000  108000
##
## Statistiques descriptives pour le salaire au premier emploi en 2019 :
##    moyenne ecart_type minimum maximum
## 1 30270.06   8947.851    9000   83700
##
## Analyse descriptive pour le fichier: C:\Users\gouda\OneDrive\Documents\INFO4A Montpellier\projetdatas
## Statistiques descriptives pour le salaire actuel en 2020 :
##    moyenne ecart_type minimum maximum
## 1 35511.71   8859.159   11426   95000
##
## Statistiques descriptives pour le salaire au premier emploi en 2020 :
##    moyenne ecart_type minimum maximum
## 1 29683.26       7608    5100   60000
##
## Analyse descriptive pour le fichier: C:\Users\gouda\OneDrive\Documents\INFO4A Montpellier\projetdatas
## Statistiques descriptives pour le salaire actuel en 2021 :
##    moyenne ecart_type minimum maximum
## 1 36261.23   8945.599   10993   96000
##
## Statistiques descriptives pour le salaire au premier emploi en 2021 :
##    moyenne ecart_type minimum maximum
## 1 31106.35   6135.875   14400   60000
##
## Analyse descriptive pour le fichier: C:\Users\gouda\OneDrive\Documents\INFO4A Montpellier\projetdatas
## Statistiques descriptives pour le salaire actuel en 2022 :
##    moyenne ecart_type minimum maximum
## 1 37394.89   11598.64       0  141999
##
```

```
## Statistiques descriptives pour le salaire au premier emploi en 2022 :
##    moyenne ecart_type minimum maximum
## 1 31457.42   8718.672    2100   81000
##
## Analyse descriptive pour le fichier: C:\Users\gouda\OneDrive\Documents\INFO4A Montpellier\projetdatas
## Statistiques descriptives pour le salaire actuel en 2023 :
##    moyenne ecart_type minimum maximum
## 1 39399.53   10279.12   10000  120000
##
## Statistiques descriptives pour le salaire au premier emploi en 2023 :
##    moyenne ecart_type minimum maximum
## 1 33024.18   8134.172   10256   64000
```

```r
# Conversion de la colonne Année en facteur pour un affichage correct dans l'histogramme
df_moyennes_salaire$Annee <- as.factor(df_moyennes_salaire$Annee)

# Création de l'histogramme avec couleurs différenciées
ggplot() +
  geom_bar(data = df_moyennes_salaire, aes(x = Annee, y = Moyenne_Salaire_Actuel, fill = "Salaire Actuel
  geom_bar(data = df_moyennes_salaire, aes(x = Annee, y = Moyenne_Salaire_Premier_Emploi, fill = "Salai
  labs(title = "Comparaison des Moyennes de Salaire Actuel et Premier Emploi par Année",
       x = "Année",
       y = "Moyenne de Salaire",
       fill = "Type de Salaire") +
  scale_fill_manual(values = c("Salaire Actuel" = "blue", "Salaire Premier Emploi" = "orange")) +
  theme_minimal()
```



```r
# Stocker les identifiants uniques de chaque fichier dans une liste
identifiants_par_fichier <- list()

for (chemin in chemins_fichiers) {
  data <- read.csv(chemin)
  identifiants <- data %>% distinct(identifiant)
  identifiants_par_fichier[[chemin]] <- identifiants
}

# Compter le nombre d'occurrences de chaque identifiant à travers tous les fichiers
table_identifiants <- table(unlist(lapply(identifiants_par_fichier, function(x) x$identifiant)))

# Identifier les identifiants qui apparaissent exactement 3 fois
identifiants_3_occurrences <- names(table_identifiants[table_identifiants == 3])

# Afficher les identifiants qui apparaissent exactement 3 fois
cat("\nIdentifiants qui apparaissent exactement 3 fois dans tous les fichiers :\n")
```

```
##
## Identifiants qui apparaissent exactement 3 fois dans tous les fichiers :
```

```r
print(identifiants_3_occurrences)
```

```
##   [1] "1757709965251920" "1758627194075820" "1759195102887890"
##   [4] "1766816750910820" "1766816763108060" "1766816841421730"
##   [7] "1766817026164020" "1766817679604640" "1766825151490760"
##  [10] "1767188220767320" "1767190160340410" "1767270430761410"
##  [13] "1767362688988250" "1767472951669570" "1767472960852670"
##  [16] "1767472961734920" "1767472964490090" "1767472964563190"
##  [19] "1767473044458750" "1767638238509350" "1767641526750820"
##  [22] "1767827955884560" "1767828041541670" "1767828133549130"
##  [25] "1767833242840040" "1767833322084940" "1767833324843730"
##  [28] "1767833604358820" "1767838795874690" "1768183496273810"
##  [31] "1768183505228300" "1768183506095160" "1768183587378060"
##  [34] "1768183587449460" "1768183597444170" "1768183769929160"
##  [37] "1768549695730200" "1768549780066770" "1768549796148560"
##  [40] "1768549883109980" "1768556234866690" "1768989176927280"
##  [43] "1769007803921480" "1769007890066910" "1775257595835560"
##  [46] "1775263662780760" "1775263748033210" "1775264733646980"
##  [49] "1775270642923870" "1775345181363770" "1775413776498840"
##  [52] "1775413928705090" "1775426942739380" "1775503433566090"
##  [55] "1775569781638920" "1775570479028540" "1775643742622870"
##  [58] "1775789449534080" "1775873539752900" "1775940934139300"
##  [61] "1775951424215310" "1775955957977140" "1775955967798490"
##  [64] "1775956035087940" "1775956121915440" "1776019526187560"
##  [67] "1776091815313030" "1776091895895960" "1776102122857150"
##  [70] "1776102353371030" "1776106641864830" "1776106717098990"
##  [73] "1776106723950220" "1776106799883870" "1776106802247870"
##  [76] "1776106871435680" "1776106879687360" "1776106941550150"
##  [79] "1776166903795880" "1776166906953760" "1776166914449430"
##  [82] "1776166978376740" "1776166979781670" "1776167053468420"
##  [85] "1776167055017630" "1776167057227520" "1776167060979250"
```

```
##  [88] "1776167061760460" "1776167142122940" "1776170062868870"
##  [91] "1776177653526070" "1776242025600950" "1776551931058330"
##  [94] "1776615819557690" "1776621549469460" "1776621853639030"
##  [97] "1776695289443830" "1776844335602450" "1782327052505070"
## [100] "1782327052506100" "1782328943040360" "1782332205420030"
## [103] "1782332211989640" "1782332281044560" "1782332987613640"
## [106] "1782333524403100" "1782336121852570" "1782336190229570"
## [109] "1782336255547400" "1782336321386440" "1782340107568330"
## [112] "1782390418773570" "1782392903986310" "1782520259462620"
## [115] "1782520524460350" "1782588592603190" "1782592418482250"
## [118] "1782650225991380" "1782910863618510" "1782910866773040"
## [121] "1782917292365580" "1782917360433220" "1782921218686600"
## [124] "1782921284995230" "1782921357689910" "1783036913564170"
## [127] "1783049799429120" "1783100789632700" "1783100906791020"
## [130] "1783100908156800" "1783100980486340" "1783100980576070"
## [133] "1783101102983360" "1783101112615370" "1783101170208130"
## [136] "1783101174120860" "1783101174761720" "1783103471503020"
## [139] "1783103600401140" "1783106099598170" "1783164893041410"
## [142] "1783165012690460" "1783165143274690" "1783165151574090"
## [145] "1783165214972650" "1783174101378910" "1783236178137310"
## [148] "1783239889874180" "1783298107756990" "1783429202826460"
## [151] "1783429206115370" "1783429209187550" "1783429277924880"
## [154] "1783483872013320" "1783483938740120" "1783492828241480"
## [157] "1783492894845730" "1783551660836590" "1783551729336410"
## [160] "1783551851748090" "1783679157044340" "1783800339299580"
## [163] "1788405114733970" "1788410676370390" "1788410899702870"
## [166] "1788414696486980" "1788460091672310" "1788460149544450"
## [169] "1788460151741930" "1788460155174270" "1788460214031210"
## [172] "1788516594039430" "1788516656192290" "1788516762629580"
## [175] "1788516818654960" "1788522928111060" "1788522987504360"
## [178] "1788523040756520" "1788523094979770" "1788530340467290"
## [181] "1788636361004640" "1788636474868090" "1788639923539010"
## [184] "1788686096893490" "1788692359007600" "1788692414170340"
## [187] "1788752092801610" "1788753613143020" "1788855334679300"
## [190] "1788913234316040" "1789022859354490" "1789034153455870"
## [193] "1789034153473140" "1789078790593410" "1789078845865610"
## [196] "1789083315570470" "1789138902924160" "1789139006721990"
## [199] "1789139009510230" "1789139016135670" "1789139067979590"
## [202] "1789139068514980" "1789139125979470" "1789139176706330"
## [205] "1789196417036320" "1789245952283060" "1789245962819270"
## [208] "1789246007837330" "1789248290449950" "1789312769981880"
## [211] "1789312887151270" "1789357128908910" "1789420509162880"
## [214] "1789470186881510" "1789471732111400" "1789471841659060"
## [217] "1789578472568690" "1789582890113650" "1789582994412320"
## [220] "1789582999876920" "1789638170068160" "1789743836670460"
## [223] "1789743839991670" "1789743995690810" "1789751529102610"
## [226] "1789798905274890" "1793784758875840" "1793786767761630"
## [229] "1793838299424030" "1793841853377000" "1793993226818980"
## [232] "1794041578267320" "1794190819221150" "1794282911392610"
## [235] "1794332318260460" "1794332416683640" "1794332417128940"
## [238] "1794332465622220" "1794332471045900" "1794332471066140"
## [241] "1794332474084770" "1794628360740710" "1794635241835900"
## [244] "1794680868627280" "1794680914700640" "1794775550866700"
## [247] "1794775598358830" "1794775651335950" "1794779467201750"
```

```
## [250] "1794828417351500" "1794970940624460" "1798487221646170"
## [253] "1798487267951360" "1798751286695650" "1798760221846800"
## [256] "1799024859296950" "1799416649983310" "1802845972740320"
## [259] "1802846090696890" "1802929765219250" "1806661953120000"
## [262] "1814011059426950" "1857534014208740" "1857534014892390"
## [265] "1867188646284860" "1868183009217370" "1868183045269240"
## [268] "1868316566164700" "1868542954030230" "1875368398704990"
## [271] "1875570063107940" "1875637087387480" "1876166559014830"
## [274] "1876551296220380" "1882325974805300" "1882331845956370"
## [277] "1882462128942360" "1882522578241470" "1882874480356570"
## [280] "1882920801202420" "1883483808591720" "1883678675918150"
## [283] "1888404796843130" "1888462074344500" "1888516386797580"
## [286] "1888692414711220" "1889078511389260" "1889471531298430"
## [289] "1889471542205030" "1889477070705420" "1889798491780700"
## [292] "1894332194288450" "1894680652132240" "1899261168591870"
## [295] "1902847412238600" "1903737113808890"
```

```r
# Créer une boucle pour chaque triplet d'identifiants
# Créer une liste pour stocker les pourcentages d'augmentation pour chaque triplet d'identifiants
pourcentages_augmentation_1_an <- list()
pourcentages_augmentation_2_ans <- list()

# Créer une boucle pour chaque triplet d'identifiants
for (identifiant in identifiants_3_occurrences) {

  # Filtrer les données pour l'identifiant spécifique
  data_identifiant <- lapply(chemins_fichiers, function(chemin) {
    read.csv(chemin) %>% filter(identifiant == identifiant)
  })

  # Vérifier si le salaire actuel est mentionné dans les trois fichiers
  if (all(sapply(data_identifiant, function(df) "salaire_annuel_brut_avec_prime" %in% names(df)))) {
    # Extraire les salaires actuels pour les individus ayant rempli la case "salaire_annuel_brut_avec_p
    salaires_actuels <- sapply(data_identifiant, function(df) {
      # Filtrer les valeurs non numériques, NA et ceux qui ont rempli la case "salaire_annuel_brut_avec
      df_numeric <- df[is.numeric(df$salaire_annuel_brut_avec_prime) & !is.na(df$salaire_annuel_brut_ave
      if (ncol(df_numeric) > 0) {
        return(df_numeric$salaire_annuel_brut_avec_prime)
      } else {
        return(numeric(0))
      }
    })

    # Vérifier que tous les salaires actuels sont numériques
    if (all(sapply(salaires_actuels, function(x) length(x) > 0 && all(!is.na(x))))) {
      # Calculer les pourcentages d'augmentation d'une année à l'autre
      pourcentages_1_an <- numeric()
      pourcentages_2_ans <- numeric()
      for (i in 1:(length(salaires_actuels) - 2)) {
        pourcentage_augmentation_1_an <- ((salaires_actuels[[i + 1]][1] - salaires_actuels[[i]][1]) / sa
        pourcentage_augmentation_2_ans <- ((salaires_actuels[[i + 2]][1] - salaires_actuels[[i+1]][1]) /
        pourcentages_1_an <- c(pourcentages_1_an, pourcentage_augmentation_1_an)
        pourcentages_2_ans <- c(pourcentages_2_ans, pourcentage_augmentation_2_ans)
      }
```

```
      # Stocker les pourcentages dans les listes correspondantes
      pourcentages_augmentation_1_an[[identifiant]] <- pourcentages_1_an
      pourcentages_augmentation_2_ans[[identifiant]] <- pourcentages_2_ans
    } else {
      cat("Erreur: Les données de salaire actuel pour l'identifiant", identifiant, "ne sont pas valides
    }
  }
}

# Créer un vecteur pour stocker les moyennes des pourcentages d'augmentation après 1 an et 2 ans
moyennes_pourcentages_1_an <- numeric()
moyennes_pourcentages_2_ans <- numeric()

# Calculer la moyenne des pourcentages d'augmentation après chaque année
for (annee_diff in 1:(length(chemins_fichiers) - 1)) {
  moyenne_pourcentage_1_an <- mean(sapply(pourcentages_augmentation_1_an, function(pourcentages) pource
  moyenne_pourcentage_2_ans <- mean(sapply(pourcentages_augmentation_2_ans, function(pourcentages) pour
  moyennes_pourcentages_1_an <- c(moyennes_pourcentages_1_an, moyenne_pourcentage_1_an)
  moyennes_pourcentages_2_ans <- c(moyennes_pourcentages_2_ans, moyenne_pourcentage_2_ans)
}
```

```
## Warning in mean.default(sapply(pourcentages_augmentation_1_an,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA

## Warning in mean.default(sapply(pourcentages_augmentation_2_ans,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA

## Warning in mean.default(sapply(pourcentages_augmentation_1_an,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA

## Warning in mean.default(sapply(pourcentages_augmentation_2_ans,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA

## Warning in mean.default(sapply(pourcentages_augmentation_1_an,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA

## Warning in mean.default(sapply(pourcentages_augmentation_2_ans,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA

## Warning in mean.default(sapply(pourcentages_augmentation_1_an,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA

## Warning in mean.default(sapply(pourcentages_augmentation_2_ans,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA
```

```
## Warning in mean.default(sapply(pourcentages_augmentation_1_an,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA
```

```
## Warning in mean.default(sapply(pourcentages_augmentation_2_ans,
## function(pourcentages) pourcentages[annee_diff])): l'argument n'est ni
## numérique, ni logique : renvoi de NA
```

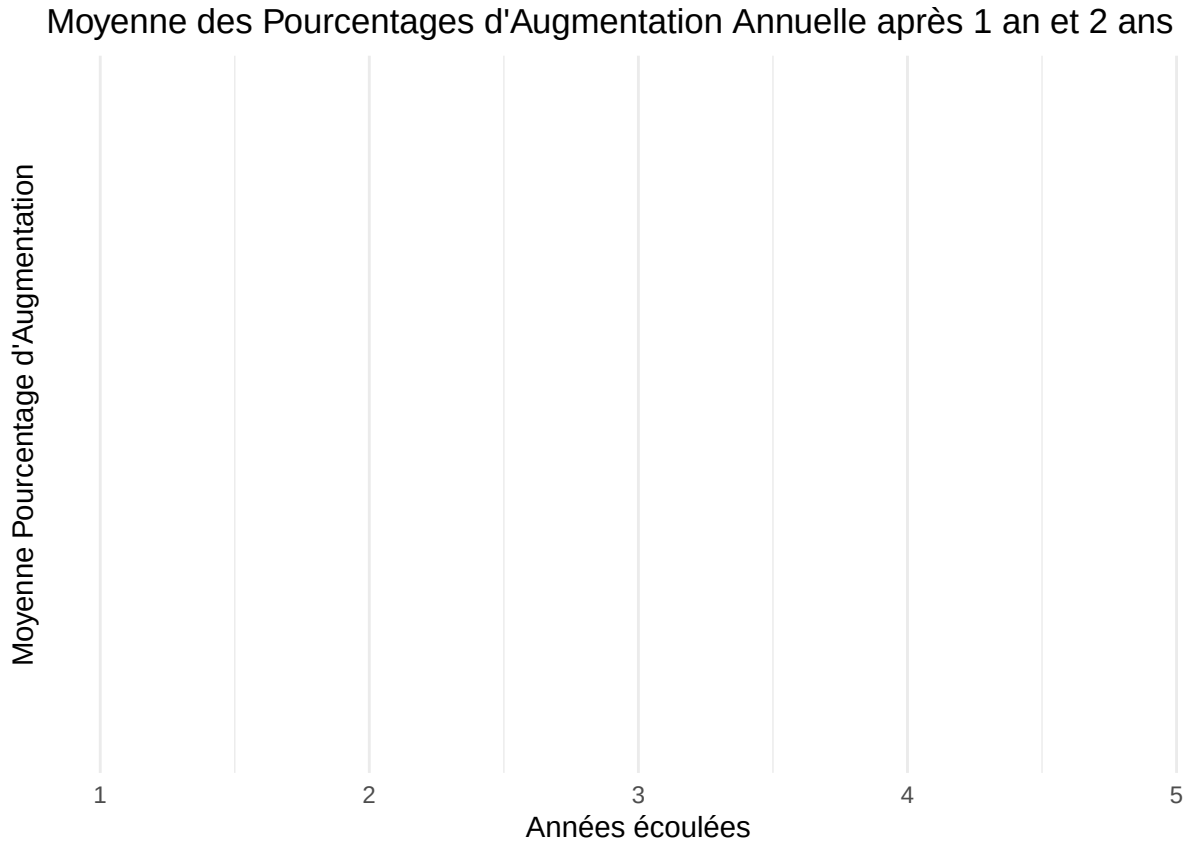
```
# Créer un graphique pour afficher les moyennes des pourcentages d'augmentation au fil des années
ggplot() +
  geom_line(aes(x = 1:(length(chemins_fichiers) - 1), y = moyennes_pourcentages_1_an), color = "blue", l
  geom_line(aes(x = 1:(length(chemins_fichiers) - 1), y = moyennes_pourcentages_2_ans), color = "red", l
  labs(title = "Moyenne des Pourcentages d'Augmentation Annuelle après 1 an et 2 ans",
       x = "Années écoulées",
       y = "Moyenne Pourcentage d'Augmentation") +
  theme_minimal() +
  scale_color_manual(values = c("blue", "red")) +
  scale_linetype_manual(values = c("solid", "dashed")) +
  guides(color = guide_legend(title = "Période"))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning in geom_line(aes(x = 1:(length(chemins_fichiers) - 1), y =
## moyennes_pourcentages_1_an), : Ignoring unknown parameters: `label`
```

```
## Warning in geom_line(aes(x = 1:(length(chemins_fichiers) - 1), y =
## moyennes_pourcentages_2_ans), : Ignoring unknown parameters: `label`
```

```
## Warning: Removed 5 rows containing missing values (`geom_line()`).
## Removed 5 rows containing missing values (`geom_line()`).
```




```r
# Stocker les données dans un seul dataframe pour l'ANOVA
data_anova <- df_moyennes_salaire %>%
  pivot_longer(cols = c(Moyenne_Salaire_Actuel, Moyenne_Salaire_Premier_Emploi),
               names_to = "Type_Salaire",
               values_to = "Moyenne_Salaire")

# Exécuter l'ANOVA
anova_result <- aov(Moyenne_Salaire ~ Annee * Type_Salaire, data = data_anova)

# Afficher le résumé de l'ANOVA
summary(anova_result)
```

```
##                     Df   Sum Sq  Mean Sq
## Annee                5 20783984  4156797
## Type_Salaire         1 92816451 92816451
## Annee:Type_Salaire   5   814209   162842
```

```r
# ANOVA TEMPORALITE

# Charger les données depuis les fichiers et sélectionner les colonnes pertinentes
dataframes <- map(chemins_fichiers, ~ read.csv(.x, sep = ",") %>%
  mutate(identifiant = as.character(identifiant)) %>%
  select(identifiant, remuneration_annuelle_brute_avec_prime_premier_emploi, remuneration_prime))

# Combinez les dataframes en un seul
```

```r
merged_data <- bind_rows(dataframes)

# Écrivez les données combinées dans un fichier CSV
write.csv(merged_data, "C:\\Users\\gouda\\OneDrive\\Documents\\INFO4A Montpellier\\projetDataR\\axetempo

# Lisez les données combinées à partir du fichier CSV
axetemporalite <- read.csv("C:\\Users\\gouda\\OneDrive\\Documents\\INFO4A Montpellier\\projetDataR\\axe

# ANOVA pour la variable : remuneration_annuelle_brute_avec_prime_premier_emploi
anova <- aov(remuneration_prime ~ remuneration_annuelle_brute_avec_prime_premier_emploi, data = axetempo
summary(anova)
```

```
##                                                   Df    Sum Sq   Mean Sq
## remuneration_annuelle_brute_avec_prime_premier_emploi   1 1.192e+10 1.192e+10
## Residuals                                             472 3.694e+10 7.827e+07
##                                                   F value Pr(>F)
## remuneration_annuelle_brute_avec_prime_premier_emploi   152.3 <2e-16 ***
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 3947 observations effacées parce que manquantes
```