Générer un rapport d'analyses

DAËRON Djayan

```
library(firstlibdjayan)
library(dplyr)
#> Warning: le package 'dplyr' a été compilé avec la version R 4.3.3
#>
#> 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
```

Introduction

Dans cette vignette, nous allons explorer l'utilisation des fonctions suivantes :

- `summary.commune()` : Cette fonction génère un résumé des informations pour une commune donnée.
- `summary.departement()` : Cette fonction génère un résumé des informations pour un département donné.
- `generer_rapport()` : Cette fonction génère un rapport au format PDF en utilisant un fichier Quarto, avec des paramètres pour la commune et le département.

Nous allons démontrer ces fonctions en utilisant des exemples concrets.

1. Résumé des informations d'une commune

La fonction `summary.commune()` prend un objet de type `commune` (un `data.frame` contenant les informations des élus d'une commune) et génère un résumé de ces informations, incluant le nom de la commune, le nombre d'élus, la distribution des âges des élus et l'élu le plus âgé.

```r
# Exemple d'utilisation de summary.commune
# Supposons que 'commune_data' est un data.frame représentant une commune

# Création d'un exemple fictif de données pour la commune
commune_data <- data.frame(
  Code.du.département = rep(45, 10),
  Libellé.du.département = rep("Loiret", 10),
  Code.de.la.commune = rep(45141, 10),
  Libellé.de.la.commune = rep("Faverelles", 10),
  Nom.de.l.élu = c("CHAUX", "COLLE", "DAUBRY", "EUGENE", "GOIRAND",
                   "LECUYER", "LETEUR", "MARET", "MARIE-LOUISE", "PIERROT"),
  Prénom.de.l.élu = c("Annie", "Sylvain", "Christiane", "Jacques", "Dominique",
                      "Frédérique", "Manuel", "Frédérick", "Joëlle", "Pascal"),
  Date.de.naissance = as.Date(c("1954-09-30", "1982-10-22", "1952-11-26", "1951-07-14",
                                "1962-04-20", "1963-03-25", "1965-12-15", "1975-11-08",
                                "1964-08-04", "1966-04-01")),
  Age = c(75, 37, 71, 74, 23, 48, 12, 52, 85, 38),
  Code.de.la.catégorie.socio.professionnelle = c(18, 3, 7, 5, 3, 4, 2, 1, 13, 11),
  Libellé.de.la.fonction = c(NA, NA, "2ème adjoint au Maire", "Maire", NA, NA, "1er adjoint a
  Date.de.mise.en.place = as.Date(c("2020-05-18", "2020-05-18", "2020-05-23", "2020-05-23",
                                    "2020-05-18", "2020-05-23", "2020-05-18", "2020-05-18", "2
  Code.pays = rep("FR", 10),
  stringsAsFactors = FALSE
)


# Attribuer la classe commune

commune_data <- creer_commune(commune_data)


# Résumé de la commune

summary(commune_data)
#> $nom_commune
#> [1] "Faverelles"
#>
#> $nombre_elus
#> [1] 10
#>
#> $distribution_ages_elus
#> $distribution_ages_elus$Q0
```

```
#> [1] 42
#>
#> $distribution_ages_elus$Q25
#> [1] 58.25
#>
#> $distribution_ages_elus$Q50
#> [1] 60.5
#>
#> $distribution_ages_elus$Q75
#> [1] 68
#>
#> $distribution_ages_elus$Q100
#> [1] 73
#>
#>
#> $elu_plus_age
#> $elu_plus_age$nom
#> [1] "EUGENE"
#>
#> $elu_plus_age$prénom
#> [1] "Jacques"
#>
#> $elu_plus_age$age
#> [1] 73
```

Le résumé pour cette commune peut inclure des informations telles que :

- Nom de la commune : Faverelles
- Nombre d'élus : 5
- Distribution des âges des élus : 45, 56, 34, 70, 62
- L'élu le plus âgé : Durand (70 ans)

2. Résumé des informations d'un département

La fonction `summary.departement()` fonctionne de manière similaire mais pour un département entier. Elle prend un objet de type `departement` (un `data.frame` contenant les informations des communes et des élus d'un département) et retourne des statistiques globales sur le département, telles que le nombre de communes, le nombre d'élus, et la distribution des âges des élus.

```r
# Exemple d'utilisation de summary.departement
# Supposons que 'departement_data' est un data.frame représentant un département

# Création d'un exemple fictif de données pour le département
departement_data <- data.frame(
  Code.du.département = rep(45, 10),
  Libellé.du.département = rep("Loiret", 10),
  Code.de.la.commune = rep(45141, 10),
  Libellé.de.la.commune = rep("Faverelles", 10),
  Nom.de.l.élu = c("CHAUX", "COLLE", "DAUBRY", "EUGENE", "GOIRAND",
                   "LECUYER", "LETEUR", "MARET", "MARIE-LOUISE", "PIERROT"),
  Prénom.de.l.élu = c("Annie", "Sylvain", "Christiane", "Jacques", "Dominique",
                      "Frédérique", "Manuel", "Frédérick", "Joëlle", "Pascal"),
  Date.de.naissance = as.Date(c("1954-09-30", "1982-10-22", "1952-11-26", "1951-07-14",
                                "1962-04-20", "1963-03-25", "1965-12-15", "1975-11-08",
                                "1964-08-04", "1966-04-01")),
  Age = c(75, 37, 71, 74, 23, 48, 12, 52, 85, 38),
  Code.de.la.catégorie.socio.professionnelle = c(18, 3, 7, 5, 3, 4, 2, 1, 13, 11),
  Libellé.de.la.fonction = c(NA, NA, "2ème adjoint au Maire", "Maire", NA, NA, "1er adjoint a
  Date.de.mise.en.place = as.Date(c("2020-05-18", "2020-05-18", "2020-05-23", "2020-05-23",
                                    "2020-05-18", "2020-05-23", "2020-05-18", "2020-05-18", "2
  Code.pays = rep("FR", 10),
  stringsAsFactors = FALSE
)


# Attribuer la classe département

departement_data <- creer_departement(departement_data)


# Résumé du département
summary(departement_data)
#> $nom_departement
#> [1] "Loiret"
#>
#> $nombre_communes
#> [1] 1
#>
#> $nombre_elus
#> [1] 10
#>
```

```
#> $distribution_age_elus
#> $distribution_age_elus$Q0
#> [1] 42
#>
#> $distribution_age_elus$Q25
#> [1] 58.25
#>
#> $distribution_age_elus$Q50
#> [1] 60.5
#>
#> $distribution_age_elus$Q75
#> [1] 68
#>
#> $distribution_age_elus$Q100
#> [1] 73
#>
#>
#> $elu_plus_age
#> $elu_plus_age$nom
#> [1] "EUGENE"
#>
#> $elu_plus_age$age
#> [1] 73
#>
#>
#> $elu_plus_jeune
#> $elu_plus_jeune$nom
#> [1] "COLLE"
#>
#> $elu_plus_jeune$age
#> [1] 42
#>
#>
#> $commune_plus_jeune
#> $commune_plus_jeune$nom
#> [1] "Faverelles"
#>
#> $commune_plus_jeune$distribution_ages
#> $commune_plus_jeune$distribution_ages$Q0
#> [1] 42
#>
#> $commune_plus_jeune$distribution_ages$Q25
```

```
#> [1] 58.25
#>
#> $commune_plus_jeune$distribution_ages$Q50
#> [1] 60.5
#>
#> $commune_plus_jeune$distribution_ages$Q75
#> [1] 68
#>
#> $commune_plus_jeune$distribution_ages$Q100
#> [1] 73
#>
#>
#>
#> $commune_plus_agee
#> $commune_plus_agee$nom
#> [1] "Faverelles"
#>
#> $commune_plus_agee$distribution_ages
#> $commune_plus_agee$distribution_ages$Q0
#> [1] 42
#>
#> $commune_plus_agee$distribution_ages$Q25
#> [1] 58.25
#>
#> $commune_plus_agee$distribution_ages$Q50
#> [1] 60.5
#>
#> $commune_plus_agee$distribution_ages$Q75
#> [1] 68
#>
#> $commune_plus_agee$distribution_ages$Q100
#> [1] 73
```

Le résumé pour ce département pourrait inclure des informations comme :

- Nom du département : Loire-Atlantique
- Nombre de communes : 3
- Nombre d'élus : 5
- Distribution des âges des élus : 45, 56, 34, 70, 62
- L'élu le plus âgé : Durand (70 ans)
- La commune avec la moyenne d'âge la plus basse : Nantes
- La commune avec la moyenne d'âge la plus élevée : Saint-Nazaire

3. Générer un rapport d'analyses

La fonction `generer_rapport()` génère un rapport au format PDF en utilisant un fichier Quarto `.qmd`. Elle prend en entrée les codes de la commune et du département, et génère un fichier PDF avec les informations correspondantes.

```
# Exemple d'utilisation de generer_rapport
# Supposons que nous voulons générer un rapport pour Nantes et Loire-Atlantique

generer_rapport(commune = 44109,
                departement = 44,
                output = "rapport_nantes_et_loire_atlantique")
#>
#>
#> processing file: rapport.qmd
#>
  |
  |                                                      |   0%
  |
  |...                                                   |   7%
  |
  |.......                                               |  13% [unnamed-chunk-1]
  |
  |..........                                            |  20%
  |
  |..............                                        |  27% [unnamed-chunk-2]
  |
  |..................                                    |  33%
  |
  |......................                                |  40% [unnamed-chunk-3]
  |
  |.........................                             |  47%
  |
  |.............................                         |  53% [unnamed-chunk-4]
  |
  |................................                      |  60%
  |
  |....................................                  |  67% [unnamed-chunk-5]
  |
  |.......................................               |  73%
  |
  |...........................................           |  80% [unnamed-chunk-6]
```

```
  |
  |.....................................       |  87%
  |
  |.................................................   |  93% [unnamed-chunk-7]
  |
  |.................................................| 100%

#> output file: rapport.knit.md
#>
#> pandoc --output ..\..\..\..\..\..\Desktop\firstlibdjayan\vignettes\rapport_nantes_et_loire
#>   to: html
#>   standalone: true
#>   section-divs: true
#>   html-math-method: mathjax
#>   wrap: none
#>   default-image-extension: png
#>
#> metadata
#>   document-css: false
#>   link-citations: true
#>   date-format: long
#>   lang: en
#>   title: Rapport
#>   author: Djayan DAËRON
#>
#> Output created: ..\..\..\..\..\..\Desktop\firstlibdjayan\vignettes\rapport_nantes_et_loire
#> Le rapport a été généré avec succès.
```

Cette fonction crée un fichier PDF appelé `rapport_nantes_et_loire_atlantique.html` qui inclut des analyses sur la commune de Nantes et le département de Loire-Atlantique. Les données peuvent être ajustées pour correspondre à des valeurs réelles provenant de vos jeux de données.

Conclusion

Nous avons montré comment utiliser les fonctions `summary.commune()`, `summary.departement()`, et `generer_rapport()` pour générer des rapports d'analyses détaillés. Ces fonctions peuvent être adaptées à d'autres communes et départements pour créer des rapports personnalisés.

N'hésitez pas à adapter ces fonctions en fonction de vos propres données pour générer des rapports complets et personnalisés.