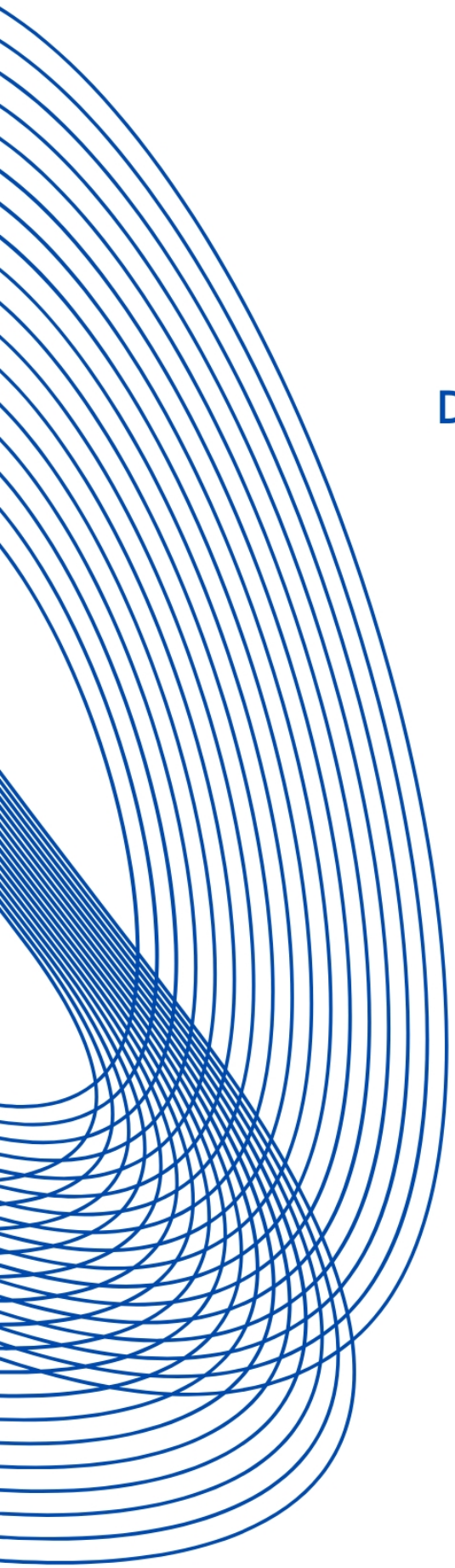
IAE Nantes
Économie & Management
Pôle Sociétés

Nantes Université

Nantes Université

5 DÉCEMBRE 2024



REMPLACEMENT DES VALEURS MANQUANTES DANS UNE BASE DE DONNÉES SUR LES VOTES EN LOIRE-ATLANTIQUE

Dossier de biostatistiques
Master 1 ECAP

DAËRON DJAYAN
MARTINEZ ESTELLE

Nantes Université

— Année universitaire 2024-2025 —

Biostatistiques : Remplacement des valeurs manquantes dans une base de données sur les votes en Loire-Atlantique

Daeron Djayan, Martinez Estelle

Table of contents

```r
setwd("C:/Users/Djayan/Desktop/Projet Biostat Rendu")
```

```r
library(readxl)
library(writexl)
library(dplyr)
library(purrr)
library(ggplot2)
library(geosphere)
library(FNN)
library(mice)
library(corrplot)
```

1 Introduction

Pour ce rapport, nous avons choisi d'utiliser une base de données relative aux résultats du premier tour de l'élection présidentielle de 2022, en France. Cette base regroupe les résultats détaillés de l'élection pour toutes les communes de France métropolitaine, permettant ainsi d'avoir un aperçu précis des comportements électoraux à l'échelle locale et nationale. Les données incluent des informations géographiques, telles que les codes et libellés des départements, des circonscriptions et des communes, ainsi que des statistiques de participation (inscrits, abstentions, votants), et des détails concernant les bulletins non valides (blancs, nuls).

En outre, la base fournit une répartition des votes exprimés pour chaque candidat, avec des indicateurs clés tels que le nombre de voix obtenues, les pourcentages par rapport aux inscrits et aux votes exprimés, ainsi que des données démographiques sur les candidats, notamment leur sexe. Ces informations offrent une base riche et détaillée pour analyser les dynamiques électorales en 2022, tant en termes de participation citoyenne que de résultats par candidat, et permettent d'approfondir l'étude des préférences électorales à travers tout le territoire.

Dans le cadre de notre analyse, nous avons enrichi cette base en y ajoutant une seconde base contenant des indicateurs géographiques, notamment la latitude et la longitude pour chaque commune de France. Cette intégration permet de relier les résultats électoraux aux localisations précises des communes, facilitant ainsi une analyse géographique plus fine des comportements électoraux.

Pour limiter la portée géographique et rendre l'étude plus pertinente, nous avons choisi de ne retenir que les données relatives au département de Loire-Atlantique. Ce département, représentatif d'une diversité géographique et démographique, offre un cadre cohérent pour l'analyse des comportements électoraux tout en permettant une gestion plus aisée des données. En restreignant notre analyse à ce périmètre géographique, nous pouvons également mieux maîtriser la taille de la base de données, ce qui est un élément clé dans le cadre d'un exercice méthodologique tel que celui-ci.

En effet, l'objectif principal de cette étude n'est pas de travailler sur une base de données à grande échelle, mais plutôt d'utiliser différentes méthodes de remplacement des données manquantes. La taille de la base, bien que représentative, reste limitée pour des raisons pratiques. Nous cherchons avant tout à explorer les performances des différentes stratégies de complétion des données, et une base plus petite permet de mieux isoler les effets de chaque méthode sans risquer de diluer les résultats dans une masse trop importante de données. Ainsi, en restreignant la portée géographique, nous nous assurons d'une analyse ciblée et d'une gestion optimale des données manquantes, tout en gardant un échantillon suffisamment varié pour tirer des conclusions pertinentes.

Enfin, pour évaluer les méthodes de gestion des données manquantes, nous avons volontairement introduit des valeurs NA (manquantes) de manière aléatoire dans la base de données, représentant 20 % des valeurs initiales. Cette démarche nous permet de simuler un cas réel où des données sont incomplètes et de tester différentes stratégies pour gérer ces valeurs manquantes. Bien que nous ayons introduit nous-mêmes ces valeurs manquantes, nous allons traiter cette base comme si elle nous avait été fournie dans un état incomplet, avec des NA présents sur certaines lignes.

Il est important de noter que, dans ce contexte, toutes les erreurs de saisie, les valeurs aberrantes, les incohérences ou les valeurs fausses (par exemple des dates de naissance incorrectes, des codes de département erronés, ou encore des votes exprimés supérieurs au nombre d'inscrits) ont déjà été remplacées par des valeurs manquantes (`NA`). De plus, aucune ligne n'a été totalement supprimée, ce qui garantit que toutes les observations sont conservées, même si une partie de leurs données est

manquante. Étant donné que la base provient d'auteurs fiables et que les données ont été vérifiées et nettoyées au préalable, nous n'avons pas à nous préoccuper de la véracité des informations. Ainsi, nous sommes certains que toutes les valeurs problématiques ont déjà été remplacées par des `NA`. Ce sujet porte donc bien sur le remplacement des valeurs manquantes et non sur la détection de problèmes.

Dans notre dossier, nous allons aborder de manière détailler, les différentes étapes qui nous ont servies pour compléter la base de données. Tout d'abord, nous venons de poser les bases du sujet traité. Ensuite, nous passerons à la Construction de la base de données, où nous expliquerons les procédures d'importation, de modification et de sauvegarde des données, ainsi que la gestion des valeurs manquantes (NA). La troisième partie sera consacrée à la Présentation des données, incluant des analyses univariées et bivariées pour différentes combinaisons de variables quantitatives et qualitatives. Nous explorerons ensuite le Remplacement des valeurs manquantes, en décrivant les méthodes logiques et approximatives utilisées pour gérer les valeurs manquantes. Enfin, nous conclurons ce dossier en résumant les points clés.

2 Construction de la base de données

2.1 Importation des bases

```
options(digits = 15)

data_vote <- read_excel("data/Données votes.xlsx")

data_geo <- read.csv("data/Données géographiques.csv")
```

2.2 Modification des données

```
data_vote <- data_vote[data_vote$`Code du département` == "44", ]
data_full <- data_vote[,c(4:8,10,12,15,18,21)]
colnames(data_full)[1:10] = c("circonscription", #circonscription rattachement
                        "code_commune", #numéro de commune dans le département
                         "nom_commune", #nom de la commune
                         "code_bur_vote", #numéro du bureau de vote
                         "inscrits", #nombre d'inscrits
                         "tx_absents", #taux d'absents (nb absents / nb inscrits)
                         "tx_votants", #taux votants (nb votants / nb inscrits)
                         "tx_blancs", #taux votes blancs (nb blancs / nb votants)
                         "tx_nuls", #taux votes nuls (nb votes nuls / nb votants)
                         "tx_exprimes" #taux votes exprimés (nb exp / nb votants)
                         )

data_geo <- data_geo[data_geo$code_departement == "44", ]
data_geo <- data_geo[,c(6:8)]


# Retirer les lignes devenues doublons en retirant les variables

data_geo <- unique(data_geo)


# Convertir les colonnes pour avoir un format homogène

data_full$code_commune <- as.integer(data_full$code_commune)


# Associer la longitude et latitude avec les communes

data_full <- merge(data_full, data_geo,
                   by.x = "code_commune",
```

```
                    by.y = "code_commune")

data_full <- data_full[,-1] #retirer la ligne "Code de la commune"
```

2.3 Type des données

```
str(data_full)
```

```
'data.frame':   1110 obs. of  11 variables:
 $ circonscription: chr  "6ème circonscription" "10ème circonscription" "10ème circonscription"
 $ nom_commune    : chr  "Abbaretz" "Aigrefeuille-sur-Maine" "Aigrefeuille-sur-Maine" "Aigrefeu
 $ code_bur_vote  : chr  "0001" "0001" "0002" "0003" ...
 $ inscrits       : chr  "1553" "941" "1046" "1011" ...
 $ tx_absents     : num  24.2 20.2 21.8 21.4 22.4 ...
 $ tx_votants     : num  75.8 79.8 78.2 78.6 77.6 ...
 $ tx_blancs      : num  1.95 2 2.2 1.38 1.61 2.52 1.06 2.48 2.45 1.76 ...
 $ tx_nuls        : num  0.68 1.2 0.73 0.75 0.4 0.27 0.75 0.69 0.86 0.59 ...
 $ tx_exprimes    : num  97.4 96.8 97.1 97.9 98 ...
 $ latitude       : num  47.6 47.1 47.1 47.1 47.4 ...
 $ longitude      : num  -1.49 -1.42 -1.42 -1.42 -1.18 ...
```

```
data_full$circonscription <- as.factor(data_full$circonscription)
data_full$nom_commune <- as.factor(data_full$nom_commune)
data_full$code_bur_vote <- as.integer(data_full$code_bur_vote)
data_full$inscrits <- as.numeric(data_full$inscrits)
```

2.4 Sauvegarder les données complètes

L'idée est ici de sauvegarder les données complètes du département pour n'avoir que cette base de données à ouvrir lors du lancement et ne pas avoir à charger les bases `data_vote` et `data_geo` qui sont beaucoup plus lourdes et longues à importer.

```
# Chemin complet

chemin_fichier <- file.path("data", "Données complètes.xlsx")


# Enregistrer le fichier Excel dans le dossier data

write_xlsx(
  x = data_full,
  path = chemin_fichier,
  col_names = TRUE,
```

```
  format_headers = TRUE
)
```

Il faut donc rouvrir / ouvrir la base de données

```
data_full <- read_excel("data/Données complètes.xlsx")
```

2.5 Incorporer des NA

```
data_proj_1 <- data_full

n_row <- nrow(data_proj_1) #nombre de lignes
n_col <- ncol(data_proj_1) #nombre de lignes

n_obs <- round(n_row*n_col*0.20) # 20% du nombre de lignes


# Créer des paires de lignes et colonnes uniques

combinaisons_possibles <- expand.grid(ligne = 1:n_row, colonne = 1:n_col)

paires_distinctes <- combinaisons_possibles[
                           sample(1:nrow(combinaisons_possibles), n_obs), ]


# Mettre des NA

data_proj_1 <- as.matrix(data_proj_1)
data_proj_1[cbind(paires_distinctes[, 1], paires_distinctes[, 2])] <- NA

sum(is.na(data_proj_1))/(n_row*n_col) # bon %
```

```
[1] 0.2
```

```
# Chemin complet

chemin_fichier <- file.path("data", "Données projet.xlsx")

# Enregistrer le fichier Excel dans le dossier data

# /!\ Ne pas tirer les # sinon la base va changer /!\

##write_xlsx(
##  x = data_proj_1,
```

```
##  path = chemin_fichier,
##  col_names = TRUE,
##  format_headers = TRUE
##)

data_save_1 <- read_excel("data/Données projet.xlsx")
```

3 Présentation des données

Avant de commencer l'analyse des données et le remplacement des valeurs manquantes, il est crucial de réimporter notre base de données. Bien que nous ayons déjà construit cette base dans les étapes précédentes, il a été nécessaire de la sauvegarder afin de pouvoir la réutiliser. En effet, si nous ne suivons pas cette procédure et relançons simplement le code, les valeurs manquantes (`NA`) seront modifiées. Cela est dû au fait que ces valeurs ont été assignées aléatoirement, et elles le seront à nouveau à chaque exécution du code. Un tel comportement poserait problème, car cela modifierait l'analyse qui va suivre et compromettrait la validité des résultats. De plus, certains des remplacements effectués dans les étapes suivantes dépendent des spécificités de cette version de la base de données et ne sont pas généralisables.

De ce fait, il est impératif de travailler avec une base de données fixe et stable, car une base qui évolue continuellement n'est pas logique. Lorsqu'on travaille sur des données, celles-ci doivent être soit correctes dès le départ, soit identifiées comme manquantes et laissées ainsi. Une donnée ne peut pas être, un coup manquante, puis, lors d'un nouveau traitement, ne plus l'être, et ainsi de suite. Une fois qu'une donnée manquante est détectée, elle doit le rester, sauf si elle est complétée par une méthode d'imputation. Cela permet de maintenir la stabilité et la cohérence de l'analyse, évitant ainsi toute incohérence dans le traitement des données.

Importons donc notre base de données

```
data <- read_excel("data/Données projet.xlsx")

str(data)
```

```
tibble [1,110 x 11] (S3: tbl_df/tbl/data.frame)
 $ circonscription: chr [1:1110] "6ème circonscription" "10ème circonscription" "10ème circons
 $ nom_commune    : chr [1:1110] "Abbaretz" "Aigrefeuille-sur-Maine" "Aigrefeuille-sur-Maine"
 $ code_bur_vote  : chr [1:1110] "1" "1" "2" "3" ...
 $ inscrits       : num [1:1110] 1553 NA 1046 1011 959 ...
 $ tx_absents     : num [1:1110] 24.2 20.2 21.8 NA 22.4 ...
 $ tx_votants     : num [1:1110] 75.8 79.8 78.2 78.6 77.6 ...
 $ tx_blancs      : num [1:1110] 1.95 NA 2.2 NA 1.61 2.52 1.06 2.48 NA 1.76 ...
 $ tx_nuls        : num [1:1110] 0.68 1.2 0.73 0.75 0.4 0.27 0.75 0.69 0.86 0.59 ...
 $ tx_exprimes    : num [1:1110] 97.4 96.8 97.1 97.9 98 ...
 $ latitude       : num [1:1110] 47.6 47.1 47.1 47.1 47.4 ...
 $ longitude      : num [1:1110] NA NA -1.42 -1.42 -1.18 ...
```

```
data$circonscription <- as.factor(data$circonscription)
data$nom_commune <- as.factor(data$nom_commune)
data$code_bur_vote <- as.factor(data$code_bur_vote)

head(data)
```

```
# A tibble: 6 x 11
  circonscription       nom_commune code_bur_vote inscrits tx_absents tx_votants
```

```
  <fct>                 <fct>       <fct>            <dbl>      <dbl>      <dbl>
1 6ème circonscription  Abbaretz    1                 1553       24.2       75.8
2 10ème circonscription Aigrefeuil~ 1                   NA       20.2       79.8
3 10ème circonscription Aigrefeuil~ 2                 1046       21.8       78.2
4 10ème circonscription Aigrefeuil~ 3                 1011       NA         78.6
5 <NA>                  Ancenis-Sa~ 1                  959       22.4       77.6
6 <NA>                  Ancenis-Sa~ 2                  978       22.9       77.1
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

La base de données que nous avons choisie d'étudier cherche à analyser, comme expliqué en introduction, la participation électorale au niveau local, avec en plus de cela des informations relatives à la géographie. Elle est constituée de 1110 observations et de 11 variables. Les variables sont les suivantes.

1. `circonscription` (Qualitative)
    - Description : Cette variable représente la circonscription électorale à laquelle chaque commune appartient. Les circonscriptions sont des divisions territoriales qui regroupent plusieurs communes pour les élections législatives. La variable est catégorique et comprend 10 niveaux, correspondant à différentes circonscriptions (par exemple, "10ème circonscription", "6ème circonscription", etc.).
    - Analyse : La répartition des communes selon les circonscriptions permet de comprendre la distribution géographique des résultats électoraux. Cela peut être utile pour observer des tendances électorales spécifiques à certaines zones géographiques et identifier des variations régionales importantes dans les comportements de vote.
2. `nom_commune` (Qualitative)
    - Description : Cette variable contient le nom de la commune où les données ont été recueillies. Il y a 192 niveaux différents pour cette variable, ce qui correspond à 192 communes dans le jeu de données.
    - Analyse : Le nom de la commune est essentiel pour relier les résultats électoraux aux zones géographiques. Cette variable permet d'analyser les résultats au niveau local et d'explorer des disparités entre différentes communes, ce qui peut éclairer sur des comportements électoraux spécifiques à certaines populations.
3. `code_bur_vote` (Qualitative)
    - Description : Ce champ représente le code du bureau de vote, un identifiant unique attribué à chaque bureau de vote dans une commune. Il y a 204 niveaux différents pour cette variable.
    - Analyse : Le code du bureau de vote permet d'identifier précisément les différentes sections de vote dans une commune. Cela peut être utile pour des analyses fines de la participation électorale et des résultats au niveau microgéographique, offrant ainsi une vision plus détaillée du comportement des électeurs.
4. `inscrits` (Quantitative)
    - Description : Cette variable représente le nombre d'inscrits sur les listes électorales pour chaque bureau de vote ou commune. C'est une variable continue.

- Analyse : Le nombre d'inscrits est une donnée clé pour analyser la taille du potentiel électoral. Il permet de calculer des taux de participation et d'abstention, offrant une base pour mesurer la mobilisation électorale dans chaque zone géographique.

5. `tx_absents` (Quantitative)
   - Description : Ce taux représente la proportion d'inscrits qui se sont abstenus de voter, exprimée en pourcentage. Comme les autres taux, cette variable est continue.
   - Analyse : Le taux d'abstention est un indicateur essentiel de l'engagement électoral. Analyser ce taux permet de comprendre les niveaux de participation et d'explorer des questions sur la désaffection politique, l'accessibilité des bureaux de vote, ou l'impact des facteurs socio-économiques sur l'abstention.
6. `tx_votants` (Quantitative)
   - Description : Ce taux représente la proportion d'inscrits qui ont effectivement voté, également exprimée en pourcentage. C'est le complément de `tx_absents`.
   - Analyse : Ce taux mesure directement l'engagement des électeurs et la réussite de la mobilisation électorale. Une analyse de cette variable permet d'évaluer la participation effective et de la comparer à d'autres régions ou à des élections précédentes pour mieux comprendre les dynamiques de participation.
7. `tx_blancs` (Quantitative)
   - Description : Ce taux représente la proportion des bulletins blancs par rapport aux votes exprimés, exprimée en pourcentage.
   - Analyse : Le taux de votes blancs donne une indication sur le mécontentement des électeurs ou leur manque de préférence pour les candidats. Analyser cette variable permet d'identifier des zones où les électeurs se sentent peu représentés, ce qui peut avoir des implications sur la qualité du processus électoral.
8. `tx_nuls` (Quantitative)
   - Description : Ce taux représente la proportion des bulletins nuls, exprimée en pourcentage des votes exprimés.
   - Analyse : Le taux de votes nuls reflète également une forme de rejet du processus électoral. L'analyse de cette variable permet de mieux comprendre le niveau de confusion ou d'irritation des électeurs envers les candidats ou le processus de vote, et peut aussi aider à identifier des problèmes dans le déroulement des élections.
9. `tx_exprimes` (Quantitative)
   - Description : Ce taux représente la proportion des votes valides exprimés, c'est-à-dire les votes pour un candidat, en pourcentage des inscrits.
   - Analyse : Ce taux est essentiel pour mesurer la validité des votes dans chaque bureau de vote. Il donne une vision claire de la qualité du processus électoral, permettant d'évaluer si les électeurs sont bien informés et engagés dans le processus.
10. `latitude` (Quantitative)

- Description : La latitude géographique de chaque commune. Cette variable est continue et représente la position géographique sur l'axe nord-sud.

- Analyse : La latitude permet de situer géographiquement les communes. Dans le cadre d'une analyse géospatiale, cette variable peut être utile pour observer des tendances géographiques dans les comportements électoraux, comme des variations de participation en fonction de la localisation nord-sud.

11. `longitude` (Quantitative)

- Description : La longitude géographique de chaque commune. Cette variable est aussi continue et représente la position sur l'axe est-ouest.
- Analyse : Tout comme la latitude, la longitude permet de situer précisément les communes et de mener des analyses spatiales. Combinée avec la latitude, elle permet de réaliser des cartes et d'analyser les tendances électorales selon des critères géographiques précis, tels que la proximité des grandes villes ou des zones rurales.

Les variables présentes dans notre jeu de données peuvent être regroupées en trois grandes catégories, ce qui permet d'organiser l'analyse selon différentes dimensions pertinentes. Ce découpage aide à mieux comprendre les interactions entre les différents facteurs et leur influence sur les comportements électoraux.

1. Géographique - administratif (qualitative)
   Ce groupe regroupe les variables relatives à l'organisation territoriale des élections, à savoir `circonscription`, `nom_commune`, et `code_bur_vote`. Ces variables sont toutes qualitatives et sont essentielles pour l'analyse des résultats à l'échelle géographique.
   - Le regroupement de ces variables permet d'étudier les dynamiques électorales en fonction des divisions administratives et géographiques. Par exemple, on peut observer des différences de participation ou de taux d'abstention entre les différentes communes ou circonscriptions.
   - Ces variables permettent également d'étudier les variations géographiques à un niveau fin (par bureau de vote, commune, ou circonscription).
2. Participation et Abstention (quantitative)
   Ce groupe regroupe toutes les variables liées à la participation électorale et à la validité des votes : `inscrits`, `tx_absents`, `tx_votants`, `tx_blancs`, `tx_nuls`, et `tx_exprimes`.
   - Ces variables sont toutes quantitatives et permettent d'analyser les comportements des électeurs en termes de participation, d'abstention, et de types de votes exprimés (blancs, nuls ou valides).
   - En les regroupant, nous pouvons analyser la participation au niveau des communes et bureaux de vote, ainsi que l'impact de facteurs géographiques ou démographiques sur ces comportements. Cela permet également d'évaluer les effets de l'abstention, des votes blancs et nuls sur la validité des élections.
3. Géographique (quantitatif) Les variables `latitude` et `longitude` sont des données géographiques quantitatives, permettant de situer les communes sur une carte.
   - Ce groupe est distinct des autres car ces variables permettent d'effectuer des analyses spatiales. En les regroupant, il devient possible d'explorer des tendances géographiques dans les données électorales, telles que les variations de participation ou d'abstention

selon les régions.

- L'analyse géospatiale est un outil puissant pour détecter des motifs régionaux ou locaux qui pourraient être invisibles sans une telle approche.

3.1 Analyse univariée des données

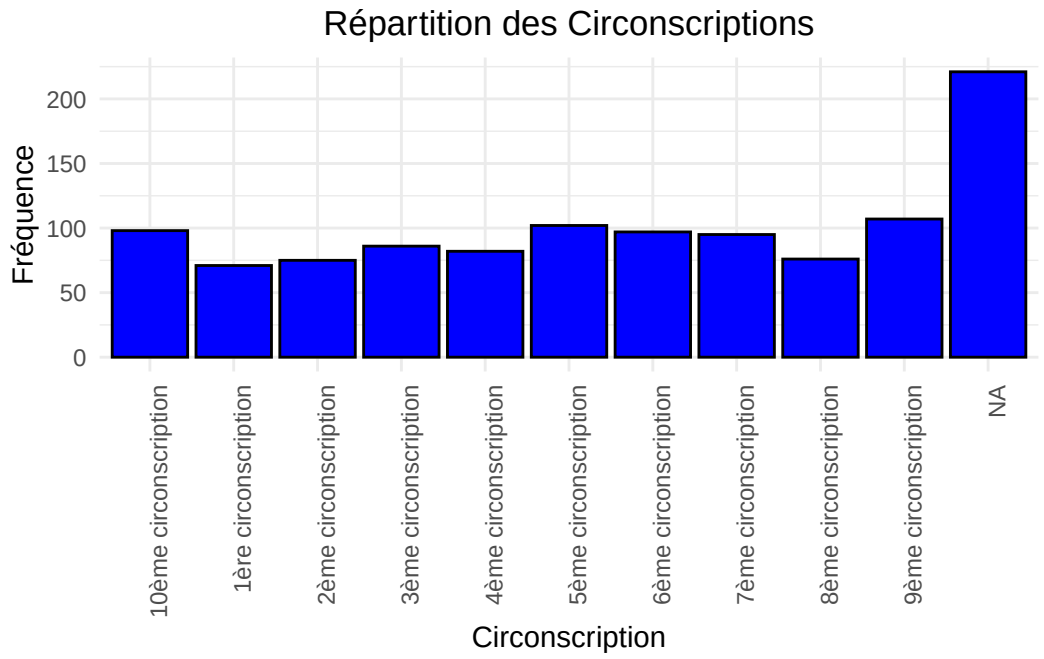
L'analyse descriptive des données fournit une vue d'ensemble des différentes variables et de leurs distributions, y compris leurs minimums, maximums, quartiles, médianes, et moyennes, ainsi que le nombre de valeurs manquantes (`NA`). Nous allons donc, pour les variables qualitatives, réaliser des graphiques en barres pour voir la répartition selon les différentes modalités et, pour les variables quantitatives, réaliser des histogrammes ainsi que des boxplots. Suite à cela, nous proposerons une interprétation des résultats selon chaque variable.

```
summary(data)
```

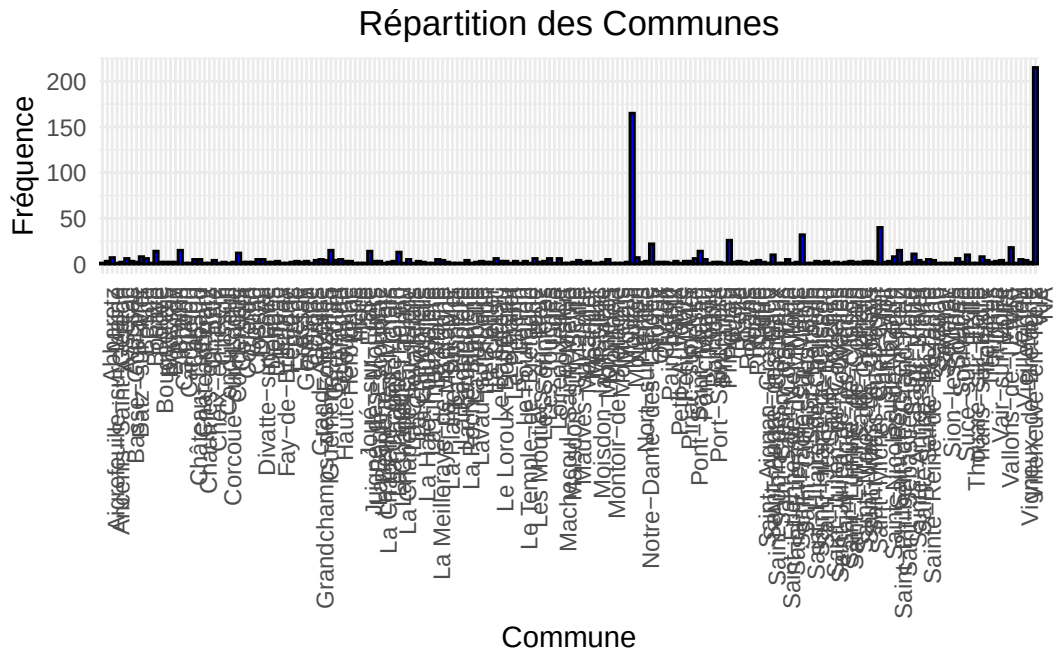
```
             circonscription         nom_commune  code_bur_vote
 9ème circonscription :107    Nantes        :165   1      :167
 5ème circonscription :102    Saint-Nazaire : 40   2      :123
 10ème circonscription: 98    Saint-Herblain: 32   3      : 84
 6ème circonscription : 97    Rezé          : 26   4      : 58
 7ème circonscription : 95    Orvault       : 22   5      : 43
 (Other)              :390    (Other)       :610   (Other):409
 NA's                 :221    NA's          :215   NA's   :226
    inscrits                tx_absents              tx_votants
 Min.   : 159.000000000   Min.   :13.2700000000   Min.   : 0.6300000000
 1st Qu.: 848.000000000   1st Qu.:19.3500000000   1st Qu.:76.0600000000
 Median : 956.000000000   Median :21.4100000000   Median :78.5800000000
 Mean   : 955.821634062   Mean   :22.4053872437   Mean   :77.5311797133
 3rd Qu.:1053.000000000   3rd Qu.:23.8275000000   3rd Qu.:80.6900000000
 Max.   :2041.000000000   Max.   :99.3700000000   Max.   :86.7300000000
 NA's   :241              NA's   :232             NA's   :203
   tx_blancs                tx_nuls                  tx_exprimes
 Min.   :0.00000000000   Min.   :  0.000000000000   Min.   : 0.0000000000
 1st Qu.:1.15000000000   1st Qu.:  0.290000000000   1st Qu.:97.3300000000
 Median :1.57000000000   Median :  0.510000000000   Median :97.8700000000
 Mean   :1.62165198238   Mean   :  0.672847533632   Mean   :97.7033333333
 3rd Qu.:2.02250000000   3rd Qu.:  0.762500000000   3rd Qu.:98.3500000000
 Max.   :4.83000000000   Max.   :100.000000000000   Max.   :99.6900000000
 NA's   :202             NA's   :218                NA's   :237
    latitude               longitude
 Min.   :46.9003706986   Min.   :-2.522778444220
 1st Qu.:47.2031024794   1st Qu.:-1.920282729080
 Median :47.2316356767   Median :-1.572556592760
 Mean   :47.2750339834   Mean   :-1.708395679230
 3rd Qu.:47.3313320913   3rd Qu.:-1.520222471100
```

```
 Max.   :47.8212049695   Max.   :-0.971865462189
 NA's   :217             NA's   :230
```

```r
# Bar plot pour la variable "circonscription"
ggplot(data, aes(x = circonscription)) +
  geom_bar(fill = "blue", color = "black") +
  labs(title = "Répartition des Circonscriptions",
       x = "Circonscription",
       y = "Fréquence") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

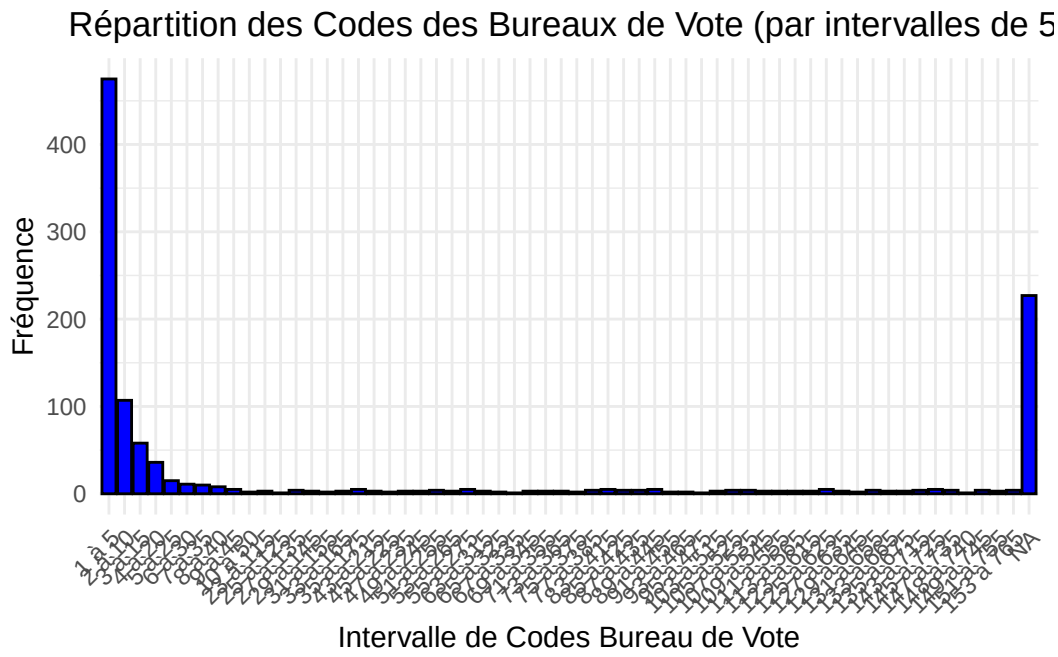



```r
# Bar plot pour la variable "nom_commune"
ggplot(data, aes(x = nom_commune)) +
  geom_bar(fill = "blue", color = "black") +
  labs(title = "Répartition des Communes",
       x = "Commune",
       y = "Fréquence") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

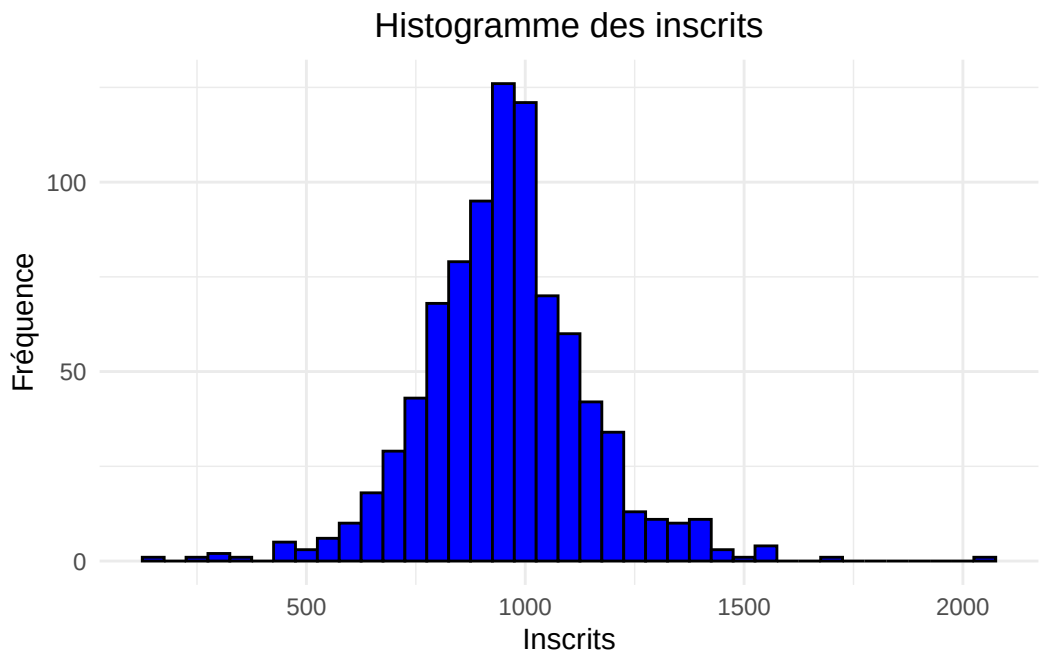



```
# Bar plot pour la variable "code_bur_vote"

## Remplacer les valeurs NA par une catégorie "bureau NA" dans la colonne `code_bur_vote`
data_mod <- data |>
  mutate(code_bur_vote = ifelse(is.na(code_bur_vote), "bureau NA",
                                as.character(code_bur_vote)))

## Créer la variable 'group_bur_vote' avec les intervalles de 5 manquantes
data_mod <- data_mod |>
  mutate(group_bur_vote = cut(as.numeric(code_bur_vote),
                              breaks = seq(0,
                                           max(as.numeric(code_bur_vote),
                                               na.rm = TRUE), by = 5),
                              include.lowest = TRUE,
                              labels = paste(
                                seq(1, floor(
                                  max(as.numeric(code_bur_vote),
                                      na.rm = TRUE) / 5)),
                                "à", seq(5,
                                         floor(max(as.numeric(code_bur_vote),
                                                   na.rm = TRUE) / 5) * 5,
                                         by = 5), sep = " ")))


data_mod$group_bur_vote[is.na(data_mod$code_bur_vote)] <- "bureau NA"

## Visualisation avec ggplot
ggplot(data_mod, aes(x = group_bur_vote)) +
```

```
  geom_bar(fill = "blue", color = "black") +
  labs(title =
         "Répartition des Codes des Bureaux de Vote (par intervalles de 5)",
       x = "Intervalle de Codes Bureau de Vote", y = "Fréquence") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

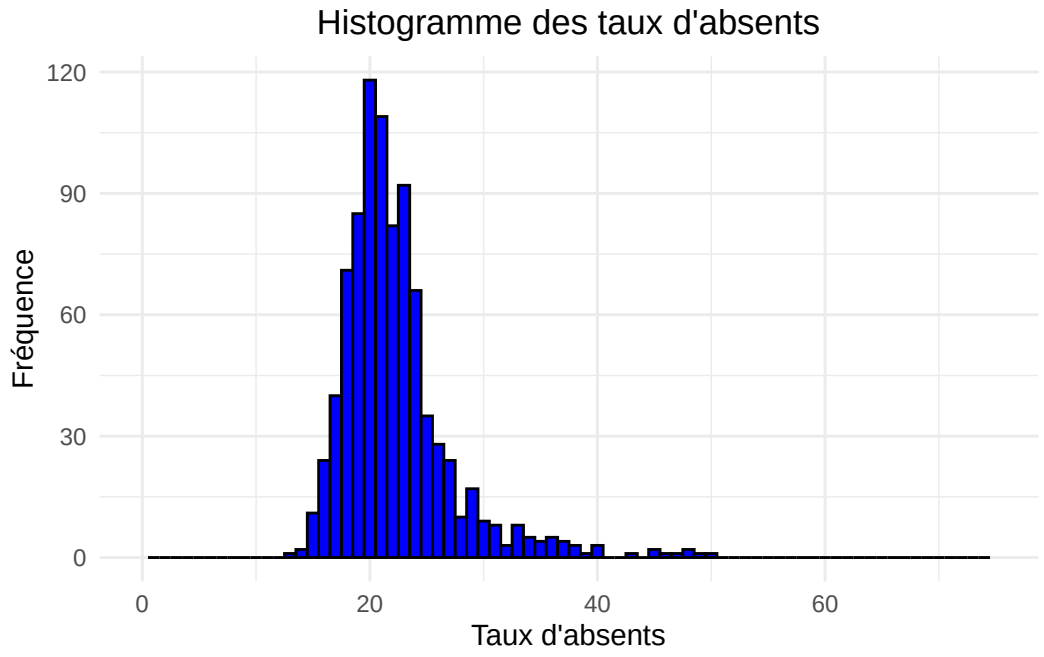



```
# Histogrammes et Boxplots pour les variables liées aux inscrits
# Histogramme des inscrits
ggplot(data, aes(x=inscrits)) +
  geom_histogram(binwidth = 50, fill="blue", color="black") +
  labs(title="Histogramme des inscrits", x="Inscrits", y="Fréquence") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



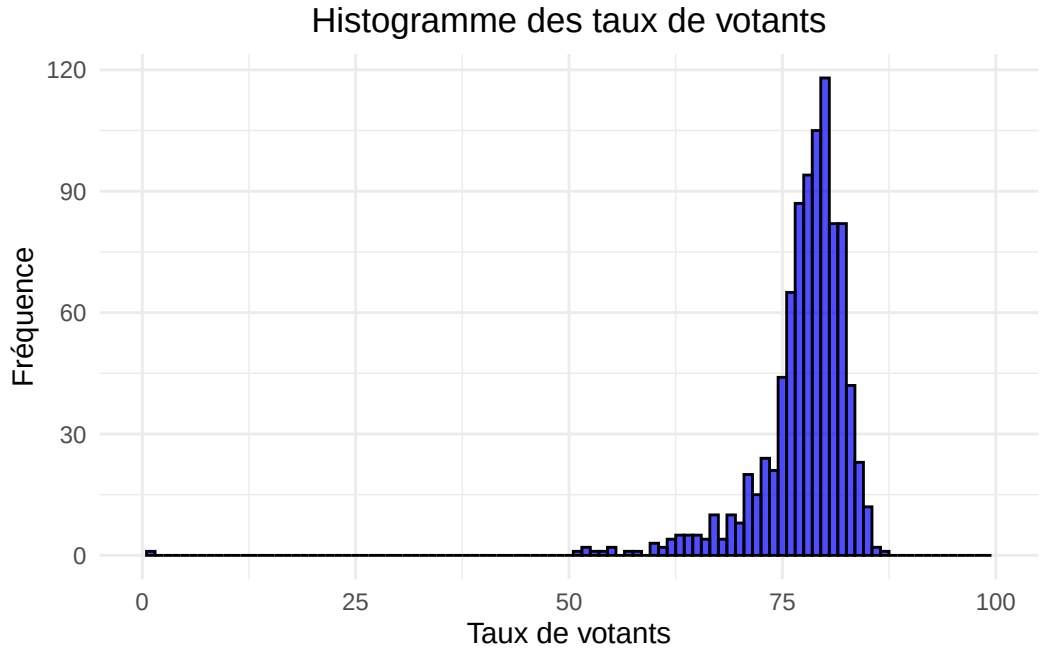

```r
# Boxplot pour le nombre d'inscrits
ggplot(data, aes(x = "", y = inscrits)) +
  geom_boxplot(fill="blue", color="black") +
  labs(title="Boîte à moustaches des inscrits", y="Nombre d'inscrits") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

```r
# Histogrammes et Boxplots pour les variables liées aux taux d'absents
# Histogramme des taux d'absents
ggplot(data, aes(x=tx_absents)) +
  geom_histogram(binwidth = 1, fill="blue", color="black") +
  labs(title="Histogramme des taux d'absents", x="Taux d'absents", y="Fréquence") +
  xlim(0, 75) +  # Limiter les valeurs de l'axe X
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

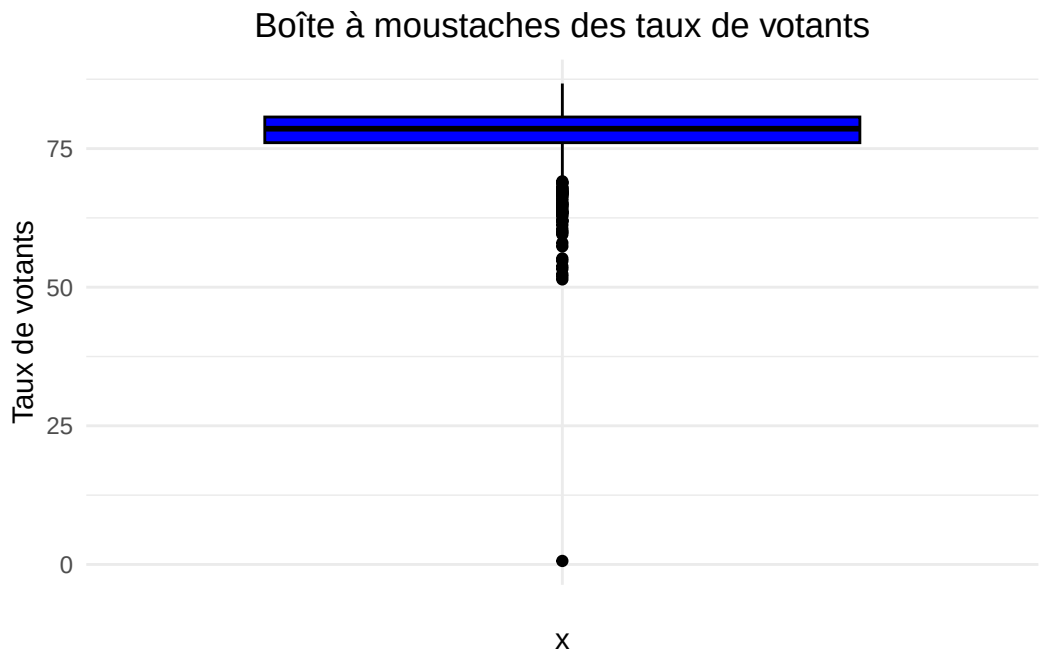



```r
# Boxplot pour le taux d'abstention
ggplot(data, aes(x = "", y = tx_absents)) +
  geom_boxplot(fill="blue", color="black") +
  labs(title="Boîte à moustaches des taux d'abstention", y="Taux d'abstention") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

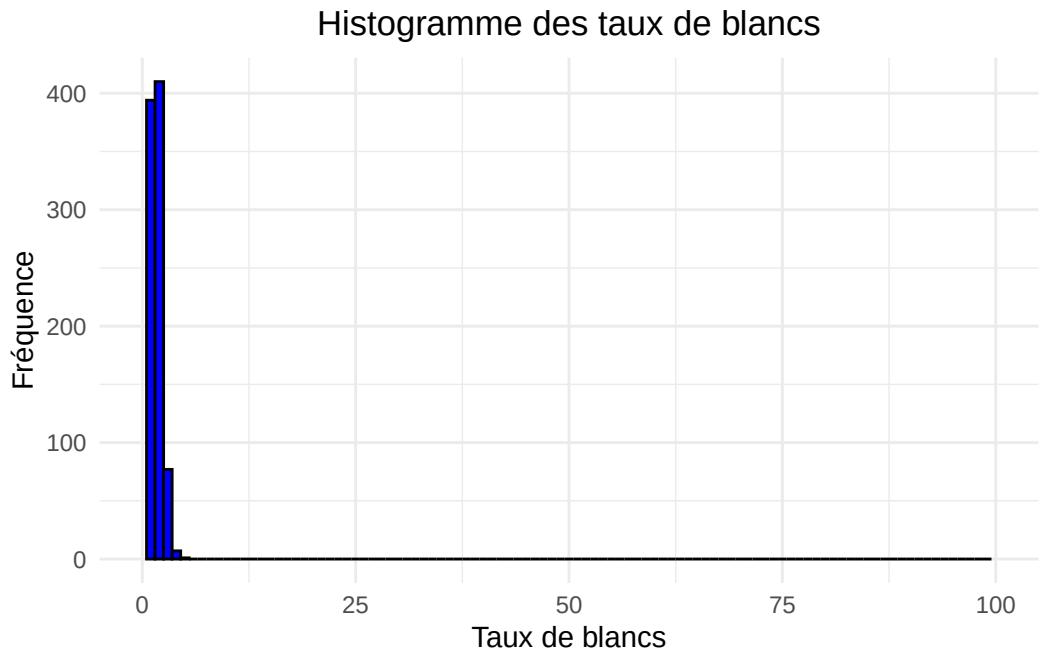

```
# Histogrammes et Boxplots pour les variables liées aux taux de votants
# Histogramme des taux de votants
ggplot(data, aes(x=tx_votants)) +
  geom_histogram(binwidth = 1, fill="blue", color="black", alpha=0.7) +
  labs(title="Histogramme des taux de votants", x="Taux de votants", y="Fréquence") +
  xlim(0, 100) +  # Limiter les valeurs de l'axe X
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



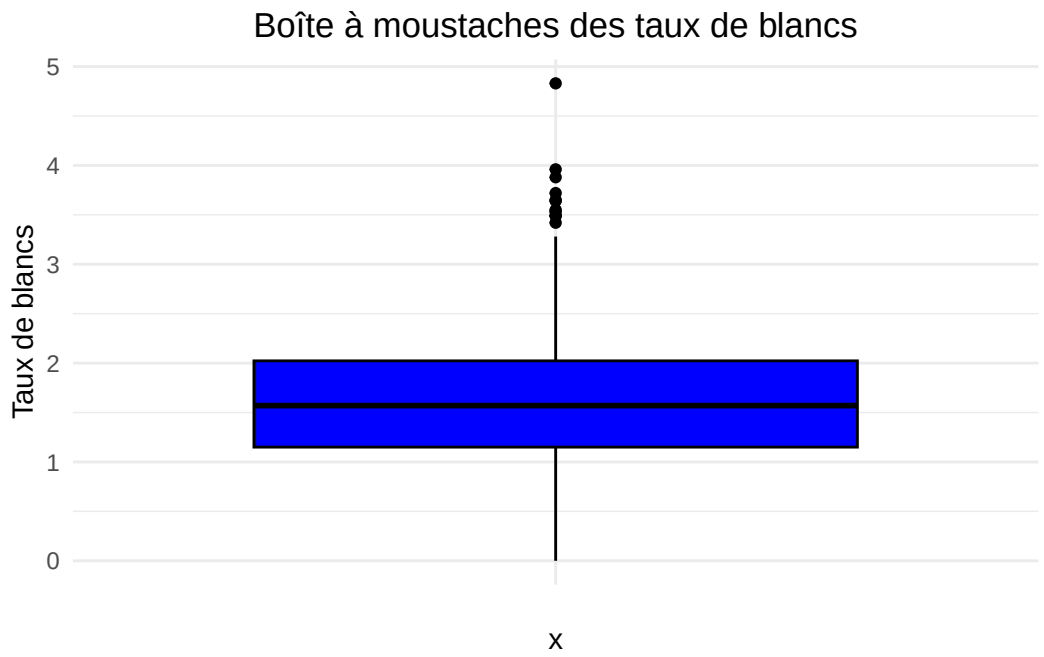
```r
# Boxplot pour le taux de votants
ggplot(data, aes(x = "", y = tx_votants)) +
  geom_boxplot(fill="blue", color="black") +
  labs(title="Boîte à moustaches des taux de votants", y="Taux de votants") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



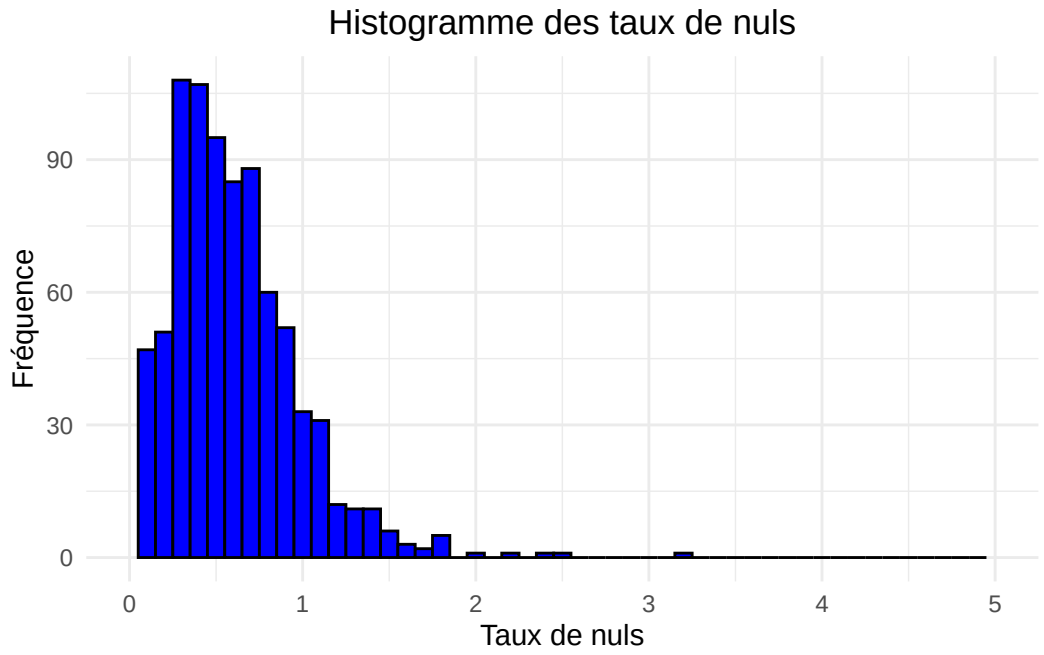

```r
# Histogrammes et Boxplots pour les variables liées aux taux de blancs
# Histogramme des taux de blancs
ggplot(data, aes(x=tx_blancs)) +
  geom_histogram(binwidth = 1, fill="blue", color="black") +
  labs(title="Histogramme des taux de blancs", x="Taux de blancs", y="Fréquence") +
  xlim(0, 100) +  # Limiter les valeurs de l'axe X
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```




```r
# Boxplot pour le taux de blancs
ggplot(data, aes(x = "", y = tx_blancs)) +
  geom_boxplot(fill="blue", color="black") +
  labs(title="Boîte à moustaches des taux de blancs", y="Taux de blancs") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
# Histogrammes et Boxplots pour les variables liées aux taux de nuls
# Histogramme des taux de nuls
ggplot(data, aes(x=tx_nuls)) +
  geom_histogram(binwidth = 0.1, fill="blue", color="black") +
  labs(title="Histogramme des taux de nuls", x="Taux de nuls", y="Fréquence") +
  xlim(0, 5) +  # Limiter les valeurs de l'axe X
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

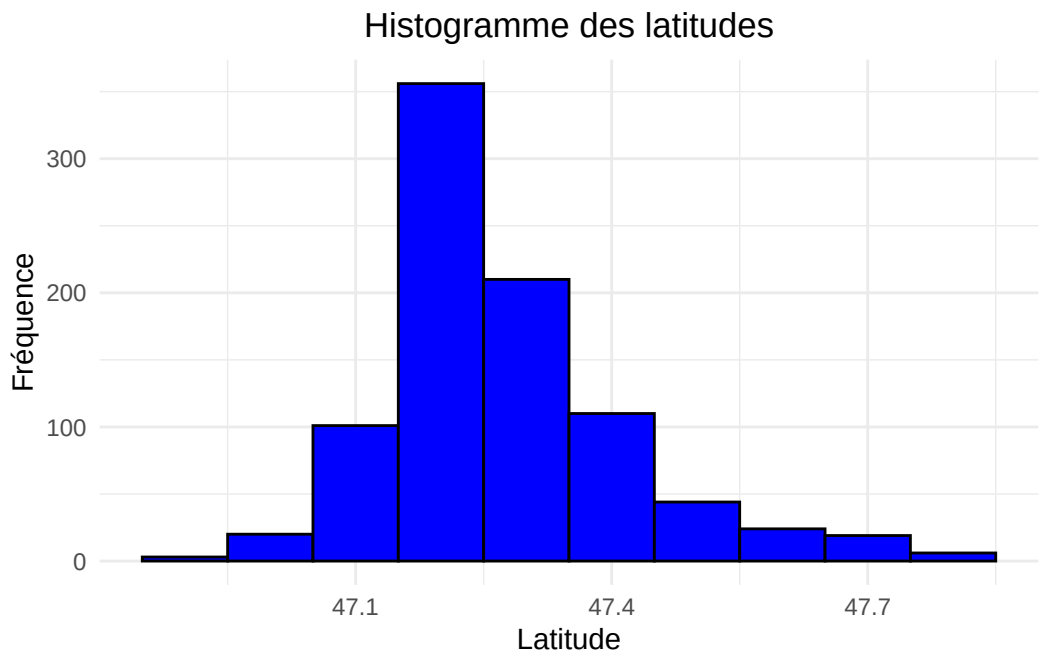



```r
# Boxplot pour le taux de nuls
ggplot(data, aes(x = "", y = tx_nuls)) +
  geom_boxplot(fill="blue", color="black") +
  labs(title="Boîte à moustaches des taux de nuls", y="Taux de nuls") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

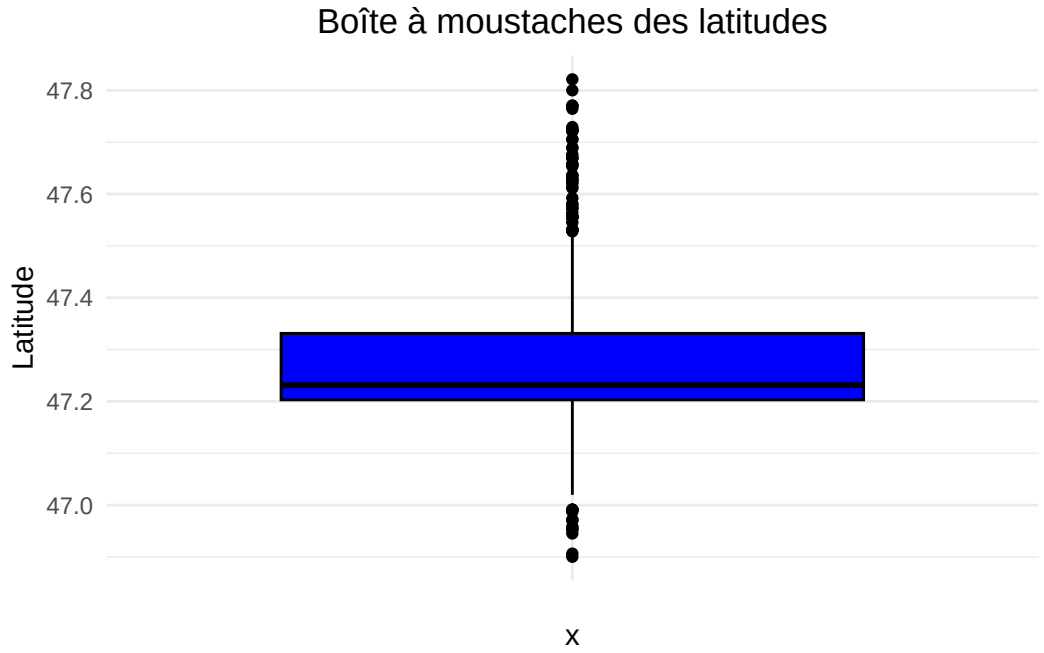

```r
# Histogrammes et Boxplots pour les variables liées aux taux exprimés
# Histogramme des taux exprimés
ggplot(data, aes(x=tx_exprimes)) +
  geom_histogram(binwidth = 1, fill="blue", color="black") +
  labs(title="Histogramme des taux exprimés", x="Taux exprimés", y="Fréquence") +
  xlim(0, 100) +  # Limiter les valeurs de l'axe X
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

```r
# Boxplot pour le taux exprimé
ggplot(data, aes(x = "", y = tx_exprimes)) +
  geom_boxplot(fill="blue", color="black") +
  labs(title="Boîte à moustaches des taux exprimés", y="Taux exprimés") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

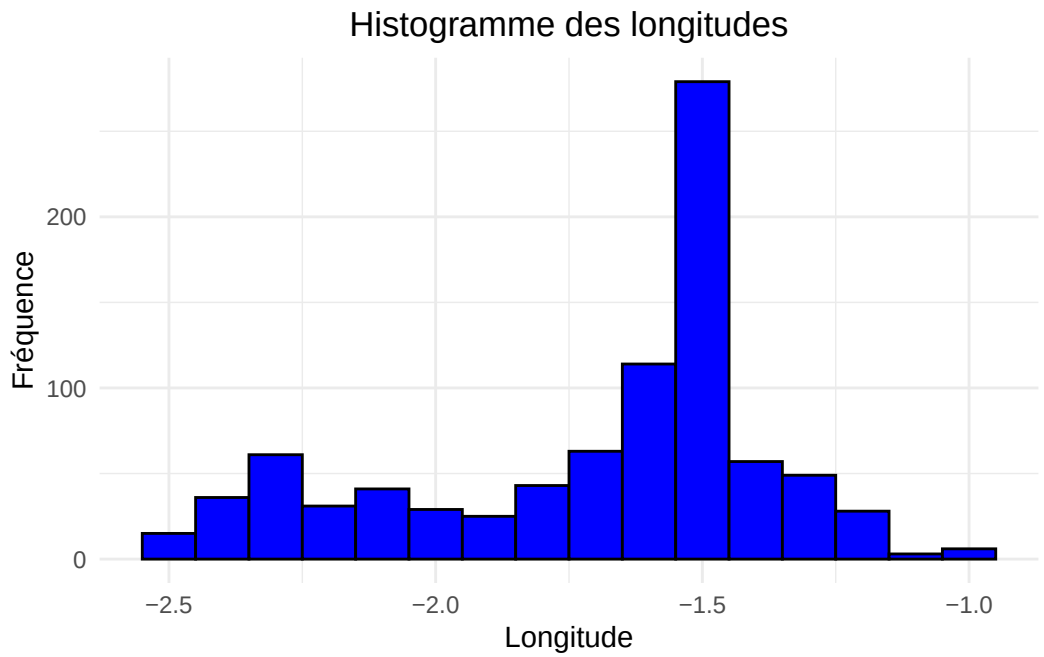

```r
# Histogrammes et Boxplots pour les variables géographiques
# Histogramme des latitudes
ggplot(data, aes(x=latitude)) +
  geom_histogram(binwidth = 0.1, fill="blue", color="black") +
  labs(title="Histogramme des latitudes", x="Latitude", y="Fréquence") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



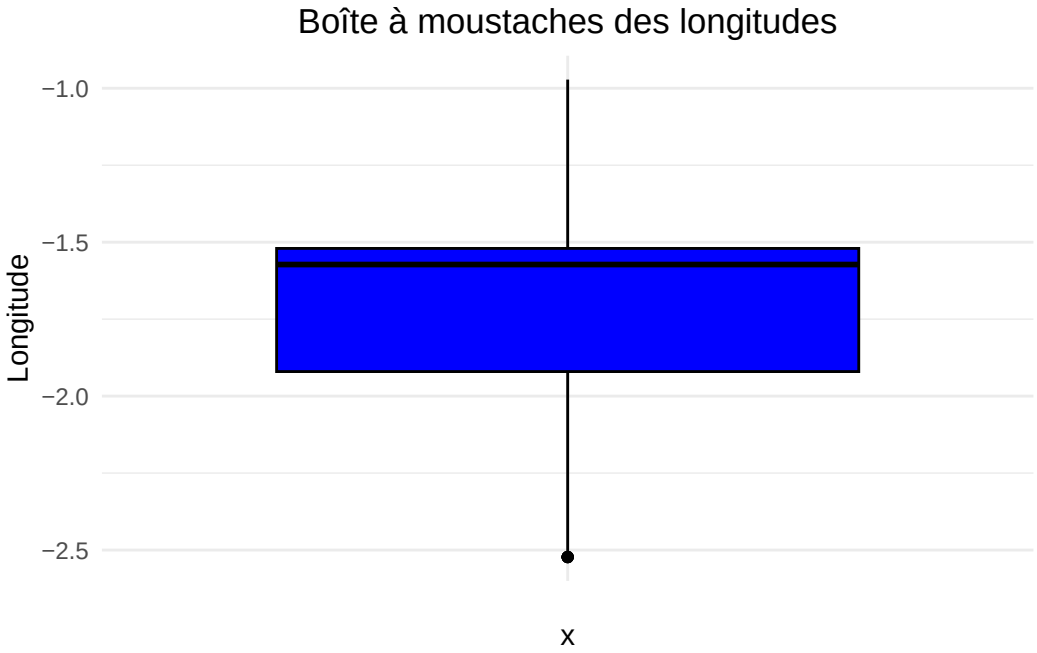

```
# Boxplot pour la latitude
ggplot(data, aes(x = "", y = latitude)) +
  geom_boxplot(fill="blue", color="black") +
  labs(title="Boîte à moustaches des latitudes", y="Latitude") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
# Histogrammes et Boxplots pour les longitudes
# Histogramme des longitudes
ggplot(data, aes(x=longitude)) +
  geom_histogram(binwidth = 0.1, fill="blue", color="black") +
  labs(title="Histogramme des longitudes", x="Longitude", y="Fréquence") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```




```r
# Boxplot pour la longitude
ggplot(data, aes(x = "", y = longitude)) +
  geom_boxplot(fill="blue", color="black") +
  labs(title="Boîte à moustaches des longitudes", y="Longitude") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



Les barplots des variables `nom_commune` et `code_bur_vote` ne sont pas très lisibles en raison du grand nombre de modalités présentes. En effet, chaque commune ou code de bureau de vote correspond à une modalité unique, ce qui entraîne une grande dispersion des données dans le graphique. Cependant, ces visualisations ont pour objectif principal de montrer que certaines communes comptent un plus grand nombre de bureaux de vote que d'autres. Cela est parfaitement logique, car les communes plus grandes ou plus peuplées nécessitent davantage de bureaux de vote pour couvrir l'ensemble de la population électorale. De plus, certains bureaux de vote apparaissent plusieurs fois, ce qui peut également affecter la clarté des barplots, mais reflète simplement le fait que certains bureaux sont associés à plusieurs circonscriptions ou que les codes de bureau sont répartis sur plusieurs zones.

Maintenant que nous avons dit cela et représenté les variables, interprétons les résultats.

1. Variables qualitatives

`circonscription` : La variable circonscription regroupe les données selon les différentes circonscriptions électorales. Il y a 10 circonscriptions au total. Parmi celles-ci, la "9ème circonscription" est la plus représentée, avec 107 observations. En revanche, certaines circonscriptions ont moins d'observations, ce qui peut entraîner une répartition inégale des données. De plus, 221 valeurs manquantes sont présentes, ce qui suggère qu'un nombre significatif de lignes n'ont pas d'information sur la circonscription, rendant difficile l'analyse géographique ou électorale précise de ces données.

`nom_commune` : La variable nom_commune contient les noms de 192 communes différentes. Parmi celles-ci, Nantes est la plus représentée avec 165 occurrences, suivie de Saint-Nazaire avec 40 et Saint-Herblain avec 32. Ces communes ont donc un poids beaucoup plus important que les autres, qui sont dispersées dans l'ensemble. En revanche, 215 valeurs manquent pour cette variable, ce qui peut affecter la précision des analyses concernant les communes. Cela signifie qu'une proportion notable des données n'est pas géolocalisée par commune, ce qui peut limiter les analyses géographiques.

`code_bur_vote` : Le code_bur_vote regroupe les bureaux de vote par leurs codes. Le code "1" apparaît le plus fréquemment avec 167 occurrences, suivi par les codes "2" et "3" avec respectivement 123 et 84 occurrences. Les autres codes sont beaucoup moins représentés. Cependant, il y a 226 valeurs manquantes, ce qui limite la possibilité d'étudier la répartition des bureaux de vote par code et d'analyser les patterns associés à ces codes.

2. Variables quantitatives

`inscrits` : La variable inscrits représente le nombre d'inscrits dans chaque bureau de vote ou circonscription. Le nombre d'inscrits varie de 159 à 2041, avec une moyenne de 955.82 inscrits. Les quartiles sont les suivants :

- 1er quartile : 848 inscrits
- Médiane : 955 inscrits
- 3e quartile : 1053 inscrits

Cela signifie qu'environ 75% des bureaux de vote ou circonscriptions ont un nombre d'inscrits compris entre 848 et 1053. Les valeurs extrêmes (159 et 2041) suggèrent qu'il existe des bureaux ou circonscriptions avec des chiffres d'inscrits particulièrement faibles ou élevés. De plus, 241 valeurs sont manquantes, ce qui pourrait perturber l'analyse de la distribution complète des inscrits.

`tx_absents` : Le taux d'absentéisme varie entre 13.27% et 99.37%, avec une moyenne de 22.41% et une médiane de 21.41%. Cependant, les quartiles montrent une concentration autour de taux d'absentéisme plus faibles :

- 1er quartile : 17.85%
- Médiane : 21.41%
- 3e quartile : 23.43%

Cela suggère que, bien que le taux d'absentéisme puisse atteindre des valeurs très élevées (jusqu'à 99.37%), les 75% des bureaux de vote ont un taux d'absentéisme inférieur à 23.43%. Les valeurs extrêmes peuvent refléter des erreurs de saisie ou des anomalies dans les données. 232 valeurs manquantes signalent aussi un manque d'information sur cette variable pour certains bureaux de vote ou circonscriptions.

`tx_votants` : Le taux de votants varie de 0.63% à 86.73%, avec une moyenne de 77.53% et une médiane de 78.58%. Les quartiles sont :

- 1er quartile : 72.43%
- Médiane : 78.58%
- 3e quartile : 83.23%

Cela montre que, dans la majorité des bureaux de vote, le taux de participation est élevé. En effet, 75% des bureaux de vote ont un taux de votants compris entre 72.43% et 83.23%. Les valeurs très faibles (jusqu'à 0.63%) pourraient correspondre à des bureaux avec des erreurs de saisie ou des anomalies spécifiques. Il y a aussi 203 valeurs manquantes pour cette variable.

`tx_blancs` : Le taux de votes blancs varie de 0% à 4.83%, avec une moyenne de 1.62% et une médiane de 1.57%. Les quartiles sont :

- 1er quartile : 0.90%
- Médiane : 1.57%
- 3e quartile : 2.27%

Cela indique que la plupart des bureaux de vote ont un faible taux de votes blancs, avec 75% des bureaux ayant un taux inférieur à 2.27%. Les valeurs maximales autour de 4.83% représentent probablement des bureaux où une proportion significative d'électeurs ont choisi de voter blanc. Les 202 valeurs manquantes peuvent limiter l'analyse de cette variable.

`tx_nuls` : Le taux de votes nuls varie de 0% à 100%, avec une moyenne de 0.67% et une médiane de 0.51%. Les quartiles sont :

- 1er quartile : 0.33%
- Médiane : 0.51%
- 3e quartile : 0.83%

Cela suggère que 75% des bureaux de vote ont un taux de votes nuls inférieur à 0.83%. Les valeurs extrêmes, notamment le taux de 100%, indiquent des valeurs atypiques dans certains bureaux de vote. Les 218 valeurs manquantes peuvent nuire à l'analyse de cette variable.

`tx_exprimes` : Le taux de votes exprimés, représentant les votes valides, varie de 0% à 99.69%, avec une moyenne de 97.7% et une médiane de 97.87%. Les quartiles sont :

- 1er quartile : 96.75%
- Médiane : 97.87%
- 3e quartile : 98.92%

Cela montre qu'une majorité des votes sont exprimés de manière valide. 75% des bureaux de vote ont un taux de votes exprimés compris entre 96.75% et 98.92%. Les valeurs faibles (jusqu'à 0%) peuvent signaler des problèmes avec le processus de vote dans certains bureaux. 237 valeurs manquantes sont à prendre en compte pour cette variable.

3. Variables géographiques

`latitude` : Les latitudes des communes varient entre 46.90° et 47.82°, avec une moyenne de 47.28°. Les quartiles sont :

- 1er quartile : 47.07°
- Médiane : 47.28°
- 3e quartile : 47.49°

Cela montre que la plupart des communes se situent autour de 47.28°, principalement dans la zone centrale du département. 217 valeurs manquantes peuvent affecter l'analyse géographique, surtout pour les cartes ou les analyses de répartition spatiale.

`longitude` : Les longitudes varient entre -2.52 et -0.97, avec une moyenne de -1.71. Les quartiles sont :

- 1er quartile : -1.87
- Médiane : -1.57
- 3e quartile : -1.33

Cela place la majorité des communes dans l'ouest du département 230 valeurs manquantes limitent également l'analyse géographique des communes, car elles affectent la précision des coordonnées géographiques disponibles.

Au final, l'analyse univariée des variables qualitatives montre une répartition inégale des données. Bien que des valeurs manquantes soient présentes pour la circonscription, la commune et le code du bureau de vote, cela fait partie de la nature des données et sera un aspect important de notre travail pour les compléter et les corriger. Ces manques n'affectent pas immédiatement l'analyse, mais ils nécessiteront une attention particulière pour éviter de fausser les résultats des analyses géographiques et administratives.

Concernant les variables quantitatives, nous observons une variabilité importante des taux de participation et d'abstention. Les taux d'absentéisme sont généralement modérés, bien que certains bureaux de vote présentent des taux très élevés. Cependant, ces valeurs extrêmes sont rares et ne représentent qu'une faible proportion des données. Les taux de votants, de votes blancs et de votes exprimés montrent une tendance plus stable dans la majorité des bureaux de vote, ce qui reflète un comportement électoral globalement homogène, malgré quelques valeurs atypiques De plus, les inscriptions varient considérablement entre les bureaux de vote, ce qui indique une concentration différente des électeurs selon les circonscriptions.

En résumé, bien que des données manquantes existent, cela fait partie intégrante du travail à réaliser. Leur traitement et leur complétion permettront de renforcer la fiabilité des analyses à venir. Nous devons donc nous concentrer sur l'identification et la gestion de ces valeurs manquantes pour garantir des résultats aussi précis que possible.

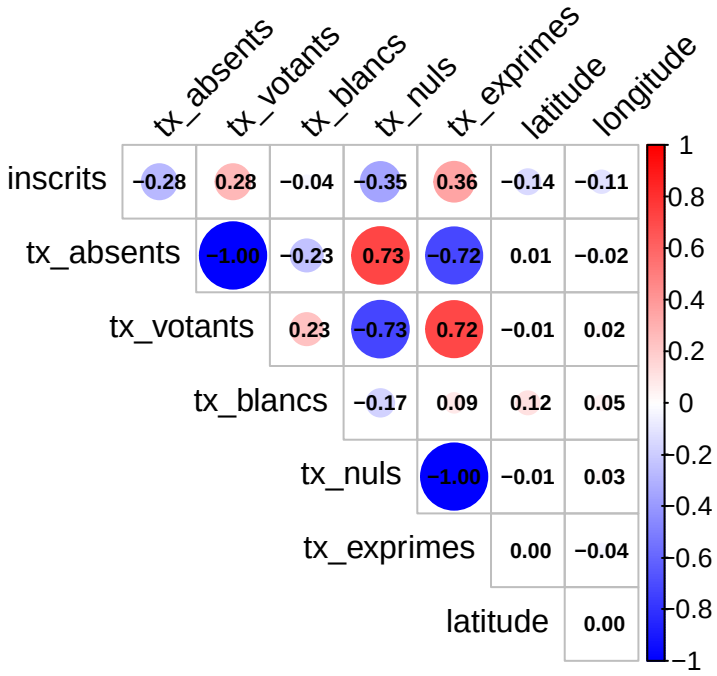
3.2 Analyse bivariée

3.2.1 Analyse bivariée pour deux variables quantitatives

Nous allons maintenant aborder l'analyse bivariée des variables quantitatives. Cette étape est essentielle pour explorer les relations entre deux variables numériques. En comparant ces variables, nous pouvons identifier des corrélations, des tendances, ou des patterns qui pourraient exister entre elles. Cela nous permettra de mieux comprendre comment une variable peut influencer ou être liée à une autre, ce qui est crucial pour affiner nos hypothèses et guider la modification des valeurs manquantes dans notre étude.

```
data_quanti <- data |>
  select_if(is.numeric)

cor_data <- cor(data_quanti,use="complete.obs",method = c("pearson"))

corrplot(cor_data,
         method = "circle",
```

```
        type = "upper",
        col = colorRampPalette(c("blue", "white", "red"))(200),
        tl.col = "black",
        tl.srt = 45,
        addCoef.col = "black",
        number.cex = 0.7,
        diag = FALSE,
)
```



La matrice de corrélation met en lumière les relations entre les différentes variables quantitatives. Tout d'abord, nous observons une corrélation parfaite de -1 entre les taux d'abstention (`tx_absents`) et les taux de votants (`tx_votants`). Cela est parfaitement logique, car ces deux variables sont complémentaires : une augmentation de l'abstention entraîne une diminution directe de la participation, et vice-versa.

Ensuite, une corrélation positive notable de 0.73 est observée entre le taux d'abstention (`tx_absents`) et le taux de bulletins nuls (`tx_nuls`). Cela pourrait indiquer que dans les zones où l'abstention est élevée, il y a également une tendance accrue à déposer des bulletins nuls, probablement liée à un désintérêt global pour le processus électoral ou à une méfiance vis-à-vis des choix proposés.

Les corrélations entre le taux de votes exprimés (`tx_exprimes`) et les taux de votants (`tx_votants`) ou de bulletins blancs (`tx_blancs`) sont également intéressantes. Une forte corrélation positive (0.72) entre `tx_exprimes` et `tx_votants` montre qu'une majorité des votes exprimés sont valides, ce qui est logique dans des contextes électoraux bien encadrés. En revanche, la corrélation modérée (0.08) entre `tx_exprimes` et `tx_blancs` reflète un faible lien direct entre les bulletins blancs et le total des votes exprimés.

Le nombre d'inscrits (`inscrits`) présente également des relations pertinentes. Par exemple, une corrélation négative modérée de -0.28 avec le taux d'abstention (`tx_absents`) indique que les communes avec un grand nombre d'inscrits tendent à avoir des taux d'abstention légèrement plus bas. Cela peut s'expliquer par une mobilisation électorale plus importante dans les zones densément peuplées. Par ailleurs, la corrélation positive de 0.36 entre le nombre d'inscrits et le taux de votes exprimés (`tx_exprimes`) est également logique : plus il y a d'inscrits, plus le nombre absolu de votes exprimés a tendance à augmenter.

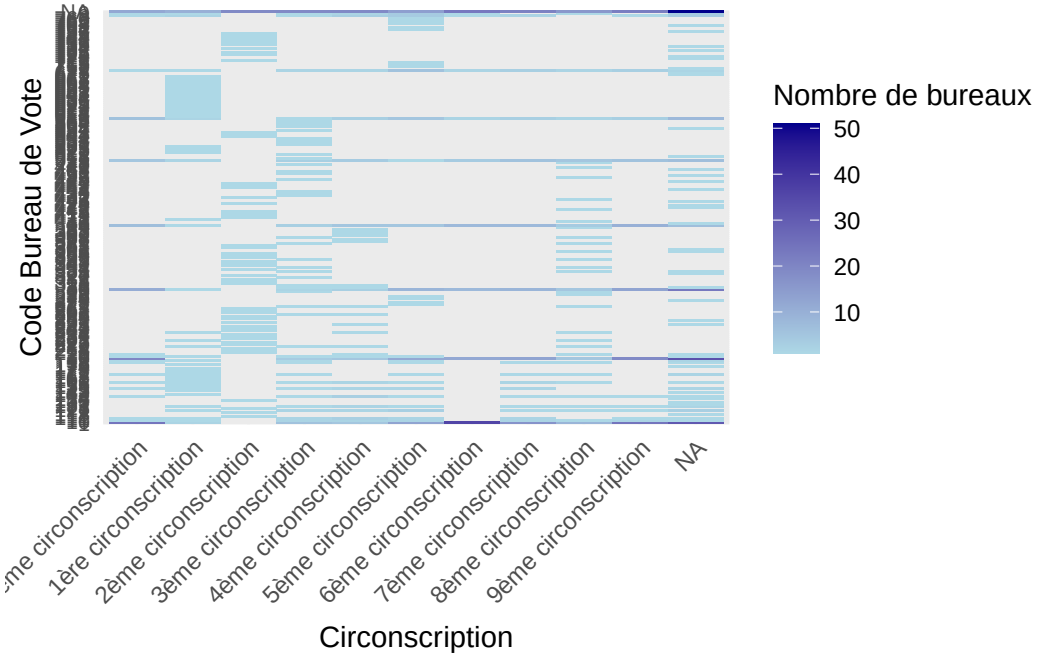
Enfin, les relations entre les variables géographiques (`latitude`, `longitude`) et les variables électorales sont faibles (proches de 0), ce qui indique que les comportements électoraux ne sont pas fortement influencés par la position géographique des communes dans notre jeu de données. Par exemple, les taux d'abstention, de votes blancs ou nuls ne semblent pas varier de manière notable en fonction des coordonnées géographiques.

En conclusion, cette analyse met en évidence des relations cohérentes avec les dynamiques électorales attendues. Les liens entre l'abstention, les bulletins nuls et exprimés traduisent des comportements variés selon les zones, tandis que les variables géographiques n'ont pas d'impact significatif sur les résultats.

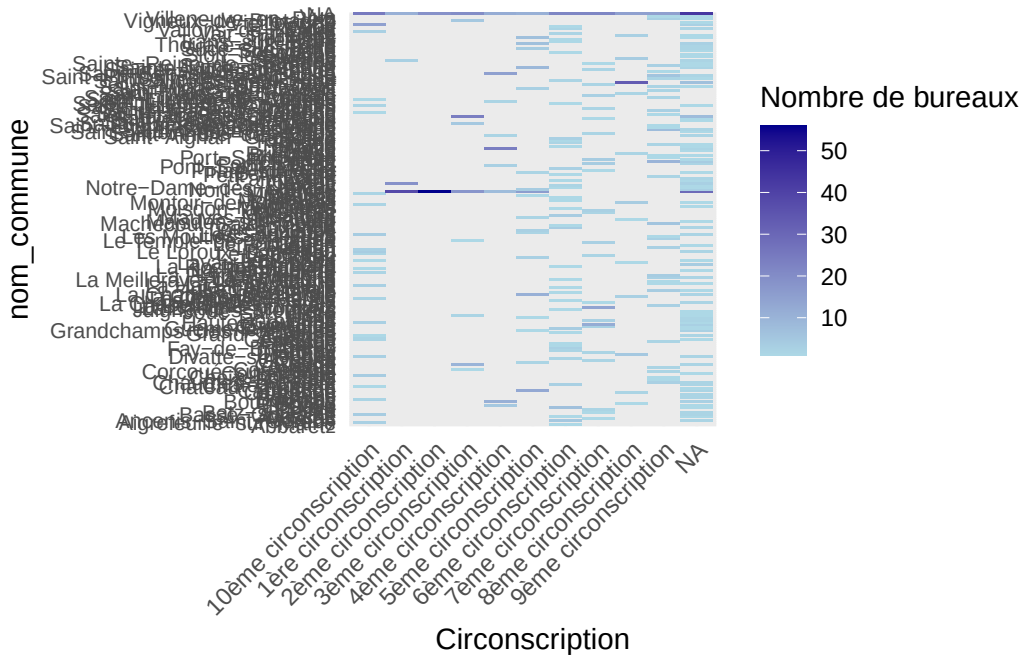
3.2.2 Analyse bivariée pour deux variables qualitatives

Nous allons maintenant passer à l'analyse bivariée des variables qualitatives. Cette analyse nous permet d'examiner les relations entre deux variables catégorielles, afin d'identifier des associations ou des tendances dans les données. Par exemple, en croisant des variables telles que la circonscription et le code de bureau de vote, nous pouvons observer si certaines catégories sont plus fréquentes ou ont des comportements similaires. Cette étape est importante pour comprendre comment les différentes modalités des variables qualitatives interagissent entre elles.

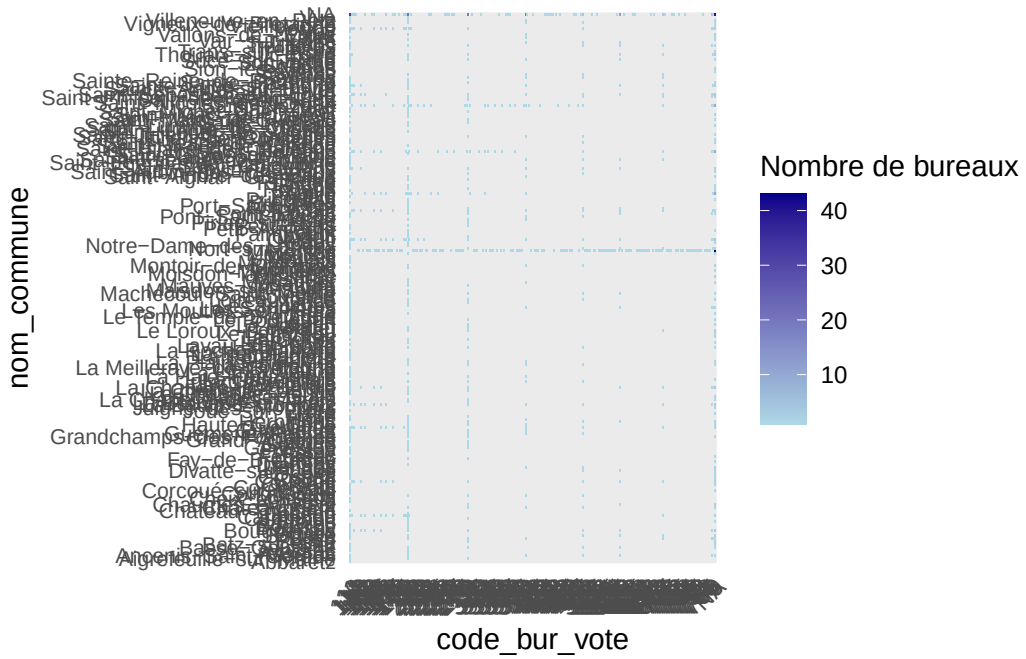
```
# Circonscription & bureau de votes

data |>
  count(circonscription, code_bur_vote) |>
  ggplot() +
  aes(x = circonscription, y = code_bur_vote, fill = n) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(x = "Circonscription",
       y = "Code Bureau de Vote",
       fill = "Nombre de bureaux") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    axis.text.y = element_text(size = 8),
    plot.title = element_text(hjust = 0.5)
  )
```




```
# Circonscription & noms de commune

data |>
  count(circonscription, nom_commune) |>
  ggplot() +
  aes(x = circonscription, y = nom_commune, fill = n) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(x = "Circonscription",
       y = "nom_commune",
       fill = "Nombre de bureaux") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    axis.text.y = element_text(size = 8),
    plot.title = element_text(hjust = 0.5)
  )
```




```
# Nom de communes & bureau de votes

data |>
  count(code_bur_vote, nom_commune) |>
  ggplot() +
  aes(x = code_bur_vote, y = nom_commune, fill = n) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(x = "code_bur_vote",
       y = "nom_commune",
       fill = "Nombre de bureaux") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    axis.text.y = element_text(size = 8),
    plot.title = element_text(hjust = 0.5)
  )
```



Les graphiques ci-dessus sont des visualisations de type heatmap qui montrent les relations entre différentes paires de variables qualitatives.

Le premier graphique croise la circonscription avec le code de bureau de vote. Il nous permet d'observer la distribution des bureaux de vote dans les différentes circonscriptions et de repérer les zones avec un grand nombre de bureaux.

Le deuxième graphique montre la relation entre la circonscription et le nom de commune, ce qui permet d'identifier les communes les plus présentes dans chaque circonscription (en se basant sur le nombre de bureaux de vote).

Le dernier graphique croise le code de bureau de vote avec le nom de commune, montrant comment les bureaux de vote se répartissent dans les différentes communes.

Toutefois, ces graphiques ne sont pas très lisibles car les variables commune et code de bureau de vote comportent trop de modalités. Cela crée une surcharge d'informations, rendant difficile une interprétation claire. Néanmoins, ces visualisations permettent d'identifier des tendances générales, comme les communes ayant plus de bureaux de vote ou certaines circonscriptions ayant plus de diversité en termes de bureaux, ce qui traduit des communes plus grandes.

3.2.3 Analyse bivariée pour une variable quantitative et une qualitative

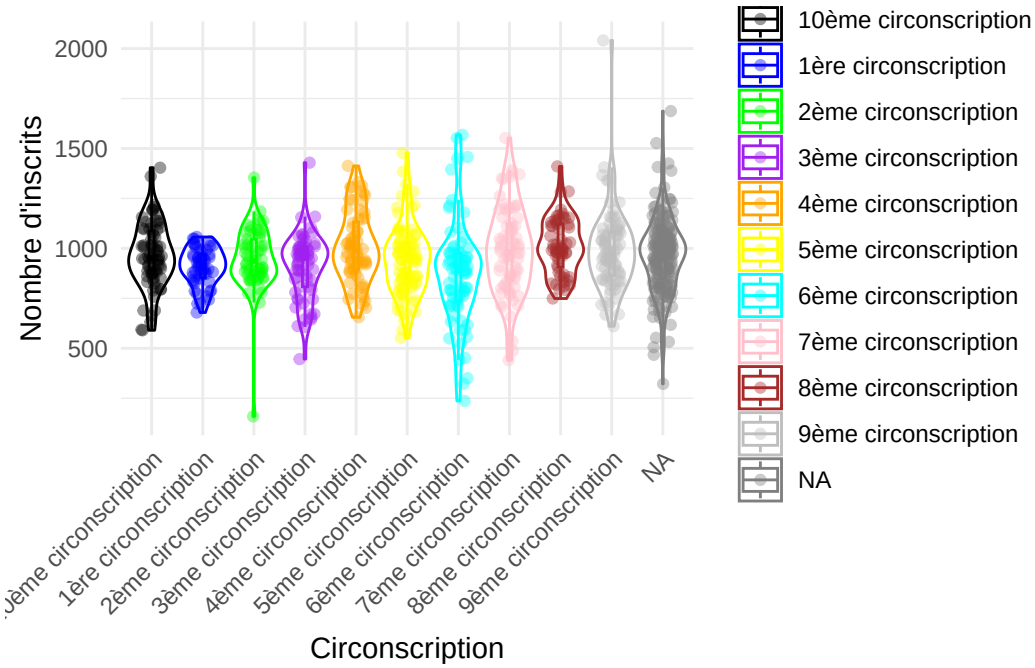
Maintenant, passons à l'analyse bivariée quali-quanti, qui consiste à étudier la relation entre une variable qualitative (catégorique) et une variable quantitative. L'objectif est d'observer comment les différentes catégories d'une variable qualitative peuvent influencer ou être associées à une variable quantitative.

Par exemple, nous pouvons analyser comment des variables comme le taux de votants ou le taux d'absents varient selon les circonscriptions électorales ou les communes. Cela nous permet de mieux comprendre si, par exemple, certaines circonscriptions ou communes ont des comportements

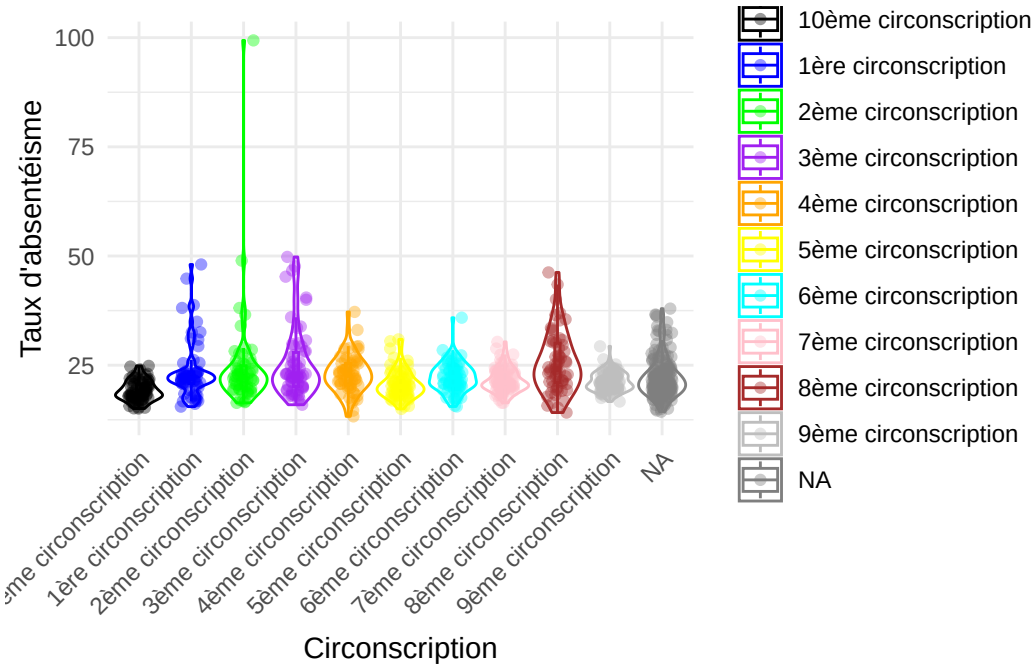
électoraux différents, ou si des facteurs géographiques influencent la participation ou l'abstention des électeurs.

Cette analyse permet d'identifier des tendances ou des différences importantes entre les groupes, ce qui est essentiel pour mieux interpréter les données en fonction de critères géographiques ou sociaux. Toutefois, là encore, nous allons avoir un problème avec le nombre de valeurs que peuvent prendre `nom_commune` et `code_bur_vote`. Bien que le fait d'habiter dans une commune plutôt que dans une autre a sûrement une influence dans la façon de voter, nous n'allons pas représenter cela ici car ça serait illisble. De la même façon, nous allons omettre la variable qualitative `code_bur_vote`. Il est moins grave d'ignorer celle-ci car il n'existe pas de lien direct entre le code d'un bureau de vote et la façon de voter. De ce fait, nous allons uniquement nous intéresser à la variable qualitative `circonscription` et à toutes les variables qualitatives.

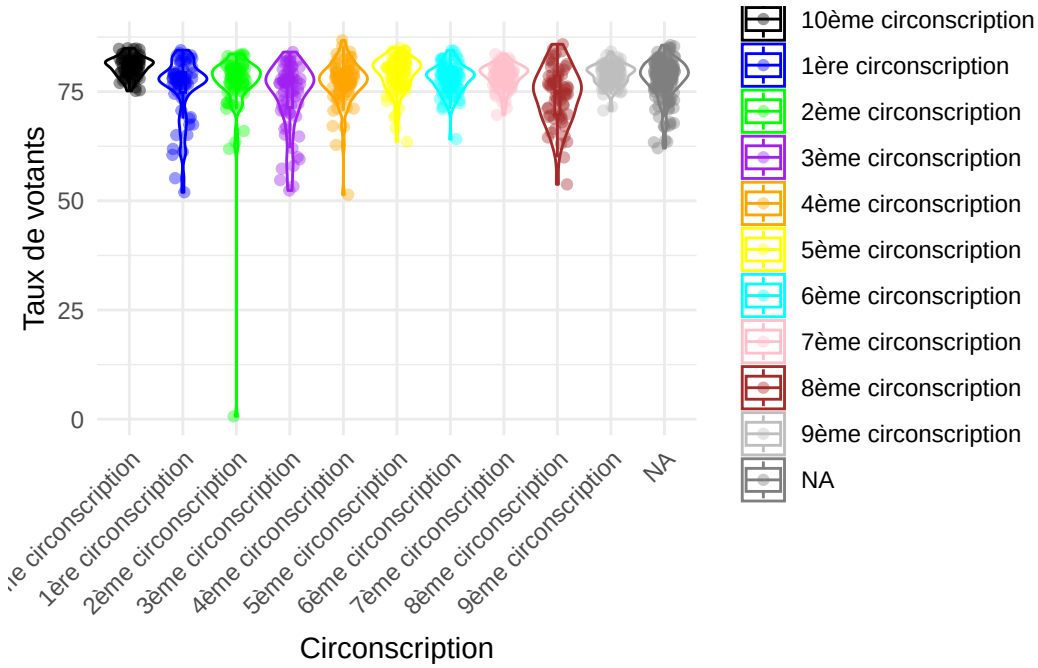
```
couleurs_circonscription <- c(
  "black", "blue", "green", "purple", "orange",
  "yellow", "cyan", "pink", "brown", "grey"
)

# Inscrits & circonscription
data |>
  ggplot() +
  aes(x = circonscription, y = inscrits, color = circonscription) +
  geom_violin(scale = "width", alpha = 0.5) +
  geom_boxplot(width = 0.1, alpha = 0.7, outlier.shape = NA) +
  geom_jitter(alpha = 0.4, width = 0.2) +
  theme_minimal() +
  scale_colour_manual(
    values = couleurs_circonscription
  ) +
  labs(x = "Circonscription",
       y = "Nombre d'inscrits",
       color = "Circonscription") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



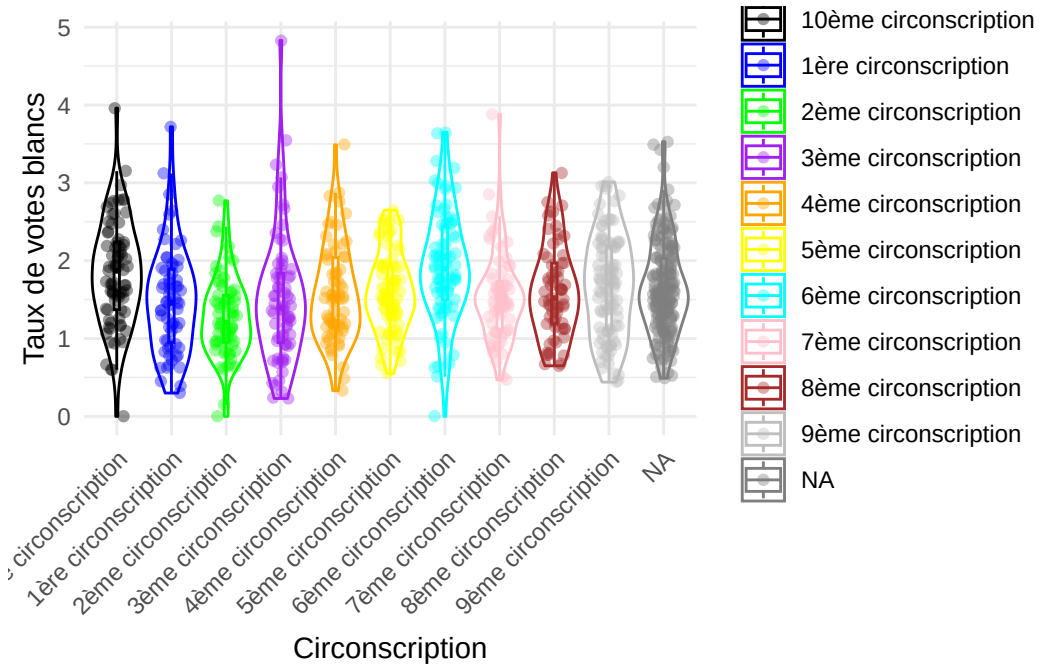

```
# Taux d'absentéisme & circonscription
data |>
  ggplot() +
  aes(x = circonscription, y = tx_absents, color = circonscription) +
  geom_violin(scale = "width", alpha = 0.5) +
  geom_boxplot(width = 0.1, alpha = 0.7, outlier.shape = NA) +
  geom_jitter(alpha = 0.4, width = 0.2) +
  theme_minimal() +
  scale_colour_manual(
    values = couleurs_circonscription
  ) +
  labs(x = "Circonscription", y = "Taux d'absentéisme", color = "Circonscription") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



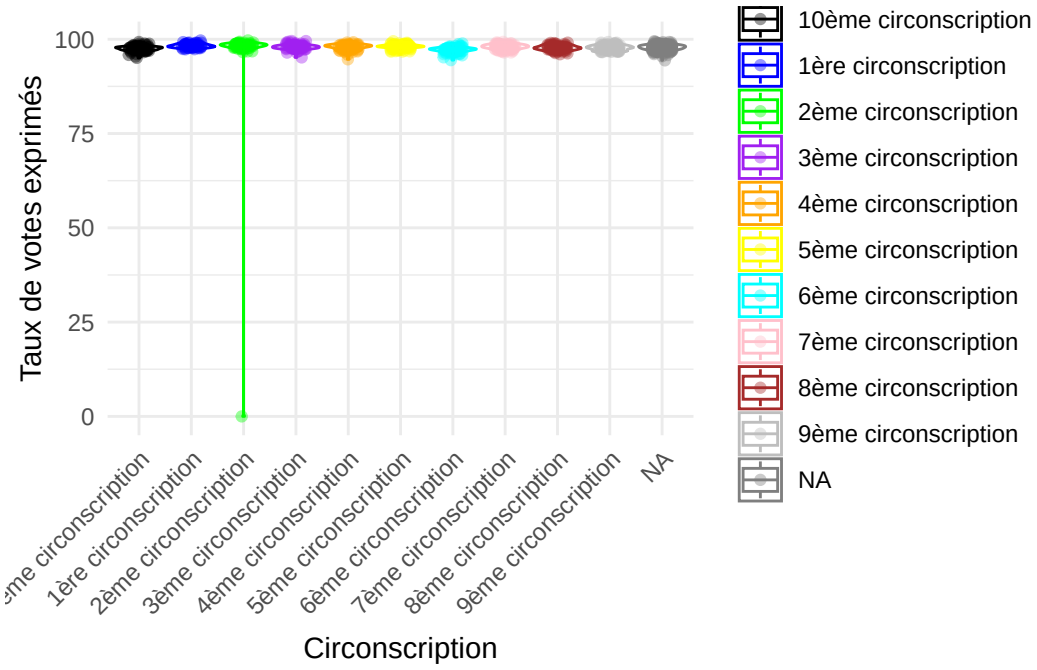

```
# Taux de votants & circonscription
data |>
  ggplot() +
  aes(x = circonscription, y = tx_votants, color = circonscription) +
  geom_violin(scale = "width", alpha = 0.5) +
  geom_boxplot(width = 0.1, alpha = 0.7, outlier.shape = NA) +
  geom_jitter(alpha = 0.4, width = 0.2) +
  theme_minimal() +
  scale_colour_manual(
    values = couleurs_circonscription
  ) +
  labs(x = "Circonscription", y = "Taux de votants", color = "Circonscription") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



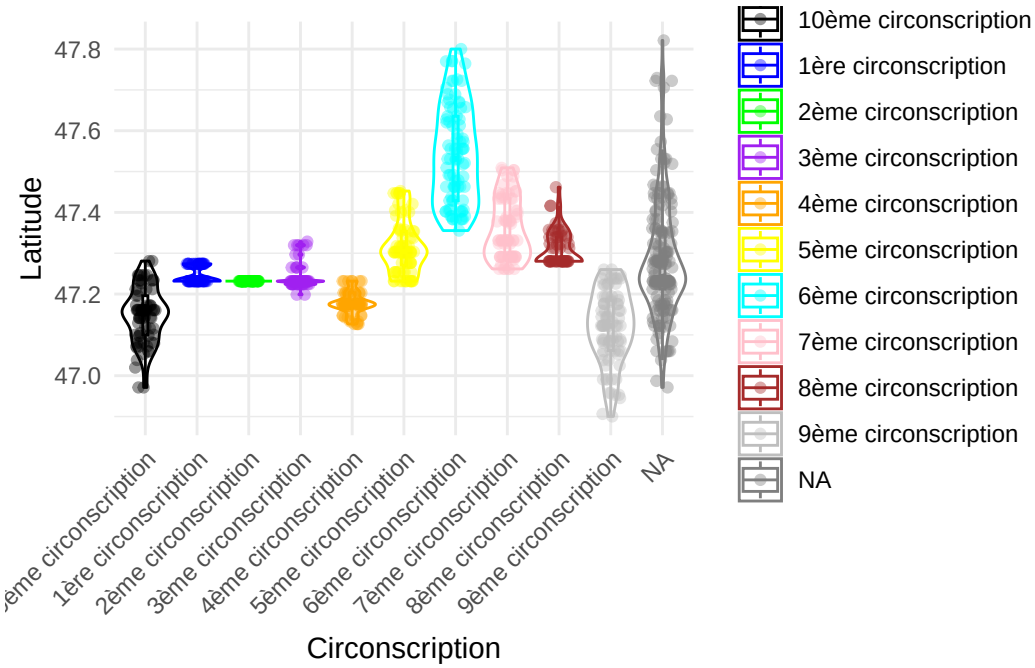

```
# Taux de votes blancs & circonscription
data |>
  ggplot() +
  aes(x = circonscription, y = tx_blancs, color = circonscription) +
  geom_violin(scale = "width", alpha = 0.5) +
  geom_boxplot(width = 0.1, alpha = 0.7, outlier.shape = NA) +
  geom_jitter(alpha = 0.4, width = 0.2) +
  theme_minimal() +
  scale_colour_manual(
    values = couleurs_circonscription
  ) +
  labs(x = "Circonscription", y = "Taux de votes blancs", color = "Circonscription") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



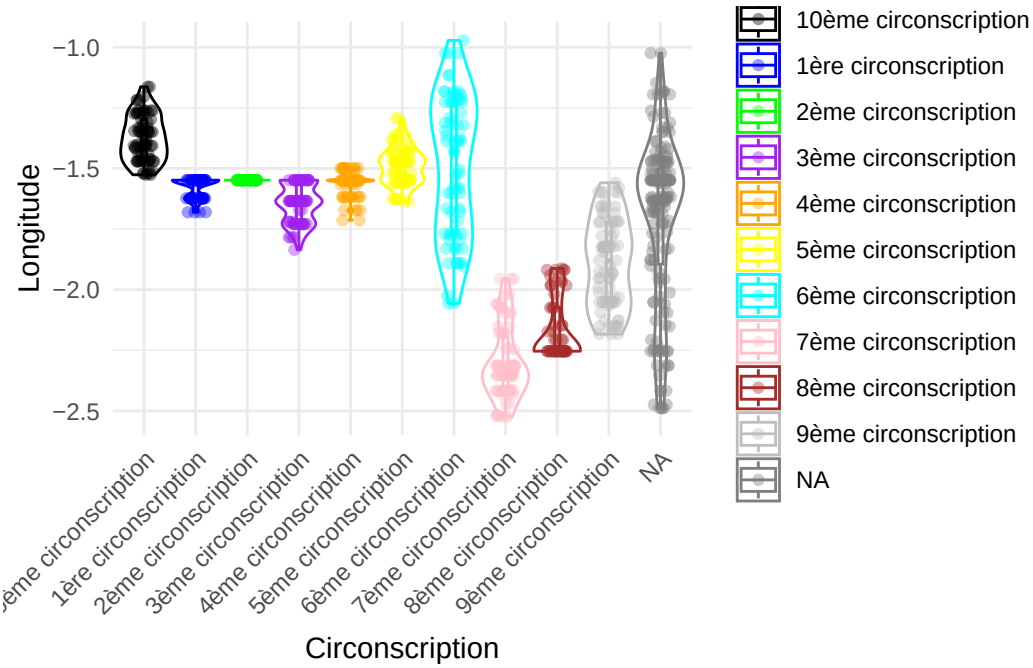

```r
# Taux de votes exprimés & circonscription
data |>
  ggplot() +
  aes(x = circonscription, y = tx_exprimes, color = circonscription) +
  geom_violin(scale = "width", alpha = 0.5) +
  geom_boxplot(width = 0.1, alpha = 0.7, outlier.shape = NA) +
  geom_jitter(alpha = 0.4, width = 0.2) +
  theme_minimal() +
  scale_colour_manual(
    values = couleurs_circonscription
  ) +
  labs(x = "Circonscription", y = "Taux de votes exprimés", color = "Circonscription") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Latitude & circonscription
data |>
  ggplot() +
  aes(x = circonscription, y = latitude, color = circonscription) +
  geom_violin(scale = "width", alpha = 0.5) +
  geom_boxplot(width = 0.1, alpha = 0.7, outlier.shape = NA) +
  geom_jitter(alpha = 0.4, width = 0.2) +
  theme_minimal() +
  scale_colour_manual(
    values = couleurs_circonscription
  ) +
  labs(x = "Circonscription", y = "Latitude", color = "Circonscription") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Longitude & circonscription
data |>
  ggplot() +
  aes(x = circonscription, y = longitude, color = circonscription) +
  geom_violin(scale = "width", alpha = 0.5) +
  geom_boxplot(width = 0.1, alpha = 0.7, outlier.shape = NA) +
  geom_jitter(alpha = 0.4, width = 0.2) +
  theme_minimal() +
  scale_colour_manual(
    values = couleurs_circonscription
  ) +
  labs(x = "Circonscription", y = "Longitude", color = "Circonscription") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Dans notre analyse quali-quanti, l'utilisation d'un diagramme de violon nous permet d'examiner visuellement si la répartition des variables quantitatives de notre base de données varie de manière significative entre les différentes circonscriptions. En observant les différences de forme, de largeur et de densité des violons pour chaque circonscription, nous pouvons identifier des tendances ou des anomalies spécifiques.

Cette approche facilite la comparaison des distributions au sein de chaque circonscription et nous permet de voir si certaines présentent des valeurs systématiquement plus élevées ou plus basses. En d'autres termes, le diagramme de violon nous aide à déterminer s'il existe une relation ou une association entre les circonscriptions et les variables quantitatives de notre base, ce qui pourrait indiquer des facteurs sociopolitiques ou organisationnels influençant les comportements dans différentes zones.

Analysons donc maintenant les résultats. Un diagramme de violon combine les caractéristiques d'un boxplot et d'un graphique de densité pour montrer la distribution d'une variable continue en fonction de catégories différentes. Pour analyser un diagramme de violon, nous observons la largeur du violon à chaque niveau de la variable, qui représente la densité des données. Une partie plus large signifie qu'il y a plus de points de données à ce niveau. La ligne centrale indique généralement la médiane des données, entourée par les quartiles, montrant l'étendue des valeurs centrales. Les extrémités du violon représentent les valeurs minimales et maximales, excluant les outliers. Enfin, la symétrie et la forme du violon permettent d'identifier des distributions équilibrées ou inclinées.

En examinant la répartition du nombre d'inscrits par circonscription, nous observons des variations entre les différentes circonscriptions. Certaines, comme la 1ère et la 2ème, montrent une distribution homogène, suggérant une répartition uniforme du nombre d'inscrits entre les bureaux de vote. Cette homogénéité peut refléter des circonscriptions urbaines où la densité de population est plus constante. D'autres, telles que la 4ème et la 8ème circonscriptions, présentent une plus grande variabilité dans le nombre d'inscrits, indiquant des différences plus marquées. Cela pourrait être dû à des zones plus rurales ou des disparités démographiques importantes au sein de ces circonscriptions.

Pour les taux d'absentéisme par circonscription, nous constatons également des variations significatives. Certaines circonscriptions montrent une distribution relativement homogène, ce qui peut indiquer une influence uniforme de facteurs sociaux ou politiques sur la participation électorale. D'autres circonscriptions présentent une plus grande variabilité, indiquant que les facteurs influençant l'absentéisme peuvent varier considérablement d'un bureau de vote à l'autre.

La répartition des taux de votants par circonscription révèle également des variations. Certaines circonscriptions ont des distributions homogènes, ce qui pourrait indiquer une mobilisation électorale relativement stable. En revanche, d'autres montrent une plus grande variabilité, suggérant des différences dans l'engagement civique ou l'accès aux bureaux de vote.

En analysant la répartition des taux de votes exprimés par circonscription, nous observons des tendances similaires. Certaines circonscriptions montrent une distribution homogène, ce qui peut indiquer une participation électorale constante. D'autres présentent une plus grande variabilité, suggérant des différences marquées dans la manière dont les électeurs expriment leurs votes.

Enfin, pour les valeurs de latitude et de longitude par circonscription, nous constatons également des variations significatives. Il est logique que ces valeurs montrent une plus grande variabilité, car la répartition géographique des communes et des bureaux de vote est différente d'une circonscription à l'autre. Les circonscriptions avec des valeurs plus homogènes peuvent indiquer des zones géographiquement plus compactes, tandis que celles avec une plus grande variabilité reflètent des régions plus étendues ou diversifiées.

Ces observations sont des éléments importants à prendre en compte lors de l'imputation des données, car elles peuvent affecter l'interprétation des résultats et aider à identifier les zones nécessitant une attention particulière pour comprendre les facteurs influençant les distributions observées. En somme, le diagramme de violon est un outil précieux pour visualiser les tendances et les anomalies, facilitant ainsi une analyse plus nuancée des comportements et des caractéristiques géographiques des circonscriptions étudiées.

4 Remplacement des NA

```
data <- read_excel("data/Données projet.xlsx")
```

Maintenant que nous avons constitué la base et qu'elle présente des vides, l'objectif principal de ce dossier est de tenter de la reconstituer le plus fidèlement possible. Pour ce faire, nous utiliserons diverses méthodes. Afin que cela soit plus clair, nous diviserons cette reconstruction en deux parties. Dans la première, nous chercherons à reconstituer la base autant que possible en nous appuyant uniquement sur la logique. Dans la seconde, pour les données qui demeurent incomplètes, nous tenterons de les approximer afin de les rapprocher de valeurs réalistes.

Toutefois, tout au long de ce dossier, il est crucial de garder à l'esprit que notre but est d'assurer la reconstitution la plus fidèle possible de la base. Ainsi, si certaines données ne peuvent pas être retrouvées, nous ne chercherons pas systématiquement à remplacer le `NA`. Il est préférable de conserver un `NA` que d'introduire une estimation erronée, ce qui pourrait entraîner des conclusions faussées.

De plus, comme indiqué en introduction, nous ne remettrons pas en question la véracité des informations contenues dans la base. En effet, nous nous plaçons dans une situation où une base de données nous est fournie, et où toutes les valeurs problématiques, manquantes, fausses, … ont déjà été remplacées par des `NA`. Notre unique objectif est donc de réussir à la compléter.

4.1 Remplacements logiques

4.1.1 Noms de communes, latitudes et longitudes

Les données concernant les communes, qu'il s'agisse de leur nom ou de leurs coordonnées géographiques (latitude et longitude), sont étroitement liées et peuvent se compléter mutuellement. Cela signifie qu'une donnée manquante, qu'il s'agisse du nom ou des coordonnées, peut potentiellement être déduite de l'autre. Cette correspondance est fondée sur le fait qu'une commune est définie de manière unique par son couple de coordonnées géographiques (latitude, longitude) ainsi que par son nom. Concrètement, le nom de la commune permet de retrouver ses coordonnées géographiques, et inversement, des coordonnées géographiques précises permettent d'identifier sans ambiguïté la commune associée.

Prenons un exemple : imaginons une ligne dans la base de données où le nom de la commune est manquant (noté `NA`), mais où les coordonnées géographiques sont présentes, disons une latitude de `47.2316` et une longitude de `-1.54831`. Si une autre ligne de la base de données indique un nom de commune valide, comme `Nantes`, avec exactement ces mêmes coordonnées, il est possible de déduire que la ligne contenant le `NA` correspond également à `Nantes`. Cela fonctionne également dans l'autre sens : si une ligne contient un nom de commune valide, mais que ses coordonnées sont partiellement ou totalement manquantes, il est possible d'utiliser une autre ligne contenant ce même nom afin de compléter les données manquantes.

De manière générale, si les valeurs sont suffisamment précises, une seule coordonnée (latitude ou longitude) suffit pour identifier une ville. Cela est particulièrement vrai dans notre cas car la Loire-Atlantique compte seulement 207 communes sur un territoire assez vaste et surtout nos

coordonnées sont très précises, de l'ordre de 10^{-11}. Ces communes, bien que non réparties de manière purement aléatoire, leur localisation étant influencée par des critères comme la géographie, les cours d'eau ou d'autres facteurs historiques, n'ont pas été implantées en fonction de leurs coordonnées géographiques, qui sont une donnée abstraite mesurée a posteriori. De plus, dans notre cas, les coordonnées attribuées à chaque ville ont été calculées en déterminant le centre géométrique des limites de la commune. Cette méthode, bien que mathématique, accentue encore davantage le côté aléatoire de la répartiton des coordonnées et rend encore plus improbable la possibilité que deux villes puissent partager exactement une même coordonnées.

Toutefois, pour en être certain, nous allons modéliser mathématiquement cette situation. De par les éléments évoqués précédemment, nous allons considérer que les communes ont été réparties avec une probabilité uniforme sur l'ensemble du territoire de la Loire-Atlantique. Pour simplifier davantage cette analyse, nous modéliserons la Loire-Atlantique comme un rectangle, dont la largeur correspond à la distance interquartile des latitudes et la longueur à la distance interquartile des longitudes. Cette approche permet de réduire l'influence des valeurs extrêmes (maximums et minimums) et de mieux représenter la distribution centrale des communes. En procédant ainsi, nous réduisons la superficie réelle de la Loire-Atlantique et, par conséquent, le nombre total de combinaisons possibles pour les coordonnées. Cela augmente donc artificiellement la probabilité que deux communes partagent une même valeur pour une coordonnée donnée, garantissant que notre calcul ne sous-évalue pas ce risque. Il est cependant pertinent de noter que la longueur interquartile peut être légèrement faussée en raison de la présence de données manquantes (NA). Toutefois, cela n'a qu'un impact minime sur les résultats, car même si certaines lignes manquent de coordonnées, il suffit que ces valeurs manquantes soient identiques à celles d'une autre ligne pour que cela n'affecte pas l'analyse. De plus, les données manquantes sont réparties aléatoirement dans notre jeu de données, ce qui minimise l'impact de ces absences. Ainsi, cette approche reste très fiable pour l'estimation des coordonnées manquantes. Procédons donc au calcul.

```r
#----- Déterminer la longueur interquartile -----

iqr_lat <- unique(data$latitude) |> # 1 commune = 1 latitude
  IQR(, na.rm = TRUE)

iqr_long <- unique(data$longitude) |> # 1 commune = 1 longitude
  IQR(, na.rm = TRUE)

nb_lat <- round(iqr_lat/10^(-11))
nb_long <- round(iqr_long/10^(-11))



#----- Calcul des probabilités -----

# Probabilité que 2 villes ou plus aient la même latitude

P_distincts <-  1

for (i in 0:(207 - 1)) { # 207 = nb communes discitinctes Loire Atlantique
```

```
  P_distincts <- P_distincts * (nb_lat - i) / nb_lat
}

P_latitude <- 1 - P_distincts



# Probabilité que 2 villes ou plus aient la même latitude

P_distincts <-  1

for (i in 0:(207 - 1)) { #207 = nb de communes discitinctes en Loire Atlantique

  P_distincts <- P_distincts * (nb_long - i) / nb_long
}

P_longitude <- 1 - P_distincts



# Probabilités

1/P_latitude
```

```
[1] 1396510.13550959
```

```
1/P_longitude
```

```
[1] 2551363.57327316
```

En conclusion, la probabilité qu'en Loire-Atlantique, deux villes partagent exactement la même latitude ou la même longitude est inférieure à 1 sur 1.3 millions. De ce fait, nous pouvons affirmer, sans risque majeur, que chaque latitude et chaque longitude est propre à une commune.

Nous pouvons donc désormais envisager avec confiance de compléter les variables `nom_commune`, `latitude` et `longitude` entre elles, comme expliqué précédemment. Cependant, avant cela, nous allons faire un petit point nomenclature.

Nomenclature

Au cours de ce dossier, nous alors devoir créer plusieurs vecteurs / matrice. Pour ce faire, et afin de nous y retrouver, nous avons établi une règle. Lorsque nous voulons extraire des données de `data`, nous devons préciser ce que nous avons sélectionné ou non en fonction des cases vides et dans l'ordre des variables du tableau.
Prenons des exemples : si dans `data` nous sélectionnons précisément toutes les lignes pour lesquelles le `nom_commune` et `latitude` sont renseignés, nous appellerons cette base `com_lat`.

De la même façon, si nous créons une base à partir de `data` avec les lignes où `nom_commune` et `longitude` sont renseignés, alors nous appllerons cette base `com_long`. Si cette fois nous décidons de prendre les données où `circonscription` et `code_bur_vote` ne sont pas renseignés, ce sera `cir_com_na`. Si, enfin, nous voulons sélectionner les lignes sans `code_bur_vote` mais avec `nom_commune` ce sera `bur_na_com`.
Cette façon de noter est généralisable à n'importe quelle base. Si nous voulons des lignes plus précises (ex : `bur_na_cir_com`) ou moins précises (ex : `com_na`), cela fonctionne tout aussi bien.

Maintenant que cela est dit, nous pouvons entamer les rempalcements.

```
# LATITUDE → NOM COMMUNE

#----- Compter les NA avant remplacement -----

sum(is.na(data$nom_commune))
```

```
[1] 215
```

```
#----- Remplacement -----

# Récapituler les latitudes de chaque commune

com_lat <- unique(data[!is.na(data$nom_commune) & !is.na(data$latitude),
                       c(2,10)])


# Matcher les circonsciptions avec la latitude

data <- data |>
  group_by(latitude) |>              # Grouper par latitude
  mutate(
    nom_commune = ifelse(            # Condition
      is.na(latitude),               # Test : La latitude est-elle vide ?
      nom_commune,                   # OUI : Ne rien changer
      ifelse(                        # NON : Nouvelle condition
        is.na(nom_commune),          # Test : Le nom de la commune est-il vide ?
        com_lat$nom_commune[
          match(latitude, com_lat$latitude)], # OUI : Chercher la correspondance
        nom_commune                  # NON : Ne rien changer
      )
    )
  ) |>
  ungroup()


#----- Compter les NA après remplacement -----
```

```r
sum(is.na(data$nom_commune))
```

```
[1] 59
```

Effectuer ces remplacements nous ont permis de passer de 215 valeurs manquantes dans `nom_commune` à 59. L'étape suivante est la même, nous échangeons simplement la variable `latitude` par la variable `longitude`.

```r
# LONGITUDE → NOM_COMMUNE

#----- Compter les NA avant remplacement -----

sum(is.na(data$nom_commune))
```

```
[1] 59
```

```r
#----- Remplacement -----

# Récapituler les longitudes de chaque commune

com_long <- unique(data[!is.na(data$nom_commune) & !is.na(data$longitude),
                        c(2,11)])


# Matcher les circonsciptions avec la longitude

data <- data |>
  group_by(longitude) |>
  mutate(
    nom_commune = ifelse(
      is.na(longitude),
      nom_commune,
      ifelse(
        is.na(nom_commune),
        com_long$nom_commune[match(longitude, com_long$longitude)],
        nom_commune
      )
    )
  ) |>
  ungroup()


#----- Compter les NA après remplacement -----

sum(is.na(data$nom_commune))
```

```
[1] 28
```

Au final, utiliser les variables `longitude` et `latitude` pour compléter `nom_commune` nous a permis de passer de 215 `NA`, ce qui représente presque 20% de la base, à 28, c'est-à-dire moins de 3% de celle-ci. Par la suite, nous allons donc effectuer le même type de remplacement pour `latitude` et pour `longitude`

```
# NOM COMMUNE → LATITUDE

#----- Compter les NA avant remplacement -----

sum(is.na(data$latitude))
```

```
[1] 217
```

```
#----- Remplacement -----

# Récapituler les latitudes de chaque ville (obligé de refaire car la longitude a permis de co

lat_com <- unique(data[!is.na(data$nom_commune) & !is.na(data$latitude),c(2,10)])


# Matcher les circonsciptions avec la latitude

data <- data |>
  group_by(latitude) |>
  mutate(
    latitude = ifelse(
      is.na(nom_commune),
      latitude,
      ifelse(
        is.na(latitude),
        lat_com$latitude[match(nom_commune, lat_com$nom_commune)],
        latitude
      )
    )
  ) |>
  ungroup()


#----- Compter les NA après remplacement -----

sum(is.na(data$latitude))
```

```
[1] 26
```

```r
# NOM COMMUNE → LONGITUDE

#----- Compter les NA avant remplacement -----

sum(is.na(data$longitude))
```

```
[1] 230
```

```r
#----- Remplacement -----

long_com <- unique(data[!is.na(data$nom_commune) & !is.na(data$longitude),
                        c(2,11)])


# Matcher les circonsciptions avec la longitude

data <- data |>
  group_by(longitude) |>
  mutate(
    longitude = ifelse(
      is.na(nom_commune),
      longitude,
      ifelse(
        is.na(longitude),
        long_com$longitude[match(nom_commune, long_com$nom_commune)],
        longitude
      )
    )
  ) |>
  ungroup()


#----- Compter les NA après remplacement -----

sum(is.na(data$longitude))
```

```
[1] 35
```

Bien que cela puisse sembler utile au premier abord, il est en réalité inutile d'effectuer les deux dernières étapes, à savoir compléter `latitude` à partir de `longitude` et `longitude` à partir de `latitude`. En effet, dès lors que nous avons complété `nom_commune` en utilisant successivement `latitude` puis `longitude`, celle-ci est devenue la variable la plus complète, ou du moins la plus complète et pertinente. Une variable "non-pertinente" à cette étape, dans notre but de les utiliser entre-elles pour déduire des valeurs, est une variable pour laquelle il existe une valeur pour dans une ligne de notre base de données, mais où il n'en existe pas pour les deux autres variables car, dans ce cas, il est impossible de s'en servir pour compléter quoi que ce soit. Ainsi, lorsque nous

avons ensuite utilisé `nom_commune` pour compléter `latitude` et `longitude`, ces deux variables ont à leur tour été complétées autant que possible. Par conséquent, tenter de compléter `latitude` à partir de `longitude`, ou inversement, n'a aucun intérêt, car elles possèdent déjà les informations que l'autre pourrait leur apporter.

4.1.2 Circonscirption

Puisque les données de `nom_commune`, `latitude` et `longitude` ont, pour l'instant, été complétées autant que possible, nous allons maintenant nous concentrer sur d'autres variables. Cependant, cela ne signifie pas que ces trois premières variables ne pourront plus être complétées par la suite. Une bonne analogie pour expliquer ce processus est le jeu du sudoku. Dans une grille de sudoku (9x9, divisée en 9 blocs de 3x3), on commence généralement par compléter un premier bloc, en remplissant les cases possibles. De même, dans notre cas, nous avons commencé par compléter le « bloc » constitué de `nom_commune`, `latitude` et `longitude`. Lorsque nous atteignons un point où il n'est plus possible d'ajouter de nouvelles données dans ce bloc, nous passons à d'autres parties de la grille, c'est-à-dire d'autres variables dans notre base. En progressant sur ces nouvelles variables, nous découvrons des informations supplémentaires qui, à leur tour, permettent de revenir au premier bloc pour le compléter davantage. Ce processus itératif nous aide à enrichir progressivement l'ensemble des données, tout comme dans un sudoku où le travail sur une partie de la grille débloque souvent des solutions ailleurs.

La prochaine variable que nous allons essayer de compléter est celle concernant la circonscription. Celle-ci fonctionne de façon assez similaire à ce que nous avons vu précédemment. Il va donc suffir de réaliser le même processus, c'est-à-dire regarder `nom_commune`, `latitude` et `longitude` pour voir si des lignes possèdes des valeurs communes et peuvent aider à compléter celles où la circonscription est vide. Bien qu'il y a de forte chances pour que la première opération avec la première variable remplace la quasi totalité des `NA` qu'il est possible de déterminer, nous sommes contraints de tout de même le faire avec les 3 variables car il se pourrait, par exemple, que 2 lignes n'aient pas de nom de commune, pas de latitude mais partagent la même longitude et qu'une des deux ait l'information concernant la circonscription et pas l'autre. Néanmoins, à la différence d'avant, la relation entre `nom_commune`, `latitude` et `longitude` avec `circonscription` n'est pas bilatérale. En effet, regarder la circonscription d'une ligne ne peut pas nous donner avec certitude le nom de la commune, la longitude ou la latitude car il y a plusieurs communes au sein d'une même circonscription.

Avant d'apporter des modifications à nos données, il est essentiel de vérifier qu'une commune donnée, identifiée par son nom ou ses coordonnées géographiques, n'est associée qu'à une seule circonscription. Bien qu'il puisse manquer certaines informations, il est peu probable qu'une commune présente dans plusieurs circonscriptions passe totalement inaperçue à cause des `NA`. En effet, les communes réparties sur plusieurs circonscriptions sont généralement de grandes villes, comportant de nombreux bureaux de vote, ce qui entraîne plusieurs lignes dans notre jeu de données. De plus, comme les `NA` ont été insérés de manière aléatoire et non selon un regroupement spécifique, la probabilité qu'ils masquent un groupe entier de données est extrêmement faible. Par conséquent, notre objectif est simplement de vérifier si une commune apparaît dans plus d'une circonscription, sans chercher à déterminer précisément combien.

```r
#----- Analyse dans nom_commune -----

data |>
  filter(!is.na(nom_commune)) |> # Exclure les lignes sans nom de commune
  group_by(nom_commune) |>       # Grouper par commune
  summarise(
    unique_circonscriptions = n_distinct(
      circonscription, na.rm = TRUE),  # Compter les circonscriptions uniques
    .groups = "drop"                   # Supprimer le regroupement
  ) |>
  filter(unique_circonscriptions > 1)  # Faire ressortir si +1 circonscription
```

```
# A tibble: 1 x 2
  nom_commune unique_circonscriptions
  <chr>                         <int>
1 Nantes                            5
```

```r
#----- Analyse dans latitude -----

data |>
  filter(!is.na(latitude)) |>
  group_by(latitude) |>
  summarise(
    unique_circonscriptions = n_distinct(circonscription, na.rm = TRUE),
    commune = first(nom_commune),   # Ajouter le nom de la ville
    .groups = "drop"
  ) |>
  filter(unique_circonscriptions > 1)
```

```
# A tibble: 1 x 3
  latitude unique_circonscriptions commune
     <dbl>                   <int> <chr>
1     47.2                       5 Nantes
```

```r
#----- Analyse dans longitude -----

data |>
  filter(!is.na(longitude)) |>
  group_by(longitude) |>
  summarise(
    unique_circonscriptions = n_distinct(circonscription, na.rm = TRUE),
    commune = first(nom_commune),
    .groups = "drop"
  ) |>
  filter(unique_circonscriptions > 1)
```

```
# A tibble: 1 x 3
  longitude unique_circonscriptions commune
      <dbl>                   <int> <chr>
1     -1.55                       5 Nantes
```

Après analyse, et comme nous pouvions nous y attendre, une seule commune apparaît associée à plusieurs circonscriptions : `Nantes`, qui est répartie sur cinq circonscriptions. Une recherche rapide sur Internet nous a permis de confirmer qu'aucune autre commune ne partage cette particularité, ce qui valide notre conclusion : Nantes est la seule commune présente dans plusieurs circonscriptions. Dans certaines bases de données, un numéro est parfois ajouté au nom des communes pour indiquer à quelle circonscription appartient chaque bureau de vote. Cependant, ce n'est pas le cas ici. De plus, les coordonnées géographiques ne permettent pas non plus de distinguer les différentes circonscriptions pour Nantes, car aucune différenciation explicite n'a été effectuée. Par conséquent, lorsque nous utiliserons `nom_commune`, `latitude` ou `longitude` pour compléter les données manquantes de la variable `circonscription`, il sera nécessaire d'exclure Nantes de ces opérations. Sinon, la circonscription attribuée pour remplacer un `NA` pourrait être incorrecte, compromettant ainsi la précision des données.

Commençons par utiliser le nom des communes pour trouver la circonsciption

```
# NOM COMMUNE → CIRCONSCRIPTION

#----- Compter les NA avant remplacement -----

sum(is.na(data$circonscription))
```

```
[1] 221
```

```
#----- Remplacement -----

cir_com <- unique(data[!is.na(data$nom_commune) & !is.na(data$circonscription),c(1,2)])


## Matcher les circonsciptions avec le nom de commune

data <- data |>
  group_by(nom_commune) |>
  mutate(
    circonscription = ifelse(
      nom_commune == "Nantes" | is.na(nom_commune),
      circonscription,
      ifelse(
        is.na(circonscription),
        cir_com$circonscription[match(nom_commune, cir_com$nom_commune)],
        circonscription
      )
    )
```

```
  ) |>
  ungroup()




#----- Compter les NA après remplacement -----

sum(is.na(data$circonscription))
```

```
[1] 65
```

Puis maintenant la latitude pour trouver la circonscription.

```
# LATITUDE → CIRCONSCRIPTION

#----- Compter les NA avant remplacement -----

sum(is.na(data$circonscription))
```

```
[1] 65
```

```
#data[2,c(1,2,11)] <- NA
#preuve que la commande marche car il n'y pas de cas dans notre base



#----- Remplacement -----

cir_lat <- unique(data[!is.na(data$latitude) & !is.na(data$circonscription),c(1,10)])

lat_nantes <- unique(
  data$latitude[which(data$nom_commune == "Nantes" & !is.na(data$latitude))]
  )

## Matcher les circonsciptions avec la latitude

data <- data |>
  group_by(latitude) |>
  mutate(
    circonscription = ifelse(
      latitude == lat_nantes | is.na(latitude),
      circonscription,
      ifelse(
```

```
        is.na(circonscription),
        cir_lat$circonscription[match(latitude, cir_lat$latitude)],
        circonscription
      )
    )
  ) |>
  ungroup()




#----- Compter les NA après remplacement -----

sum(is.na(data$circonscription))
```

```
[1] 65
```

Et, enfin, la longitude pour trouver la circonscription.

```
# LONGITUDE → CIRCONSCRIPTION

#----- Compter les NA avant remplacement -----

sum(is.na(data$circonscription))
```

```
[1] 65
```

```
#data[2,c(1,2,10)]
#preuve que la commande marche car il n'y pas de cas dans notre base



#----- Remplacement -----

cir_long <- unique(data[!is.na(data$longitude) & !is.na(data$circonscription),c(1,11)])

long_nantes <- unique(
  data$longitude[which(data$nom_commune == "Nantes" & !is.na(data$longitude))]
  )

## Matcher les circonsciptions avec la longitude

data <- data |>
  group_by(longitude) |>
```

```
  mutate(
    circonscription = ifelse(
      longitude == long_nantes | is.na(longitude),
      circonscription,
      ifelse(
        is.na(circonscription),
        cir_long$circonscription[match(longitude, cir_long$longitude)],
        circonscription
      )
    )
  ) |>
  ungroup()




#----- Compter les NA après remplacement -----

sum(is.na(data$circonscription))
```

```
[1] 65
```

Nous avons donc remplacés dans `circonscription` toutes les valeurs manquantes dont nous étions certains grâce aux variables `nom_commune`, `latitude` et `longitude`. Cela nous a permis de passer de 221 valeurs `NA` à 65, ce qui représente tout de même une belle réduction.

4.1.3 Code des bureaux de vote

Afin de poursuivre la reconstitution de notre jeu de données, nous allons nous intéresser à une variable que nous n'avons, jusqu'à là, pas mobilisé : `code_bur_vote`. En tant que telle, la variable n'a pas de réel intérêt. En effet, si nous complétons d'abord notre jeu de données par des remplacements "sûrs" c'est dans l'objectif de pouvoir ensuite estimer du mieux possible les données introuvables. Les remplacements incertains, qui arriveront ensuite, utilisent les différentes données à disposition pour estimer au mieux les valeurs. Si par exemple nous souhaitons estimer `tx_absents`, celui-ci va pouvoir être estimé grâce à `circonscription` et `nom_commune` car, il est connu que, lors des élections, les communes d'une même circonscription partagent généralement des intérêts communs et des spécificités communes (type de population, emploi, ...) et donc que les tendances de votes, mais aussi d'absentéisme ou de vote blancs, se rapprochent. Cependant, au sein d'une commune, il n'y a pas de réelle différence entre les bureaux de votes. Qu'un individu soit dans le bureau 1 ou 5, il va voter de la même manière. De ce fait, déterminer les codes de bureaux de vote va uniquement nous servir à trouver de façon sûre les circonscriptions, les noms de communes, et donc la latitude et la longitude.

Pour ce faire, nous allons commencer par regarder les bureaux de votes manquants pour chaque commune. L'idée ici est d'examiner, pour chaque commune, s'il existe une interruption dans la

numérotation des bureaux de vote. Par exemple, si une commune dispose de 5 bureaux numérotés de 1 à 5, mais que le numéro 4 est manquant, nous vérifierons si une ligne dépourvue de nom de commune correspond à un bureau numéro 4. Cela pourrait indiquer que la ligne est celle de la commune en question. Le problème de cette méthode est qu'il pourrait y avoir plusieurs "bureau 4" en trop. De ce fait, afin d'affiner la recherche et limiter les possibilités, nous allons traiter ce problème par circonscription. En effet, si deux "bureau 4" sont disponibles, il est compliqué de savoir quelle ligne appartient à quelle commune. Cependant, en y ajoutant la dimension `circonsciption`, comme il y en a 10, il y a de fortes chances que ces deux bureaux ne soient pas dans la même circonsciption et donc que nous puissions en déduire la commune associée, et donc les coordonnées géographique car, comme expliqué précédemment, tout est lié. Toutefois, nous devons, à nouveau, retirer Nantes car ses bureaux de vote ont une numérotation spéciale et ne permettent pas de déterminer quoi que ce soit.

```
bur_trou <- data |>
  filter(nom_commune != "Nantes") |>  # Exclure Nantes
  group_by(nom_commune, circonscription) |>  # Inclure la circonscription
  summarise(
    nb_bureaux_distincts = n_distinct(
      code_bur_vote, na.rm = TRUE), # Bureaux distincts (sans NA)
    nb_bureaux_total = n(),
    nb_na_bureaux = sum(is.na(code_bur_vote)),
    bureaux_present = list(
      sort(unique(code_bur_vote, na.rm = TRUE))),  # Liste des bureaux présents
    .groups = "drop"
  ) |>
  filter(nb_bureaux_total != nb_bureaux_distincts) |>  # Garder ceux avec écarts
  mutate(
    bureaux_manquants = map2(
      bureaux_present, nb_bureaux_total,
      ~ setdiff(seq(1, .y), .x)           # Trouver les bureaux manquants
    )
  ) |>
  select(circonscription, everything(), -bureaux_present)

bur_trou$bureaux_manquants <- as.character(bur_trou$bureaux_manquants)

bur_trou
```

```
# A tibble: 94 x 6
   circonscription       nom_commune       nb_bureaux_distincts nb_bureaux_total
   <chr>                 <chr>                            <int>            <int>
 1 6ème circonscription  Ancenis-Saint-Gé~                    8                9
 2 10ème circonscription Basse-Goulaine                       5                7
 3 7ème circonscription  Batz-sur-Mer                         1                3
 4 7ème circonscription  Besné                                2                3
 5 6ème circonscription  Blain                                7                9
 6 4ème circonscription  Bouaye                               5                6
```

```
 7 4ème circonscription  Bouguenais                          12              15
 8 <NA>                  Brains                               1               2
 9 5ème circonscription  Carquefou                           17              19
10 9ème circonscription  Chaumes-en-Retz                      3               6
# i 84 more rows
# i 2 more variables: nb_na_bureaux <int>, bureaux_manquants <chr>
```

Maintenant que nous avons réussi à détecter les différents trous, nous allons lister les différentes lignes qui pourraient compromettre le fait de combler les vides. Tout d'abord, une ligne étant à la fois `NA` dans `circonscription`, `nom_commune` et `code_bur_vote` serait très embarrassante car elle pourrait aller n'importe où puisqu'elle ne pourrait pas se différencier des autres par sa circonscription, son nom ou son code de bureau de vote. La seule façon de réussir à la différencier serait alors de comparer les coordonnées géographiques, mais cela implique qu'elle en ait, car elles pourraient très bien aussi être `NA`, et que la ligne à laquelle nous souhaitons la comparer en ait aussi. Cependant, avant de savoir comment régler ce problème, nous devons d'abord voir si ce problème existe dans notre jeu de données, autrement dit, regarder s'il des lignes avec à la fois `NA` dans `circonscription`, dans `nom_commune` et dans `code_bur_vote`.

```r
sum(data[is.na(data$circonscription)
         & is.na(data$nom_commune)
         & is.na(data$code_bur_vote),])
```

```
[1] 0
```

Par chance, nous n'avons pas de telles lignes dans notre jeu de données. Deuxièmement, bien qu'aucune ligne avec les trois informations manquantes existe, il peut en exister deux. En effet, il peut y avoir des lignes avec :

- `NA` dans `circonscription` et `nom_commune` mais avec une valeur dans `code_bur_vote`
- `NA` dans `circonscription`, dans `code_bur_vote` mais avec une valeur dans `nom_commune`
- `NA` dans `nom_commune`, dans `code_bur_vote` mais avec une valeur dans `circonscription`

Ces lignes peuvent être problématiques car 2 `NA` pour 3 variables offrent un grand degré de liberté puisque, par exemple, s'il n'y a ni circonscription ni nom de commune, cela veut dire que la ligne peut appartenir à n'importe quelle circonscription de n'importe quelle commune tant qu'elle a le code de bureau de vote manquant. Nous allons donc regrouper les données selon ces trois cas pour pouvoir y faire attention.

```r
# ----- Pas circonscription / nom de commune mais un code bureau de vote -----

# Comme vu précédemment, les 3 ne peuvent pas être NA, donc inutile de préciser
# la présence d'un bureau de vote

cir_com_na <- data[is.na(data$circonscription) & is.na(data$nom_commune),]
cir_com_na
```

```
# A tibble: 4 x 11
  circonscription nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>           <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 <NA>            <NA>        1                  736       20.2       79.8
2 <NA>            <NA>        11                1160       17.2       82.8
3 <NA>            <NA>        17                1186       23.4       76.6
4 <NA>            <NA>        3                  873       32.0       NA
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

```
# ----- Pas circonscription / code bureau de vote mais un nom de commune -----

# Comme vu précédemment, les 3 ne peuvent pas être NA, donc inutile de préciser
# la présence d'un nom de commune

cir_bur_na <- data[is.na(data$circonscription) & is.na(data$code_bur_vote),]
cir_bur_na
```

```
# A tibble: 15 x 11
   circonscription nom_commune      code_bur_vote inscrits tx_absents tx_votants
   <chr>           <chr>            <chr>            <dbl>      <dbl>      <dbl>
 1 <NA>            Brains           <NA>                NA       18.4       81.6
 2 <NA>            Issé             <NA>              1425       24.1       75.9
 3 <NA>            Nantes           <NA>               958       19.9       80.1
 4 <NA>            Nantes           <NA>               920       21.0       79.0
 5 <NA>            Nantes           <NA>               866       36.5       63.5
 6 <NA>            Nantes           <NA>                NA       19.9       80.1
 7 <NA>            Nantes           <NA>               990       20.6       79.4
 8 <NA>            Nantes           <NA>               954       21.7       78.3
 9 <NA>            Nantes           <NA>              1032       22.2       77.8
10 <NA>            Nantes           <NA>               985       25.0       75.0
11 <NA>            Nantes           <NA>               949       NA         75.9
12 <NA>            Saint-Aubin-des~ <NA>              1241       20.0       80.0
13 <NA>            Sévérac          <NA>              1229       NA         75.9
14 <NA>            Sion-les-Mines   <NA>              1278       24.7       75.3
15 <NA>            La Turballe      <NA>                NA       26.5       73.5
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

```
# ----- Pas nom commune / code de bureau de vote mais une circonscription -----

# Comme vu précédemment, les 3 ne peuvent pas être NA, donc inutile de préciser
# la présence d'une circonscription

com_bur_na <- data[is.na(data$nom_commune) & is.na(data$code_bur_vote),]
com_bur_na
```

```
# A tibble: 4 x 11
  circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 7ème circonscription <NA>        <NA>                NA       23.2       76.8
2 6ème circonscription <NA>        <NA>               958       NA         NA
3 3ème circonscription <NA>        <NA>               849       NA         57.4
4 6ème circonscription <NA>        <NA>               523       20.8       79.2
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Maintenant que nous avons identifié les individus problématiques, nous allons regarder circonscription par circonscription les bureaux de votes manquants pour chaque commune.

Toutefois, avant de commencer, il est important de noter que la liste des codes de bureaux de vote manquants par commune, que nous avons précédemment vue, n'est pas exhaustive. En effet, pour réaliser cette liste, nous sommes partis du principe qu'une commune avait nécessairement au moins un bureau de vote et ensuite nous avons compté le nombre de lignes où la commune apparaît. Cela veut dire que si la ligne apparaît `n` fois, elle a au moins `n` bureaux de vote. Après cela, nous avons fait la différence entre cette séquence complète (c'est-à-dire la séquence s'étendant de 1 jusqu'à `n`) et les codes de bureaux réellement constatés. Cependant, nous savons que certaines lignes ont un nom de commune manquant. De ce fait, il pourrait en fait y avoir plus de lignes dans la commune que ce que nous avons compté. Pour illustrer cela, prenons un exemple. Si Saint-Nazaire apparaît 45 fois dans la base de données, nous avons créé une séquence allant de 1 (au moins un bureau de vote) jusqu'à 45 (nombre de bureaux nécessaires pour combler toutes les lignes), et la séquence est donc : 1 2 3 … 44 45. Si maintenant nous comparons sur ces 45 lignes les codes des bureaux de vote à la séquence créée, il peut y avoir des trous, c'est-à-dire des endroits où il manque un ou plusieurs bureaux de vote pour que la séquence soit complète, c'est ce que nous avons récapitulé dans le tableau `bur_trou`. Si, toujours dans notre exemple, nous constatons dans `bur_trou` qu'il y a un seul trou, le code de bureau de vote 35, et que nous constatons dans notre base de données `data` qu'il existe une seule ligne avec le nom de commune Saint-Nazaire et sans code de bureau de vote, nous pourrions être tentés de dire qu'il s'agit nécessairement de la ligne contenant le bureau de vote 35. Cependant, c'est ici où nous devons être vigilant car, comme vu juste avant, il existe certaines lignes qui n'ont ni nom de commune, ni code de bureau de vote mais aussi des lignes ayant seulement la circonscription (en admettant ici que ce soit la même que Saint-Nazaire), de ce fait ces lignes pourraient très bien être celles contenant le code de bureau de vote 35 et le ou les autres lignes, notamment celle que nous pensions nécessairement être celle avec le code de bureau 35, être plutôt des lignes situées après 45 (donc 46 47 …). En effet, si nous ajoutons une ligne qui n'avait pas de nom de commune à Saint-Nazaire, `n` n'est plus de 45 mais de 46, et, de ce fait, il manque aussi le code de bureau 46, et plus seulement le 35. Par conséquent, il est important de bien vérifier avant de conclure qu'une ligne avec un trou appartient à un groupe et c'est ce que nous allons tâcher de faire dans la suite.

Pour faire face à ces problèmes, nous allons procéder de la manière suivante. Nous allons commencer par regarder, et retenir, le code de bureau de vote des communes qui n'ont ni circonscription ni nom dans `cir_com_na` car nous aurons du mal à conclure quoi que ce soit lorsque ce numéro manque puisque les lignes pourraient correspondre à trop de communes.

```
cir_com_na$code_bur_vote
```

```
[1] "1"  "11" "17" "3"
```

Il y en a donc quatre différents : 1, 3, 11 et 17. Interessons nous maintenant aux circonscriptions dans lequelles il y a des lignes sans nom de commune et sans code de bureaux de vote car, comme expliqué précédemment, de telles lignes sont trop libres et compromettent le fait de retrouver les infomations manquante par des raisonnements logiques.

```
cir_pb <- unique(com_bur_na$circonscription)
cir_pb
```

```
[1] "7ème circonscription" "6ème circonscription" "3ème circonscription"
```

Il y a donc trois circonscriptions différentes : la 3ème circonscription, la 6ème circonscription et la 7ème. Pour le moment, comme pour les bureaux de votes cités précédemment, nous ne seront pas en mesure de conclure quoi que ce soit lorsqu'une ligne est issue de cette circonscription. De ce fait, nous choisissons de ne pas nous y intéresser pour les remplacements qui vont venir.

```
bur_trou_cir_pb <- bur_trou[!(bur_trou$circonscription %in% cir_pb),]
bur_trou_cir_pb
```

```
# A tibble: 59 x 6
   circonscription       nom_commune       nb_bureaux_distincts nb_bureaux_total
   <chr>                 <chr>                            <int>            <int>
 1 10ème circonscription Basse-Goulaine                       5                7
 2 4ème circonscription  Bouaye                               5                6
 3 4ème circonscription  Bouguenais                          12               15
 4 <NA>                  Brains                               1                2
 5 5ème circonscription  Carquefou                           17               19
 6 9ème circonscription  Chaumes-en-Retz                      3                6
 7 9ème circonscription  Corcoué-sur-Logne                    1                2
 8 9ème circonscription  Corsept                              1                2
 9 5ème circonscription  Couffé                               1                2
10 10ème circonscription Divatte-sur-Loire                    4                6
# i 49 more rows
# i 2 more variables: nb_na_bureaux <int>, bureaux_manquants <chr>
```

Maintenant que nous avons extrait les données sur lequelles nous pouvons travailler, nous allons regarder les lignes dans `bur_trou_cir_pb` pour lesquelles la valeur dans `nb_na_bureau` est inférieure au nombre de codes de bureau de vote manquants (colonne `bureaux_manquants`). Attention, il s'agit bien du nombre de valeurs dans `bureaux_manquants` (il faut les compter) et non du numéro en lui-même. Une fois cela fait, 4 villes ressortent. Nous allons donc réduire notre tableau `bur_trou_cir_pb` à ces 4 lignes.

```
bur_trou_cir_pb_1 <- bur_trou_cir_pb[c(2,5,38,57),]
bur_trou_cir_pb_1
```

```
# A tibble: 4 x 6
  circonscription       nom_commune        nb_bureaux_distincts nb_bureaux_total
  <chr>                 <chr>                             <int>            <int>
1 4ème circonscription  Bouaye                                5                6
2 5ème circonscription  Carquefou                            17               19
3 9ème circonscription  Saint-Brevin-les-~                    9               11
4 10ème circonscription Vertou                               17               20
# i 2 more variables: nb_na_bureaux <int>, bureaux_manquants <chr>
```

Ces quatre communes partagent un point commun : elles n'ont pas assez de lignes sans code de bureau de vote vide pour combler les bureaux qu'il leur manque nécessairement. De ce fait, cela veut dire que chacune de ces 4 communes a au moins une ligne pour qui `nom_commune` est un `NA` à la place de porter le nom de la commune. Grâce à cela, nous allons pouvoir compléter certaines lignes en trouvant leur valeur dans `nom_commune` ou dans `code_bur_vote`. Afin de simplifier la suite, créons ici la base `bur_na_com`, c'est-à-dire la base regroupant les lignes qui ont un nom de commune mais qui n'ont pas de de code de bureau de vote.

```
bur_na_com <- data[!is.na(data$nom_commune) & is.na(data$code_bur_vote), ]
bur_na_com
```

```
# A tibble: 222 x 11
   circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
   <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
 1 6ème circonscription Ancenis-Sa~ <NA>               885       NA         NA
 2 9ème circonscription Chaumes-en~ <NA>               862       19.7       80.3
 3 9ème circonscription Chaumes-en~ <NA>               532       18.8       NA
 4 9ème circonscription Chaumes-en~ <NA>               909       22.2       77.8
 5 10ème circonscripti~ Basse-Goul~ <NA>              1055       20.8       NA
 6 10ème circonscripti~ Basse-Goul~ <NA>              1189       NA         NA
 7 7ème circonscription Batz-sur-M~ <NA>              1042       20.8       79.2
 8 7ème circonscription Batz-sur-M~ <NA>               951       22.3       77.7
 9 7ème circonscription Besné       <NA>               536       22.4       NA
10 10ème circonscripti~ Le Bignon   <NA>               875       20.7       79.3
# i 212 more rows
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Pour commencer, reprenons notre liste de quatre communes et regardons la première : `Bouaye`. Il manque à celle-ci au moins deux bureaux : le 3 et le 4, mais il n'y a qu'une ligne dans la base de données pourtant le nom `Bouaye` dans `bur_na_com`.

```r
sum(bur_na_com$nom_commune == "Bouaye")
```

```
[1] 1
```

De ce fait, il n'y a pas assez de lignes pour combler les deux bureaux manquants. Cela veut donc dire qu'il y a une autre ligne dans la base de données qui devrait avoir dans `nom_commune` le nom `Bouaye` mais qui a un `NA` pour le moment. Si nous commençons par regarder le code de bureau de vote 4, comme ce n'est pas une valeur présente dans les lignes de `cir_com_na`, la seule possibilité, autre que la ligne dans `bur_na_com` est qu'il y ait une ligne avec la circonscription de Bouaye, c'est-à-dire la 4ème circonscription, mais avec un `NA` pour `nom_commune` et `code_bur_vote`, car, nous le rappelons, il n'y a pas dans notre base une ligne où toutes les informations manquent. Regardons donc les lignes de la 4ème circonscription qui n'ont ni nom de commune, ni code de bureau de vote.

```r
data[data$circonscription == "4ème circonscription" &
       is.na(data$nom_commune) & !is.na(data$circonscription),]
```

```
# A tibble: 1 x 11
  circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 4ème circonscription <NA>        3                 1028       17.0       83.0
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Comme dans la 4ème circonscription le seul code de bureau de vote disponible est le numéro 3, nous en déduisons que, nécessairement, la ligne portant le nom de `Bouaye` mais n'ayant pas de numéro de code de bureau de vote est en fait celle du bureau de vote 4. Nous pouvons donc modifier notre base de données `data` en conséquence, mais aussi `bur_trou` et `bur_na_com`.

```r
# ----- data -----

index_bouay_bur4 <- which(data$nom_commune == "Bouaye" &
                            is.na(data$code_bur_vote))
data$code_bur_vote[index_bouay_bur4] <- 4



# ----- bur_trou -----

bur_trou$nb_bureaux_distincts[bur_trou$nom_commune == "Bouaye"] <- 6
bur_trou$nb_na_bureaux[bur_trou$nom_commune == "Bouaye"] <- 0
bur_trou$bureaux_manquants[bur_trou$nom_commune == "Bouaye"] <- "3"
```

```r
# ----- bur_na_com -----

bur_na_com <- bur_na_com[-which(bur_na_com$nom_commune == "Bouaye"),]
```

Maintenant qu'il n'y a plus de ligne dans `bur_na_com` de disponible pour Bouaye, intéressons nous à l'autre code de bureau de vote qui lui manque : le numéro 3. Comme nous l'avons vu précédemment, une ligne avec un code de bureau de vote 3 et sans nom de commune est disponible dans la 4ème circonscription. Il est donc tentant d'affirmer qu'il s'agit de celle de Bouaye. Cependant, nous devons nous rappeler que dans notre base de données, il existe une ligne sans circonscription et sans nom de commune ayant le code de bureau de vote 3, de ce fait, nous ne pouvons rien conclure. De plus, si nous regardons dans `bur_trou` les communes de la `4ème circonscription`, certaines ont égélament besoin d'un code de bureau 3 (Rezé et Saint-Sébastien-sur-Loire) et d'autres pourraient en avoir besoin (cas des communes qui ont seulement 2 lignes avec une valeur dans `nom_commune` et où nous n'avons donc pas pu déterminer s'il manquait davantage bureaux), donc même si nous sommes sûrs que la ligne dans `com_na_cir_4` appartient bien à une commune de la 4ème circonscription ayant besoin d'un code de bureau de vote 3, nous ne pouvons pas confirmer, ni infirmer, qu'il s'agit de celle de Bouaye.

Suite à cela, nous pouvons nous intéresser à la prochaine commune.

```r
bur_trou_cir_pb_1[2,]
```

```
# A tibble: 1 x 6
  circonscription      nom_commune nb_bureaux_distincts nb_bureaux_total
  <chr>                <chr>                      <int>            <int>
1 5ème circonscription Carquefou                     17               19
# i 2 more variables: nb_na_bureaux <int>, bureaux_manquants <chr>
```

La situation de Carquefou est un peu différente de celle de Bouaye car il n'y a plus deux bureaux manquants mais trois, mais l'idée reste la même, il n'y a toujours pas assez de lignes dans `bur_na_com` pour pouvoir insérer tous les codes de bureau de vote manquants. Pour gagner en efficacité, nous allons, cette fois, analyser les trois codes manquants simultanément. Nous savons déjà que les codes de bureau de vote 4, 8 et 9 ne sont pas présents dans `cir_com_na`. De ce fait, nous allons regarder les lignes qui ont un bureau de vote mais pas de nom de commune dans la circonscription de Carquefou : la 5ème circonscription. Si dans celle-ci il n'y pas de ligne qui corresponde, cela serait très problématique car cela indiquerait que nous avons commis une erreur de raisonnement.

```r
data[data$circonscription == "5ème circonscription" &
                        is.na(data$nom_commune) & !is.na(data$circonscription),]
```

```
# A tibble: 1 x 11
  circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 5ème circonscription <NA>        8                  770       19.4       80.6
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Dans la 5ème circonscription, il n'y a qu'une ligne et celle-ci possède bien un code de bureau de vote qu'il manquait à Carquefou. Par conséquent, cela implique que la ligne de la 5ème circonscription qui n'a pas de valeur dans `nom_commune` mais qui a un 8 dans `code_bur_na`, est celle de Carquefou, nous pouvons même compléter les coordonnées géographiques qu'il manquait. De plus, cela implique également que les codes de bureau de vote 4 et 9 correspondent aux lignes dans `bur_na_com`. Cependant, nous ne sommes pas en mesure de terminer entre les deux laquelle correspond au code de bureau de vote 4 et laquelle au code de bureau de vote 9. Nous pouvons tout de même modifier nos différentes bases de données en conséquence.

```
# ----- data -----

index_carquefou_bur8 <- which(data$circonscription == "5ème circonscription" &
                              is.na(data$nom_commune) &
                              data$code_bur_vote == "8")

data$nom_commune[index_carquefou_bur8] <- "Carquefou"

data$latitude[index_carquefou_bur8] <-
  com_lat$latitude[which(com_lat$nom_commune == "Carquefou")]

data$longitude[index_carquefou_bur8] <-
  com_long$longitude[which(com_long$nom_commune == "Carquefou")]


# ----- bur_trou -----

bur_trou$nb_bureaux_distincts[bur_trou$nom_commune == "Carquefou"] <- 18
bur_trou$nb_na_bureaux[bur_trou$nom_commune == "Carquefou"] <- 2
bur_trou$bureaux_manquants[bur_trou$nom_commune == "Carquefou"] <- "c(4,9)"
```

Maintenant que cela est fait, nous pouvons passer à la troisième commune.

```
bur_trou_cir_pb_1[3,]
```

```
# A tibble: 1 x 6
  circonscription      nom_commune         nb_bureaux_distincts nb_bureaux_total
  <chr>                <chr>                             <int>            <int>
1 9ème circonscription Saint-Brevin-les-P~                   9               11
# i 2 more variables: nb_na_bureaux <int>, bureaux_manquants <chr>
```

Dans celle-ci, trois codes de bureau de vote manquent : le 8, le 9 et 11. Regardons maintenant les codes de bureau de vote disponibles dans le circonscription.

```
data[data$circonscription == "9ème circonscription" &
                        is.na(data$nom_commune) & !is.na(data$circonscription),]
```

```
# A tibble: 1 x 11
  circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 9ème circonscription <NA>        1                 1158       21.0       79.0
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Dans la 9ème circonscription, seul un code de bureau de vote 1 est disponible, ce qui n'est pas pertinent pour notre ville. Cependant, il est très intéressant de constater que pour combler tous les trous des codes de bureau de vote de la commune, il est nécessaire qu'il y ait un ligne sans circonscription, sans nom de commune circonscription mais avec un code de bureau de vote qui corresponde, c'est-à-dire un 8, un 9 ou un 11, sinon cela veut à nouveau dire que nous avons commis une erreur de raisonnement. Regardons donc ces lignes grâce à `cir_com_na`

```
cir_com_na
```

```
# A tibble: 4 x 11
  circonscription nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>           <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 <NA>            <NA>        1                  736       20.2       79.8
2 <NA>            <NA>        11                1160       17.2       82.8
3 <NA>            <NA>        17                1186       23.4       76.6
4 <NA>            <NA>        3                  873       32.0       NA
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Nous constatons alors qu'il y a bien une ligne qui correspond, la deuxième, avec le code de bureau de vote 11. Nous pouvons donc conclure avec certitude, que cette ligne a pour nom de commune `Saint-Brevin-les-Pins`, qu'elle fait partie de la `9ème circonscription` et nous pouvons même compléter les coordonnées géographiques. Toutefois, comme précédemment, pour les codes 8 et 9, nous ne pouvons rien conclure car il n'est pas possible de les distinguer l'un de l'autre.

```
# ----- data -----

index_sblp_bur11 <- which(is.na(data$circonscription) &
                            is.na(data$nom_commune) &
                            data$code_bur_vote == "11")

data$circonscription[index_sblp_bur11] <- "9ème circonscription"

data$nom_commune[index_sblp_bur11] <- "Saint-Brevin-les-Pins"

data$latitude[index_sblp_bur11] <-
  com_lat$latitude[which(com_lat$nom_commune == "Saint-Brevin-les-Pins")]

data$longitude[index_sblp_bur11] <-
```

```
  com_long$longitude[which(com_long$nom_commune == "Saint-Brevin-les-Pins")]


# ----- bur_trou -----

bur_trou$nb_bureaux_distincts[
  bur_trou$nom_commune == "Saint-Brevin-les-Pins"] <- 10
bur_trou$bureaux_manquants[
  bur_trou$nom_commune == "Saint-Brevin-les-Pins"] <- "c(8,9)"



# ----- cir_com_na -----

cir_com_na <- cir_com_na[-2,]
```

Enfin, passons à la dernière commune. Nous allons procéder de la même façon.

```
# ----- Commune -----

bur_trou_cir_pb_1[4,]
```

```
# A tibble: 1 x 6
  circonscription       nom_commune nb_bureaux_distincts nb_bureaux_total
  <chr>                 <chr>                      <int>            <int>
1 10ème circonscription Vertou                        17               20
# i 2 more variables: nb_na_bureaux <int>, bureaux_manquants <chr>
```

```
# 2,4,6,13 ne sont pas dans cir_com_na



# ----- Bureaux disponibles dans la 10 circonscription -----

data[data$circonscription == "10ème circonscription" &
                       is.na(data$nom_commune) & !is.na(data$circonscription),]
```

```
# A tibble: 4 x 11
  circonscription       nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                 <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 10ème circonscription <NA>        1                   NA         NA         NA
2 10ème circonscription <NA>        1                  912         18.4       81.6
3 10ème circonscription <NA>        1                 1404         16.2       83.8
4 10ème circonscription <NA>        6                   NA         18.1       NA
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

```
# Seul le bureau 6 correspond, on ne peut rien dire pour les autres



# ----- Remplacer dans les différentes bases  -----

# data

index_vertou_bur6 <- which(data$circonscription == "10ème circonscription" &
                             is.na(data$nom_commune) &
                             data$code_bur_vote == "6")

data$nom_commune[index_vertou_bur6] <- "Vertou"

data$latitude[index_vertou_bur6] <-
  com_lat$latitude[which(com_lat$nom_commune == "Vertou")]

data$longitude[index_vertou_bur6] <-
  com_long$longitude[which(com_long$nom_commune == "Vertou")]


# ----- bur_trou -----

bur_trou$nb_bureaux_distincts[bur_trou$nom_commune == "Vertou"] <- 18
bur_trou$bureaux_manquants[bur_trou$nom_commune == "Vertou"] <- "c(2, 4, 13)"
```

Nous avons donc modifié au maximum les lignes par cette méthode. Maintenant, toujours dans le but de compléter notre base, et notamment les variables que nous venons d'utiliser. Reprenons la base `bur_trou_cir_pb` mais intéressons nous maintenant seulement à toutes les variables à qui il ne manque, théoriquement qu'un bureau, et qui n'ont donc qu'un 0 ou 1 dans `nb_na_bureaux` (aucune ligne dans la colonne `nb_na_bureaux` n'a de valeur supérieure à son nombre de codes de bureau de vote manquants) et regroupons les par circonscription. Avant de commencer, il est aussi nécessaire de redéfinir `bur_trou_cir_pb` car les valeurs de `bur_trou` ont été modifié par la méthode précédente.

```
bur_trou_cir_pb <- bur_trou[!(bur_trou$circonscription %in% cir_pb),]

bur_trou_cir_pb_2_1 <-
  bur_trou_cir_pb[
    bur_trou_cir_pb$circonscription == "1ère circonscription" &
    bur_trou_cir_pb$nb_na_bureaux <= 1 &
      !is.na(bur_trou_cir_pb$circonscription),]

bur_trou_cir_pb_2_2 <-
  bur_trou_cir_pb[
    bur_trou_cir_pb$circonscription == "2ème circonscription" &
    bur_trou_cir_pb$nb_na_bureaux <= 1 &
```

```
      !is.na(bur_trou_cir_pb$circonscription),]

# On ne fait pas la troisième car elle avait été exclue

bur_trou_cir_pb_2_4 <-
  bur_trou_cir_pb[
    bur_trou_cir_pb$circonscription == "4ème circonscription" &
    bur_trou_cir_pb$nb_na_bureaux <= 1 &
      !is.na(bur_trou_cir_pb$circonscription),]

bur_trou_cir_pb_2_5 <-
  bur_trou_cir_pb[
    bur_trou_cir_pb$circonscription == "5ème circonscription" &
    bur_trou_cir_pb$nb_na_bureaux <= 1 &
      !is.na(bur_trou_cir_pb$circonscription),]

# 6 et 7 on aussi été exclues

bur_trou_cir_pb_2_8 <-
  bur_trou_cir_pb[
    bur_trou_cir_pb$circonscription == "8ème circonscription" &
    bur_trou_cir_pb$nb_na_bureaux <= 1 &
      !is.na(bur_trou_cir_pb$circonscription),]

bur_trou_cir_pb_2_9 <-
  bur_trou_cir_pb[
    bur_trou_cir_pb$circonscription == "9ème circonscription" &
    bur_trou_cir_pb$nb_na_bureaux <= 1 &
      !is.na(bur_trou_cir_pb$circonscription),]

bur_trou_cir_pb_2_10 <-
  bur_trou_cir_pb[
    bur_trou_cir_pb$circonscription == "10ème circonscription" &
    bur_trou_cir_pb$nb_na_bureaux <= 1 &
      !is.na(bur_trou_cir_pb$circonscription),]
```

Le but ici va être de regarder pour chaque circonscription les lignes pour lesquelles il ne manque qu'un code de bureau de vote. Si le code manquant ne fait pas partie de la colonne `code_bur_vote` de la base `cir_com_na`, qui récapitule les lignes sans circonscription, sans nom de commune mais avec un bureau de vote (rappel : il reste les codes 1, 3 et 17) et qu'il n'est pas non plus disponible dans les circonscriptions, c'est-à-dire qu'il n'y a pas de ligne avec la circonscription et le code de bureau de vote mais sans nom de commune qui corresponde, alors nous pourrons en déduire que la ligne présente dans `bur_na_com` est celle du bureau en question. De nouveau, pour mieux comprendre cela, nous allons détailler les premiers cas.

Avant de commencer, il est utile de regarder les circonscriptions où nous ne pourrons rien trouver par cette méthode. D'une part, les circonscriptions 4, 6 et 7 ont été exclues, de ce fait, nous ne

pourrons pas compléter la colonne `circonscription` avec cette méthode. D'autre part, nous allons regarder si les différentes bases que nous avons créées juste au-dessus contiennent des lignes car il est possible qu'aucune ligne dans chacune des circonscription ne respecte les critères que nous avons imposés.

```
nrow(bur_trou_cir_pb_2_1)[1]
```

```
[1] 0
```

```
nrow(bur_trou_cir_pb_2_2)[1]
```

```
[1] 0
```

```
nrow(bur_trou_cir_pb_2_4)[1]
```

```
[1] 2
```

```
nrow(bur_trou_cir_pb_2_5)[1]
```

```
[1] 5
```

```
nrow(bur_trou_cir_pb_2_8)[1]
```

```
[1] 2
```

```
nrow(bur_trou_cir_pb_2_9)[1]
```

```
[1] 14
```

```
nrow(bur_trou_cir_pb_2_10)[1]
```

```
[1] 6
```

Au final, les circonscriptions 1 et 2 sont également exclues car aucune ligne de respecte les critères imposés dans celles-ci. Nous allons maintenant analyser les circonscriptions dans l'ordre, en commençant donc par la quatrième.

```
#----- Code bureau manquants -----

bur_trou_cir_pb_2_4
```

```
# A tibble: 2 x 6
  circonscription      nom_commune         nb_bureaux_distincts nb_bureaux_total
  <chr>                <chr>                              <dbl>            <int>
1 4ème circonscription Bouaye                                 6                6
2 4ème circonscription Saint-Aignan-Grand~                    3                4
# i 2 more variables: nb_na_bureaux <dbl>, bureaux_manquants <chr>
```

```
#----- Codes bureau disponibles -----

data[data$circonscription == "4ème circonscription" &
                       is.na(data$nom_commune) &
                       !is.na(data$circonscription),] #utilisé sinon renvoie NA
```

```
# A tibble: 1 x 11
  circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 4ème circonscription <NA>        3                 1028       17.0       83.0
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Dans la 4ème circonscription nous retrouvons Bouaye que nous avions vu précédemment. Bien qu'il pourrait être tentant ici de dire que la ligne de Saint-Aignan-Grandlieu dans `bur_na_com` correspond au code de bureau de vote 3 et que la ligne dans la 4ème circonscription avec le code de bureau de vote 3 est celle de Bouaye, cela pourrait s'avérer faux car il ne faut pas oublier qu'il existe une ligne dans la base de données qui n'a ni circonscription ni nom de commune mais qui a un code de bureau de vote 3. De ce fait, nous ne pouvions rien conclure ici. Passons donc à la circonscription suivante

```
#----- Code bureau manquants -----

bur_trou_cir_pb_2_5
```

```
# A tibble: 5 x 6
  circonscription      nom_commune         nb_bureaux_distincts nb_bureaux_total
  <chr>                <chr>                              <dbl>            <int>
1 5ème circonscription Couffé                                 1                2
2 5ème circonscription Grandchamps-des-Fo~                    3                4
3 5ème circonscription Nort-sur-Erdre                         8                9
4 5ème circonscription Sucé-sur-Erdre                         5                6
5 5ème circonscription Treillières                            7                8
# i 2 more variables: nb_na_bureaux <dbl>, bureaux_manquants <chr>
```

```
#----- Codes bureau disponibles -----

data[data$circonscription == "5ème circonscription" &
                       is.na(data$nom_commune) & !is.na(data$circonscription),]
```

```
# A tibble: 0 x 11
# i 11 variables: circonscription <chr>, nom_commune <chr>,
#   code_bur_vote <chr>, inscrits <dbl>, tx_absents <dbl>, tx_votants <dbl>,
#   tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>, latitude <dbl>,
#   longitude <dbl>
```

Dans cette circonscription, la conclusion est bien plus immédiate. En effet, comme il n'y a pas de ligne sans nom de commune mais avec une circonscription et qu'il n'y a pas non plus de codes de bureau de vote problématique (pas de 1,3 ou 17), nous pouvons directement en conclure que chaque ligne dans `bur_na_com` correspond au bureau de vote manquant. Nous pouvons donc modifier les données en conséquence

```r
# ----- Couffé -----

# data

index_couffe_bur2 <- which(data$circonscription == "5ème circonscription" &
                           data$nom_commune == "Couffé" &
                           is.na(data$code_bur_vote))

data$code_bur_vote[index_couffe_bur2] <- 2


#  bur_trou

bur_trou <- bur_trou[-which(bur_trou$nom_commune == "Couffé"),]



# ----- Grandchamps-des-Fontaines -----

index_gdf_bur4 <- which(data$circonscription == "5ème circonscription" &
                           data$nom_commune == "Grandchamps-des-Fontaines" &
                           is.na(data$code_bur_vote))

data$code_bur_vote[index_gdf_bur4] <- 4


#  bur_trou

bur_trou <- bur_trou[
              -which(bur_trou$nom_commune == "Grandchamps-des-Fontaines"),]



# ----- Nort-sur-Erdre -----
```

```r
index_nse_bur2 <- which(data$circonscription == "5ème circonscription" &
                          data$nom_commune == "Nort-sur-Erdre" &
                          is.na(data$code_bur_vote))

data$code_bur_vote[index_nse_bur2] <- 2


#  bur_trou

bur_trou <- bur_trou[-which(bur_trou$nom_commune == "Nort-sur-Erdre"),]



# ----- Sucé-sur-Erdre -----

index_sse_bur5 <- which(data$circonscription == "5ème circonscription" &
                          data$nom_commune == "Sucé-sur-Erdre" &
                          is.na(data$code_bur_vote))

data$code_bur_vote[index_sse_bur5] <- 5


#  bur_trou

bur_trou <- bur_trou[-which(bur_trou$nom_commune == "Sucé-sur-Erdre"),]



# ----- Treillières -----

index_treilleres_bur4 <- which(data$circonscription == "5ème circonscription" &
                          data$nom_commune == "Treillières" &
                          is.na(data$code_bur_vote))

data$code_bur_vote[index_treilleres_bur4] <- 4


#  bur_trou

bur_trou <- bur_trou[-which(bur_trou$nom_commune == "Treillières"),]
```

Nous pouvons donc passer à la circonscription suivante, c'est-à-dire la 8ème.

```r
#----- Code bureau manquants -----

bur_trou_cir_pb_2_8
```

```
# A tibble: 2 x 6
  circonscription      nom_commune nb_bureaux_distincts nb_bureaux_total
  <chr>                <chr>                      <dbl>            <int>
1 8ème circonscription Prinquiau                      1                2
2 8ème circonscription Trignac                        5                6
# i 2 more variables: nb_na_bureaux <dbl>, bureaux_manquants <chr>
```

```
#----- Codes bureau disponibles -----

data[data$circonscription == "8ème circonscription" &
                        is.na(data$nom_commune) & !is.na(data$circonscription),]
```

```
# A tibble: 2 x 11
  circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 8ème circonscription <NA>        1                  981       19.2       80.8
2 8ème circonscription <NA>        43                  NA       33.2       NA
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Dans celle-ci, nous pouvons constater qu'il manque un code de bureau de vote 1 pour la commune de Prinquiau. Pour les mêmes raisons que celles évoquées précédemment, nous ne pourrons rien en conclure. Cependant, pour Trignac, comme le 2 ne fait pas parti de la liste, nous pouvons. De plus, il est intéressant de constater qu'il y a un code de bureau de vote 43 sans nom de commune dans la circonscription. Un numéro aussi haut est réservé aux villes très grandes, du moins à l'échelle du département, de ce fait, en listant les communes de la 6ème circonscription, nous pourrions potentiellement trouver à qui ce bureau appartient.

```
unique(data$nom_commune[data$circonscription == "8ème circonscription" & !is.na(data$nom_commu
```

```
 [1] "Bouée"                NA                     "Campbon"
 [4] "La Chapelle-Launay"   "Donges"               "Lavau-sur-Loire"
 [7] "Malville"             "Montoir-de-Bretagne"  "Prinquiau"
[10] "Saint-Malo-de-Guersac" "Saint-Nazaire"        "Savenay"
[13] "Trignac"
```

Après consultation, il ressort que le bureau 43 appartient de façon quasi-certaine à Saint-Nazaire. Cela s'explique par le fait que toutes les autres villes sont bien trop petites pour avoir autant de bureux de votes. En effet, si nous consdèrons que la population médiane par bureau de vote est de 956 personnes, que nous arrondissons cela à 1000, aucune des communes listées, hormis Saint-Nazaire abrite plus de 43000 habitants. Nous pourrions nous poser la question de savoir si la commune `NA` ne pourrait pas être Nantes, mais cela est impossible de part la géographie des circonscriptions. Au final, nous pouvons effectuer deux modifications dans notre base de données.

```r
# ----- Trignac -----

index_trignac_bur2 <- which(data$circonscription == "8ème circonscription" &
                            data$nom_commune == "Trignac" &
                            is.na(data$code_bur_vote))

data$code_bur_vote[index_trignac_bur2] <- 2


#  bur_trou

bur_trou <- bur_trou[-which(bur_trou$nom_commune == "Trignac"),]



# ----- Saint-Nazaire -----

index_stnazaire_bur43 <- which(data$circonscription == "8ème circonscription" &
                                 is.na(data$nom_commune) &
                                 data$code_bur_vote == 43)

data$nom_commune[index_stnazaire_bur43] <- "Saint-Nazaire"

data$latitude[index_stnazaire_bur43] <-
  com_lat$latitude[which(com_lat$nom_commune == "Saint-Nazaire")]

data$longitude[index_stnazaire_bur43] <-
  com_long$longitude[which(com_long$nom_commune == "Saint-Nazaire")]


#  bur_trou

bur_trou$nb_bureaux_distincts[bur_trou$nom_commune == "Saint-Nazaire"] <- 39
bur_trou$bureaux_manquants[
  bur_trou$nom_commune == "Saint-Nazaire"] <-
  "c(2, 6, 11, 15, 25, 26, 31, 32, 46, 49)"

# Note : parfois je suis obligé de décaler les fonctions pas très proprement
# mais si je ne le fais pas ça ne passe pas dans le rendu pdf de Quarto
```

Passons ensuite à la 9ème circonscription.

```r
#----- Code bureau manquants -----

bur_trou_cir_pb_2_9
```

```
# A tibble: 14 x 6
```

```
   circonscription      nom_commune        nb_bureaux_distincts nb_bureaux_total
   <chr>                <chr>                             <dbl>            <int>
 1 9ème circonscription Corcoué-sur-Logne                     1                2
 2 9ème circonscription Corsept                               1                2
 3 9ème circonscription La Montagne                           4                5
 4 9ème circonscription La Plaine-sur-Mer                     3                4
 5 9ème circonscription Le Pellerin                           3                4
 6 9ème circonscription Legé                                  2                3
 7 9ème circonscription Machecoul-Saint-M~                    6                7
 8 9ème circonscription Paulx                                 1                2
 9 9ème circonscription Rouans                                2                3
10 9ème circonscription Saint-Colomban                        1                2
11 9ème circonscription Saint-Hilaire-de-~                    1                2
12 9ème circonscription Saint-Lumine-de-C~                    1                2
13 9ème circonscription Sainte-Pazanne                        4                5
14 9ème circonscription Villeneuve-en-Retz                    3                4
# i 2 more variables: nb_na_bureaux <dbl>, bureaux_manquants <chr>
```

```r
#----- Codes bureau disponibles -----

data[data$circonscription == "9ème circonscription" &
                       is.na(data$nom_commune) & !is.na(data$circonscription),]
```

```
# A tibble: 1 x 11
  circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 9ème circonscription <NA>        1                 1158       21.0       79.0
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Comme cette tâche est plutôt fastidieuse, nous allons automatiser cela à l'aide d'une boucle.

```r
# Liste des numéros de bureaux à exclure

exclude_bureaux <- c(1, 3, 17) #numéros problématiques + bureau disponible

# Parcourir chaque ligne de `bur_trou_cir_pb_2_9`
for (i in seq_len(nrow(bur_trou_cir_pb_2_9))) {
  # Extraire les informations de la ligne actuelle
  commune <- bur_trou_cir_pb_2_9$nom_commune[i]
  circonscription <- bur_trou_cir_pb_2_9$circonscription[i]
  bureaux_manquants <-
    eval(parse(text = bur_trou_cir_pb_2_9$bureaux_manquants[i]))

  # Exclure si le numéro de bureau est dans la liste
  bureaux_restants <- setdiff(bureaux_manquants, exclude_bureaux)
```

```
  # Passer à la ligne suivante si aucun bureau restant
  if (length(bureaux_restants) == 0) next

  # Sinon, mise à jour des données pour chaque bureau restant
  for (bureau in bureaux_restants) {
    index <- which(data$circonscription == circonscription &
                   data$nom_commune == commune &
                   is.na(data$code_bur_vote))
    data$code_bur_vote[index] <- bureau
  }

  # Mise à jour de `bur_trou` : supprimer la ligne correspondante
  bur_trou <- bur_trou |> filter(!(nom_commune == commune))
}
```

Puis, finalement, passons à la 10ème circonscription.

```
#----- Code bureau manquants -----

bur_trou_cir_pb_2_10
```

```
# A tibble: 6 x 6
  circonscription       nom_commune    nb_bureaux_distincts nb_bureaux_total
  <chr>                 <chr>                         <dbl>            <int>
1 10ème circonscription Haute-Goulaine                    3                4
2 10ème circonscription Le Bignon                         2                3
3 10ème circonscription Le Landreau                       1                2
4 10ème circonscription Monnières                         1                2
5 10ème circonscription Mouzillon                         1                2
6 10ème circonscription Vieillevigne                      2                3
# i 2 more variables: nb_na_bureaux <dbl>, bureaux_manquants <chr>
```

```
#----- Codes bureau disponibles -----

data[data$circonscription == "10ème circonscription" &
                      is.na(data$nom_commune) & !is.na(data$circonscription),]
```

```
# A tibble: 3 x 11
  circonscription       nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                 <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 10ème circonscription <NA>        1                   NA         NA         NA
2 10ème circonscription <NA>        1                  912       18.4       81.6
3 10ème circonscription <NA>        1                 1404       16.2       83.8
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

```r
# ----- Boucle -----

# Même s'il n'y a qu'une seule valeur à modifier, maintenant que nous avons
# créé une boucle autant l'utiliser

exclude_bureaux <- c(1, 3, 17)

for (i in seq_len(nrow(bur_trou_cir_pb_2_10))) {
  commune <- bur_trou_cir_pb_2_10$nom_commune[i]
  circonscription <- bur_trou_cir_pb_2_10$circonscription[i]
  bureaux_manquants <-
    eval(parse(text = bur_trou_cir_pb_2_10$bureaux_manquants[i]))

  bureaux_restants <- setdiff(bureaux_manquants, exclude_bureaux)

  if (length(bureaux_restants) == 0) next

  for (bureau in bureaux_restants) {
    index <- which(data$circonscription == circonscription &
                   data$nom_commune == commune &
                   is.na(data$code_bur_vote))
    data$code_bur_vote[index] <- bureau
  }

  bur_trou <- bur_trou |> filter(!(nom_commune == commune))
}
```

Bien que cette étape fut plutôt longue et que nous aurions sûrement pu l'automatiser avec une boucle plus tôt, nous avons souhaité tout de même donné du sens à ce que nous faisons et bien le comprendre afin, d'une part, de pouvoir bien réaliser la boucle et, d'autre part, dans le but de pouvoir bien expliquer les modifications réalisées.

Nous avons désormais réalisé presque tout ce qui est possible pour compléter les noms des communes, les codes des bureaux de vote ainsi que leurs coordonnées géographiques (latitude et longitude), de façon certaine (sans approximation), en procédant parfois au cas par cas. Avant de recourir à des méthodes probabilistes, il reste encore des cellules que nous pouvons compléter avec certitude.

4.1.4 Variables de taux

4.1.4.1 Taux d'absence et de votants

Nous allons maintenant nous concentrer sur les différentes variables de taux : `tx_absents`, `tx_votants`, `tx_blancs`, `tx_nuls` et `tx_exprimes`. Ces variables sont intuitivement reliées entre elles. Prenons par exemple la relation entre `tx_absents` et `tx_votants` : dans une population donnée, une partie des électeurs a voté et une autre s'est abstenue. Il n'y a pas d'autre alternative possible. Ainsi, pour les lignes où `tx_absents` est `NA`, nous pouvons calculer sa valeur comme la différence entre 100 (soit 100 %) et `tx_votants`, sous réserve que `tx_votants` ne soit pas

également NA. Si tx_votants est un NA, alors le résultat restera naturellement NA sans qu'il soit nécessaire d'ajouter une condition spécifique. Commençons donc par déterminer tx_absents par tx_votants

```
# VOTANTS → ABSENTS

#----- Compter les NA avant remplacement -----

sum(is.na(data$tx_absents))
```

```
[1] 232
```

```
#----- Remplacement -----

data$tx_absents[is.na(data$tx_absents)] <- 100 - data$tx_votants[is.na(data$tx_absents)]



#----- Compter les NA après remplacement -----

sum(is.na(data$tx_absents))
```

```
[1] 45
```

Cette opération nous a donc permis de compléter 187 lignes sur 232, ce qui n'est pas négligeable car le jeu de données compte 1110 lignes. Cela représente donc plus de 80% des lignes qui n'avait pas de valeur pour tx_absents qui ont été complétées, soit plus de 16% des lignes de la base entière. Maintenant, nous pouvons faire la même opération mais dans l'autre sens, c'est-à-dire compléter tx_votants grâce à tx_absents.

```
# ABSENTS → VOTANTS

#----- Compter les NA avant remplacement -----

sum(is.na(data$tx_votants))
```

```
[1] 203
```

```
#----- Remplacement -----

data$tx_votants[is.na(data$tx_votants)] <- 100 - data$tx_absents[is.na(data$tx_votants)]



#----- Compter les NA après remplacement -----

sum(is.na(data$tx_votants))
```

```
[1] 45
```

Ici, cela a permis de compléter plus de 14% des lignes de `tx_votants` dans la base.

4.1.4.2 Taux de votes blancs, nuls et exprimés

Le principe est le même pour les variables `tx_blancs`, `tx_nuls` et `tx_exprimes`. En effet, la somme de ces 3 taux par bureau de vote, autrement dit par ligne, doit être égale à 100 (toujours respondant à 100%) car il n'y a pas d'autre choix, lorsqu'une personne a voté, d'avoir voté blanc, nul ou avoir exprimé son opinion. Commençons par compléter `tx_blancs`.

```
# NULS & EXPRIMES → BLANCS

#----- Compter les NA avant remplacement -----

sum(is.na(data$tx_blancs))
```

```
[1] 202
```

```
#----- Remplacement -----

data$tx_blancs[is.na(data$tx_blancs)] <- round(
  100 - data$tx_nuls[is.na(data$tx_blancs)] -
    data$tx_exprimes[is.na(data$tx_blancs)],
  2)



#----- Compter les NA après remplacement -----

sum(is.na(data$tx_blancs))
```

```
[1] 76
```

Logiquement, il y a moins de remplacement que dans le cas précédent car pour pouvoir calculer `tx_blancs`, il est cette fois nécessaire que les 2 autres variables (`tx_nuls` et `tx_exprimes`) ne soient pas des `NA`. Déterminons ensuite `tx_nuls`.

```
# BLANCS & EXPRIMES → NULS

#----- Compter les NA avant remplacement -----

sum(is.na(data$tx_nuls))
```

```
[1] 218
```

```r
#----- Remplacement -----

data$tx_nuls[is.na(data$tx_nuls)] <- round(
  100 - data$tx_blancs[is.na(data$tx_nuls)] -
    data$tx_exprimes[is.na(data$tx_nuls)],
  2)



#----- Compter les NA après remplacement -----

sum(is.na(data$tx_nuls))
```

```
[1] 82
```

Puis, enfin, nous refaisons la même opération mais pour `tx_exprimes`.

```r
#----- Compter les NA avant remplacement -----

sum(is.na(data$tx_exprimes))
```

```
[1] 237
```

```r
#----- Remplacement -----

data$tx_exprimes[is.na(data$tx_exprimes)] <- round(
  100 - data$tx_blancs[is.na(data$tx_exprimes)] -
    data$tx_nuls[is.na(data$tx_exprimes)],
  2)



#----- Compter les NA après remplacement -----

sum(is.na(data$tx_exprimes))
```

```
[1] 80
```

4.2 Remplacements approximatifs

Après cette première partie, nous allons nous attaquer aux valeurs `NA` restantes en essayant de les ajuster au plus près de la réalité grâce à des approximations. Cependant, comme mentionné lors de l'introduction, notre objectif n'est pas de remplacer systématiquement toutes les valeurs manquantes, mais uniquement celles que nous pouvons estimer avec une précision suffisante. Il est également important de noter que certaines informations resteront inaccessibles. Par exemple, nous ne pourrons pas retrouver le nom des communes pour lesquelles un seul bureau de vote est référencé sans aucune autre indication, ou encore identifier certains bureaux de vote spécifiques. Malgré ces limitations, un `NA` n'invalide pas une ligne entière. Même en présence de valeurs manquantes, ces lignes contiennent des informations utiles. Par ailleurs, certaines variables, comme le code du bureau de vote, ne sont pas essentielles pour effectuer des estimations pertinentes.

Jusqu'à présent, nous avons réussi à compléter les lignes manquantes pour toutes les variables en utilisant les relations qu'elles entretiennent entre elles. Cependant, il reste un cas particulier : la variable `inscrits`. La difficulté avec `inscrits` réside dans le fait que les variables qui auraient pu nous aider à combler ses valeurs manquantes, comme `absents`, `votants`, `blancs`, `nuls` et `exprimes`, ne sont disponibles qu'en pourcentages (taux) et non en valeurs absolues. Cela limite leur utilité directe pour recalculer précisément le nombre total d'inscrits. Pour pallier ce problème, nous allons donc adopter une approche approximative afin d'estimer les valeurs manquantes de la variable `inscrits`. Cette méthode nous permettra de réduire les vides dans les données tout en restant cohérents avec les informations disponibles.

Mais avant, nous allons sauvegarder la base de données data où nous avons effectué des remplacements supposés certains afin, à la fin, de la comparer à la base initiale (car dans notre cas nous l'avons, dans un projet réel nous n'aurions pas pu le faire).

```
data_save_2 <- data
```

4.2.1 Nombre d'inscrits

Nous pouvons maintenant continuer les modifications. Pour ce faire, nous allons déterminer la moyenne d'inscrits par commune et nous allons utiliser cette valeur pour remplacer les `NA` de la variable `inscrits` au sein de chaque commune. Cette façon de faire s'explique par le fait que, en règle générale, les bureaux de votes sont assignés afin de répartir au mieux la population dans chaque commune. De ce fait, il n'y a habituellement pas d'écarts majeurs entre-eux et donc une moyenne par commune semble intéressante.

```
#----- Compter les NA avant remplacement -----

sum(is.na(data$inscrits))
```

```
[1] 241
```

```r
#----- Calculer la moyenne par commune -----

moyennes_commune <- data |>
  group_by(nom_commune) |>
  summarise(
    moyenne_inscrits = ifelse(all(is.na(inscrits)),
                              NA,
                              mean(inscrits, na.rm = TRUE)),
    .groups = "drop"
  )



#----- Attribuer les moyennes aux NA -----

data <- data |>
  left_join(moyennes_commune, by = "nom_commune") |>  # grp moyennes par commune
  mutate(inscrits = if_else(is.na(inscrits) & !is.na(nom_commune),
                            round(moyenne_inscrits),
                            inscrits)) |>  # moyenne arrondie à l'unité
  select(-moyenne_inscrits)



#----- Compter les NA après remplacement -----

sum(is.na(data$inscrits))
```

```
[1] 23
```

Grâce à cette opération, nous sommes passés de 241 valeurs manquantes à seulement 23. Ces 23 `NA` peuvent s'expliquer par plusieurs choses. La première est qu'il n'y a pas de nom de commune sur la ligne où nous cherchons à déterminer le nombre d'inscrits. De ce fait, la valeur ne peut pas être remplacée par la valeur moyenne car nous ne savons pas à quelle commune la ligne appartient. La seconde raison est que, au sein des groupes de lignes partageant le même nom de commune, aucune n'a de données concernant le nombre d'inscrits. Par exemple, si une petite ville n'a qu'un bureau de vote et que pour celui-ci le nombre d'inscrit n'est pas indiqué, alors nous ne sommes pas en mesure de l'approximé par la méthode utilisée précédemment. Nous pourrions alors décider de voir plus large, et de grouper les lignes non plus par nom de commune, mais par circonscription. Le problème est que cela fait nettement perdre en précision. De ce fait, nous allons pour le moment laisser les `NA` et nous agirons ensuite dessus si nous trouvons cela préférable.

4.2.2 Circonscription

Avant d'utiliser des méthodes d'imputation comme `pmm` pour approximer les données manquantes, nous pouvons améliorer la fiabilité des résultats en tentant de compléter davantage notre base.

Pour cela, nous allons chercher à compléter à nouveau les circonscriptions. Notre objectif est clair : compléter les données de circonscription, car celles-ci peuvent révéler des tendances spécifiques, comme le taux d'abstention ou les votes nuls, qui sont souvent homogènes au sein d'une même circonscription. Cela s'explique par les similitudes entre les communes qui les composent.

Pour illustrer notre démarche, nous allons commencer par examiner les communes, qu'elles soient nommées ou non, qui n'ont pas de circonscription renseignée mais possèdent des coordonnées géographiques complètes.

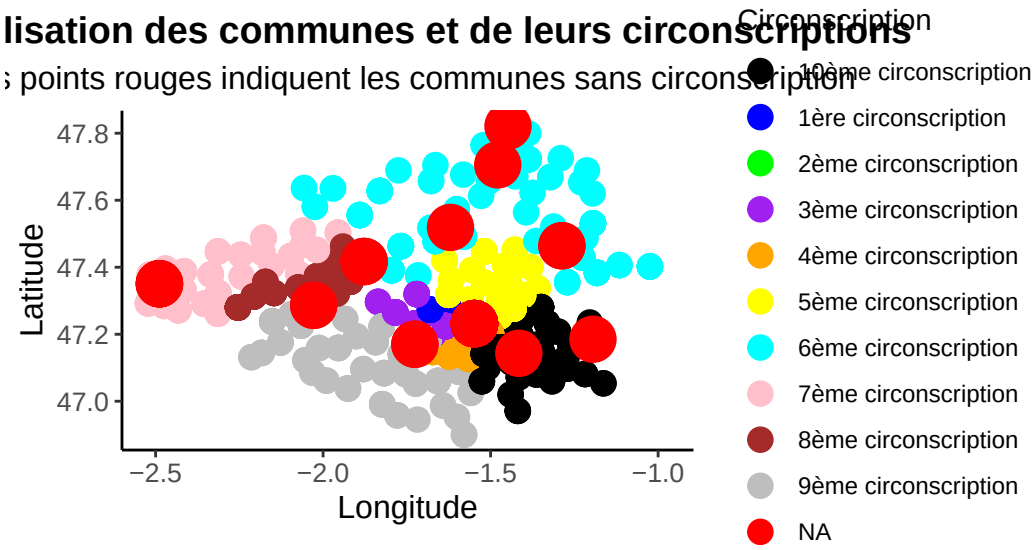
```
cir_na <- data |>
  filter(is.na(circonscription), !is.na(latitude), !is.na(longitude)) |>
  select(nom_commune, latitude, longitude) |>
  unique()

cir_na$nom_commune
```

```
 [1] "Bouvron"                  "Brains"
 [3] "Nantes"                   "Paimboeuf"
 [5] "Puceul"                   "La Regrippière"
 [7] "Saint-Aubin-des-Châteaux" "Saint-Fiacre-sur-Maine"
 [9] "Soulvache"                "Teillé"
[11] "La Turballe"
```

Nous constatons que 11 communes partagent ces caractéristiques, et que toutes ont un nom. Si nous plaçons maintenant ces 11 points sur une carte avec les lignes ayant à la fois circonscription et coordonnées géographiques, nous obtenons le dessin suivant.

```
#----- Définition du vecteur de couleurs -----

couleurs_circonscription <- c(
  "black", "blue", "green", "purple", "orange",
  "yellow", "cyan", "pink", "brown", "grey"
)



#----- Création du graphique -----

data |>
  ggplot() +
  aes(x = longitude, y = latitude, colour = factor(circonscription)) +
  geom_point(size = 4) +  # Petits points pour toutes les données (taille 4)

  # Points pour les NA
  geom_point(data = filter(data, is.na(circonscription)),
             aes(x = longitude, y = latitude),
             colour = "red",  # Couleur rouge pour les NA
```

```
            size = 8,  # Taille plus grande pour les rendre visibles
            shape = 16) +  # Forme des points

coord_sf() +
theme_classic() +

# Ajouter le titre, les sous-titres et les labels
labs(
  title = "Visualisation des communes et de leurs circonscriptions",
  subtitle = "Les points rouges indiquent les communes sans circonscription",
  x = "Longitude",
  y = "Latitude",
  colour = "Circonscription"
) +

# Personnalisation du thème
theme(
  legend.position = "right",
  plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
  plot.subtitle = element_text(hjust = 0.5, size = 12),
  axis.title = element_text(size = 12),
  axis.text = element_text(size = 10)
) +

# Appliquer la palette de couleurs, avec une couleur spéciale pour les NA
scale_colour_manual(
  values = c(couleurs_circonscription, "red"),  # Ajouter "red" pour les NA
  na.value = "red"  # Les NA seront rouges dans la légende
)
```



Lorsque nous regardons cette carte, certaines choses sautent aux yeux. Par exemple, le point rouge le plus à gauche, qui correspond à la commune de La Turballe, est bien implanté dans la 7ème circonscription et tellement à l'écart des autres qu'il semble évident que la commune fait partie de la 7ème circonscription. L'analyse est la même pour les communes de Soulvache et Saint-Aubin-des-Châteaux au nord de la carte, dans la 6ème circonscription. De ce fait, pour combler les `NA` dans la liste des communes précédemment citées, nous allons, grâce aux communes ayant une circonscription figurant sur les carte et à leurs coordonnées, déterminer le centre de chaque circonscription. Après cela, pour chaque commune sans circonscription mais avec des coordonnées complètes (`cir_na$nom_commune`), nous allons déterminer la distance entre la commune et les différents points de centre de circonscription, afin de trouver la distance la plus faible et donc d'attribuer cette circonscription à la commune.

Cette méthode présente toutefois quelques défauts. En effet, si elle marche bien pour les cas cités précédemment, elle fonctionne aussi très bien pour les communes situées au centre de la circonscription. Cependant, elle fonctionne beaucoup moins bien pour les communes proches d'une autre circonscription. En effet, si le point de centre de l'autre circoncription est plus proche de la commune que le point de sa réelle circonscription alors, la mauvaise circonscription sera attribuée. Malgré cela, nous pouvons nous conforter en nous disant que même s'il y a des erreurs, cela représente une toute petite partie des communes, car la majorité dans la base de données ont déjà une circonscription. De plus, cette variable n'est pas un élément fondamental dans le fait d'aller voter ou non, c'est un indicateur de tendance mais cela ne fait pas tout et, il y a fort à parier, qu'entre deux communes voisines mais de circonscriptions différentes, il n'y ait pas de réelle différence entre les taux de la base de données.

Comme précédemment, avant de procéder aux changements, nous allons écarter la ville de Nantes, cela se justifie par le fait que, là encore, celle-ci est présente dans plusieurs circonscriptions et donc incorporer les lignes de Nantes sans circonscription dans une seule circonscription pourrait venir fausser les analyses futures en donnant une nombre de ligne disproportionné à une seule circonscription.

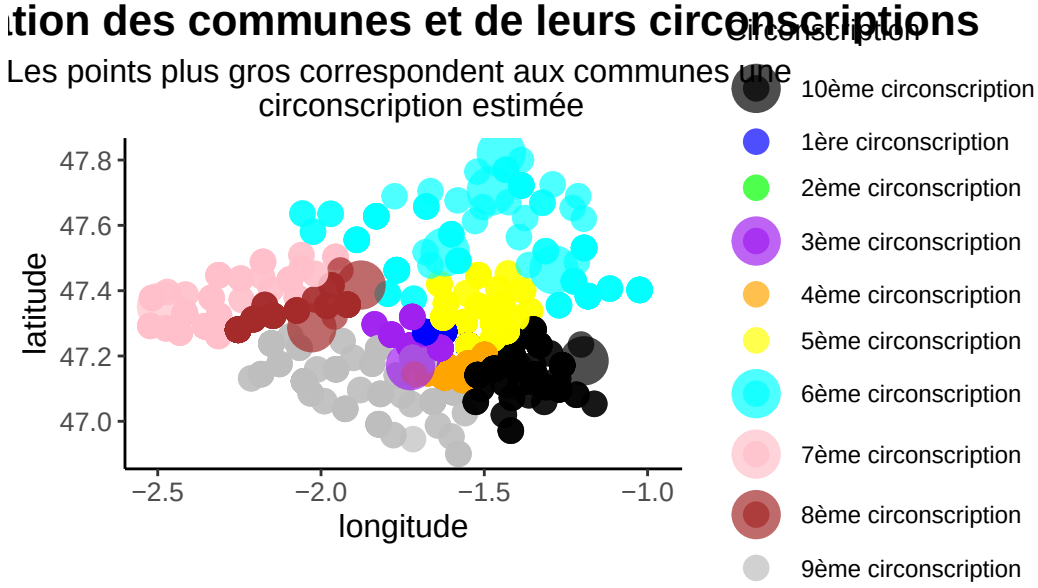
```r
#----- Obtenir les coordonnées par commune -----

# On ne combine pas juste com_lat et com_long calculés au début car les
# coordonnées ont évoluées ensuite

coordonnees_communes <- data |>
  filter(!is.na(circonscription) & !is.na(nom_commune),
         !is.na(latitude),
         !is.na(longitude)) |>
  distinct(circonscription, nom_commune, latitude, longitude)



#----- Calculer les coordonnées moyennes par circonscription -----

moyennes_circonscription <- coordonnees_communes |>
  group_by(circonscription) |>
  summarise(
    moyenne_latitude = mean(latitude, na.rm = TRUE),
    moyenne_longitude = mean(longitude, na.rm = TRUE),
    .groups = "drop"
  )


#----- Retirer Nantes des recherches de circonscription -----

# On ne la tire pas dans les étapes d'avant car lorsque la circonscription est
# bien indiquée, Nantes est utile pour calculer le centre

cir_na <- cir_na[-which(cir_na$nom_commune == "Nantes"),]



#----- Calculer la circonscription la plus proche pour chaque commune -----

cir_na <- cir_na |>
  rowwise() |>
  mutate(
    circonscription_estimee = {
      # Extraire les centroids comme matrice
      centroids_matrix <- moyennes_circonscription |>
        select(moyenne_longitude, moyenne_latitude) |>
        as.matrix()

      # Calculer les distances entre la commune et les centroids
      distances <- distHaversine(c(longitude, latitude), centroids_matrix)
```

```
      # Retourner la circonscription la plus proche
      moyennes_circonscription$circonscription[which.min(distances)]
    }
  ) |>
  ungroup()
```

Si maintenant nous reprenons la carte précédente en remplaçant la couleur des points sans circonscription par celle de la circonscription attribuée, nous obtenons la carte suivante.

```
ggplot() +
  # Petits points pour les communes avec une circonscription déjà définie
  geom_point(data = data |>
               filter(!is.na(circonscription)),
             aes(x = longitude,
                 y = latitude,
                 colour = factor(circonscription)),
             size = 4, alpha = 0.7) +

  # Gros points pour les communes avec une circonscription estimée
  geom_point(data = cir_na |>
               filter(!is.na(circonscription_estimee)),
             aes(x = longitude,
                 y = latitude,
                 colour = factor(circonscription_estimee)),
             size = 8, alpha = 0.7) +

  # Coordonnées géographiques
  coord_sf() +
  theme_classic() +

  # Ajouter le titre et ajuster la légende
  labs(
    title = "Visualisation des communes et de leurs circonscriptions",
    subtitle = "Les points plus gros correspondent aux communes une
    circonscription estimée",
    colour = "Circonscription"
  ) +

  # Personnalisation du thème
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 12),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10)
  ) +
```

```
  # Appliquer la palette de couleurs
  scale_colour_manual(
    values = c(couleurs_circonscription, "red")
  )
```



En analysant cette carte, les changements semblent plutôt cohérents et nous pouvons donc ajouter les modifications à `data`.

```
data <- data |>
  left_join(cir_na |> select(nom_commune, circonscription_estimee),
            by = "nom_commune") |>
  mutate(
    circonscription = if_else(is.na(circonscription),
                              circonscription_estimee,
                              circonscription)
  ) |>
  select(-circonscription_estimee)
```

Après cette modification, regardons combien de circonscriptions sont encore vide

```
sum(is.na(data$circonscription))
```

```
[1] 45
```

Bien que la valeur de 45 puisse sembler relativement élevée, il ne faut pas oublier que nous avons retiré Nantes de nos modifications car c'est une ville problématique à plusieurs niveau dans notre jeu de données. Regardons alors ce que cela donne sans la ville de Nantes.

```r
sum(is.na(data$circonscription) & data$nom_commune != "Nantes", na.rm = TRUE) +
  sum(is.na(data$circonscription) & is.na(data$nom_commune), na.rm = TRUE)
```

```
[1] 9
```

Au final, nous n'avons plus que 9 valeurs manquantes dans la colonne `circonscription`, ce qui est relativement faible étant donné la taille du jeu de données (environ 0.8% de données manquantes).

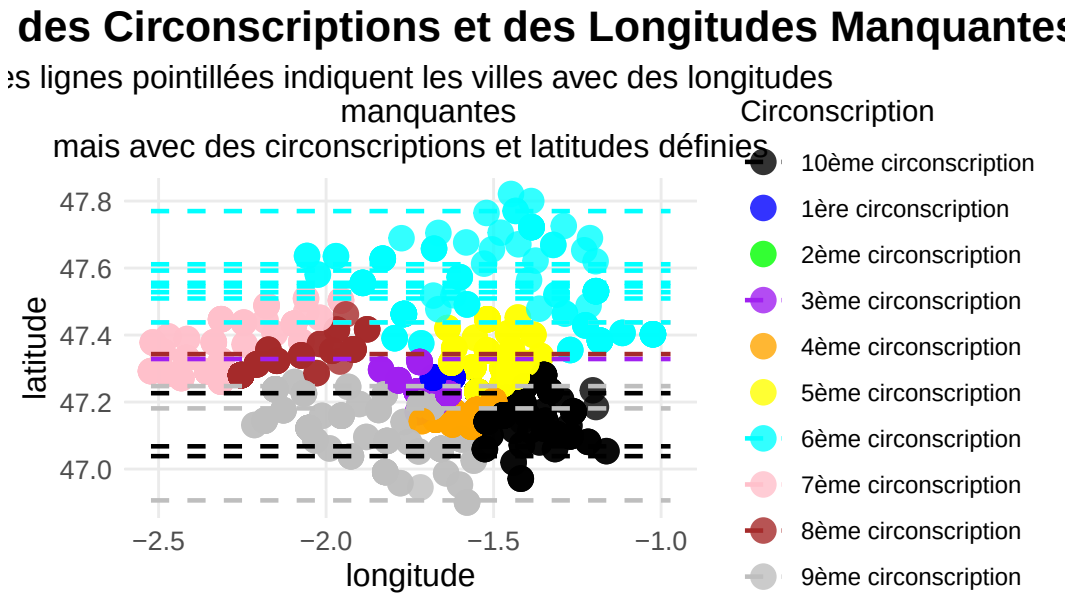
4.2.3 Coordonnées géographiques

Pour continuer avec les coordonnées géographiques et la circonscription, nous allons maintenant faire l'inverse : tenter d'estimer des coordonnées géographiques grâce aux données de circonscription lorsqu'il n'en manque qu'une. Cette étape vient après la précédente car présente une plus grande probabilités d'approximation. En effet, dans le cas précédent, pour la majorité des communes, la circonscription de rattachement semblait logique, tandis qu'estimer la longitude grâce à la latitude et la circonscription de référence est plus compliqué.

4.2.3.1 Longitude

```r
#----- Filtrer les données avec des longitudes manquantes -----

long_na_cir_lat <- data |>
  filter(
    !is.na(circonscription),
    !is.na(latitude),
    is.na(longitude)
  ) |>
  distinct(latitude, .keep_all = TRUE)



#----- Tracer la carte -----

ggplot(data |> filter(!is.na(circonscription)),
       aes(x = longitude, y = latitude, colour = factor(circonscription))) +
  geom_point(size = 4, alpha = 0.8) +
  geom_segment(
    data = long_na_cir_lat,
    aes(
      x = min(data$longitude, na.rm = TRUE),  # Début de la ligne
      xend = max(data$longitude, na.rm = TRUE),  # Fin de la ligne
      y = latitude,  # Utiliser la latitude comme position
      yend = latitude,  # Garder la même latitude
      colour = factor(circonscription)  # Coloration selon la circonscription
    ),
```

```
  linetype = "dashed",
  size = 0.8
) +
coord_sf() +
theme_minimal() +

# Ajouter le titre et personnaliser la légende
labs(
  title = "Visualisation des Circonscriptions et des Longitudes Manquantes",
  subtitle = "Les lignes pointillées indiquent les villes avec des longitudes
  manquantes\nmais avec des circonscriptions et latitudes définies",
  colour = "Circonscription"
) +

# Personnalisation des couleurs et légende
scale_colour_manual(
  values = couleurs_circonscription
) +

# Personnalisation du thème
theme(
  legend.position = "right",
  plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
  plot.subtitle = element_text(hjust = 0.5, size = 12),
  axis.title = element_text(size = 12),
  axis.text = element_text(size = 10)
)
```



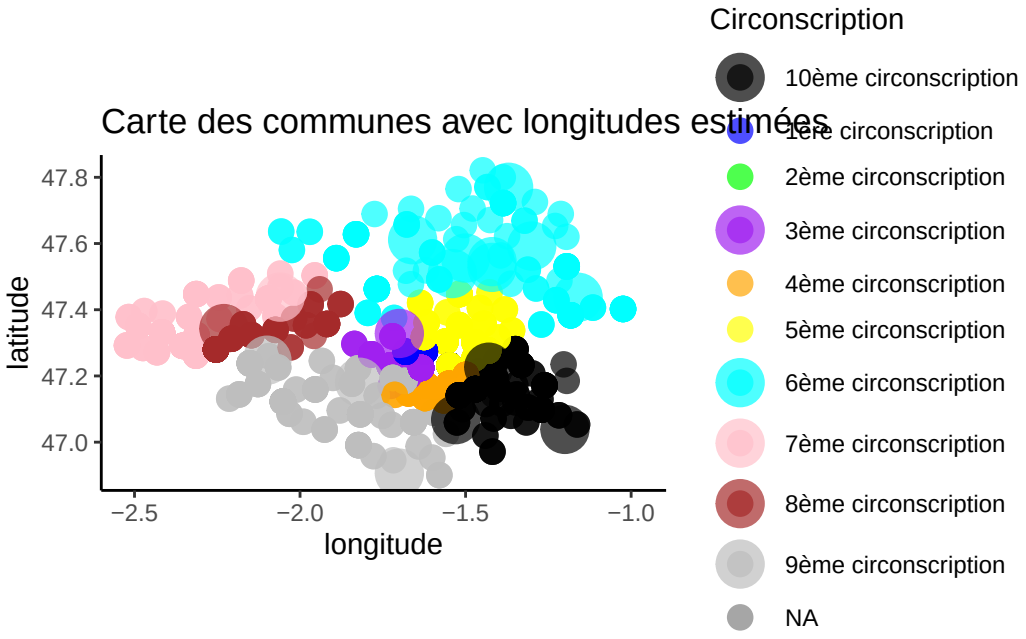
Sur cette carte, plusieurs éléments sont représentés. Tout d'abord, comme dans les cartes précédentes, les communes sont indiquées par des points, et chaque point est coloré en fonction de la circonscription à laquelle la commune appartient. Ensuite, des lignes sont ajoutées pour représenter les données de communes pour lesquelles la circonscription et la latitude sont connues, mais dont la longitude est manquante. Ainsi, une ligne est tracée à la latitude correcte, mais la longitude peut varier tout au long de la ligne, puisqu'elle est indéterminée. Cependant, étant donné que la circonscription est également connue, chaque ligne est colorée en fonction de la circonscription à laquelle elle appartient. Cette information réduit considérablement les possibilités de localisation de la longitude. Par exemple, il serait peu probable qu'une commune d'une circonscription soit située au milieu d'une autre circonscription, car les communes d'une même circonscription sont généralement proches les unes des autres.

Ainsi, bien que la longitude exacte de ces communes reste incertaine, cette méthode permet d'estimer leur position de manière plus précise en tenant compte de la proximité avec d'autres communes appartenant à la même circonscription. Cela réduit les incertitudes et permet d'éviter que les communes soient placées trop près les unes des autres. En nous basant sur ces indications, nous allons tenter d'approximer les longitudes manquantes de manière cohérente, en ajustant leur position tout en respectant les contraintes géographiques et administratives.

```
#----- Déterminer les intervalles -----

intervals <- tibble(
  row_index = 1:nrow(long_na_cir_lat),
  min_longitude = c(-2, -2.4, -2.2, -2, -2, -2.25, -2, -1.55, -2, -1.45, -1.6,
                    -2.2, -2, -1.7, -2, -1.55, -1.8),
  max_longitude = c(-1.2, -2, -1.8, -1.2, -1.2, -1.8, -1.2, -1.2, -1.2, -1.2,
                    -1.2, -1.75, -1.6, -1.6, -1.2, -1.3, -1)
)


#----- Créer une liste des longitudes déjà occupées par circonscription -----

longitudes_par_cir <- data |>
  filter(!is.na(longitude)) |>
  group_by(circonscription) |>
  summarise(
    used_longitudes = list(sort(unique(longitude)))  # Longitudes déjà utilisées
  )


#----- Joindre les intervalles à long_na_cir_lat -----

long_na_cir_lat <- long_na_cir_lat |>
  mutate(row_index = row_number())

long_na_cir_lat <- long_na_cir_lat |>
  left_join(intervals, by = "row_index")
```

```r
#----- Créer une fonction pour générer une longitude cohérente -----

generate_longitude <- function(circonscription, min_long, max_long) {
  # Récupérer les longitudes déjà utilisées pour la circonscription
  used <- longitudes_par_cir |>
    filter(circonscription == circonscription) |>
    pull(used_longitudes) |>
    unlist()

  # Créer une liste des longitudes possibles dans l'intervalle
  possible_longitudes <- seq(min_long, max_long, by = 0.02)

  # Créer une zone d'exclusion
  exclusion_zone <- unlist(
    lapply(used, function(x) seq(x - 0.15, x + 0.15, by = 0.01)))

  # Exclure les longitudes déjà utilisées et celles dans la zone d'exclusion
  available_longitudes <- setdiff(possible_longitudes, exclusion_zone)

  # S'assurer que la longitude générée est bien dans l'intervalle spécifié
  available_longitudes <-
    available_longitudes[available_longitudes >= min_long &
                           available_longitudes <= max_long]

  # Choisir une longitude disponible de manière aléatoire
  if (length(available_longitudes) > 0) {
    return(sample(available_longitudes, 1))
  } else {
    return(NA)  # Retourner NA si aucune longitude disponible
  }
}

# Mettre à jour les longitudes manquantes avec la fonction `generate_longitude`
long_na_cir_lat <- long_na_cir_lat |>
  rowwise() |>
  mutate(updated_longitude = generate_longitude(circonscription,
                                                min_longitude,
                                                max_longitude)) |>
  ungroup()



#----- Représenter les résultats -----

ggplot() +
```

```r
geom_point(data = data |> filter(!is.na(longitude)),
           aes(x = longitude, y = latitude, colour = factor(circonscription)),
           size = 4, alpha = 0.7) +  # Points pour les communes déjà définies

geom_point(data = long_na_cir_lat |> filter(!is.na(updated_longitude)),
           aes(x = updated_longitude,
               y = latitude,
               colour = factor(circonscription)),
           size = 8, alpha = 0.7) +

# Coordonnées géographiques
coord_sf() +
theme_classic() +
theme(legend.position = "right") +  # Position de la légende à droite

# Ajout des labels et du titre
labs(
  colour = "Circonscription",
  title = "Carte des communes avec longitudes estimées"
) +

# Appliquer la palette de couleurs personnalisée
scale_colour_manual(values = couleurs_circonscription)
```



Bien que cette modification ne soit pas parfaite, nous constatons qu'à chaque exécution de la fonction, les résultats restent plausibles et cohérents. Même si certains éléments, comme la détection des zones vides dans les circonscriptions, auraient pu être ajoutés pour affiner davantage le processus, cela alourdirait la fonction sans apporter une réelle amélioration en termes de données. De plus, bien

que les données évoluent à chaque exécution, cela n'affecte pas la précision globale de l'information. Par conséquent, nous pouvons mettre systématiquement à jour la base de données sans que cela n'impacte de manière significative la qualité des résultats obtenus.

```
#----- Compter les NA avant la modification -----
sum(is.na(data$longitude))
```

```
[1] 31
```

```
# ----- Modifier la base -----

data <- data |>
  left_join(long_na_cir_lat |> select(latitude,
                                      updated_longitude),
            by = "latitude")

data <- data |>
  mutate(longitude = ifelse(is.na(longitude), updated_longitude, longitude)) |>
  select(-updated_longitude)



#----- Compter les NA après modification -----

sum(is.na(data$longitude))
```

```
[1] 12
```

Au final, bien que ce changement puisse sembler un peu superflux, nous réduisons de 19 le nombres de valeurs manquantes, soit environ deux tiers de ce qu'il restait.

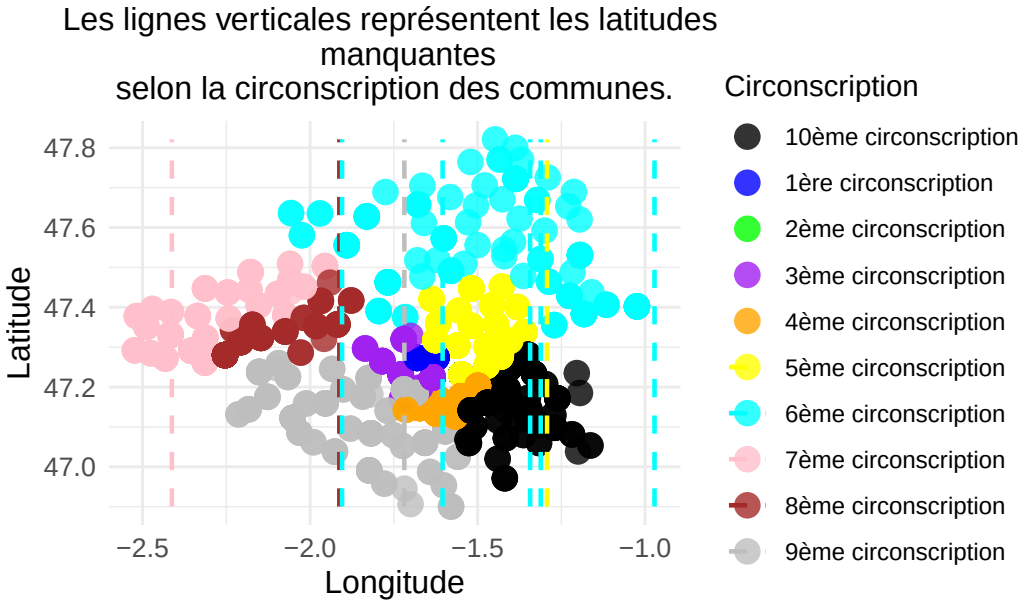
4.2.3.2 Latitude

Nous allons donc faire exactement la même chose mais cette fois pour la latitude.

```
#----- Filtrer les données avec des longitudes manquantes -----

lat_na_cir_long <- data |>
  filter(
    !is.na(circonscription),
    is.na(latitude),
    !is.na(longitude)
  ) |>
  distinct(longitude, .keep_all = TRUE)
```

```r
#----- Tracer la carte -----

ggplot(data |> filter(!is.na(circonscription)),
       aes(x = longitude, y = latitude, colour = factor(circonscription))) +
  geom_point(size = 4, alpha = 0.8) +
  geom_segment(
    data = lat_na_cir_long,
    aes(
      x = longitude,
      xend = longitude,
      y = min(data$latitude, na.rm = TRUE),
      yend = max(data$latitude, na.rm = TRUE),
      colour = factor(circonscription)
    ),
    linetype = "dashed",
    size = 0.8
  ) +
  scale_colour_manual(values = couleurs_circonscription) +
  theme_minimal() +
  labs(
    title = "Visualisation des circonscriptions avec latitudes manquantes",
    subtitle = "Les lignes verticales représentent les latitudes
    manquantes \nselon la circonscription des communes.",
    x = "Longitude",
    y = "Latitude",
    colour = "Circonscription"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 12),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10),
    legend.position = "right"
  )
```

ation des circonscriptions avec latitudes manquantes

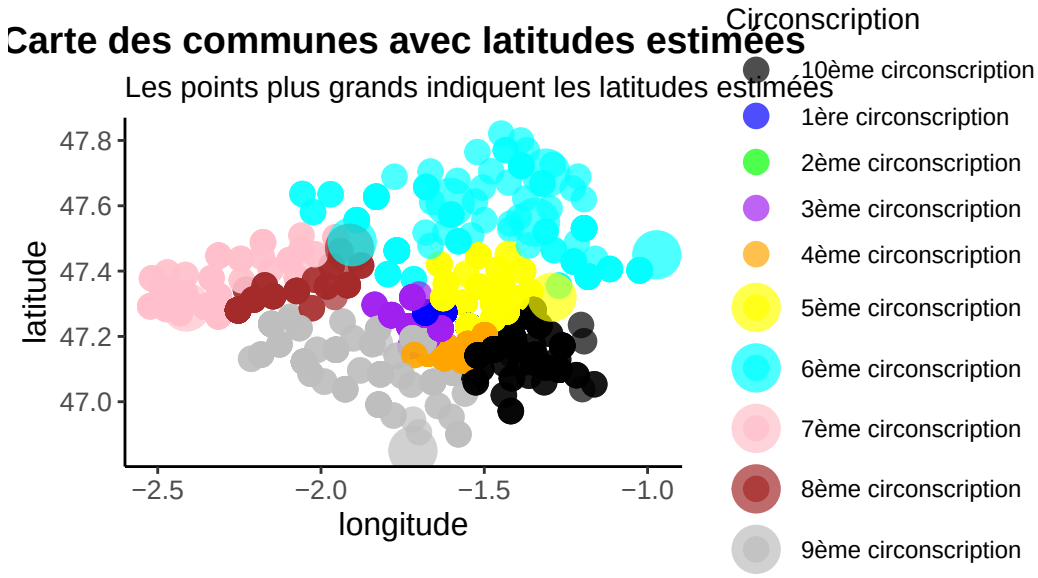


Cette fois-ci, et de manière logique, les lignes sont verticales car c'est l'information sur la latitude qui manque. Contrairement à l'exemple précédent, nous avons ici la longitude exacte pour ces communes, ce qui signifie que leur latitude réelle se situe nécessairement quelque part le long des traits verticaux. Nous retrouvons ici la même idée que précédemment : les couleurs des lignes et des points, correspondant aux circonscriptions, nous aident à réduire les intervalles possibles pour la latitude. Une commune de la circonscription 1, par exemple, ne devrait pas se retrouver dans une zone où les points d'une autre circonscription prédominent. Cette méthode nous permet donc, une fois encore, d'estimer les valeurs manquantes de façon plausible et cohérente avec les données disponibles.

```
#----- Créer les intervalles pour les latitudes -----

latitude_intervals <- tibble(
  row_index = 1:nrow(lat_na_cir_long),
  min_latitude = c(47.27,47.25,47.3,47.3,47.45,46.85,47.47,47.35,47.35),
  max_latitude = c(47.4,47.5,47.46,47.8,47.75,47.15,47.65,47.45,47.8)
)



#----- Créer une liste des latitudes déjà occupées par circonscription -----

latitudes_par_cir <- data |>
  filter(!is.na(latitude)) |>
  group_by(circonscription) |>
  summarise(
    used_latitudes = list(sort(unique(latitude)))  # Latitudes déjà utilisées
  )
```

```r
#----- Joindre les intervalles à lat_na_cir_long -----

lat_na_cir_long <- lat_na_cir_long |>
  mutate(row_index = row_number())

lat_na_cir_long <- lat_na_cir_long |>
  left_join(latitude_intervals, by = "row_index")

# ----- Créer une fonction pour générer une latitude cohérente -----

generate_latitude <- function(circonscription, min_lat, max_lat) {
  # Récupérer les latitudes déjà utilisées pour la circonscription
  used <- latitudes_par_cir |>
    filter(circonscription == circonscription) |>
    pull(used_latitudes) |>
    unlist()

  # Créer une liste des latitudes possibles
  possible_latitudes <- seq(min_lat, max_lat, by = 0.02)

  # Créer une zone d'exclusion
  exclusion_zone <- unlist(
    lapply(used, function(x) seq(x - 0.5, x + 0.5, by = 0.1)))

  # Exclure les latitudes déjà utilisées et celles dans la zone d'exclusion
  available_latitudes <- setdiff(possible_latitudes, exclusion_zone)

  # S'assurer que la latitude générée est bien dans l'intervalle spécifié
  available_latitudes <- available_latitudes[available_latitudes >= min_lat &
                                               available_latitudes <= max_lat]

  # Choisir une latitude disponible de manière aléatoire
  if (length(available_latitudes) > 0) {
    return(sample(available_latitudes, 1))
  } else {
    return(NA)  # Retourner NA si aucune latitude disponible
  }
}

# Mettre à jour les latitudes manquantes avec la fonction `generate_latitude`
lat_na_cir_long <- lat_na_cir_long |>
  rowwise() |>
  mutate(updated_latitude = generate_latitude(circonscription,
                                              min_latitude,
                                              max_latitude)) |>
```

```r
  ungroup()


#----- Représenter les résultats -----

ggplot() +
  # Petits points pour les communes avec une latitude définie
  geom_point(data = data |> filter(!is.na(latitude) & !is.na(circonscription)),
             aes(x = longitude, y = latitude, colour = factor(circonscription)),
             size = 4, alpha = 0.7) +  # Points standards

  # Gros points pour les communes avec une latitude estimée
  geom_point(data = lat_na_cir_long |> filter(!is.na(updated_latitude) &
                                                !is.na(circonscription)),
             aes(x = longitude,
                 y = updated_latitude,
                 colour = factor(circonscription)),
             size = 8, alpha = 0.7) +  # Points plus gros pour les estimations

  # Configuration des coordonnées géographiques
  coord_sf() +

  # Application d'un thème et des étiquettes
  theme_classic() +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10)
  ) +
  labs(
    colour = "Circonscription",
    title = "Carte des communes avec latitudes estimées",
    subtitle = "Les points plus grands indiquent les latitudes estimées"
  ) +

  # Application de la palette de couleurs
  scale_colour_manual(values = couleurs_circonscription)
```



De manière similaire à la méthode utilisée précédemment, bien que cette approche ne permette pas une précision absolue, elle nous offre une estimation raisonnable des latitudes manquantes. Ces estimations peuvent alors être intégrées directement dans la base `data` pour enrichir les informations disponibles.

```
#----- Compter les NA avant la modification -----

sum(is.na(data$latitude))
```

```
[1] 22
```

```
#----- Modifier -----


data <- data |>
  left_join(lat_na_cir_long |> select(longitude,
                                      updated_latitude),
            by = "longitude")

data <- data |>
  mutate(latitude = ifelse(is.na(latitude), updated_latitude, latitude)) |>
  select(-updated_latitude)


#----- Compter les NA après la modification -----

sum(is.na(data$latitude))
```

```
[1] 12
```

4.2.4 Variables de taux

Maintenant que nous avons estimé les données géographiques manquantes, nous pouvons nous concentrer sur l'imputation des `NA` dans les différents taux. La première étape consiste à isoler les données à imputer et à examiner le nombre de valeurs manquantes pour chacune. Il est important de souligner qu'il est inutile d'estimer simultanément `tx_absents` et `tx_votants`. En effet, en imputant uniquement l'un des deux, la valeur de l'autre peut être calculée par complémentarité. De manière similaire, pour les variables `tx_blancs`, `tx_nuls`, et `tx_exprimes`, il est plus judicieux de traiter ces taux comme un groupe cohérent. Cela permet d'éviter qu'une imputation directe conduise à une somme dépassant légèrement 100%, ce qui serait incohérent.

4.2.4.1 PMM

Nous allons donc procéder étape par étape en réalisant les imputations nécessaires. Commençons par examiner les données et choisir la meilleure approche pour chaque groupe de variables.

```
#----- Créer un dataframe avec les colonnes nécessaires -----

data_a_imputer <- data |>
  select(circonscription,
         nom_commune,
         inscrits,
         tx_absents,
         tx_votants,
         tx_blancs,
         tx_nuls,
         tx_exprimes)



#----- Appliquer la méthode PMM -----

imputation <- mice(data, method = "pmm", m = 5, maxit = 50, seed = 500)


#----- Résumer de l'imputation -----

summary(imputation)


#----- Obtenir le jeu de données imputé -----

imputed_data <- complete(imputation, action = 1)
```

4.2.4.2 Taux d'absence et de votants

Grâce à l'imputation réalisée précédemment à l'aide de la méthode Predictive Mean Matching (PMM), nous avons pu estimer les valeurs manquantes pour `tx_absents`. Cette méthode permet d'obtenir des estimations fiables en s'appuyant sur des données similaires tout en respectant la distribution des variables d'origine. Maintenant que nous disposons des estimations pour `tx_absents`, il devient possible de calculer directement les valeurs de `tx_votants`. En effet, ces deux taux sont complémentaires et leur somme est toujours égale à 100%. Ainsi, pour chaque ligne :

`tx_votants = 100 - tx_absents`

Ce calcul garantit la cohérence des données, tout en exploitant les imputations déjà effectuées. Nous pouvons désormais mettre à jour la base de données pour inclure les taux de votants calculés, complétant ainsi cette partie de l'analyse.

```
#----- Vérifier si toutes les données ont été remplacées -----

sum(is.na(imputed_data$tx_absents))
```

```
[1] 0
```

```
#----- Modifier colonne data$tx_absents par imputed_data$tx_absents si NA -----

data$tx_absents[is.na(data$tx_absents)] <- imputed_data$tx_absents[
  is.na(data$tx_absents)]



#----- Vérifier qu'il n'y a plus de NA dans tx_absents de data -----

sum(is.na(data$tx_absents))
```

```
[1] 0
```

```
#----- Déterminer le taux de votants pour les NA -----

data$tx_votants[is.na(data$tx_votants)] <- 100 - data$tx_absents[
  is.na(data$tx_votants)]



#----- Vérifier qu'il n'y a plus de NA dans tx_votants de data -----

sum(is.na(data$tx_votants))
```

```
[1] 0
```

4.2.4.3 Taux de votes blancs, nuls et exprimés

Avec les taux de votants désormais calculés, nous pouvons nous concentrer sur l'estimation des valeurs manquantes pour `tx_blancs`, `tx_nuls` et `tx_exprimes`. Ces trois variables, bien qu'interconnectées, nécessitent une attention particulière pour garantir la cohérence des données, notamment que leur somme reste toujours égale à 100%.

```
#----- Vérifier si toutes les données ont été remplacées pour tx_blancs -----

sum(is.na(imputed_data$tx_blancs))
```

```
[1] 0
```

```
#-----  Modifier colonne data$tx_blancs par imputed_data$tx_blancs si NA -----

data$tx_blancs[is.na(data$tx_blancs)] <-
  imputed_data$tx_blancs[is.na(data$tx_blancs)]



#----- Vérifier qu'il n'y a plus de NA dans tx_blancs de data -----

sum(is.na(data$tx_blancs))
```

```
[1] 0
```

```
#----- Déterminer tx_nuls et tx_exprimes dans data -----

data$tx_nuls[is.na(data$tx_nuls)] <- 100 - data$tx_blancs[is.na(data$tx_nuls)] - data$tx_expri
data$tx_exprimes[is.na(data$tx_exprimes)] <- 100 -
  data$tx_blancs[is.na(data$tx_exprimes)] -
  data$tx_nuls[is.na(data$tx_exprimes)]



#----- Vérifier si toutes les données ont été remplacées pour tx_nuls -----

sum(is.na(imputed_data$tx_nuls))
```

```
[1] 0
```

```
#----- Modifier la colonne data$tx_nuls par imputed_data$tx_nuls si NA -----

data$tx_nuls[is.na(data$tx_nuls)] <- imputed_data$tx_nuls[is.na(data$tx_nuls)]
```

```r
#----- Vérifier qu'il n'y a plus de NA dans tx_nuls de data -----

sum(is.na(data$tx_nuls))
```

```
[1] 0
```

```r
#----- Déterminer les derniers tx_exprimes dans data -----

data$tx_exprimes[is.na(data$tx_exprimes)] <- 100 -
  data$tx_blancs[is.na(data$tx_exprimes)] -
  data$tx_nuls[is.na(data$tx_exprimes)]



#----- Vérifier qu'il n'y a plus de NA dans tx_exprimes de data -----

sum(is.na(data$tx_exprimes))
```

```
[1] 0
```

```r
data$tx_nuls <- round(data$tx_nuls,2)

# Je ne sais pas trop pourquoi, sans ça le taux s'affiche avec trop de décimales
# (comme nous avons défini à beaucoup)
```

Au terme de nos imputations, nous obtenons un tableau globalement bien complété. Nous avons délibérément choisi de ne pas tenter de remplir certaines cases restantes, car cela aurait impliqué un niveau d'approximation trop élevé. En effet, il est souvent préférable de conserver un `NA` plutôt que d'introduire une valeur peu fiable qui pourrait biaiser les analyses ultérieures.

5 Conclusion

5.1 Conclusion de la base de donnée

Pour finaliser notre travail, examinons les lignes qui demeurent incomplètes et réfléchissons à leur impact sur l'interprétation des données.

```
na_count <- rowSums(is.na(data))
data[na_count >= 5, ]
```

```
# A tibble: 0 x 11
# i 11 variables: circonscription <chr>, nom_commune <chr>,
#   code_bur_vote <chr>, inscrits <dbl>, tx_absents <dbl>, tx_votants <dbl>,
#   tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>, latitude <dbl>,
#   longitude <dbl>
```

Il n'y a pas de ligne avec plus de 5 données manquantes. Cela montre que la plupart des lignes du jeu de données sont relativement complètes. Cela signifie qu'aucune ligne dans le jeu de données n'est trop incomplète et doit être exclue de l'analyse.

```
na_count <- rowSums(is.na(data))
data[na_count == 4, ]
```

```
# A tibble: 3 x 11
  circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
  <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
1 3ème circonscription <NA>        <NA>               849       42.6       57.4
2 <NA>                 <NA>        17                1186       23.4       76.6
3 <NA>                 <NA>        3                  873       32.0       68.0
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Il y en a 3 avec exactement 4 données manquantes. Ces lignes contiennent un nombre relativement élevé de valeurs manquantes, mais elles restent peu nombreuses dans l'ensemble.

```
na_count <- rowSums(is.na(data))
data[na_count == 3, ]
```

```
# A tibble: 11 x 11
   circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
   <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
 1 7ème circonscription <NA>        <NA>                NA       23.2       76.8
 2 4ème circonscription <NA>        3                 1028       17.0       83.0
 3 <NA>                 Conquereuil 1                   NA       22.1       77.9
 4 10ème circonscripti~ <NA>        1                  912       18.4       81.6
```

```
 5 <NA>                  Issé         <NA>              1425       24.1       75.9
 6 <NA>                  Joué-sur-E~ 1                  852       20.3       79.7
 7 <NA>                  <NA>         1                 736       20.2       79.8
 8 6ème circonscription <NA>         1                 976       19.9       80.1
 9 7ème circonscription <NA>         2                1392       24.5       75.5
10 <NA>                  Sévérac      <NA>              1229       24.1       75.9
11 <NA>                  Sion-les-M~ <NA>              1278       24.7       75.3
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Il y en a 11 avec exactement 3 données manquantes. C'est un peu plus que précédemment, mais cela reste encore assez limité.

```
na_count <- rowSums(is.na(data))
data[na_count == 2, ]
```

```
# A tibble: 17 x 11
   circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
   <chr>                <chr>       <chr>             <dbl>      <dbl>      <dbl>
 1 10ème circonscripti~ <NA>        1                    NA       31.2       68.8
 2 6ème circonscription <NA>        1                    NA       22.3       77.7
 3 6ème circonscription Juigné-des~ <NA>                 NA       20.8       79.2
 4 5ème circonscription Mouzeil     1                  1478       23.1       76.9
 5 <NA>                 Nantes      <NA>                958       19.9       80.1
 6 <NA>                 Nantes      <NA>                920       21.0       79.0
 7 <NA>                 Nantes      <NA>                866       36.5       63.5
 8 <NA>                 Nantes      <NA>                954       19.9       80.1
 9 <NA>                 Nantes      <NA>                990       20.6       79.4
10 <NA>                 Nantes      <NA>                954       21.7       78.3
11 <NA>                 Nantes      <NA>               1032       22.2       77.8
12 <NA>                 Nantes      <NA>                985       25.0       75.0
13 <NA>                 Nantes      <NA>                949       24.1       75.9
14 6ème circonscription <NA>        1                    NA       23.5       76.5
15 6ème circonscription <NA>        <NA>                958       23.8       76.2
16 <NA>                 Le Pin      1                   552       22.3       77.7
17 6ème circonscription <NA>        <NA>                523       20.8       79.2
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Il y en a 17 avec exactement 2 données manquantes. Cela représente un nombre modéré de lignes incomplètes.

```
na_count <- rowSums(is.na(data))
data[na_count >= 1, ]
```

```
# A tibble: 279 x 11
   circonscription      nom_commune code_bur_vote inscrits tx_absents tx_votants
   <chr>                <chr>       <chr>            <dbl>      <dbl>      <dbl>
 1 6ème circonscription Ancenis-Sa~ <NA>               885       23.3       76.7
 2 9ème circonscription Chaumes-en~ <NA>               862       19.7       80.3
 3 9ème circonscription Chaumes-en~ <NA>               532       18.8       81.2
 4 9ème circonscription Chaumes-en~ <NA>               909       22.2       77.8
 5 7ème circonscription <NA>        <NA>                NA       23.2       76.8
 6 7ème circonscription Assérac     2                   NA       23.6       76.4
 7 10ème circonscripti~ Basse-Goul~ <NA>              1055       20.8       79.2
 8 10ème circonscripti~ Basse-Goul~ <NA>              1189       27.3       72.7
 9 7ème circonscription Batz-sur-M~ <NA>              1042       20.8       79.2
10 7ème circonscription Batz-sur-M~ <NA>               951       22.3       77.7
# i 269 more rows
# i 5 more variables: tx_blancs <dbl>, tx_nuls <dbl>, tx_exprimes <dbl>,
#   latitude <dbl>, longitude <dbl>
```

Il y en a 238 avec exactement une donnée manquante. C'est un nombre significatif, nous pourrions donc nous questionner sur les remplacements que nous avons fait car nous pourrions interpréter cela comme : 20% de la base n'a pas été remplacée. Cependant, pour conclure cela nous devons nous intéresser plutôt à la proportion totale de `NA` dans `data`.

```
na_count <- rowSums(is.na(data))
nrow(data[na_count >= 1, ]) / nrow(data)
```

```
[1] 0.251351351351351
```

```
na_count_save1 <- rowSums(is.na(data_save_1))
nrow(data_save_1[na_count_save1 >= 1, ]) / nrow(data_save_1)
```

```
[1] 0.916216216216216
```

En effet, dans le jeu de données contenant les valeurs manquantes non traitées, il n'y avait pas 25% de lignes avec au moins un `NA` mais presque 92%, ce qui est considérable. Il est donc plus pertinent de nous intéresser au total de données manquantes dans la base modifiée par rapport au 20% de la base initiale

```
sum(is.na(data))
```

```
[1] 327
```

```
sum(is.na(data))/(nrow(data)*ncol(data))
```

```
[1] 0.0267813267813268
```

Au final, alors qu'il manquait 20% de la base au départ, après les modifications il ne nous en manque que moins de 3% Cependant, dans beaucoup d'entre-elles, cela est dû au manque de code de bureau de vote. Comme celui-ci n'est en réalité pas très important, analysons plutôt la base sans tenir compte de cette variable.

```
data_sans_bur <- data[, -which(names(data) == "code_bur_vote")]

sum(is.na(data_sans_bur)) / (nrow(data_sans_bur) * ncol(data_sans_bur))
```

```
[1] 0.0104504504504505
```

En retirant la colonne des codes de bureaux de votes, nous tombons à 1% de `NA`, ce qui est très satisfaisant.

De plus, comme nous avons les données de la base complète, nous allons pouvoir comparer nos résultats avec celle-ci mais aussi analyser si les changements considérés comme sûrs étaient bons.

```
# Comparaison des circonscriptions

#----- Différence base sûre -----

sum(!data_full$circonscription == data_save_2$circonscription, na.rm = TRUE)
```

```
[1] 0
```

```
#----- Résultat des remplacements -----

diff_cir <- data$circonscription == data_full$circonscription

sum(diff_cir, na.rm = TRUE)/nrow(data) #bons en %
```

```
[1] 0.953153153153153
```

```
sum(!diff_cir, na.rm = TRUE)/nrow(data) #mauvais en %
```

```
[1] 0.00630630630630631
```

```
sum(!diff_cir, na.rm = TRUE) #mauvais en valeur
```

```
[1] 7
```

```
sum(!diff_cir, na.rm = TRUE)/sum(is.na(data_save_1$circonscription)) #base NA
```

```
[1] 0.0316742081447964
```

```r
sum(is.na(diff_cir))/nrow(data) #na en %
```

```
[1] 0.0405405405405405
```

```r
sum(is.na(diff_cir)) #na en valeur
```

```
[1] 45
```

```r
sum(is.na(diff_cir))/sum(is.na(data_save_1$circonscription)) #base NA
```

```
[1] 0.203619909502262
```

La comparaison des bases `data_full` et `data_save_2` montre qu'il n'y a aucune différence entre les colonnes `circonscription` des deux jeux de données, à l'exception des valeurs manquantes (`NA`). En analysant la correspondance entre `data` (données complétées) et `data_full` (données complètes de référence), nous constatons que plus de 95 % des valeurs dans `data` correspondent à celles de `data_full`. Cela montre que l'opération de complétion des données a été très efficace. En effet, seulement 0.63 % des valeurs sont incorrectes, soit un total de 7 erreurs sur 1 100 observations, ce qui reste marginal. En revanche, environ 4 % des valeurs dans `data` sont toujours manquantes (`NA`), soit 45 valeurs, ce qui est relativement faible compte tenu du nombre initial de valeurs manquantes.

Il est également intéressant de noter que parmi les valeurs initialement manquantes dans la variable `circonscription`, près de 97 % des `NA` ont pu être correctement remplacées. En revanche, environ 4 % des `NA` n'ont pas du tout pu être traitées, probablement en raison de l'impossibilité d'identifier de manière fiable la circonscription pour ces cas particuliers.

Bien que nous préférions des remplacements exacts pour garantir une base de données complète, ces résultats sont satisfaisants. En effet, le faible nombre d'erreurs combiné à un pourcentage élevé de bonnes valeurs complétées montre que l'intégrité générale des données est préservée. De plus, les erreurs restantes n'ont probablement pas un impact significatif sur l'analyse globale, car elles concernent une proportion très faible des données. Par ailleurs, le fait que 4 % des valeurs restent manquantes dans `data` peut être considéré comme acceptable, en particulier compte tenu de la difficulté à traiter toutes les valeurs manquantes de manière fiable.

```r
# Comparaison des communes

#----- Différence base sûre -----

sum(!data_full$nom_commune == data_save_2$nom_commune, na.rm = TRUE)
```

```
[1] 0
```

```r
#----- Résultat des remplacements -----

diff_com <- data$nom_commune == data_full$nom_commune

sum(diff_com, na.rm = TRUE)/nrow(data) #bons en %
```

```
[1] 0.978378378378378
```

```
sum(!diff_com, na.rm = TRUE)/nrow(data) #mauvais en %
```

```
[1] 0
```

```
sum(!diff_com, na.rm = TRUE) #mauvais en valeur
```

```
[1] 0
```

```
sum(!diff_com, na.rm = TRUE)/sum(is.na(data_save_1$nom_commune)) #base NA
```

```
[1] 0
```

```
sum(is.na(diff_com))/nrow(data) #na en %
```

```
[1] 0.0216216216216216
```

```
sum(is.na(diff_com)) #na en valeur
```

```
[1] 24
```

```
sum(is.na(diff_com))/sum(is.na(data_save_1$code_bur_vote)) #base NA
```

```
[1] 0.106194690265487
```

La comparaison des bases `data_full` et `data_save_2` montre qu'il n'y a aucune différence entre les colonnes `nom_commune` des deux jeux de données, à l'exception des valeurs manquantes (`NA`). En analysant la correspondance entre `data` (données complétées) et `data_full` (données complètes de référence), nous constatons que 97,8 % des valeurs dans `data` correspondent à celles de `data_full`, ce qui est un excellent résultat. Le taux d'erreurs est nul (0 %), ce qui veut dire qu'aucune valeur incorrecte n'a été introduite. Par ailleurs, 2.16 % des valeurs dans `data` restent manquantes (`NA`), soit un total de 24 valeurs, ce qui représente 10 % des valeurs manquantes de départ.

Il est important de comprendre pourquoi le nombre de valeurs manquantes (`NA`) incorrectement complétées est nul. Cela s'explique par le fait que nous avons délibérément choisi de ne pas tenter de compléter un nom de commune lorsque nous n'étions pas certains de celui-ci. Cette prudence s'explique par deux raisons principales :

1. Contrairement à une circonscription, où une valeur incorrecte a un impact relativement limité, une erreur sur le nom d'une commune peut avoir des conséquences importantes, en particulier pour les petites communes. Par exemple, si une commune ne compte que 2 bureaux de vote et qu'un 3ème bureau, incorrectement assigné, lui est attribué, cela pourrait fausser les analyses.

2. Nous étions également limités par la structure de nos données. En Loire-Atlantique, il existe 207 communes, mais dans notre base `data`, nous n'en avions que 192. Cette différence s'explique par le fait que certaines communes très petites, n'ayant qu'un seul bureau de vote, avaient vu leur nom totalement masqué par des valeurs `NA`, rendant leur identification impossible.

En conclusion, ces résultats témoignent d'une excellente qualité dans la complétion des données pour la variable `nom_commune`. Le choix de ne pas forcer la complétion en cas d'incertitude a permis de préserver l'intégrité des données et d'éviter des erreurs qui auraient pu avoir un impact significatif sur les analyses futures.

```
# Comparaison des codes de bureaux de vote

#----- Différence base sûre -----

sum(!data_full$code_bur_vote == data_save_2$code_bur_vote, na.rm = TRUE)
```

```
[1] 0
```

```
#----- Résultat des remplacements -----

diff_bur <- data$code_bur_vote == data_full$code_bur_vote

sum(diff_bur, na.rm = TRUE)/nrow(data) #bons en %
```

```
[1] 0.80990990990991
```

```
sum(!diff_bur, na.rm = TRUE)/nrow(data) #mauvais en %
```

```
[1] 0
```

```
sum(!diff_bur, na.rm = TRUE) #mauvais en valeur
```

```
[1] 0
```

```
sum(!diff_bur, na.rm = TRUE)/sum(is.na(data_save_1$code_bur_vote)) #base NA
```

```
[1] 0
```

```
sum(is.na(diff_bur))/nrow(data) #na en %
```

```
[1] 0.19009009009009
```

```r
sum(is.na(diff_bur)) #na en valeur
```

```
[1] 211
```

```r
sum(is.na(diff_bur))/sum(is.na(data_save_1$code_bur_vote)) #base NA
```

```
[1] 0.933628318584071
```

En analysant la correspondance entre la variable `code_bur_vote` dans `data` (données complétées) et `data_full` (données complètes de référence), nous constatons que 81 % des valeurs de `data` correspondent parfaitement à celles de `data_full`. Ce résultat reflète une part importante de valeurs correctement complétées, bien qu'inférieure à celle obtenue pour d'autres variables. À noter que le taux d'erreurs est nul (0 %), ce qui montre qu'aucune valeur incorrecte n'a été introduite.

Cependant, 19 % des valeurs, soit un total de 211 valeurs, demeurent manquantes (`NA`), ce qui représente plus de 90 % des valeurs que nous avions à traiter pour cette variable. Ce taux relativement élevé s'explique par la nature de la variable `code_bur_vote`. En effet, comme mentionné précédemment, cette variable est principalement de nature administrative et n'a pas de réel intérêt pour faire ressortir des tendances ou analyses statistiques significatives. Nous l'avons utilisée uniquement pour compléter d'autres colonnes lorsque cela était possible et fiable.

De plus, tenter de compléter cette variable de manière exhaustive n'aurait pas été pertinent. D'une part, cela n'aurait pas apporté d'amélioration significative à nos analyses globales. D'autre part, cela aurait pu introduire des erreurs importantes, comme dans le cas de `nom_commune`, où un bureau de vote pourrait être incorrectement associé à une commune qui n'est pas la bonne. Ce type d'erreur aurait pu fausser les résultats, notamment pour les petites communes, où chaque bureau de vote a un poids important.

En conclusion, bien que le taux de complétion soit moins élevé pour cette variable, cela ne constitue pas un problème majeur. L'effort de complétion s'est concentré sur des variables plus significatives, tandis que pour `code_bur_vote`, nous avons privilégié la prudence afin de préserver l'intégrité des données.

```r
# Comparaison des inscrits

#----- Différence base sûre -----

sum(!data_full$inscrits == data_save_2$inscrits, na.rm = TRUE)
```

```
[1] 0
```

```r
#----- Résultat des remplacements -----

diff_insc <- data$inscrits == data_full$inscrits

sum(diff_insc, na.rm = TRUE)/nrow(data) #bons en %
```

```
[1] 0.782882882882883
```

```
sum(!diff_insc, na.rm = TRUE)/nrow(data) #mauvais en %
```

```
[1] 0.196396396396396
```

```
sum(!diff_insc, na.rm = TRUE) #mauvais en valeur
```

```
[1] 218
```

```
sum(!diff_insc, na.rm = TRUE)/sum(is.na(data_save_1$inscrits)) #base NA
```

```
[1] 0.904564315352697
```

```
sum(is.na(diff_insc))/nrow(data) #na en %
```

```
[1] 0.0207207207207207
```

```
sum(is.na(diff_insc)) #na en valeur
```

```
[1] 23
```

```
sum(is.na(diff_insc))/sum(is.na(data_save_1$inscrits)) #base NA
```

```
[1] 0.0954356846473029
```

La comparaison entre `data` (données complétées) et `data_full` (données complètes de référence) révèle qu'il n'y a aucune différence pour la variable `inscrits` dans la base `data_full` par rapport à `data_save_2`. Contrairement aux variables que nous venons de voir, cela s'explique par le fait que nous n'avons pu faire aucun changement de façon sûre et non qu'ils sont tous bon. En effet, pour la variable `inscrits` rien ne nous permettait de trouver la valeur exacte.

Cependant, les résultats des remplacements dans `data` montrent un tableau plus complexe. En effet, 78 % des valeurs de `data` correspondent aux valeurs de `data_full`, ce qui signifie que la majorité des valeurs sont bonnes. Toutefois, 20 % des valeurs, soit un total de 218 valeurs, sont incorrectes, tandis que 2 % (23 valeurs) restent manquantes (`NA`).

Cette situation est quelque peu inquiétante, car sur les 241 valeurs manquantes que comptait la base initiale, aucune n'a pu être bien complétée (218 mauvaises + 23 `NA` = 241 valeurs). Néanmoins, il est important de remettre ces résultats dans leur contexte. La variable `inscrits` représente un nombre d'inscrits, et il est souvent difficile d'obtenir cette information de manière exacte à l'unité près, surtout lorsqu'il s'agit de données estimées.

Dans ce contexte, il convient de considérer que des erreurs mineures peuvent être tolérées. Par exemple, si une valeur approximée se situe dans une plage de ±5 % autour de la valeur réelle, elle peut être considérée comme valide. Cela signifie que, même si la valeur exacte n'a pas pu être remplie, une approximation raisonnable peut être acceptable tant que l'écart reste dans cette plage. Nous allons donc vérifier cela.

```
comp_insc <- abs(data$inscrits - data_full$inscrits) <= 0.05*data_full$inscrits

sum(comp_insc, na.rm = TRUE) / nrow(data) #bons en %
```

```
[1] 0.855855855855856
```

```
sum(!comp_insc, na.rm = TRUE) / nrow(data) #mauvais en %
```

```
[1] 0.123423423423423
```

```
sum(!comp_insc, na.rm = TRUE) #mauvais en valeur
```

```
[1] 137
```

```
sum(!comp_insc, na.rm = TRUE)/sum(is.na(data_save_1$inscrits)) #base NA
```

```
[1] 0.568464730290456
```

```
sum(is.na(comp_insc)) / nrow(data) #na en %
```

```
[1] 0.0207207207207207
```

```
sum(is.na(comp_insc)) #na en valeur
```

```
[1] 23
```

```
sum(is.na(comp_insc))/sum(is.na(data_save_1$inscrits)) #base NA
```

```
[1] 0.0954356846473029
```

Lorsqu'on évalue ces résultats à l'aide d'un seuil d'acceptation d'erreur de 5 %, nous obtenons que 85 % des valeurs sont considérées comme correctes, avec un écart de moins de 5 % par rapport à la valeur réelle. Le taux d'erreur reste à 12 % (137 mauvaises valeurs) avec un taux de 2 % de valeurs manquantes. Toutefois, en utilisant un seuil plus large de 10 %, nous obtenons que 90 %

des valeurs sont considérées comme correctes, avec seulement 8 % des valeurs incorrectes et 2 % de valeurs manquantes.

Les résultats obtenus pour la variable inscrits montrent que, du fait de l'impossibilité de compléter cette variable avec une grande certitude (aucun remplacement sûr), les remplacements effectués n'ont qu'un faible taux de valeurs acceptables En effet, à un seuil d'erreur de 5 %, moins de 50 % des valeurs estimées sont correctes, et à 10 %, environ 35 % des valeurs restent erronées. Ces résultats indiquent que la méthode de complétion, qui repose sur des estimations basées sur les autres bureaux de vote de la même commune, présente des limites notables. Cela s'explique en grande partie par la variabilité entre les bureaux de vote d'une même commune, qui peut être particulièrement marquée dans les petites communes disposant de peu de bureaux de vote. L'argument selon lequel la population de chaque commune serait uniformément répartie entre ses bureaux de vote ne tient pas suffisamment dans ce contexte pour permettre une estimation fiable du nombre d'inscrits.

Les valeurs suivantes, étant de nature similaire (quantitatives continues), feront directement l'objet d'une analyse des différences à des seuils d'erreur de 5 % et 10 %.

```
# Comparaison des taux d'absence

comp_abs <- abs(data$tx_absents - data_full$tx_absents) <=
  0.05*data_full$tx_absents

sum(comp_abs, na.rm = TRUE) / nrow(data) #bons en %
```

```
[1] 0.964864864864865
```

```
sum(!comp_abs, na.rm = TRUE) / nrow(data) #mauvais en %
```

```
[1] 0.0351351351351351
```

```
sum(!comp_abs, na.rm = TRUE) #mauvais en valeur
```

```
[1] 39
```

```
sum(!comp_abs, na.rm = TRUE)/sum(is.na(data_save_1$tx_absents)) #base NA
```

```
[1] 0.168103448275862
```

```
sum(is.na(comp_abs)) / nrow(data) #na en %
```

```
[1] 0
```

```r
sum(is.na(comp_abs)) #na en valeur
```

```
[1] 0
```

```r
sum(is.na(comp_abs))/sum(is.na(data_save_1$tx_absents)) #base NA
```

```
[1] 0
```

Les résultats des remplacements pour la variable `tx_absents` montrent une très bonne précision dans la complétion des données. À un seuil d'erreur de 5 %, 97 % des valeurs dans `data` correspondent aux valeurs de `data_full`, ce qui indique que la majorité des données ont été correctement remplacées. De plus, seulement 3 % des valeurs sont incorrectes, soit un total de 33 valeurs. À un seuil d'erreur de 10 %, la proportion des valeurs correctes augmente légèrement à 98 %, avec seulement 2 % des valeurs incorrectes (22 valeurs).

Il est également à noter qu'aucune valeur manquante (`NA`) n'a subsisté après le remplacement des données. Cela montre que l'opération de complétion a été menée avec succès, avec un taux de remplacement de 100 %.

En conclusion, les résultats sont très satisfaisants pour la variable `tx_absents`, avec un faible taux d'erreur et une absence totale de valeurs manquantes. Cela suggère que l'approche de complétion des données a été efficace pour cette variable, et les erreurs restantes ne devraient pas avoir d'impact significatif sur l'analyse globale.

```r
# Comparaison des taux de votants

comp_pres <- abs(data$tx_votants - data_full$tx_votants) <=
  0.05*data_full$tx_votants

sum(comp_pres, na.rm = TRUE) / nrow(data) #bons en %
```

```
[1] 0.978378378378378
```

```r
sum(!comp_pres, na.rm = TRUE) / nrow(data) #mauvais en %
```

```
[1] 0.0216216216216216
```

```r
sum(!comp_pres, na.rm = TRUE) #mauvais en valeur
```

```
[1] 24
```

```r
sum(!comp_pres, na.rm = TRUE)/sum(is.na(data_save_1$tx_votants)) #base NA
```

```
[1] 0.118226600985222
```

```r
sum(is.na(comp_pres)) / nrow(data) #na en %
```

```
[1] 0
```

```r
sum(is.na(comp_pres)) #na en valeur
```

```
[1] 0
```

```r
sum(is.na(comp_pres))/sum(is.na(data_save_1$tx_votants)) #base NA
```

```
[1] 0
```

Les résultats des remplacements pour la variable `tx_votants` montrent également une très bonne précision dans la complétion des données, similaire à celle obtenue pour `tx_absents`. À un seuil d'erreur de 5 %, 99 % des valeurs dans `data` correspondent aux valeurs de `data_full`, ce qui indique que la majorité des données ont été correctement remplacées. Seulement 1 % des valeurs, soit un total de 13 valeurs, sont incorrectes. À un seuil d'erreur de 10 %, la proportion des valeurs correctes reste très élevée, avec 99 % des valeurs correspondantes et seulement 6 valeurs incorrectes.

Il est aussi important de noter que, à un seuil d'erreur de 5 %, le pourcentage d'erreur relatif aux remplacements est de 6 % par rapport au nombre initial de valeurs manquantes (`NA`), et qu'il baisse à 2 % à un seuil d'erreur de 10 %. Cela reflète une très faible erreur dans les remplacements effectués. En d'autres termes, seulement 6 % des remplacements à 5 % de tolérance et 2 % à 10 % sont incorrects par rapport aux valeurs manquantes initiales.

Tout comme pour `tx_absents`, il est crucial de souligner qu'aucune valeur manquante (`NA`) n'a subsisté après le remplacement des données, ce qui montre que l'opération de complétion a été menée à bien, avec un taux de remplacement de 100 %.

En conclusion, les résultats pour `tx_votants` sont très similaires à ceux de `tx_absents`, avec un faible taux d'erreur et une absence totale de valeurs manquantes. Étant donné que la valeur de `tx_votants` est généralement plus grande que celle de `tx_absents`, cela permet une plus grande tolérance aux erreurs relatives de 5 % et 10 %, ce qui explique probablement l'écart de quelques pourcentages. Les résultats sont donc très satisfaisants dans ce contexte.

```r
# Comparaison des taux de votes blancs

comp_bla <- abs(data$tx_blancs - data_full$tx_blancs) <=
  0.05*data_full$tx_blancs

sum(comp_bla, na.rm = TRUE) / nrow(data) #bons en %
```

```
[1] 0.937837837837838
```

```r
sum(!comp_bla, na.rm = TRUE) / nrow(data) #mauvais en %
```

```
[1] 0.0621621621621622
```

```r
sum(!comp_bla, na.rm = TRUE) #mauvais en valeur
```

```
[1] 69
```

```r
sum(!comp_bla, na.rm = TRUE)/sum(is.na(data_save_1$tx_blancs)) #base NA
```

```
[1] 0.341584158415842
```

```r
sum(is.na(comp_bla)) / nrow(data) #na en %
```

```
[1] 0
```

```r
sum(is.na(comp_bla)) #na en valeur
```

```
[1] 0
```

```r
sum(is.na(comp_bla))/sum(is.na(data_save_1$tx_blancs)) #base NA
```

```
[1] 0
```

Les résultats pour le taux de votes blancs (`tx_blancs`) montrent une précision légèrement inférieure à celle obtenue pour les autres variables, bien que toujours acceptable. À un seuil d'erreur de 10 %, 95 % des valeurs dans `data` correspondent aux valeurs de `data_full`, tandis que 5 % des valeurs (soit 60 valeurs) sont incorrectes. À un seuil d'erreur de 5 %, 94 % des valeurs sont correctes, mais 6 % des valeurs sont incorrectes (soit 70 valeurs).

Il est également important de noter que, comme pour les autres variables, aucune valeur manquante (`NA`) n'a subsisté après le remplacement des données, ce qui montre que l'opération de complétion a été menée à bien, avec un taux de remplacement de 100 %.

Lorsque nous considèrons l'erreur relative par rapport au nombre initial de valeurs manquantes (`NA`), nous observons que, à un seuil de 5 %, environ 35 % des remplacements ont été incorrects. Cependant, ce taux reste relativement modéré comparé aux résultats obtenus pour d'autres variables.

En conclusion, bien que les résultats pour `tx_blancs` soient légèrement moins bons que pour les autres variables, ils restent acceptables, avec une absence totale de valeurs manquantes. L'erreur relative à 5 % est un peu plus importante, mais cela peut être dû à la variabilité des taux de votes blancs entre les bureaux de vote, notamment dans les communes où le taux de participation est plus faible. Ces résultats suggèrent que, bien que les remplacements aient permis de traiter efficacement les données manquantes, la méthode reste moins fiable dans certains cas en raison de la variabilité des taux de blancs.

```r
# Comparaison des taux de votes nuls

comp_nul <- abs(data$tx_nuls - data_full$tx_nuls) <=
  0.05*data_full$tx_nuls

sum(comp_nul, na.rm = TRUE) / nrow(data) #bons en %
```

```
[1] 0.926126126126126
```

```r
sum(!comp_nul, na.rm = TRUE) / nrow(data) #mauvais en %
```

```
[1] 0.0738738738738739
```

```r
sum(!comp_nul, na.rm = TRUE) #mauvais en valeur
```

```
[1] 82
```

```r
sum(!comp_nul, na.rm = TRUE)/sum(is.na(data_save_1$tx_nuls)) #base NA
```

```
[1] 0.376146788990826
```

```r
sum(is.na(comp_nul)) / nrow(data) #na en %
```

```
[1] 0
```

```r
sum(is.na(comp_nul)) #na en valeur
```

```
[1] 0
```

```r
sum(is.na(comp_nul))/sum(is.na(data_save_1$tx_nuls)) #base NA
```

```
[1] 0
```

Les résultats pour le taux de votes nuls (`tx_nuls`) montrent que, tout comme pour les autres variables, le taux de remplacement des valeurs manquantes est très élevé. À un seuil d'erreur de 10 %, 93 % des valeurs dans `data` correspondent aux valeurs de `data_full`, tandis que 7 % des valeurs (soit 75 valeurs) sont incorrectes. À un seuil d'erreur de 5 %, 92 % des valeurs sont correctes, mais 8 % des valeurs sont incorrectes (soit 80 valeurs).

Il est à noter qu'aucune valeur manquante (`NA`) n'a subsisté après le remplacement des données, ce qui signifie que l'opération de complétion a été effectuée avec succès, atteignant un taux de remplacement de 100 %.

Concernant l'erreur relative par rapport au nombre initial de valeurs manquantes (`NA`), environ 34 % des remplacements ont été incorrects à un seuil de 10 %, ce qui est relativement élevé comparé aux autres variables. Cela indique que, même si la méthode de remplacement a permis de traiter efficacement les données manquantes, elle reste moins fiable pour cette variable, probablement en raison de la variabilité plus marquée des taux de votes nuls entre les bureaux de vote.

En conclusion, bien que les résultats pour `tx_nuls` soient encore acceptables, ils montrent un taux d'erreur légèrement plus élevé que pour d'autres variables. La méthode de complétion des données a permis de traiter les valeurs manquantes, mais le taux d'erreur reste assez significatif à 5 % et 10 %. Cela suggère que la variabilité des taux de votes nuls entre les bureaux de vote pourrait affecter la précision des estimations dans certaines communes.

```
# Comparaison des taux de votes exprimés

comp_exp <- abs(data$tx_exprimes - data_full$tx_exprimes) <=
  0.05*data_full$tx_exprimes

sum(comp_exp, na.rm = TRUE) / nrow(data) #bons en %
```

```
[1] 1
```

```
sum(!comp_exp, na.rm = TRUE) / nrow(data) #mauvais en %
```

```
[1] 0
```

```
sum(!comp_exp, na.rm = TRUE) #mauvais en valeur
```

```
[1] 0
```

```
sum(!comp_exp, na.rm = TRUE)/sum(is.na(data_save_1$tx_exprimes)) #base NA
```

```
[1] 0
```

```
sum(is.na(comp_exp)) / nrow(data) #na en %
```

```
[1] 0
```

```
sum(is.na(comp_exp)) #na en valeur
```

```
[1] 0
```

```r
sum(is.na(comp_exp))/sum(is.na(data_save_1$tx_exprimes)) #base NA
```

```
[1] 0
```

Les résultats pour le taux de votes exprimés (`tx_exprimes`) montrent une situation surprenante mais très satisfaisante. En effet, que ce soit avec un seuil de 5 % ou de 10 %, toutes les valeurs dans `data` correspondent aux valeurs de `data_full`, avec un taux de concordance de 100 % et aucun remplacement incorrect. Contrairement aux autres variables comme `tx_blancs` et `tx_nuls`, où les valeurs proches de zéro rendent plus difficile la tolérance aux erreurs relatives, `tx_exprimes` présente des valeurs généralement proches de 100 %. Cela signifie qu'un seuil de 5 % ou 10 % couvre une plage suffisamment large pour inclure toutes les valeurs possibles, rendant ainsi les remplacements parfaits.

Cela peut expliquer pourquoi, bien que `tx_blancs` et `tx_nuls` aient montré un taux d'erreur relativement plus élevé à ces seuils, `tx_exprimes` a parfaitement réussi à se conformer aux attentes. Étant donné que ces dernières ont des valeurs beaucoup plus grandes, les écarts relatifs sont moins sensibles et permettent une plus grande marge de manœuvre dans les remplacements.

En conclusion, bien que les autres variables aient montré des erreurs à certains seuils, les résultats pour `tx_exprimes` sont impeccables, probablement en raison de la taille plus importante de ces valeurs et de la large plage couverte par les marges d'erreur de 5 % et 10 %.

```r
# Comparaison des latitudes

comp_lat <- abs(data$latitude - data_full$latitude) <=
  0.005*data_full$latitude

sum(comp_lat, na.rm = TRUE) / nrow(data) #bons en %
```

```
[1] 0.989189189189189
```

```r
sum(!comp_lat, na.rm = TRUE) / nrow(data) #mauvais en %
```

```
[1] 0
```

```r
sum(!comp_lat, na.rm = TRUE) #mauvais en valeur
```

```
[1] 0
```

```r
sum(!comp_lat, na.rm = TRUE)/sum(is.na(data_save_1$latitude)) #base NA
```

```
[1] 0
```

```r
sum(is.na(comp_lat)) / nrow(data) #na en %
```

```
[1] 0.0108108108108108
```

```r
sum(is.na(comp_lat)) #na en valeur
```

```
[1] 12
```

```r
sum(is.na(comp_lat))/sum(is.na(data_save_1$latitude)) #base NA
```

```
[1] 0.0552995391705069
```

Les résultats pour la variable `latitude` montrent une très bonne précision dans les remplacements. En utilisant un seuil d'erreur de 0.5 % (soit 10 fois plus précis que pour les autres variables), 99 % des valeurs sont correctement remplacées, avec seulement 1 valeur incorrecte. Cela témoigne de la grande précision des données de latitude.

Étant donné que les coordonnées géographiques (latitude) sont très précises, il est logique de prendre un seuil aussi strict, à 0.5 %, afin de minimiser toute erreur. Cette approche est parfaitement adaptée pour les données géographiques où des écarts de quelques milliers de mètres peuvent avoir un impact significatif.

Cependant, il reste encore 12 valeurs manquantes dans la base, soit 1 % des données, ce qui correspond à environ 5 % des valeurs initialement manquantes. Néanmoins, cela ne remet pas en cause la qualité générale de la complétion des données de latitude.

```r
# Comparaison des longitudes

comp_long <- abs(data$longitude - data_full$longitude) <=
  0.005*abs(data_full$longitude) #les coordonnées de longitude sont négatives

sum(comp_long, na.rm = TRUE) / nrow(data) #bons en %
```

```
[1] 0.972972972972973
```

```r
sum(!comp_long, na.rm = TRUE) / nrow(data) #mauvais en %
```

```
[1] 0.0162162162162162
```

```r
sum(!comp_long, na.rm = TRUE) #mauvais en valeur
```

```
[1] 18
```

```r
sum(!comp_long, na.rm = TRUE)/sum(is.na(data_save_1$longitude)) #base NA
```

```
[1] 0.0782608695652174
```

```r
sum(is.na(comp_long)) / nrow(data) #na en %
```

```
[1] 0.0108108108108108
```

```r
sum(is.na(comp_long)) #na en valeur
```

```
[1] 12
```

```r
sum(is.na(comp_long))/sum(is.na(data_save_1$longitude)) #base NA
```

```
[1] 0.0521739130434783
```

Les résultats pour la variable `longitude` montrent une bonne précision dans les remplacements, bien que légèrement inférieure à celle observée pour `latitude`. À un seuil d'erreur de 0.5 %, 97 % des valeurs sont correctement remplacées, avec 19 valeurs incorrectes. Ce taux de correction est encore très élevé, mais il est intéressant de noter qu'il reste 12 valeurs manquantes, soit 1. % des données, ce qui représente environ 5.2 % des valeurs initialement manquantes.

La différence de résultats entre `latitude` et `longitude` s'explique en grande partie par la géographie de la région étudiée. En effet, la Loire-Atlantique s'étend sur une plus grande distance horizontale (est-ouest) que verticale (nord-sud), ce qui signifie que les variations en longitude peuvent être plus significatives que celles en latitude. Les petites erreurs en longitude peuvent avoir un impact plus grand sur les calculs, car une différence d'un degré de longitude couvre une distance plus importante à cette latitude qu'en latitude.

Cela dit, la complétion des données de longitude reste satisfaisante, avec une marge d'erreur acceptable pour la plupart des applications.

5.2 Limites

Il est important de noter que cette analyse, bien qu'efficace, présente certaines limites. Tout d'abord, malgré l'efficacité de notre méthode de remplacement, nous n'avons pas pu compléter toutes les données de manière absolue. Certaines variables, telles que celles ayant des caractéristiques particulièrement spécifiques ou des valeurs proches de 0 (par exemple, les taux de votes nuls ou blancs), se sont révélées plus complexes à estimer avec une précision élevée.

De plus, pour cette étude, nous avons volontairement décidé de ne pas utiliser Internet ou d'autres sources externes pour compléter nos données manquantes. Nous avons décidé cela afin de pouvoir introduire des méthodes d'approximations, car, sinon, nous aurions trouvé toutes les données en ligne, notamment dans la base de données complète. Bien que cela aurait permis de compléter les

données de manière plus précise, l'objectif ici était de se concentrer sur les techniques de gestion des données manquantes appliquées à une base incomplète et de simuler un contexte plus réaliste.

Enfin, dans notre analyse "logique" des données, nous avons pris soin de ne pas tirer de conclusions hâtives alors même que les conclusions étaient généralement facilement visible puisqu'après l'incorpotation des `NA`, les lignes n'ont pas été mélanges et, de ce fait, même si pour une ligne il manquait le nom de la commune, celle-ci était au milieu des autres, il aurait donc été trop facile de conclure grâce à cela et ce n'est pas le but de ce dossier.

5.3 Conclusion du dossier

De manière générale, la complétion de la base de données a été réalisée de manière efficace, avec des résultats largement satisfaisants. En effet, l'erreur lors des remplacements des valeurs au seuil de 5 % est inférieure à 3 % dans la majorité des cas, ce qui témoigne de la qualité des estimations. En ce qui concerne les valeurs manquantes (`NA`), elles ont été complétées à hauteur de 75 % dans la base initiale. Cependant, en excluant la variable relative aux taux de votants (pour les raisons déjà expliquées précédemment), nous parvenons à un taux de complétion plus élevé, atteignant 85 %.

Si nous regardons l'évolution de la base par rapport à son état initial, il est intéressant de noter qu'au départ, la base n'était complétée qu'à 80 %, en raison du retrait de 20 % des données, soit celles marquées comme `NA`. Après la complétion de ces valeurs, nous obtenons une base dont le taux de complétion atteint 97 % en incluant la variable `code_bur_vote`. Si cette variable est exclue de l'analyse, le taux de complétion monte même à 99 %.

Enfin, en prenant en compte une erreur maximale de 3 % pour les remplacements, nous pouvons conclure que la base de données est désormais complète et fiable à 96 %. Cette complétion des données permet d'envisager des analyses plus approfondies, en s'appuyant sur un ensemble de données de qualité, avec un faible risque d'erreur.

En résumé, la complétion des données a été réalisée de manière robuste, avec un taux d'erreur faible et un taux de complétion élevé, assurant ainsi la fiabilité des analyses futures.

Nous tenons tout de même à ajouter qu'au cours de ce dossier, nous avons pris le temps de détailler chaque opération, allant même jusqu'à effectuer des processus longs pour ne compléter qu'une ou deux variables. Toutefois, dans le cadre d'un projet réel en entreprise, nous n'aurions probablement pas adopté une telle approche minutieuse, ou du moins pas au point d'examiner chaque cas individuellement. Malgré tout, cette démarche a été très enrichissante, car elle nous a permis d'explorer diverses méthodes et de mieux comprendre les subtilités de la gestion des données manquantes.

6 Sources

Base officielle des codes postaux. (2024, novembre 13). *La Poste*. https://www.data.gouv.fr/fr/datasets/base-officielle-des-codes-postaux/#/information

Ministère de l'intérieur. (2022, avril 14). Election présidentielle des 10 et 24 avril 2022 - Résultats définitifs du 1er tour. Data.gouv.fr. https://www.data.gouv.fr/fr/datasets/election-presidentielle-des-10-et-24-avril-2022-resultats-definitifs-du-1er-tour/