Projet Traitement de données

Masse Ba - Pape T. Gueye - Mamadou Mbodj

29/03/2021

0.Introduction

Le but de ce travail est d'effectuer le traitement des données issues de l'enquête sur les effets de la covid19 et les stratégies de résilience du système académique à Dakar. Ces données sont contenues dans deux bases : **base établissement** et **base élève**. Ces deux bases seront, dans la suite du document, nommées respectivement : **dataAdmin** et **df_eleve**. Le traitement de ces données consistera à 1) détecter et corriger les valeurs aberrantes et/ou atypiques, 2) détecter les valeurs manquantes (non réponses totales/partielles, hors champs et missing values) et enfin, 3) Imputer ces valeurs manquantes.

Méthodologie : D'abord, concernant les valeurs aberrantes, trois méthodes de détection ont été utilisée à savoir Grubbs, boxplot et Hampel. L'apurement consiste donc, pour chacune de ces approches, à : 1- détecter les valeurs aberrantes ; 2- corriger les valeurs aberrantes (imputation par la moustache gauche ou moustache droite du Boxplot, Winsorization, Réduction des poids). Ensuite, pour le traitement des données manquantes, nous avons, premièrement, procéder à la distinction entre les non réponses totales et partielles afin de rectifier par une repondération. Deuxèmement, nous avons distinguer les hors champs en se basant sur le questionnaire et la logique des données et nous leur avons attribué le **code hors champs 90909**. Enfin, nous avons terminé avec l'imputation des valeurs manquantes dues aux erreurs de l'enquête en utilisant différentes méthodes jugées pertinentes selon les variables. Principalement, les méthode d'imputation présentes dans notre travail sont : la méthode d'imputation par la moyenne et la méthode Knn. Cette dernière méthode est appliquée aux données quantitatives (imputation par la moyenne des K plus proche voisin) et aux données catégorielles (imputation par le mode des k plus proche voisin).

Première partie : Détection et traitement des valeurs abérrantes

Dans cette première partie, il s'agit de déterminer les valeurs atypiques et abérrantes des variables quantitatives contenues dans les deux bases. Après avoir chargé et visualisé les deux bases, nous allons commencer par la base établissement, ensute la base élève, en suivant l'ordre des variables quantitatives dans les sections.

1. Traitement de la base Etablissement : section 1 et 2

Nous allons détecter les valeurs abérrantes contenues dans la base des données relatives aux établissements qui ont été enquêtés.

```
library(haven)
dataAdmin <- read_sav("../raw/data_admin1.sav")
#View(dataAdmin)
```

Quelques manipulations prélminaires (le résultat est masqué car étant trop long).

```
summary(dataAdmin)
```

```
names(dataAdmin)
```

```
library(questionr)
lookfor(dataAdmin)
```

1.1. Année de création des écoles

Sur l'année de création des établissement, on note certaines écoles qui ont des dates de création très anciennes : des dates où à ce moment il n'y avait pas d'école au Sénégal (1848, 1928, etc.).

```
summary(dataAdmin$I_1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1848    1997    2006    2081    2012    9999       6
```

```
dataAdmin$I_1[dataAdmin$I_1<1950 & !is.na(dataAdmin$I_1)]
```

```
## [1] 1928 1938 1848 1937
```

Les modalités 9999 ("inconnue") et les valeurs manquantes sont les étalissements indexés ci-dessous.

```
which(dataAdmin$I_1>2020 & !is.na(dataAdmin$I_1))
```

```
## [1] 132 167
```

```
which(is.na(dataAdmin$I_1))
```

```
## [1] 149 151 158 166 203 206
```

Pour rectifier les dates abérrantes à gauche et les valeurs manquantes, on les remplace par 9999.

```
dataAdmin$I_1[which(dataAdmin$I_1<1950 & is.na(dataAdmin$I_1))]<- 9999
```

1.2. Recodage des "ne sait pas"

La modalité "ne sait pas" exste pour certaines variables et doit être renseignée par 9999. Cependant, certains enquêteurs ont mis 999, d'autres 99999, d'autres même 999999. Nous allons donc procéder à l'uniformisation de cette modalité en mettant 9999 partout.

```
#numéros de colonnes des variables concernées
x=c(34,35,38,39,42,43,51,52,61,62,63,64,65,66,67,68,69,70,71,72,74,83,92,121)
for (j in x) {
  dataAdmin[which(dataAdmin[,j]==999 |dataAdmin[,j]==99999
                                     |dataAdmin[,j]==999999
                                     |dataAdmin[,j]==9999999), j] <- 9999
}
```

1.3. Traitement des autres variables quantitatives

Nous passons à la détection des outliers des variables quantiitatives (effectifs dans les établissements, nombre de classe, montants des frais et dépenses liés à la gestion de l'établissement, etc.). Pour cela, nous effectuerons, dans les lignes suivantes, des tests de Grubbs de détection des outliers sur l'ensemble de ces différentes variables.

Dans un premier temps, nous définirons une fonction qui effectue des tests itératifs de Grubs sur une variable qu'elle prend en argument jusqu'au test qui donne une p-value supérieur ou égale à 0,05 %. Elle est nommée **detect_outliers()**.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(outliers)
library(pander)
```

```
detect_outliers=function(var1){
  test<-var1[!is.na(var1)] %>% grubbs.test(type = 10)
  vect=var1[!is.na(var1)]
  while (test$p.value<0.05) {
    vect<-vect[which(vect!= max(vect))]
    test<-vect[] %>% grubbs.test(type = 10)
  }
  Ha=test$alternative
  p_val=test$p.value
  high_val=max(vect)
  result_test=data.frame("Hypothese alternative"=Ha, "p-value"=p_val)
  print(summary(var1))
  pandoc.table(result_test)
  return(high_val)
  print("============================")
}
```

Ensuite nous mettons dans un objet dénommé **index_test** les numéros de colonne de toutes les variables sur lesquelles on veut effectuer un test de détection de valeurs abérrantes. Le résultat test qui donne la p-value supérieure à 0,05 % s'affiche ainsi que l'hypothèse alternative qui est formulée comme suit : "La valeur X est un outlier". On conclut donc qu'à partir de X, les valeurs supérieures (ou inférieures selon que le test est à gauche ou à droite) sont des outliers.

```
index_test=c(34,35,38,39,42,43,51,52,61,62,63,64,65,66,67,68,69,70,71,72,74,83,92,121)
for (j in index_test) {
  detect_outliers(dataAdmin[,j])
}
```

1.4. Rectification par winsorisation

Pour réctifier les valeurs aberrantes précédemment détectées sur les variables quantitatives, nous allons procéder par winsorisation en remplaçant les valeurs aberrantes par une valeur non-atypique. Le choix de cette valeur non-atypique est basé sur les tests de Grubbs que nous venons d'effectuer. Elle correspondra à la plus grande (resp. plus petite) valeur non atypique décelée par le test, c-à-d, la valeur à partir de laquelle la p value du test est supérieur au seuil 0.05. Ci-dessous, la récupération des valeurs (bornes) que l'on doit allouer aux valeurs aberrantes.

```
wins=seq(1,length(index_test),1)
for (j in wins) {
  wins[j]=detect_outliers(dataAdmin[,index_test[j]])
}
```

```
##  I_5_bis_S_sec
##  Min.   :   0.0
##  1st Qu.:  15.0
##  Median :  36.5
```

```
##  Mean   : 109.0
##  3rd Qu.: 108.2
##  Max.   :1293.0
##  NA's   :106
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------ ---------
##  highest value 274 is an   0.1426
##          outlier
## ----------------------------------
##
##  I_5_bis_L_sec
##  Min.   :   0.0
##  1st Qu.:  26.5
##  Median :  54.0
##  Mean   : 127.0
##  3rd Qu.: 144.5
##  Max.   :1144.0
##  NA's   :91
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------ ---------
##  highest value 400 is an   0.06961
##          outlier
## ----------------------------------
##
##  I_5_bis_S_pre
##  Min.   :  0.00
##  1st Qu.: 12.75
##  Median : 29.00
##  Mean   : 77.87
##  3rd Qu.: 86.00
##  Max.   :962.00
##  NA's   :106
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------ ---------
##  highest value 253 is an   0.1034
##          outlier
## ----------------------------------
##
##  I_5_bis_L_pre
##  Min.   :   0
##  1st Qu.:  27
##  Median :  50
##  Mean   : 112
##  3rd Qu.: 120
##  Max.   :1022
##  NA's   :91
##
## ----------------------------------
```

```
##   Hypothese.alternative    p.value
## ------------------------ ---------
##  highest value 275 is an   0.05157
##          outlier
## ----------------------------------
##
##   I_5_bis_S_tl
##  Min.   :  0.00
##  1st Qu.: 11.75
##  Median : 34.50
##  Mean   : 82.50
##  3rd Qu.: 90.00
##  Max.   :981.00
##  NA's   :106
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------ ---------
##  highest value 246 is an   0.06021
##          outlier
## ----------------------------------
##
##   I_5_bis_L_tl
##  Min.   :  0.0
##  1st Qu.: 35.5
##  Median : 75.0
##  Mean   :135.7
##  3rd Qu.:176.5
##  Max.   :946.0
##  NA's   :91
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------ ---------
##  highest value 433 is an   0.05369
##          outlier
## ----------------------------------
##
##     I_5_3_a
##  Min.   :  5000
##  1st Qu.: 14000
##  Median : 17000
##  Mean   : 21463
##  3rd Qu.: 25000
##  Max.   :135000
##  NA's   :127
##
## -------------------------------------
##    Hypothese.alternative     p.value
## ------------------------- ---------
##  highest value 33000 is an   0.3802
##           outlier
## -------------------------------------
##
```

```
##     I_5_3_b
##  Min.   : 5000
##  1st Qu.:15900
##  Median :18000
##  Mean   :21459
##  3rd Qu.:25000
##  Max.   :60000
##  NA's   :127
##
## ------------------------------------
##    Hypothese.alternative     p.value
## -------------------------- ---------
##  highest value 35000 is an   0.5019
##           outlier
## ------------------------------------
##
##     I_6_a_1
##  Min.   : 9.0
##  1st Qu.:15.5
##  Median :22.0
##  Mean   :27.0
##  3rd Qu.:36.0
##  Max.   :50.0
##  NA's   :203
##
## ----------------------------------------
##      Hypothese.alternative        p.value
## ------------------------------ ---------
##  highest value 50 is an outlier   0.3012
## ----------------------------------------
##
##     I_6_a_2
##  Min.   :  10.0
##  1st Qu.:  30.0
##  Median :  70.0
##  Mean   : 616.2
##  3rd Qu.: 203.0
##  Max.   :9999.0
##  NA's   :181
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------ ---------
##  highest value 203 is an   0.07592
##          outlier
## ----------------------------------
##
##     I_6_a_3
##  Min.   :   8.0
##  1st Qu.:  52.5
##  Median : 121.0
##  Mean   : 599.3
##  3rd Qu.: 300.0
##  Max.   :9999.0
```

```
##  NA's   :131
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------- ---------
##  highest value 500 is an   0.06171
##          outlier
## ----------------------------------
##
##     I_6_a_4
##  Min.   :   2.0
##  1st Qu.:  16.0
##  Median :  44.5
##  Mean   : 578.2
##  3rd Qu.: 102.2
##  Max.   :9999.0
##  NA's   :144
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------- ---------
##  highest value 147 is an   0.09481
##          outlier
## ----------------------------------
##
##     I_6_a_5
##  Min.   :   5
##  1st Qu.:   6
##  Median :  39
##  Mean   :4010
##  3rd Qu.:9999
##  Max.   :9999
##  NA's   :201
##
## -----------------------------------
##   Hypothese.alternative     p.value
## ------------------------- ---------
##  highest value 9999 is an   0.6806
##          outlier
## -----------------------------------
##
##     I_6_a_6
##  Min.   :   1
##  1st Qu.:   5
##  Median :  28
##  Mean   :2268
##  3rd Qu.: 300
##  Max.   :9999
##  NA's   :197
##
## -----------------------------------
##   Hypothese.alternative     p.value
## ------------------------- ---------
##  highest value 9999 is an   0.2359
```

```
##          outlier
## ------------------------------------
##
##     I_6_b_1
##  Min.   : 30000
##  1st Qu.: 90000
##  Median :150000
##  Mean   :126667
##  3rd Qu.:175000
##  Max.   :200000
##  NA's   :203
##
## ------------------------------------
##   Hypothese.alternative     p.value
## -------------------------- ---------
##  lowest value 30000 is an   0.2771
##          outlier
## ------------------------------------
##
##     I_6_b_2
##  Min.   : 9999
##  1st Qu.:40000
##  Median :50000
##  Mean   :46489
##  3rd Qu.:55000
##  Max.   :85556
##  NA's   :181
##
## -------------------------------------
##    Hypothese.alternative     p.value
## -------------------------- ---------
##  highest value 85556 is an   0.5632
##           outlier
## -------------------------------------
##
##     I_6_b_3
##  Min.   :   9999
##  1st Qu.:  50000
##  Median :  60000
##  Mean   : 102411
##  3rd Qu.:  87500
##  Max.   :1000000
##  NA's   :131
##
## -------------------------------------
##    Hypothese.alternative      p.value
## -------------------------- ---------
##  highest value 104445 is an    0.72
##           outlier
## -------------------------------------
##
##     I_6_b_4
##  Min.   :   9999
##  1st Qu.:  65625
```

```
##  Median :  90000
##  Mean   : 147624
##  3rd Qu.: 120000
##  Max.   :1000000
##  NA's   :144
##
## -------------------------------------
##    Hypothese.alternative     p.value
## -------------------------- ---------
##  highest value 2e+05 is an   0.1928
##           outlier
## -------------------------------------
##
##     I_6_b_5
##  Min.   :   9999
##  1st Qu.:  70000
##  Median : 200000
##  Mean   : 306000
##  3rd Qu.: 250000
##  Max.   :1000000
##  NA's   :201
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------ ---------
##  lowest value 9999 is an   0.5347
##          outlier
## ----------------------------------
##
##     I_6_b_6
##  Min.   :  9999
##  1st Qu.: 35000
##  Median : 60000
##  Mean   : 85833
##  3rd Qu.: 75000
##  Max.   :280000
##  NA's   :197
##
## ----------------------------------
##   Hypothese.alternative    p.value
## ------------------------ ---------
##  lowest value 9999 is an   0.3389
##          outlier
## ----------------------------------
##
##       I_7
##  Min.   :  41.0
##  1st Qu.: 110.5
##  Median : 180.0
##  Mean   : 625.0
##  3rd Qu.: 917.0
##  Max.   :1654.0
##  NA's   :203
##
```

```
## ------------------------------------
##   Hypothese.alternative     p.value
## ------------------------- ---------
##  highest value 1654 is an   0.07432
##          outlier
## ------------------------------------
##
##       I_10
##  Min.   :  2.00
##  1st Qu.:  8.00
##  Median : 15.00
##  Mean   : 19.64
##  3rd Qu.: 24.00
##  Max.   :110.00
##  NA's   :88
##
## ----------------------------------------
##      Hypothese.alternative        p.value
## ------------------------------ ---------
##  highest value 56 is an outlier   0.1569
## ----------------------------------------
##
##       II_3
##  Min.   :   0.0
##  1st Qu.:   3.0
##  Median :   4.0
##  Mean   : 337.1
##  3rd Qu.:   9.0
##  Max.   :9999.0
##  NA's   :145
##
## ----------------------------------------
##      Hypothese.alternative        p.value
## ------------------------------ ---------
##  highest value 10 is an outlier   0.2419
## ----------------------------------------
##
##      II__11
##  Min.   :       80
##  1st Qu.:    15250
##  Median :  1500000
##  Mean   : 22966456
##  3rd Qu.: 12000000
##  Max.   :600000000
##  NA's   :60
##
## --------------------------------------
##     Hypothese.alternative      p.value
## ---------------------------- ---------
##  highest value 2.1e+07 is an   0.06755
##            outlier
## --------------------------------------
```

Ensuite l'affectation de ces valeurs récupérées aux aberrantes. NB : Les valeurs manquantes et les nsp ne

sont pas concernées par cette imputation. Celles-ci seront gérées dans la suite.

```
newvar=seq(1,length(index_test),1)
for (i in newvar) {
dataAdmin[,index_test[i]][which(dataAdmin[,index_test[i]]>wins[i] &
                                  !is.na(dataAdmin[,index_test[i]]) &
                                  dataAdmin[,index_test[i]]!=9999),]<-wins[i]
}
```

2. Traitement de la base Etablissement : section 3

2.1. Affectation du code 9999 au ne Ne sait pas

Nous uniformisons le codage des *"ne sait pas"* pour facilité le traitement.

```
for (variable in c("III__37","III__36","III__35","III__20","III__8","III__4")) {
  dataAdmin[which(dataAdmin[variable]==9999999999|
                    dataAdmin[variable]==999999999|
                    dataAdmin[variable]==99999999|
                    dataAdmin[variable]==9999999|
                    dataAdmin[variable]==999999|
                    dataAdmin[variable]==99999|
                    dataAdmin[variable]==999),variable] <- 9999
}
```

2.2. Calcul des moustaches gauches et droites pour les variables ayant des outliers

On corrige les valeurs aberrantes en les remplaçant soit par la moustache gauche ou par la moustache droite selon qu'elles soient inférieures (remplacement par la moustache gauche) ou supérieures (par la moustache droite).

```
library(stats)
library(questionr)
mg<-seq(1,6,1)
md<-seq(1,6,1)
list_variable<-c("III__37","III__36","III__35","III__20","III__8","III__4")
for(i in seq(1,6,1)){
mg[i]=quantile(dataAdmin[which(dataAdmin[list_variable[i]]!=9999),
list_variable[i]],0.25,
na.rm=T)-1.5*(quantile(dataAdmin[which(dataAdmin[list_variable[i]]!=9999),
list_variable[i]], 0.75, na.rm = T)-
quantile(dataAdmin[which(dataAdmin[list_variable[i]]!=9999),
                   list_variable[i]], 0.25, na.rm = T))

md[i]=quantile(dataAdmin[which(dataAdmin[list_variable[i]]!=9999),
              list_variable[i]],0.75,na.rm=T)+1.5*(quantile(
             dataAdmin[which(dataAdmin[list_variable[i]]!=9999),
             list_variable[i]], na.rm = T, 0.75)-quantile(
             dataAdmin[which(
             dataAdmin[list_variable[i]]!=9999),list_variable[i]],
             0.25, na.rm = T))
 }
```

2.3. Correction des outliers

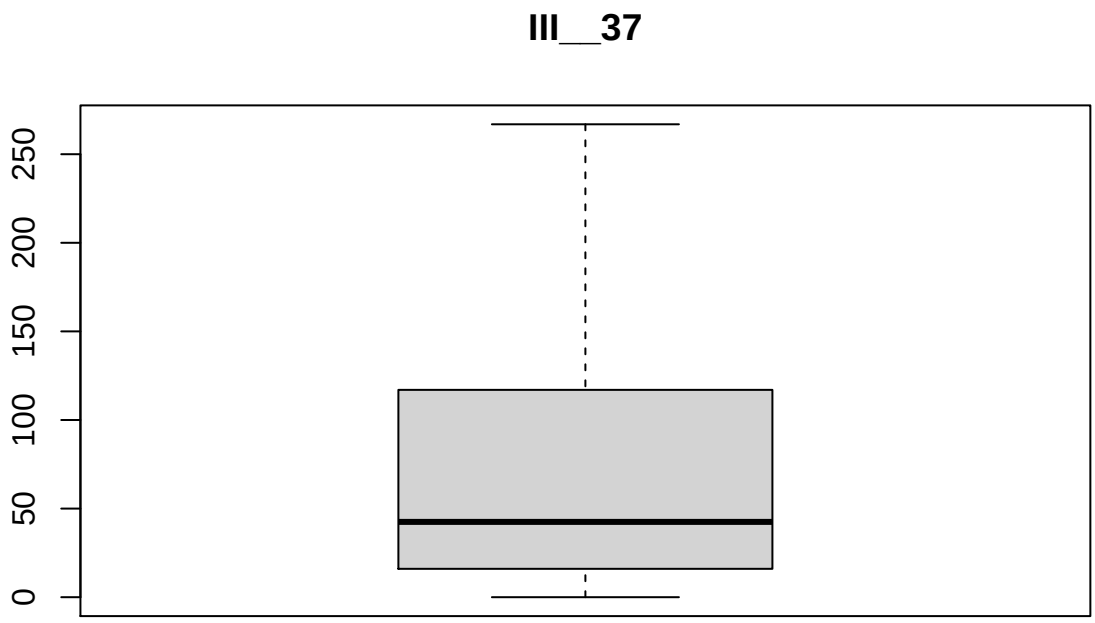
Ici, on applique l'imputation des variables selectionnées afin de corriger le maximum d'incohérence pour les valeurs aberrantes

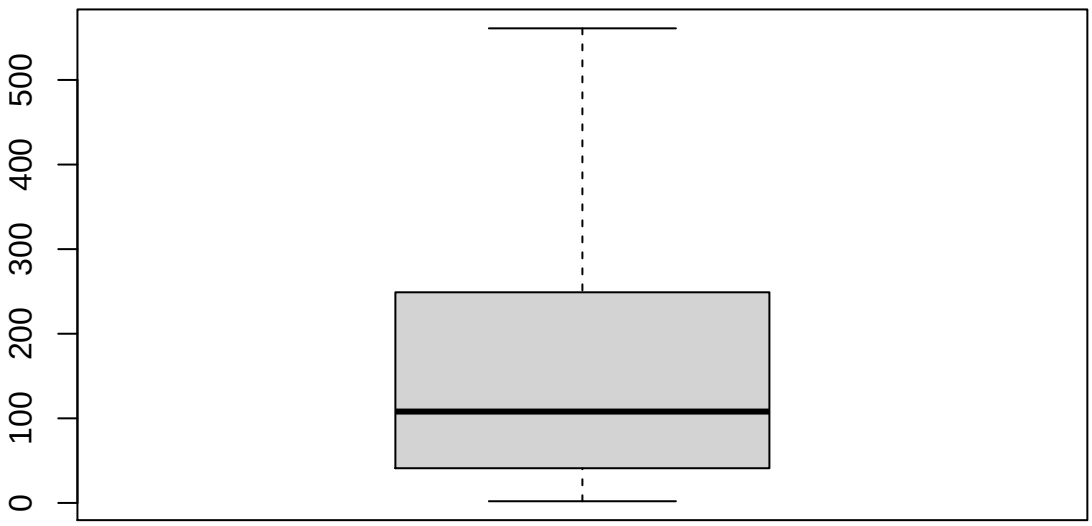
```
#library(dplyr)
list_variable<-c("III__37","III__36","III__35","III__20","III__8","III__4")
for (i in seq(1,6,1)) {
  dataAdmin[which(dataAdmin[list_variable[i]]<mg[i] &
dataAdmin[list_variable[i]]!=9999),list_variable[i]]<-mg[i]

    dataAdmin[which(dataAdmin[list_variable[i]]>md[i] &
dataAdmin[list_variable[i]]!=9999), list_variable[i]]<-md[i]
}
```

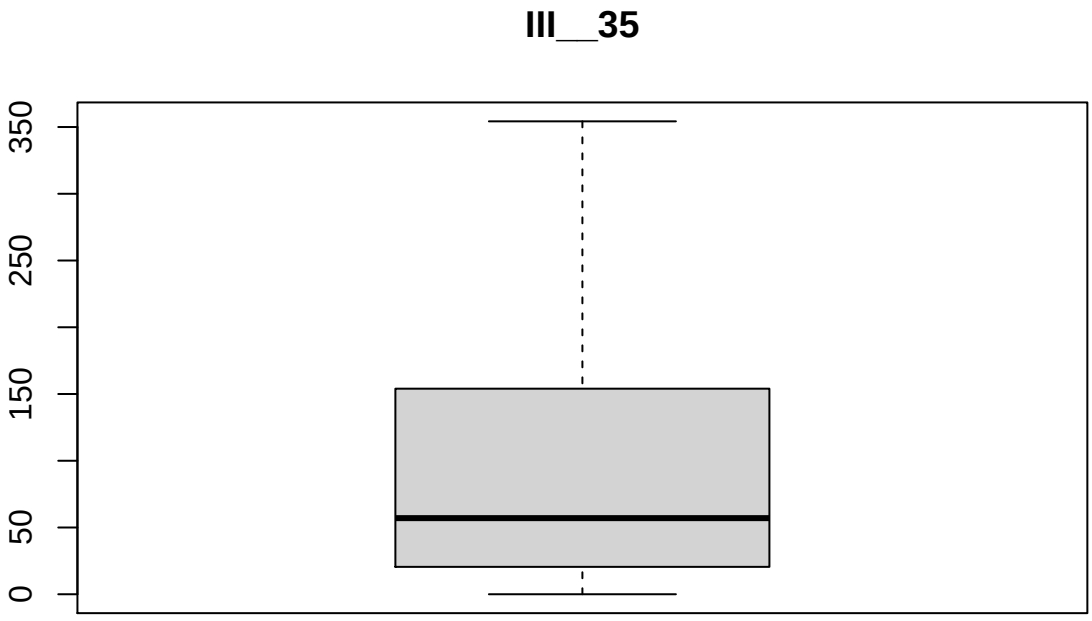
2.4 Vérification si la correction est établie

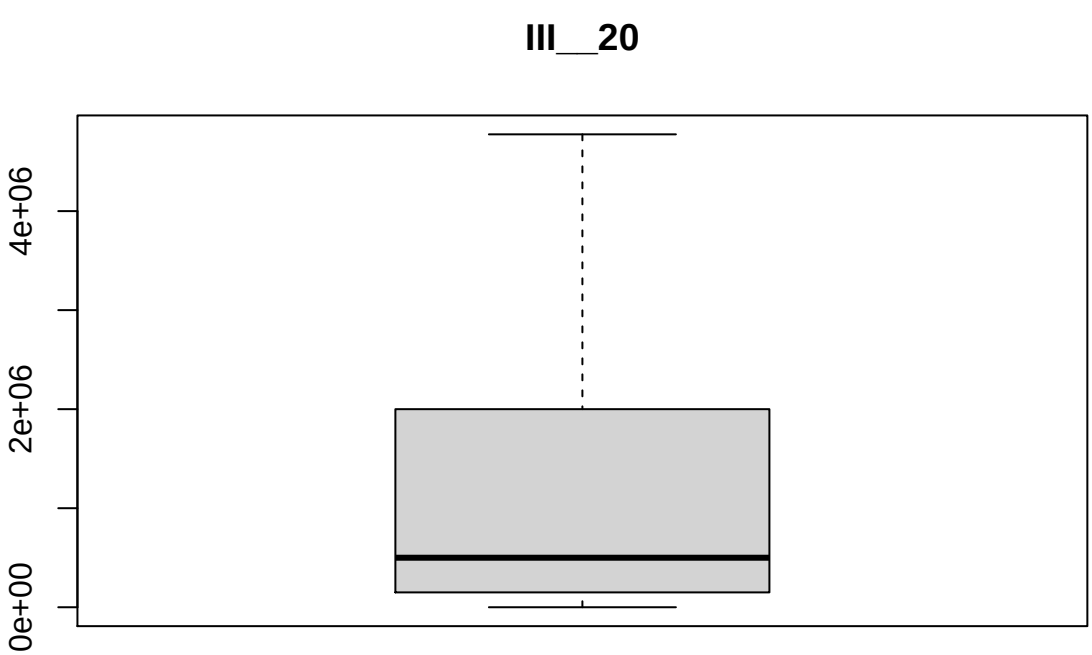
Après vérification, nous constatons que les valeurs inferieures à la moustache gauche ont ete imputees par la moustaches gauches, et celles superieures à la moustache droite ont ete remplace par la moustache droite.

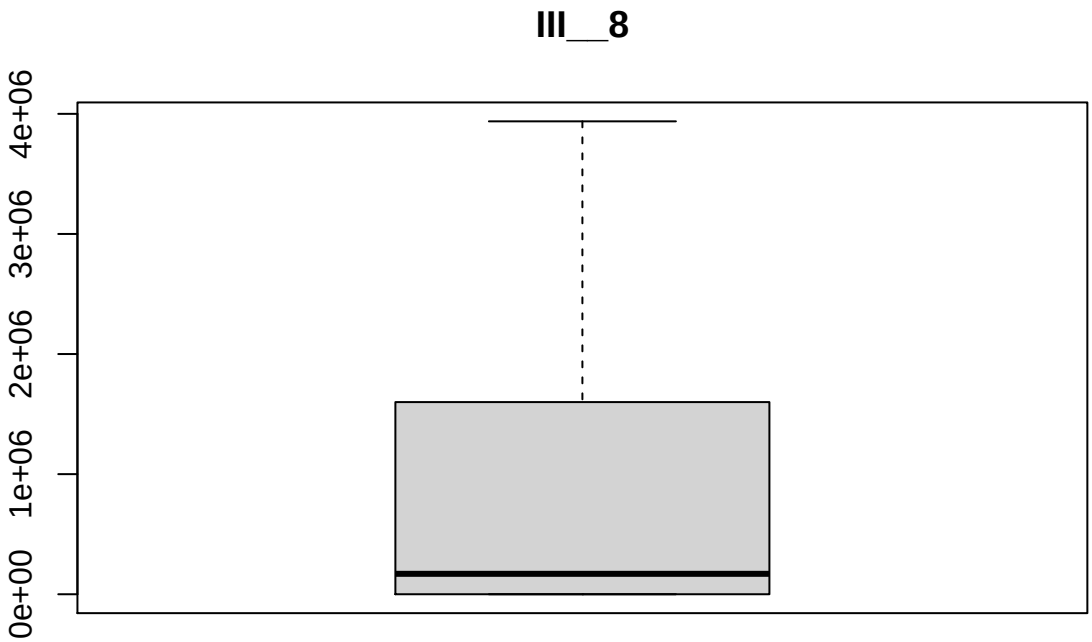
```
for (i in c("III__37","III__36","III__35","III__20","III__8","III__4")) {
  boxplot(dataAdmin[which(dataAdmin[i]!=9999),i] , main=i)
}
```

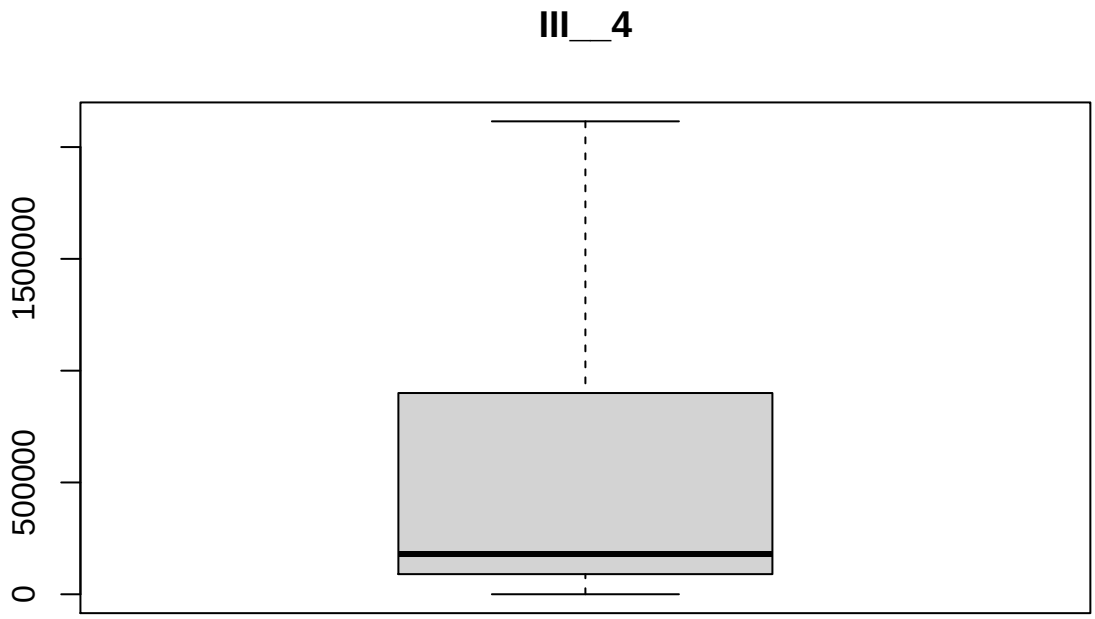



III__36
0
100
200
300
400
500


III__35
0
50
150
250
350


III__20
0e+00
2e+06
4e+06


III__8
0e+00
1e+06
2e+06
3e+06
4e+06



3. Traitement de la base Elève

Le traitement des sections 2 et 3 de la base élève a occasionné l'usage de la *méthode de Hampel pour la détection des outliers* et *la méthode de repondération* pour corriger ces derniers.

A présent nous allons analyser la base élève en détectant et en corrigeant les valeurs aberrantes contenues dans cette base.

3.1. Détection de valeurs aberrantes : Méthode Hampel

3.1.1. Présentation de la méthode de Hampel

La méthode, dite de Hampel, consiste à considérer comme outliers les valeurs en dehors de l'intervalle constitué par la médiane, plus ou moins k déviation absolue de la médiane :

$$I = [median \breve{} k * mad \ ; \ median + k * mad]$$
$$Avec \quad mad = \text{Median Absolute Deviation}$$
$$et \quad mad = median \ (|y_i \breve{} \tilde{y}|)$$
$$et \quad \tilde{y} = median(y)$$

Généralement k est fixé à 3.

3.1.2. Création d'une fonction pour appliquer la méthode

Ici nous créons une fonction permettant de détecter les valeurs aberrantes en utilisant la méthode de Hampel. La fonction prend en entrée la base de donnée, le nom de la variable et le paramètre k. Elle calcule les bornes

supérieur et inférieur de l'intervalle de Hampel. La fonction renvoie une liste contenant les indicatrices outliers supérieurs à la borne supérieur, les indicatrices des outliers inférieurs à la borne inférieur et la liste combinée.

```
out_index <- function(df,var,k){
  # Création des bornes
    binf <- median(df[[var]], na.rm = T) - k*mad(df[[var]], na.rm = T)
    bsup <- median(df[[var]], na.rm = T) + k*mad(df[[var]], na.rm = T)
  # Recupération des index
    out_gauche <- df[[var]] < binf
    out_droite <- df[[var]] > bsup
    union <- out_gauche | out_droite
    out <- list( out_gauche,  out_droite, union)
    names(out) <- c('out_gauche', 'out_droite', 'union')
  return(out)
}
```

3.1.3. Application de la méthode

La méthode ci-dessus sera appliquée aux valeurs aberrantes de la section 2 de la base élève.

a. Importation de la base ELEVE

```
library("haven")
df_eleve <- read_sav("../raw/data_eleve1.sav")
```

b. Section II : Liste de variables continues

Nous commençons d'abord par identifier les variables continues de notre section. Elles se présentent dans le tableau ci-dessous.

```
##
## ------------------------------------------
##  Variable               Labels
## ---------- --------------------------------
##   II_16       Combien depensiez-vous en
##              moyenne, par mois,
##               en connexion internet avant
##                       la pandemiee
##
##   II_17        Combien depensez vous en
##              moyenne par mois
##              en connexion internet avec la
##                        pandemiee
##
##   II_25      Quel volume horaire par jour
##              consacrez vous              en
##                moyenne aux applications ?
##
##   II_29     Si avec repetiteur: Nombre de
##                  jours par semaine ?
##
##   II_30      Si avec repetiteur: Quel est
##                 le paiement par mois ?
## ------------------------------------------
```

```
## Recupération des variables continues de la section
#library("dplyr")
df_sect2 <- df_eleve %>% select(c(II_16,II_17,II_25,II_29,II_30))
```

Détection des outliers pour chaque variable

Maintenant nous allons appliquer notre fonction à toute les variables identifiées comme continues. Ensuite on renvoie le nombre d'outliers pour chaque variables.

```
for (col in colnames(df_sect2)){
  cat("Résultat de la variable: ", col)
  result <- summarise(df_sect2,
                      median = median(df_sect2[[col]], na.rm = T),
          borne_inf = median - 3 * mad(df_sect2[[col]], na.rm = T),
          borne_sup = median + 3 * mad(df_sect2[[col]], na.rm = T)
          )
  pandoc.table(result)
  out <- out_index(df_sect2, col, 3)
  cat("Nombre d'outliers inférieur à la borne inf :",
      sum(out$out_gauche, na.rm = T), "\n")
  cat("Nombre d'outliers supérieur à la borne sup :",
      sum(out$out_droite, na.rm = T), "\n", "\n","\n")

}
```

```
## Résultat de la variable:  II_16
## --------------------------------
##  median   borne_inf   borne_sup
## -------- ----------- -----------
##   2000      -4672       8672
## --------------------------------
##
## Nombre d'outliers inférieur à la borne inf : 0
## Nombre d'outliers supérieur à la borne sup : 217
##
##
## Résultat de la variable:  II_17
## --------------------------------
##  median   borne_inf   borne_sup
## -------- ----------- -----------
##   3000      -5896       11896
## --------------------------------
##
## Nombre d'outliers inférieur à la borne inf : 0
## Nombre d'outliers supérieur à la borne sup : 202
##
##
## Résultat de la variable:  II_25
## --------------------------------
##  median   borne_inf   borne_sup
## -------- ----------- -----------
##    3       -1.448       7.448
## --------------------------------
##
## Nombre d'outliers inférieur à la borne inf : 0
## Nombre d'outliers supérieur à la borne sup : 46
##
##
## Résultat de la variable:  II_29
```

```
## --------------------------------
##  median   borne_inf   borne_sup
## -------- ----------- -----------
##    3       -1.448       7.448
## --------------------------------
##
## Nombre d'outliers inférieur à la borne inf : 0
## Nombre d'outliers supérieur à la borne sup : 0
##
##
## Résultat de la variable:  II_30
## --------------------------------
##  median   borne_inf   borne_sup
## -------- ----------- -----------
##   9000     -31030       49030
## --------------------------------
##
## Nombre d'outliers inférieur à la borne inf : 0
## Nombre d'outliers supérieur à la borne sup : 20
##
##
```

3.2. Correction des valeurs abberrantes: Réduction du poids de sondage

3.2.1. Méthodologie

Une valeur aberrante associée à un poids élevé peut représenter une grande part du totale. Par exemple avec l'estimateur d'Horvizt Thompson, $\hat{y} = \sum \omega_i * y_i$, un poids, ω_i , élevé associé à une observation, y_i , élevée va tirer la valeur estimée. Dès lors la méthode de réduction des poids de sondage permet de neutraliser l'effet du poids en le fixant à 1 pour les valeurs aberrantes.

3.2.2. Application

Afin d'appliquer la méthode de repondération nous aurons besoin de la variable poids qu'il faudra d'abord construire.

a. Création du poids de sondage

Le poids de sondage sera défini ici comme l'inverse de la probabilité d'inclusion. Il sera donc :

$$poids = \frac{Effectif\ de\ l'etablissement}{Effectif\ de\ l'echantillon}$$

L'effectif des établissements sera la somme des effectifs des différents niveaux qui le composent. La liste des variables à sommer pour l'obtenir est ci-dessous.

```
## Base administration
#library(dplyr)
library(pander)
library(questionr)
dataAdmin <- read_sav("../raw/data_admin1.sav")

# Recupération des variables effectifs (seconde)
    df <- slice(lookfor(dataAdmin, "effectif"), -c(13,14,21))
    pandoc.table(df[,2:3])
```

```
##
## ---------------------------------------------
```

```
##    variable                label
## --------------- ------------------------------
##  I_5_bis_S_sec    I.5.Bis.1. Effectif de la
##                      sÃ©rie S en Seconde
##
##  I_5_bis_L_sec    I.5.Bis.2. Effectif de la
##                      sÃ©rie L en Seconde
##
##  I_5_bis_T_sec    I.5.Bis.3. Effectif de la
##                      sÃ©rie T en Seconde
##
##  I_5_bis_G_sec    I.5.Bis.4. Effectif de la
##                      sÃ©rie G en Seconde
##
##  I_5_bis_S_pre    I.5.Bis.5. Effectif de la
##                     sÃ©rie S en PremiÃ¨re
##
##  I_5_bis_L_pre    I.5.Bis.6. Effectif de la
##                     sÃ©rie L en PremiÃ¨re
##
##  I_5_bis_T_pre    I.5.Bis.7. Effectif de la
##                     sÃ©rie T en PremiÃ¨re
##
##  I_5_bis_G_pre    I.5.Bis.8. Effectif de la
##                     sÃ©rie G en PremiÃ¨re
##
##  I_5_bis_S_tl     I.5.Bis.9. Effectif de la
##                     sÃ©rie S en Terminale
##
##  I_5_bis_L_tl     I.5.Bis.10. Effectif de la
##                     sÃ©rie L en Terminale
##
##  I_5_bis_T_tl     I.5.Bis.11. Effectif de la
##                     sÃ©rie T en Terminale
##
##  I_5_bis_G_tl     I.5.Bis.12. Effectif de la
##                     sÃ©rie G en Terminale
##
##     I_6_a_1     Effectif du cycle PrÃ©patoire
##
##     I_6_a_2      Effectif du cycle Technicien
##                             supÃ©rieur
##
##     I_6_a_3       Effectif du cycle Licence
##
##     I_6_a_4           Effectif du cycle
##                       Master/IngÃ©nieur
##
##     I_6_a_5       Effectif du cycle Doctorat
##
##     I_6_a_6         Effectif Autres cycles
## ----------------------------------------------
```

Grâce aux fonctions **mutate()** et **rowSum()** nous pouvons créer l'effectif total de chaque établissement dans

la base dataAdmin. Ensuite il suffira de les joindre dans la base élèves.

```
#library(dplyr)
# Recupération de la liste des noms des variables "effectifs"
  variable <- df['variable']
# Création d'une base temporaire avec seulement les variables "effectif"
  temp2 <- select(dataAdmin, variable$variable )
# Création de l'Effectif des établissements
  dataAdmin <- mutate(dataAdmin, effectif_etablissement = rowSums(temp2, na.rm = T))
# Recupération de la variable "Effectif des établissements"
#et "l'identifiant de l'établissement"
  df_eff_etablissement <- dataAdmin %>% select(etabissement, effectif_etablissement)
```

Effectif des élèves enquêtés par établissement

```
#library(dplyr)
# Pour chaque établissement on compte le nombre d'élève dans la base
df_eleve <- df_eleve %>% group_by(etabissement) %>%
  mutate(effectif_echantillon = n()) %>%  ungroup()
```

Correction de l'id etabissement avant jointure

```
#library(dplyr)
# Gardons les valeurs uniques de dataAdmin
df_eff_etablissement <- df_eff_etablissement %>% distinct(etabissement, .keep_all = T)
# Suppression des na de la variable etablissement
df_eleve <- df_eleve %>% filter(!is.na(etabissement))
```

Nous pouvons maintenant joindre l'effectif échantilloné et celui de l'établissement dans la base elève. Ci-dessous nous pouvons voir la forme de la base pour les 15 première ligne.

```
#library(dplyr)
# Jointure base élève et effectif des établissements
df_eleve <- df_eleve %>% left_join(df_eff_etablissement, by="etabissement")
# Observation des effectifs créés dans la base élève
df_eleve %>% select(etabissement, effectif_etablissement,
                    effectif_echantillon) %>%  arrange(etabissement) %>%
                    head(15)  %>% pandoc.table()
```

```
##
## ----------------------------------------------------------------
##  etabissement   effectif_etablissement   effectif_echantillon
## -------------- ------------------------ ----------------------
##       1                  498                      11
##
##       1                  498                      11
##
##       1                  498                      11
##
##       1                  498                      11
##
##       1                  498                      11
##
##       1                  498                      11
##
##       1                  498                      11
##
##       1                  498                      11
##
```

```
##       1                  498                       11
##
##       1                  498                       11
##
##       1                  498                       11
##
##       1                  498                       11
##
##       2                   NA                       8
##
##       2                   NA                       8
##
##       2                   NA                       8
##
##       2                   NA                       8
## ------------------------------------------------------------
```

Nous remarquons que 189 élèves/étudiants n'ont pas de correspondance dans la base établissement. Pour corriger cette anomalie, nous proposons d'affecter à ces individus le poids moyen.

```
#library(dplyr)
## Création du poids
df_eleve <- mutate(df_eleve, poids = effectif_etablissement/effectif_echantillon)
## Imputation des poids manquant par la moyenne
df_eleve <- mutate(df_eleve, poids = ifelse(is.na(poids), mean(poids, na.rm=T), poids))
## Observation du résultat
df_eleve %>% select(etabissement, effectif_etablissement,
                    effectif_echantillon, poids) %>%
                    arrange(etabissement) %>% head(15)  %>%
                    pandoc.table()
```

```
##
## ------------------------------------------------------------------------
##  etabissement   effectif_etablissement   effectif_echantillon   poids
## -------------- ------------------------ ---------------------- -------
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
##       1                  498                       11            45.27
##
```

```
##       1                  498                      11            45.27
##
##       2                   NA                      8             105.2
##
##       2                   NA                      8             105.2
##
##       2                   NA                      8             105.2
##
##       2                   NA                      8             105.2
## ------------------------------------------------------------------------
```

b. Application de la méthode de repondération

Correction par repondération

```
reponderation <- function(data, var){
  # Recupére l'index des out de var
  out <- out_index(data, var,3)$union
  out_weight <- sum(out, na.rm = T)
  # Evaluer le poids perdu
  lost_weight <-sum(data[out,]$poids-1, na.rm = T)

  # Créer un nouveau poids spécifique à var en assignant un poid de 1 à tous les out
  new_poids <- ifelse(out,1,
              data$poids+lost_weight*data$poids/(sum(data$poids)-out_weight))
  return(new_poids)
}
```

La réduction des poids réduit donc les valeurs des estimateurs (voir tableau ci-dessous).

```
##
## ------------------------------------------------
##  Variables     Estimateur1        Estimateur2
## ----------- ------------------ ------------------
##    II_16     852526331.712653   457880505.064226
##
##    II_17     1032131880.1222    640277008.152717
##
##    II_25     346993.20720068    283602.30619041
##
##    II_29     49174.161852838    49174.161852838
##
##    II_30     275112588.614094   206028036.269659
## ------------------------------------------------
```

Deuxième partie : Imputation des données manquantes

Dans cette deuxième partie il s'agit de détecter et de corriger les valeurs manquantes. La procédure adoptée consiste à d'abord identifier les différentes formes de valeurs manquantes (totales ou partielles). Ensuite, nous appliquons une correction de la non réponse totale par repondération et une correction des non réponse partielle par différentes méthodes d'imputaion (imputation par la moyenne, Hotdeck, KNN, etc). ## 1. Correction de la non réponse : Base établissement Section 1 et 2 ### 1.1. Détection et traitement des-non réponses totales

1.1.1. Détection des-non réponses totales

On crée une variable binaire **is_partial** qui est vraie lorsque l'observation n'est pas une non réponse totale.

```
dataAdmin <- dataAdmin %>% mutate(is_partial=(dataAdmin$s0_02==1))
table(dataAdmin$is_partial)
```

```
##
## FALSE  TRUE
##     6   200
```

On note la présence de deux non-réponses totales. Ces deux observations seront traitées en allouant leur poids aux autres observations afin de compenser leur absence.

1.1.2. Traitement des-non réponses totales par repondération

D'abord, notons que tous les établissements ont le même poids égal à l'unité. Cela est motivé par le fait que l'hypothèse de l'enquête est qu'un recensement est effecué au niveau des établissements.

```
#Création de la variable poids
dataAdmin <- dataAdmin %>% mutate(poids=1)
#Somme des poids des non-réponses totales
poids_perdu <- dataAdmin[which(dataAdmin$is_partial==0),"poids"] %>% sum()
#Redistribution du poids perdus aux autres observations
dataAdmin <- dataAdmin %>% mutate(poids=ifelse(is_partial==T,
 poids+poids_perdu*poids/(sum(poids)-poids_perdu), 0))
#Base finale éliminée des non-réponses totales
dataAdmin <- dataAdmin %>% filter(is_partial==T)
```

1.2. Traitement des-non réponses patielles

1.2.1 Codage des hors champs

On parle de hors champs quand une valeur manquante est du au fait l'enquêté n'est pas concerné par la question. Partculièrement, les sauts automatiques de certaines questions sont des hors champs. Nous allons donc allouer aux "hors champs" des variables quantitatives la valeur 90909 en se basant sur la structure du questionnaire.

Les effectifs : Nous devons parcourir tous les niveaux pour chaque série et recoder par 90909 lorsque la série d'est pas enseigné dans l'établissement ou lorsque le niveau d'étude n'existe pas. L'algorithme ci-dessous traduit la création d'une matrice (niveaux, séries) que l'on va parcourir par deux boucles for afin d'automatiser le recodage. Lorsque la série ou le niveau d'étude n'existent pas, on remplace l'effectf par 90909.

```
#Création d'une matrice niveau*série à parcourir
series=c("Niveaux", "I_4_1_1", "I_4_1_2", "I_4_1_3", "I_4_1_4")
seconde=c("I_5_1_1", "I_5_bis_S_sec", "I_5_bis_L_sec", "I_5_bis_T_sec", "I_5_bis_G_sec")
premiere=c("I_5_1_2", "I_5_bis_S_pre", "I_5_bis_L_pre", "I_5_bis_T_pre",
        "I_5_bis_G_pre")
terminale=c("I_5_1_3", "I_5_bis_S_tl", "I_5_bis_L_tl", "I_5_bis_T_tl", "I_5_bis_G_tl")
serie_niveau=rbind(seconde, premiere, terminale) %>% data.frame()
names(serie_niveau)=series

for (serie in c("I_4_1_1", "I_4_1_2", "I_4_1_3", "I_4_1_4")) {
  for (j in 1:3) {
    a=serie_niveau[j,serie] %>% as.vector()
    #Lorsque la série n'existe pas ou que l'établissement est un institut supérieur
    dataAdmin[which(dataAdmin[serie]==0 | is.na(dataAdmin[serie])),a] <- 90909
    #Lorsque le niveau n'existe pas ou que l'établissement est un institut supérieur
    dataAdmin[which(dataAdmin[serie_niveau[j,1]]==0 |
                    is.na(dataAdmin[serie_niveau[j,1]])),a] <- 90909
  }
```

```
}
```

Les mensualités : Cette question ne concernent que les lycées privés. Lorsqu'il s'agit un institut supérieur ou un établissement public, on remplace par 90909 car c'est un hors-champs.

```
for (var in c("I_5_3", "I_5_3_a", "I_5_3_b")) {
  dataAdmin[which(dataAdmin$I_2==2|dataAdmin$I_3==1),var] <- 90909
}
```

Réduction du personnel : Il s'agit du nombre d'éléments réduits dans le personnel. Lorsqu'il n'y a pas de réduction, la question ne doit pas être posée.

```
dataAdmin[which(dataAdmin$II_2==2),"II_3"] <- 90909
```

Scolarité impayée : Lorsque l'établissement n'a pas enregistré une portion d'impayé, les deux question (pourcentage d'impayé et montant moyen) ne doivent pas être posées.

```
dataAdmin[which(dataAdmin$II_9==2),"II_10"] <- 90909
dataAdmin[which(dataAdmin$II_9==2),"II_11"] <- 90909
```

Correction unité de la variable scolarité impayée : Pour la vriable "II_11", certaines valeurs ont été renseignées en milliers et d'autres à l'unité près. Pour convertir l'ensemble des valeurs à l'unité, nous prenons comme hypothèse que lorsque la valeur renseignée est inférieur ou égale à 10 000, cette valeur est en milliers et on doit donc la multiplier par 1000. Sinon, on la laisse telle qu'elle est renseignée.

```
dataAdmin <- dataAdmin %>% mutate(II__11=ifelse(II__11<10000 &
                            II__11!=9999, II__11*1000, II__11))
```

1.2.2 Imputation des valeurs manquantes

Après avoir traité les non-réponses totales et recoder les hors champs, il nous reste, pour les variables quantitatives, que des valeurs manquantes dues à l'enquête. Nous allons les imputer pour garantir la stabilité des estimateurs qui pourront être calculés avec ces données. Il existe plusieurs méthodes d'imputations applicables selon le contexte et le type de variable. On distingue deux grands groupes : **les méthodes déterministes** et **les méthodes aléatoires**.

```
library(VIM)
```

```
## Loading required package: colorspace
## Loading required package: grid
## VIM is ready to use.
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
##
## Attaching package: 'VIM'
## The following object is masked from 'package:datasets':
##
##     sleep
```

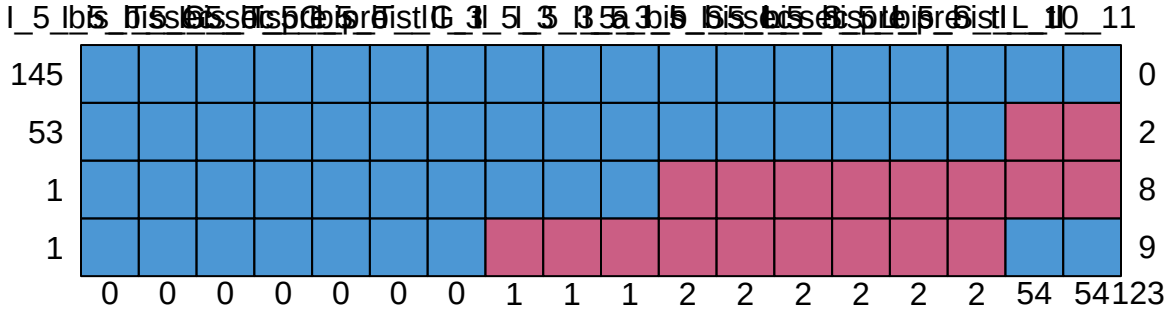
```
library(mice)
```

```
##
## Attaching package: 'mice'
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

Visualisation des missing value à travers les patterns.

```
#Toutes les variables à visualiser dans un vecteur
to_imput=c("I_5_bis_S_sec", "I_5_bis_L_sec", "I_5_bis_T_sec",
          "I_5_bis_G_sec", "I_5_bis_S_pre", "I_5_bis_L_pre",
          "I_5_bis_T_pre", "I_5_bis_G_pre", "I_5_bis_S_tl",
          "I_5_bis_L_tl", "I_5_bis_T_tl", "I_5_bis_G_tl", "I_5_3",
          "I_5_3_a", "I_5_3_b", "II_3", "II__10", "II__11")
#Visualisation graphique
dataAdmin[,to_imput] %>% md.pattern()
```




```
##     I_5_bis_T_sec I_5_bis_G_sec I_5_bis_T_pre I_5_bis_G_pre I_5_bis_T_tl
## 145             1             1             1             1            1
## 53              1             1             1             1            1
## 1               1             1             1             1            1
## 1               1             1             1             1            1
##                 0             0             0             0            0
##     I_5_bis_G_tl II_3 I_5_3 I_5_3_a I_5_3_b I_5_bis_S_sec I_5_bis_L_sec
## 145            1    1     1       1       1             1             1
## 53             1    1     1       1       1             1             1
## 1              1    1     1       1       1             0             0
## 1              1    1     0       0       0             0             0
##                0    0     1       1       1             2             2
##     I_5_bis_S_pre I_5_bis_L_pre I_5_bis_S_tl I_5_bis_L_tl II__10 II__11
```

```
## 145             1             1             1             1      1      1   0
## 53              1             1             1             1      0      0   2
## 1               0             0             0             0      0      0   8
## 1               0             0             0             0      1      1   9
##                 2             2             2             2     54     54 123
```

Nous allons imputer les valeurs manquantes en passant par une méthode détermniste : imputation par la moyenne. La fonction utilisée pour calculer la moyenne de chaque variable est **colMeans()** car dans l'algorithme, tel que défini, les données sur lesquels on calcule la moyenne sont au format *tibble*.

```
for (var in to_imput) {
  dataAdmin[which(is.na(dataAdmin[var])),var] <- dataAdmin[which(dataAdmin[var]!=90909
        & dataAdmin[var]!=9999),var] %>% colMeans()
}
```

Nous procédons maintenant à la suppression du code hors-champs et nous laisserons donc les emplacements vides comme en principe (en accord avec le questionnaire).

```
for (var in to_imput) {
  dataAdmin[which(dataAdmin[var]==90909),var] <- NA
}
```

Ensuite le résumé des variables après traitement.

```
  dataAdmin[to_imput] %>% summary()
```

```
##  I_5_bis_S_sec    I_5_bis_L_sec  I_5_bis_T_sec   I_5_bis_G_sec
##  Min.   :   0.0   Min.   :   0   Min.   :109.0   Min.   :  0.00
##  1st Qu.:  15.0   1st Qu.:  27   1st Qu.:140.8   1st Qu.: 15.00
##  Median :  38.5   Median :  56   Median :172.5   Median : 30.00
##  Mean   : 109.0   Mean   : 127   Mean   :172.5   Mean   : 66.87
##  3rd Qu.: 109.0   3rd Qu.: 142   3rd Qu.:204.2   3rd Qu.: 42.50
##  Max.   :1293.0   Max.   :1144   Max.   :236.0   Max.   :528.00
##  NA's   :98       NA's   :83     NA's   :198     NA's   :185
##  I_5_bis_S_pre    I_5_bis_L_pre  I_5_bis_T_pre   I_5_bis_G_pre
##  Min.   :  0.00   Min.   :   0   Min.   :113.0   Min.   :  0.00
##  1st Qu.: 13.00   1st Qu.:  27   1st Qu.:135.8   1st Qu.:  8.50
##  Median : 29.00   Median :  50   Median :158.5   Median : 25.00
##  Mean   : 77.87   Mean   : 112   Mean   :158.5   Mean   : 60.93
##  3rd Qu.: 85.75   3rd Qu.: 120   3rd Qu.:181.2   3rd Qu.: 39.00
##  Max.   :962.00   Max.   :1022   Max.   :204.0   Max.   :498.00
##  NA's   :98       NA's   :83     NA's   :198     NA's   :185
##   I_5_bis_S_tl    I_5_bis_L_tl    I_5_bis_T_tl    I_5_bis_G_tl
##  Min.   :  0.0   Min.   :  0.0   Min.   : 90.0   Min.   :  0.00
##  1st Qu.: 12.0   1st Qu.: 36.0   1st Qu.:113.5   1st Qu.:  6.00
##  Median : 35.0   Median : 75.0   Median :137.0   Median : 25.00
##  Mean   : 82.5   Mean   :135.7   Mean   :137.0   Mean   : 57.67
##  3rd Qu.: 86.5   3rd Qu.:175.0   3rd Qu.:160.5   3rd Qu.: 48.00
##  Max.   :981.0   Max.   :946.0   Max.   :184.0   Max.   :412.00
##  NA's   :98      NA's   :83      NA's   :198     NA's   :185
##      I_5_3          I_5_3_a          I_5_3_b          II_3
##  Min.   : 5000   Min.   :  5000   Min.   : 5000   Min.   :   0.0
##  1st Qu.:14000   1st Qu.: 14000   1st Qu.:15950   1st Qu.:   3.0
##  Median :16000   Median : 17000   Median :18000   Median :   4.0
##  Mean   :19388   Mean   : 21467   Mean   :21463   Mean   : 337.1
##  3rd Qu.:25000   3rd Qu.: 25000   3rd Qu.:25000   3rd Qu.:   9.0
##  Max.   :55000   Max.   :135000   Max.   :60000   Max.   :9999.0
```

```
##  NA's   :120     NA's   :120      NA's   :120     NA's   :139
##      II__10           II__11
##  Min.   :  1.00   Min.   :     9999
##  1st Qu.: 43.75   1st Qu.:   863000
##  Median : 56.40   Median :  6000000
##  Mean   : 56.40   Mean   : 23938047
##  3rd Qu.: 75.00   3rd Qu.: 25601494
##  Max.   :100.00   Max.   :600000000
##
```

2. Correction de la non-réponse: base établissement section 3

2.1 Correction des hors champs

Dans cette partie, on detecte les hors champs en les remplaeant par le code 90909 que nous avons ainsi defini. Pour une meilleure apprehension, les commenataires sont toutefois joints au bout des codes specifies pour mieux cerner les parties où l'on distingue les hors champs.

```
#library(haven)
#library(dplyr)
  dataAdmin <-  dataAdmin %>% mutate(is_partial = (s0_02 ==1))

  table(dataAdmin$is_partial) ## On obtient 6 non réponses totales
```

```
##
## TRUE
##  200
```

```
  dataAdmin <-  dataAdmin %>% filter(is_partial==T)
  # Suppression des doublons
  dataAdmin <-  dataAdmin %>% distinct(dataAdmin, keep_all=T)

  # Visualisation des numéros de colonnes des varaibles
      ##names(dataAdmin)

  ## Detection et codage des hors champs et des ne sait pas
    ## Detecter les hors champs en se basant sur le questionnaire
    ## Les hors champs auront le code : 90909

# Ici, si l'etablissement affirme ne pas s'engager dans les cours en ligne,
# on remplace tous les NA engendres par les vides par le code 90909.
for (col in 132:173){
   dataAdmin[, col] <- sapply(dataAdmin[, col],
    function(x){
      if(is.numeric(x)){
        x <- ifelse(dataAdmin$III__1__1!=1 & is.na(x), 90909, x)
      }else{
        x <- ifelse(dataAdmin$III__1__1!=1 & is.na(x), "90909", x)
      }

    })

}

# Lorsque l'etablissement a octroye des forfaits internet / allocation internet
#a ses etudiants et enseignants alors on renseigne le montant
```

```
#total correspondant au cout engendre par l'octroi.
  dataAdmin$III__4<-ifelse(dataAdmin$III__3==4 &
                             is.na(dataAdmin$III__4), 90909,dataAdmin$III__4)


#   La question III__5 est posee lorsque le repondant e renseigner
# la modalite cours e distance e la question III.1. Elle permet
# de faire un filtre sur la question suivant qui sera sautee au
# cas oe l'etablissement repondait NON a cette question.

  for (col in 156:167){
   dataAdmin[, col] <- sapply(dataAdmin[, col],
    function(x){
      if(is.numeric(x)){
        x <- ifelse(dataAdmin$III__5==2 & is.na(x), 90909, x)
      }else{
        x <- ifelse(dataAdmin$III__5==2 & is.na(x), "90909", x)
      }

    })

  }

# La question III__11 renseigne si l'etablissement
#fait des cours en presentiel. Ainsi, lorsque la reponse
#est NON, on passe a la question III__24 sur la strategies
#adoptees pour la reinsertion des eleves ayant abandonnes.
#Les hors champs sont toujours renseignes par le code 90909.

  for (col in 174:192){
   dataAdmin[, col] <- sapply(dataAdmin[, col],
    function(x){
      if(is.numeric(x)){
        x <- ifelse(dataAdmin$III__11==2 & is.na(x), 90909, x)
      }else{
        x <- ifelse(dataAdmin$III__11==2 & is.na(x), "90909", x)
      }

    })

  }


# (Si III.27.0 = 1, allez a III.30) c'est a dire si les cours en ligne sont utilises
#comme methodes d'evaluation, on applique un saut. De ce fait, il s'ensuit une
#possibilite de NA dans les variables suivantes. D'ou la necessite de corriger les non
#reponses pour les variables auxlesquels les questions ne leur concernent pas et
#detecter bien evidemment, les vraies valeurs manquantes.

    for (col in 193:209){
   dataAdmin[, col] <- sapply(dataAdmin[, col],
    function(x){
      if(is.numeric(x)){
```

```
        x <- ifelse(dataAdmin$III__26__0==1 & is.na(x), 90909, x)
      }else{
        x <- ifelse(dataAdmin$III__26__0==1 & is.na(x), "90909",x)
      }

    })

    }

# Si III.32=2, mettre fin au questionnaire. Et, sur ce, les hors
# champs seront corriges pour voir si le saut a un reel impact
# sur la presence ou non des valeurs manquantes.

  for (col in 210:216){
   dataAdmin[, col] <- sapply(dataAdmin[, col],
    function(x){
      if(is.numeric(x)){
        x <- ifelse(dataAdmin$III__31==2 & is.na(x), 90909, x)
      }else{
        x <- ifelse(dataAdmin$III__31==2 & is.na(x), "90909",x)
      }

    })

  }
```

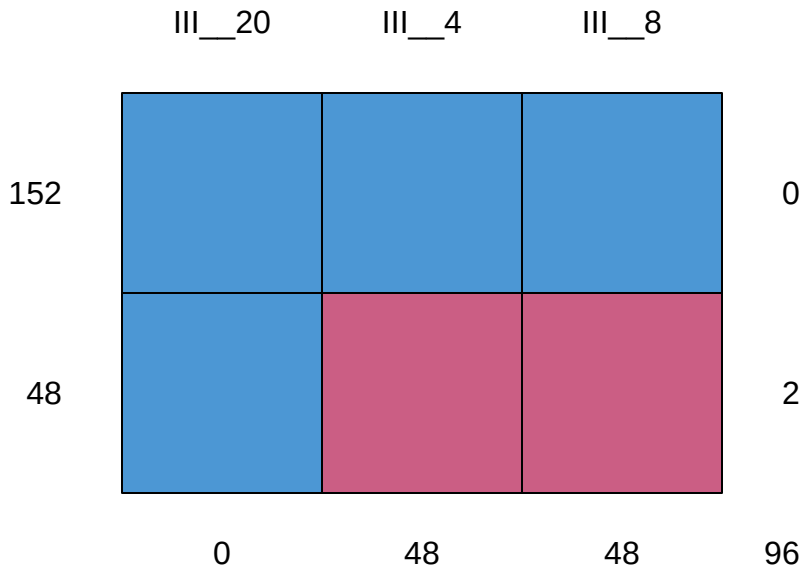
2.2. Imputation des données manquantes

Afin d'aborder correctement l'imputation des donnees manquantes, il faut en distinguer les causes, surtout si elles ne sont pas le simple fruit du hasard.

2.2.1. Visualisation des données manquantes par md.patterns

La visualisation des depenses par md.pattern pour simplifier le modele d'imputation. En outre, le modele manquant pourrait suggerer quelles variables pourraient potentiellement etre utiles pour l'imputation des entrees manquantes. Ainsi, en faisant le md.pattern, on remarque que: -la variable III___20 n'a pas de donnees manquantes -la variable III___8 a une seule donnee manquante -la variable III___4 a deux donnees manquantes. En plus de cela, on peut voir aussi que 67, 85, 48 etablissements ont respectivement 0, 1 et 2 valeurs manquantes.

```
library(mice)
md.pattern(dataAdmin[,c("III__4","III__8", "III__20")])
```




```
##     III__20 III__4 III__8
## 152       1      1      1  0
## 48        1      0      0  2
##           0     48     48 96
```

2.2.2. Imputation des dépenses

L'imputation des valeurs manquantes pour les depenses des etablissements est examinee dans notre cas de figure par la moyenne. En effet, vue que l'on s'interesse au prive et comme que le Coronavirus etait à ses debuts, donc toutes les ecoles ont quasiment adopte les memes demarches, en organisant les cours en ligne via les plateformes ou en allouant des forfaits pour appuyer leur personnel ou eleves etc. D'oe l'utilite de la moyenne qui sera representative.

```
#library(dplyr)
library(stats)

# Depenses liees aux allocations de forfaits
a<-filter(dataAdmin,  dataAdmin$III__4!=9999  &  dataAdmin$III__4!=90909)
dataAdmin$III__4<-ifelse(is.na(dataAdmin$III__4) ,
                         mean(a$III__4, na.rm = T), dataAdmin$III__4)

# Depenses liees a l'organisation des cours en ligne
b<-filter(dataAdmin,  dataAdmin$III__8!=9999  &  dataAdmin$III__8!=90909)
dataAdmin$III__8<-ifelse(is.na(dataAdmin$III__8) ,
                         mean(b$III__8, na.rm = T), dataAdmin$III__8)

# Depenses liees  l'organisation des cours en presentiel
c<-filter(dataAdmin,  dataAdmin$III__20!=9999  &  dataAdmin$III__20!=90909)
```

```
dataAdmin$III__20<-ifelse(is.na(dataAdmin$III__20) ,
                          mean(c$III__20, na.rm = T), dataAdmin$III__20)
```

2.2.2. Imputation des effectifs

Pour retrouver les effectifs des etablissements ayant des donnees manquantes, nous procederons par une imputation simple par la moyenne.

```
#library(dplyr)
library(stats)

# Effectif total de votre etablissement cette annee
d<-filter(dataAdmin, dataAdmin$III__35!=9999 & dataAdmin$III__35!=90909)
dataAdmin$III__35<-ifelse(is.na(dataAdmin$III__35),
                      mean(d$III__35,na.rm=T),dataAdmin$III__35)

# Nombre d'admis au bac cette annee
e<-filter(dataAdmin, dataAdmin$III__36!=9999 & dataAdmin$III__36!=90909)
dataAdmin$III__36<-ifelse(is.na(dataAdmin$III__36),
                      mean(e$III__36,na.rm=T),dataAdmin$III__36)

# Effectif de l'annee passee
f<-filter(dataAdmin, dataAdmin$III__37!=9999  & dataAdmin$III__37!=90909)
dataAdmin$III__37<-ifelse(is.na(dataAdmin$III__37),
                      mean(f$III__37,na.rm=T),dataAdmin$III__37)
```

2.2.3. Imputation des variables qualitatives

L'Imputation se fera dans notre cas par la methode du plus proche voisin Knn. On impute avec les variables "etabissement", "s0_02", "III___1","III___11","III___18","III___15","III___29" et ce, selon les 10 plus proches voisin.

```
## Identification des variables concernees
list_index <- c("III__1__1", "III__1__2", "III__1__3", "III__1__4", "III__2__1",
                "III__2__2","III__2__3","III__2__4","III__2__5",
                "III__2__6","III__2__7","III__2__8","III__2__9",
                "III__2__10","III__3","III__5","III__6__1",
                "III__6__2","III__6__3","III__6__4","III__6__5",
                "III__6__6","III__7","III__9","III__10__1","III__10__2",
                "III__10__3","III__10__4","III__14",
                "III__19","III__26__0","III__26__1","III__26__2","III__26__3",
                "III__26__4","III__26__5","III__26__6","III__27__1","III__27__2",
                "III__27__3", "III__28")


## Methode KNN :
# Chargement de la library
library(VIM)
nb_na <- c()
## Transformation des variables en facteurs
for (col in list_index){
  dataAdmin[[col]] <- dataAdmin[[col]] %>% factor()
}
## Imputation avec KNN et observation des na
for (col in list_index){
 dataAdmin <-  kNN(dataAdmin, variable = col,
```

```
                    dist_var = c("etabissement", "s0_02", "III__1",
                                 "III__11", "III__18", "III__15","III__29" ), k=10)
}
```

3. Analyse et correction de la non réponse : Base élève

Dans la phase de détection et de correction des données manquantes nous avons, d'abord , comme dans la base administration, *identifié les types de données manquantes (non réponse totale et non réponse partielle)*. Ensuite, nous procédons à la détection et au *recodage des hors champs* puis, enfin, nous proposons une *imputation selon le contexte de l'enquête, une imputation avec la méthode Hotdeck et une imputation avec la méthode KNN.*

Nous pouvons distinguer deux types de non réponse : la *non réponse totale* et la *non réponse partielle*. Nous proposotion une méthode permettant leur détection ainsi que leur correction.

3.1 Détection et correction des non réponses totales

3.1.1 Détection de la non réponse totales

Dans notre analyse, nous considérons comme *non réponse totale* tout individu qui ne souhaite *ni participer à l'enquête, ni remettre l'enquête à une autre date.* La détection de ces derniers se fait comme suit.

```
## library(dplyr)
## Création d'une variable indicatrice des non répondant partiels
df_eleve <- df_eleve %>% mutate(is_partial = (s0__2 ==1))

## Effectif des cas de non réponse total (modalité false)
table(df_eleve$is_partial)
```

```
##
## FALSE  TRUE
##    12  1850
```

3.1.2 Correction par repondération des non réponses totales

La repondération consiste à allouer le poids des non répondants totaux à ceux qui ont répondu totalement ou partiellement à toutes les questions. Après repondération on restreind notre échantillon à ces derniers.

```
## Calcul du poids total des non répondant total
  poids_na_total = sum(df_eleve[!df_eleve$is_partial,]$poids)
## Repondération
  df_eleve <- df_eleve %>% mutate(poids = ifelse(is_partial,
       poids + poids_na_total*poids/(sum(poids)-poids_na_total), 0 ))
## Suppression des non réponses totales
  df_eleve <-  df_eleve %>% filter(is_partial==T)
```

3.2 Détection et codage des hors champs

Afin de ne pas confondre les non réponses hors champs et celles dites de valeur manquante, nous proposons de coder les manquants hors champs par le code 90909 pour les champs numériques et "90909" pour les charactères. Ainsi nous allons parcourrir la base en identifiant les variables de saut et appliquer le codage qu'il faut.

Après avoir transformé les manquants des hors champ par les codes qu'il faut nous pouvons facilement détecter les valeurs manquantes de notre base et les corriger. Sachant que la base est constituée de sections lesquelles subdivisées en sous section, nous allons procéder sous section par sous section.

3.3 Détection et correction de la non réponse partielle

3.3.1 STRATEGIES DE RESILIENCE ET UTILISATION DES TIC

```
## Détection des valeurs manquantes
##
verif_na <- df_eleve[, 44:239] %>%
            is.na() %>%
            colSums() %>%
            as.data.frame()
names(verif_na) <- "Nb_na"
verif_na$variable <- names(df_eleve[, 44:239])

verif_na[which(verif_na$Nb_na!=0),] %>% select(Nb_na) %>%
  arrange(desc(Nb_na)) %>% pandoc.table()
```

```
##
## ------------------------------------------------------
##                                              Nb_na
## ------------------------------------------- -------
##  **reserved_name_for_field_list_labels_80**    1837
##
##  **reserved_name_for_field_list_labels_87**    1834
##
##                 **II__15__9**                 1832
##
##  **reserved_name_for_field_list_labels_98**    1827
##
##               **II__14__10__1**               1827
##
##  **reserved_name_for_field_list_labels_126**   1784
##
##                 **II__15__8**                 1751
##
##               **II__14__9__1**                1737
##
##                  **poids2**                   1289
##
##                 **II__14__8**                 1207
##
##  **reserved_name_for_field_list_labels_232**    880
##
##                 **II__15__7**                  860
##
##                 **II__15__3**                  846
##
##                 **II__15__4**                  768
##
##                 **II__15__2**                  626
##
##                 **II__14__7**                  617
##
##                 **II__14__3**                  609
##
##                 **II__14__4**                  547
```

```
##
##                **II__15__1**                  515
##
##                **II__14__2**                  441
##
##                **II__15__5**                  410
##
##                **II__14__1**                  360
##
##                **II__14__5**                  287
##
##                **II__15__6**                  195
##
##  **reserved_name_for_field_list_labels_166**    180
##
##         **effectif_etablissement**             135
##
##                **II__14__6**                   99
##
##                  **II_17**                     96
##
##                  **II_16**                     91
##
##                 **III__14**                    78
##
##                 **III__17**                    78
##
##                **III__6__1**                   31
##
##                **III__6__2**                   31
##
##                **III__6__3**                   31
##
##                 **III__0**                      4
##
##                 **III__1**                      4
##
##                 **III__2**                      4
##
##                 **III__3**                      4
##
##                 **III__4**                      4
##
##                 **III__5**                      4
##
##                 **III__7**                      4
##
##                 **III__8**                      4
##
##                 **III__9**                      4
##
##                 **III__10**                     4
##
##                 **III__11**                     4
```

```
##
##                  **III__12**                    4
##
##                  **III__13**                    4
##
##                  **III__15**                    4
##
##                  **III__16**                    4
## ------------------------------------------------------
```

Nous constatons que les valeurs manquantes concernent la sous section sur l'utilisation des applications (volume d'utilisation et variation), les dépenses avant et pendant la pandémie et la section 3.

a. Utilisation des applications

Sur les volumes horaires consacrés aux applications (II___14___1, II___14___2, II___14___3, II___14___4, ...), la question est posée sans condition sur les questions antérieures, donc on s'attendait à 1839 réponses. Cependant, pour facebook et instagram, par exemple, *1479/1839* et *1398/1839* se sont respectivement prononcés sur la question.

Nous supposons que ces cas vide correspondent à des cas où l'individu n'utilise pas l'application. Les na sont imputés par 0.

```
# Recupération des indices des variables concercées
list_index <- 88:99
#library(dplyr)
nomb_na <- c()
i=1

## Imputation
for (col in list_index){
  df_eleve[, col] <- sapply(df_eleve[, col],
  function(x){
      x <- ifelse(is.na(x), 0, x)
    })
}
```

b. Variation des volumes horaires

En restant toujours sur les applications, on se rend compte que l'effectif ayant donné des informations sur l'utilisation des applications (par exemple *1479 pour facebook*) est supérieur à celui ayant renseigné les variations sur l'utilisation (*1324 pour facebook toujours*). Pour cette anomalie nous proposons d'utiliser la méthode des *K plus proche voisin* pour imputer la valeur manquante. *Dans le cas de variables catégorielle la fonction impute les modalités manquantes par la modalité maximale des k plus proche voisin.* Les variables concernées sont II___15___1, II___15___2 à II___15___9.

Pour cette méthode nous utilisons pour la distance des variables telles que : - etabissement : le code de l'établissement - s0___2 : souhait de participer à l'enquête - II___2 : Moyen de connexion à internet - II___4 : Problème de connexion - II_17 : Dépense de connexion

```
## Identification des variables concernées
list_index2 <- c("II__15__1", "II__15__2", "II__15__3",
                 "II__15__4", "II__15__5", "II__15__6", "II__15__7",
                 "II__15__8", "II__15__9")

## Méthode KNN :
# Chargement de la library
library(VIM)
```

```
nb_na <- c()
## Transformation des variables en facteur
for (col in list_index2){
  df_eleve[[col]] <- df_eleve[[col]] %>% factor()
}
## Imputation avec KNN et observation des na
for (col in list_index2){
 df_eleve <-  kNN(df_eleve, variable = col,
                  dist_var = c("etabissement", "s0__2", "II__2",
                               "II__4", "II_17"), k=10)
}
```

c. Dépense de connexion

Les dépenses de connexion (II___16 et II___17) concernent les *1839 et seuls 1748 et 1743* ont répondu, respectivement pour les *dépenses avant et pendant la pandémie*. Pour ces variables continues nous proposons la méthode hotdeck pour l'imputation avec comme variable de domaines l'établissement et le statut de l'individu (étudiant ou élève)

```
## Chargement de la librairie VIM pour l'imputation par hotdeck
library(VIM)
df_eleve <- hotdeck(df_eleve, variable = "II_16",
                    domain_var = c("etabissement", "I__6") )
df_eleve <- hotdeck(df_eleve, variable = "II_17",
                    domain_var = c("etabissement", "I__6") )
```

3.3.2 APPRECIATION DES STRATEGIES DE RESILIENCE DES ETABLISSEMENTS ET DE L'ETAT

Avez vous repris les cours pendant la pandémie (III___0) : 1847 réponses sur 1851. Nous constatons que ces 4 individus sont absents pour toute les autres variables de la section. On en conclu qu'il n'ont pas repris les cours en présentiel. On impute la valeur 0 à III___0.

```
## Correction de la variable III__0
df_eleve$III__0 <- if_else(is.na(df_eleve$III__0), 0, 1)
```

Les valeurs manquantes des questions de cette section sont donc des hors champs. On reprend donc le codage de ces valeurs manquantes en tant que hors champs.

```
for (col in 197:239){
   df_eleve[, col] <- sapply(df_eleve[, col],
    function(x){
      if(is.numeric(x)){
        x <- ifelse(df_eleve$III__0!=1 & is.na(x), 90909, x)
      }else{
        x <- ifelse(df_eleve$III__0!=1 & is.na(x), "90909", x)
      }

    })
}
```

Ainsi donc la section 2 et 3 de la base élève sont corrigées de toutes valeurs manquantes.

Conclusion

En somme, le travail d'apurement que nous avons effectué s'est inspiré grandement du cours de pratique des enquêtes que nous avons effectué en classe d'ITS4. Au terme de l'apurement de nos bases établissements et

élèves, nous avons corrigé aux mieux le maximum d'incohérences par les méthodes de Grubbs, Hampel et Boxplot selon leur adaptabilité aux différentes parties des bases traitées aux fins de pouvoir faire une analyse synthétique et avoir une information claire et précise à partir de ces données d'enquête.