P7

brunop31

22/07/2020

```
###On télécharge les libraries utiles
library(dplyr)
library(ggplot2)
library(gglorenz)
library(tidyr)
library(reshape2)
library(reldist)
library(rstatix)
library(car)
library(lmtest)
library(olsrr)
library(nortest)
library(corrplot)
select<-dplyr::select
```

```
###Mission 1#################################################################

###On charge le tableau principal du projet de la WID
df<-read.csv("data-projet7.csv")

df$income<-df$income%>%as.vector()%>%sub(",",".",.)%>%
  as.numeric()

df$gdpppp<-df$gdpppp%>%as.vector()%>%sub(",",".",.)%>%
  as.vector()%>%as.numeric()

###Je supprime les pays dont les données sont trop éloignées de 2008
###car mon étude est faite sur cette année là
###Je supprime des pays qui poseront problème plutard
df<-filter(df, year_survey %in% 2006:2010)%>%
  filter(!(country %in% c("FJI", "XKX", "PSE")))

###J'augmente de 3.6% les revenu de 2006 car la croissance du gdpppp
###mondial a été de 3.6% entre 2006 et 2008
df[df$year_survey == 2006,"gdpppp"]<-
  df[df$year_survey == 2006,"gdpppp"]*1.036

df[df$year_survey == 2006,"income"]<-
  df[df$year_survey == 2006,"income"]*1.036

###Je divise par 0.97 les pays étudié en 2009 car le gdpppp mondiale
```

```r
###a chuté de 3% entre 2008 et 2009
df[df$year_survey == 2009,"gdpppp"]<-
  df[df$year_survey == 2009,"gdpppp"]/0.97

df[df$year_survey == 2009,"income"]<-
  df[df$year_survey == 2009,"income"]/0.97

###Les autres variations sont négligeable je laisse les données 2007 et
###2011 telles quelles

###Je considère que mes données correspondent à l'année 2008
###après les ajustements
df$year_survey<-2008

###Je fait des vérification sur les na et le nombre de ligne par pays
sapply(df,function(x) sum(is.na(x)))
```

```
##      country  year_survey     quantile nb_quantiles       income       gdpppp
##            0            0            0            0            0            0
```

```r
df%>%group_by(country)%>%count()
```

```
## # A tibble: 111 x 2
## # Groups:   country [111]
##    country     n
##    <fct>   <int>
##  1 ALB       100
##  2 ARG       100
##  3 ARM       100
##  4 AUT       100
##  5 AZE       100
##  6 BEL       100
##  7 BFA       100
##  8 BGD       100
##  9 BGR       100
## 10 BIH       100
## # ... with 101 more rows
```

```r
df%>%group_by(year_survey)%>%count()
```

```
## # A tibble: 1 x 2
## # Groups:   year_survey [1]
##   year_survey     n
##         <dbl> <int>
## 1        2008 11099
```

```r
###Il manquait un percentile à la lituanie je les ajouté en faisant
###la moyenne du précédant et du suivant
df<-df%>%rbind(df[5940,])

df[11100,"quantile"]<-41
```

```r
df[11100,"income"]<-(4868.4507 + 4895.8306)/2

df<-df%>%arrange(country, quantile)

rownames(df)<-1:11100

sapply(df,function(x) sum(is.na(x)))
```

```
##      country  year_survey     quantile nb_quantiles       income       gdpppp
##            0            0            0            0            0            0
```

```r
###Je charge une table pour faire la correspondance entre
###l'écriture en 3 lettres des pays et le nom complet
country_full<-read.csv("sql-pays.csv", encoding = "UTF-8", header = FALSE)%>%
  select("country" = V4, "country_full" = V5)

country_full$country<-country_full$country%>%as.vector()
country_full$country_full<-country_full$country_full%>%as.vector()

###J'en ajoute deux qui sont présent dans mon étude et pas sur le fichier
country_full[242,]<-c("SRB","Serbie")
country_full[243,]<-c("MNE","Monténégro")

###Je change le nom de la macédoine du nord (trop long)
country_full[country_full$country == "MKD", "country_full"]<-"Macédoine du Nord"

###Je fait une jointure entre mes deux tables pour ajouter la colonne
###nom complet a ma table de base
dfs<-inner_join(country_full,df)
```

```
## Joining, by = "country"
```

```r
###Je charge une table de la FAO avec la pop en 2008 des pays
fao<-read.csv("pop_2008.csv",
              encoding = "UTF-8")%>%
  select("country_full" = "Zone", "pop" = "Valeur")

###Je modifie les country_full qui ne correspondent pas exactement
fao$country_full<-fao$country_full%>%as.vector()

fao$country_full[fao$country_full == "États-Unis d'Amérique"]<-"États-Unis"

fao$country_full[fao$country_full ==
               "Bolivie (État plurinational de)"]<-"Bolivie"

fao$country_full[fao$country_full == "République centrafricaine"]<-
  "République Centrafricaine"

fao$country_full[fao$country_full == "Chine, continentale"]<-"Chine"

fao$country_full[fao$country_full == "Chine, Taiwan Province de"]<-
  "Taïwan"
```

```r
fao$country_full[fao$country_full == "République démocratique du Congo"]<-
  "République Démocratique du Congo"

fao$country_full[fao$country_full == "Tchéquie"]<-"République Tchèque"

fao$country_full[fao$country_full == "République dominicaine"]<-
  "République Dominicaine"

fao$country_full[fao$country_full == "Iran (République islamique d')"]<-
  "République Islamique d'Iran"

fao$country_full[fao$country_full == "République démocratique populaire lao"]<-
  "République Démocratique Populaire Lao"

fao$country_full[fao$country_full == "Soudan (ex)"]<-"Soudan"

fao$country_full[fao$country_full == "Eswatini"]<-"Swaziland"

fao$country_full[fao$country_full == "République arabe syrienne"]<-
  "République Arabe Syrienne"

fao$country_full[fao$country_full ==
  "Royaume-Uni de Grande-Bretagne et d'Irlande du Nord"]<-
  "Royaume-Uni"

fao$country_full[fao$country_full ==
               "Venezuela (République bolivarienne du)"]<-"Venezuela"


###Je fais une jointure entre ma table principale et la table de la FAO
###pour obtenir ma colonne pop
df<-inner_join(dfs,fao)
```

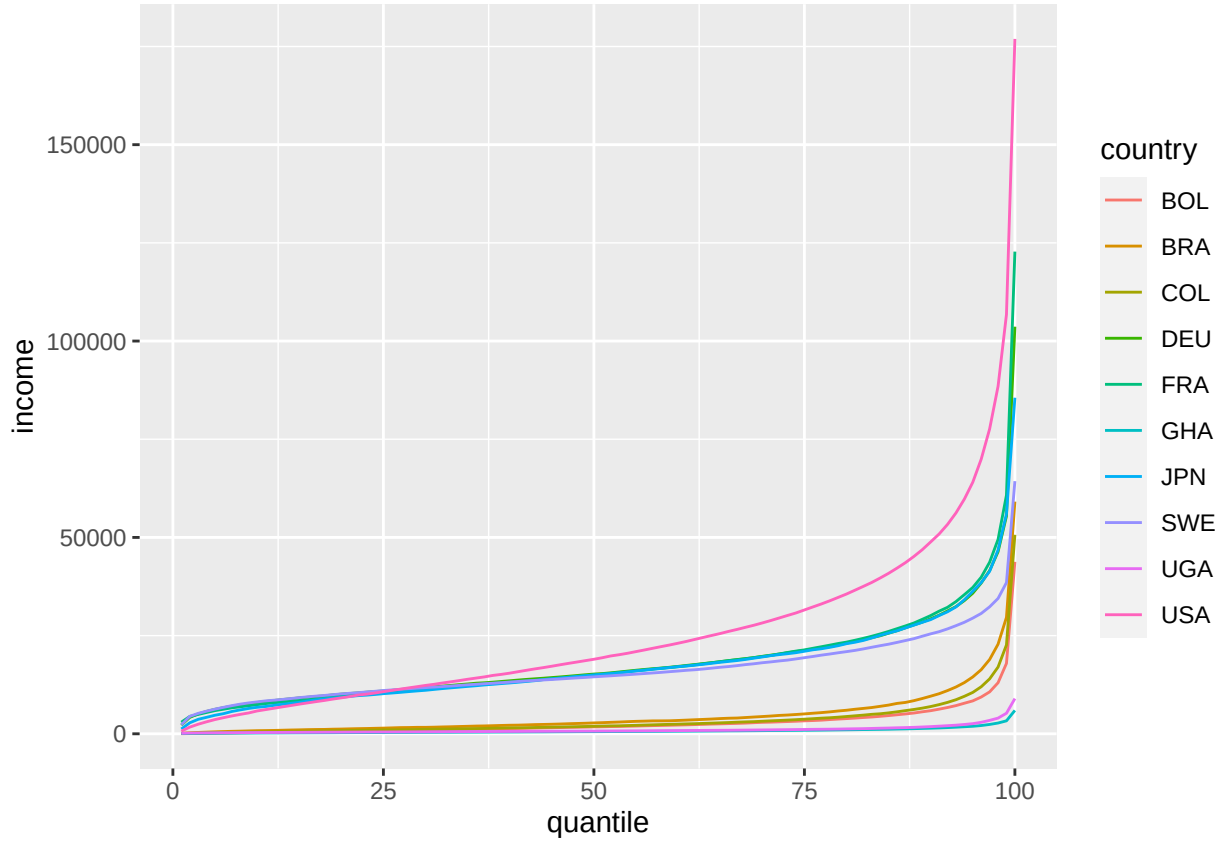
```
## Joining, by = "country_full"
```

```r
###Je test si il me manque des pays
filter(dfs, !(dfs$country_full %in% df$country_full))%>%select(country_full)%>%
  distinct()
```

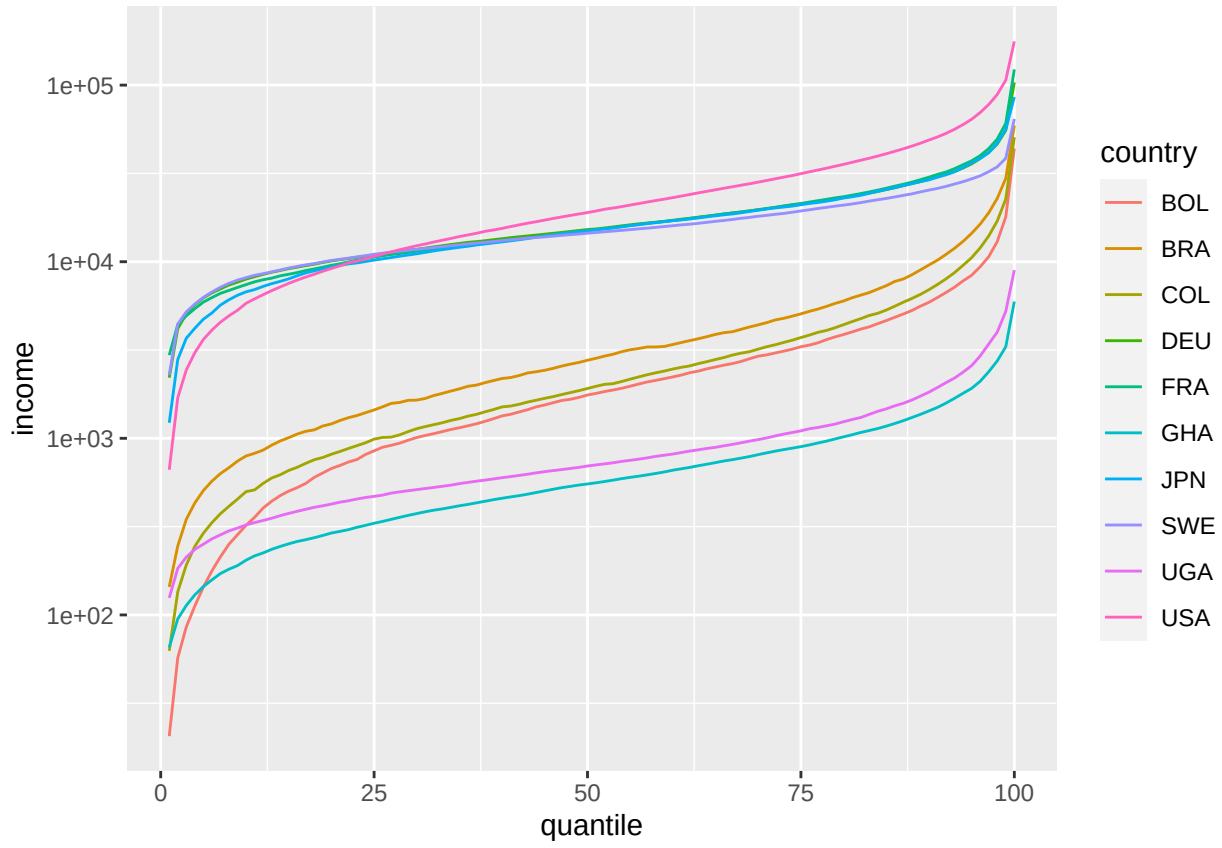
```
## [1] country_full
## <0 rows> (or 0-length row.names)
```

```r
###Je calcul le nombre de pays analysé dans l'étude
pays_analyse<-nrow(df)/100

###Je calcul la pop analysée dans l'étude
pop_analyse<-sum(df$pop)/100

###Je calcul le pourcentage de pop analysé dans l'étude
prct_pop<-pop_analyse*100/7041194.301
```

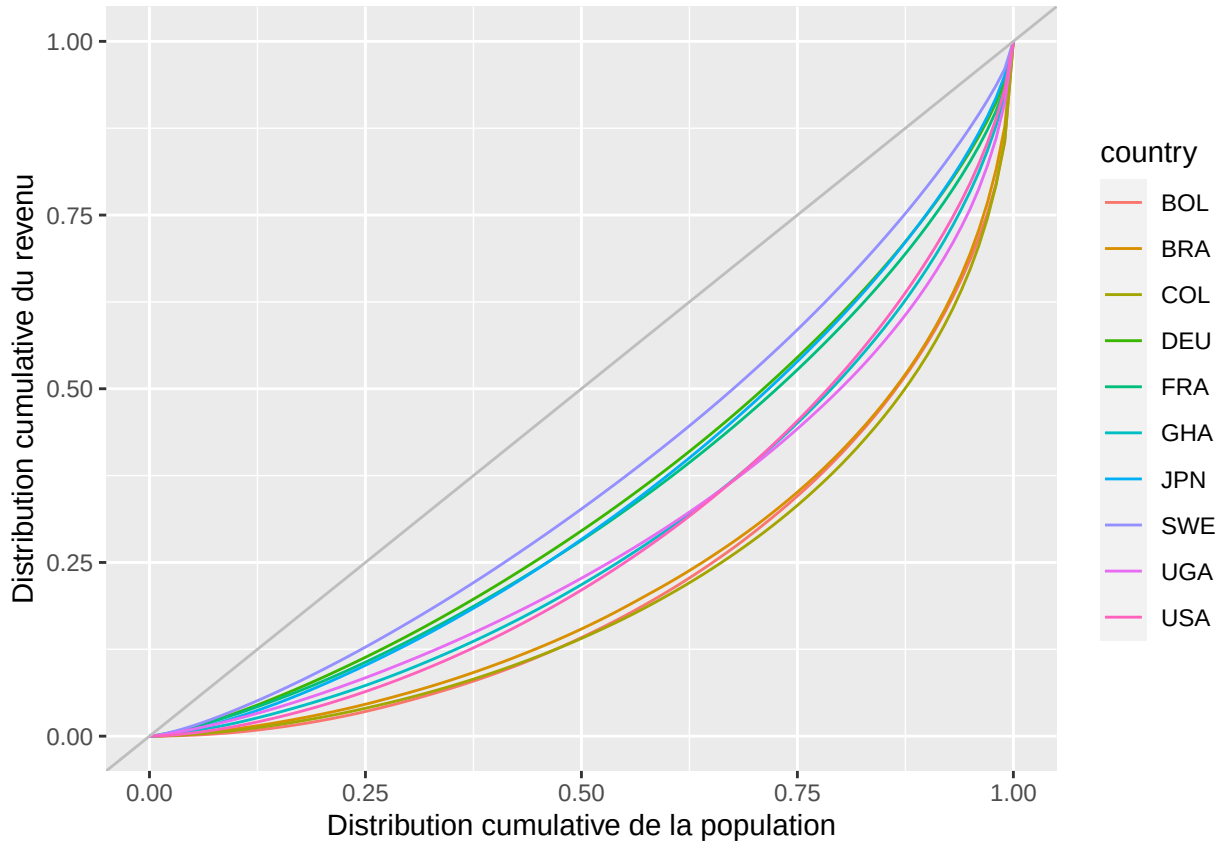
```r
###Mission2#############################################################

###Je cré un vecteur avec les pays qui vont m'intéresser plus
###particulièrement pour la suite
liste_pays<-c("FRA","SWE","BRA","JPN","DEU","USA","GHA","COL","UGA",
              "BOL")

###Je cré une table avec seulement ces pays là
dff<-df%>%filter(country %in%
  liste_pays)

###Je trace la courbe du revenu par rapport à la classe pour mes 10 pays
ggplot(dff,
       aes(x = quantile, y = income))+
  geom_line(aes(colour = country))
```




```r
###Je trace la même chose mais avec une échelle log10
ggplot(dff,
       aes(x = quantile, y = income))+
  geom_line(aes(colour = country)) +
  scale_y_continuous(trans='log10')
```




```
###Je trace les courbe de lorenz de ces pays
ggplot(dff, aes(income)) +
  stat_lorenz(aes(colour = country)) +
  geom_abline(color = "grey") +
  xlab("Distribution cumulative de la population") +
  ylab("Distribution cumulative du revenu")
```



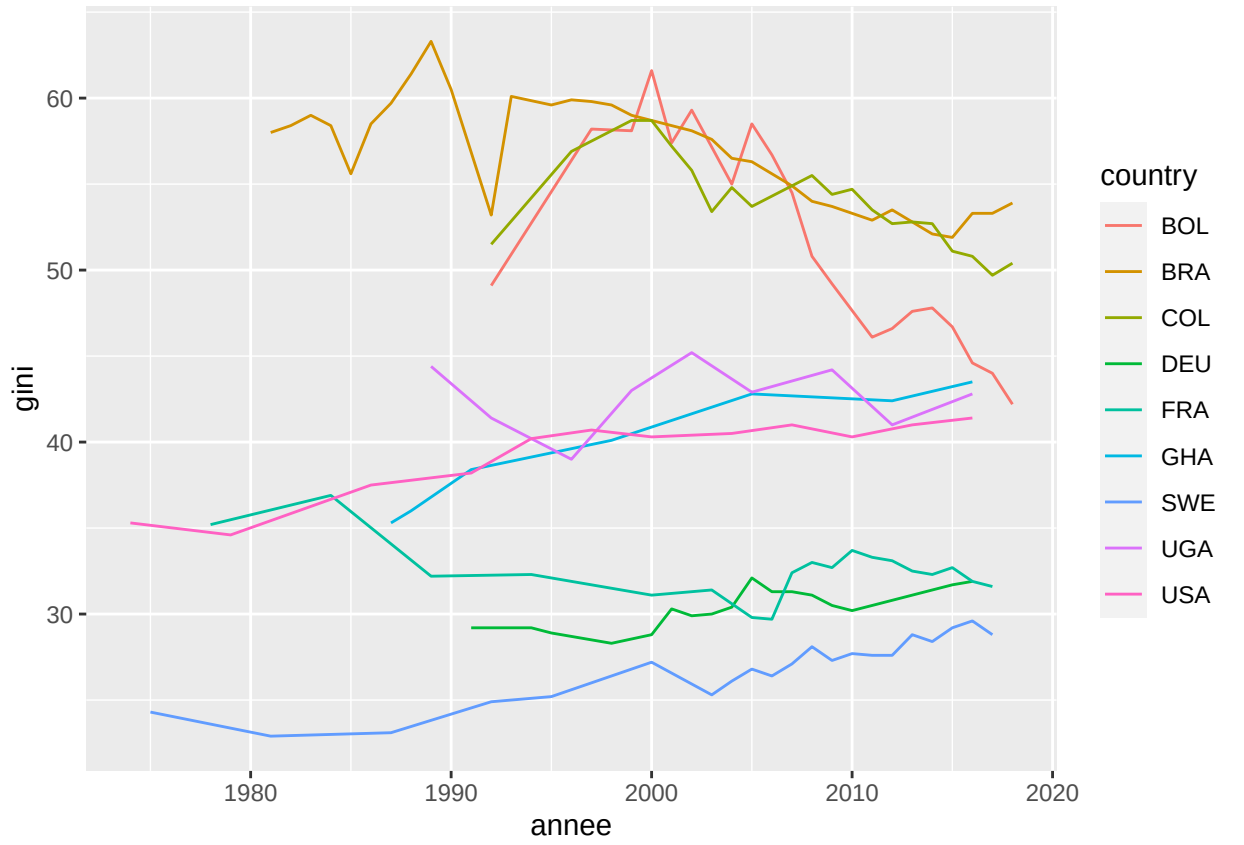

```
###Je crée une colonne gini dans ma table principale et je la
###remplis en calculant les indices de chaque pays
df$gini<-0

for (i in 0:pays_analyse-1){
  df$gini[(i*100+1):((i+1)*100)]<-
    round(gini(df[row.names(df) ==
                    (i*100+1):((i+1)*100),"income"])*100,1)
}

###Je crée une table rank qui classe les pays par indice de gini
rank<-df%>%group_by(country_full,gini)%>%count()%>%
  arrange(gini)%>%cbind(row.names(.))%>%
  rename(rank = "...4")
```

```
## New names:
## * NA -> ...4
```

```
rank$rank<-rank$rank%>%as.numeric()

###Les 5 pays avec l'indice de gini le plus faible
top5_gini<-rank[1:5,1:2]

###Les 5 pays avec l'indice de gini le plus élevé
worst5_gini<-rank[(nrow(rank)-4):nrow(rank),1:2]%>%arrange(desc(gini))
```

```r
###le rank de l'indice de gini de la france
fr_rank<-rank[rank$country_full == "France", c("country_full", "rank")]
```

```r
###Je charge une table contenant les indices de gini de nombreux
###pays sur plusieurs années
gini<-read.csv("gini_index.csv", encoding = "UTF-8", skip = 4)%>%
      select(-"X")

###Je ne garde que depuis 1970 pas assez de données pour les dates
###précédentes
gini<-select(gini, c(2,15:64))

###Je réorganise le tableau et je le mets en forme
gini<-melt(data = gini, id.vars = 1,
                measure.vars = 2:51)

gini$variable<-gsub("^.","",gini$variable)


colnames(gini)<-c("country","annee","gini")

gini$annee<-gini$annee%>%as.numeric()

###je supprime les NAs pour que mon graphe relie tous les points
###même ceux séparer par plusieurs années
gini<-gini%>%na.omit()

###Je trace le graph des indices de gini de mes 10 pays par rapport
###à l'année
ggplot(gini%>%filter(country %in%
  c("FRA","SWE","BRA","DEU","USA","GHA","COL","UGA","BOL")),
  aes(x = annee, y = gini ))+
  geom_line(aes(colour = country))
```




```
###Mission 3#############################################################

###Je classe mes pays par région
amerique_latine<-read.csv("amerique_latine.csv",
                encoding = "UTF-8")%>%
  select("country_full" = "Zone")

amerique_latine$region<-"amerique_latine"

europe<-read.csv("europe.csv",
                encoding = "UTF-8")%>%
  select("country_full" = "Zone")

europe$region<-"europe"

asie<-read.csv("asie.csv",
                encoding = "UTF-8")%>%
  select("country_full" = "Zone")

asie$region<-"asie"

afrique<-read.csv("afrique.csv",
                encoding = "UTF-8")%>%
  select("country_full" = "Zone")

afrique$region<-"afrique"
```

```r
fao_region<-rbind(afrique,asie,europe,amerique_latine)


fao_region%>%group_by(country_full)%>%count()
```

```
## # A tibble: 207 x 2
## # Groups:   country_full [207]
##    country_full       n
##    <fct>          <int>
##  1 Afrique du Sud     1
##  2 Algérie            1
##  3 Angola             1
##  4 Bénin              1
##  5 Botswana           1
##  6 Burkina Faso       1
##  7 Burundi            1
##  8 Cabo Verde         1
##  9 Cameroun           1
## 10 Comores            1
## # ... with 197 more rows
```

```r
fao_region$country_full<-fao_region$country_full%>%as.vector()

fao_region$country_full[fao_region$country_full ==
                      "États-Unis d'Amérique" ]<-"États-Unis"

fao_region$country_full[fao_region$country_full ==
               "Bolivie (État plurinational de)"]<-"Bolivie"

fao_region$country_full[fao_region$country_full == "République centrafricaine"]<-
  "République Centrafricaine"

fao_region$country_full[fao_region$country_full == "Chine, continentale"]<-"Chine"

fao_region$country_full[fao_region$country_full == "Chine, Taiwan Province de"]<-
  "Taïwan"

fao_region$country_full[fao_region$country_full == "République démocratique du Congo"]<-
  "République Démocratique du Congo"

fao_region$country_full[fao_region$country_full == "Tchéquie"]<-"République Tchèque"

fao_region$country_full[fao_region$country_full == "République dominicaine"]<-
  "République Dominicaine"

fao_region$country_full[fao_region$country_full == "Iran (République islamique d')"]<-
  "République Islamique d'Iran"

fao_region$country_full[fao_region$country_full == "République démocratique populaire lao"]<-
  "République Démocratique Populaire Lao"

fao_region$country_full[fao_region$country_full == "Soudan (ex)"]<-"Soudan"
```

```
fao_region$country_full[fao_region$country_full == "Eswatini"]<-"Swaziland"

fao_region$country_full[fao_region$country_full == "République arabe syrienne"]<-
  "République Arabe Syrienne"

fao_region$country_full[fao_region$country_full ==
  "Royaume-Uni de Grande-Bretagne et d'Irlande du Nord"]<-
  "Royaume-Uni"

fao_region$country_full[fao_region$country_full ==
               "Venezuela (République bolivarienne du)"]<-"Venezuela"


dfs<-inner_join(df,fao_region)
```

```
## Joining, by = "country_full"
```

```
sapply(df,function(x) sum(is.na(x)))
```

```
##      country country_full  year_survey     quantile nb_quantiles       income
##            0            0            0            0            0            0
##       gdpppp          pop         gini
##            0            0            0
```

```
filter(df, !(df$country %in% dfs$country))
```

```
## [1] country      country_full year_survey  quantile     nb_quantiles
## [6] income       gdpppp       pop          gini
## <0 rows> (or 0-length row.names)
```

```
df<-dfs

df$region[df$country %in% c("FIN","SWE","NOR","ISL","DNK")]<-
  "europe_nord"

df$region[df$country %in% c("USA","CAN")]<-"amerique_nord"
```

```
###J'attribut à chaque pays uun coef d'élasticité celon sa région
df$elasticite[df$region %in% c("afrique","amerique_latine")] <- 0.66
df$elasticite[df$region == "europe" | df$country == "USA"] <- 0.4
df$elasticite[df$region %in% c("asie")] <- 0.5
df$elasticite[df$region %in% c("europe_nord")| df$country == "CAN"] <- 0.2

sapply(df,function(x) sum(is.na(x)))
```

```
##      country country_full  year_survey     quantile nb_quantiles       income
##            0            0            0            0            0            0
##       gdpppp          pop         gini       region   elasticite
##            0            0            0            0            0
```

```r
###Je crée une fonction qui me permet de créer les tables de proba
###conditionnelles celon le coef, le nbre de quantile et le nombre
###d'individu par quantile
f_proba<-function(rho_j, nb_quantile, ind_quantile){

set.seed(100)

n<-nb_quantile*ind_quantile
ln_y_parent<-rnorm(n,0,1)
epsilon<-rnorm(n,0,1)
alpha<-0


y_child<-exp(alpha + rho_j*ln_y_parent + epsilon)
y_parent<-exp(ln_y_parent)


table<-data.frame(y_child,y_parent)
table$num<-row.names(table)

x<-arrange(table, y_child)
x$quantile<-trunc((row.names(x)%>%as.numeric()-1)/(n/nb_quantile))+1

y<-arrange(table, y_parent)
y$quantile<-trunc((row.names(x)%>%as.numeric()-1)/(n/nb_quantile))+1

table<-inner_join(x,y, by = c("num","y_child","y_parent"))%>%
  select("c_i_child" = "quantile.x", "c_i_parent" = "quantile.y")

table<-table%>%group_by(c_i_child,c_i_parent)%>%count()

x<-data.frame(c_i_child = factor(1:nb_quantile),
              c_i_parent = factor(1:nb_quantile))

y<-expand(x, c_i_child, c_i_parent)

table$c_i_child<- table$c_i_child%>%as.factor()
table$c_i_parent<- table$c_i_parent%>%as.factor()

table<-full_join(table,y)

table$n[is.na(table$n)]<-0
table$proba_cond<-table$n/(n/nb_quantile)

table
}

a<-f_proba(0.9, 100, 1000)
```

```
## Joining, by = c("c_i_child", "c_i_parent")
```

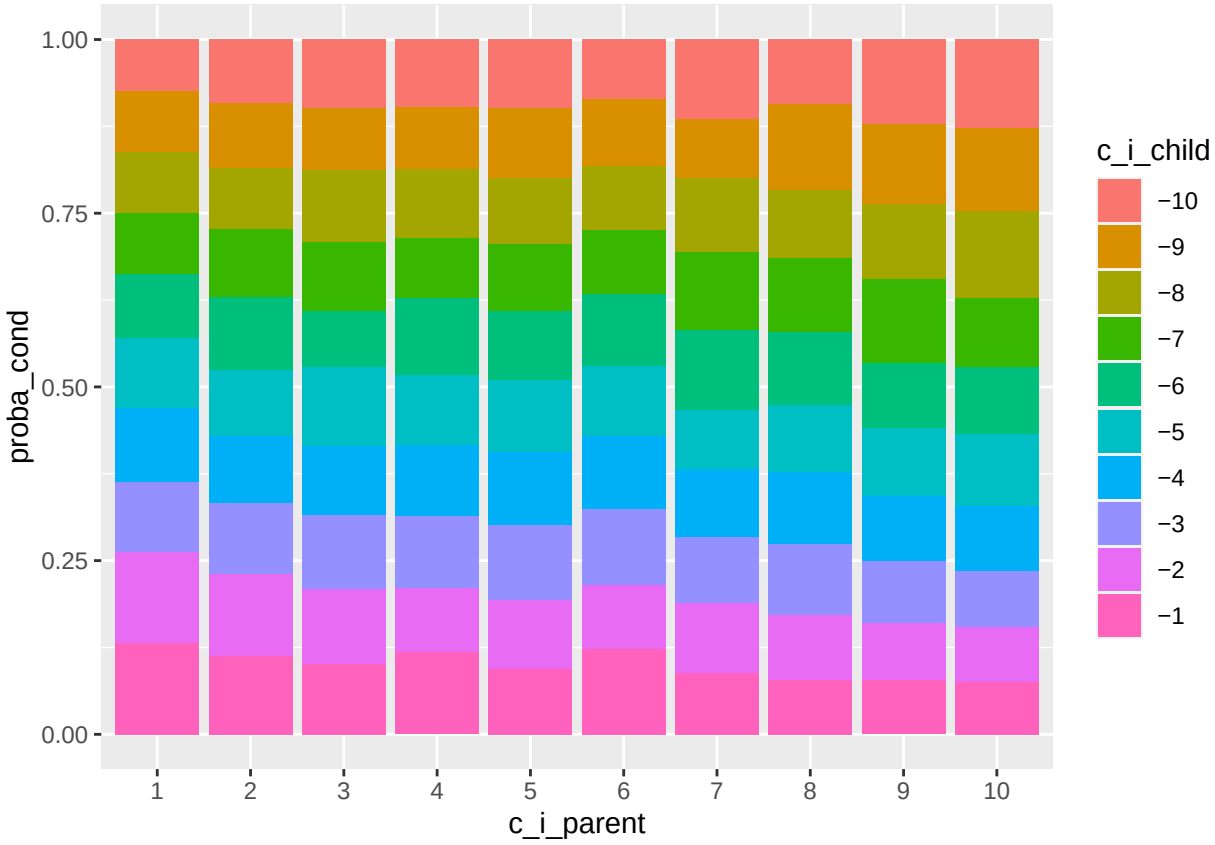
```r
###Je crée deux tables et je dessine les histogrammes associés
table_proba_cond0.1<-f_proba(0.1, 10, 1000)
```

```
## Joining, by = c("c_i_child", "c_i_parent")
```

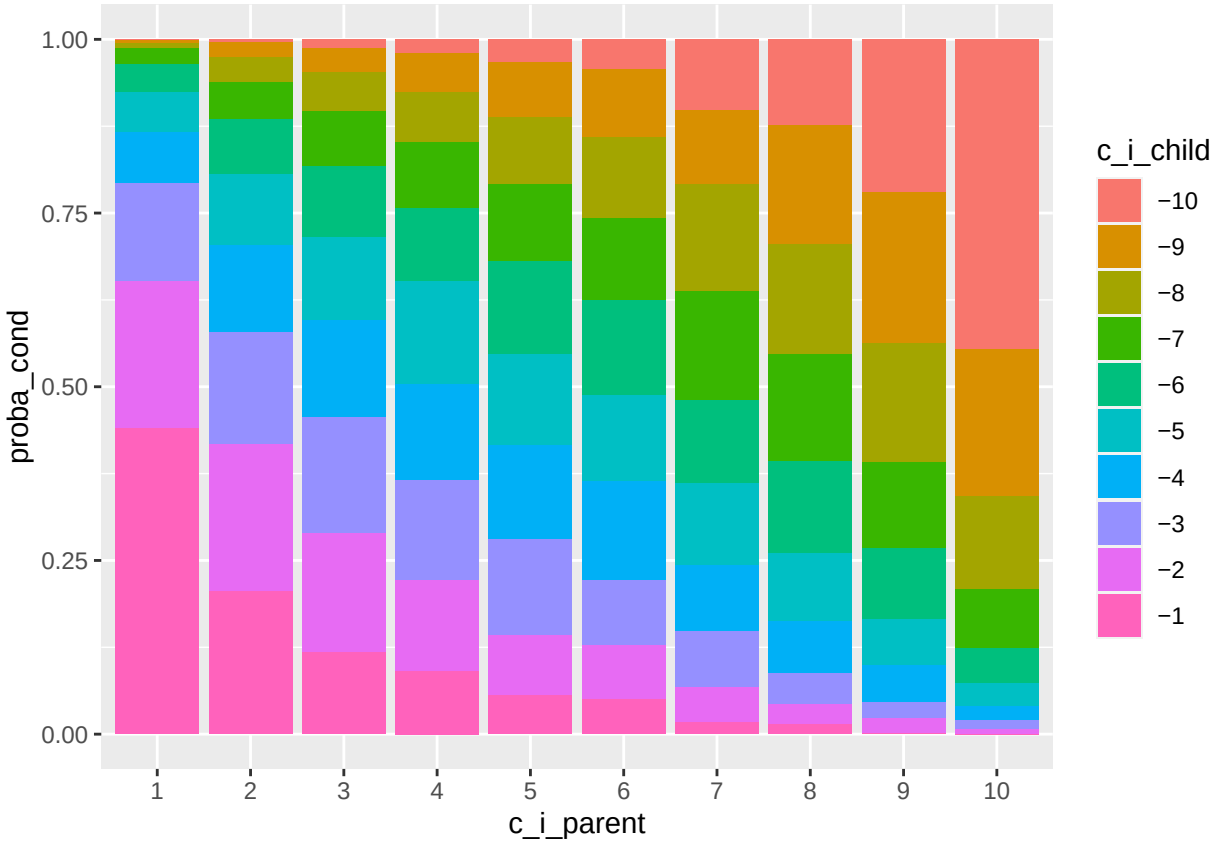
```
table_proba_cond0.9<-f_proba(0.9, 10, 1000)
```

```
## Joining, by = c("c_i_child", "c_i_parent")
```

```
ggplot(table_proba_cond0.1,
       aes(c_i_parent, proba_cond, fill = desc(c_i_child)%>%as.factor()))+
  geom_bar(stat = "identity") + labs( fill = "c_i_child")
```




```
ggplot(table_proba_cond0.9,
       aes(c_i_parent, proba_cond, fill = desc(c_i_child)%>%as.factor()))+
  geom_bar(stat = "identity") + labs( fill = "c_i_child")
```




```
###Mission 4############################################################

###Je crée une fonction qui me permet d'ajouter la colonne c_i_parent
###dans ma table principale.
###La fonction crée la classe de revenu de 50000 parents 500 pour chaque
###quantile celon un coef d'élasticité donné
f<-function(rho_j){


nb_quantile<-100
n<-nb_quantile*500
ln_y_parent<-rnorm(n,0,1)
epsilon<-rnorm(n,0,1)
alpha<-0


y_child<-exp(alpha + rho_j*ln_y_parent + epsilon)
y_parent<-exp(ln_y_parent)


table<-data.frame(y_child,y_parent)
table$num<-row.names(table)

x<-arrange(table, y_child)
x$quantile<-trunc((row.names(x)%>%as.numeric()-1)/(n/nb_quantile))+1
```

```
y<-arrange(table, y_parent)
y$quantile<-trunc((row.names(x)%>%as.numeric()-1)/(n/nb_quantile))+1

table<-inner_join(x,y, by = c("num","y_child","y_parent"))%>%
  select("c_i_child" = "quantile.x", "c_i_parent" = "quantile.y")

table$c_i_parent
}
```

```
###Je crée une fonction pour pouvoir répéter la fonction précédente
###de façon aléatoire un nombre de fois donné pour un coef donné
rep_f<-function(x, n){
b<-c()
for (i in 1:n) {

  b<-c(b,f(x))
}
b
}
```

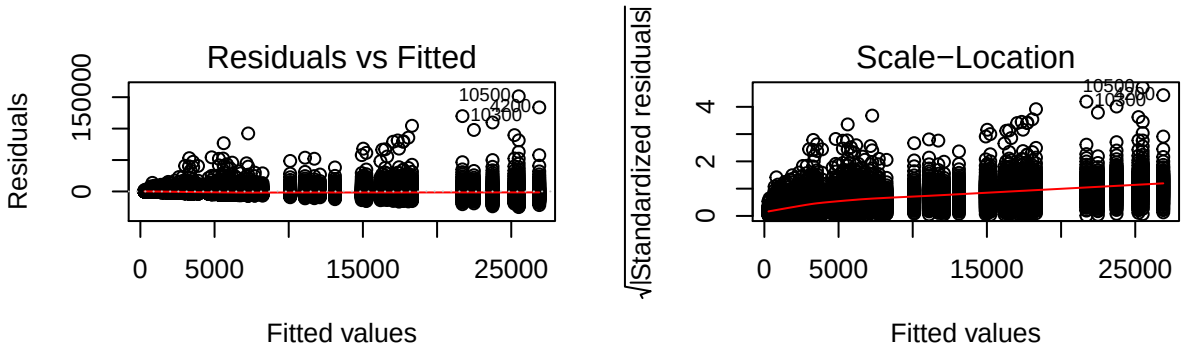
```
###Je clone 500 fois chaque individu
df500<-df[rep(1:nrow(df), each = 500),]%>%rename("c_i_child" ="quantile")

###Je compte combien de pays sont associés à chaque coef d'élasticité
n0.2<-df%>%filter(elasticite == 0.2)%>%nrow()/100
n0.4<-df%>%filter(elasticite == 0.4)%>%nrow()/100
n0.5<-df%>%filter(elasticite == 0.5)%>%nrow()/100
n0.66<-df%>%filter(elasticite == 0.66)%>%nrow()/100

###Je crée ma colonne c_i_parent pour mes 5550000 enfants en répétant
###la fonction f autant de fois qu'il y a de pays par coef
a1<-rep_f(0.2,n0.2)
a2<-rep_f(0.4,n0.4)
a3<-rep_f(0.5,n0.5)
a4<-rep_f(0.66,n0.66)

###Je crée la colonne en ordonnant du plus petit coef au plus grand
c_i_parent<-c(a1,a2,a3,a4)

###Je colle ma colonne en ordonnant ma table principale de la même façon
###pour que ça corresponde
df500<-cbind(df500%>%arrange(elasticite),c_i_parent)
```

```
###Je crée une fonction qui fait les tests et graphs de regression linéaire
f_graph<- function(fit, p){

df_function<-fit$model
alpha <- 0.05
n <- dim(df_function)[1]
analyses <- data.frame(obs= 1:n)
analyses$levier <- hat(model.matrix(fit))
seuil_levier <- 2*p/n
```

```
analyses$rstudent <- rstudent(fit)
seuil_rstudent <- qt(1-alpha/2,n-p-1)

influence <- influence.measures(fit)
names(influence)
colnames(influence$infmat)

analyses$dcook <- influence$infmat[,"cook.d"]
seuil_dcook <- 4/(n-p)

layout(matrix(1:4, 2, 2))


return(
  list(
   plot(fit),

    ggplot(data.frame(fit$residuals),aes(fit.residuals)) +
 geom_histogram(aes(y=..count..), colour="blue", fill= "white")+
  geom_vline(aes(xintercept=mean(fit.residuals), color="red"))+
  xlab("Residuals values")+ ylab("Number of residuals"),

ggplot(data=analyses,aes(x=obs, y=levier))+
   geom_bar(stat="identity",colour="steelblue")+
   geom_hline(yintercept=seuil_levier,col="red")+
   theme_minimal()+
   xlab("Observation")+
   ylab("Leviers")+
   scale_x_continuous(breaks=seq(0,n,by=5)),

ggplot(data=analyses,aes(x=obs,y=rstudent))+
  geom_bar(stat="identity",colour="steelblue")+
  geom_hline(yintercept=-seuil_rstudent,col="red")+
  geom_hline(yintercept=seuil_rstudent,col="red")+
  theme_minimal()+
  xlab("observation")+
  ylab("Résidus studentisés")+
  scale_x_continuous(breaks=seq(0,n,by=5)),

summary(fit),

vif(fit),

shapiro.test(sample(fit$residuals,5000)),

ad.test(fit$residuals),

ks.test(fit$residuals, "pnorm")
)
)
}
```

```r
###Je fais des regressions linéaires pour expliquer au mieux
###la variable income.

###Anova income ~ country
fit_aov<-lm(income ~ country, df)

###Graphe analyse résidus
layout(matrix(1:4, 2, 2))
plot(fit_aov)
```

```
## hat values (leverages) are all = 0.01
##  and there are no factor predictors; no plot no. 5
```




```r
###test normalité résidu
shapiro.test(sample(fit_aov$residuals,5000))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sample(fit_aov$residuals, 5000)
## W = 0.52619, p-value < 2.2e-16
```

```r
ad.test(fit_aov$residuals)
```

```
##
##  Anderson-Darling normality test
##
## data:  fit_aov$residuals
## A = 1140.4, p-value < 2.2e-16
```

```r
ks.test(fit_aov$residuals, "pnorm")
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  fit_aov$residuals
## D = 0.65499, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```r
###test homogénéité résidu
fligner.test(residuals(fit_aov)~df$country)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  residuals(fit_aov) by df$country
## Fligner-Killeen:med chi-squared = 5271.7, df = 110, p-value < 2.2e-16
```

```r
bartlett.test(residuals(fit_aov)~df$country)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  residuals(fit_aov) by df$country
## Bartlett's K-squared = 18474, df = 110, p-value < 2.2e-16
```

```r
###Analyse
summary(fit_aov)
```

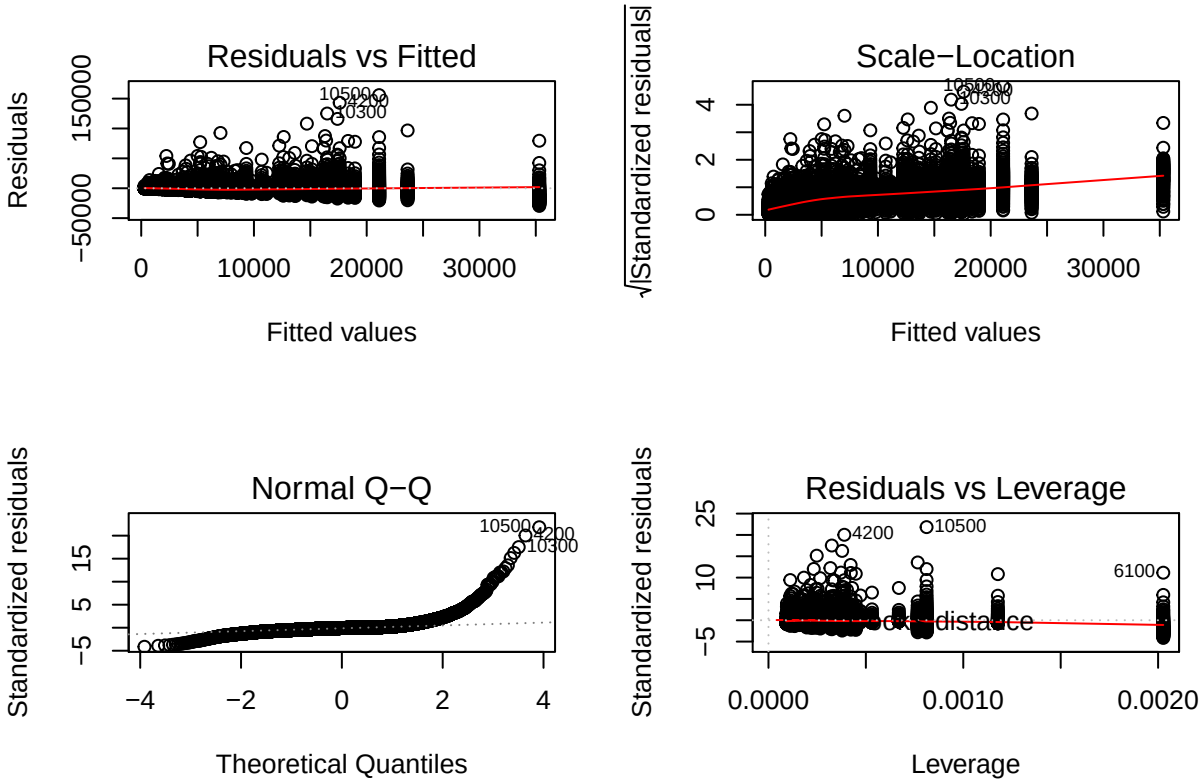
```
##
## Call:
## lm(formula = income ~ country, data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -24840  -1907   -393    493 151425
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2994.83     685.65   4.368 1.27e-05 ***
## countryARG   2853.05     969.66   2.942 0.003264 **
## countryARM  -1366.45     969.66  -1.409 0.158804
```

```
## countryAUT  13642.77     969.66  14.070  < 2e-16 ***
## countryAZE   -637.40     969.66  -0.657 0.510973
## countryBEL  12029.78     969.66  12.406  < 2e-16 ***
## countryBFA  -2048.49     969.66  -2.113 0.034659 *
## countryBGD  -1996.14     969.66  -2.059 0.039557 *
## countryBGR   1990.15     969.66   2.052 0.040153 *
## countryBIH   3339.86     969.66   3.444 0.000575 ***
## countryBLR    926.33     969.66   0.955 0.339438
## countryBOL     21.43     969.66   0.022 0.982365
## countryBRA   1812.65     969.66   1.869 0.061598 .
## countryBTN  -1478.90     969.66  -1.525 0.127244
## countryCAF  -2183.53     969.66  -2.252 0.024351 *
## countryCAN  20744.81     969.66  21.394  < 2e-16 ***
## countryCHL   4274.87     969.66   4.409 1.05e-05 ***
## countryCHN   -472.07     969.66  -0.487 0.626380
## countryCIV  -2594.99     969.66  -2.676 0.007457 **
## countryCMR  -1200.34     969.66  -1.238 0.215782
## countryCOD  -2718.81     969.66  -2.804 0.005058 **
## countryCOL    552.18     969.66   0.569 0.569061
## countryCRI   2585.56     969.66   2.666 0.007677 **
## countryCYP  14350.56     969.66  14.800  < 2e-16 ***
## countryCZE   5240.46     969.66   5.404 6.64e-08 ***
## countryDEU  15066.89     969.66  15.538  < 2e-16 ***
## countryDNK  14048.32     969.66  14.488  < 2e-16 ***
## countryDOM    563.57     969.66   0.581 0.561114
## countryECU    388.91     969.66   0.401 0.688369
## countryEGY   -964.34     969.66  -0.995 0.319996
## countryESP  10122.16     969.66  10.439  < 2e-16 ***
## countryEST   4707.23     969.66   4.855 1.22e-06 ***
## countryFIN  13311.50     969.66  13.728  < 2e-16 ***
## countryFRA  15314.58     969.66  15.794  < 2e-16 ***
## countryGBR  18714.77     969.66  19.300  < 2e-16 ***
## countryGEO  -1631.07     969.66  -1.682 0.092577 .
## countryGHA  -2231.71     969.66  -2.302 0.021380 *
## countryGIN  -2298.82     969.66  -2.371 0.017769 *
## countryGRC   8732.44     969.66   9.006  < 2e-16 ***
## countryHND    301.44     969.66   0.311 0.755905
## countryHRV   4721.64     969.66   4.869 1.14e-06 ***
## countryHUN   3106.51     969.66   3.204 0.001361 **
## countryIDN  -1618.93     969.66  -1.670 0.095029 .
## countryIND  -2070.56     969.66  -2.135 0.032755 *
## countryIRL  14715.91     969.66  15.176  < 2e-16 ***
## countryIRN   2837.83     969.66   2.927 0.003434 **
## countryIRQ  -1289.32     969.66  -1.330 0.183658
## countryISL  23893.68     969.66  24.641  < 2e-16 ***
## countryISR   8105.49     969.66   8.359  < 2e-16 ***
## countryITA  11930.39     969.66  12.304  < 2e-16 ***
## countryJOR     53.80     969.66   0.055 0.955754
## countryJPN  14438.13     969.66  14.890  < 2e-16 ***
## countryKAZ   -755.68     969.66  -0.779 0.435805
## countryKEN  -2475.51     969.66  -2.553 0.010694 *
## countryKGZ  -1221.61     969.66  -1.260 0.207756
## countryKHM  -1508.49     969.66  -1.556 0.119811
## countryKOR  12232.74     969.66  12.615  < 2e-16 ***
```

```
## countryLAO  -1991.42     969.66  -2.054 0.040025 *
## countryLBR  -2379.82     969.66  -2.454 0.014132 *
## countryLKA  -1116.89     969.66  -1.152 0.249413
## countryLTU   3628.83     969.66   3.742 0.000183 ***
## countryLUX  22222.73     969.66  22.918  < 2e-16 ***
## countryLVA   3769.64     969.66   3.888 0.000102 ***
## countryMAR   -657.23     969.66  -0.678 0.497917
## countryMDA   -845.66     969.66  -0.872 0.383163
## countryMDG  -2649.59     969.66  -2.732 0.006296 **
## countryMEX    891.00     969.66   0.919 0.358179
## countryMKD   1604.57     969.66   1.655 0.097999 .
## countryMLI  -2313.75     969.66  -2.386 0.017043 *
## countryMNE   3553.07     969.66   3.664 0.000249 ***
## countryMNG   -656.74     969.66  -0.677 0.498235
## countryMOZ  -2302.35     969.66  -2.374 0.017595 *
## countryMRT  -1196.22     969.66  -1.234 0.217360
## countryMWI  -2080.56     969.66  -2.146 0.031922 *
## countryMYS   3197.28     969.66   3.297 0.000979 ***
## countryNER  -2344.70     969.66  -2.418 0.015619 *
## countryNGA  -2302.97     969.66  -2.375 0.017564 *
## countryNIC   -397.58     969.66  -0.410 0.681800
## countryNLD  14733.81     969.66  15.195  < 2e-16 ***
## countryNOR  19488.55     969.66  20.098  < 2e-16 ***
## countryNPL  -2080.44     969.66  -2.146 0.031932 *
## countryPAK  -2106.99     969.66  -2.173 0.029808 *
## countryPAN   2299.13     969.66   2.371 0.017754 *
## countryPER    335.70     969.66   0.346 0.729194
## countryPHL  -1467.70     969.66  -1.514 0.130150
## countryPOL   2746.89     969.66   2.833 0.004622 **
## countryPRT   7103.85     969.66   7.326 2.54e-13 ***
## countryPRY    283.25     969.66   0.292 0.770205
## countryROU    323.51     969.66   0.334 0.738665
## countryRUS   4161.94     969.66   4.292 1.78e-05 ***
## countrySDN  -1875.92     969.66  -1.935 0.053064 .
## countrySLV   -139.61     969.66  -0.144 0.885524
## countrySRB   1884.92     969.66   1.944 0.051933 .
## countrySVK   3101.75     969.66   3.199 0.001384 **
## countrySVN   9111.18     969.66   9.396  < 2e-16 ***
## countrySWE  13189.39     969.66  13.602  < 2e-16 ***
## countrySWZ  -2448.15     969.66  -2.525 0.011592 *
## countryTHA   -152.34     969.66  -0.157 0.875161
## countryTJK   -860.96     969.66  -0.888 0.374614
## countryTLS  -2267.22     969.66  -2.338 0.019397 *
## countryTUR   3055.64     969.66   3.151 0.001630 **
## countryTWN  13505.39     969.66  13.928  < 2e-16 ***
## countryTZA  -2406.06     969.66  -2.481 0.013104 *
## countryUGA  -1977.09     969.66  -2.039 0.041479 *
## countryUKR    354.56     969.66   0.366 0.714632
## countryURY   2295.96     969.66   2.368 0.017912 *
## countryUSA  22508.75     969.66  23.213  < 2e-16 ***
## countryVEN    286.34     969.66   0.295 0.767775
## countryVNM  -1567.46     969.66  -1.617 0.106015
## countryYEM  -1952.19     969.66  -2.013 0.044110 *
## countryZAF   2623.07     969.66   2.705 0.006838 **
```

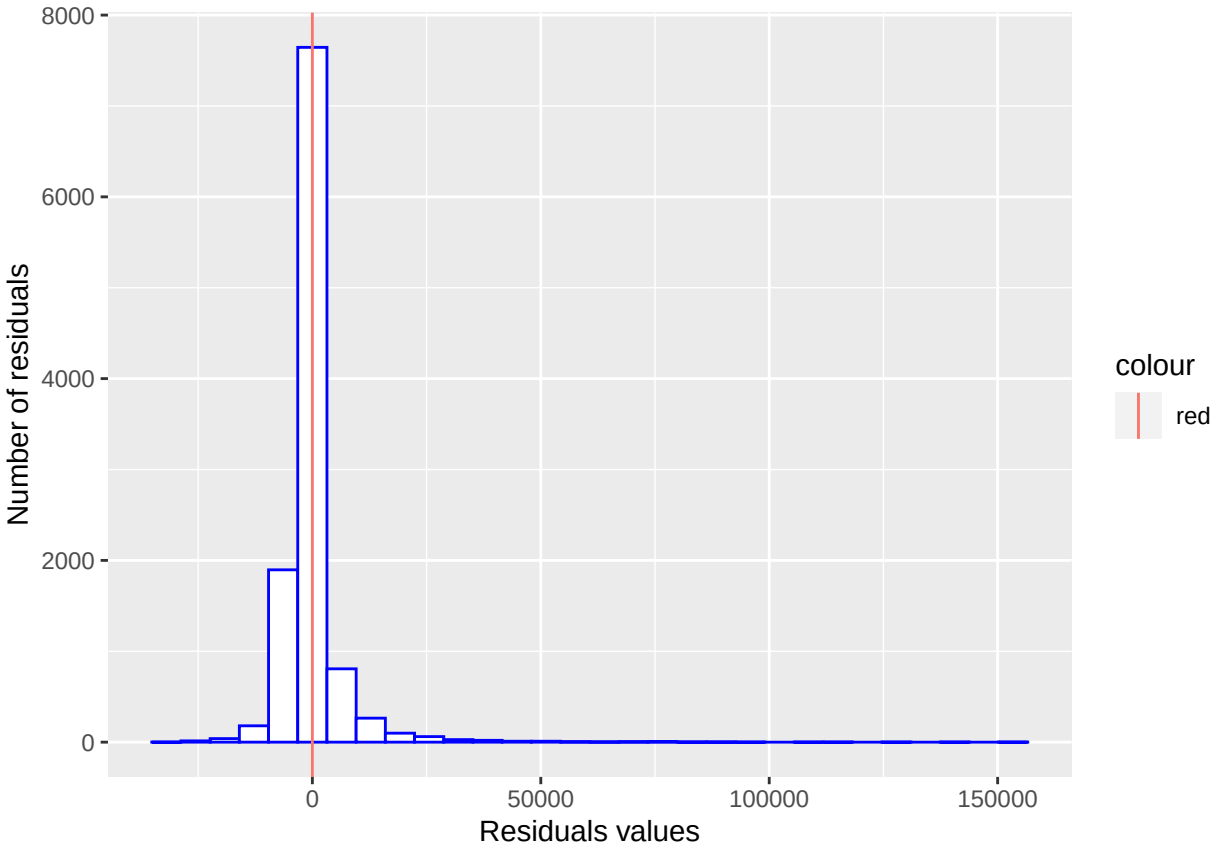
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6857 on 10989 degrees of freedom
## Multiple R-squared:  0.4914, Adjusted R-squared:  0.4863
## F-statistic: 96.54 on 110 and 10989 DF,  p-value: < 2.2e-16
```

```
###regression linéaire
fit<-lm(income~ gdpppp + gini, df)

###les test et graphiques
f_graph(fit, 2)
```

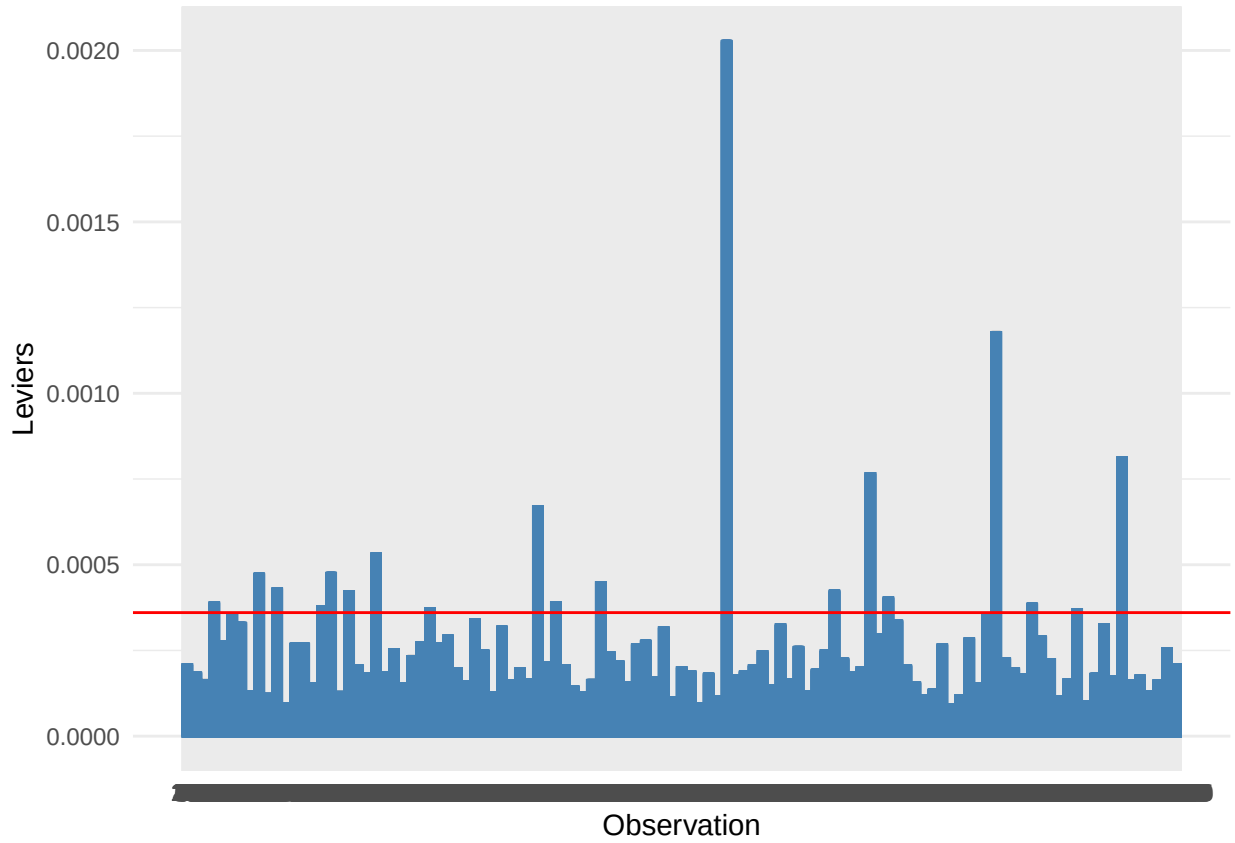



```
## [[1]]
## NULL
##
## [[2]]
```

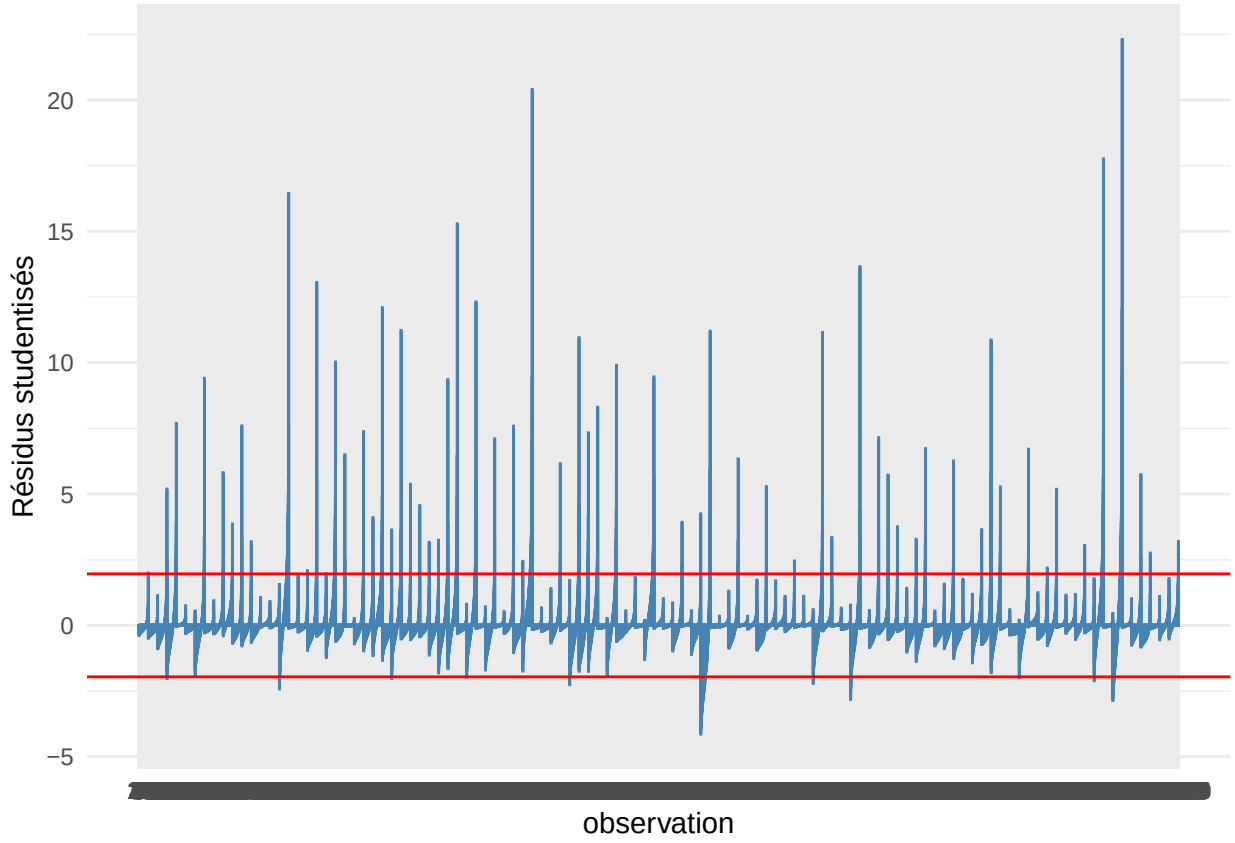
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


8000
6000
4000
2000
0
Number of residuals
0
50000
100000
150000
Residuals values
colour
red

```
##
## [[3]]
```



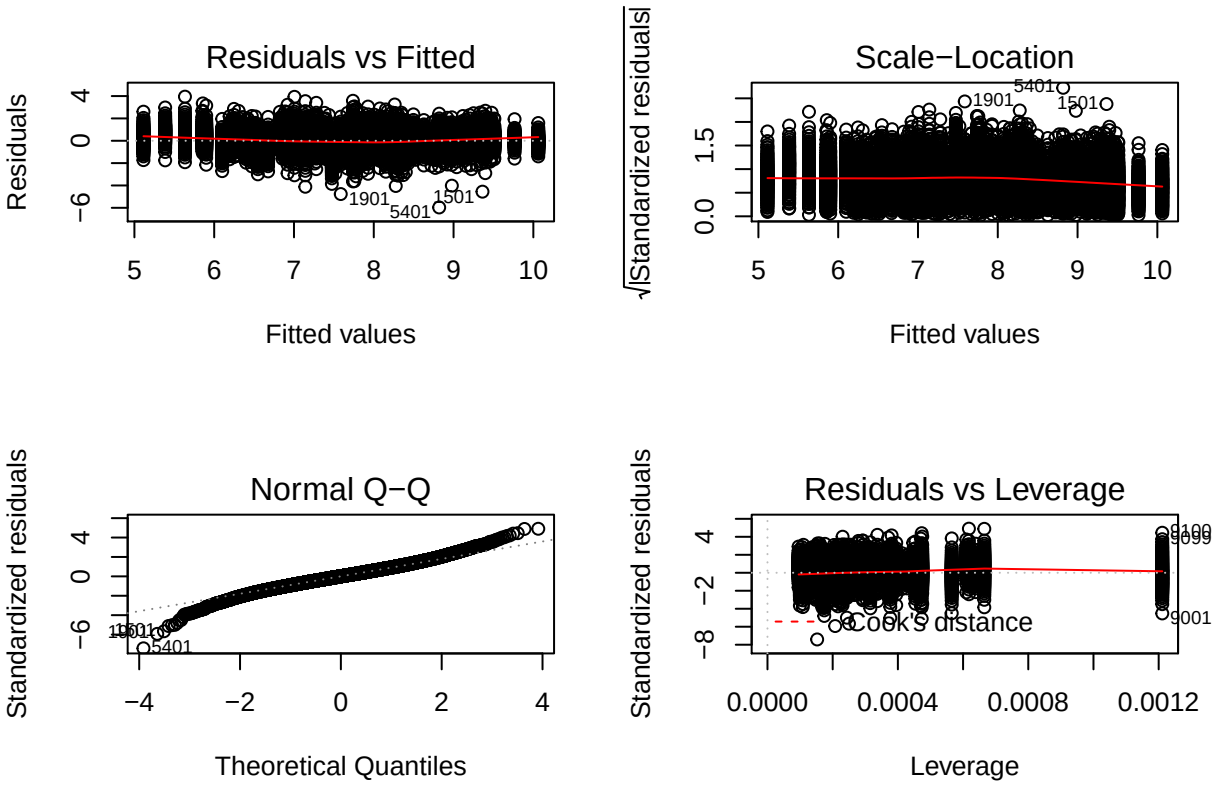

```
##
## [[4]]
```



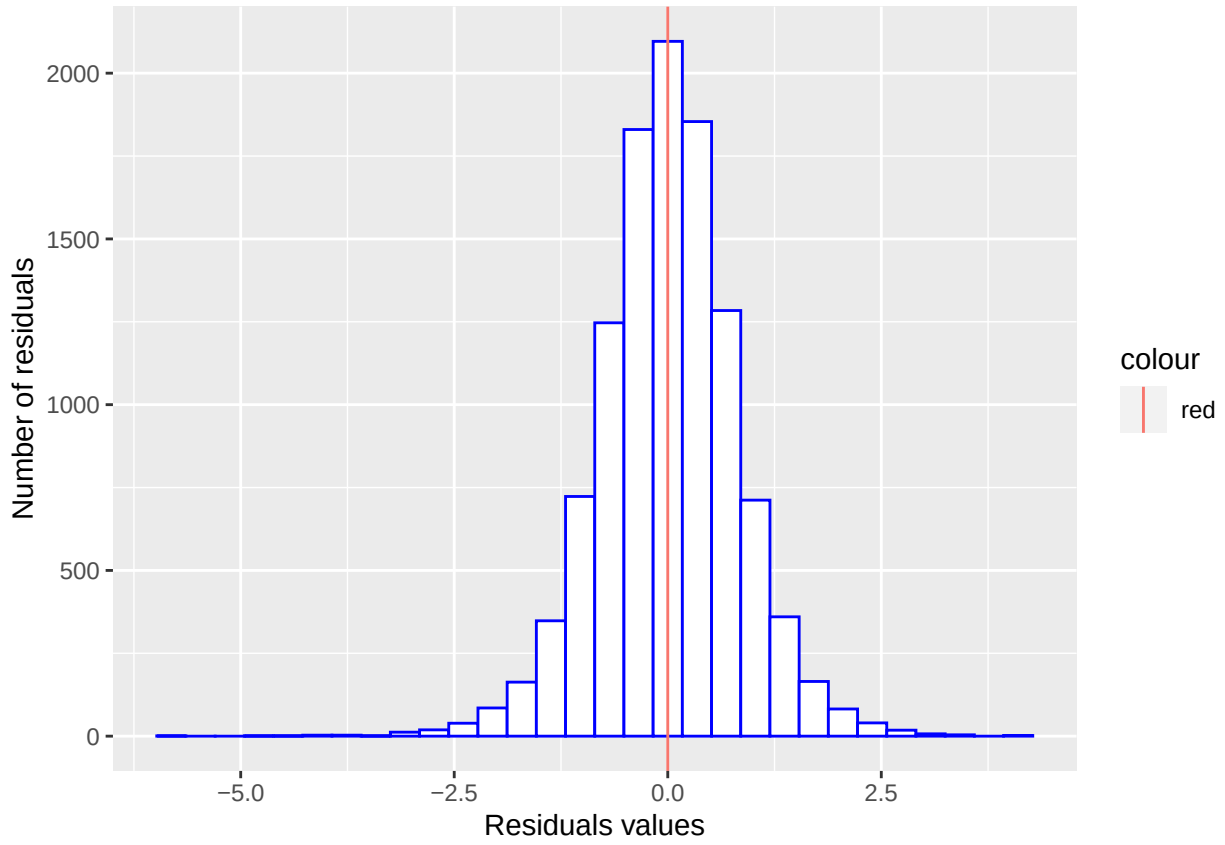

```
##
## [[5]]
##
## Call:
## lm(formula = income ~ gdpppp + gini, data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -29528  -2410   -474    573 155824
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.279e+02  3.539e+02  -1.492   0.1359
## gdpppp       4.834e-01  5.561e-03  86.929   <2e-16 ***
## gini         1.669e+01  8.282e+00   2.015   0.0439 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7142 on 11097 degrees of freedom
## Multiple R-squared:  0.4428, Adjusted R-squared:  0.4427
## F-statistic:  4410 on 2 and 11097 DF,  p-value: < 2.2e-16
##
##
## [[6]]
##   gdpppp     gini
## 1.188466 1.188466
```

```
##
## [[7]]
##
##  Shapiro-Wilk normality test
##
## data:  sample(fit$residuals, 5000)
## W = 0.5402, p-value < 2.2e-16
##
##
## [[8]]
##
##  Anderson-Darling normality test
##
## data:  fit$residuals
## A = 1065.7, p-value < 2.2e-16
##
##
## [[9]]
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  fit$residuals
## D = 0.6412, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

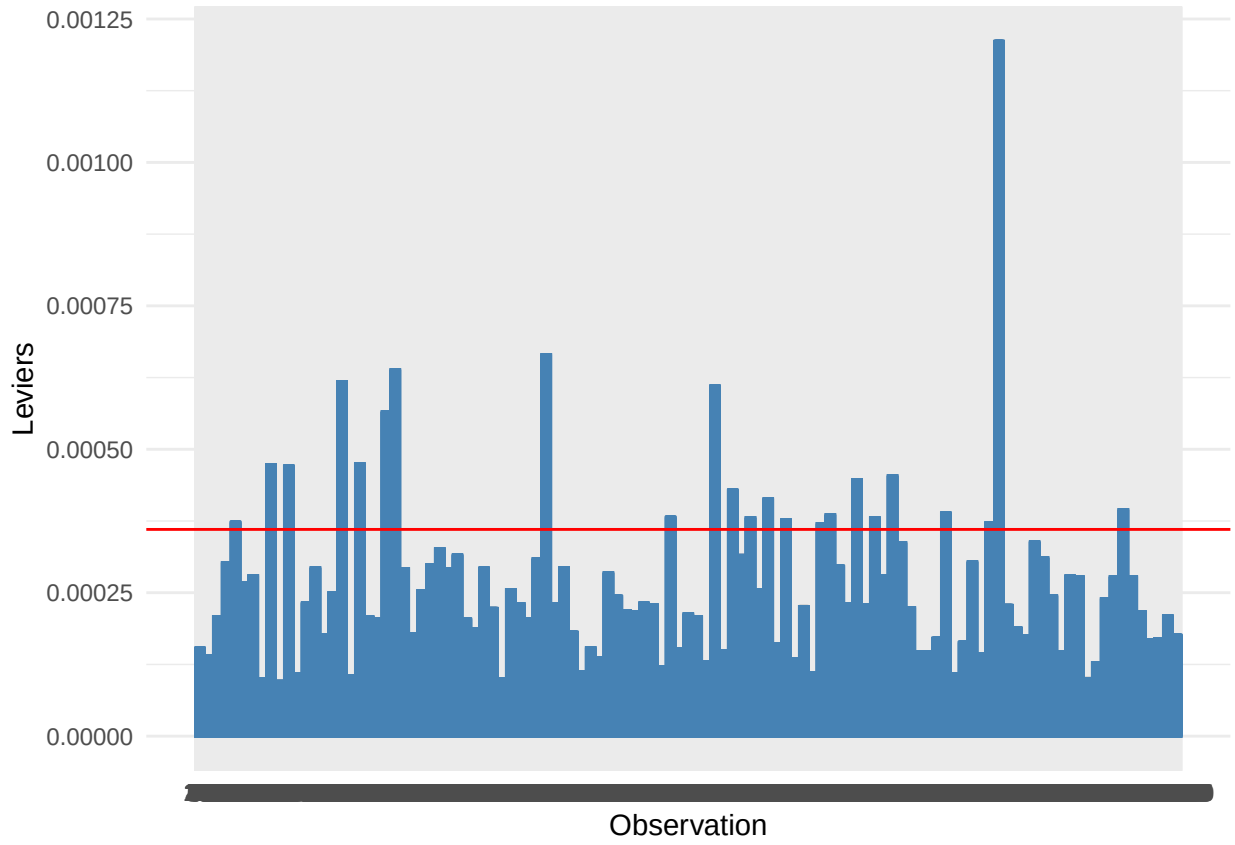
```r
###regression linéaire
fit_ln<-lm(log(income)~ log(gdpppp) + gini, df)

###les test et graphiques
f_graph(fit_ln, 2)
```



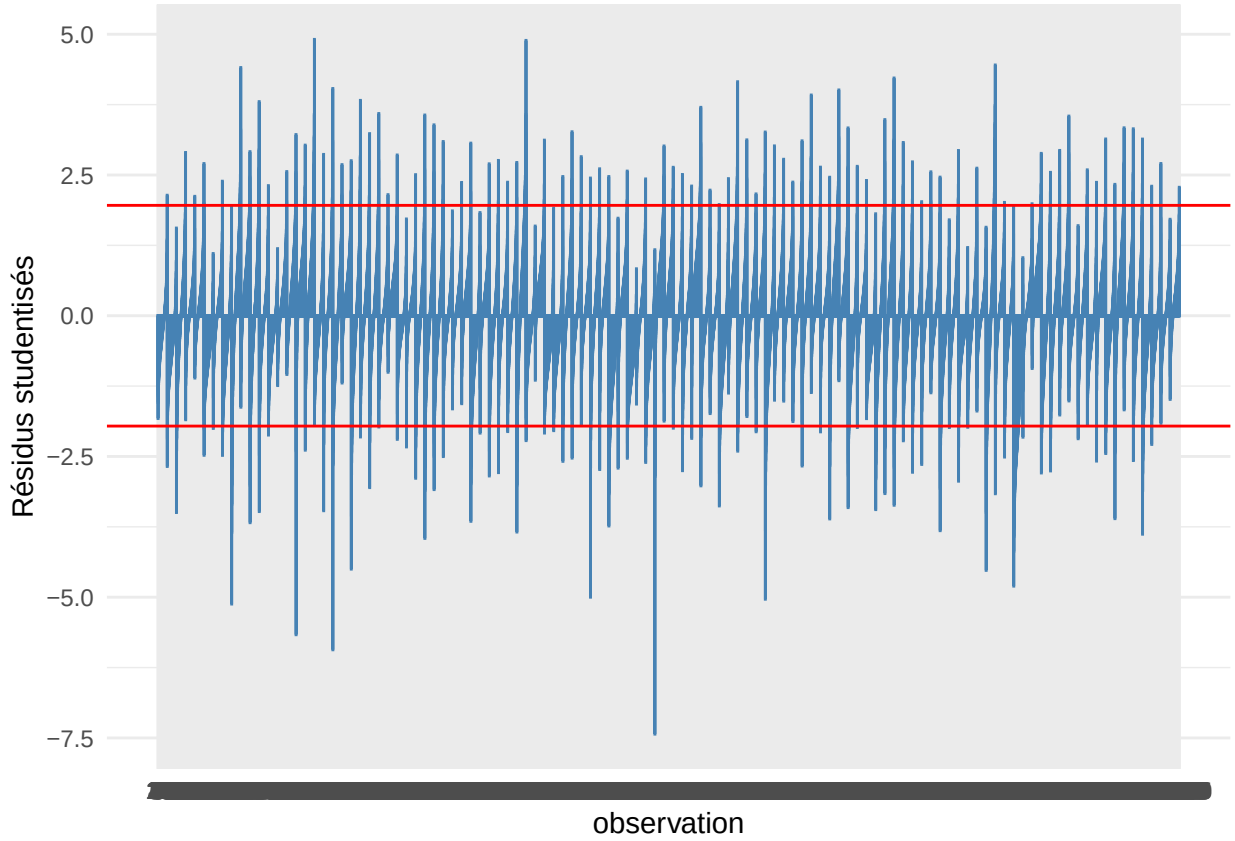

```
## [[1]]
## NULL
##
## [[2]]

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


2000
1500
1000
500
0
Number of residuals
−5.0
−2.5
0.0
2.5
Residuals values
colour
red

```
##
## [[3]]
```


0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
Leviers
Observation

```
##
## [[4]]
```




```
##
## [[5]]
##
## Call:
## lm(formula = log(income) ~ log(gdpppp) + gini, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9681 -0.4841  0.0046  0.4907  3.9463
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.8785807  0.0740290   11.87   <2e-16 ***
## log(gdpppp)  0.8599718  0.0063631  135.15   <2e-16 ***
## gini        -0.0153025  0.0008966  -17.07   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8049 on 11097 degrees of freedom
## Multiple R-squared:  0.6632, Adjusted R-squared:  0.6631
## F-statistic: 1.092e+04 on 2 and 11097 DF,  p-value: < 2.2e-16
##
##
## [[6]]
## log(gdpppp)        gini
##    1.096456    1.096456
```

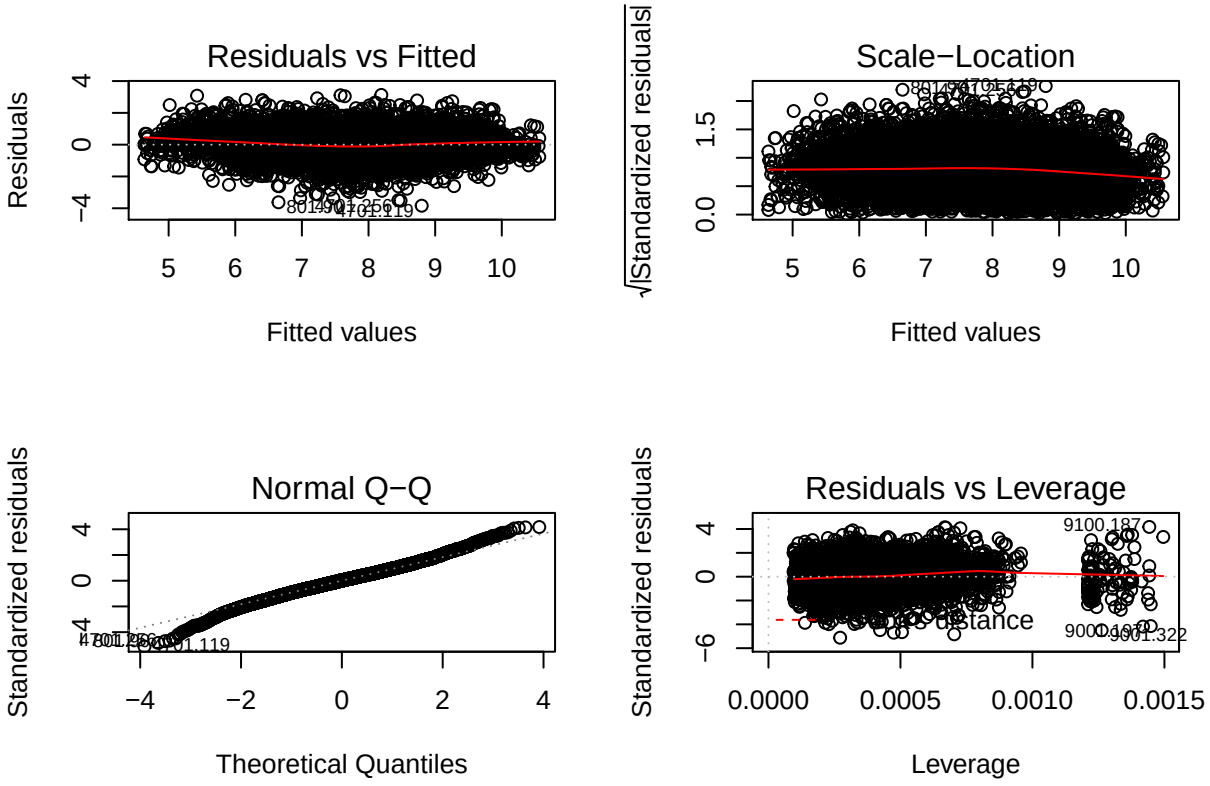
```
##
## [[7]]
##
##  Shapiro-Wilk normality test
##
## data:  sample(fit$residuals, 5000)
## W = 0.99073, p-value < 2.2e-16
##
##
## [[8]]
##
##  Anderson-Darling normality test
##
## data:  fit$residuals
## A = 18.289, p-value < 2.2e-16
##
##
## [[9]]
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  fit$residuals
## D = 0.072424, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
###regression linéaire
fit500_ln<-lm(log(income)~ log(gdpppp) + gini + c_i_parent,df500)

###résumé analyse
summary(fit500_ln)
```

```
##
## Call:
## lm(formula = log(income) ~ log(gdpppp) + gini + c_i_parent, data = df500)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4462 -0.4582  0.0053  0.4655  4.3493
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.360e-01  3.104e-03   108.2   <2e-16 ***
## log(gdpppp)  8.600e-01  2.626e-04  3275.4   <2e-16 ***
## gini        -1.530e-02  3.699e-05  -413.6   <2e-16 ***
## c_i_parent   1.074e-02  1.092e-05   983.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7427 on 5549996 degrees of freedom
## Multiple R-squared:  0.7132, Adjusted R-squared:  0.7132
## F-statistic: 4.6e+06 on 3 and 5549996 DF,  p-value: < 2.2e-16
```

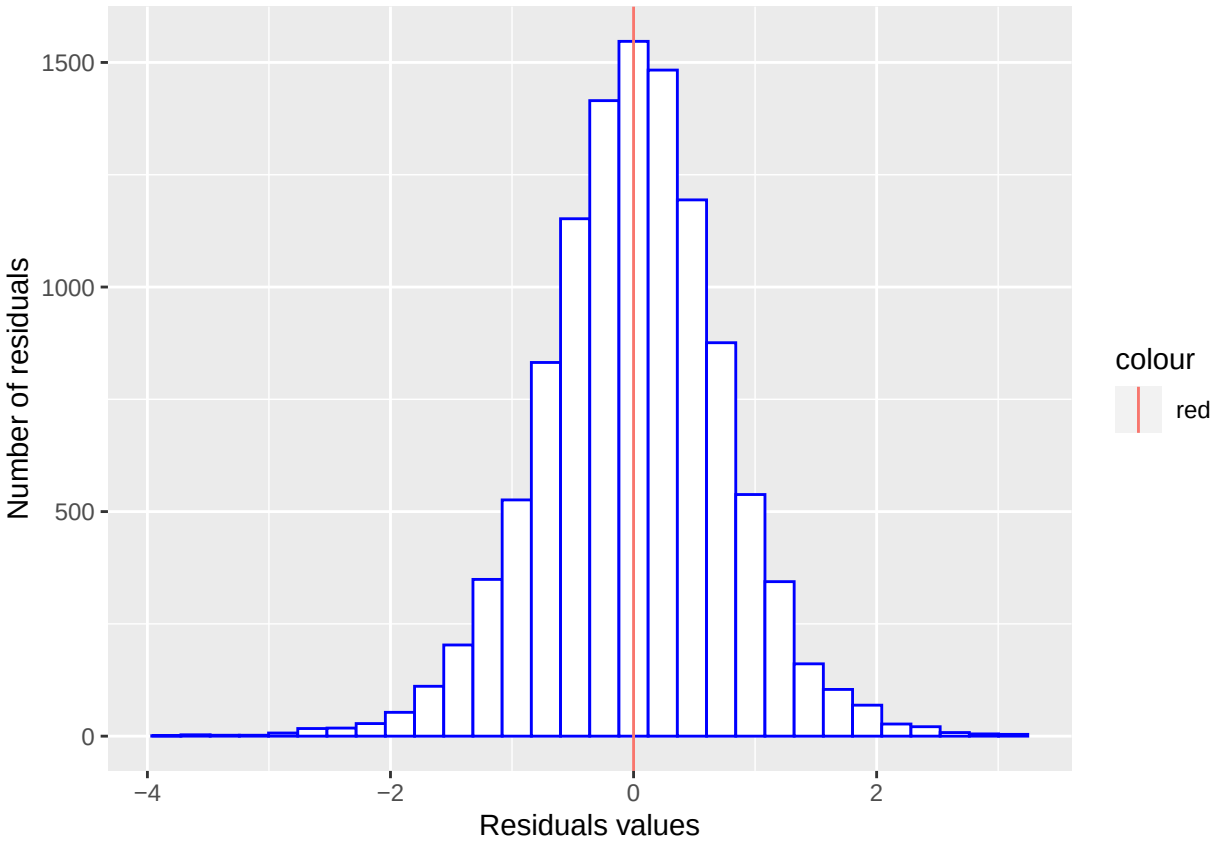
```r
###regression linéaire(sur 11100 pays au hasard)
###graphiques et test trop long sur l'ensemble (résultat très proche)
fit500_ln_<-lm(log(income)~ log(gdpppp) + gini + c_i_parent,
               df500[sample(1:nrow(df500),11100),])


###les graphiques et tests
f_graph(fit500_ln_, 3)
```

```
## Warning in ks.test(fit$residuals, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test
```

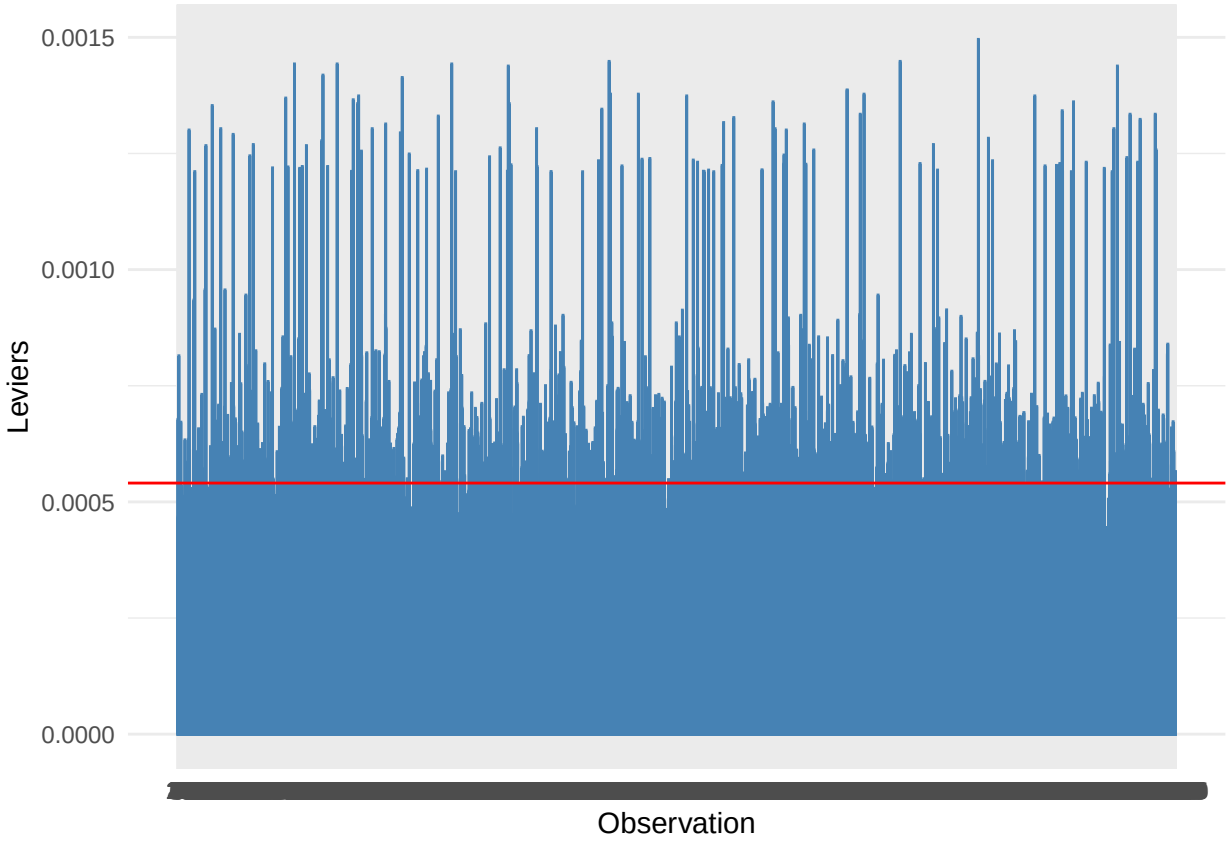



```
## [[1]]
## NULL
##
## [[2]]
```

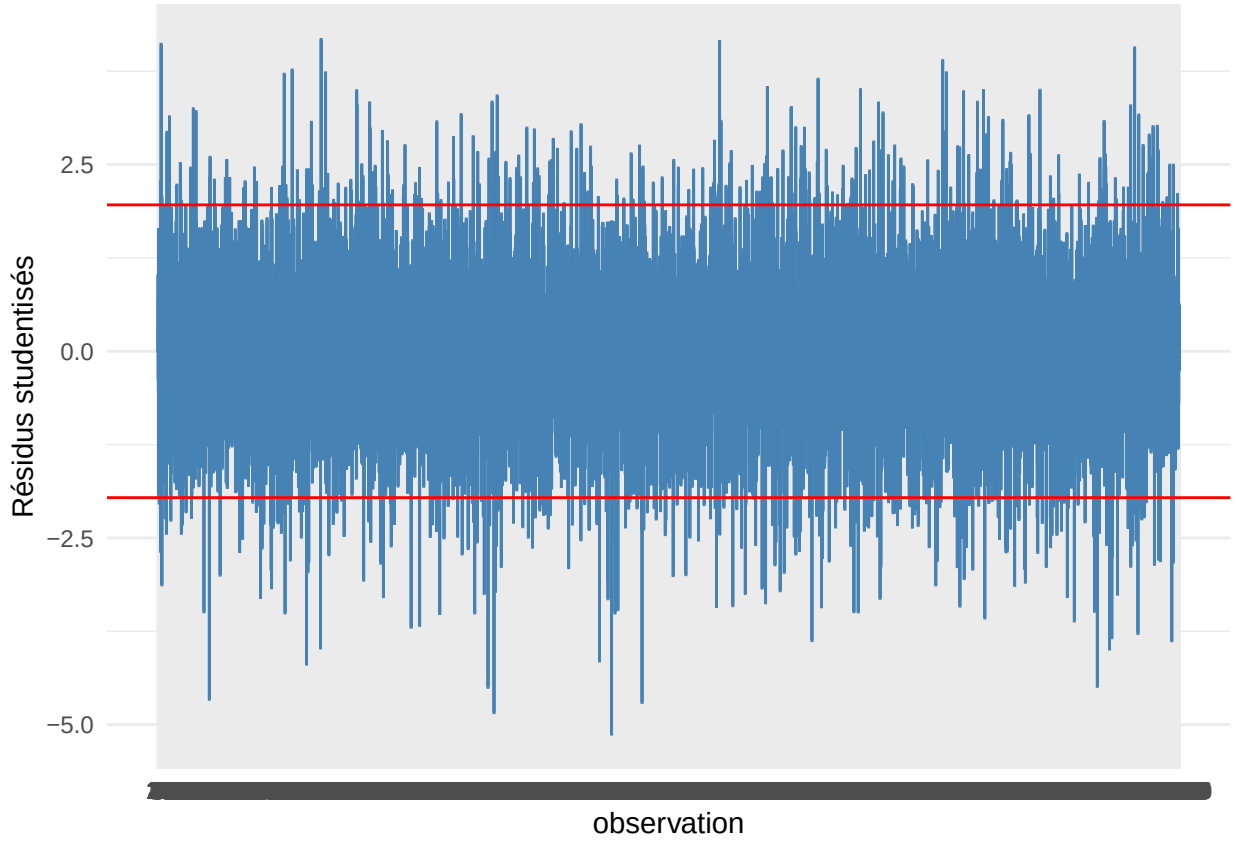
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


1500
1000
500
0
Number of residuals
−4
−2
0
2
Residuals values
colour
red

```
##
## [[3]]
```




```
##
## [[4]]
```




```
##
## [[5]]
##
## Call:
## lm(formula = log(income) ~ log(gdpppp) + gini + c_i_parent, data = df500[sample(1:nrow(df500),
##     11100), ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8399 -0.4609  0.0186  0.4650  3.1273
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.4865032  0.0699954   6.951 3.84e-12 ***
## log(gdpppp)  0.8475278  0.0059206 143.149  < 2e-16 ***
## gini        -0.0158293  0.0008355 -18.946  < 2e-16 ***
## c_i_parent   0.0105804  0.0002471  42.814  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7499 on 11096 degrees of freedom
## Multiple R-squared:  0.7041, Adjusted R-squared:  0.704
## F-statistic:  8801 on 3 and 11096 DF,  p-value: < 2.2e-16
##
##
## [[6]]
```

```
## log(gdpppp)        gini  c_i_parent
##    1.090393    1.091253    1.000833
##
## [[7]]
##
##  Shapiro-Wilk normality test
##
## data:  sample(fit$residuals, 5000)
## W = 0.99503, p-value = 4.438e-12
##
##
## [[8]]
##
##  Anderson-Darling normality test
##
## data:  fit$residuals
## A = 11.759, p-value < 2.2e-16
##
##
## [[9]]
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  fit$residuals
## D = 0.087138, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
###Je crée une fonction pour analyser les résidues studentiser en valeur
###absolu

lm_analyse<- function(df, fit){

df$rstudent <- rstudent(fit)%>%abs()

df
}
```

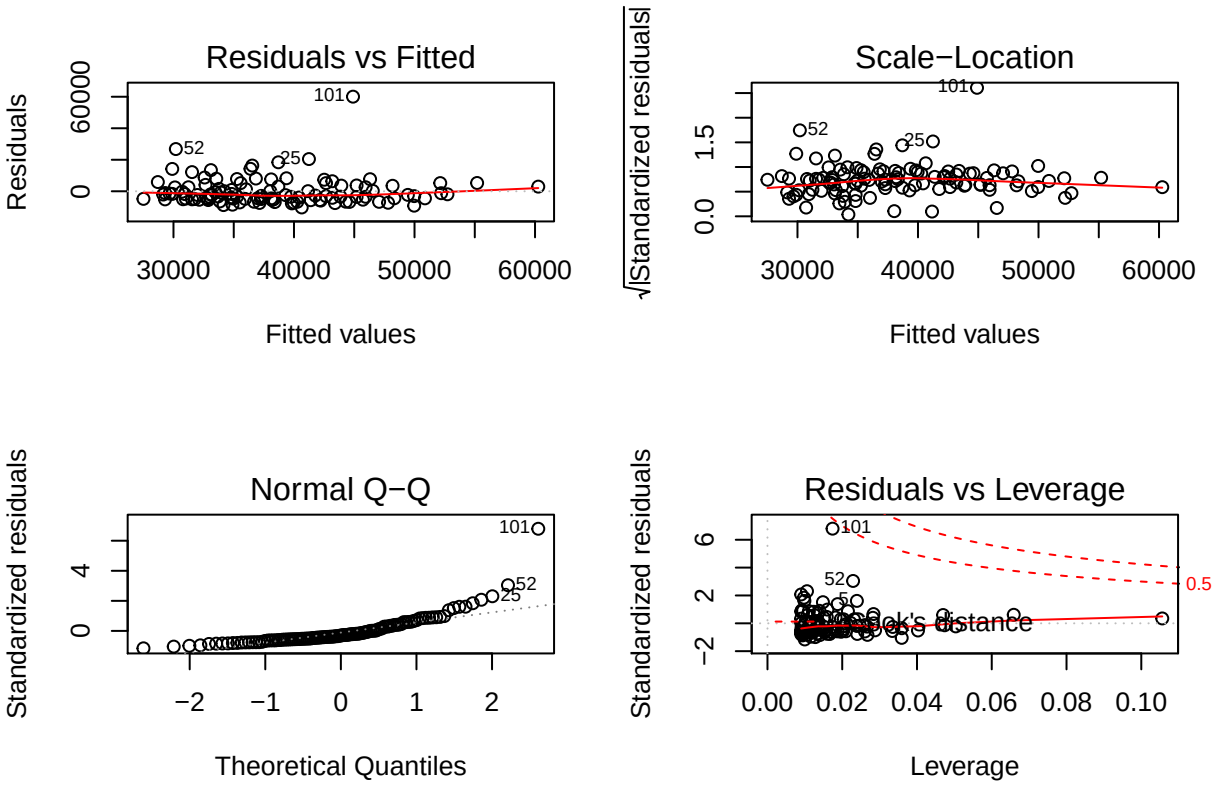
```
###je crée la colonne rstudent qui donne la valeur absolue de la valeur
###du résidu studentisé
df_rstudent<-lm_analyse(df500, fit500_ln)

x<-df%>%select(-"income",-"quantile",-"nb_quantiles")%>%distinct()

###Je somme les erreurs par pays
df_rstudent_sum<-df_rstudent%>%group_by(country_full)%>%
  summarise(error = sum(rstudent))%>%inner_join(x)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## Joining, by = "country_full"
```

```r
fit_error<-lm(error~ gini, df_rstudent_sum)

###Analyse
summary(fit_error)
```

```
##
## Call:
## lm(formula = error ~ gini, data = df_rstudent_sum)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -10269  -4984  -2714   2966  60052
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10272.73    3681.70   2.790  0.00622 **
## gini          746.21      94.84   7.868 2.82e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8916 on 109 degrees of freedom
## Multiple R-squared:  0.3622, Adjusted R-squared:  0.3564
## F-statistic:  61.9 on 1 and 109 DF,  p-value: 2.823e-12
```

```r
###Graphe analyse résidus
layout(matrix(1:4, 2, 2))
plot(fit_error)
```



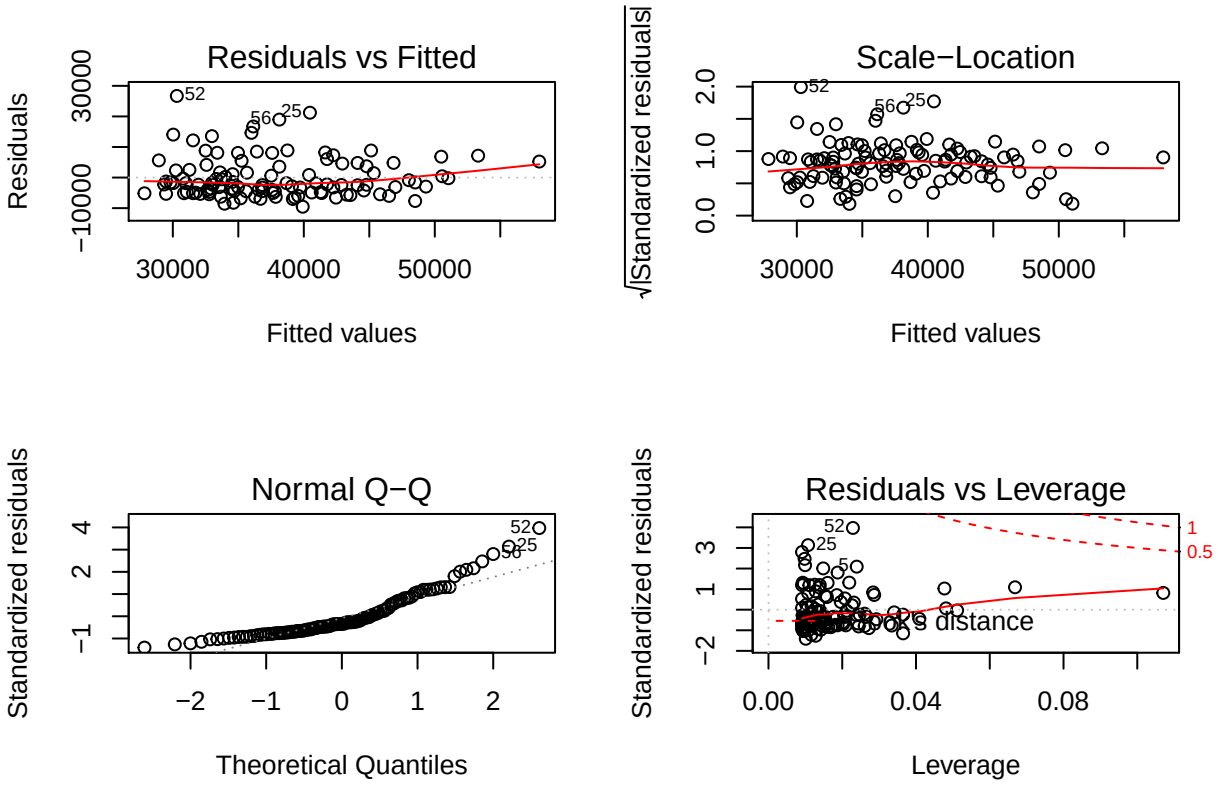

```
df_rstudent_sum2<-df_rstudent_sum%>%
  filter(row.names(df_rstudent_sum) != 101)
```

```
fit_error2<-lm(error~ gini, df_rstudent_sum2)

###Analyse
summary(fit_error2)
```

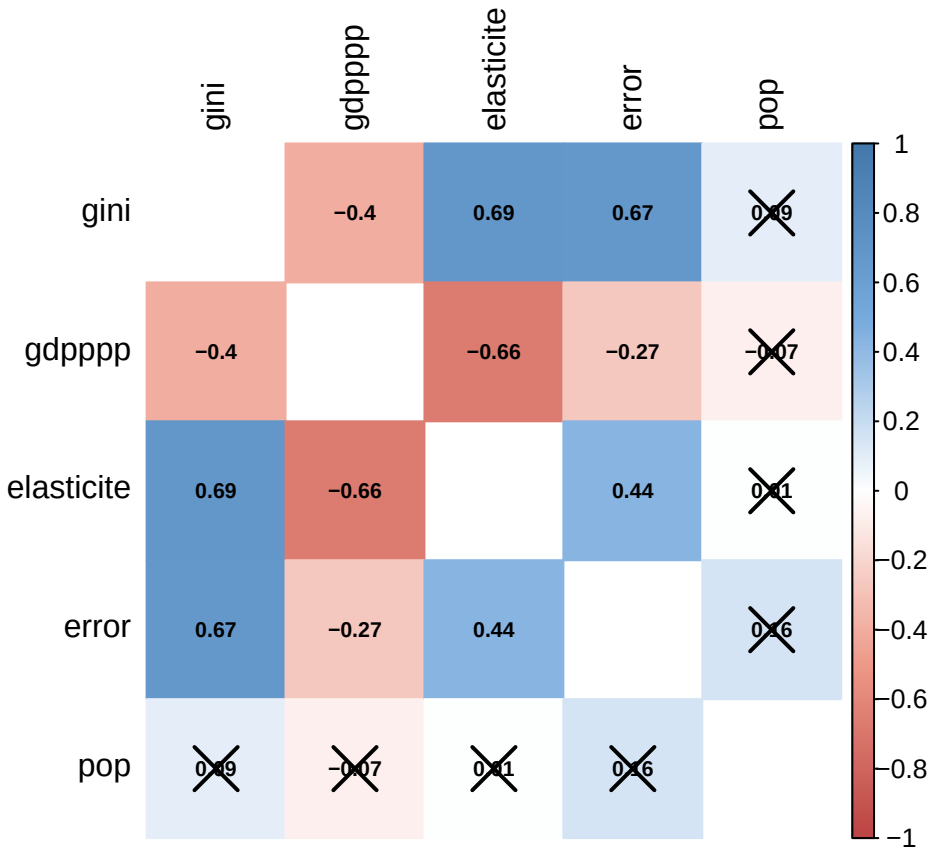
```
##
## Call:
## lm(formula = error ~ gini, data = df_rstudent_sum2)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -9545  -4733  -2295   3660  26693
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11974.16    2814.66   4.254 4.48e-05 ***
## gini          686.60      72.65   9.451 8.26e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6801 on 108 degrees of freedom
## Multiple R-squared:  0.4527, Adjusted R-squared:  0.4476
```

```
## F-statistic: 89.32 on 1 and 108 DF,  p-value: 8.261e-16
```

```
###Graphe analyse résidus
layout(matrix(1:4, 2, 2))
plot(fit_error2)
```




```
###je réalise la matrice des corrélation de l'erreur et des variables
###propre à chaque pays
df_cor<-select(df_rstudent_sum2,
               c("gini","gdpppp","elasticite",
                 "error","pop"))
cormat <- cor(df_cor, method = "pearson")
p.mat <- cor.mtest(df_cor)$p
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD",
"#4477AA"))
corrplot(cormat, method = "color", col = col(200),
type = "full", order = "original", number.cex = .7,
addCoef.col = "black", # Add coefficient of correlation
tl.col = "black", tl.srt = 90, # Text label color and rotation
# Combine with significance
p.mat = p.mat, sig.level = 0.05,
# hide correlation coefficient on the principal diagonal
diag = FALSE)
```



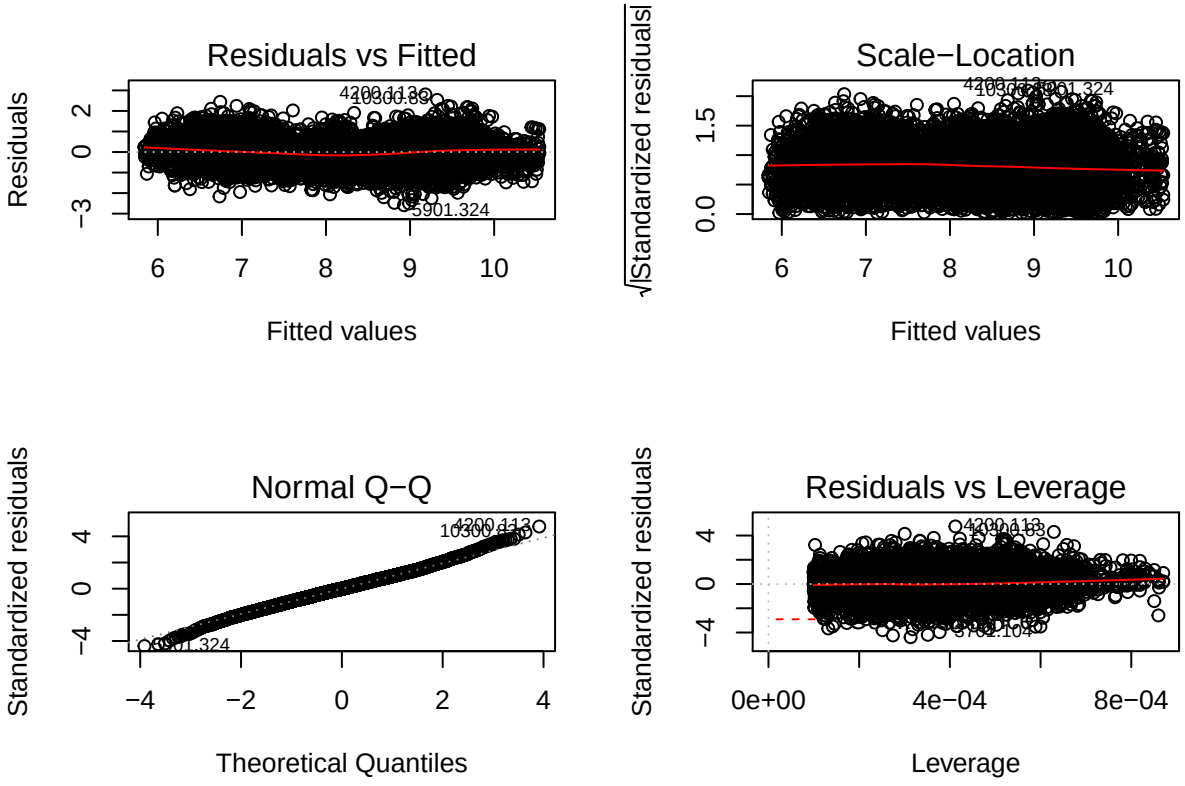

```
df500_gini<-df500%>%filter(gini<35)

fit500_ln_gini<-lm(log(income)~ log(gdpppp) + gini + c_i_parent,
                   df500_gini)

summary(fit500_ln_gini)
```

```
##
## Call:
## lm(formula = log(income) ~ log(gdpppp) + gini + c_i_parent, data = df500_gini)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.93288 -0.37977  0.00113  0.37669  2.81283
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.348e+00  6.256e-03  -215.5   <2e-16 ***
## log(gdpppp)  9.380e-01  3.336e-04  2812.3   <2e-16 ***
## gini         2.317e-02  1.428e-04   162.3   <2e-16 ***
## c_i_parent   7.119e-03  1.302e-05   546.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5884 on 2449996 degrees of freedom
```

```
## Multiple R-squared:  0.7921, Adjusted R-squared:  0.7921
## F-statistic: 3.111e+06 on 3 and 2449996 DF,  p-value: < 2.2e-16
```

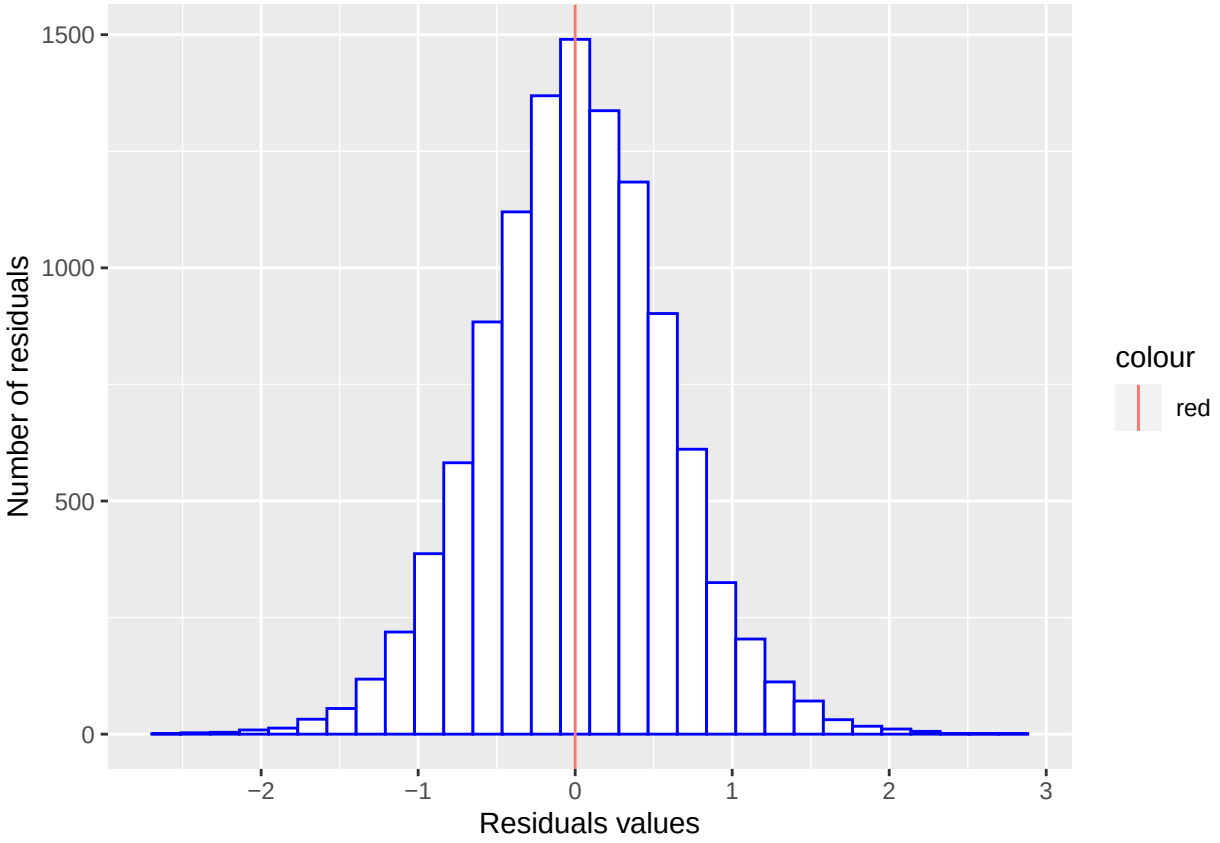
```
###regression linéaire(sur 11100 pays au hasard)
###graphiques et test trop long sur l'ensemble (résultat très proche)
fit500_gini_<-lm(log(income)~ log(gdpppp) + gini + c_i_parent,
              df500_gini[sample(1:nrow(df500_gini),11100),])


###les graphiques et tests
f_graph(fit500_gini_, 3)
```

```
## Warning in ks.test(fit$residuals, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test
```

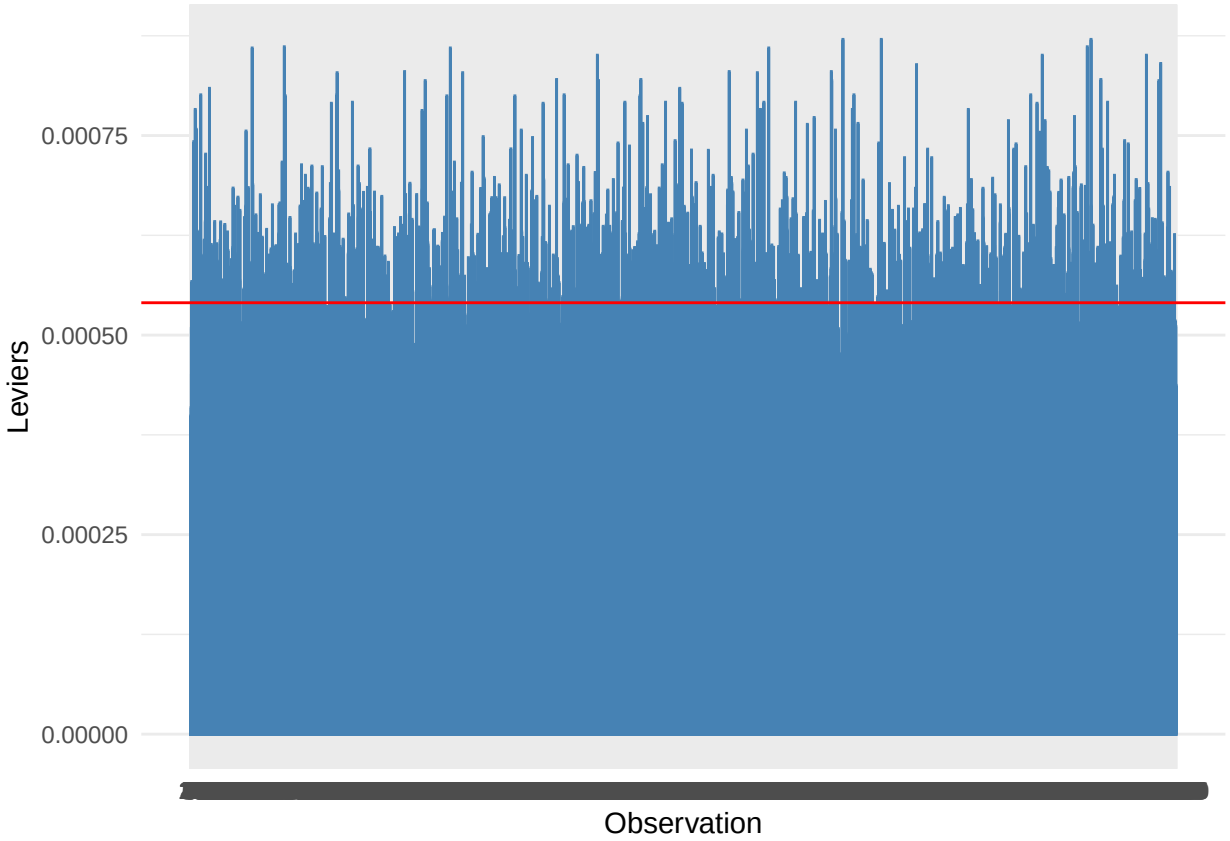



```
## [[1]]
## NULL
##
## [[2]]
```

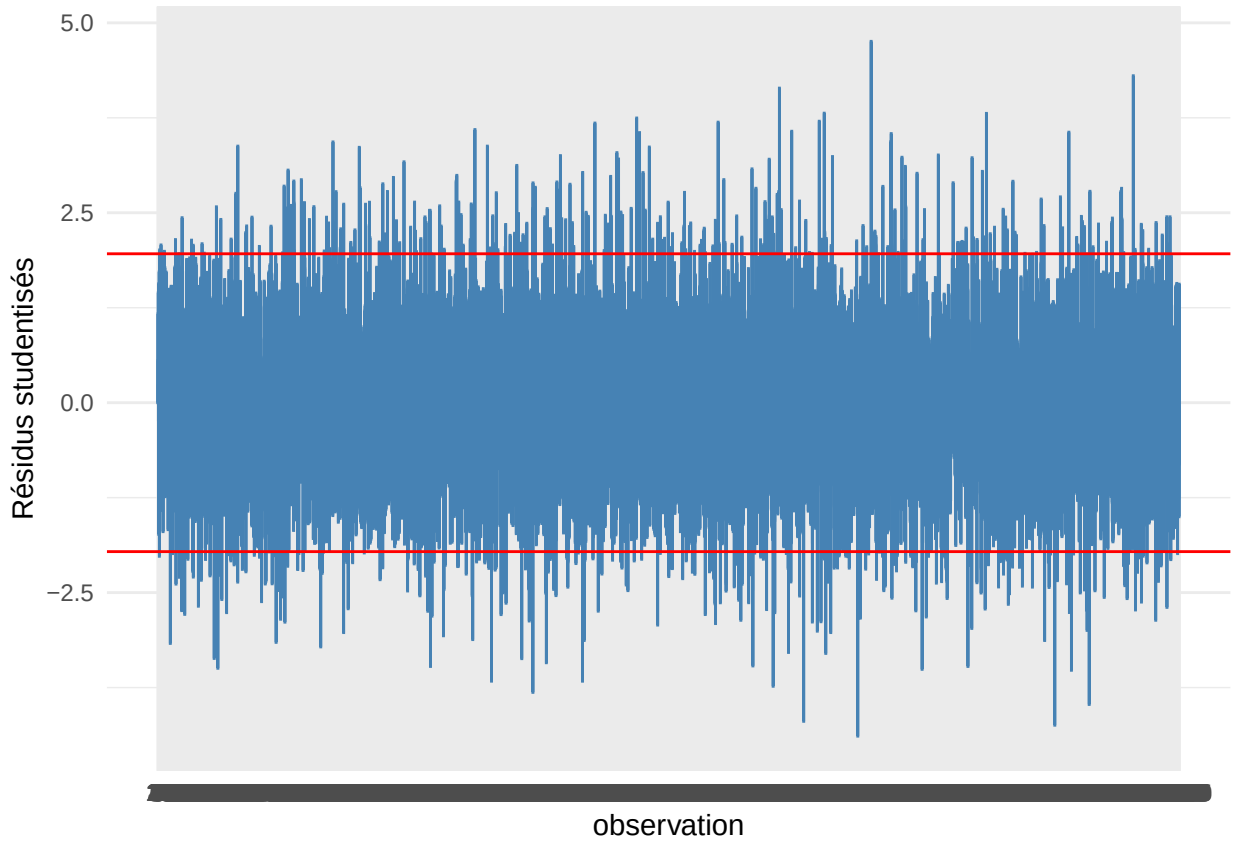
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


1500
1000
500
0
Number of residuals
−2
−1
0
1
2
3
Residuals values
colour
red

```
##
## [[3]]
```




```
##
## [[4]]
```




```
##
## [[5]]
##
## Call:
## lm(formula = log(income) ~ log(gdpppp) + gini + c_i_parent, data = df500_gini[sample(1:nrow(df500_gi
##     11100), ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.58861 -0.38629 -0.00296  0.38621  2.80456
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.2619012  0.0930891  -13.56   <2e-16 ***
## log(gdpppp)  0.9313418  0.0049734  187.26   <2e-16 ***
## gini         0.0221421  0.0021269   10.41   <2e-16 ***
## c_i_parent   0.0071879  0.0001948   36.90   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5903 on 11096 degrees of freedom
## Multiple R-squared:  0.7897, Adjusted R-squared:  0.7897
## F-statistic: 1.389e+04 on 3 and 11096 DF,  p-value: < 2.2e-16
##
##
## [[6]]
```

```
## log(gdpppp)        gini  c_i_parent
##    1.192981    1.192741    1.000279
##
## [[7]]
##
##  Shapiro-Wilk normality test
##
## data:  sample(fit$residuals, 5000)
## W = 0.99799, p-value = 4.362e-06
##
##
## [[8]]
##
##  Anderson-Darling normality test
##
## data:  fit$residuals
## A = 3.3316, p-value = 2.458e-08
##
##
## [[9]]
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  fit$residuals
## D = 0.13411, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
###Un modèle basé sur les pays ayant un indice de gini inférieur
###à 35 donne une meilleur prédiction sur ces pays là
```

```
df500_gini<-df500%>%filter(gini>35)
fit500_ln_gini<-lm(log(income)~ log(gdpppp) + gini + c_i_parent,
                 df500_gini)

summary(fit500_ln_gini)
```

```
##
## Call:
## lm(formula = log(income) ~ log(gdpppp) + gini + c_i_parent, data = df500_gini)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4592 -0.5124  0.0071  0.5251  4.4103
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.089e-01  4.494e-03   135.5   <2e-16 ***
## log(gdpppp)  7.987e-01  4.278e-04  1866.8   <2e-16 ***
## gini        -1.363e-02  6.883e-05  -198.0   <2e-16 ***
## c_i_parent   1.375e-02  1.634e-05   841.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.8237 on 3049996 degrees of freedom
## Multiple R-squared:  0.579,  Adjusted R-squared:  0.579
## F-statistic: 1.398e+06 on 3 and 3049996 DF,  p-value: < 2.2e-16
```

```
###Un modèle basé sur les pays ayant un indice de gini suérieur
###à 35 donne une moins bonne prédiction sur ces pays là
```

```
df_rstudent$fitted<-fit500_ln$fitted.values%>%exp()

df_rstudent%>%arrange(desc(fitted))%>%select(country_full)%>%distinct()
```

```
##                                   country_full
## 6008.45                             Luxembourg
## 7601.88                                Norvège
## 4603.462                               Irlande
## 7204.6                                Pays-Bas
## 305.200                               Autriche
## 4101.453                               Islande
## 9401.404                                 Suède
## 2702.43                               Danemark
## 3201.399                              Finlande
## 612.132                               Belgique
## 10405.141                           États-Unis
## 3508.69                              Allemagne
## 1501.72                                 Canada
## 10205.424                         Royaume-Uni
## 8909.499                             Slovénie
## 5015.256                                Japon
## 3305.377                               France
## 9104.220                              Espagne
## 4804.18                                Italie
## 2511.178                               Chypre
## 3708.233                                Grèce
## 2627.226                    République Tchèque
## 2012.122                               Taïwan
## 8708.164                           Slovaquie
## 4715.126                               Israël
## 8302.141                              Portugal
## 4005.283                               Hongrie
## 3105.276                               Estonie
## 5402.120                    République de Corée
## 2405.200                               Croatie
## 5907.182                              Lituanie
## 8205.296                               Pologne
## 5705.387                              Lettonie
## 1205.86                                Bélarus
## 8603.72                   Fédération de Russie
## 5107.442                           Kazakhstan
## 1108.430                              Bulgarie
## 11008.309                           Monténégro
## 10906.188                              Serbie
## 8504.88                               Roumanie
## 216.125                              Argentine
```

```
## 6303.493                              Malaisie
## 10716.231                            Venezuela
## 9712.479                               Turquie
## 6623.346                               Mexique
## 1806.484                                 Chili
## 10624.49                               Uruguay
## 4408.375           République Islamique d'Iran
## 7818.391                               Panama
## 107.478                             Azerbaïdjan
## 10001.428                    Macédoine du Nord
## 2316.251                            Costa Rica
## 9905.72                                Ukraine
## 99.5                                   Albanie
## 902.475                     Bosnie-Herzégovine
## 1016.307                               Brésil
## 504.246                                Arménie
## 8018.344                                 Pérou
## 9606.158                             Thaïlande
## 2815.330                 République Dominicaine
## 2916.42                               Équateur
## 2109.40                               Colombie
## 9012.277                        Afrique du Sud
## 10112.109                               Égypte
## 5207.81                               Jordanie
## 3031.10                            El Salvador
## 1905.365                                 Chine
## 706.112                                Bhoutan
## 3405.259                               Géorgie
## 9322.2                                Swaziland
## 1705.4                                Sri Lanka
## 4303.172                              Indonésie
## 6909.408                                  Maroc
## 6717.496                               Mongolie
## 7912.362                               Paraguay
## 4504.79                                    Iraq
## 6804.155                  République de Moldova
## 4207.471                                  Inde
## 8110.210                           Philippines
## 812.97                                  Bolivie
## 7703.465                               Pakistan
## 8900.8                                 Viet Nam
## 9289.79                                  Soudan
## 3917.272                               Honduras
## 10805.459                                Yémen
## 7323.6                                Nicaragua
## 5519.251                            Kirghizistan
## 6519.245                              Mauritanie
## 9504.229                              Tadjikistan
## 5602.92   République Démocratique Populaire Lao
## 1308.405                                Cambodge
## 1417.126                                Cameroun
## 7519.499                                Nigéria
## 5319.456                                  Kenya
## 4920.461                          Côte d'Ivoire
```

```
## 406.430                              Bangladesh
## 3613.315                                  Ghana
## 8410.339                             Timor-Leste
## 10340.119          République-Unie de Tanzanie
## 7106.401                                  Népal
## 10600.9                             Burkina Faso
## 6417.277                                  Mali
## 9898.4                                 Ouganda
## 3814.30                                  Guinée
## 6132.271                              Madagascar
## 7007.257                              Mozambique
## 6213.331                                  Malawi
## 7426.457                                   Niger
## 1605.413              République Centrafricaine
## 5806.234                                 Libéria
## 2219.406       République Démocratique du Congo
```

```
df_rstudent$fitted<-fit500_ln$fitted.values%>%exp()

df_rstudent%>%arrange(desc(income))%>%select(country_full)%>%distinct()
```

```
##                               country_full
## 10500                           États-Unis
## 4200                               Islande
## 10300                           Royaume-Uni
## 1600                                Canada
## 3400                                France
## 7700                               Norvège
## 6100                            Luxembourg
## 3600                             Allemagne
## 1900                                 Chili
## 2600                                Chypre
## 7300                             Pays-Bas
## 4700                              Irlande
## 2800                             Danemark
## 5100                                Japon
## 2100                              Taïwan
## 700                              Belgique
## 3300                             Finlande
## 9100                       Afrique du Sud
## 5500                    République de Corée
## 4900                               Italie
## 400                              Autriche
## 4800                               Israël
## 9500                                Suède
## 3800                                Grèce
## 1100                              Brésil
## 8400                             Portugal
## 2400                           Costa Rica
## 7900                               Panama
## 4000                             Honduras
## 6400                             Malaisie
## 8700                    Fédération de Russie
## 9200                               Espagne
```

```
## 2200                          Colombie
## 4500           République Islamique d'Iran
## 10700                          Uruguay
## 6700                           Mexique
## 900                            Bolivie
## 300                          Argentine
## 8000                           Paraguay
## 9800                           Turquie
## 2900               République Dominicaine
## 9000                           Slovénie
## 6000                           Lituanie
## 2500                           Croatie
## 2700                 République Tchèque
## 3000                           Équateur
## 5800                           Lettonie
## 3200                           Estonie
## 8300                           Pologne
## 1000                 Bosnie-Herzégovine
## 8100                             Pérou
## 1200                           Bulgarie
## 11100                        Monténégro
## 10100                 Macédoine du Nord
## 4100                           Hongrie
## 3100                        El Salvador
## 10800                          Venezuela
## 7400                           Nicaragua
## 8800                           Slovaquie
## 9700                           Thaïlande
## 7000                               Maroc
## 100                              Albanie
## 11000                            Serbie
## 2000                             Chine
## 1800                           Sri Lanka
## 8600                            Roumanie
## 5300                           Jordanie
## 10200                            Égypte
## 1700            République Centrafricaine
## 4600                               Iraq
## 6800                           Mongolie
## 6600                         Mauritanie
## 1300                            Bélarus
## 1500                           Cameroun
## 200                          Azerbaïdjan
## 4400                           Indonésie
## 8200                         Philippines
## 10000                          Ukraine
## 6300                             Malawi
## 8900                           Viet Nam
## 9600                         Tadjikistan
## 6900               République de Moldova
## 10900                            Yémen
## 800                              Bhoutan
## 5200                          Kazakhstan
## 9900                             Ouganda
```

```
## 7100                              Mozambique
## 5600                            Kirghizistan
## 3500                                 Géorgie
## 10600                           Burkina Faso
## 1400                                Cambodge
## 5700  République Démocratique Populaire Lao
## 7600                                Nigéria
## 600                                 Arménie
## 4300                                   Inde
## 3700                                 Ghana
## 500                               Bangladesh
## 9300                                  Soudan
## 7800                                Pakistan
## 7500                                   Niger
## 7200                                   Népal
## 8500                             Timor-Leste
## 3900                                 Guinée
## 5900                                Libéria
## 9400                              Swaziland
## 10400           République-Unie de Tanzanie
## 6200                             Madagascar
## 6500                                   Mali
## 5000                           Côte d'Ivoire
## 2300       République Démocratique du Congo
## 5400                                  Kenya
```