Annalyse de données

Abdoul Oudouss Diakite, Othmane ETTADLAOUI

01 June, 2022

Contents

Introduction

Ce projet d'analyse de donnés consiste à l'explication de différentes manières le nombre de crime aux Etats-Unis. Pour se faire, nous disposons d'un jeu de disponible sur GitHub que nous appellerons `Comminities`(*Télécharger*) dorénavant.

communityname	State	countyCode	communityCode	fold	pop	perHoush	pctBlack	pctW
BerkeleyHeightstownship	NJ	39	5320	1	11980	3.10	1.37	91
Marpletownship	PA	45	47616	1	23123	2.82	0.80	95
Tigardcity	OR	?	?	1	29344	2.43	0.74	94
Gloversvillecity	NY	35	29443	1	16656	2.40	1.70	97
Bemidjicity	MN	7	5068	1	11245	2.76	0.53	89

Description du jeu de données

Le jeu de données dispose de 147 variables pour 2215 observations. Les valeurs manquantes sont représentées par `"?"`.
De nombreuses variables sont incluses afin que les algorithmes qui sélectionnent ou apprennent les poids des attributs puissent être testés. Cependant, les attributs clairement non liés n'ont pas été inclus; les attributs ont été sélectionnés s'il y avait un lien plausible avec la criminalité (N = 125), plus les variables de criminalité qui sont des variables dépendantes potentielles. Les variables incluses dans l'ensemble de données impliquent la communauté, telles que le pourcentage de la population considérée comme urbaine et le revenu familial médian, et impliquent l'application de la loi, telles que le nombre d'agents de police par habitant et le pourcentage d'agents affectés aux unités de lutte contre la drogue. Les attributs de crime (N = 18) qui pourraient être prédits sont les 8 crimes considérés comme des «crimes indexés» par le FBI (Meurtres, viols, vols qualifiés, ….), versions par habitant (en fait pour 100 000 habitants) de chacun, et crimes violents par habitant et crimes non violents par habitant).
Pour faciliter la tâche aux lecteurs de se projet, nous avons créés un fichier csv nommé `Description.csv`(*Télécharger*) qui contient les noms des variables dans la colonne `feature` et leur descriptions dans la colonne `Description`

feature	Description
communityname	Community name - not predictive - for information only (string)
state	US state (by 2 letter postal abbreviation)(nominal)
countyCode	numeric code for county - not predictive, and many missing values (numeric)
communityCode	numeric code for community - not predictive and many missing values (numeric)
fold	fold number for non-random 10 fold cross validation, potentially useful for debuggi
population	population for community

Source des données

UCI Machine Learning[1]

[1] https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime+Unnormalized#

Chapter 1

Régression linéaire

1.1 Introduction

La régression lineaire est une méthode statistique qui permet de trouver une relation lineaire entre des variables quantitives, une à expliquer et d'autres explicatives. C'est en fait un ajustement affine de la forme :

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i \tag{1.1}$$

$$i \in \{1, 2, 3 \ldots, n\}$$

- y_i représentent la $i$ème valeur de la variable dépendantes y.
- x_{ij} représente la mesure de la $i$ème observation de la variable explicative X_j
- les β_j sont les paramètres inconnus du modèle à estimer
- ϵ_i représente le bruit associé à la $i$ème observation

L'équation précédente peut être écrite sous une forme matricielle de cette manière :

$$y = X\beta + \epsilon \tag{1.2}$$

avec :

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad X = \begin{pmatrix} 1 & x_{11} & x_{12} & ... & x_{1p} \\ 1 & x_{21} & x_{22} & ... & x_{2p} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & ... & x_{np} \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} \quad \epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix} \tag{1.3}$$

Commençons par importer le jeu de données que nous nommerons $dfcom$:

```
Communities = read.csv("data/Communities.csv",row.names = 1)
```

communityname	State	countyCode	communityCode	fold	pop	perHoush	pctI
BerkeleyHeightstownship	NJ	39	5320	1	11980	3.10	
Marpletownship	PA	45	47616	1	23123	2.82	
Tigardcity	OR	?	?	1	29344	2.43	
Gloversvillecity	NY	35	29443	1	16656	2.40	
Bemidjicity	MN	7	5068	1	11245	2.76	
Springfieldcity	MO	?	?	1	140494	2.45	
Norwoodtown	MA	21	50250	1	28700	2.60	
Andersoncity	IN	?	?	1	59459	2.45	
Fargocity	ND	17	25700	1	74111	2.46	
Wacocity	TX	?	?	1	103590	2.62	

1.2 Application de la régression linéaire simple

Corrélation

Comme nous l'avons mentionner dans l'introduction, le but de se projet est d'expliquer de différentes manières les meurtes aux USA. Par conséquent, on peut choisir comme variable dépendante, les crimes(`murders`) et chercher les variables explicatives. Dans le cas de le régression linéaire simple il doit exister une corrélation assez importante entre la variable y(`muders`) et X que nous recherchons actuellement. Donc commençons par filtrer les fortes corrélation avec la variable y dans notre jeu de données.

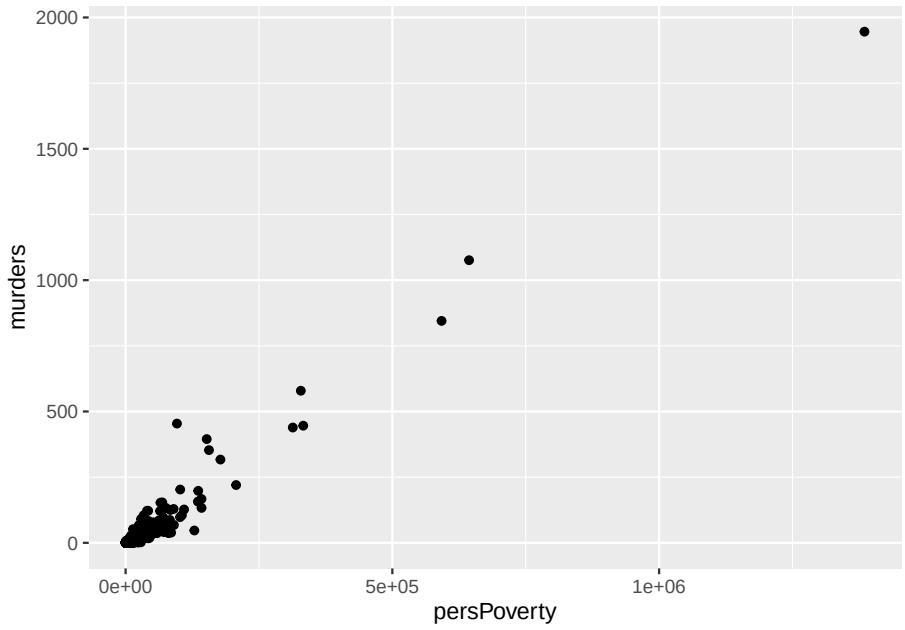
```
# Correlation matrix
corCom = correlation::correlation(Communities)
# Filtered correlation, bound =0.8
corCom[(corCom$r>0.8) & corCom$Parameter2=='murders',]
```

```
## # Correlation Matrix (pearson-method)
##
## Parameter1       | Parameter2 |    r |       95% CI | t(2213) |         p
## ------------------------------------------------------------------------
## pop              |    murders | 0.96 | [0.96, 0.96] |  159.80 | < .001***
## persUrban        |    murders | 0.96 | [0.95, 0.96] |  156.13 | < .001***
## persPoverty      |    murders | 0.98 | [0.97, 0.98] |  211.42 | < .001***
## kidsBornNevrMarr |    murders | 0.98 | [0.98, 0.98] |  221.27 | < .001***
## numForeignBorn   |    murders | 0.89 | [0.88, 0.90] |   92.94 | < .001***
## houseVacant      |    murders | 0.90 | [0.89, 0.90] |   95.29 | < .001***
## persEmergShelt   |    murders | 0.89 | [0.88, 0.90] |   93.14 | < .001***
## persHomeless     |    murders | 0.85 | [0.84, 0.86] |   76.49 | < .001***
##
## p-value adjustment method: Holm (1979)
## Observations: 2215
```

Nuage de points

Le tableau précédent indique les variables fortement corrélées avec notre *output* `murders`. Prenons l'exemple de la variable `persPoverty` qui représente le nombre de personnes sous le seuil de pauvreté.

```
library(ggplot2)
fig = ggplot(data = Communities,aes(x=persPoverty,y=murders))+
  geom_point()
fig
```



Entraînement de modèle & Droite de régression

La figure précédente laisse parraître qu'il pourrait effectivement exister une relation linéaire entre `murders` et `persPoverty`. Appliquons la fonction `lm()` pour voir ce qu'il en est vraiment ! Pour faire une analyse des résidus pltard, nous n'entrainerons que 75% du jeu de données et le reste servira à la prédiction.

```
library(dplyr)
# Train_Test_Splite
set.seed(1345)
# Pourcentage de donnees correspondant a 25%
per = dim(Communities)[1]%/%4
echantillon <- sample(1:dim(Communities)[1]) %>% .[1:per]
lmDataTrain = Communities[-echantillon,c("murders","persPoverty")]
lmDataTest = Communities[echantillon,c("murders","persPoverty")]
```

```
#Model Training
lmSimple <- lm(murders~persPoverty,data = lmDataTrain)
summary(lmSimple)
```

```
##
## Call:
```

```
## lm(formula = murders ~ persPoverty, data = lmDataTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -136.43   -1.13    1.32    2.52  317.80
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.271e+00  3.340e-01  -9.796   <2e-16 ***
## persPoverty  1.449e-03  7.447e-06 194.519   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.4 on 1660 degrees of freedom
## Multiple R-squared:  0.958,  Adjusted R-squared:  0.9579
## F-statistic: 3.784e+04 on 1 and 1660 DF,  p-value: < 2.2e-16
```

La sortie de la fonction `summary()` indique des $p - values$ très inférieures à 5%, donc on rejette l'hypothèse de nullité des β. On peut aussi constater que le coefficient de détermination R^2 vaut 0.958 ce qui signifie que notre modèle a un score de 95.8%. Ce dernier reflète une bonne qualité du modèle.

```
ggstatsplot::ggscatterstats(
  data = lmDataTrain,
  x = persPoverty,
  y = murders,
  xlab  = "le nombre de personnes sous le seuil de pauvreté",
  ylab  = "le nombre de meurtres",
  title = "Droite de régression",
  messages = FALSE
)
```

```
## Registered S3 method overwritten by 'ggside':
##   method from
##   +.gg   ggplot2
```

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

Prédiction

A présent, nous pouvons utiliser notre échantillon non entrainé de données pour prédire à partir de notre model, quel aurait était le nombre de meurtres pour chauque x_i.

```
X_test=as.data.frame(lmDataTest[["persPoverty"]])
colnames(X_test)="persPoverty"
y_predict = predict(object = lmSimple,X_test)
```

Graphes des résidus

On peut représenter le graphe des $\hat{y}$ prédits et des y. Pour un modèle parfait, le nuage de point doit être sur la première bissectrice.

```
ggplot(data =lmDataTest) +
  geom_point(aes(persPoverty,murders),color = 'darkgreen',
             size =2,shape=22,fill ="darkgreen") +
  geom_point(aes(x = persPoverty, y =y_predict), color ='blue') +
  geom_segment(aes(x =persPoverty ,
```

```
                    y = murders, xend = persPoverty, yend = y_predict),
                color = 'red')
```

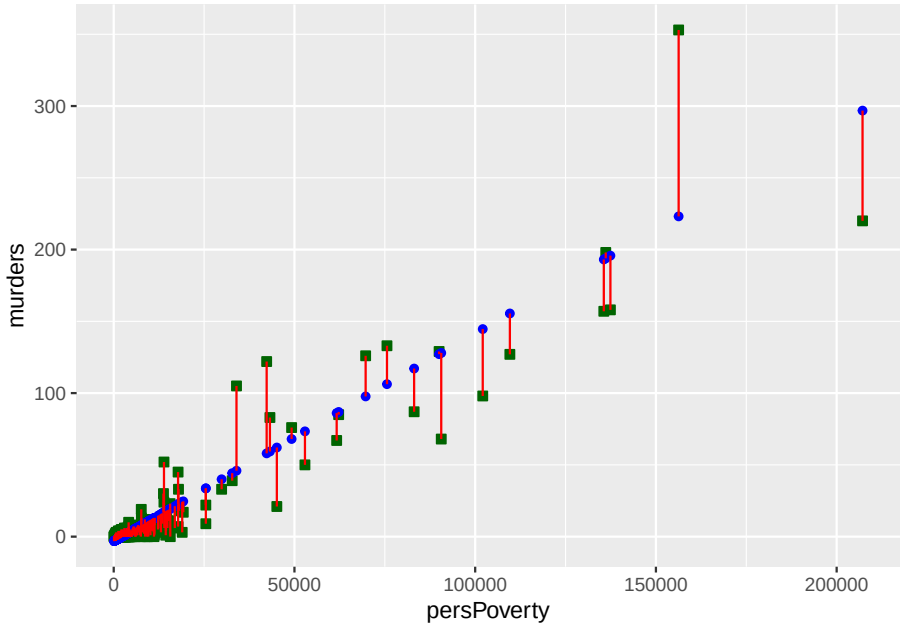


Figure 1.1: Les segments en rouges représentent les résidus, les carrés verts les y non entrainés qui ont servi au test et les points bleus représentent les y prédits à partir de notre modèle.

On peut aussi visualiser la répartition des résidus du modèle `lmSimple` autour de leur moyenne 0.

```
plot(lmSimple$residuals)
```

1.3 Application de la régression linéaire multiple

Dans cette partie on peut s'intéresser à la relation entre le nombre de meurtres et plusieurs autres variables. Un modèle de régression linéaire multiple pourrait faire l'affaire. Comme dans le cas de la régression linéaire simple, on va commencer par étudier la corrélation entre les variables explicatives. Nous allons choisir dans notre exemple, les variables qui sont corrélées avec `murders` comme varioles indépendantes.

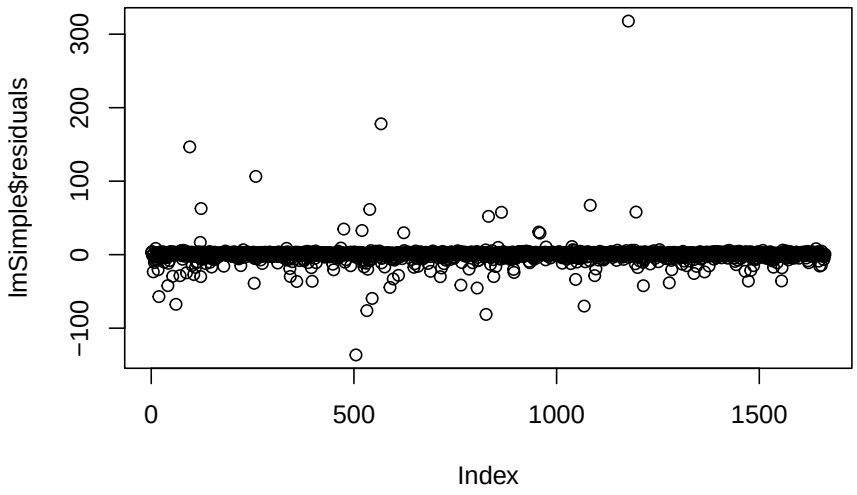


Figure 1.2: On constate une répartition des résidus autour de 0

Ces dernières ne doivent pas être parfaitement corrélées entre elles. En effet dans la solution estimée du coefficient β, on devra inversé la matrice X des variables explicatives. De ce fait une corrélation impliquera que la matrice ne soit pas de rang plein, donc non inversible.

$$\hat{\beta} = (X'X)^{-1}(X'y)$$

Commençons par selectionner nos variables pour la régression linéaire multiple.

```
library(dplyr)
lmMultpleDF = Communities %>% select(murders,pop, persUrban, persPoverty,
                              kidsBornNevrMarr, numForeignBorn,
                              houseVacant, persEmergShelt, persHomeless)
```

Entraînement de modèle

Le langage R permet d'entraîner le modèle linéaire multiple grâce à la fonction `lm()`. Pour indiquer à la fonction que nous sommes dans le cas d'une régression multiple, l'argument `formula()` doit recevoir `y~X1+X2+...+XP` et pour notre exemple `murders~pop+persUrban+....` Lors qu'on précise l'argument data de la fonction lm et que les données ne contiennent que les variables à étudier,

l'argument formula peut dans ce cas recevoir juste `y~.`. En Pratique, pour notre jeu de données `lmMultpleDF`, voici le code approprié :

```
# train test split
lmMultiple_Test = lmMultpleDF[echantillon,]
lmMultiple_Train = lmMultpleDF[-echantillon,]
# Model training
lmMultiple = lm(formula =  murders~.,data = lmMultiple_Train)
summary(lmMultiple)
```

```
##
## Call:
## lm(formula = murders ~ ., data = lmMultiple_Train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -109.924   -1.099    0.412    1.206  175.748
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -4.318e-01  3.698e-01  -1.168 0.243166
## pop              -1.096e-04  3.312e-05  -3.310 0.000953 ***
## persUrban         1.071e-05  2.910e-05   0.368 0.713030
## persPoverty       2.131e-04  6.680e-05   3.190 0.001452 **
## kidsBornNevrMarr  3.119e-03  1.196e-04  26.080  < 2e-16 ***
## numForeignBorn    3.743e-04  1.809e-05  20.695  < 2e-16 ***
## houseVacant       1.824e-03  1.338e-04  13.629  < 2e-16 ***
## persEmergShelt    6.397e-03  1.853e-03   3.452 0.000571 ***
## persHomeless     -4.637e-02  4.110e-03 -11.282  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.41 on 1653 degrees of freedom
## Multiple R-squared:  0.9747, Adjusted R-squared:  0.9746
## F-statistic:  7972 on 8 and 1653 DF,  p-value: < 2.2e-16
```

La sortie de la fonction `summary()` indique qu'on peut se passer de la variable `persUrban` dans l'explication de `murders` par un modèle linéaire. En effet, son coefficient β pourrait être nul car les $p-values$ est supérieure à $\alpha = 5\%$.

```
library(dplyr)
## Train test split
lmMultpleDF = lmMultpleDF %>% select(-persUrban)
lmMultiple_Test = lmMultpleDF[echantillon,]
```

```
lmMultiple_Train = lmMultpleDF[-echantillon,]
# Model training
lmMultiple = lm(formula =  murders~.,data = lmMultiple_Train)
summary(lmMultiple)
```

```
##
## Call:
## lm(formula = murders ~ ., data = lmMultiple_Train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -110.080   -1.137    0.422    1.222  175.907
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -5.027e-01  3.155e-01  -1.593 0.111255
## pop              -9.813e-05  1.092e-05  -8.983  < 2e-16 ***
## persPoverty       2.098e-04  6.619e-05   3.170 0.001555 **
## kidsBornNevrMarr  3.122e-03  1.194e-04  26.150  < 2e-16 ***
## numForeignBorn    3.739e-04  1.804e-05  20.720  < 2e-16 ***
## houseVacant       1.822e-03  1.337e-04  13.630  < 2e-16 ***
## persEmergShelt    6.371e-03  1.851e-03   3.441 0.000593 ***
## persHomeless     -4.644e-02  4.105e-03 -11.312  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.41 on 1654 degrees of freedom
## Multiple R-squared:  0.9747, Adjusted R-squared:  0.9746
## F-statistic:  9116 on 7 and 1654 DF,  p-value: < 2.2e-16
```

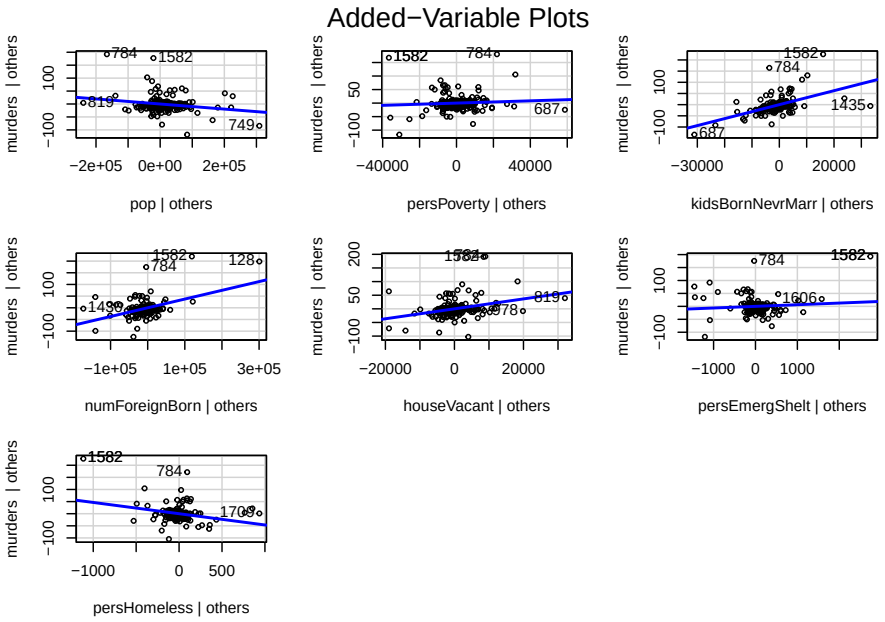
De plus, le coefficient d'ajustement R^2 est de 0.9747, soit un score de 97.47% pour notre modèle ce qui est un très bon résultat.

Graphes de régression

Lorsque le nombre de variables explicatives dépasse 2, il est impossible de représenter sur un même graphes le nuage de points formé par y. En effet la dimension physique maximale est de 3. Il existe plusieurs packages qui donnent des représentations assez significatives en deux dimensions du modèle linéaire multiple tel que `car`[1].
La représentation de la droite de régression peut se faire sur chaque dimension des variables explicatives grâce à la fonction `avPlots()` de la librairie `car`.

[1] package car : https://cran.r-project.org/web/packages/car/index.html

```
library(car)
avPlots(lmMultiple)
```



Prédiction

La prédiction dans le modèle linéaire multiple se fait aussi avec la fonction `predict()` comme dans le cas simple. On va utiliser notre modèle déjà entraîné avec 75% de notre jeu de données `lmMultpleDF` et les 20% pour effectuer une prédiction. Cela peut nous permettre de voir les résidus entre les valeurs prédites et les valeurs réelles de y.

```
y_hat = predict(lmMultiple,lmMultiple_Test[,-1])
head(y_hat,10)
```

```
##       2037        174        210        683        821        519       1593
##  0.2445685 19.3517750  1.4280286  5.4097354  7.8093372  0.8155394 -1.6654378
##       1098       1266       1153
##  1.5812592 -1.1213912 -0.7649363
```

1.3.1 Graphe des résidus

On peut visualiser les résidus entre les variables y_i observés et les $\hat{y}_i$ prédites :

```
y_test = lmMultiple_Test$murders
ggplot() +
  geom_point(aes(x = y_test, y = y_hat)) +
  geom_abline(slope = 1, color ='blue') +
  geom_segment(aes(x =y_test ,
                   y = y_test, xend = y_test, yend = y_hat),
               color = 'red') +
  ylab("Predicted murders")+xlab("Murders")
```

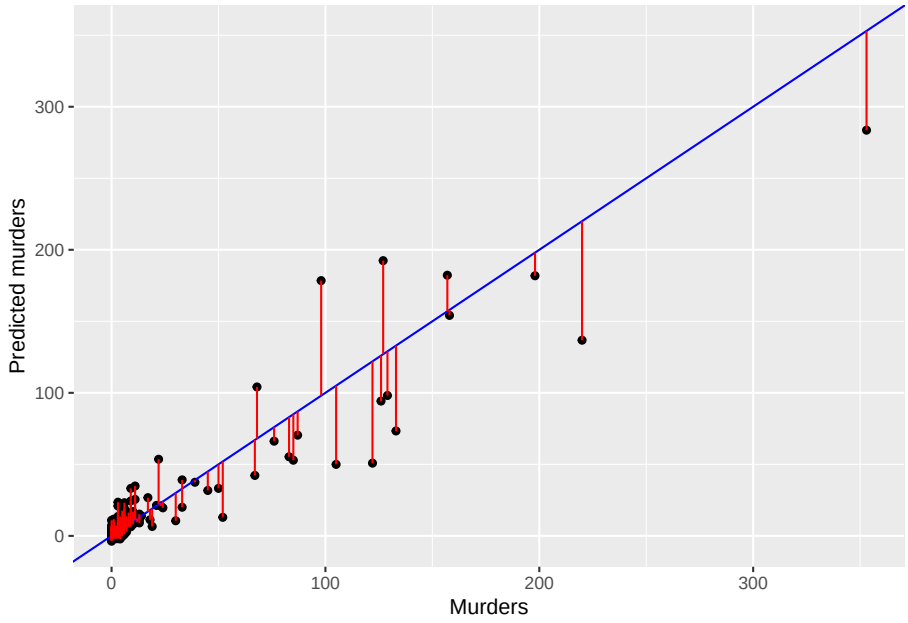


Figure 1.3: Les résidus sont assez proches de 0 cela reflète la bonne qualité du modèle.

Chapter 2

Principal Component Analysis and Factor Analysis

Nous allons effectuer une Analyse en Composantes Principales(ACP) pour la variables numériques de notre jeu de données `Communities`

Sélection des variables numériques

```
library(dplyr)
PCA_df <- Communities %>% select_if(is.numeric)
```

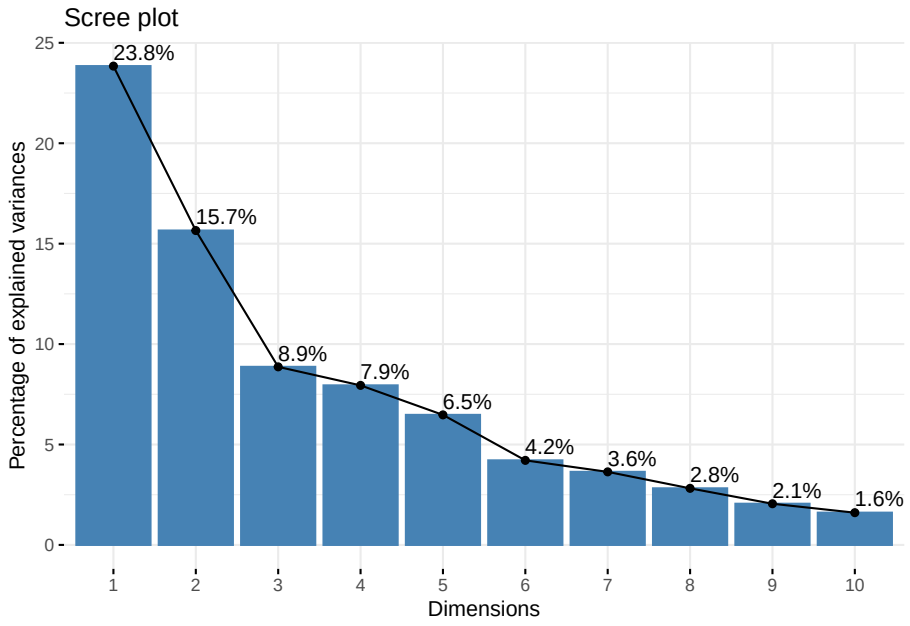
Application de la fonction PCA

L'objectif est de réduire la dimension de notre jeu de données `PCA_df` tout en conservant un pourcentage élevé des informations.

```
Communities_PCA <- FactoMineR::PCA(PCA_df,scale.unit=TRUE,
                                   graph = FALSE)
```

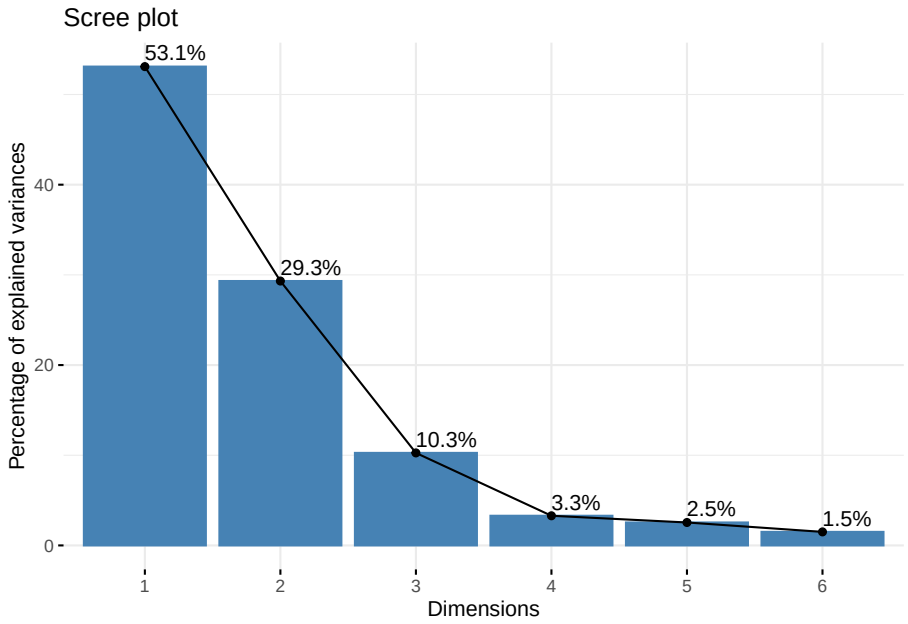
Pourcentage d'explication

```
factoextra::fviz_eig(Communities_PCA,addlabels = TRUE)
```



Nous constatons que deux dimensions sont insuffisantes pour représenter 75% des informations. Pour y remédier nous allons diminuer le nombre de variables de `PCA_df`(6 par exemple).

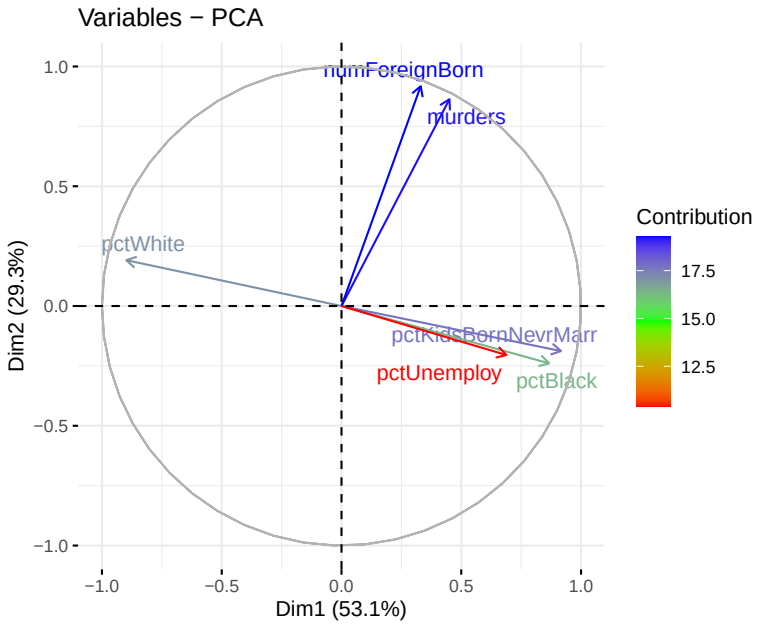
```
library(dplyr)
PCA_df = PCA_df %>% select(murders,pctWhite,pctBlack,
                           pctKidsBornNevrMarr,
                           numForeignBorn,pctUnemploy)
Communities_PCA <- FactoMineR::PCA(PCA_df,scale.unit=TRUE,
                                   graph = F)
factoextra::fviz_eig(Communities_PCA,addlabels = TRUE)
```



Ce graphe nous permet de remarquer que deux composantes principales sont suiffisantes pour représenter 82.4% de l'information ce qui est supérieur à 75% notre pourcentage seuil.

Graphiques de corrélation des variables

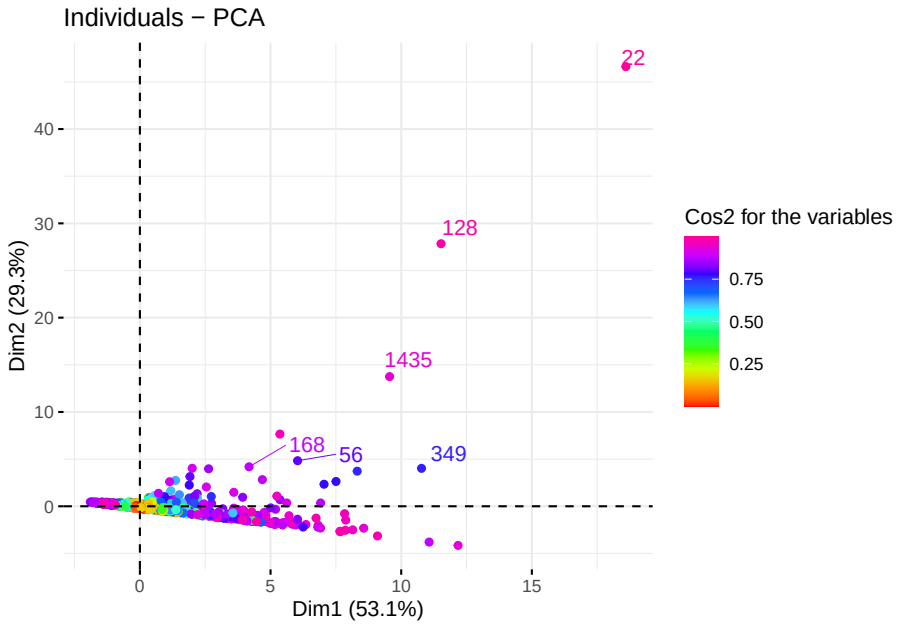
```
factoextra::fviz_pca_var(Communities_PCA,
             col.var = "contrib",
             gradient.cols = rainbow(3),
             repel = TRUE,
             legend.title='Contribution')
```



À partie du graphe précédent, nous pouvons constater une forte corrélation entre le pourcentage de personnes sans emplois `pctUnemploy`, d'enfants nés hors mariage `pctKidsBornNevrMarr` et de personnes de la communauté noire américaine `pctBlack`. Ces dernières sont peu corrélées avec `murders`.

Graphes des individus

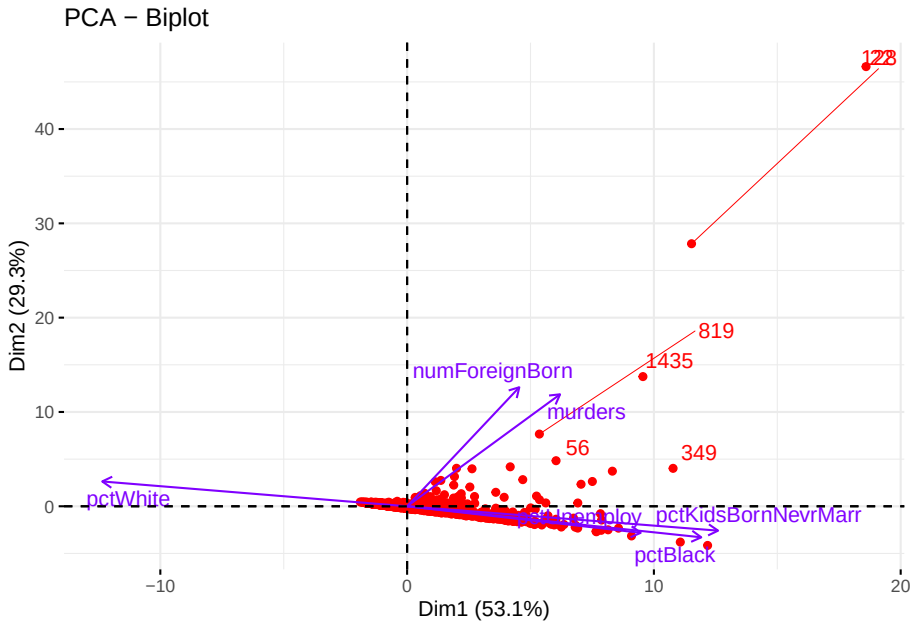
```
factoextra::fviz_pca_ind(Communities_PCA,
                 col.ind ='cos2',
                 gradient.cols = rainbow(10),
                 repel = TRUE,
                 legend.title = "Cos2 for the variables",
                 )
```



Une variable avec un score élevé suivant un axe signifie qu'elle a fortement contribué à la création de cet axe. Une variable moyenne c'est celle qui est proche de l'origine.

Biplot individus et variables

```
factoextra::fviz_pca_biplot(Communities_PCA, repel = TRUE,
                col.var = rainbow(4)[4],
                col.ind = rainbow(1)
                )
```



Prenons la la variable numéro 819 (C'est à dire la ligne 819 de `PCA_df`). Nous savons que les variables {murders} et {numForeignBorn} sont fortement corrélées. Alors si le nombre de meutres {murders} est très faible par rapport à la moyenne, alors le nombre de personnes nées à l'étranger{numForeignBorn} l'est aussi.

Un raisonnement similaire peut être fait pour les variable fortement et négativement corrélées comme le pourcentage de personnes sans emplois `pctUnemploy` et de personnes de la communauté blanche américaine `pctWhite` sauf que dans ce cas, les variables varieront dans de sens opposés. Nous pouvons vérifier ces informations tirées sur le graphes précédent dans notre data set grâce à la cellule de code suivante.

```
library(dplyr)
variable819 <- rbind(PCA_df[819,],apply(PCA_df,2,mean),
                     apply(PCA_df,2,min),
                     apply(PCA_df,2,max)
                     ) %>%  dplyr::select(murders,numForeignBorn) %>%
                       `rownames<-`(.,c('Variable819','mean','min','max'))
variable819
```

```
##                murders numForeignBorn
## Variable819  446.000000     290374.000
## mean           7.764786       6277.274
```

```
## min            0.000000         20.000
## max         1946.000000    2082931.000
```

Chapter 3

L'analyse de la variance

Dans cette section on va s'intéresser aux meurtres commis par unité de gang déployé `gangUnit`. Pour cette dernière, l'encodage est le suivant : 0 signifie "Meaning", 10 signifie "Oui" et 5 signifie temps partiel("Part-Time").

Sélection des données

On va sélectionner les données qui serviront l'ANOVA puis ajjouter une nouvelle collonne pour le décodage de `gangUnit`.

```
library(dplyr)
aov_data = Communities %>% select(murders,gangUnit) %>%
          mutate(GangUnit_means = case_when(gangUnit=="0"~"Non",
                                            gangUnit=="10"~"Oui",
                                            gangUnit=="5"~"Part-Time",
                                            gangUnit=="?"~NA_character_))
```

Les boites à moustaches

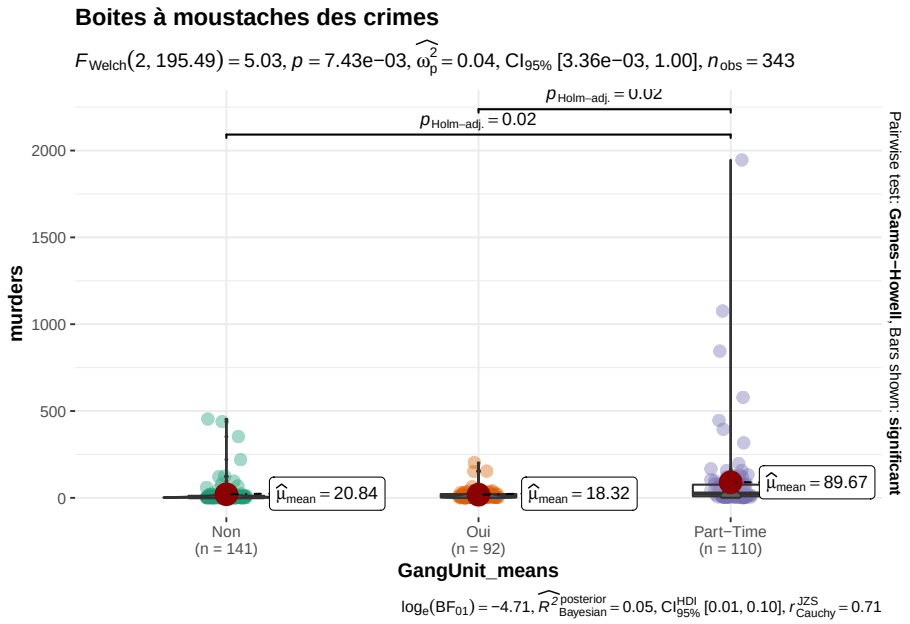
Le premier travail à faire lors d'une ANOVA est la représentation des boites à moustaches. Nous allons utliser le packages *ggstatsplot*[1] qui donne une sortie avec plusieurs informations.

```
library(ggstatsplot)
```

[1] https://indrajeetpatil.github.io/ggstatsplot/

```
## You can cite this package as:
##      Patil, I. (2021). Visualizations with statistical details: The 'ggstatsplot' ap
##      Journal of Open Source Software, 6(61), 3167, doi:10.21105/joss.03167
```

```
ggbetweenstats(
  data = aov_data,
  x     = GangUnit_means,
  y     = murders,
  title = "Boites à moustaches des crimes"
)
```



Dû à des valeurs manquantes, notre représentation porte sur 343 observations. Les boites à moustache indiquent qu'en moyenne, le nombre de meurtres varie lors que `gangUnit` change de modalité. Passons à l'anova pour en savoir plus.

Application de l'anova à un facteur

Le logiciel **R** nous permet de faire l'analyse de la variance grâce à la fonction *aov*().

```
murders_aov = aov(murders~GangUnit_means,data = aov_data)
summary(murders_aov)
```

```
##                Df  Sum Sq Mean Sq F value   Pr(>F)
```

```
## GangUnit_means   2  364694  182347   9.377 0.000109 ***
## Residuals      340 6611431   19445
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 1872 observations deleted due to missingness
```

Comme le laissaient paraître les boxplots, d'après le tableau précédent, la $p - value = 0.000109 < \alpha = 5\%$, alors l'effet de gang(`gangUnit`) sur le nombre de crimes est significatif.

References

U. S. Department of Commerce, Bureau of the Census, Census Of Population And Housing 1990 United States: Summary Tape File 1a & 3a (Computer Files),

U.S. Department Of Commerce, Bureau Of The Census Producer, Washington, DC and Inter-university Consortium for Political and Social Research Ann Arbor, Michigan. (1992)

U.S. Department of Justice, Bureau of Justice Statistics, Law Enforcement Management And Administrative Statistics (Computer File) U.S. Department Of Commerce, Bureau Of The Census Producer, Washington, DC and Inter-university Consortium for Political and Social Research Ann Arbor, Michigan. (1992)

U.S. Department of Justice, Federal Bureau of Investigation, Crime in the United States (Computer File) (1995)