

Projet STA203 Cortambert - Riou

Pierre Cortambret - William Riou

28/04/2021

Introduction

Le but de cette analyse est de créer un modèle qui permet de déterminer la teneur en sucre des cookies par spectrométrie. Le spectre s'étend sur 700 fréquences (variables explicatives) sur 72 individus (cookies). En travaillant sur ce jeu de données, les méthodes de statistique inférentielle rencontrent rapidement les limites des jeux de données de taille importante. Il va falloir déterminer quelques variables explicatives dont le *cos2* est important pour pouvoir modéliser plus facilement le problème.

1 - Un peu de théorie

2 - Analyse exploratoire

1 - Mise en forme des données

```
rm(list=objects())
graphics.off()
setwd("~/OneDrive/Cours/2020-2021/08-STA203/PROJET")
```

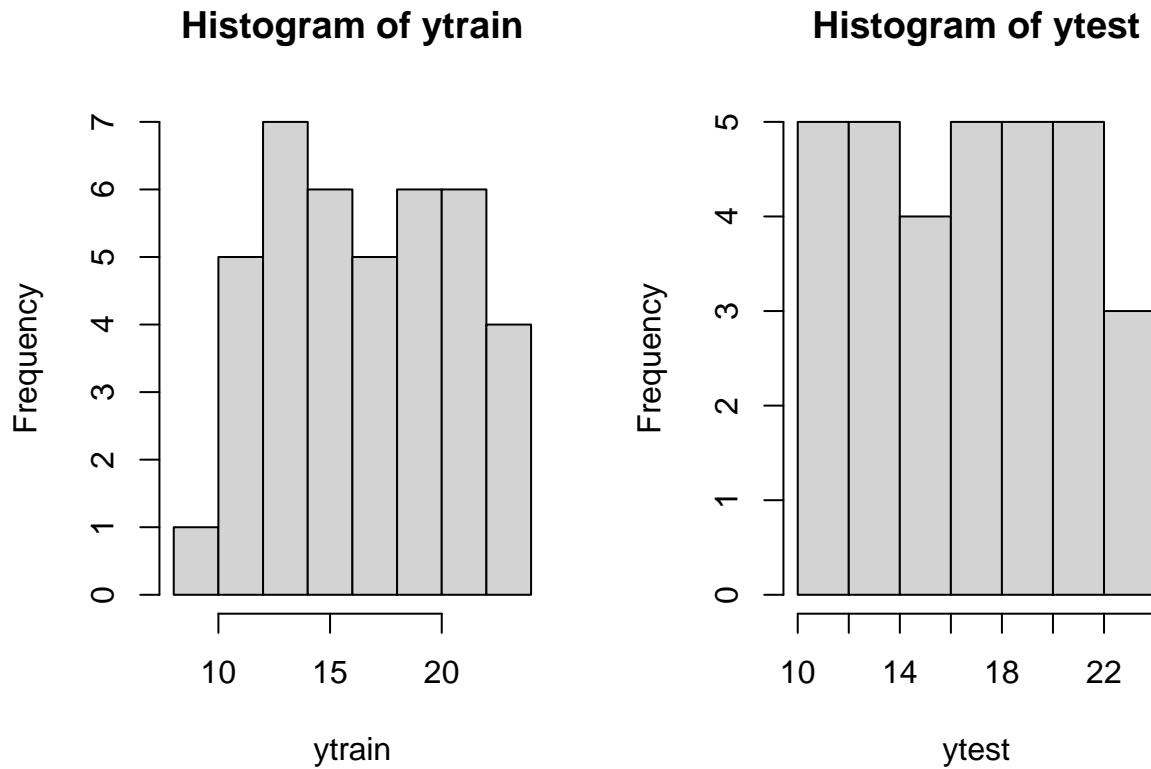
Créons le jeu d'apprentissage

```
load("cookie.app.RData")
n <- nrow(cookie.app)
p <- ncol(cookie.app)
xtrain <- data.frame(cookie.app,nrow=n,ncol=p)
ytrain <- xtrain[,1]
xtrain <- xtrain[,-1]
```

Créons maintenant le jeu de test:

```
load("cookie.val.RData")
n1 <- nrow(cookie.val)
p1 <- ncol(cookie.val)
xtest <- data.frame(cookie.val,nrow=n1,ncol=p1)
ytest <- xtest[,1]
xtest <- xtest[,-1]
```

```
par(mfrow=c(1,2))
hist(ytrain)
hist(ytest)
```



2 - Analyse par composantes principales

```
par(mfrow=c(2,2))

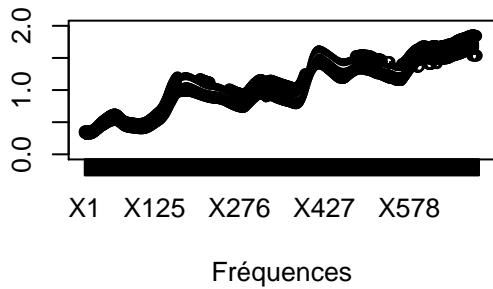
boxplot(xtrain,main="Représentation en boxplot \n du jeu d'apprentissage",xlab="Fréquences",ylim=c(0,2))

matplot(t(xtrain),main="Représentation du \n jeu d'apprentissage",xlab="Fréquences",ylab = "xtrain",ylim=c(0,2))

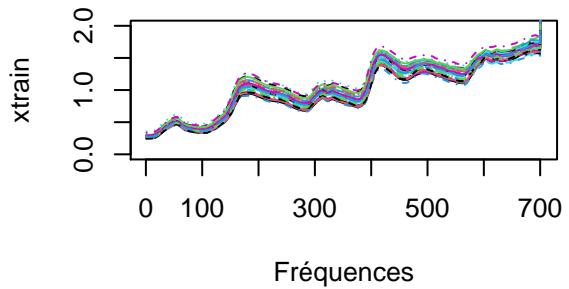
boxplot(t(xtrain),main="Représentation en boxplot \n du jeu d'apprentissage",xlab="Cookies",ylim=c(0,2))

matplot(xtrain,main="Représentation du \n jeu d'apprentissage",xlab="Cookies",ylab = "xtrain",ylim=c(0,2))
```

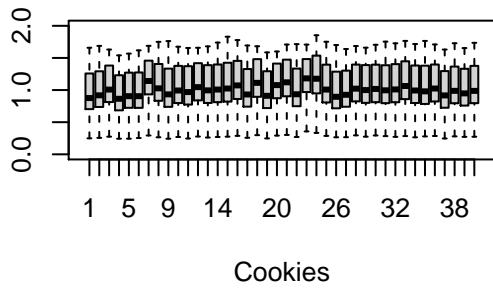
**Représentation en boxplot
du jeu d'apprentissage**



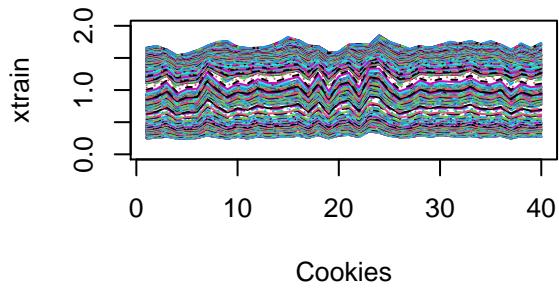
**Représentation du
jeu d'apprentissage**



**Représentation en boxplot
du jeu d'apprentissage**



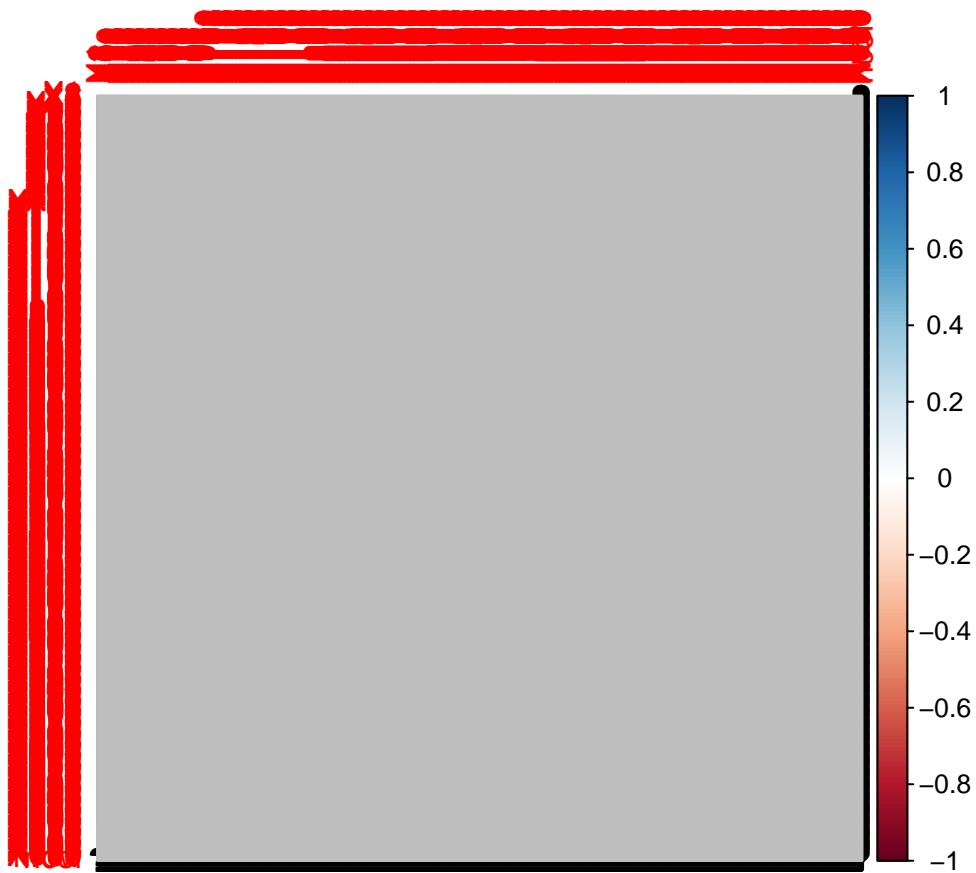
**Représentation du
jeu d'apprentissage**



```
X <- scale(xtrain, scale = TRUE, center = TRUE)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
C <- cor(X)
corrplot(C)
```



Nous constatons que l'étude n'est pas faisable avec 700 variables explicatives. Il faut procéder préalablement à une analyse par composantes principales.

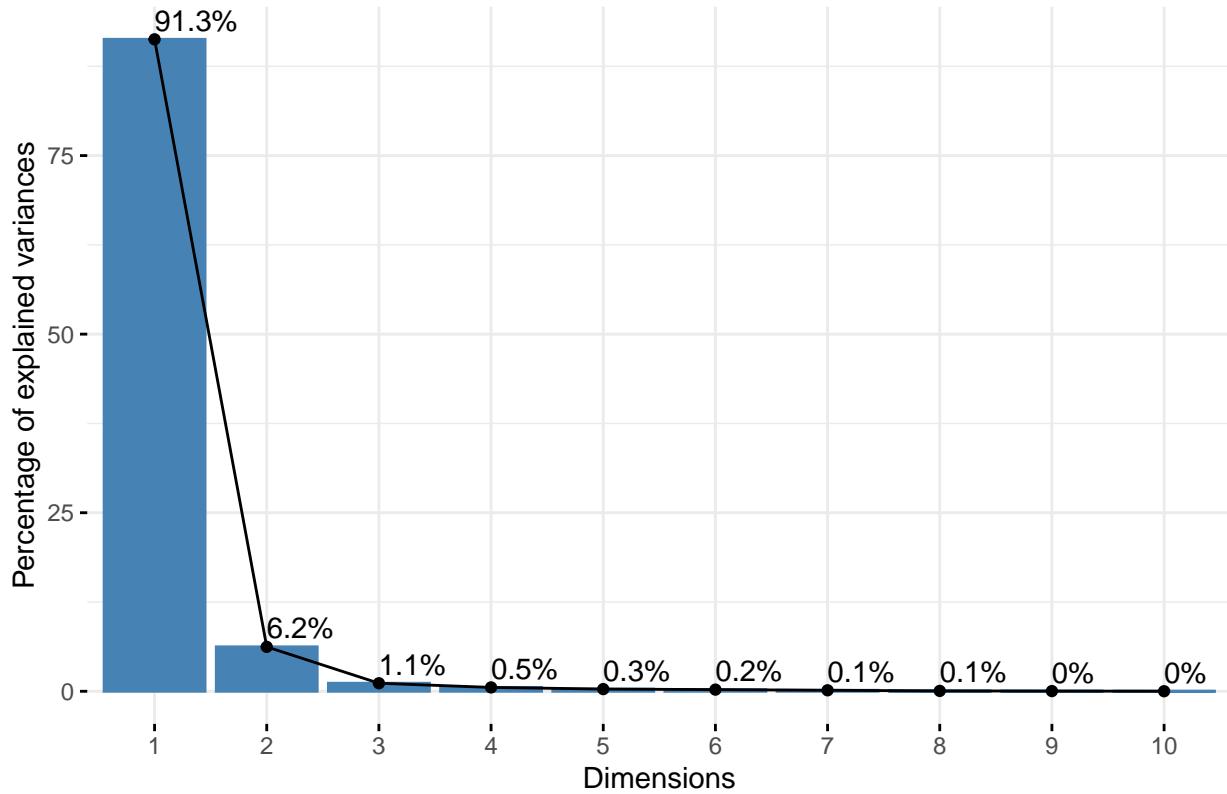
```
library(FactoMineR)
library(factoextra)

## Loading required package: ggplot2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

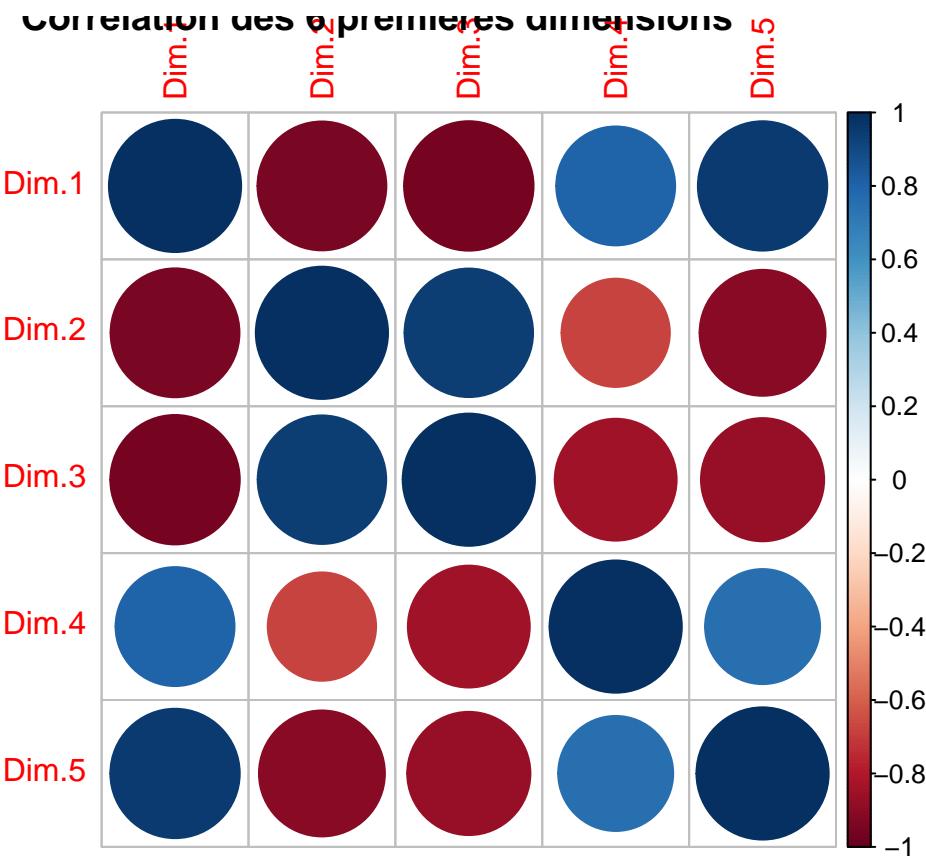
pca.res <- PCA(xtrain,graph = FALSE, scale.unit = TRUE)
fviz_eig(pca.res,main = "Histogramme des valeurs propres de l'analyse par composantes principales",addl
```

Histogramme des valeurs propres de l'analyse par composantes principales



Nous pouvons retenir 2 voire 3 dimensions. Refaisons maintenant l'analyse préliminaire avec les 6 premières dimensions.

```
C <- cor(pca.res$var$coord[1:6,])
corrplot(C, main = "Corrélation des 6 premières dimensions")
```

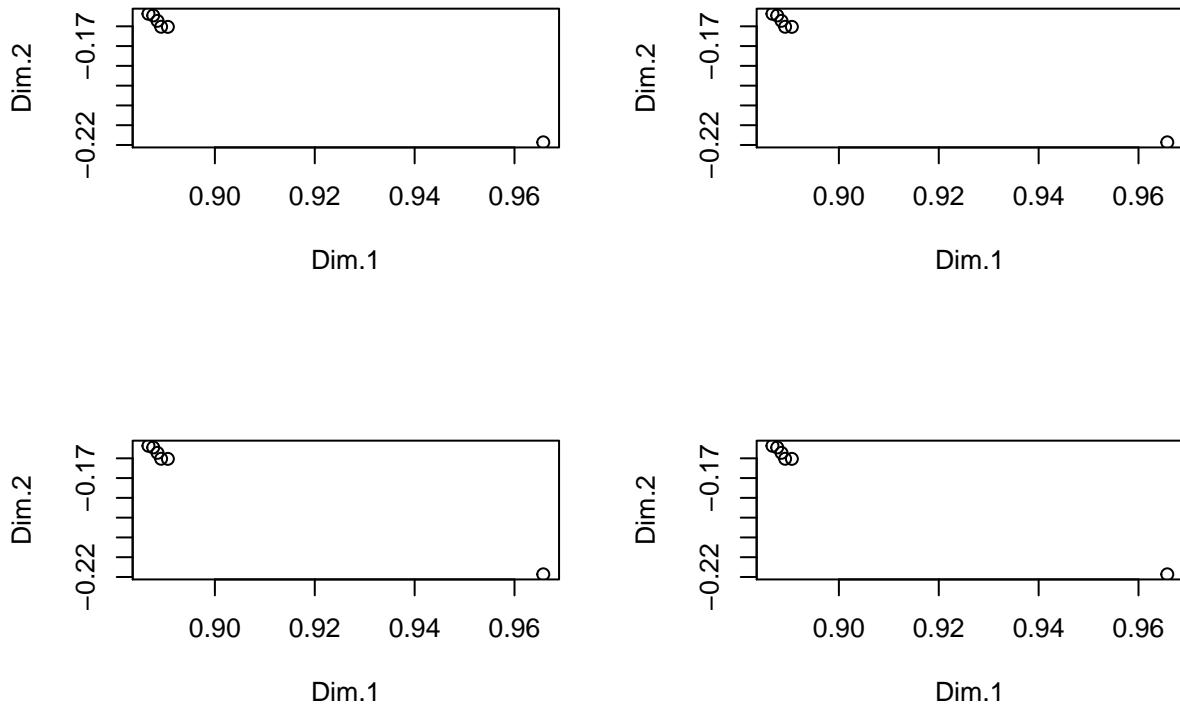


```
X <- rbind(pca.res$var$coord[1:5,] ,pca.res$var$coord[39,])
par(mfrow=c(2,2))
plot(X)
plot(X,axes = c(2,3))

## Warning in if (axes) { : la condition a une longueur > 1 et seul le premier
## élément est utilisé

## Warning in if (frame.plot) localBox(...): la condition a une longueur > 1 et
## seul le premier élément est utilisé

plot(X)
plot(X)
```



```
#plot(pca.res, choix = "ind", axes = c(5,6), main="Représentation des individus sur les axes principaux 5
```

3 - Codage de la fonction reconstruct

```
Xm <- apply(xtrain , 1 , mean)
Xsd <- apply(xtrain , 1 , sd)
reconstruct <- function(res,nr,Xm,Xsd){
  PCA(res,axes = 1:nr)
}
#reconstruct(pca.res,6,Xm,Xsd)

library(caret)

## Loading required package: lattice

par(mfrow=c(2,3))
y <- c(1,2,3,4,5,39)
indi = 24
xtrain[indi,y]

##          X1          X2          X3          X4          X5          X39
## 24 0.333249 0.333136 0.333193 0.333362 0.333193 0.515623
```

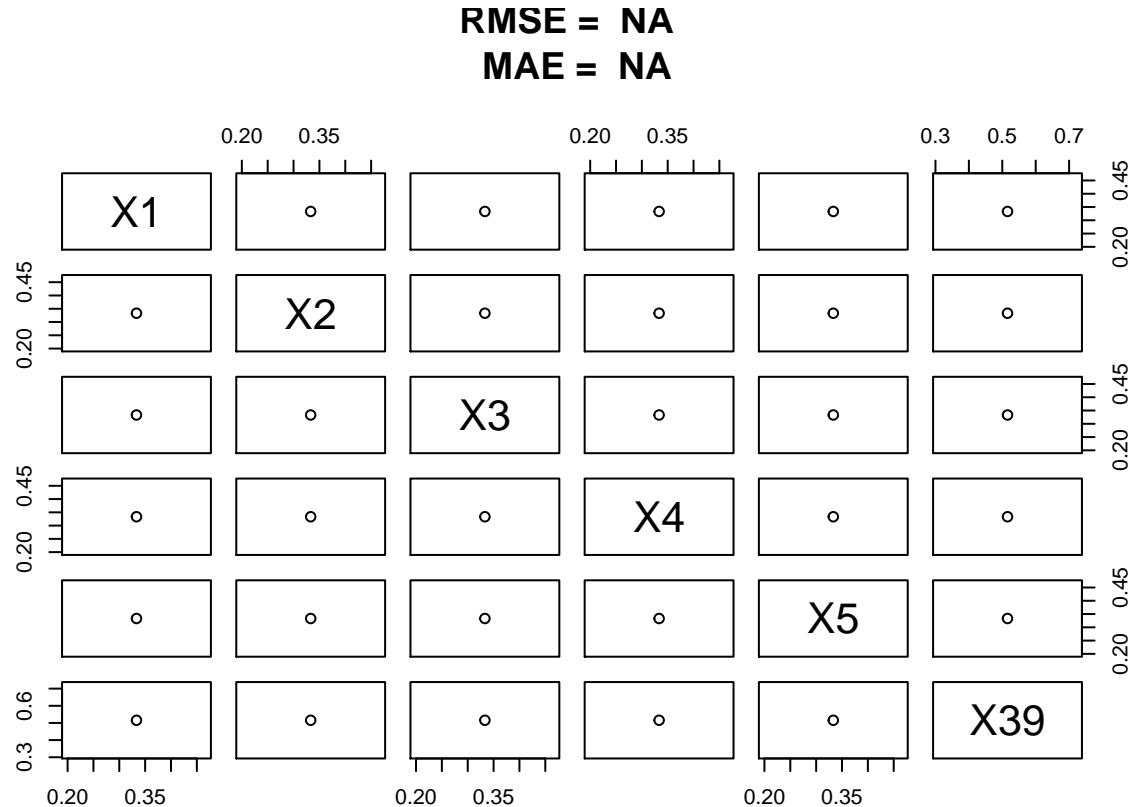
```

plot(xtrain[indi,y],main = paste("RMSE = ",c(RMSE(xtrain[indi,y],ytrain[indi])),"\n MAE = ",c(MAE(xtrain[indi,y],ytrain[indi])))

## Warning in mean.default((pred - obs)^2, na.rm = na.rm): argument is not numeric
## or logical: returning NA

## Warning in mean.default(abs(pred - obs), na.rm = na.rm): argument is not numeric
## or logical: returning NA

```



3 - Régression pénalisée

1 -

```
#lambda de 10^6 à 10^-10
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-1
```

```
gridRidge <- 10^seq(6, -10, length=100)
```

lambda=0 : pas de Ridge regression penalty, revient à minimiser Somme des Carrés Résiduels
lambda=+inf : Ridge regression penalty max, séparation des données avec la moyenne de tous les variables explicatives

```
x <- cbind(ytrain,xtrain)
xRidge_train <- model.matrix(ytrain~., data=x)
#ridge_fit <- glmnet(xRidge_train, ytrain, alpha=0, lambda=gridRidge, family="binomial")
#plot(ridge_fit)
```

4 - Régression logistique pénalisée

1 -

L'expérience est modélisée comme la réalisation de n variable aléatoires indépendantes Y_i de loi de Bernoulli d'espérance $\pi(x_i) = \mathbb{P}(Y_i = 1; x_i) : Y_i \rightarrow \mathcal{B}(1, \pi(x_i))$

telle que $logit(\pi(x_i)) = \log\left(\frac{\pi(x_i)}{1-\pi(x_i)}\right) = x_i\theta$ La variable aléatoire Y suit une loi binomiale $\mathcal{B}(1, \pi(x_k))$

```
z <- ifelse(ytrain>18,1,0)
ztest <- ifelse(ytest>18,1,0)
```