# Numerical Calibration of the Lasso

ENS Advanced Math, Non parametrics

Franck Picard, Fall 2020

## Preliminaries

Groups of students are asked to send an `R markdown` report generated via `R studio` to franck.picard@ens-lyon.fr at the end of the tutorial. You will need `Rstudio`, LaTeX and packages for markdown:

```
library(knitr)
library(rmarkdown)
library(graphics) ##Necessary for plots
library(pracma)
library(glmnet)
```

This report should answer the questions by commentaries and codes generating appropriate graphical outputs. A cheat sheet of the markdown syntax can be found here.

## 1 Regression model

This project aims at studying the empirical properties of the LASSO based on simulated data. The statistical framework is the linear regression model, such that

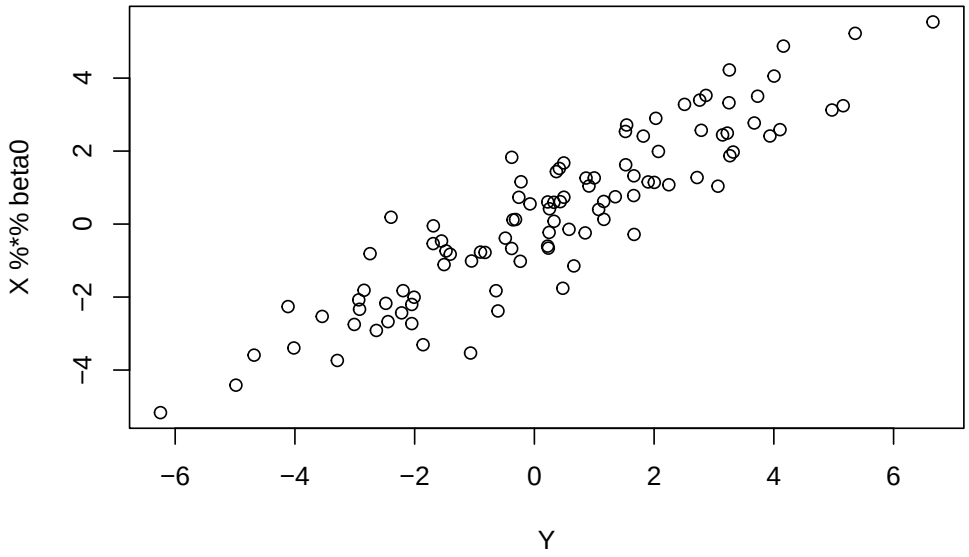
$$Y_i = x_i^T \beta^* + \varepsilon_i, \ \varepsilon_i \sim \mathcal{N}(0, \sigma^2),$$

with $Y$ a vector in $\mathbb{R}^n$, and $\beta^*$ a vector in $\mathbb{R}^p$ with $p_0$ non-null elements. In the following $J_0 = \{j \in \{1, ..., p\}, \beta_j^* \neq 0\}$. For simplification, we will consider that there is only one distinct non null value in $\beta^*$: $\beta^* = \beta_0^* \times (1, ..., 1, 0, ...0)$.

## 2 Simulation of observations

```
n       = 100
p       = 10
p0      = 5
sigma   = 1
sigmaX  = 1
b0      = 1
beta0   = c(rep(b0,p0),rep(0,p-p0))
X       = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
```

```
Y       = X%*%beta0 + rnorm(n,0,sigma)

plot(Y,X%*%beta0)
```




## 3 Lasso and the `glmnet` R-package

In practice, model parameters are estimated using the `glmnet` R-package to compute the lasso estimator

$$\widehat{\beta}_\lambda = \min_{\beta_0,\beta} \frac{1}{N} \sum_{i=1}^{N} w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda \left[ (1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1 \right],$$

with $\alpha = 1$ for the Lasso, $\alpha = 0$ for Ridge Regression and $\alpha \in ]0,1[$ for Elastic Net. In a first step you are invited to check the online documentation of the package that is very complete. Then the purpose of calibration is to determine the value of the hyperparameter $\lambda$ based on the observations.

```
library(glmnet)
n       = 100
p       = 10
p0      = 5
sigma   = 1
sigmaX  = 1
b0      = 1
beta0   = c(rep(b0,p0),rep(0,p-p0))
X       = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
Y       = X%*%beta0 + rnorm(n,0,sigma)
```

```
fit = glmnet(X, Y)
plot(fit,label=TRUE)
names(fit)
print(fit)
coef(fit)
```

## 4 Numerical Calibration in practice

Parameter $\lambda$ is chosen by cross validation using `cv.glmnet` such that:

```
library(glmnet)
n       = 100
p       = 10
p0      = 5
sigma   = 1
sigmaX  = 1
b0      = 1
beta0   = c(rep(b0,p0),rep(0,p-p0))
X       = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
Y       = X%*%beta0 + rnorm(n,0,sigma)

lambda.cv = cv.glmnet(X,Y, family = "gaussian",intercept=F)$lambda.1se
bh        = glmnet(X,Y,family = "gaussian",intercept=F,
                   lambda=lambda.cv)$beta
if ( sum(abs(bh))==0 ) {bh = rep(0,p)}
bh        = as.vector(bh)
```

The first part of your projet will be to re-implement the cross validation procedure, and to verify that your implementation is correct based on the `cv.glmnet` function that will be used to check your results. You will also implement the calibration of $\lambda$ based on the AIC and on the BIC.

## 5 Simulations setting

The performance of the lasso depends on different factors, and numerical simulations are used to study the impact of these factors on the capacity of the lasso to select the dimension of the model. Among those factors, we can identify $n$ (number of observations), $p$ (dimensionality), $p_0$ and $\beta^*$ (strength of the signal), $\sigma$ (strength of noise). Studying the impact of all factors would not be realistic, so we focus on:

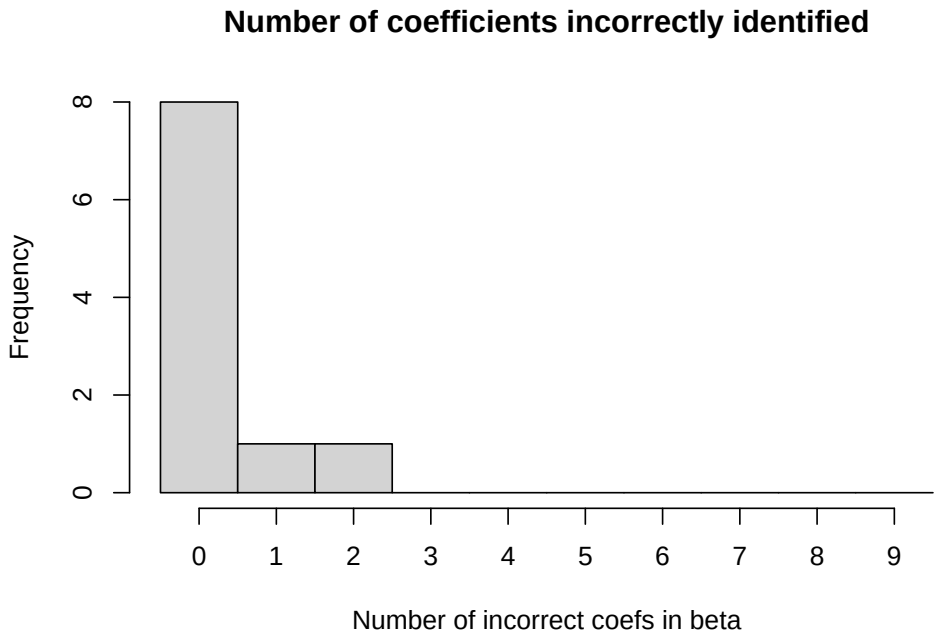
- $n/p$ will be chosen so that we study the "low-dimensional regime" as well as the "high-dimensional regime".
- $p_0$ will be fixed at 5
- $\beta_0^* = 1$
- $\sigma$ will vary.

As an indicator of performance, we will study only the dimension of the selected model, ie $\hat{s} = \sum_{j=1}^{p} 1_{\hat{\beta}_j \neq 0}$. In order to conduct your simulations, you will start

from the following example code:

```
n      = 100
p      = 10
p0     = 5  #Number of nonzero coefficients
sigma  = 1
sigmaX = 1
b0     = 1
beta0  = c(rep(b0,p0),rep(0,p-p0))
B      = 10 #Number of simulations
res    = matrix(NA,ncol=5,nrow=B) #Stores n,p, sigma, number of nonzero
                  #coefficients recovered, number of nz incorrectly identified

betah  = matrix(NA, ncol = p, nrow = B)

# fixed design setting
X = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})

for (b in 1:B){
  Y     = X%*%beta0 + rnorm(n,0,sigma)
  # estimate betah using the calibrated lasso
  # betah =
  lambda.cv = cv.glmnet(X,Y, family = "gaussian",intercept=F)$lambda.1se
  temp = glmnet(X,Y,family = "gaussian",intercept=F, lambda=lambda.cv)$beta
  ##the value returned by glmnet(...)$beta is a sparse matrix
  #In a sparse matrix : @i (non zero indices); @x non zero values

  indices_identified = c(rep(0, p)) #Fill indices_identified with 1 at
                                    #indices in temp@i, set others to 0
  for (index in temp@i) {
    indices_identified[index] = 1
  }

  betah[b,]  = c(temp@i, rep(NA, p-length(temp@i))) #We store in betah[b,]
    #the nonzero indices and we fill with NA to get a vector of the proper size
  res[b,] = c(n, p, sigma, length(temp@i), sum(abs(indices_identified - beta0)))
}
res           = as.data.frame(res)
colnames(res) = c("n","p","sigma","nz", "nzcorrect")


hist1<-hist(res$nz, breaks = 1:10, axes = FALSE,
            xlab="Number of non-zero coefs in beta",
            main = "Number of coefficients identified")
axis(side=1,at=hist1$mids,labels=1:9) #So that breakpoints are centered
axis(side=2)          #Because of previous trick, we had to hide both axis,
```


```
                    #so we display again the Y-axis


hist2 <- hist(res$nzcorrect, breaks = 0:10, axes = FALSE,
              xlab="Number of incorrect coefs in beta",
              main = "Number of coefficients incorrectly identified")
axis(side=1,at=hist2$mids,labels=0:9)
axis(side=2)
```




## 5.1 AIC

In this section, we implement the Akaike Information Criterion (AIC):

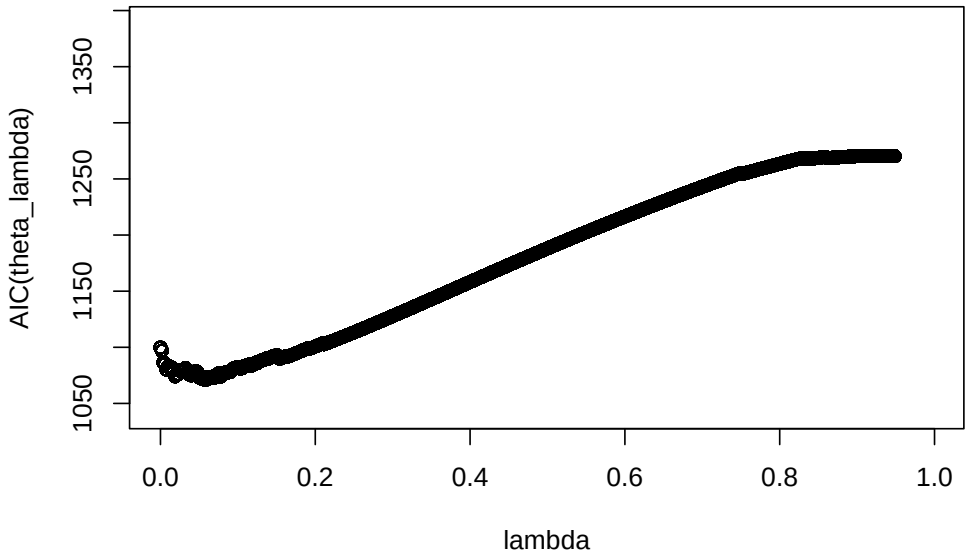
$$AIC = 2k - 2ln(\hat{L})$$

where $\hat{L}$ is the maximum value of the likelihood function for the model under consideration. The procedure consists of computing the Lasso with parameters $\lambda_1, \cdots, \lambda_N$.

```
likelihood <- function(betal, Yl, sigma2 = 1.0){
  S<-0.0
  S <- S + t(Yl-betal)%*%(Yl-betal)
  n<- length(Yl)

  return (1/sqrt(2 * pi* sigma2))**(n/2)*exp(-S/(2*sigma2))
}



n      = 200
p      = 150
p0     = 5  #Number of nonzero coefficients
sigma  = 1
sigmaX = 1
b0     = 1
beta0  = c(rep(b0,p0),rep(0,p-p0))
betah  = matrix(NA, ncol = p, nrow = B)
X = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
```

```r
Y     = X%*%beta0 + rnorm(n,0,sigma)



N <- 500 #Number of lambda tested
lambda_step <- 0.002 #Difference between two consecutive lambda
lambda_init <- 0.0 #First value used by lambda

lambda <- lambda_init
results_frame <- data.frame(
  lambda = c(),
  betah = c(),
  k = c(),
  loglikelihood = c(),
  AIC = c(),
  BIC = c()
)

for (b in 1:N){#The increment is lambda
  temp = glmnet(X,Y,family = "gaussian",intercept=F, lambda=lambda)$beta
  ##the value returned by glmnet(...)$beta is a sparse matrix
  #In a sparse matrix : @i (non zero indices); @x non zero values

  betah <- c(rep(0,p))
  j<-0
  for(i in temp@i){
    betah[i+1] <- temp@x[j+1]
    j<-j+1
  }

  #llik = log(((1/sqrt(2 * pi))**(p/2))*exp(-t(Y-betah)%*%(Y-betah)/(2)))
  RSS = t(Y-X%*%betah)%*%(Y-X%*%betah) #Simpler expression for log likelihood in Gauss
  new_row_df <- data.frame(
    lambda = lambda,
    betah = betah,
    k = j,
    negloglikelihood = -log(RSS),
    AIC = 2*j + n * log(RSS),
    BIC = NA
  )
  results_frame <- rbind(results_frame, new_row_df)

  lambda <- lambda+lambda_step

}
lambda_cv = cv.glmnet(X,Y, family = "gaussian",intercept=F)$lambda.1se  #We compute th
s <- c(rep(lambda_cv, 1000))  #And fill an axis with it
```

```
axis_lambda_cv <- xy.coords(x=s, y = 1:1000)  #that we will give to the plot

plot(x = results_frame$lambda, y = results_frame$AIC, type = "n", xlab = "lambda", yla
plot.xy(axis_lambda_cv, type = "l") #type = "l" to draw lines
plot.window(c(0.95*lambda_init, 1.05*lambda ), ylim = c(0.95*min(results_frame$AIC), 1
plot.xy( list(results_frame$lambda, results_frame$AIC, NULL, NULL), type="p")
```




We then define the best value for $\lambda$ with respect to AIC as:

$$\lambda_{AIC} = argmin_{\lambda_i} AIC.$$

Hence, for each $i$ we need to compute the likelihood of the model with parameter $\beta_{\lambda_i}$ and $k$ as the number of nonzero coefficients of $\beta_{\lambda_i}$. The likelihood, with Gaussianity assumption and covariance matrix $\sigma I$, is given by

$$\hat{L_{\lambda_i}} = \Pi_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}(Y_i - \beta)^t (Y_i - \beta)$$

. In our setting, we compute $S = \sum_i (Y_i - \beta_\lambda)^2$ and then

$$L_Y(\lambda) = (2\pi\sigma^2)^{-\frac{n}{2}} \exp(-\frac{1}{2\sigma^2} S)$$

For each instance of $Y = X\theta^* + \epsilon$ for given parameters $p/n$(dimensionality)\$ of $k = s * (p/n)$ (sparsity), while the others are fixed; we obtain lambda_AIC, lambda_CV, lambda_BIC,..... the optimal calibration of lasso for a same problem and a given criterion.

We want to know given the dimensionality and perhaps the sparsity which criterion is optimal. For each instance of $Y = X\theta^*$ we may define

$$\lambda_{opt} = Argmin_\lambda ||X\theta^* - X\hat{\theta}_\lambda||_2$$

the "optimal" value of lambda, in the sense that given the solution $\theta^*$ (unknown in practice), we look for the lambda that would give the model with the smallest error (where the argmin could run through all lambdas or only on the one given by our criterions). Note that the choice of definition of $\lambda_{opt}$ is arbitrary, replacing the norm by an $\ell^1, \ell^\infty, \cdots$ can be justified and the argmin could ignore the dependence on $X$.

Idea : modelling $\lambda_{opt}$ as a function of the dimensionality, the sparsity, the different criterions Naive idea: Define the norm in the argmin as some sup norm so that $hatlambda_{opt}$ is the criterion that gives the lambda of the criterion which is closest from sparsity estimate. Improved: A nearest-neighbour Improved (bis): Model $\hat{\lambda}_{opt}$ as some linear combination of $p/n, s, \lambda_{AIC}, \lambda_{BIC}, \ldots$. Improved (ter): Model as a polynomial/log linear of $p/n$ and $s$ (and $\lambda_{AIC}, \lambda_{Bic}, \cdots$)

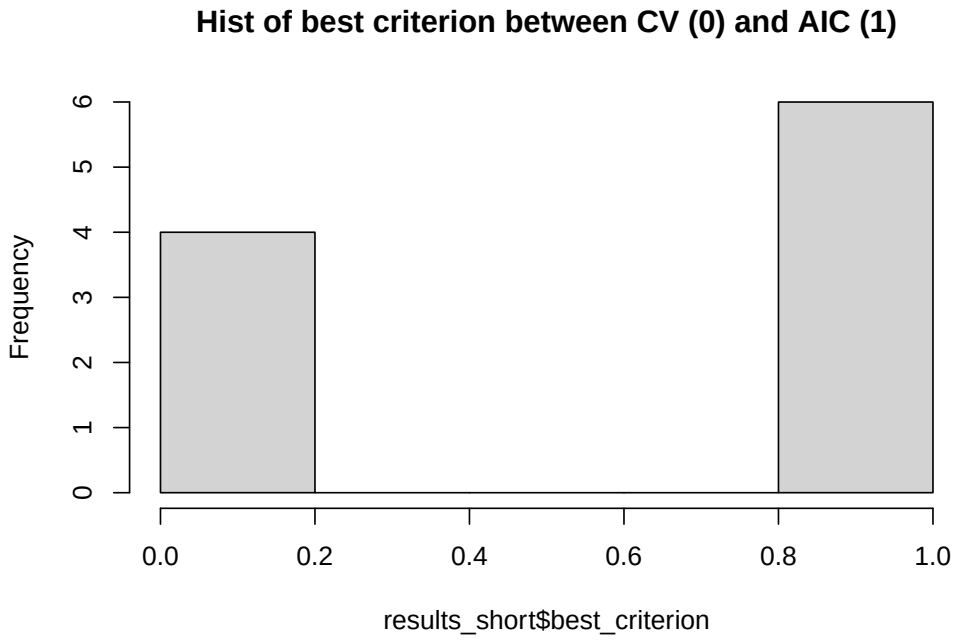
```
##We fix p, n and k
#We run M instances of Y = X\theta + eps
#Each time we obtain $\lambda_{AIC, CV, opt}$ and we create a list which contains for
#We plot the histogram to get an idea if AIC or CV is best in our setup
#TODO : same thing, but instead pick an index whenever it is the one that gives the $\
n      = 100
p      = 10
p0     = 5  #Number of nonzero coefficients
sigma  = 1
sigmaX = 1
b0     = 1
beta0  = c(rep(b0,p0),rep(0,p-p0))
betah  = matrix(NA, ncol = p, nrow = B)


B <- 20 #Number of lambda tested
lambda_step <- 0.1 #Difference between two consecutive lambda
lambda_init <- 0.0 #First value used by lambda


results_short <- data.frame(
  lambda_AIC =c(),
  lambda_CV = c(),
  lambda_opt = c(),
  best_criterion = c()
)

N<- 10 ## Number of instances of solving Y=X\theta + ups
for(t in 1:N){

  X = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
  Y = X%*%beta0 + rnorm(n,0,sigma)

  lambda <- lambda_init
```

```
results_frame <- data.frame(
  lambda = c(),
  betah = c(),
  k = c(),
  loglikelihood = c(),
  AIC = c(),
  BIC = c()
)

err_opt <- NA
lambda_opt <- 1

for (b in 1:B){#The increment is lambda
  temp = glmnet(X,Y,family = "gaussian",intercept=F, lambda=lambda)$beta
  ##the value returned by glmnet(...)$beta is a sparse matrix
  #In a sparse matrix : @i (non zero indices); @x non zero values

  betah <- c(rep(0,p))
  j<-0
  for(i in temp@i){
    betah[i+1] <- temp@x[j+1]
    j<-j+1
  }

  #llik = log(((1/sqrt(2 * pi))**(p/2))*exp(-t(Y-betah)%*%(Y-betah)/(2)))
  RSS = t(Y-X%*%betah)%*%(Y-X%*%betah) #Simpler expression for log likelihood in Gau
  new_row_df <- data.frame(
    lambda = lambda,
    betah = betah,
    k = j,
    negloglikelihood = -log(RSS),
    AIC = 2*j + n*log(RSS),
    BIC = NA
  )
  results_frame <- rbind(results_frame, new_row_df)
  if(err_opt >= Norm(betah - beta0, p=2) || is.na(err_opt)){
    lambda_opt <- lambda   #Obtaining the lambda which gives the solution closest fr
    err_opt <- Norm(betah - beta0, p = 2)
  }
  lambda <- lambda+lambda_step
}
esc <-1
lambda_min_AIC <- results_frame$AIC[0]
min_lambda_AIC <- min(results_frame$AIC)
i<-1
while (esc != 0 || i < B) {
  if( min_lambda_AIC >= results_frame$AIC[i]){
```

```
      esc = 0
      lambda_min_AIC <-results_frame$lambda[i]  #Obtaining the optimal value of lambda
    }
    i<- i + 1
  }


  new_row_short_df <- data.frame(
    lambda_AIC = lambda_min_AIC,
    lambda_CV = cv.glmnet(X,Y, family = "gaussian",intercept=F)$lambda.1se,
    lambda_opt = lambda_opt,
    best_criterion = NA
  )
  if(abs(new_row_short_df$lambda_AIC - new_row_short_df$lambda_opt) > abs(new_row_shor
    new_row_short_df$best_criterion = 0 ##0 : best criterion is CV
  }
  else{
    new_row_short_df$best_criterion = 1  ##1: best criterion is AIC
  }
  results_short <- rbind(results_short, new_row_short_df)
  print(t)

}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

```
hist(results_short$best_criterion, main = "Hist of best criterion between CV (0) and A
```




```r
###We wrap the previous code in a function
#Then we compute for varying $n/p$ whether best is AIC or CV

GetLambdaComparison <- function(n = 100,
                                p      = 10,
                                p0     = 5,  #Number of nonzero coefficients
                                sigma  = 1,
                                sigmaX = 1,
                                b0     = 1,
                                B = 20, #Number of lambda tested
                                N = 10, #Number of instances of solving Y = X\theta +
                                lambda_step = 0.1, #Difference between two consecutive
                                lambda_init = 0.0 #First value used by lambda
                                ){

  beta0  = c(rep(b0,p0),rep(0,p-p0))
  betah  = matrix(NA, ncol = p, nrow = B)

  results_short <- data.frame(
    lambda_AIC =c(),
    lambda_CV = c(),
    lambda_opt = c(),
    best_criterion = c()
  )
```

```r
for(t in 1:N){

  X = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
  Y = X%*%beta0 + rnorm(n,0,sigma)

  lambda <- lambda_init
  results_frame <- data.frame(
    lambda = c(),
    betah = c(),
    k = c(),
    loglikelihood = c(),
    AIC = c(),
    BIC = c()
  )

  err_opt <- NA
  lambda_opt <- 1
  min_lambda_AIC <- NA
  lambda_AIC <- NA

  for (b in 1:B){#The increment is lambda
    temp = glmnet(X,Y,family = "gaussian",intercept=F, lambda=lambda)$beta
    ##the value returned by glmnet(...)$beta is a sparse matrix
    #In a sparse matrix : @i (non zero indices); @x non zero values

    betah <- c(rep(0,p))
    j<-1
    for(i in temp@i){
      betah[i+1] <- temp@x[j]
      j<-j+1
    }

    #llik = log(((1/sqrt(2 * pi))**(p/2))*exp(-t(Y-betah)%*%(Y-betah)/(2)))
    RSS = t(Y-X%*%betah)%*%(Y-X%*%betah) #Simpler expression for log likelihood in C
    new_row_df <- data.frame(
      lambda = lambda,
      betah = betah,
      k = j,
      negloglikelihood = -log(RSS),
      AIC = 2*j + n*log(RSS),
      BIC = NA
    )
    results_frame <- rbind(results_frame, new_row_df)

    ##Before restarting to loop we check for the best value for lambda_opt
    if(err_opt >= Norm(betah - beta0, p=2) || is.na(err_opt)){
      lambda_opt <- lambda   #Obtaining the lambda which gives the solution closest
```

```
        err_opt <- Norm(betah - beta0, p = 2)
      }

      #And we also search for the best value for lambda under AIC
      if(min_lambda_AIC > new_row_df$AIC || is.na(min_lambda_AIC)){
        min_lambda_AIC <- new_row_df$AIC
        lambda_AIC <- lambda
      }
      lambda <- lambda+lambda_step
    }




    new_row_short_df <- data.frame(
      lambda_AIC = lambda_AIC,
      lambda_CV = cv.glmnet(X,Y, family = "gaussian",intercept=F)$lambda.1se,
      lambda_opt = lambda_opt,
      best_criterion = NA
    )

    ##We check which criterion gives the closest lambda wrt lambda_opt
    if(abs(new_row_short_df$lambda_AIC - new_row_short_df$lambda_opt) > abs(new_row_sh
      new_row_short_df$best_criterion = 0 ##0 : best criterion is CV
    }
    else{
      new_row_short_df$best_criterion = 1  ##1: best criterion is AIC
    }
    results_short <- rbind(results_short, new_row_short_df)
  }
  return(results_short)
}

n <- 100
p <- 5 #Things are simpler if n/p is an integer
ratio_best = c(rep(0,4*n/p))
result_var_np = data.frame(
  p = c(),
  num_AIC_best = c(),   ##Number of times when AIC is the best
  num_CV_best = c()   ##Number of times when CV is best
)
for(s in 5:200){#The extra parenthesis seem necessary
  results_lambda <- GetLambdaComparison(n=100, p = s, B = 20, N= 20)
  new_row_result_var_np = data.frame(
    p = s*n/p,
    num_AIC_best = sum(results_lambda$best_criterion == 1),
```

```
    num_CV_best = sum(results_lambda$best_criterion == 0)
  )
  result_var_np <- rbind(result_var_np, new_row_result_var_np)
  ratio_best[s] <- new_row_result_var_np$num_AIC_best / (new_row_result_var_np$num_CV_
  print(s)
}
```

```
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
```

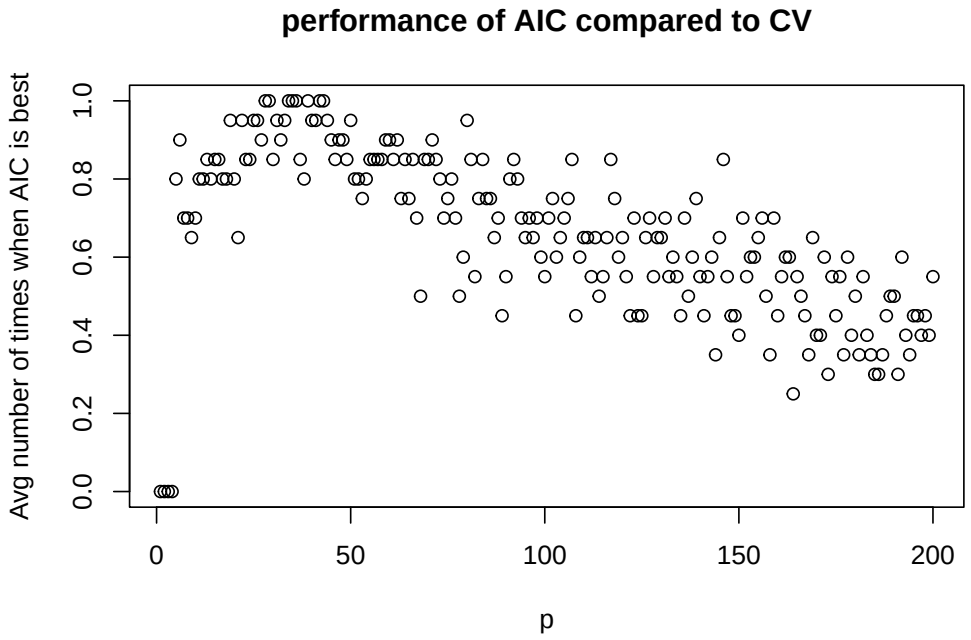
```
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
## [1] 50
## [1] 51
## [1] 52
## [1] 53
## [1] 54
## [1] 55
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
## [1] 66
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
```

```
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
## [1] 101
## [1] 102
## [1] 103
## [1] 104
## [1] 105
## [1] 106
## [1] 107
## [1] 108
## [1] 109
## [1] 110
## [1] 111
## [1] 112
## [1] 113
## [1] 114
## [1] 115
## [1] 116
## [1] 117
## [1] 118
## [1] 119
## [1] 120
## [1] 121
## [1] 122
## [1] 123
## [1] 124
## [1] 125
## [1] 126
## [1] 127
## [1] 128
## [1] 129
## [1] 130
## [1] 131
## [1] 132
## [1] 133
## [1] 134
## [1] 135
## [1] 136
## [1] 137
## [1] 138
```

```
## [1] 139
## [1] 140
## [1] 141
## [1] 142
## [1] 143
## [1] 144
## [1] 145
## [1] 146
## [1] 147
## [1] 148
## [1] 149
## [1] 150
## [1] 151
## [1] 152
## [1] 153
## [1] 154
## [1] 155
## [1] 156
## [1] 157
## [1] 158
## [1] 159
## [1] 160
## [1] 161
## [1] 162
## [1] 163
## [1] 164
## [1] 165
## [1] 166
## [1] 167
## [1] 168
## [1] 169
## [1] 170
## [1] 171
## [1] 172
## [1] 173
## [1] 174
## [1] 175
## [1] 176
## [1] 177
## [1] 178
## [1] 179
## [1] 180
## [1] 181
## [1] 182
## [1] 183
## [1] 184
## [1] 185
```

```
## [1] 186
## [1] 187
## [1] 188
## [1] 189
## [1] 190
## [1] 191
## [1] 192
## [1] 193
## [1] 194
## [1] 195
## [1] 196
## [1] 197
## [1] 198
## [1] 199
## [1] 200
```

```
plot(ratio_best, main = "performance of AIC compared to CV", ylab = "Avg number of tim
```




## 6 Overview of the project

You will first implement the cross validation and the BIC to calibrate the Lasso numerically in the regression setting. Then you will study the statistical properties of the lasso in terms of model selection in different simulation setting (high signal / high noise, low dimensional / high dimensional). You will compete CV, the AIC, and the BIC in order to determine if one procedure is more accurate than the other.

# 7 Reference

Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition, Springer, 2009