# Statistique bayésienne avec R

Julien JACQUES

## Préliminaires

Commençons par charger les libraires dont nous aurons besoin

```
library(bayess,quietly = T)
library(rjags,quietly = T)
library(BayesFactor,quietly = T)
```

## Exemple - données caterpillar

```
data("caterpillar")
y=log(caterpillar$y)
x=as.matrix(caterpillar[,1:8])
```

on normalise pour rendre les valeurs des coefficients de regression comparable

```
x=scale(x)
```

on realise un modele de regression classique

```
summary(lm(y~x))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4710 -0.4474 -0.1769  0.6121  1.5602
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.81328    0.15356  -5.296 1.97e-05 ***
## xx1         -0.52722    0.21186  -2.489   0.0202 *
## xx2         -0.39286    0.16974  -2.315   0.0295 *
## xx3          0.65133    0.38670   1.684   0.1051
## xx4         -0.29048    0.31551  -0.921   0.3664
## xx5         -0.21645    0.16865  -1.283   0.2116
## xx6          0.29361    0.53562   0.548   0.5886
## xx7         -1.09027    0.47020  -2.319   0.0292 *
## xx8         -0.02312    0.17225  -0.134   0.8944
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8821 on 24 degrees of freedom
```

```
## Multiple R-squared:  0.6234, Adjusted R-squared:  0.4979
## F-statistic: 4.967 on 8 and 24 DF,  p-value: 0.001032
```

seules les variables 1, 2 et 7 semblent significatives

Effectuons une regression Bayesienne avec un priori de Zellner

```
#res1=BayesReg(y,x,betatilde = rep(0,8),g=length(y))
res1=BayesReg(y,x,betatilde = c(0,0,1,0,0,0,0,0),g=1)
```

```
##
##           PostMean PostStError Log10bf EvidAgaH0
## Intercept  -0.8133      0.2689
## x1         -0.2596      0.2583  0.0795       (*)
## x2         -0.1934      0.2070  0.0489       (*)
## x3          0.8207      0.4715   0.028       (*)
## x4         -0.1430      0.3847 -0.1186
## x5         -0.1066      0.2056 -0.0886
## x6          0.1446      0.6531 -0.1392
## x7         -0.5368      0.5733  0.0496       (*)
## x8         -0.0114      0.2100 -0.1498
##
##
## Posterior Mean of Sigma2: 2.3861
## Posterior StError of Sigma2: 3.4322
```

Même résultat que pour l'approche classique. On peut inserer notre information a priori sur les beta, et l'importance que celle-ci prend vis-à-vis des autres données avec g (plus g est petit plus l'importance de l'a priori est grand. g=n donne autant d'importance a l'a priori qu'a une observation). Par exemple ici on introduit un a priori sur le fait que le coefficient de x3 est égale à 1, et ce avec une confiance forte (autant de poids que toutes les autres données réunies)

```
res1=BayesReg(y,x,betatilde = c(0,0,1,0,0,0,0,0),g=1)
```

```
##
##           PostMean PostStError Log10bf EvidAgaH0
## Intercept  -0.8133      0.2689
## x1         -0.2596      0.2583  0.0795       (*)
## x2         -0.1934      0.2070  0.0489       (*)
## x3          0.8207      0.4715   0.028       (*)
## x4         -0.1430      0.3847 -0.1186
## x5         -0.1066      0.2056 -0.0886
## x6          0.1446      0.6531 -0.1392
## x7         -0.5368      0.5733  0.0496       (*)
## x8         -0.0114      0.2100 -0.1498
##
##
## Posterior Mean of Sigma2: 2.3861
## Posterior StError of Sigma2: 3.4322
```

Sans surprise, x3 devient alors aussi significative que les 3 autres variables x1, x2 et x7. A noter que d'imposer un poids fort sur l'a priori, avec un a priori à 0 pour ces 3 variables, tend à réduire leur significativité. Là encore c'est logique.

On peut effectuer une sélection de variables

```
ModChoBayesReg(y,x)
```

```
##
```

```
## Number of variables less than 15
## Model posterior probabilities are calculated exactly
##
##    Top10Models PostProb
## 1        1 2 7   0.0767
## 2          1 7   0.0689
## 3      1 2 3 7   0.0686
## 4        1 3 7   0.0376
## 5        1 2 6   0.0369
## 6    1 2 3 5 7   0.0326
## 7      1 2 5 7   0.0294
## 8          1 6   0.0205
## 9      1 2 4 7   0.0201
## 10           7   0.0198
```

```
## $top10models
##  [1] "1 2 7"     "1 7"       "1 2 3 7"   "1 3 7"     "1 2 6"     "1 2 3 5 7"
##  [7] "1 2 5 7"   "1 6"       "1 2 4 7"   "7"
##
## $postprobtop10
##  [1] 0.07670048 0.06894313 0.06855427 0.03759751 0.03688912 0.03262797
##  [7] 0.02941759 0.02050185 0.02006371 0.01979095
```

Les trois variables retenues par le meilleur sont 1, 2 et 7.

Plutôt que de choisir un unique modèle, on peut moyenner les meilleurs modèles à l'aide du Bayesian Model Averaging

```
library('BMA',quietly = T)
```

```
##
## Attaching package: 'robustbase'
```

```
## The following object is masked from 'package:survival':
##
##     heart
```

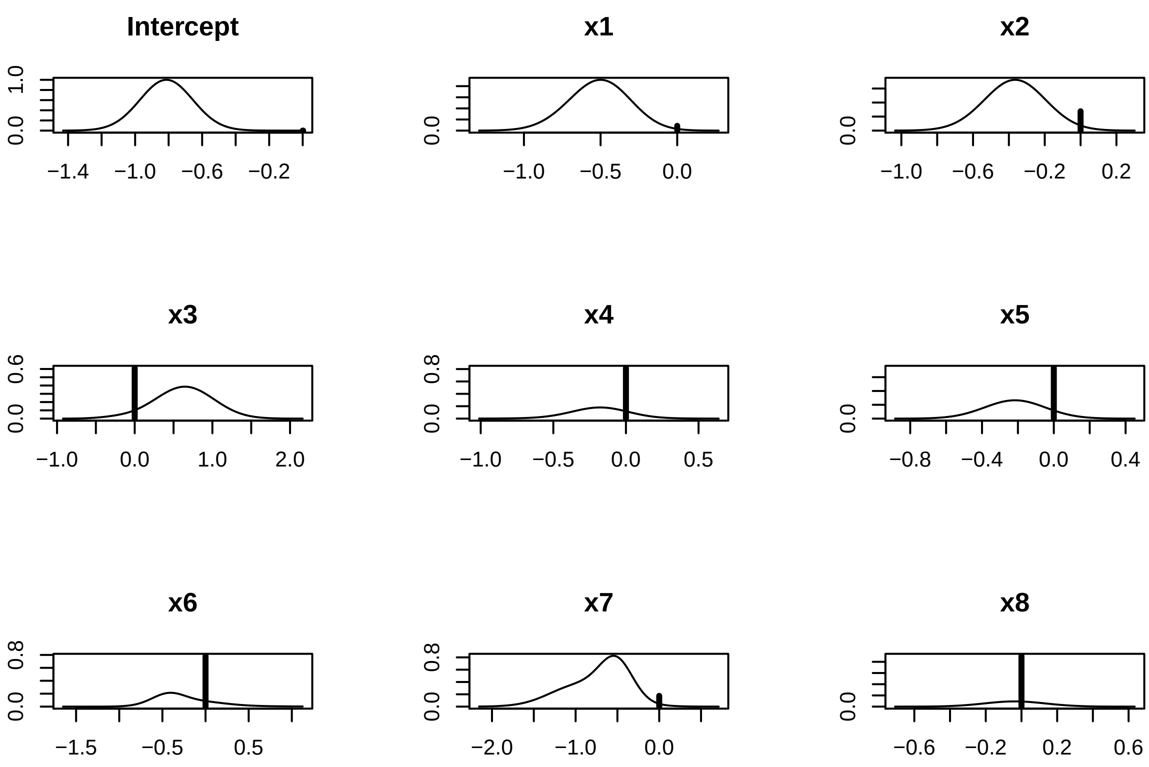
```
## Scalable Robust Estimators with High Breakdown Point (version 1.6-0)
```

```
bma=bicreg(x,y)
summary(bma)
```

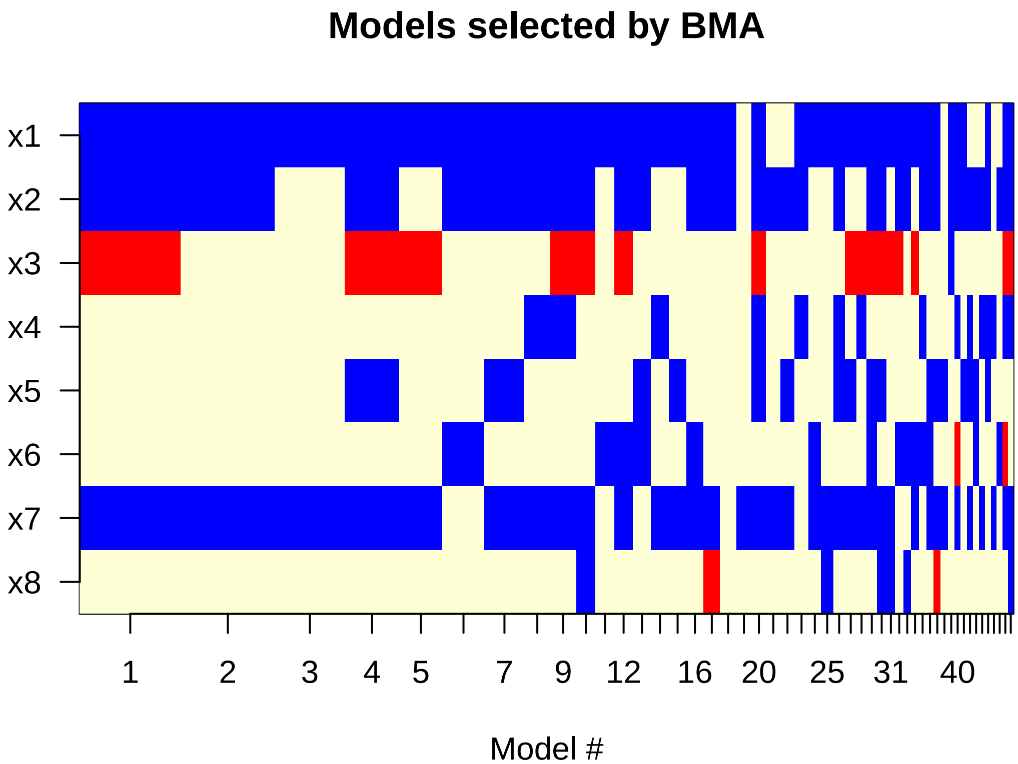
```
##
## Call:
## bicreg(x = x, y = y)
##
##
##   49  models were selected
##  Best  5  models (cumulative posterior probability =  0.3884 ):
##
##            p!=0    EV         SD       model 1   model 2   model 3   model 4
## Intercept  100.0  -0.813281  0.15861   -0.8133   -0.8133   -0.8133   -0.8133
## x1          91.4  -0.467197  0.23971   -0.6209   -0.4548   -0.4556   -0.5642
## x2          72.4  -0.265331  0.21969   -0.3541   -0.3276       .      -0.3977
## x3          38.6   0.239301  0.38959    0.6590       .         .        0.6818
## x4          18.0  -0.033645  0.11514      .         .         .          .
## x5          26.5  -0.057497  0.13029      .         .         .      -0.2200
## x6          21.4  -0.056307  0.19728      .         .         .          .
```

```
## x7          82.5  -0.595837  0.43263   -0.9762   -0.4674   -0.5738   -0.9887
## x8           9.3  -0.003435  0.05371      .         .         .         .
##
## nVar                                       4         3         2         5
## r2                                       0.580     0.531     0.469     0.608
## BIC                                    -14.6336  -14.4930  -13.8989  -13.3999
## post prob                                0.108     0.101     0.075     0.058
##            model 5
## Intercept   -0.8133
## x1          -0.6035
## x2             .
## x3           0.5869
## x4             .
## x5             .
## x6             .
## x7          -1.0346
## x8             .
##
## nVar           3
## r2           0.508
## BIC        -12.9300
## post prob    0.046
```

```
plot(bma)
```




```
imageplot.bma(bma)
```

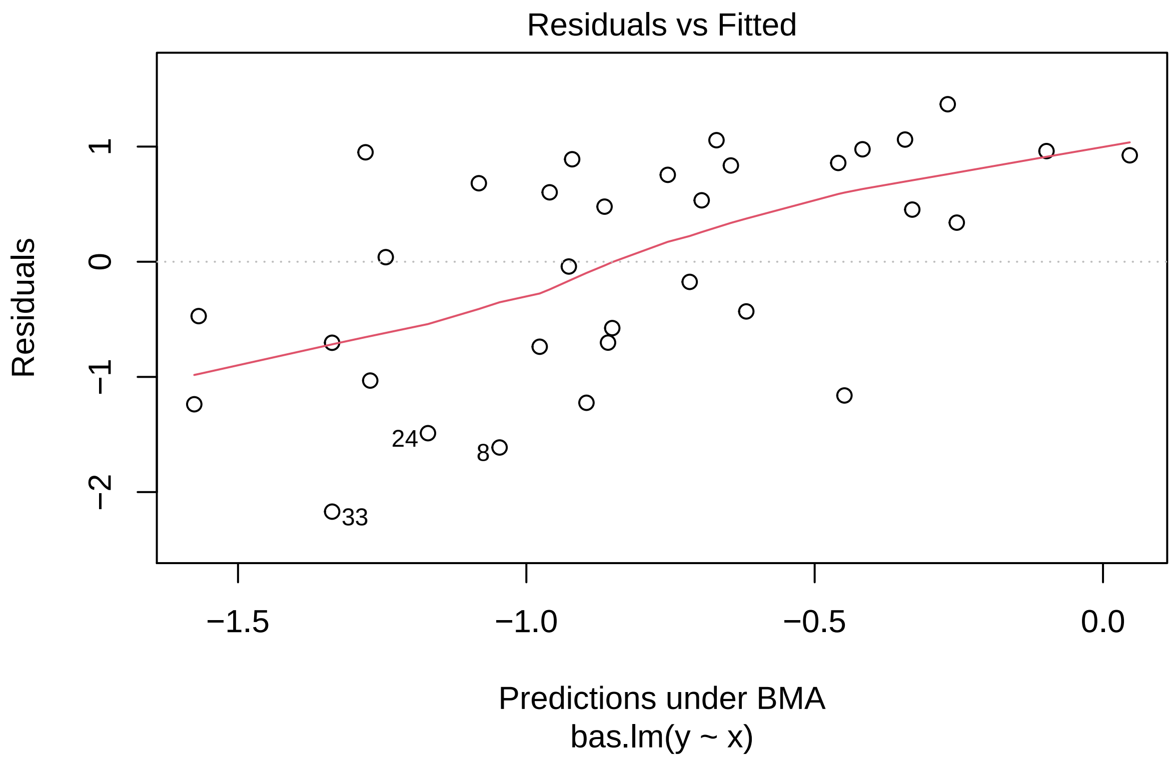



De même avec le package BAS, et un a priori beta.binomial correspondant à une loi uniforme

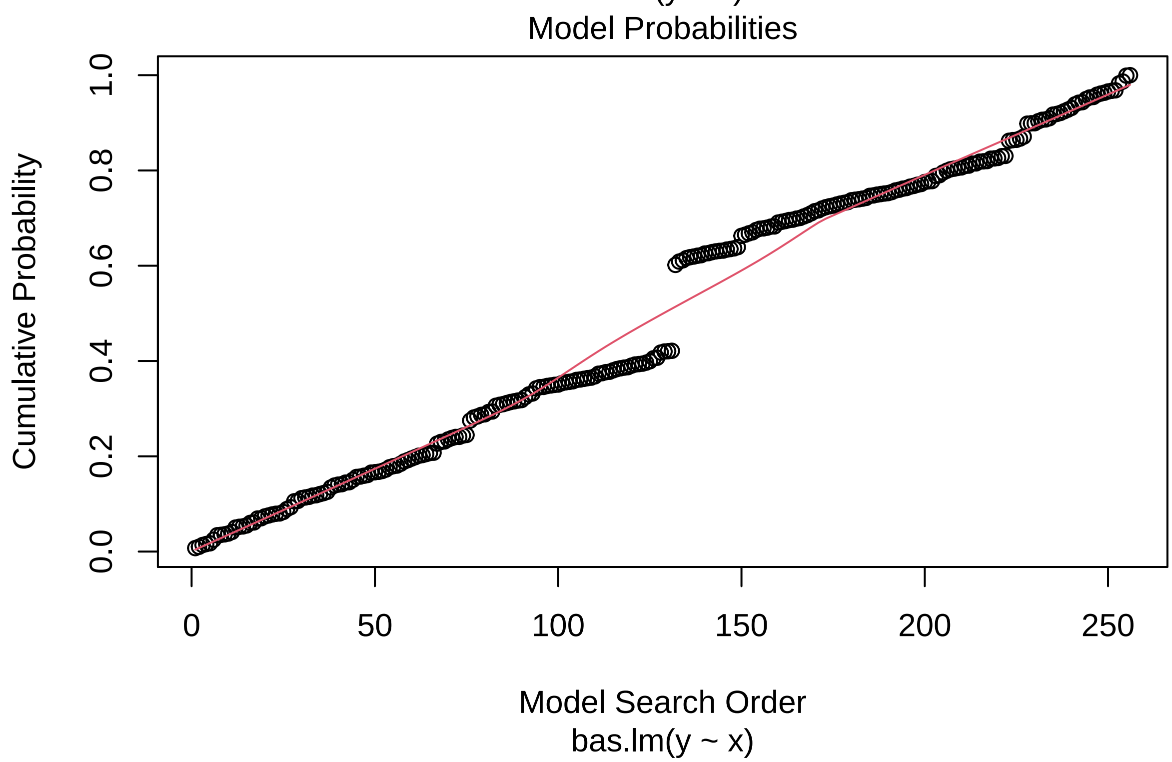
```
library('BAS',quietly = T)
bma2=bas.lm(y~x,prior="g-prior",alpha=1,modelprior = beta.binomial(1,1))
summary(bma2)
```

```
##            P(B != 0 | Y)  model 1 model 2   model 3   model 4   model 5
## Intercept      1.0000000 1.000000 1.00000 1.0000000 1.0000000 1.0000000
## xx1            0.7568572 1.000000 1.00000 1.0000000 1.0000000 1.0000000
## xx2            0.7144789 1.000000 1.00000 1.0000000 1.0000000 1.0000000
## xx3            0.6329418 1.000000 1.00000 1.0000000 1.0000000 1.0000000
## xx4            0.6042113 1.000000 1.00000 1.0000000 0.0000000 1.0000000
## xx5            0.6189930 1.000000 1.00000 1.0000000 1.0000000 0.0000000
## xx6            0.6149797 1.000000 1.00000 0.0000000 1.0000000 1.0000000
## xx7            0.7356610 1.000000 1.00000 1.0000000 1.0000000 1.0000000
## xx8            0.5678207 1.000000 0.00000 1.0000000 1.0000000 1.0000000
## BF                    NA 0.709433 1.00000 0.9498803 0.8602391 0.7450354
## PostProbs             NA 0.180300 0.03180 0.0302000 0.0273000 0.0237000
## R2                    NA 0.623400 0.62320 0.6187000 0.6101000 0.5976000
## dim                   NA 9.000000 8.00000 8.0000000 8.0000000 8.0000000
## logmarg               NA 3.204440 3.54773 3.4963103 3.3971846 3.2534060
```

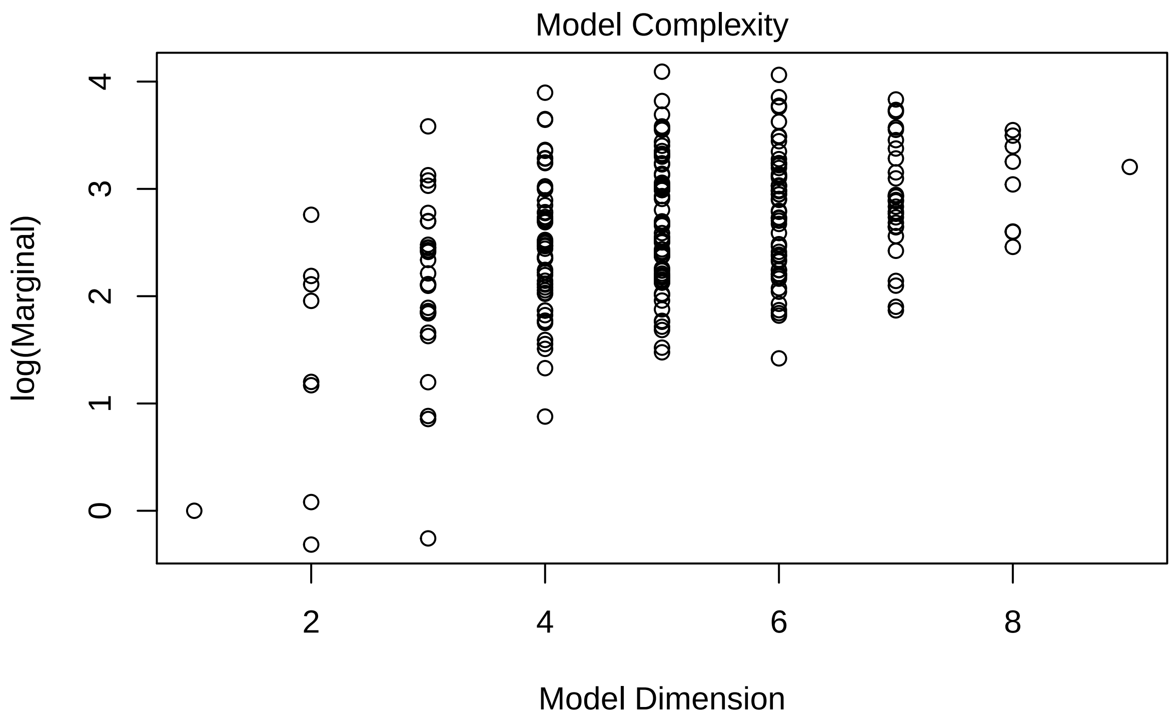
```
plot(bma2)
```



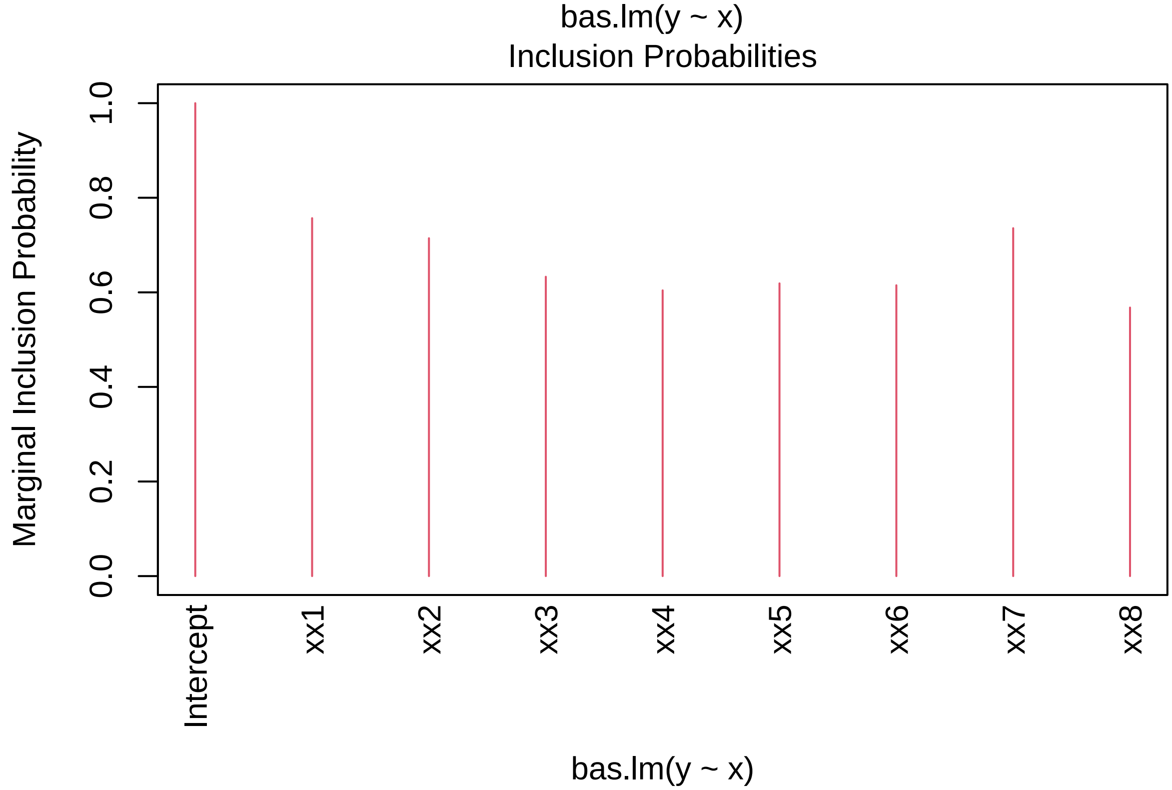
Residuals vs Fitted
Residuals
1
0
−1
−2
24
8
33
−1.5
−1.0
−0.5
0.0
Predictions under BMA
bas.lm(y ~ x)

Model Probabilities
Cumulative Probability
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
Model Search Order
bas.lm(y ~ x)

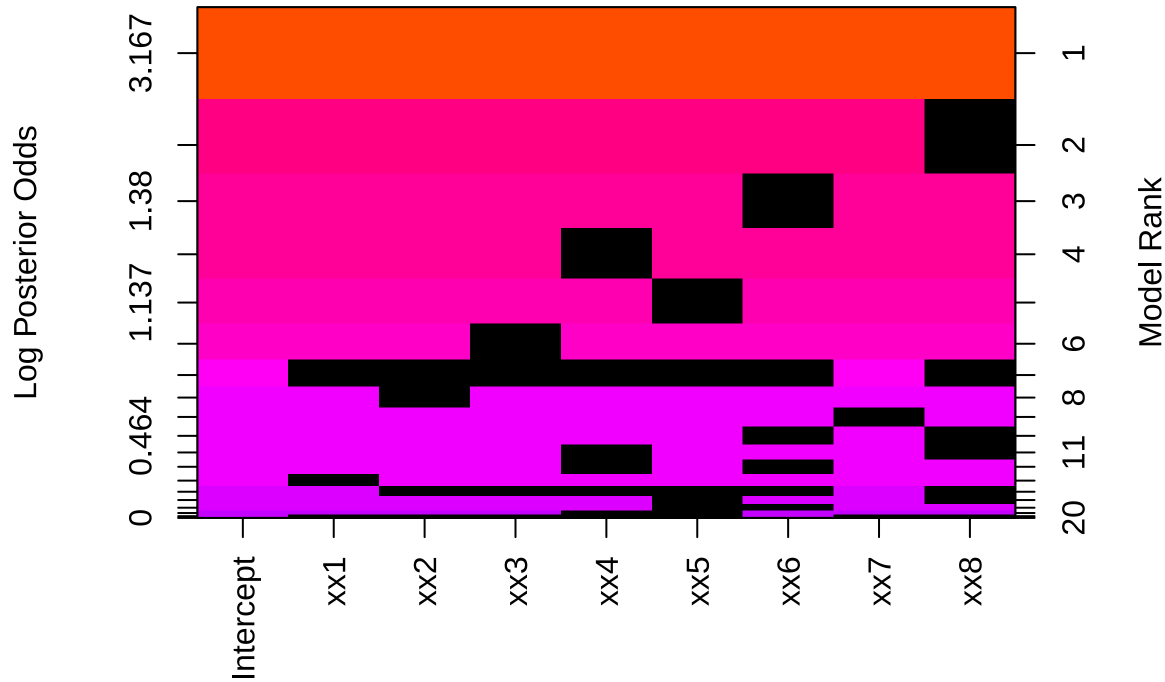


Model Complexity
log(Marginal)
0
1
2
3
4
2
4
6
8
Model Dimension

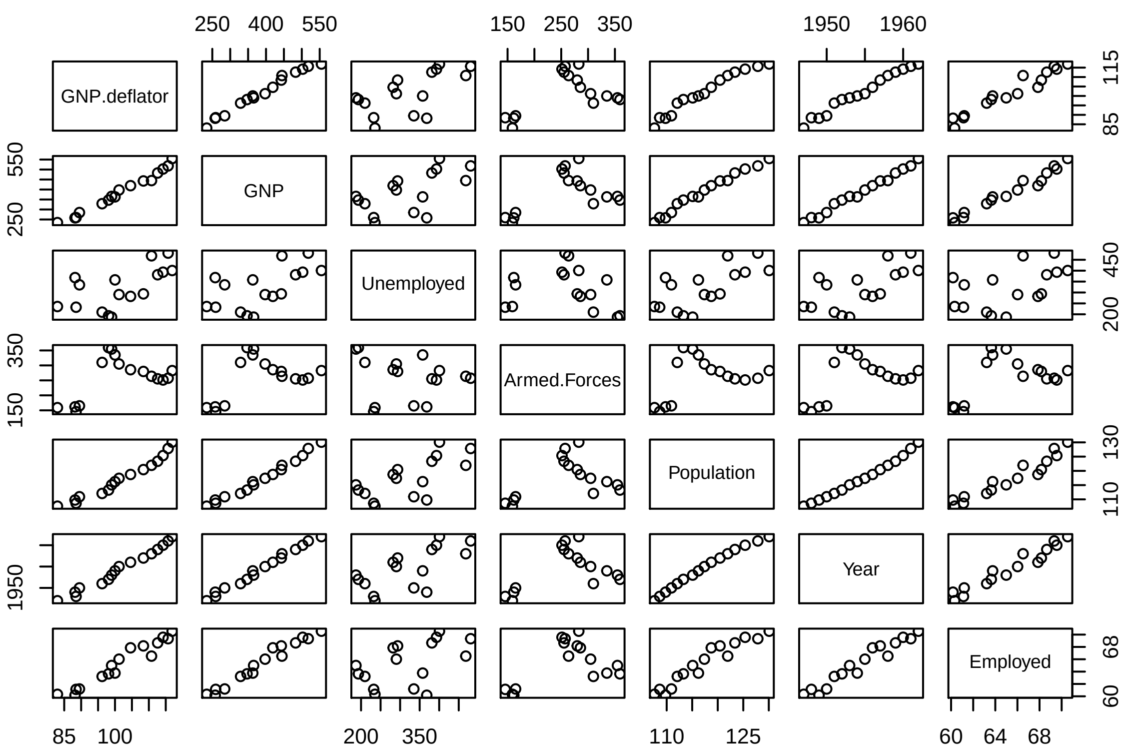
bas.lm(y ~ x)
Inclusion Probabilities
Marginal Inclusion Probability
0.0
0.2
0.4
0.6
0.8
1.0
Intercept
xx1
xx2
xx3
xx4
xx5
xx6
xx7
xx8
bas.lm(y ~ x)


```
image(bma2)
```

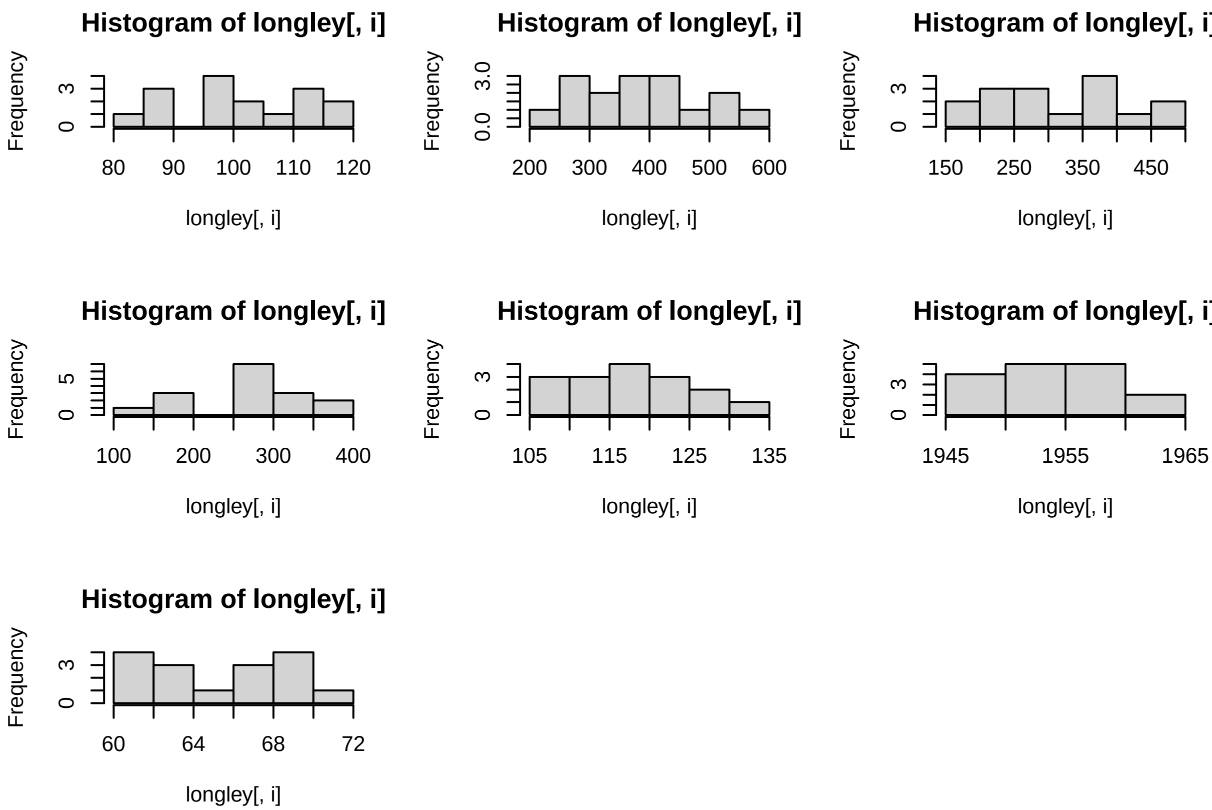



## Exercice - données longley

```
data(longley)
plot(longley)
```




```
par(mfrow=c(3,3)); for (i in 1:7) hist(longley[,i])
```




La normalité des données ne semble pas si mauvaise…

Réalisons une régression classique

```
modele=lm(Employed~.,data=longley)
summary(modele)
```

```
##
## Call:
## lm(formula = Employed ~ ., data = longley)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.41011 -0.15767 -0.02816  0.10155  0.45539
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.482e+03  8.904e+02  -3.911 0.003560 **
## GNP.deflator  1.506e-02  8.492e-02   0.177 0.863141
## GNP          -3.582e-02  3.349e-02  -1.070 0.312681
## Unemployed   -2.020e-02  4.884e-03  -4.136 0.002535 **
## Armed.Forces -1.033e-02  2.143e-03  -4.822 0.000944 ***
## Population   -5.110e-02  2.261e-01  -0.226 0.826212
## Year          1.829e+00  4.555e-01   4.016 0.003037 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3049 on 9 degrees of freedom
## Multiple R-squared:  0.9955, Adjusted R-squared:  0.9925
## F-statistic: 330.3 on 6 and 9 DF,  p-value: 4.984e-10
```

Nous pouvons réaliser une sélection de variables à l'aide d'une procédure stepwise

```
stepAIC(modele)
```

```
## Start:  AIC=-33.22
## Employed ~ GNP.deflator + GNP + Unemployed + Armed.Forces + Population +
##     Year
##
##                Df Sum of Sq     RSS     AIC
## - GNP.deflator  1   0.00292 0.83935 -35.163
## - Population    1   0.00475 0.84117 -35.129
## - GNP           1   0.10631 0.94273 -33.305
## <none>                      0.83642 -33.219
## - Year          1   1.49881 2.33524 -18.792
## - Unemployed    1   1.59014 2.42656 -18.178
## - Armed.Forces  1   2.16091 2.99733 -14.798
##
## Step:  AIC=-35.16
## Employed ~ GNP + Unemployed + Armed.Forces + Population + Year
##
##                Df Sum of Sq    RSS     AIC
## - Population    1   0.01933 0.8587 -36.799
## <none>                      0.8393 -35.163
## - GNP           1   0.14637 0.9857 -34.592
## - Year          1   1.52725 2.3666 -20.578
## - Unemployed    1   2.18989 3.0292 -16.628
## - Armed.Forces  1   2.39752 3.2369 -15.568
##
## Step:  AIC=-36.8
## Employed ~ GNP + Unemployed + Armed.Forces + Year
##
##                Df Sum of Sq    RSS     AIC
## <none>                      0.8587 -36.799
## - GNP           1    0.4647 1.3234 -31.879
## - Year          1    1.8980 2.7567 -20.137
## - Armed.Forces  1    2.3806 3.2393 -17.556
## - Unemployed    1    4.0491 4.9077 -10.908

##
## Call:
## lm(formula = Employed ~ GNP + Unemployed + Armed.Forces + Year,
##     data = longley)
##
## Coefficients:
##  (Intercept)           GNP    Unemployed  Armed.Forces          Year
##   -3.599e+03    -4.019e-02    -2.088e-02    -1.015e-02     1.887e+00
```

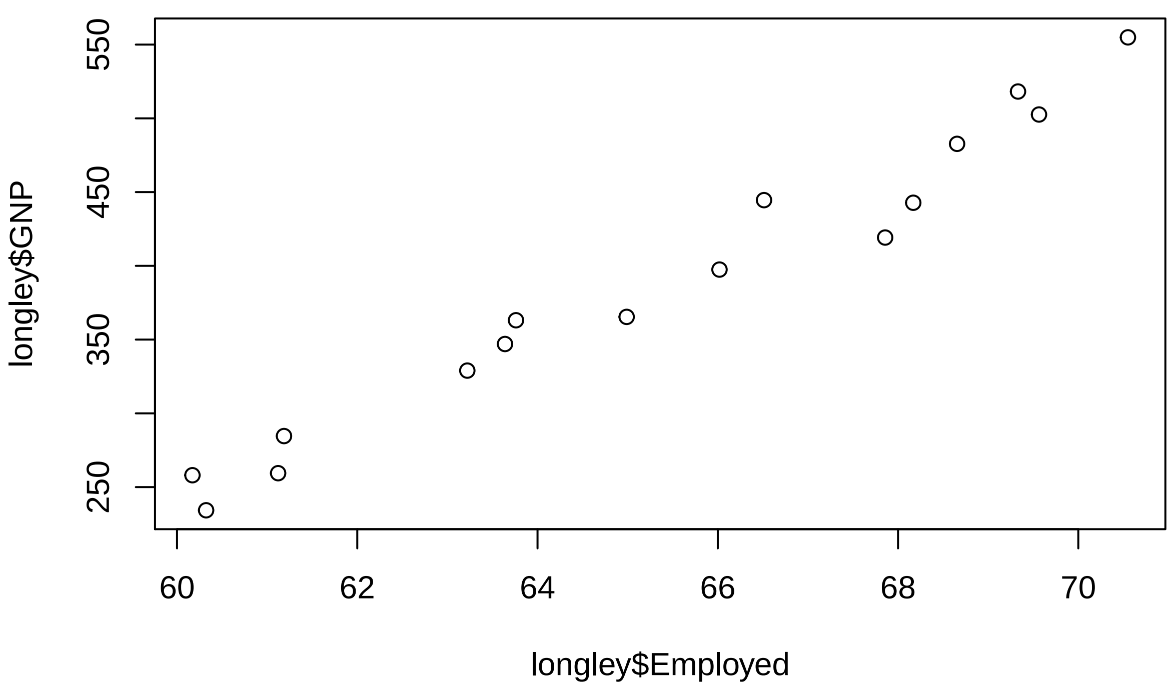
```
modele=lm(formula = Employed ~ GNP + Unemployed + Armed.Forces + Year,
    data = longley)
summary(modele)
```

```
##
## Call:
## lm(formula = Employed ~ GNP + Unemployed + Armed.Forces + Year,
##     data = longley)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42165 -0.12457 -0.02416  0.08369  0.45268
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.599e+03  7.406e+02  -4.859 0.000503 ***
## GNP          -4.019e-02  1.647e-02  -2.440 0.032833 *
## Unemployed   -2.088e-02  2.900e-03  -7.202 1.75e-05 ***
## Armed.Forces -1.015e-02  1.837e-03  -5.522 0.000180 ***
## Year          1.887e+00  3.828e-01   4.931 0.000449 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2794 on 11 degrees of freedom
## Multiple R-squared:  0.9954, Adjusted R-squared:  0.9937
## F-statistic: 589.8 on 4 and 11 DF,  p-value: 9.5e-13
```

Ces coefficients semble étrange, d'autant qu'il est communément admis que le GNP et le taux d'emploi sont positivement corrélé (Okun's Law), ce que l'on constate sur ces données

```
plot(longley$Employed,longley$GNP)
```




Passons cette fois à une régression bayesienne. Nous allons essayer de forcer l'importance de la variable GNP

```
y=longley[,7]
x=scale(longley[,-7])
summary(lm(y~x[,2]))
```

```
##
## Call:
## lm(formula = y ~ x[, 2])
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.77958 -0.55440 -0.00944  0.34361  1.44594
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  65.3170     0.1642  397.90  < 2e-16 ***
## x[, 2]        3.4542     0.1695   20.37 8.36e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6566 on 14 degrees of freedom
## Multiple R-squared:  0.9674, Adjusted R-squared:  0.965
## F-statistic: 415.1 on 1 and 14 DF,  p-value: 8.363e-12
```

Une fois les données normalisées, le coefficients de la régression de Employed en fonction de GNP est de 3. Nous allons essayer de le forcer.

```r
colnames(x)=names(longley)[-7]
res1=BayesReg(y,x,betatilde = c(0,3.45,0,0,0,0),g=1)
```

```
##
##           PostMean PostStError Log10bf EvidAgaH0
## Intercept  65.3170      0.1302
## x1          0.0787      1.0717 -0.1492
## x2          0.0014      3.8931 -0.2504
## x3         -0.9139      0.5338  0.5119      (**)
## x4         -0.3481      0.1744  0.7203      (**)
## x5         -0.1721      1.8392 -0.1483
## x6          4.2160      2.5361  0.4773       (*)
##
##
## Posterior Mean of Sigma2: 0.2712
## Posterior StError of Sigma2: 0.3992
```

Même avec cet a priori fort sur la variable GNP, elle ne sort pas significative dans le modèle. Les corrélations entre features sont trop importantes et les données trop peu nombreuse pour qu'on arrive à obtenir quelque chose de satisfaisant.

Effectuons une sélection de modèle bayésienne. La encore, les variables GNP ne ressortent pas. . .

```r
ModChoBayesReg(y,x,betatilde = c(0,3,0,0,0,0),g=1)
```

```
##
## Number of variables less than 15
## Model posterior probabilities are calculated exactly
##
##    Top10Models PostProb
## 1        3 4 6   0.1499
## 2      1 3 4 6   0.1037
## 3      3 4 5 6   0.0856
## 4      2 3 4 6   0.0661
## 5    1 3 4 5 6   0.0547
## 6    2 3 4 5 6   0.0475
## 7    1 2 3 4 6   0.0475
## 8  1 2 3 4 5 6   0.0337
## 9          3 6   0.0240
## 10       2 3 4   0.0216
```

```
## $top10models
##  [1] "3 4 6"       "1 3 4 6"     "3 4 5 6"     "2 3 4 6"     "1 3 4 5 6"
##  [6] "2 3 4 5 6"   "1 2 3 4 6"   "1 2 3 4 5 6" "3 6"         "2 3 4"
##
## $postprobtop10
##  [1] 0.14985511 0.10367545 0.08562874 0.06613600 0.05468283 0.04754353
##  [7] 0.04746943 0.03370249 0.02401620 0.02159695
```

Peut-être que la variable qui nous gène est la variable Unemployed, qui via la variable population est très liée à la variable Employed… Essayons sans celle-ci.

```
modele=lm(Employed~.,data=longley[,-3])
summary(modele)
```

```
##
## Call:
## lm(formula = Employed ~ ., data = longley[, -3])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66533 -0.30357  0.01415  0.26417  0.77112
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -403.18616  789.53789  -0.511 0.620671
## GNP.deflator    -0.17988    0.11414  -1.576 0.146112
## GNP              0.09518    0.01760   5.406 0.000299 ***
## Armed.Forces    -0.00485    0.00272  -1.783 0.104971
## Population      -0.76018    0.23816  -3.192 0.009624 **
## Year             0.27650    0.41690   0.663 0.522174
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4926 on 10 degrees of freedom
## Multiple R-squared:  0.9869, Adjusted R-squared:  0.9803
## F-statistic: 150.5 on 5 and 10 DF,  p-value: 4.479e-09
```

```
stepAIC(modele,trace = F)
```

```
##
## Call:
## lm(formula = Employed ~ GNP.deflator + GNP + Armed.Forces + Population,
##     data = longley[, -3])
##
## Coefficients:
##  (Intercept)  GNP.deflator           GNP  Armed.Forces    Population
##   120.323684     -0.136326      0.096598     -0.004689     -0.658925
```

```
BayesReg(y,x[,-3],betatilde = c(0,3,0,0,0),g=1)
```

```
##
##           PostMean PostStError Log10bf EvidAgaH0
## Intercept  65.3170      0.1578
## x1         -0.9398      1.0807  0.0336       (*)
## x2          6.0799      1.5353  2.2026    (****)
## x3         -0.1634      0.1661  0.0833       (*)
## x4         -2.5600      1.4536  0.5464      (**)
```

```
## x5           0.6373      1.7415 -0.1171
##
##
## Posterior Mean of Sigma2: 0.3985
## Posterior StError of Sigma2: 0.5866
```

```
## $postmeancoeff
## [1] 65.3170000 -0.9397654  6.0799417 -0.1633929 -2.5599763  0.6373018
##
## $postsqrtcoeff
##              GNP.deflator          GNP Armed.Forces   Population         Year
##    0.1578230    1.0807424    1.5353207    0.1661166    1.4535924    1.7415120
##
## $log10bf
## [1]  0.03363109  2.20262960  0.08329443  0.54642883 -0.11713301
##
## $postmeansigma2
## [1] 0.3985297
##
## $postvarsigma2
## [1] 0.3441228
```

```
ModChoBayesReg(y,x[,-3],betatilde = c(0,3,0,0,0),g=1)
```

```
##
## Number of variables less than 15
## Model posterior probabilities are calculated exactly
##
##    Top10Models PostProb
## 1      1 2 3 4   0.1148
## 2        2 3 4   0.1130
## 3        1 2 4   0.0981
## 4          2 4   0.0930
## 5    1 2 3 4 5   0.0876
## 6      2 3 4 5   0.0811
## 7      1 2 4 5   0.0724
## 8        2 4 5   0.0684
## 9          2 5   0.0495
## 10           2   0.0373
```

```
## $top10models
##  [1] "1 2 3 4"   "2 3 4"     "1 2 4"     "2 4"       "1 2 3 4 5" "2 3 4 5"
##  [7] "1 2 4 5"   "2 4 5"     "2 5"       "2"
##
## $postprobtop10
##  [1] 0.11478087 0.11304202 0.09812661 0.09295670 0.08764689 0.08111579
##  [7] 0.07235059 0.06839424 0.04945820 0.03726181
```

### BMA sur les données longley

```
library('BMA',quietly = T)
bma=bicreg(x[,-3],y)
summary(bma)
```

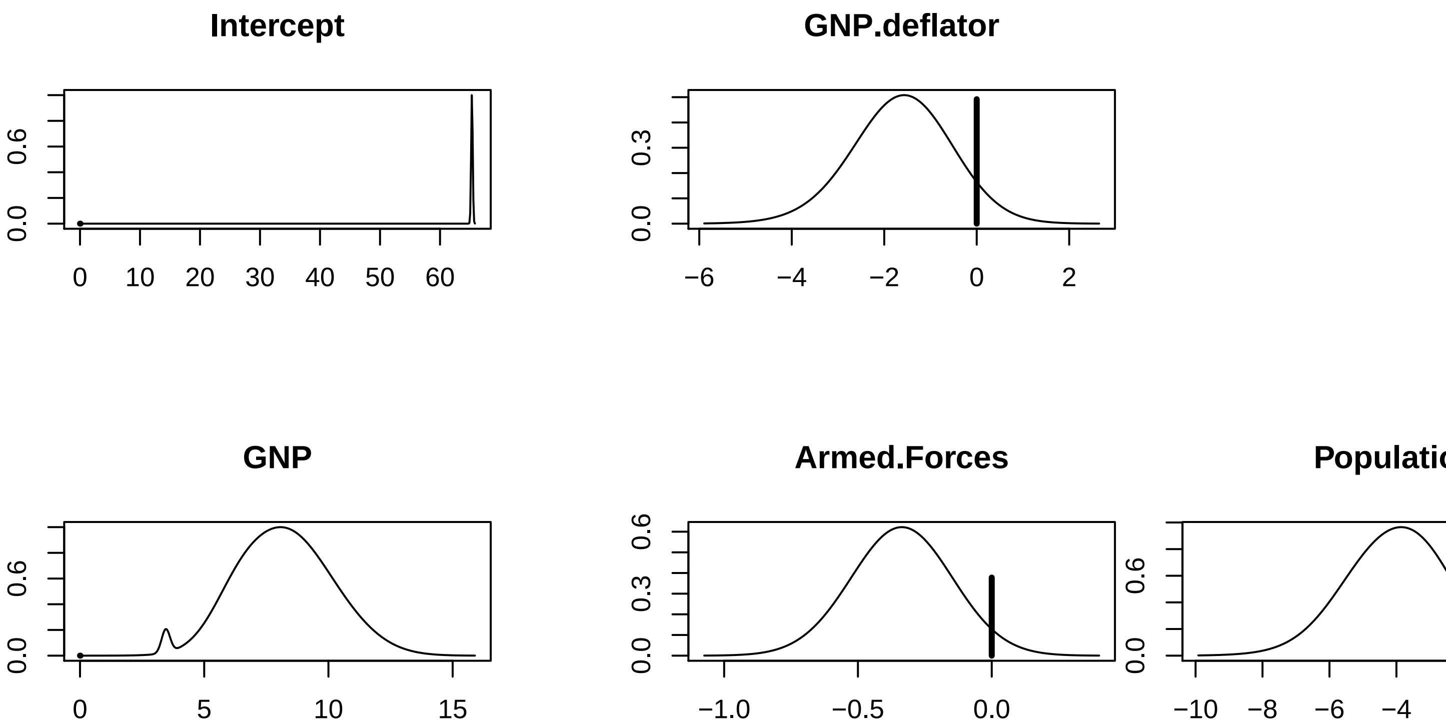
```
##
## Call:
```

```
## bicreg(x = x[, -3], y = y)
##
##
##   10  models were selected
##  Best  5  models (cumulative posterior probability =  0.8283 ):
##
##               p!=0    EV       SD      model 1   model 2   model 3   model 4
## Intercept     100.0  65.31700  0.1289   65.3170   65.3170   65.3170   65.3170
## GNP.deflator   50.8  -0.82510  1.1273   -1.4712      .         .      -1.5988
## GNP           100.0   8.17122  1.9856    9.6014    7.9444    6.2790    8.1851
## Armed.Forces   62.2  -0.20936  0.2214   -0.3263   -0.3465      .         .
## Population     96.5  -3.87761  1.6639   -4.5836   -4.3744   -2.8502   -3.1738
## Year           24.4   0.03785  1.1295      .         .         .         .
##
## nVar                                       4         3         2         3
## r2                                       0.986     0.984     0.979     0.982
## BIC                                     -57.5671  -57.3623  -56.3123  -56.3018
## post prob                                0.250     0.225     0.133     0.133
##               model 5
## Intercept      65.3170
## GNP.deflator   -1.9412
## GNP             9.4603
## Armed.Forces   -0.3375
## Population     -5.2879
## Year            1.3164
##
## nVar              5
## r2              0.987
## BIC            -55.4749
## post prob       0.088
```

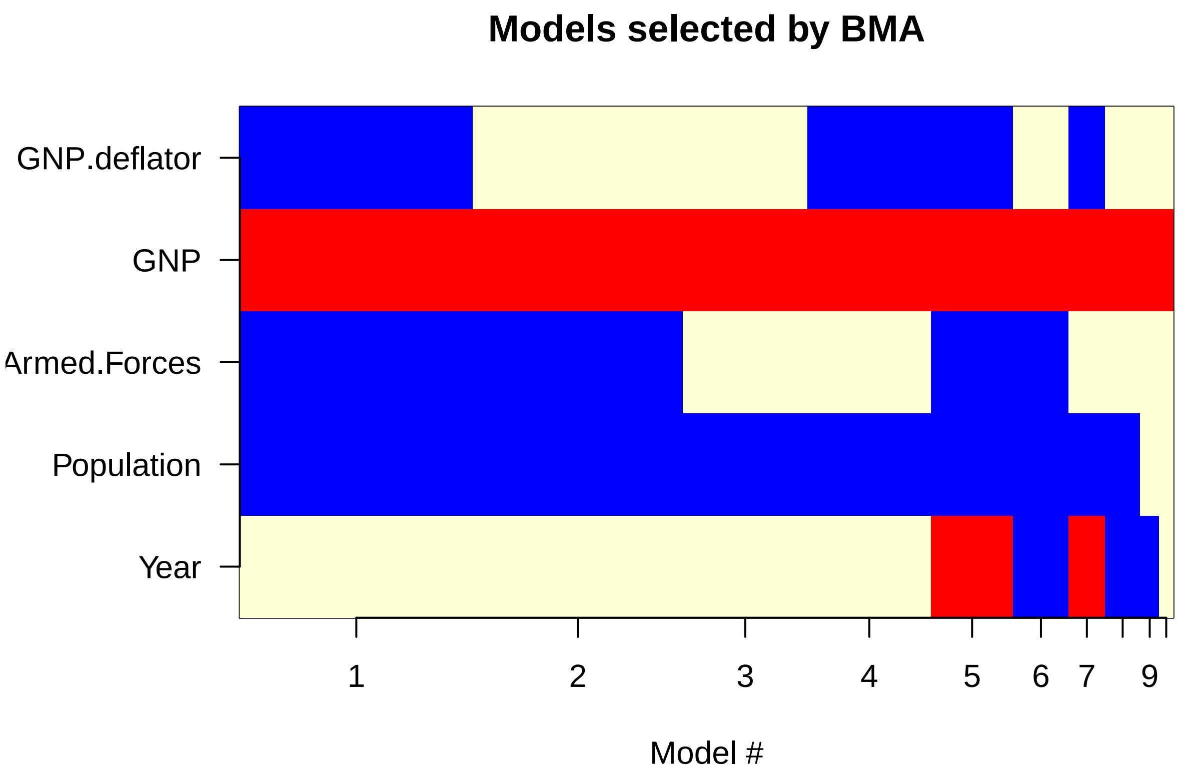
```
print(bma)
```

```
##
## Call:
## bicreg(x = x[, -3], y = y)
##
##
##  Posterior probabilities(%):
## GNP.deflator          GNP Armed.Forces   Population         Year
##        50.8        100.0         62.2         96.5         24.4
##
##  Coefficient posterior expected values:
##  (Intercept)  GNP.deflator           GNP  Armed.Forces    Population
##     65.31700      -0.82510       8.17122      -0.20936      -3.87761
##         Year
##      0.03785
```

```
plot(bma)
```




```
imageplot.bma(bma)
```



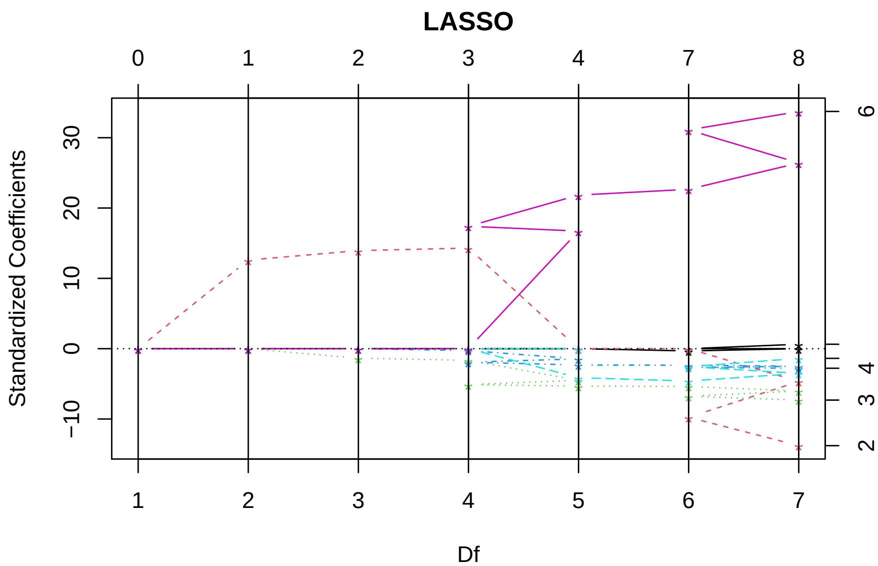

### Regression LASSO sur les données longley

Effectuons une régression LASSO sur les données longley

```
library('lars')
```

```
## Loaded lars 1.2
```

```
model_lasso=lars(as.matrix(longley[,-7]),longley$Employed,type="lasso",trace=F,normalize=TRUE)
plot(model_lasso,xvar='df', plottype='coeff')
```



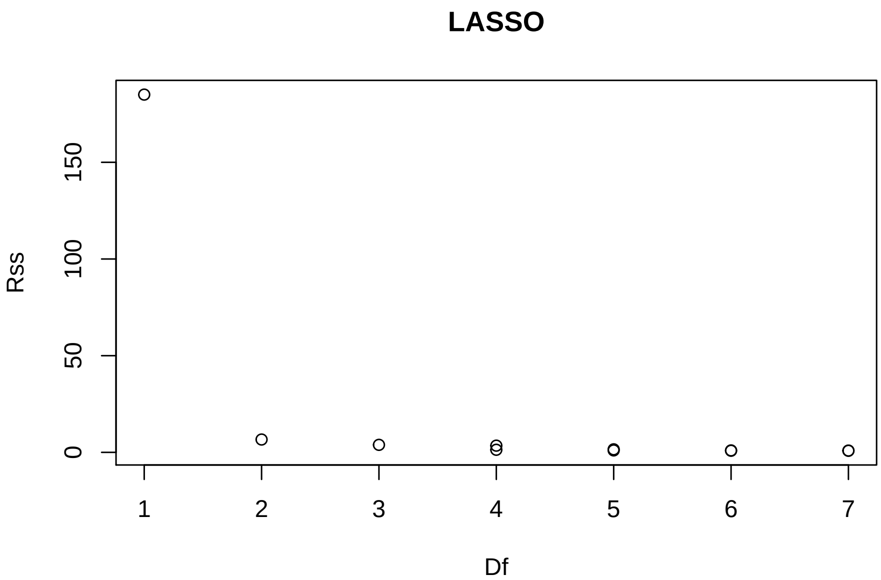

On peut afficher les valeurs des coefficients à chaque étape de notre algorithme LARS-LASSO

```
print(model_lasso$beta)
```

```
##    GNP.deflator         GNP   Unemployed  Armed.Forces  Population      Year
## 0   0.000000000  0.00000000  0.000000000  0.0000000000  0.00000000 0.0000000
## 1   0.000000000  0.03272990  0.000000000  0.0000000000  0.00000000 0.0000000
## 2   0.000000000  0.03623013 -0.003723046  0.0000000000  0.00000000 0.0000000
## 3   0.000000000  0.03716606 -0.004594753 -0.0009913275  0.00000000 0.0000000
## 4   0.000000000  0.00000000 -0.012420901 -0.0053898903  0.00000000 0.9068087
## 5   0.000000000  0.00000000 -0.014117085 -0.0071286425  0.00000000 0.9437545
## 6   0.000000000  0.00000000 -0.014712420 -0.0086141622 -0.15337341 1.1842962
## 7  -0.007698749  0.00000000 -0.014808922 -0.0087270930 -0.17076177 1.2288796
## 8   0.000000000 -0.01211579 -0.016633224 -0.0092700415 -0.13028773 1.4319208
## 9   0.000000000 -0.02533778 -0.018690775 -0.0098894188 -0.09514287 1.6865470
## 10  0.015061872 -0.03581918 -0.020202298 -0.0103322687 -0.05110411 1.8291515
## attr(,"scaled:scale")
## [1]  41.79551 384.95494 361.91645 269.52850  26.94087  18.43909
```

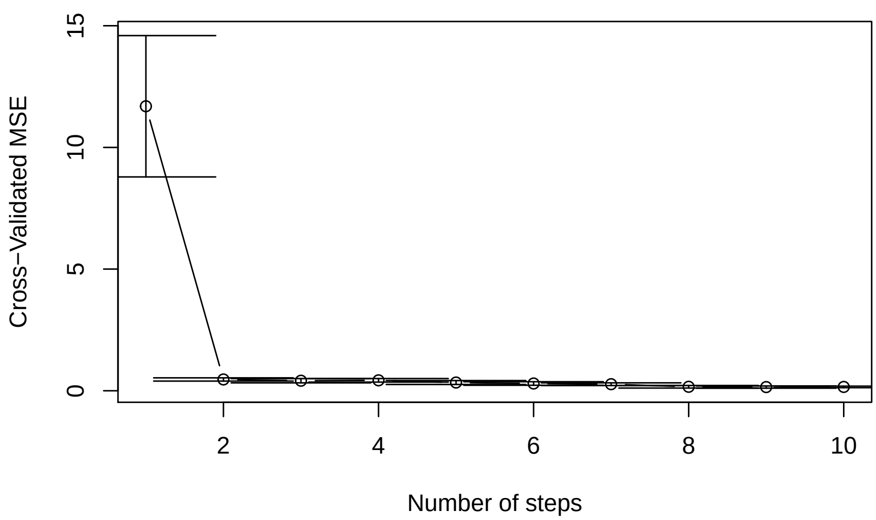
On peut représenter la valeur du Residual Sum of Square en fonction de nombre de variables actives

```
plot(summary(model_lasso)$Df,summary(model_lasso)$Rss,
     xlab='Df',ylab='Rss',main='LASSO')
```




On peut aussi calculer l'erreur quadratique moyenne (MSE) par validation croisée (10-fold ici)

```
cv=cv.lars(as.matrix(longley[,-7]),longley$Employed,K=3,trace=F,plot.it=T,se=T,type=c("lasso"),mode='st
```




On peut afficher le lambda à l'étape minimisant la CV-MSE (ici 2 ?) et les coefficients correspondants

```
print(model_lasso$lambda[4])
```

```
## [1] 0.1949294
```

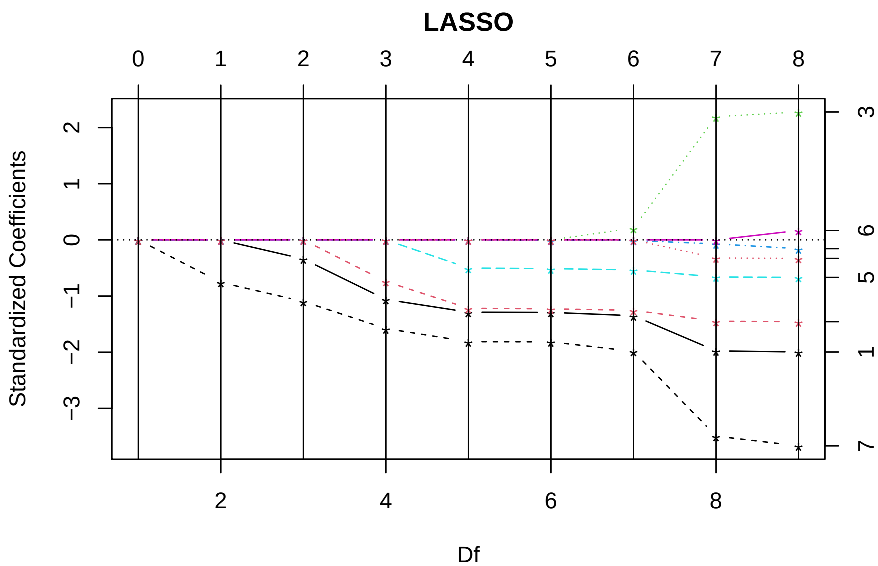
```
print(model_lasso$beta[4,])
```

```
##   GNP.deflator           GNP    Unemployed  Armed.Forces    Population
##   0.0000000000  0.0371660578 -0.0045947533 -0.0009913275  0.0000000000
##           Year
##   0.0000000000
```

La variable GNP est conservée.

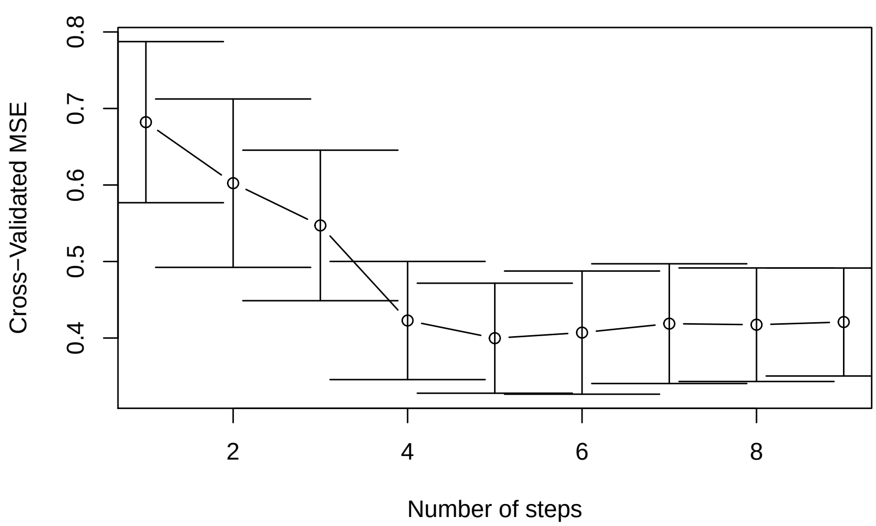
### Regression LASSO sur les données caterpillar

```
library('lars')
model_lasso=lars(as.matrix(caterpillar[,-9]),caterpillar$y,type="lasso",trace=F,normalize=TRUE)
plot(model_lasso,xvar='df', plottype='coeff')
```




On calcule la MSE par validation croisée à chaque étape de l'algo LARS

```
cv=cv.lars(as.matrix(caterpillar[,-9]),caterpillar$y,K=3,trace=F,plot.it=T,se=T,type=c("lasso"),mode='s
```




On choisit généralement l'étape la plus petite (en numéro d'étape) telle que la CVMSE moyenne soit dans l'intervalle de confiance de l'étape conduisant au plus petit CVMSE moyen. Ainsi on aurait un modèle plus parcimonieux avec une CVMSE moyenne non significativement différente de la plus faible observée.

Ici, la plus faible CVMSE moyenne est à l'étape 6. Mais la CVMSE moyenne de l'étape 3 est dans la bande de confiance; On va selectionner la solution de l'étape 3 (attention la première étape dans les sorties est l'étape 0.

```
print(model_lasso$lambda[4])
```

```
## [1] 0.6939303
```

```r
print(model_lasso$beta[4,])
```

```
##          x1          x2          x3          x4          x5          x6 
## -0.001453016 -0.017941130  0.000000000  0.000000000  0.000000000  0.000000000 
##          x7          x8 
## -0.492714400  0.000000000 
```