Applied Bayesian Analysis : NCSU ST 540

Homework 7

Bruce Campbell

In this assignment we perform Bayesian linear regression for the microbiome data on the course website

`https://www4.stat.ncsu.edu/~reich/ABA/assignments/homes.RData`

Let Y_i be the precipitation for observation i and X_{ij} equal one if OTU j is present in sample i.

First, extract the 50 OTU with the largest absolute correlation between X_{ij} and Y_i. Then fit a Bayesian linear regression model precipitation as the response and with these 50 covariates (and an intercept term) using two priors:

(1) Uninformative normal priors: $\beta_j \sim Normal(0, 100^2)$

(2) Hierarchical normal priors: $\beta_j|\tau \sim Normal(0, \tau^2)$ where $\tau^2 \sim InvGamma(0 : 01, 0 : 01)$

(3) Bayesian LASSO: $\beta_j|\tau^2 \sim DE(0, \tau^2)$ where $\tau^2 \sim InvGamma(0 : 01, 0 : 01)$

Compare convergence and the posterior distribution of the regression coeffcients under these three priors. In particular, are the same OTU's significant in all three fits?

Load data and select 50 most ocrrelated OUT variables.

```r
library(rjags)
library(coda)
library(choroplethr)
library(modeest)
load("homes.RData")

X <- OTU != 0
Y <- homes$MeanAnnualPrecipitation

C_xy <- cor(X, Y)

top <- function(x, n) {
    tail(order(x), n)
}

indices <- top(C_xy, 50)

X <- X[, indices]

top.corr <- C_xy[indices]
```

```r
DEBUG <- FALSE
if (DEBUG) {
    nSamples <- 10000
    n.chains <- 4
} else {
    nSamples <- 10000
    n.chains <- 4
}
```

It's not specified what the prior variance is for E[Y_j|X_j]. We wull assume $Y|\beta \sim N(y \cdot \beta, \sigma^2)$ where $\sigma^2 \sim InvGamma(0.1, 0.1)$

```r
n <- nrow(X)

sigma.beta  <- 100
inv.gamma.param  <- 0.1
p <- ncol(X)

model_string.normal_uniformative <- "model{
  # Likelihood
  for(i in 1:n){
    Y[i]   ~ dnorm(mu[i],1/sigma^2)
    mu[i] <- intercept +inprod(X[i,],beta[])
  }

  # Prior for beta
  for(j in 1:p){
    beta[j] ~ dnorm(0,1/sigma.beta^2)
  }
  intercept ~ dnorm(0,1/sigma.beta^2)

  # Prior for the inverse variance
  inv.var   ~ dgamma(inv.gamma.param, inv.gamma.param)
  sigma     <- 1/sqrt(inv.var)
}"

model.normal_uniformative <- jags.model(textConnection(model_string.normal_uniformative), data
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1133
##    Unobserved stochastic nodes: 52
##    Total graph size: 61100
```

```
##
## Initializing model
```

```
update(model.normal_uniformative, nSamples, progress.bar="none"); # Burnin
samp.coeff.normal_uniformative <- coda.samples(model.normal_uniformative, variable.names=c("in
```

(2) Assess convergence of the samplers

In this section we sample from our model after burn in. Although all of the plots are not presented we assesed convergence by; - viewing the time sereies for the intercept and each of the predictors. For this we utilized the coda package. - ran multiple chains and viewed evaluated the autocorrelation plots. - calculated the posterior means for the intercept and the

j - utilized the mlv funtions in the modeest to calculate the MAP estimated of the posterior modes - compared the 95% prediction intervals for the intercepts against the p-values from the logistic regression maximum likelihood model - Gelman plots are produced when not running in DEBUG mode.

Code for this is below, we run some of it conditionlally though the DEBUG variable.

We did run the model without standardizing the feature data and noted evidence that the chain might be experienceing convergence issues. There was significant autocorrelation of the chains when the data was not standardized.

```
summary(samp.coeff.normal_uniformative)
```

```
##
## Iterations = 2001:4000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean    SD Naive SE Time-series SE
## beta[1]    -2.8489 1.844  0.04124        0.12266
## beta[2]    -3.2330 1.877  0.04198        0.12695
## beta[3]    -0.2278 1.709  0.03821        0.10015
## beta[4]     1.9284 1.826  0.04083        0.12859
## beta[5]    -1.9931 1.835  0.04103        0.12107
## beta[6]    -4.1574 1.812  0.04053        0.11666
## beta[7]     1.1805 1.618  0.03618        0.10311
## beta[8]    -0.8069 1.489  0.03330        0.07588
## beta[9]    -1.9368 1.649  0.03687        0.10522
## beta[10]    0.2174 1.606  0.03591        0.10491
## beta[11]   -3.3922 1.790  0.04003        0.12877
## beta[12]   -0.1836 1.489  0.03329        0.06925
## beta[13]    2.2460 1.432  0.03202        0.06848
```

```
## beta[14]   -1.5494 1.573  0.03517        0.08260
## beta[15]    0.1981 1.690  0.03780        0.10577
## beta[16]    1.8569 1.585  0.03545        0.10736
## beta[17]   -1.8976 1.572  0.03515        0.07452
## beta[18]   -2.3273 1.889  0.04223        0.11701
## beta[19]   -4.4410 1.773  0.03964        0.13214
## beta[20]    2.6247 1.846  0.04127        0.11456
## beta[21]   -0.8153 1.789  0.04000        0.11918
## beta[22]   -1.8725 1.747  0.03906        0.11426
## beta[23]    1.3783 1.754  0.03921        0.10969
## beta[24]    3.1451 1.471  0.03288        0.07702
## beta[25]    0.4794 1.586  0.03547        0.08966
## beta[26]    2.6127 1.921  0.04295        0.13333
## beta[27]    1.6758 1.998  0.04467        0.15298
## beta[28]    4.9806 1.883  0.04211        0.12315
## beta[29]   -0.2721 1.583  0.03539        0.07472
## beta[30]   -1.2235 1.809  0.04045        0.11909
## beta[31]    0.3659 1.815  0.04059        0.12260
## beta[32]    1.4548 1.858  0.04154        0.13338
## beta[33]    2.5255 1.615  0.03611        0.11099
## beta[34]    2.6836 1.971  0.04407        0.14800
## beta[35]    3.7790 1.955  0.04371        0.14308
## beta[36]    4.1154 1.440  0.03219        0.08254
## beta[37]    3.9666 1.818  0.04066        0.11757
## beta[38]    2.2187 1.834  0.04101        0.12609
## beta[39]    2.0362 1.874  0.04190        0.11858
## beta[40]    5.6944 1.979  0.04426        0.15987
## beta[41]    2.1483 1.983  0.04434        0.14632
## beta[42]    1.4331 1.709  0.03822        0.12095
## beta[43]    1.0543 1.909  0.04268        0.14282
## beta[44]    5.3726 1.800  0.04025        0.12641
## beta[45]    6.0636 1.710  0.03823        0.10910
## beta[46]    2.6702 1.600  0.03578        0.08877
## beta[47]    1.5828 1.637  0.03661        0.09335
## beta[48]    4.9437 1.484  0.03318        0.07015
## beta[49]    6.7660 1.853  0.04144        0.13316
## beta[50]  -49.3264 8.370  0.18716        3.64808
## intercept 103.4146 8.458  0.18913        3.76796
##
## 2. Quantiles for each variable:
##
##               2.5%      25%      50%      75%     97.5%
## beta[1]    -6.4260  -4.1069  -2.8605  -1.62032   0.79149
## beta[2]    -7.0198  -4.4824  -3.1613  -1.96531   0.43524
## beta[3]    -3.4828  -1.3921  -0.2122   0.95184   3.02888
## beta[4]    -1.7355   0.7286   1.9840   3.08442   5.41860
## beta[5]    -5.5379  -3.2153  -1.9297  -0.72265   1.62382
## beta[6]    -7.7825  -5.3731  -4.1066  -2.94662  -0.75688
```

```
## beta[7]    -2.0285   0.1198   1.1666   2.20717   4.39561
## beta[8]    -3.7982  -1.8132  -0.7978   0.19613   2.01354
## beta[9]    -5.4950  -2.9933  -1.8552  -0.83346   1.12562
## beta[10]   -2.9027  -0.8614   0.2605   1.27450   3.29257
## beta[11]   -7.1090  -4.5669  -3.4078  -2.17348   0.02881
## beta[12]   -3.1434  -1.2085  -0.1634   0.78510   2.69859
## beta[13]   -0.6343   1.2865   2.2839   3.22515   4.98255
## beta[14]   -4.6442  -2.6050  -1.5317  -0.45603   1.42498
## beta[15]   -3.2630  -0.9053   0.1860   1.40144   3.45583
## beta[16]   -1.0728   0.7453   1.8190   2.96560   5.04148
## beta[17]   -5.1057  -2.9298  -1.8911  -0.85246   1.11660
## beta[18]   -5.9384  -3.5918  -2.3103  -1.12535   1.47194
## beta[19]   -7.9034  -5.6792  -4.4495  -3.18253  -1.06689
## beta[20]   -1.1236   1.3592   2.6819   3.90114   6.16150
## beta[21]   -4.4770  -1.9436  -0.7558   0.33604   2.74645
## beta[22]   -5.2758  -3.0816  -1.8751  -0.66682   1.42694
## beta[23]   -1.8328   0.1714   1.2852   2.48432   5.11035
## beta[24]    0.2048   2.1892   3.2048   4.12307   5.98084
## beta[25]   -2.6450  -0.5421   0.4422   1.51060   3.63161
## beta[26]   -1.1030   1.2801   2.6139   3.93068   6.63190
## beta[27]   -2.1057   0.3531   1.6613   3.00349   5.61430
## beta[28]    1.3459   3.6677   4.9723   6.33213   8.51748
## beta[29]   -3.5247  -1.3064  -0.2792   0.82522   2.79763
## beta[30]   -4.5832  -2.4567  -1.3230  -0.00221   2.37303
## beta[31]   -3.1201  -0.8746   0.3953   1.59839   3.92694
## beta[32]   -2.1788   0.1714   1.5862   2.77613   4.84095
## beta[33]   -0.6327   1.4585   2.5138   3.64267   5.69343
## beta[34]   -1.2111   1.3785   2.7279   4.06146   6.34556
## beta[35]   -0.1717   2.4441   3.7909   5.15861   7.37191
## beta[36]    1.2247   3.2047   4.0831   5.05763   6.89423
## beta[37]    0.4122   2.7268   3.9272   5.15224   7.61034
## beta[38]   -1.4319   0.9876   2.2267   3.42184   5.76735
## beta[39]   -1.4455   0.7138   2.0018   3.37266   5.50363
## beta[40]    1.6752   4.4052   5.7286   7.08240   9.47906
## beta[41]   -1.9606   0.8847   2.1594   3.40312   6.19247
## beta[42]   -2.0912   0.2410   1.5105   2.64362   4.57605
## beta[43]   -2.5143  -0.3010   0.9865   2.33397   4.92574
## beta[44]    1.9633   4.1447   5.3560   6.57900   8.92468
## beta[45]    2.6966   4.9024   6.0892   7.22999   9.29589
## beta[46]   -0.4001   1.6017   2.6011   3.74520   5.83380
## beta[47]   -1.6593   0.4913   1.6271   2.65805   4.72467
## beta[48]    2.1188   3.8956   4.9313   5.96122   7.93445
## beta[49]    3.1610   5.5189   6.7593   7.99422  10.54388
## beta[50]  -61.4969 -55.0335 -51.2113 -46.57698 -30.64654
## intercept  84.3989 100.7813 105.3071 108.97870 115.70364
```

```
# Sample again and estimate posterior
# means and MAP posterior modes.
```

```r
samp.coeff.normal_uniformative.jags <- jags.samples(model.normal_uniformative,
    variable.names = c("intercept", "beta"),
    n.iter = nSamples, progress.bar = "none")
posterior_means.normal_uniformative <- lapply(samp.coeff.normal_uniformative.jags,
    apply, 1, "mean")
pander(posterior_means.normal_uniformative,
    caption = "posterior means second sample")
```

- **beta**: *-2.795*, *-3.379*, *-0.3405*, *1.665*, *-2.086*, *-4.289*, *1.193*, *-1.059*, *-1.834*, *0.1559*, *-3.469*, *-0.2633*, *2.226*, *-1.598*, *0.2251*, *2.008*, *-1.989*, *-2.319*, *-4.355*, *2.509*, *-0.7167*, *-1.729*, *1.044*, *3.206*, *0.4958*, *2.913*, *2.011*, *5.28*, *-0.3127*, *-1.228*, *0.4967*, *1.639*, *2.81*, *2.541*, *3.668*, *4.054*, *3.989*, *2.153*, *1.766*, *5.699*, *1.945*, *1.313*, *1.113*, *5.297*, *6.223*, *2.58*, *1.477*, *5.004*, *7.123* and *-39.81*
- **intercept**: *93.89*

```r
posterior_modes.normal_uniformative <- lapply(samp.coeff.normal_uniformative.jags,
    apply, 1, "mlv")
posterior_modes.normal_uniformative
```

```
## $beta
## $beta[[1]]
## Mode (most likely value): -2.954711
## Bickel's modal skewness: 0.048
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[2]]
## Mode (most likely value): -2.934133
## Bickel's modal skewness: -0.158
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[3]]
## Mode (most likely value): -0.1121311
## Bickel's modal skewness: -0.06
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[4]]
## Mode (most likely value): 1.944746
## Bickel's modal skewness: -0.108
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[5]]
## Mode (most likely value): -1.946002
## Bickel's modal skewness: -0.042
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[6]]
## Mode (most likely value): -4.543172
## Bickel's modal skewness: 0.068
```

```
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[7]]
## Mode (most likely value): 1.35205
## Bickel's modal skewness: -0.064
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[8]]
## Mode (most likely value): -1.229809
## Bickel's modal skewness: 0.07
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[9]]
## Mode (most likely value): -2.036197
## Bickel's modal skewness: 0.088
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[10]]
## Mode (most likely value): 0.09464226
## Bickel's modal skewness: -0.006
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[11]]
## Mode (most likely value): -3.206774
## Bickel's modal skewness: -0.066
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[12]]
## Mode (most likely value): -0.1746224
## Bickel's modal skewness: -0.032
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[13]]
## Mode (most likely value): 2.445108
## Bickel's modal skewness: -0.086
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[14]]
## Mode (most likely value): -1.519938
## Bickel's modal skewness: -0.026
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[15]]
## Mode (most likely value): 0.07355763
## Bickel's modal skewness: 0.062
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[16]]
```

```
## Mode (most likely value): 2.313961
## Bickel's modal skewness: -0.13
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[17]]
## Mode (most likely value): -2.147015
## Bickel's modal skewness: 0.072
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[18]]
## Mode (most likely value): -2.712556
## Bickel's modal skewness: 0.182
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[19]]
## Mode (most likely value): -4.199663
## Bickel's modal skewness: -0.042
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[20]]
## Mode (most likely value): 2.660578
## Bickel's modal skewness: -0.072
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[21]]
## Mode (most likely value): -0.7449598
## Bickel's modal skewness: 0.014
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[22]]
## Mode (most likely value): -1.461026
## Bickel's modal skewness: -0.036
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[23]]
## Mode (most likely value): 0.8955537
## Bickel's modal skewness: 0.076
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[24]]
## Mode (most likely value): 3.045927
## Bickel's modal skewness: 0.064
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[25]]
## Mode (most likely value): 0.638692
## Bickel's modal skewness: -0.084
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
```

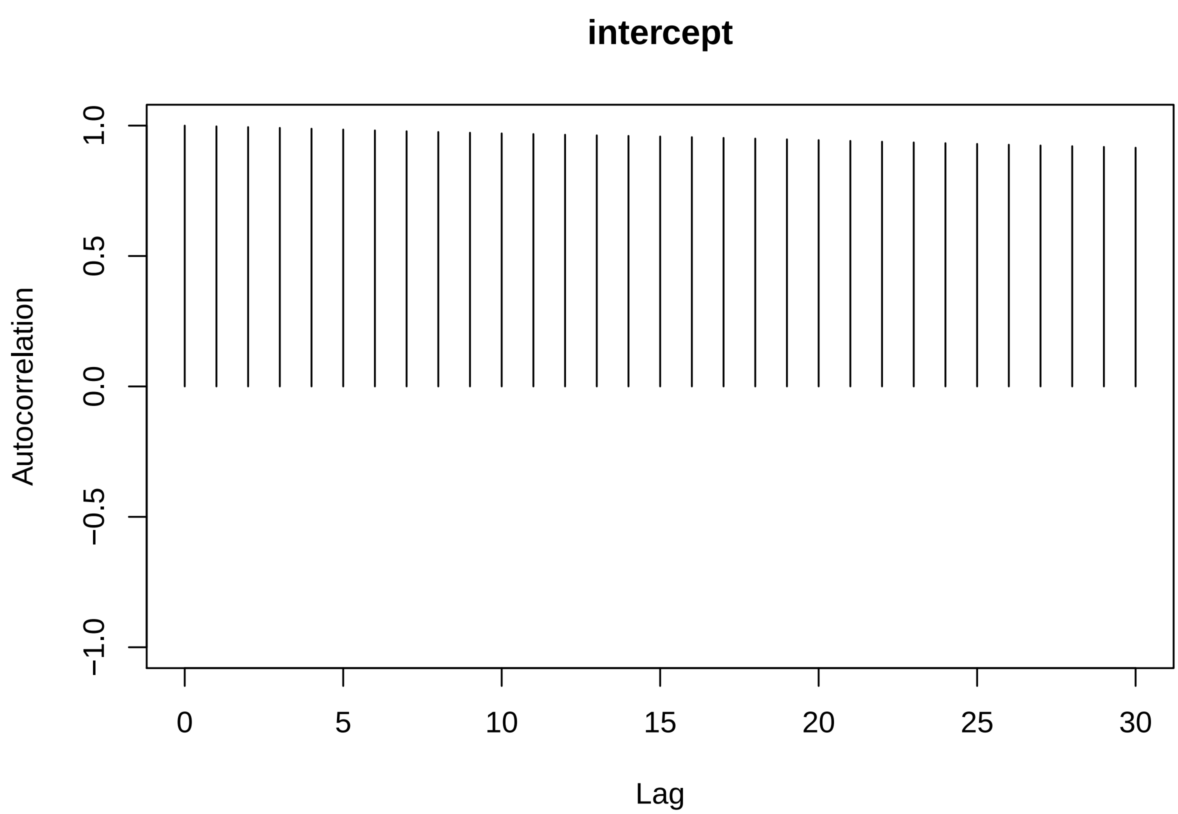
```
##
## $beta[[26]]
## Mode (most likely value): 2.503911
## Bickel's modal skewness: 0.17
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[27]]
## Mode (most likely value): 1.911481
## Bickel's modal skewness: 0.006
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[28]]
## Mode (most likely value): 5.39331
## Bickel's modal skewness: -0.024
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[29]]
## Mode (most likely value): 0.006490398
## Bickel's modal skewness: -0.154
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[30]]
## Mode (most likely value): -1.249589
## Bickel's modal skewness: -0.038
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[31]]
## Mode (most likely value): 0.2592415
## Bickel's modal skewness: 0.106
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[32]]
## Mode (most likely value): 1.547179
## Bickel's modal skewness: 0.022
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[33]]
## Mode (most likely value): 2.686196
## Bickel's modal skewness: 0.066
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[34]]
## Mode (most likely value): 2.707916
## Bickel's modal skewness: -0.032
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[35]]
## Mode (most likely value): 3.843206
```

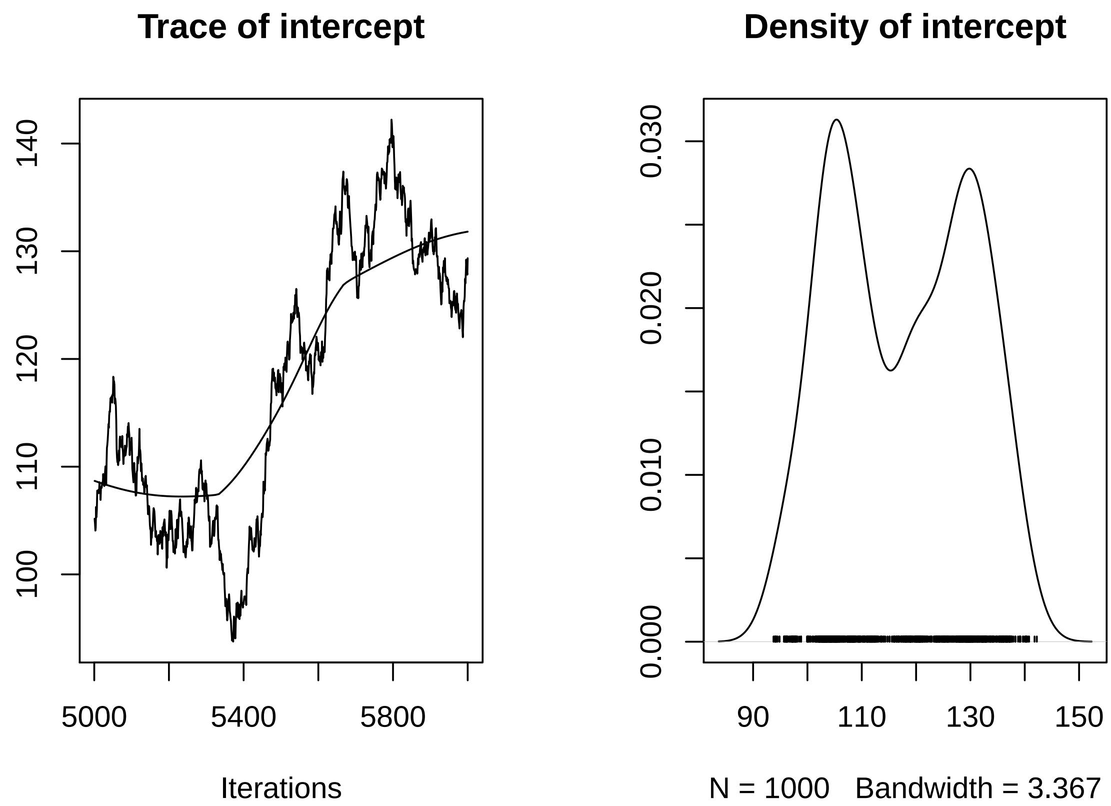
```
## Bickel's modal skewness: -0.056
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[36]]
## Mode (most likely value): 4.009898
## Bickel's modal skewness: 0.062
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[37]]
## Mode (most likely value): 4.134774
## Bickel's modal skewness: -0.082
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[38]]
## Mode (most likely value): 1.941856
## Bickel's modal skewness: 0.07
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[39]]
## Mode (most likely value): 2.008132
## Bickel's modal skewness: -0.076
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[40]]
## Mode (most likely value): 5.928882
## Bickel's modal skewness: -0.082
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[41]]
## Mode (most likely value): 1.854624
## Bickel's modal skewness: 0.048
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[42]]
## Mode (most likely value): 1.145895
## Bickel's modal skewness: 0.084
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[43]]
## Mode (most likely value): 0.9437463
## Bickel's modal skewness: 0.05
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[44]]
## Mode (most likely value): 5.139819
## Bickel's modal skewness: 0.066
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
```

```
## $beta[[45]]
## Mode (most likely value): 6.648289
## Bickel's modal skewness: -0.114
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[46]]
## Mode (most likely value): 2.429394
## Bickel's modal skewness: 0.042
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[47]]
## Mode (most likely value): 1.355147
## Bickel's modal skewness: 0.062
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[48]]
## Mode (most likely value): 5.080308
## Bickel's modal skewness: -0.028
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[49]]
## Mode (most likely value): 7.16686
## Bickel's modal skewness: -0.032
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[50]]
## Mode (most likely value): -38.979
## Bickel's modal skewness: -0.072
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
##
## $intercept
## $intercept[[1]]
## Mode (most likely value): 91.8622
## Bickel's modal skewness: 0.274
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
```

```
if (DEBUG) {
    for (i in 1:p) {
        samp.coeff <- coda.samples(model.normal_uniformative,
            variable.names = c(paste("beta[",
                i, "]", sep = "")), n.iter = nSamples,
            progress.bar = "none")
        autocorr.plot(samp.coeff)
        plot(samp.coeff)
    }
    samp.coeff <- coda.samples(model.normal_uniformative,
        variable.names = "intercept", n.iter = nSamples,
```

```
        progress.bar = "none")
    autocorr.plot(samp.coeff)
    if (n.chains > 1) {
        gelman.plot(samp.coeff)
    }
    plot(samp.coeff)
} else {
    samp.coeff <- coda.samples(model.normal_uniformative,
        variable.names = "intercept", n.iter = nSamples,
        progress.bar = "none")
    autocorr.plot(samp.coeff)
    if (n.chains > 1) {
        gelman.plot(samp.coeff)
    }
    plot(samp.coeff)
}
```





Hierarchical Normal Priors

$\beta_j|\tau \sim Normal(0, \tau^2)$ where $\tau^2 \sim InvGamma(0:01, 0:01)$

```
beta.inv.gamma.param  <- 0.01
variance.inv.gamma.param  <- 0.1
p <- ncol(X)

model_string.normal_hierarchical <- "model{
  # Likelihood
  for(i in 1:n){
    Y[i]    ~ dnorm(mu[i],1/sigma^2)
    mu[i] <- intercept +inprod(X[i,],beta[])
  }

  # Prior for beta
  for(j in 1:p){
    beta[j] ~ dnorm(0,beta.inv.gamma.param)
  }
  intercept ~ dnorm(0,beta.inv.gamma.param)

  # Prior for the inverse variance
```

```
  inv.var   ~ dgamma(variance.inv.gamma.param, variance.inv.gamma.param)
  sigma     <- 1/sqrt(inv.var)

  #Beta Prior for the inverse variance
  inv.var.beta   ~ dgamma(beta.inv.gamma.param, beta.inv.gamma.param)
}"

model.normal_hierarchical <- jags.model(textConnection(model_string.normal_hierarchical), data
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1133
##    Unobserved stochastic nodes: 53
##    Total graph size: 61099
##
## Initializing model
```

```
update(model.normal_hierarchical, nSamples, progress.bar="none"); # Burnin
samp.coeff.normal_hierarchical <- coda.samples(model.normal_hierarchical, variable.names=c("in
summary(samp.coeff.normal_hierarchical)
```

```
##
## Iterations = 2001:4000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean    SD Naive SE Time-series SE
## beta[1]   -2.6820 2.035  0.04550        0.14827
## beta[2]   -3.0371 1.827  0.04085        0.11025
## beta[3]   -0.1559 1.781  0.03982        0.09315
## beta[4]    1.6938 1.728  0.03865        0.11512
## beta[5]   -1.8104 1.894  0.04235        0.12066
## beta[6]   -4.0489 1.795  0.04013        0.10177
## beta[7]    1.1494 1.544  0.03453        0.09380
## beta[8]   -0.8696 1.487  0.03325        0.07386
## beta[9]   -1.6727 1.585  0.03543        0.09725
## beta[10]   0.1734 1.595  0.03567        0.10033
## beta[11]  -3.4598 1.752  0.03918        0.12284
## beta[12]  -0.2993 1.430  0.03198        0.06316
## beta[13]   2.2931 1.408  0.03148        0.06787
## beta[14]  -1.4014 1.566  0.03501        0.08130
## beta[15]   0.2793 1.744  0.03901        0.09917
```

```
## beta[16]   2.0443 1.648  0.03685        0.11165
## beta[17]  -1.9140 1.558  0.03484        0.07314
## beta[18]  -2.4577 1.783  0.03986        0.10051
## beta[19]  -3.7719 1.707  0.03818        0.13093
## beta[20]   2.6851 1.772  0.03963        0.10638
## beta[21]  -0.9284 1.730  0.03868        0.11071
## beta[22]  -1.6709 1.775  0.03970        0.12164
## beta[23]   1.3162 1.717  0.03840        0.11290
## beta[24]   3.0697 1.405  0.03141        0.06651
## beta[25]   0.5011 1.559  0.03485        0.09071
## beta[26]   2.6620 1.777  0.03974        0.11596
## beta[27]   1.6693 1.822  0.04074        0.12534
## beta[28]   5.1334 1.721  0.03848        0.10130
## beta[29]  -0.1447 1.501  0.03355        0.06794
## beta[30]  -1.1591 1.628  0.03641        0.09114
## beta[31]   0.4357 1.826  0.04083        0.12238
## beta[32]   1.3150 1.839  0.04112        0.11119
## beta[33]   2.6901 1.622  0.03628        0.11285
## beta[34]   2.4836 1.937  0.04330        0.14937
## beta[35]   3.4551 1.853  0.04143        0.12453
## beta[36]   4.1066 1.455  0.03253        0.09050
## beta[37]   3.8038 1.918  0.04288        0.13621
## beta[38]   1.9342 1.680  0.03757        0.08971
## beta[39]   1.8858 1.686  0.03769        0.09926
## beta[40]   5.2989 1.902  0.04253        0.14591
## beta[41]   2.1729 1.761  0.03937        0.11908
## beta[42]   1.2333 1.627  0.03639        0.10937
## beta[43]   1.0591 1.751  0.03915        0.11711
## beta[44]   5.4368 1.764  0.03944        0.12421
## beta[45]   6.3605 1.634  0.03654        0.09798
## beta[46]   2.6921 1.531  0.03423        0.06903
## beta[47]   1.5151 1.625  0.03633        0.09178
## beta[48]   4.7584 1.596  0.03568        0.07784
## beta[49]   6.7427 1.717  0.03839        0.11906
## beta[50]  22.4725 7.264  0.16242        2.82230
## intercept 31.2501 7.372  0.16484        2.71970
##
## 2. Quantiles for each variable:
##
##              2.5%      25%     50%      75%   97.5%
## beta[1]   -6.7776 -3.99352 -2.6945 -1.30448  1.3534
## beta[2]   -6.7135 -4.27056 -2.9461 -1.69931  0.2823
## beta[3]   -3.5200 -1.44776 -0.1323  1.08054  3.2489
## beta[4]   -1.6208  0.51482  1.7329  2.87180  4.9996
## beta[5]   -5.6269 -3.08310 -1.7884 -0.58068  2.0459
## beta[6]   -7.6320 -5.26688 -3.9738 -2.81153 -0.6083
## beta[7]   -1.9104  0.07543  1.1630  2.19919  4.2134
## beta[8]   -3.7986 -1.85443 -0.8522  0.12172  2.0221
```

```
## beta[9]   -4.7904 -2.74120 -1.7277 -0.59638  1.4917
## beta[10]  -3.0344 -0.93108  0.1918  1.28718  3.1959
## beta[11]  -6.8930 -4.63813 -3.4227 -2.25010 -0.1931
## beta[12]  -3.0970 -1.30235 -0.3195  0.69408  2.5258
## beta[13]  -0.6148  1.40063  2.2819  3.23948  4.9413
## beta[14]  -4.5142 -2.43035 -1.4009 -0.34531  1.6437
## beta[15]  -3.0808 -0.91305  0.2993  1.44316  3.6628
## beta[16]  -1.2199  0.90235  2.0576  3.20041  5.1014
## beta[17]  -5.0096 -2.92695 -1.9403 -0.85700  1.0627
## beta[18]  -5.9707 -3.66414 -2.4176 -1.14974  0.8258
## beta[19]  -7.2985 -4.87924 -3.7510 -2.61798 -0.4845
## beta[20]  -0.8724  1.53425  2.7354  3.91824  5.9606
## beta[21]  -4.1727 -2.18904 -0.9004  0.29842  2.4088
## beta[22]  -5.0515 -2.88797 -1.7052 -0.56491  2.0508
## beta[23]  -2.0135  0.10962  1.3567  2.48730  4.5593
## beta[24]   0.1563  2.13657  3.0832  4.02644  5.7194
## beta[25]  -2.4915 -0.56106  0.4552  1.59869  3.5464
## beta[26]  -0.6118  1.45469  2.6089  3.75735  6.2900
## beta[27]  -1.9659  0.41845  1.6815  2.93401  5.2536
## beta[28]   1.7716  3.92235  5.1296  6.28408  8.5088
## beta[29]  -3.0858 -1.13956 -0.1604  0.84229  2.8390
## beta[30]  -4.3354 -2.23548 -1.1602 -0.06534  1.8951
## beta[31]  -3.2007 -0.80389  0.4295  1.72514  3.9683
## beta[32]  -2.5375  0.10161  1.4227  2.62630  4.4469
## beta[33]  -0.5907  1.63352  2.6683  3.79342  5.8672
## beta[34]  -1.4141  1.15694  2.5048  3.78020  6.2688
## beta[35]  -0.1908  2.20930  3.4753  4.71376  7.0814
## beta[36]   1.2584  3.13739  4.1082  5.12206  6.9618
## beta[37]   0.2002  2.56955  3.7483  4.95198  7.8013
## beta[38]  -1.1981  0.74212  1.9170  3.09142  5.2016
## beta[39]  -1.4276  0.73037  1.9260  2.99549  5.2056
## beta[40]   1.6571  3.97255  5.3589  6.61185  8.9691
## beta[41]  -1.0889  0.93240  2.1783  3.34421  5.5630
## beta[42]  -2.0910  0.20015  1.2180  2.27075  4.4040
## beta[43]  -2.3937 -0.14858  1.0726  2.31261  4.3670
## beta[44]   1.9560  4.20324  5.5173  6.63265  8.7345
## beta[45]   3.3162  5.21379  6.3320  7.41985  9.6820
## beta[46]  -0.3442  1.64911  2.6821  3.74668  5.6993
## beta[47]  -1.6657  0.43198  1.5454  2.64021  4.5101
## beta[48]   1.5587  3.72427  4.7734  5.79935  7.9470
## beta[49]   3.4000  5.53727  6.7212  7.88454 10.2627
## beta[50]  10.1180 17.74788 21.9452 26.78936 39.7504
## intercept 13.5851 26.66741 31.8178 36.39551 43.5359
```

```r
#Sample again and estimate posterior means and MAP posterior modes.
samp.coeff.normal_hierarchical.jags <- jags.samples(model.normal_hierarchical, variable.names =
posterior_means.normal_hierarchical <- lapply(samp.coeff.normal_hierarchical.jags, apply, 1, "m
pander(posterior_means.normal_hierarchical, caption = "posterior means second sample")
```

- **beta**: *-2.93*, *-2.945*, *0.08475*, *1.935*, *-1.782*, *-4.205*, *1.135*, *-0.8915*, *-1.602*, *0.2209*, *-3.292*, *-0.2933*, *2.251*, *-1.338*, *0.3753*, *2.057*, *-1.812*, *-2.339*, *-3.749*, *2.621*, *-0.7913*, *-1.735*, *1.342*, *2.898*, *0.3351*, *2.813*, *1.577*, *4.985*, *-0.2655*, *-1.296*, *0.3466*, *1.359*, *2.85*, *2.401*, *3.649*, *4.114*, *3.952*, *2.242*, *1.78*, *5.207*, *2.239*, *1.413*, *1.024*, *4.981*, *6.014*, *2.491*, *1.451*, *4.903*, *6.68* and *30.89*
- **intercept**: *22.94*

```
posterior_modes.normal_hierarchical <- lapply(samp.coeff.normal_hierarchical.jags, apply, 1, "m
posterior_modes.normal_hierarchical
```

```
## $beta
## $beta[[1]]
## Mode (most likely value): -2.342992
## Bickel's modal skewness: -0.188
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[2]]
## Mode (most likely value): -2.97203
## Bickel's modal skewness: 0.018
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[3]]
## Mode (most likely value): 0.3104658
## Bickel's modal skewness: -0.102
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[4]]
## Mode (most likely value): 2.569238
## Bickel's modal skewness: -0.236
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[5]]
## Mode (most likely value): -1.633439
## Bickel's modal skewness: -0.012
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[6]]
## Mode (most likely value): -4.347219
## Bickel's modal skewness: 0.014
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[7]]
## Mode (most likely value): 1.26297
## Bickel's modal skewness: -0.066
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[8]]
## Mode (most likely value): -0.5949849
```

```
## Bickel's modal skewness: -0.112
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[9]]
## Mode (most likely value): -1.452313
## Bickel's modal skewness: -0.046
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[10]]
## Mode (most likely value): 0.25537
## Bickel's modal skewness: 0.002
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[11]]
## Mode (most likely value): -3.769336
## Bickel's modal skewness: 0.208
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[12]]
## Mode (most likely value): -0.1955757
## Bickel's modal skewness: -0.034
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[13]]
## Mode (most likely value): 1.96648
## Bickel's modal skewness: 0.174
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[14]]
## Mode (most likely value): -1.681872
## Bickel's modal skewness: 0.14
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[15]]
## Mode (most likely value): 0.35368
## Bickel's modal skewness: 0.014
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[16]]
## Mode (most likely value): 1.699989
## Bickel's modal skewness: 0.156
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[17]]
## Mode (most likely value): -1.781481
## Bickel's modal skewness: 0.006
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
```

```
## $beta[[18]]
## Mode (most likely value): -2.209299
## Bickel's modal skewness: -0.054
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[19]]
## Mode (most likely value): -3.58371
## Bickel's modal skewness: -0.03
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[20]]
## Mode (most likely value): 2.828371
## Bickel's modal skewness: -0.088
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[21]]
## Mode (most likely value): -0.4531187
## Bickel's modal skewness: -0.132
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[22]]
## Mode (most likely value): -1.541612
## Bickel's modal skewness: -0.048
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[23]]
## Mode (most likely value): 0.9764824
## Bickel's modal skewness: 0.154
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[24]]
## Mode (most likely value): 3.044333
## Bickel's modal skewness: -0.024
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[25]]
## Mode (most likely value): 0.06150933
## Bickel's modal skewness: 0.104
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[26]]
## Mode (most likely value): 2.510913
## Bickel's modal skewness: 0.086
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[27]]
## Mode (most likely value): 1.638669
## Bickel's modal skewness: 0.022
```

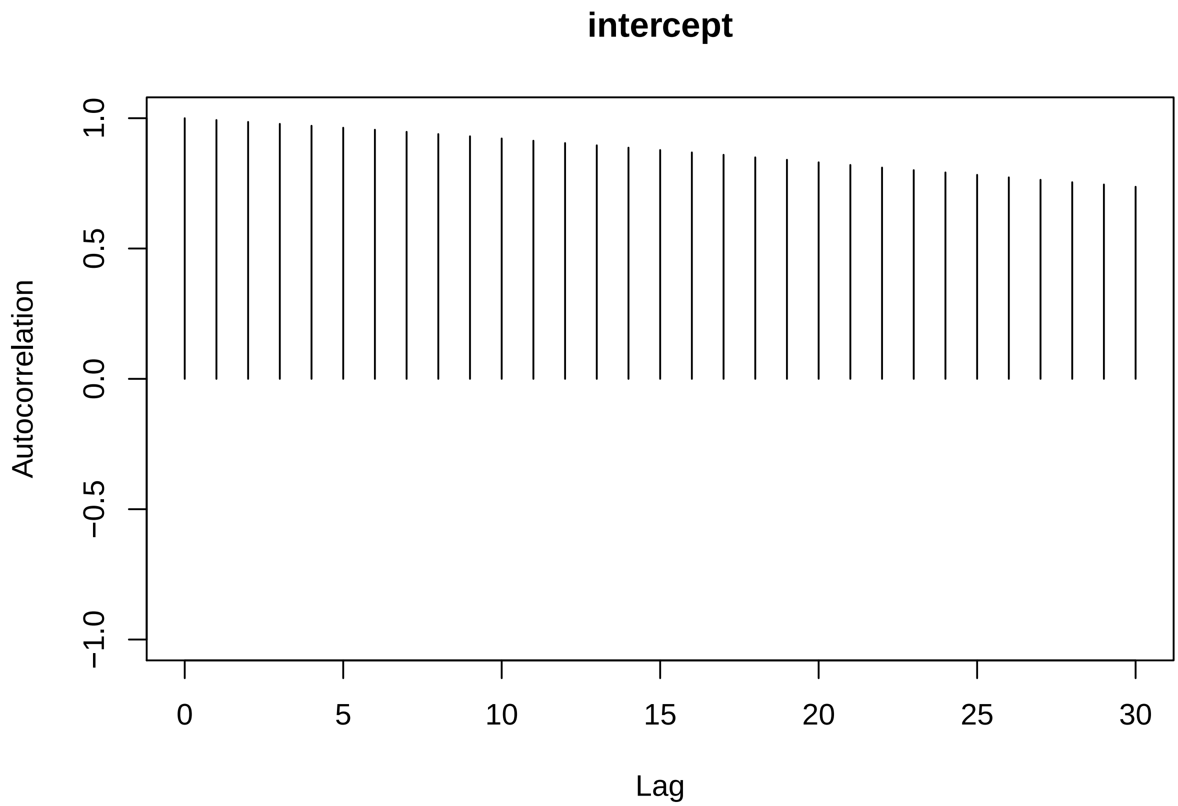
```
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[28]]
## Mode (most likely value): 5.102038
## Bickel's modal skewness: 0
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[29]]
## Mode (most likely value): -0.3772098
## Bickel's modal skewness: 0.058
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[30]]
## Mode (most likely value): -1.219673
## Bickel's modal skewness: -0.004
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[31]]
## Mode (most likely value): 0.02633409
## Bickel's modal skewness: 0.124
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[32]]
## Mode (most likely value): 1.715402
## Bickel's modal skewness: -0.116
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[33]]
## Mode (most likely value): 2.965544
## Bickel's modal skewness: -0.046
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[34]]
## Mode (most likely value): 2.491417
## Bickel's modal skewness: -0.052
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[35]]
## Mode (most likely value): 3.584443
## Bickel's modal skewness: 0.018
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[36]]
## Mode (most likely value): 4.00628
## Bickel's modal skewness: 0.058
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[37]]
```

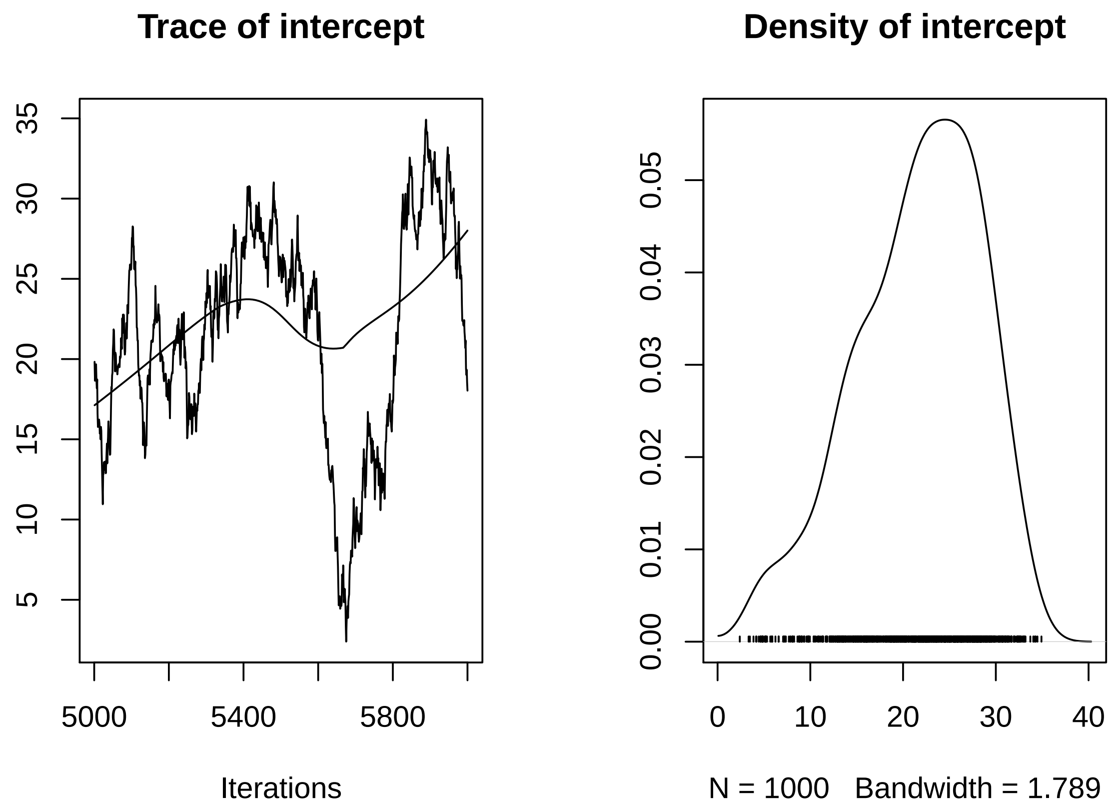
```
## Mode (most likely value): 3.812302
## Bickel's modal skewness: 0.036
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[38]]
## Mode (most likely value): 2.152922
## Bickel's modal skewness: 0.038
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[39]]
## Mode (most likely value): 1.731407
## Bickel's modal skewness: 0.016
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[40]]
## Mode (most likely value): 4.709169
## Bickel's modal skewness: 0.194
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[41]]
## Mode (most likely value): 2.065226
## Bickel's modal skewness: 0.062
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[42]]
## Mode (most likely value): 1.533204
## Bickel's modal skewness: -0.086
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[43]]
## Mode (most likely value): 0.7429706
## Bickel's modal skewness: 0.108
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[44]]
## Mode (most likely value): 4.513824
## Bickel's modal skewness: 0.178
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[45]]
## Mode (most likely value): 6.079562
## Bickel's modal skewness: -0.022
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[46]]
## Mode (most likely value): 2.465076
## Bickel's modal skewness: 0.056
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
```

```
##
## $beta[[47]]
## Mode (most likely value): 1.643477
## Bickel's modal skewness: -0.07
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[48]]
## Mode (most likely value): 5.171392
## Bickel's modal skewness: -0.096
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[49]]
## Mode (most likely value): 6.311706
## Bickel's modal skewness: 0.116
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[50]]
## Mode (most likely value): 32.98444
## Bickel's modal skewness: -0.164
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
##
## $intercept
## $intercept[[1]]
## Mode (most likely value): 22.55279
## Bickel's modal skewness: -0.02
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
```

```
if (DEBUG) {
for (i in 1:p) {
samp.coeff <- coda.samples(model.normal_hierarchical, variable.names = c(paste("beta[",i, "]",
autocorr.plot(samp.coeff)
plot(samp.coeff)
}
samp.coeff <- coda.samples(model.normal_hierarchical, variable.names = "intercept",n.iter = nSa
autocorr.plot(samp.coeff)
if(n.chains>1)
  {
  gelman.plot(samp.coeff)
  }
plot(samp.coeff)
} else {
samp.coeff <- coda.samples(model.normal_hierarchical, variable.names = "intercept",n.iter = nSa
autocorr.plot(samp.coeff)
if(n.chains>1)
  {
  gelman.plot(samp.coeff)
  }
```

```r
plot(samp.coeff)
}
```





BLASSO

```
beta.inv.gamma.param  <- 0.01
variance.inv.gamma.param  <- 0.1
p <- ncol(X)

model_string.normal_blasso <- "model{
  # Likelihood
  for(i in 1:n){
    Y[i]    ~ dnorm(mu[i],1/sigma^2)
    mu[i] <- intercept +inprod(X[i,],beta[])
  }

  # Prior for beta
  for(j in 1:p){
    beta[j] ~ ddexp(0,beta.inv.gamma.param)
  }
  intercept ~ ddexp(0,beta.inv.gamma.param)

  # Prior for the inverse variance
  inv.var   ~ dgamma(variance.inv.gamma.param, variance.inv.gamma.param)
```

```r
  sigma     <- 1/sqrt(inv.var)

  #Beta Prior for the inverse variance
  inv.var.beta   ~ dgamma(beta.inv.gamma.param, beta.inv.gamma.param)
}"

model.normal_blasso <- jags.model(textConnection(model_string.normal_blasso), data = list(Y=Y,
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1133
##    Unobserved stochastic nodes: 53
##    Total graph size: 61099
##
## Initializing model
```

```r
update(model.normal_blasso, nSamples, progress.bar="none"); # Burnin
samp.coeff.normal_blasso <- coda.samples(model.normal_blasso, variable.names=c("intercept","be
summary(samp.coeff.normal_blasso)
```

```
##
## Iterations = 2001:4000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean    SD Naive SE Time-series SE
## beta[1]   -2.6055 2.021  0.04518        0.16688
## beta[2]   -3.3741 1.787  0.03997        0.13177
## beta[3]   -0.1907 1.711  0.03825        0.11356
## beta[4]    1.5910 1.757  0.03929        0.12980
## beta[5]   -2.1598 1.742  0.03894        0.12048
## beta[6]   -4.1905 1.856  0.04151        0.12669
## beta[7]    1.2061 1.625  0.03634        0.12246
## beta[8]   -0.9414 1.445  0.03232        0.09170
## beta[9]   -2.0587 1.567  0.03504        0.11427
## beta[10]   0.3460 1.594  0.03564        0.11800
## beta[11]  -3.1976 1.782  0.03985        0.14165
## beta[12]  -0.3104 1.456  0.03255        0.07228
## beta[13]   2.2796 1.481  0.03312        0.08054
## beta[14]  -1.4890 1.521  0.03401        0.08828
## beta[15]   0.1566 1.773  0.03964        0.12951
## beta[16]   2.0381 1.567  0.03505        0.11683
```

```
## beta[17]  -1.8632 1.681  0.03759        0.09993
## beta[18]  -2.2563 1.613  0.03606        0.09449
## beta[19]  -4.3708 1.737  0.03883        0.15400
## beta[20]   2.6436 1.790  0.04002        0.12820
## beta[21]  -0.7645 1.822  0.04073        0.14036
## beta[22]  -1.7727 1.797  0.04017        0.13170
## beta[23]   1.1968 1.703  0.03809        0.12372
## beta[24]   3.1252 1.544  0.03452        0.08888
## beta[25]   0.4715 1.571  0.03514        0.11075
## beta[26]   2.6257 1.816  0.04062        0.13366
## beta[27]   1.7508 1.860  0.04159        0.15472
## beta[28]   5.2731 1.796  0.04017        0.13242
## beta[29]  -0.1500 1.488  0.03327        0.07953
## beta[30]  -1.3744 1.676  0.03748        0.11273
## beta[31]   0.3112 1.906  0.04263        0.15681
## beta[32]   1.5358 1.908  0.04266        0.16147
## beta[33]   2.5713 1.651  0.03692        0.14329
## beta[34]   2.5334 2.126  0.04755        0.19931
## beta[35]   3.7397 1.823  0.04077        0.14033
## beta[36]   4.1375 1.341  0.02998        0.08370
## beta[37]   3.8704 1.714  0.03832        0.12452
## beta[38]   2.0112 1.743  0.03898        0.11723
## beta[39]   2.1013 1.858  0.04155        0.14009
## beta[40]   5.5426 1.867  0.04174        0.17181
## beta[41]   2.1741 1.851  0.04139        0.15240
## beta[42]   1.0325 1.635  0.03657        0.13057
## beta[43]   0.9972 1.777  0.03974        0.14515
## beta[44]   5.4620 1.717  0.03839        0.13949
## beta[45]   6.3231 1.701  0.03804        0.12536
## beta[46]   2.6168 1.481  0.03312        0.08950
## beta[47]   1.6631 1.615  0.03611        0.10682
## beta[48]   4.9206 1.584  0.03543        0.09079
## beta[49]   7.0237 1.697  0.03795        0.13908
## beta[50]  49.7248 8.094  0.18100        3.76284
## intercept  4.3460 8.145  0.18213        3.87438
##
## 2. Quantiles for each variable:
##
##               2.5%       25%     50%       75%    97.5%
## beta[1]   -6.58102 -3.970107 -2.6091 -1.244124  1.38672
## beta[2]   -6.84444 -4.642646 -3.3550 -2.110058  0.08212
## beta[3]   -3.44682 -1.356009 -0.2199  0.981470  3.14549
## beta[4]   -1.73368  0.393101  1.6683  2.790388  4.90864
## beta[5]   -5.50850 -3.366946 -2.1989 -0.939177  1.14021
## beta[6]   -7.70525 -5.419473 -4.1679 -3.005513 -0.35468
## beta[7]   -1.84352 -0.027058  1.2228  2.363741  4.27220
## beta[8]   -3.86818 -1.887764 -0.9310  0.008003  2.03968
## beta[9]   -5.22956 -3.109834 -2.0428 -1.059374  1.01690
```

```
## beta[10]  -2.69283 -0.773708  0.3180  1.399031  3.57730
## beta[11]  -6.83446 -4.333758 -3.1463 -2.040282  0.24171
## beta[12]  -3.23400 -1.298848 -0.3232  0.678749  2.50670
## beta[13]  -0.55261  1.299228  2.3121  3.336695  5.11846
## beta[14]  -4.43164 -2.512116 -1.5046 -0.530485  1.60895
## beta[15]  -3.65026 -0.973097  0.2429  1.395345  3.34430
## beta[16]  -1.01564  1.031242  2.0255  3.092779  5.02712
## beta[17]  -5.12208 -3.017979 -1.9132 -0.736616  1.50300
## beta[18]  -5.32668 -3.348680 -2.2770 -1.115638  0.93105
## beta[19]  -7.87034 -5.489219 -4.2852 -3.267101 -0.74065
## beta[20]  -0.91202  1.471221  2.6107  3.859499  6.02511
## beta[21]  -4.07619 -1.991531 -0.8681  0.348353  2.97217
## beta[22]  -5.23200 -2.988252 -1.7875 -0.642313  1.70605
## beta[23]  -2.07599  0.002514  1.1620  2.394544  4.52024
## beta[24]   0.02341  2.051755  3.1992  4.211842  6.01943
## beta[25]  -2.45776 -0.659160  0.4415  1.546070  3.59379
## beta[26]  -0.74202  1.350123  2.6764  3.835029  6.24517
## beta[27]  -1.78398  0.470764  1.6760  2.957402  5.34816
## beta[28]   1.87533  4.012647  5.2661  6.455511  8.84762
## beta[29]  -3.23432 -1.094808 -0.1205  0.868974  2.59320
## beta[30]  -4.56929 -2.501374 -1.4211 -0.291810  2.07490
## beta[31]  -3.30629 -1.004965  0.3174  1.638878  3.95898
## beta[32]  -2.36029  0.262193  1.5635  2.888382  5.06746
## beta[33]  -0.60765  1.416535  2.5576  3.742800  5.75875
## beta[34]  -1.53208  0.999104  2.5475  4.047888  6.72076
## beta[35]   0.18565  2.512892  3.7562  4.917989  7.54800
## beta[36]   1.55706  3.186842  4.1348  5.038162  6.68406
## beta[37]   0.60469  2.712529  3.8103  4.944357  7.41586
## beta[38]  -1.35045  0.803868  1.9874  3.180866  5.56852
## beta[39]  -1.37684  0.795396  2.0896  3.327918  5.79814
## beta[40]   1.79691  4.304085  5.6276  6.702578  9.19128
## beta[41]  -1.43513  0.870192  2.1402  3.469386  5.86582
## beta[42]  -2.16466 -0.052169  0.9932  2.169941  4.31313
## beta[43]  -2.42724 -0.182855  0.9502  2.063031  4.58998
## beta[44]   2.27250  4.303643  5.4316  6.630241  8.87135
## beta[45]   2.95780  5.194123  6.2889  7.450622  9.59248
## beta[46]  -0.31629  1.617743  2.5755  3.589756  5.74382
## beta[47]  -1.42788  0.547218  1.6776  2.778135  4.83713
## beta[48]   1.93562  3.857429  4.8495  5.959546  8.11086
## beta[49]   3.82851  5.845558  6.9586  8.194514 10.41832
## beta[50]  32.97791 43.786195 51.0127 55.968237 62.89287
## intercept -8.92899 -2.333137  3.2623 10.139776 21.20658
```

```r
#Sample again and estimate posterior means and MAP posterior modes.
samp.coeff.normal_blasso.jags <- jags.samples(model.normal_blasso, variable.names = c("intercep
posterior_means.normal_blasso <- lapply(samp.coeff.normal_blasso.jags, apply, 1, "mean")
pander(posterior_means.normal_blasso, caption = "posterior means second sample")
```

- **beta**: *-2.715*, *-3.062*, *-0.3503*, *1.976*, *-1.731*, *-4.038*, *1.048*, *-0.9762*, *-1.638*, *0.2451*, *-3.393*, *-0.1231*, *2.287*, *-1.294*, *0.7206*, *1.91*, *-2.046*, *-2.533*, *-3.929*, *2.524*, *-1.013*, *-1.821*, *1.188*, *2.924*, *0.4639*, *2.808*, *2.096*, *4.798*, *-0.2337*, *-1.382*, *0.3174*, *1.305*, *2.512*, *2.336*, *3.458*, *4.394*, *3.523*, *2.11*, *2.072*, *5.281*, *2.227*, *1.477*, *1.11*, *5.35*, *6.153*, *2.679*, *1.495*, *4.889*, *6.956* and *42.44*
- **intercept**: *11.4*

```
posterior_modes.normal_blasso <- lapply(samp.coeff.normal_blasso.jags, apply, 1, "mlv")
posterior_modes.normal_blasso
```

```
## $beta
## $beta[[1]]
## Mode (most likely value): -2.775477
## Bickel's modal skewness: -0.012
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[2]]
## Mode (most likely value): -2.973553
## Bickel's modal skewness: -0.022
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[3]]
## Mode (most likely value): -0.4984725
## Bickel's modal skewness: 0.066
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[4]]
## Mode (most likely value): 1.857774
## Bickel's modal skewness: 0.016
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[5]]
## Mode (most likely value): -2.120046
## Bickel's modal skewness: 0.17
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[6]]
## Mode (most likely value): -3.77325
## Bickel's modal skewness: -0.098
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[7]]
## Mode (most likely value): 1.119537
## Bickel's modal skewness: -0.036
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[8]]
## Mode (most likely value): -0.8844481
## Bickel's modal skewness: -0.052
```

```
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[9]]
## Mode (most likely value): -1.279489
## Bickel's modal skewness: -0.164
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[10]]
## Mode (most likely value): -0.02734835
## Bickel's modal skewness: 0.106
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[11]]
## Mode (most likely value): -3.505024
## Bickel's modal skewness: 0.036
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[12]]
## Mode (most likely value): 0.09079765
## Bickel's modal skewness: -0.084
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[13]]
## Mode (most likely value): 2.013322
## Bickel's modal skewness: 0.146
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[14]]
## Mode (most likely value): -1.300655
## Bickel's modal skewness: 0.016
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[15]]
## Mode (most likely value): 0.7310424
## Bickel's modal skewness: -0.008
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[16]]
## Mode (most likely value): 1.656517
## Bickel's modal skewness: 0.05
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[17]]
## Mode (most likely value): -2.023985
## Bickel's modal skewness: -0.004
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[18]]
```

```
## Mode (most likely value): -2.615068
## Bickel's modal skewness: -0.002
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[19]]
## Mode (most likely value): -4.109332
## Bickel's modal skewness: 0.124
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[20]]
## Mode (most likely value): 2.554642
## Bickel's modal skewness: 0.026
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[21]]
## Mode (most likely value): -1.285455
## Bickel's modal skewness: 0.092
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[22]]
## Mode (most likely value): -1.899499
## Bickel's modal skewness: 0.002
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[23]]
## Mode (most likely value): 1.05005
## Bickel's modal skewness: 0.052
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[24]]
## Mode (most likely value): 3.11347
## Bickel's modal skewness: -0.096
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[25]]
## Mode (most likely value): -0.02320423
## Bickel's modal skewness: 0.154
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[26]]
## Mode (most likely value): 2.76146
## Bickel's modal skewness: 0.04
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[27]]
## Mode (most likely value): 2.014301
## Bickel's modal skewness: 0.01
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
```

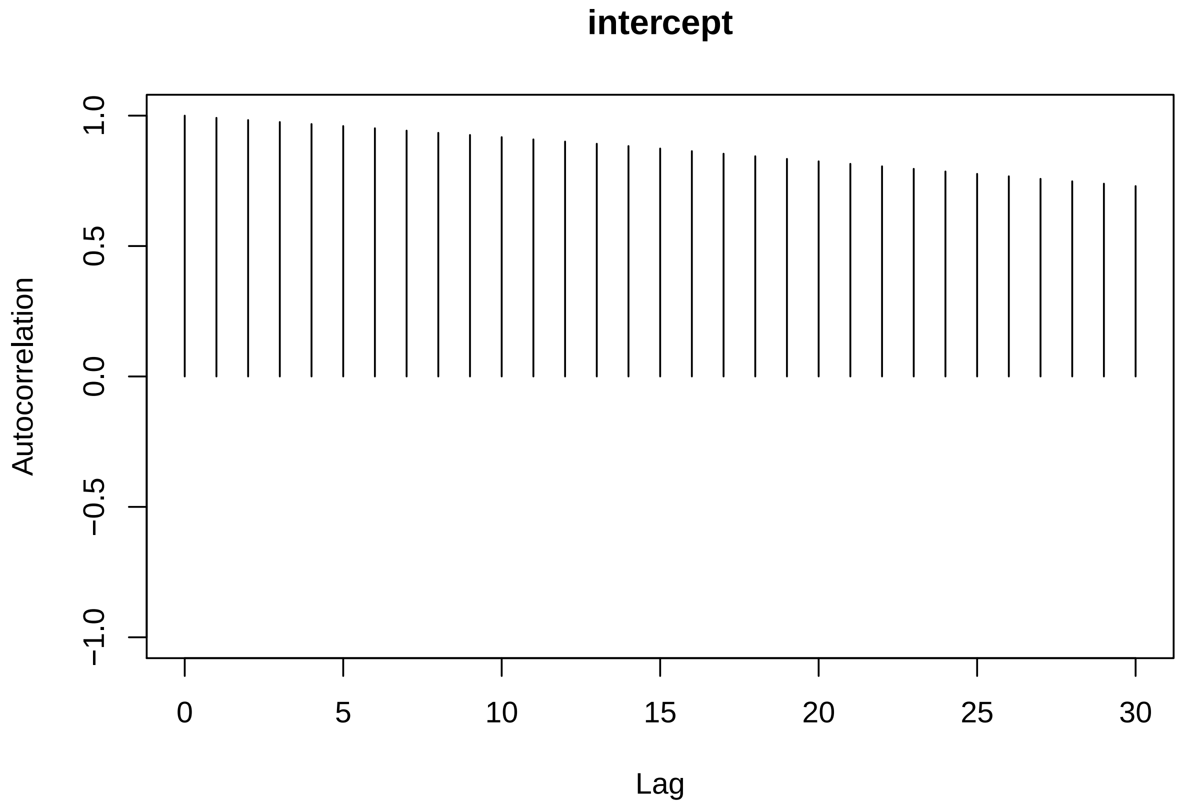
```
##
## $beta[[28]]
## Mode (most likely value): 4.71959
## Bickel's modal skewness: 0.03
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[29]]
## Mode (most likely value): 0.001552122
## Bickel's modal skewness: -0.084
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[30]]
## Mode (most likely value): -1.704171
## Bickel's modal skewness: 0.096
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[31]]
## Mode (most likely value): 0.164121
## Bickel's modal skewness: 0.048
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[32]]
## Mode (most likely value): 1.235591
## Bickel's modal skewness: 0.044
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[33]]
## Mode (most likely value): 2.3492
## Bickel's modal skewness: 0.042
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[34]]
## Mode (most likely value): 2.595862
## Bickel's modal skewness: -0.112
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[35]]
## Mode (most likely value): 3.484187
## Bickel's modal skewness: -0.014
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[36]]
## Mode (most likely value): 4.33955
## Bickel's modal skewness: 0.04
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[37]]
## Mode (most likely value): 3.389913
```

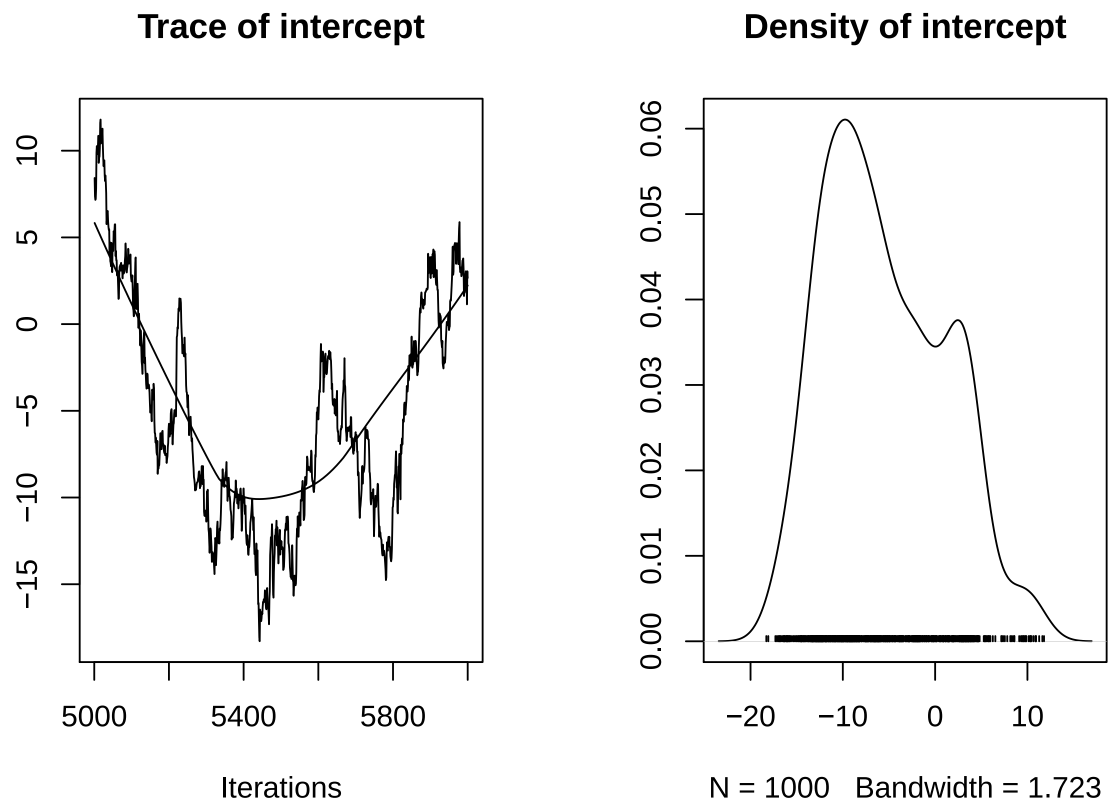
```
## Bickel's modal skewness: 0.08
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[38]]
## Mode (most likely value): 1.92555
## Bickel's modal skewness: 0.084
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[39]]
## Mode (most likely value): 1.954854
## Bickel's modal skewness: 0.03
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[40]]
## Mode (most likely value): 5.145005
## Bickel's modal skewness: 0.046
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[41]]
## Mode (most likely value): 2.351593
## Bickel's modal skewness: -0.052
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[42]]
## Mode (most likely value): 1.371547
## Bickel's modal skewness: 0.004
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[43]]
## Mode (most likely value): 0.9776426
## Bickel's modal skewness: 0.026
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[44]]
## Mode (most likely value): 5.118125
## Bickel's modal skewness: 0.12
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[45]]
## Mode (most likely value): 5.848349
## Bickel's modal skewness: 0.104
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[46]]
## Mode (most likely value): 2.607582
## Bickel's modal skewness: 0.054
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
```

```
## $beta[[47]]
## Mode (most likely value): 1.525134
## Bickel's modal skewness: -0.006
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[48]]
## Mode (most likely value): 4.865547
## Bickel's modal skewness: 0.018
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[49]]
## Mode (most likely value): 6.912887
## Bickel's modal skewness: -0.02
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
## $beta[[50]]
## Mode (most likely value): 35.54019
## Bickel's modal skewness: 0.394
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
##
##
## $intercept
## $intercept[[1]]
## Mode (most likely value): 17.81082
## Bickel's modal skewness: -0.324
## Call: mlv.default(x = array(newX[, i], d.call, dn.call))
```

```
if (DEBUG) {
for (i in 1:p) {
samp.coeff <- coda.samples(model.normal_blasso, variable.names = c(paste("beta[",i, "]", sep =
autocorr.plot(samp.coeff)
plot(samp.coeff)
}
samp.coeff <- coda.samples(model.normal_blasso, variable.names = "intercept",n.iter = nSamples
autocorr.plot(samp.coeff)
if(n.chains>1)
  {
  gelman.plot(samp.coeff)
  }
plot(samp.coeff)
} else {
samp.coeff <- coda.samples(model.normal_blasso, variable.names = "intercept",n.iter = nSamples
autocorr.plot(samp.coeff)
if(n.chains>1)
  {
  gelman.plot(samp.coeff)
  }
plot(samp.coeff)
```

```
}
```


intercept
Autocorrelation
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30
Lag



Trace of intercept
Iterations
Density of intercept
N = 1000 Bandwidth = 1.723