Multiple Linear Regression

We use the data from Barberan (2015), downloaded from http://figshare.com/articles/1000homes/1270900. The data are dust samples from the ledges above doorways from n=1,059 homes (after removing samples with missing data) in the continental US. Bioinformatics processing detects the presence or absence of 763 species (technically operational taxonomic units) of fungi. The response is the log of the number of fungi species present in the sample, which is a measure of species richness. The objective is to determine which factors influence a home's species richness. For each home, eight covariates are included in this example: longitude, latitude, annual mean temperature, annual mean precipitation, net primary productivity (NPP), elevation, the binary indicator that the house is a single-family home, and the number of bedrooms in the home. These covariates are all centered and scaled to have mean zero and variance one.

The Bayesian multiple linear regression model is

$$Y_i \sim \text{Normal}(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j, \sigma^2).$$

Three different priors for the slopes $\beta_1, \ldots, \beta_p$ are chosen for comparison purpose

1. Uninformative Gaussian: $\beta_j \sim \text{Normal}(0, 2000)$
2. Gaussian shrinkage: $\beta_j \sim \text{Normal}(0, \sigma_b^2)$ with $\sigma_b^2 \sim \text{InvGamma}(0.2, 0.2)$
3. Bayesian LASSO: $\beta_j \sim \text{DE}(0, \sigma_b^2)$ with $\sigma_b^2 \sim \text{InvGamma}(0.2, 0.2)$

In all cases, we use uninformative conjugate priors for the intercept $\beta_0 \sim \text{Normal}(0, 2000)$ and variance $\sigma^2 \sim \text{InvGamma}(0.2, 0.2)$

Load and plot the data

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.1
```

```
## Loaded modules: basemod,bugs
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:kableExtra':
##
##     group_rows
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:maps':
##
##     map
```

```
##  [1] "ID"                       "City"
##  [3] "State"                    "Latitude"
##  [5] "Longitude"                "MeanAnnual.Temperature"
##  [7] "MeanAnnualPrecipitation" "NetPrimaryProductivity"
##  [9] "Elevation"                "Residence"
## [11] "NumberBedroooms"
```

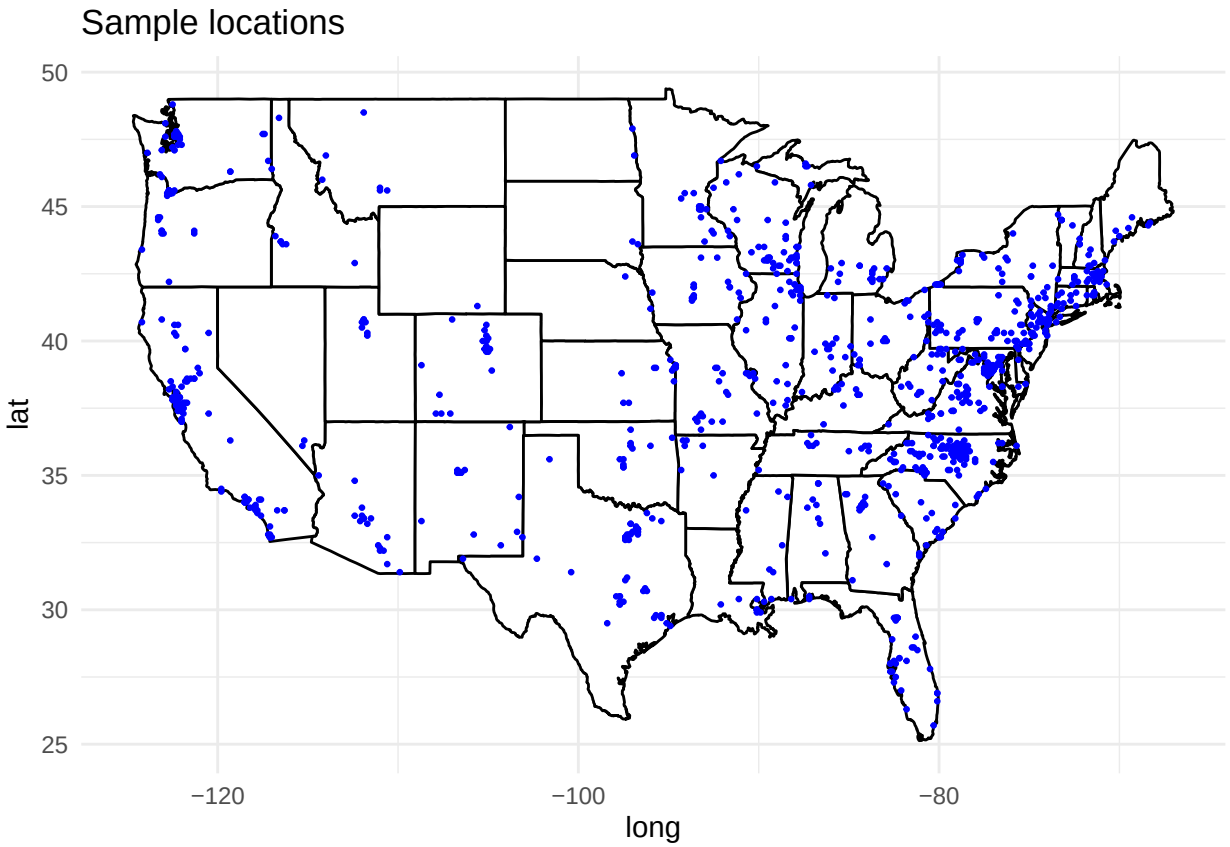
```
##                        ID                      City                     State
##                 "integer"               "character"               "character"
##                  Latitude                 Longitude   MeanAnnual.Temperature
##                 "numeric"                 "numeric"                 "numeric"
## MeanAnnualPrecipitation   NetPrimaryProductivity                 Elevation
##                 "numeric"                 "numeric"                 "numeric"
##                 Residence           NumberBedroooms
##               "character"               "character"
```

```
##Covariates(features)
city     = homes[,2]
state    = homes[,3]
lat      = homes[,4]
long     = homes[,5]
temp     = homes[,6]
precip   = homes[,7]
NPP      = homes[,8]
elev     = homes[,9]
house    = ifelse(homes[,10]=="One-family house detached from any other house",1,0)
bedrooms = as.numeric(homes[,11])
```

```
## Warning: NAs introduced by coercion
```

```
#Data Prep
OTU      = as.matrix(OTU)
nspecies = rowSums(OTU>0)
Y        = log(nspecies)
X        = cbind(long,lat,temp,precip,NPP,elev,house,bedrooms)
names    = c("Longitude","Latitude",
             "Temperature","Precipitation","NPP",
             "Elevation","Single-family home",
             "Number of bedrooms")

#Remove observations with missing data
#Combine the data into a data frame
data    = data.frame(Y, X, city, state)
#Filter out rows with any NA values
complete_rows = complete.cases(data)
```

```
Y = Y[complete_rows]
X = X[complete_rows, ]
city = city[complete_rows]
state = state[complete_rows]

#Standardize the covariates
X = as.matrix(scale(X))

# Plot the sample locations
homes_data = as.data.frame(homes)
states_map =map_data("state")
ggplot() +
  geom_polygon(data = states_map, aes(x = long, y = lat, group = group), fill = "white", color = "black
  geom_point(data = homes_data, aes(x = homes_data[, 5], y = homes_data[, 4]), color = "blue", size = 0
  ggtitle("Sample locations") +
  theme_minimal()
```



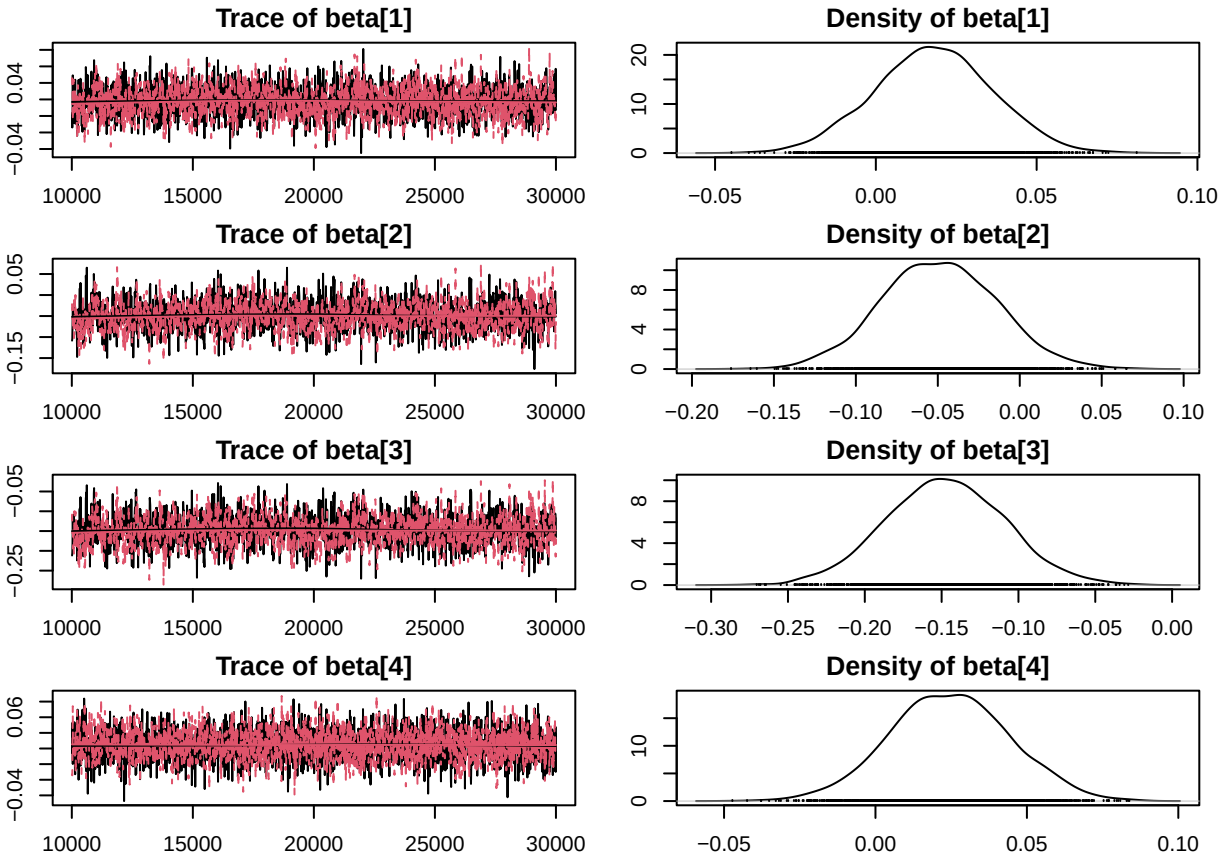
Put the data in JAGS format

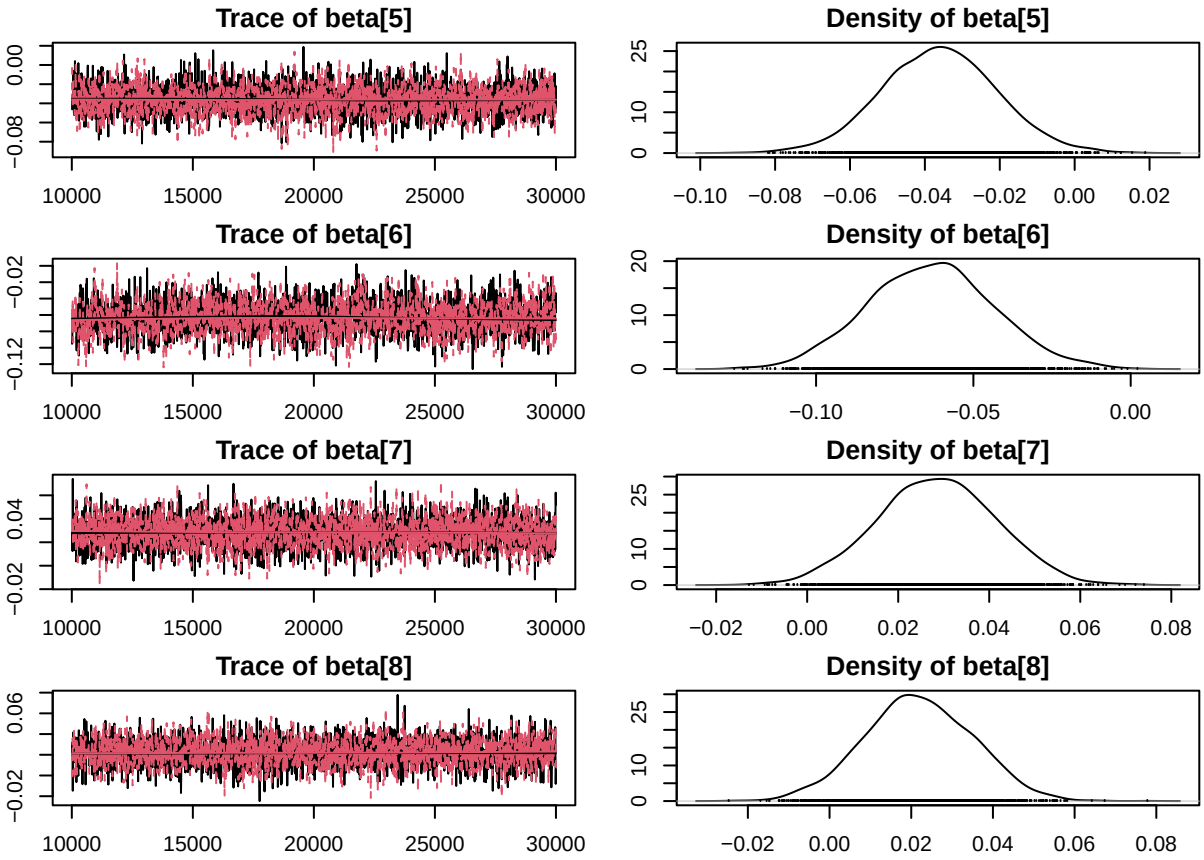
```
n        = length(Y)
p        = ncol(X)
data   = list(Y=Y,X=X,n=n,p=p)
params = c("beta")
```

```
burn     = 10000
n.iter   = 20000
thin     = 10
n.chains = 2 #parallel chains
```

(1) Fit the uninformative Gaussian model

```
model_string = textConnection("model{
  # Likelihood
   for(i in 1:n){
     Y[i] ~ dnorm(alpha+inprod(X[i,],beta[]),taue)
   }
  # Priors
   for(j in 1:p){
     beta[j] ~ dnorm(0,0.001)
   }
   alpha ~ dnorm(0,0.001)
   taue  ~ dgamma(0.1, 0.1)
}")
model = jags.model(model_string,data = data, n.chains=n.chains,quiet=TRUE)
update(model, burn, progress.bar="none")
samples1 = coda.samples(model, variable.names=params, thin=thin, n.iter=n.iter, progress.bar="none")
par(mar=c(2,2,2,2))
plot(samples1)
```


Trace of beta[1]
Density of beta[1]
Trace of beta[2]
Density of beta[2]
Trace of beta[3]
Density of beta[3]
Trace of beta[4]
Density of beta[4]




```
round(effectiveSize(samples1),1)
```

```
## beta[1] beta[2] beta[3] beta[4] beta[5] beta[6] beta[7] beta[8]
##  2271.4  1535.6  1436.2  4020.8  4109.8  2020.7  4208.5  4000.0
```

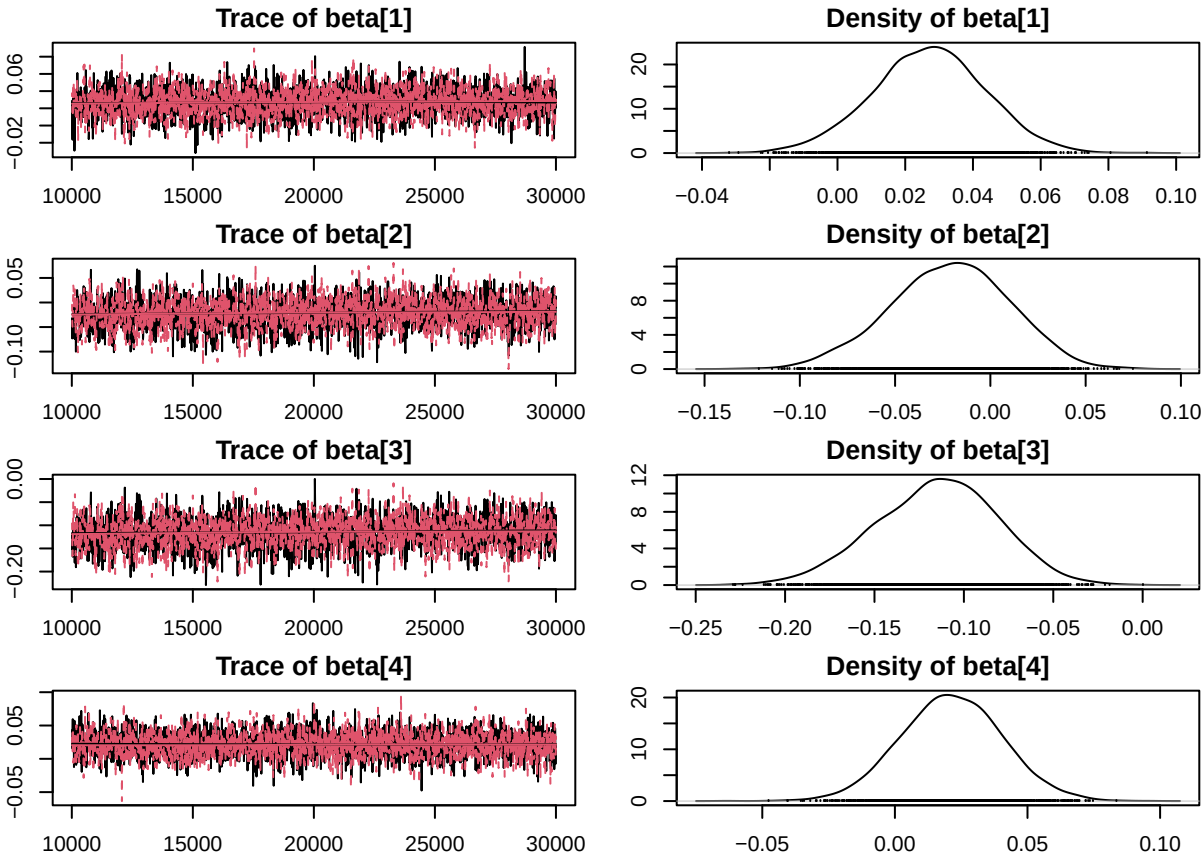
```
sum                       = summary(samples1)
rownames(sum$statistics)  = names
rownames(sum$quantiles)   = names
sum$statistics            = round(sum$statistics,3)
sum$quantiles             = round(sum$quantiles,3)
sum
```

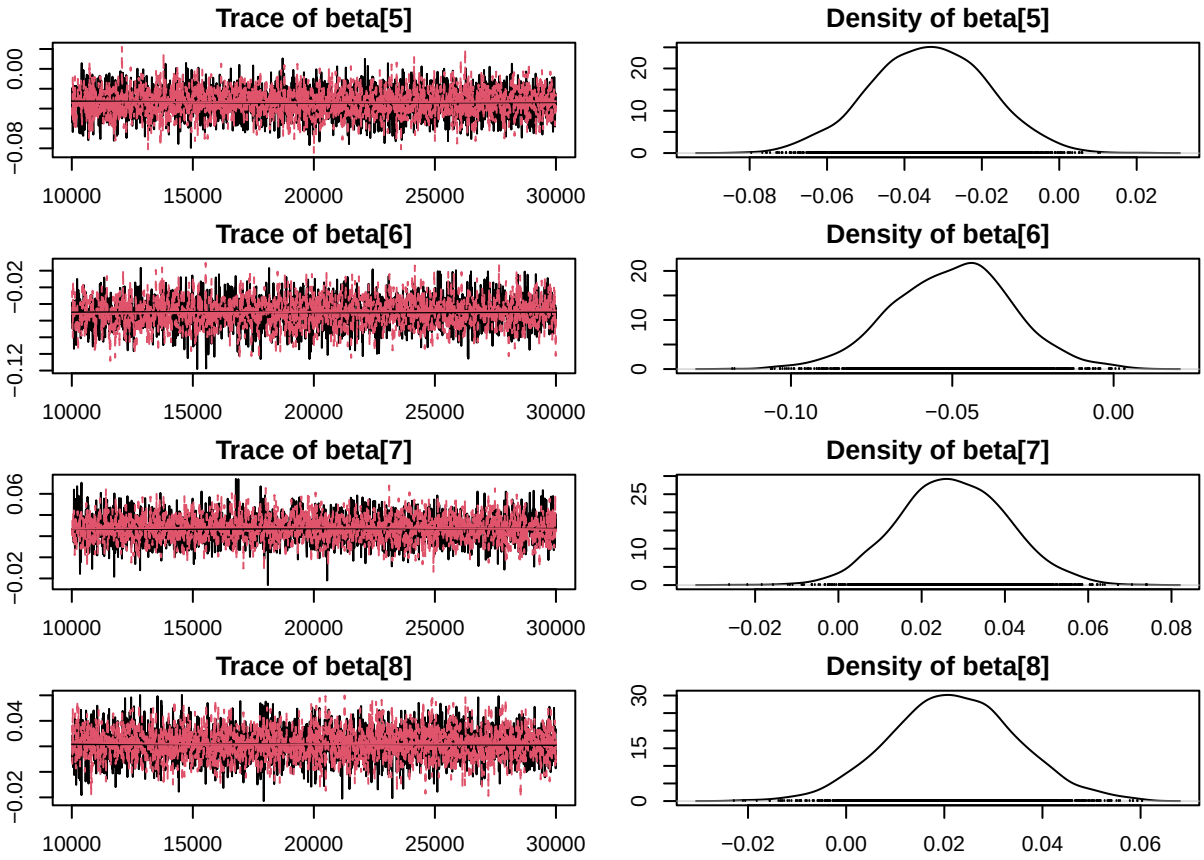
```
##
## Iterations = 10010:30000
## Thinning interval = 10
## Number of chains = 2
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                      Mean    SD Naive SE Time-series SE
## Longitude           0.018 0.018    0.000          0.000
## Latitude           -0.049 0.035    0.001          0.001
## Temperature        -0.148 0.039    0.001          0.001
```

```
## Precipitation       0.025 0.020    0.000          0.000
## NPP                -0.036 0.015    0.000          0.000
## Elevation          -0.063 0.020    0.000          0.000
## Single-family home  0.028 0.013    0.000          0.000
## Number of bedrooms  0.022 0.013    0.000          0.000
##
## 2. Quantiles for each variable:
##
##                       2.5%    25%    50%    75%  97.5%
## Longitude          -0.018  0.006  0.018  0.030  0.053
## Latitude           -0.118 -0.073 -0.049 -0.026  0.021
## Temperature        -0.222 -0.174 -0.148 -0.122 -0.071
## Precipitation      -0.014  0.011  0.024  0.038  0.063
## NPP                -0.066 -0.047 -0.036 -0.026 -0.006
## Elevation          -0.102 -0.077 -0.063 -0.050 -0.023
## Single-family home  0.003  0.020  0.028  0.037  0.054
## Number of bedrooms -0.004  0.013  0.022  0.031  0.047
```

(2) Fit the Gaussian shrinkage model

```
model_string = textConnection("model{
  # Likelihood
   for(i in 1:n){
     Y[i] ~ dnorm(alpha+inprod(X[i,],beta[]),taue)
   }
  # Priors
   for(j in 1:p){
     beta[j] ~ dnorm(0,taue*taub)
   }
   alpha ~ dnorm(0,0.001)
   taue  ~ dgamma(0.1, 0.1)
   taub  ~ dgamma(0.1, 0.1)
}")

model = jags.model(model_string,data = data, n.chains=n.chains,quiet=TRUE)
update(model, burn, progress.bar="none")
samples2 = coda.samples(model, variable.names=params, thin=thin, n.iter=n.iter, progress.bar="none")
par(mar=c(2,2,2,2))
plot(samples2)
```


Trace of beta[1]
Density of beta[1]
Trace of beta[2]
Density of beta[2]
Trace of beta[3]
Density of beta[3]
Trace of beta[4]
Density of beta[4]




```
round(effectiveSize(samples2),1)
```

```
## beta[1] beta[2] beta[3] beta[4] beta[5] beta[6] beta[7] beta[8]
##  2843.8  2061.3  2013.4  4000.0  4000.0  2529.0  4000.0  4000.0
```

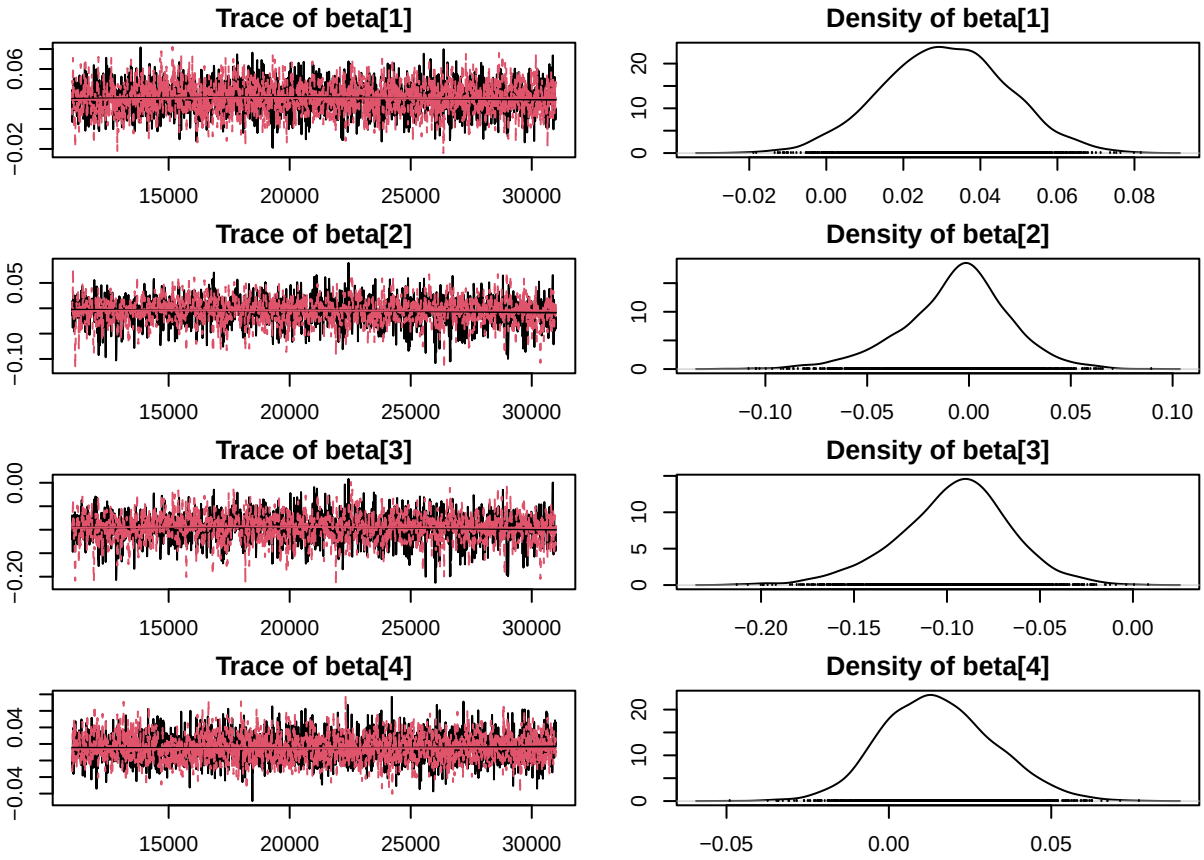
```
sum                       = summary(samples2)
rownames(sum$statistics) = names
rownames(sum$quantiles)  = names
sum$statistics           = round(sum$statistics,3)
sum$quantiles            = round(sum$quantiles,3)
sum
```

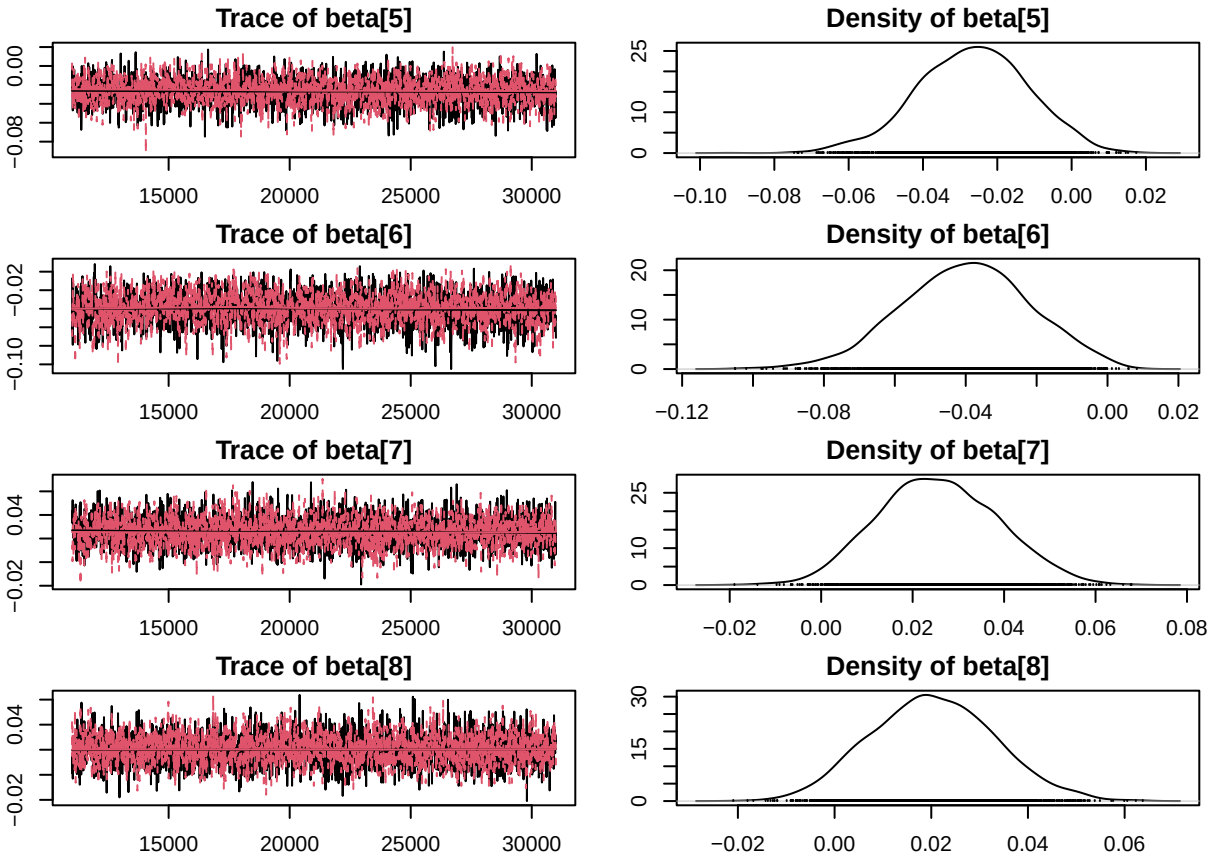
```
##
## Iterations = 10010:30000
## Thinning interval = 10
## Number of chains = 2
## Sample size per chain = 2000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                      Mean    SD Naive SE Time-series SE
## Longitude           0.027 0.017    0.000          0.000
## Latitude           -0.021 0.031    0.000          0.001
## Temperature        -0.115 0.034    0.001          0.001
```

```
## Precipitation       0.021 0.019    0.000          0.000
## NPP                -0.034 0.015    0.000          0.000
## Elevation          -0.050 0.019    0.000          0.000
## Single-family home  0.027 0.013    0.000          0.000
## Number of bedrooms  0.022 0.013    0.000          0.000
##
## 2. Quantiles for each variable:
##
##                       2.5%    25%    50%    75%  97.5%
## Longitude          -0.006  0.016  0.027  0.038  0.060
## Latitude           -0.085 -0.042 -0.020  0.000  0.036
## Temperature        -0.184 -0.138 -0.114 -0.092 -0.052
## Precipitation      -0.016  0.009  0.021  0.034  0.059
## NPP                -0.064 -0.044 -0.034 -0.023 -0.005
## Elevation          -0.087 -0.062 -0.049 -0.038 -0.014
## Single-family home  0.002  0.019  0.027  0.036  0.054
## Number of bedrooms -0.003  0.013  0.022  0.030  0.048
```

(3) Fit the Bayesian LASSO model

```
model_string = textConnection("model{
  # Likelihood
   for(i in 1:n){
     Y[i] ~ dnorm(alpha+inprod(X[i,],beta[]),taue)
   }
  # Priors
   for(j in 1:p){
     beta[j] ~ ddexp(0,taue*taub)
   }
   alpha ~ dnorm(0,0.001)
   taue  ~ dgamma(0.1, 0.1)
   taub  ~ dgamma(0.1, 0.1)
}")

model = jags.model(model_string,data = data, n.chains=n.chains,quiet=TRUE)
update(model, burn, progress.bar="none")
samples3 = coda.samples(model, variable.names=params, thin=thin, n.iter=n.iter, progress.bar="none")
par(mar=c(2,2,2,2))
plot(samples3)
```


Trace of beta[1]
Density of beta[1]
Trace of beta[2]
Density of beta[2]
Trace of beta[3]
Density of beta[3]
Trace of beta[4]
Density of beta[4]




```
round(effectiveSize(samples3),1)
```

```
## beta[1] beta[2] beta[3] beta[4] beta[5] beta[6] beta[7] beta[8]
##  2366.8  1388.3  1493.5  3537.4  3578.5  2410.2  4000.0  4000.0
```

```
sum                       = summary(samples3)
rownames(sum$statistics) = names
rownames(sum$quantiles)  = names
sum$statistics           = round(sum$statistics,3)
sum$quantiles            = round(sum$quantiles,3)
sum
```

```
##
## Iterations = 11010:31000
## Thinning interval = 10
## Number of chains = 2
## 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
## Longitude           0.031 0.016        0          0.000
## Latitude           -0.006 0.026        0          0.001
## Temperature        -0.096 0.031        0          0.001
```

```
## Precipitation       0.015 0.017        0          0.000
## NPP                -0.027 0.015        0          0.000
## Elevation          -0.040 0.018        0          0.000
## Single-family home  0.026 0.013        0          0.000
## Number of bedrooms  0.020 0.013        0          0.000
##
## 2. Quantiles for each variable:
##
##                       2.5%    25%    50%    75%  97.5%
## Longitude          -0.001  0.020  0.031  0.042  0.063
## Latitude           -0.065 -0.021 -0.004  0.010  0.043
## Temperature        -0.163 -0.114 -0.094 -0.077 -0.038
## Precipitation      -0.016  0.003  0.014  0.026  0.051
## NPP                -0.058 -0.037 -0.026 -0.016  0.001
## Elevation          -0.077 -0.052 -0.039 -0.028 -0.005
## Single-family home  0.001  0.016  0.025  0.035  0.051
## Number of bedrooms -0.004  0.011  0.020  0.029  0.046
```

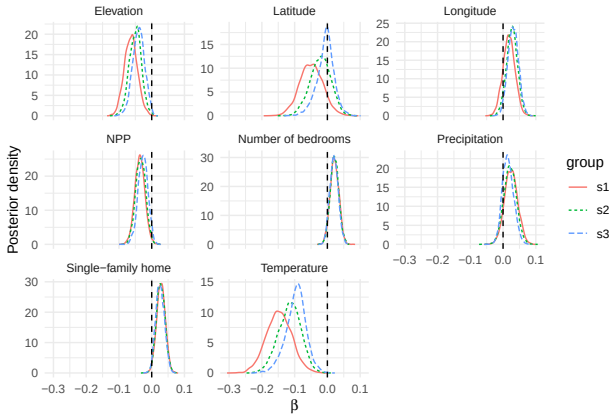
Compare the three fits

The plots below show the posterior for each covariate's slope for the three models:

1. Uninformative Gaussian is the solid line
2. Gaussian shrinkage is the dotted line
3. Bayesian LASSO is the dashed line

```r
# Create an empty list to store the data frames
df_list <- list()

for(j in 1:p){
  s1 <- c(samples1[[1]][,j], samples1[[2]][,j])
  s2 <- c(samples2[[1]][,j], samples2[[2]][,j])
  s3 <- c(samples3[[1]][,j], samples3[[2]][,j])

  # Smooth densities
  d1 <- density(s1)
  d2 <- density(s2)
  d3 <- density(s3)

  # Convert density objects to data frames
  df1 <- data.frame(x = d1$x, y = d1$y, group = "s1")
  df2 <- data.frame(x = d2$x, y = d2$y, group = "s2")
  df3 <- data.frame(x = d3$x, y = d3$y, group = "s3")

  # Combine data frames
  df_combined <- rbind(df1, df2, df3)
  df_combined$parameter <- names[j]

  # Store in list
  df_list[[j]] <- df_combined
}
```

```r
# Combine all data frames in the list into one data frame
df_final <- do.call(rbind, df_list)

ggplot(df_final, aes(x = x, y = y, color = group, linetype = group)) +
  geom_line() +
  facet_wrap(~ parameter, scales = "free_y") +
  labs(x = expression(beta), y = "Posterior density") +
  geom_vline(xintercept = 0, linetype = "dashed") +
  theme_minimal()
```



Since number of observations is so much bigger than number of features, the prior has little effect on the posterior. Based on the result,temperature, NPP, elevation, and single-family home are the most important predictors in all three models .