Breast Cancer Prediction Case Study - Bayesian Logistic Regression with Comparison of Frequentist and Bayesian Variable Selection Methods

Laura Silvana Alvarez Luque, Florencia Luque, Nicolas Bühringer, Simon Schmetz

2025-03-14

```r
library(readr);library(dplyr)
library(ggplot2);library(dplyr)
library(tidyr);library(corrplot)
library(caret);library(MCMCpack)
library(car);library(boot)
library(gridExtra); library(BAS)
```

Introduction

This case study is based on the Breast Cancer Wisconsin (Diagnostic) Data Set (https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data). The data set contains 569 observations and 32 variables. The data set is available at the UCI Machine Learning Repository. The data set contains mean (and at times min and max) values of the patient for the following numeric (continious) variables:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

The data set also contains the following Binary variables:

2) Diagnosis (M = malignant, B = benign)

Where Malignant (M) means the tumor is cancerous, while Benign (B): means that the tumor is non-cancerous.

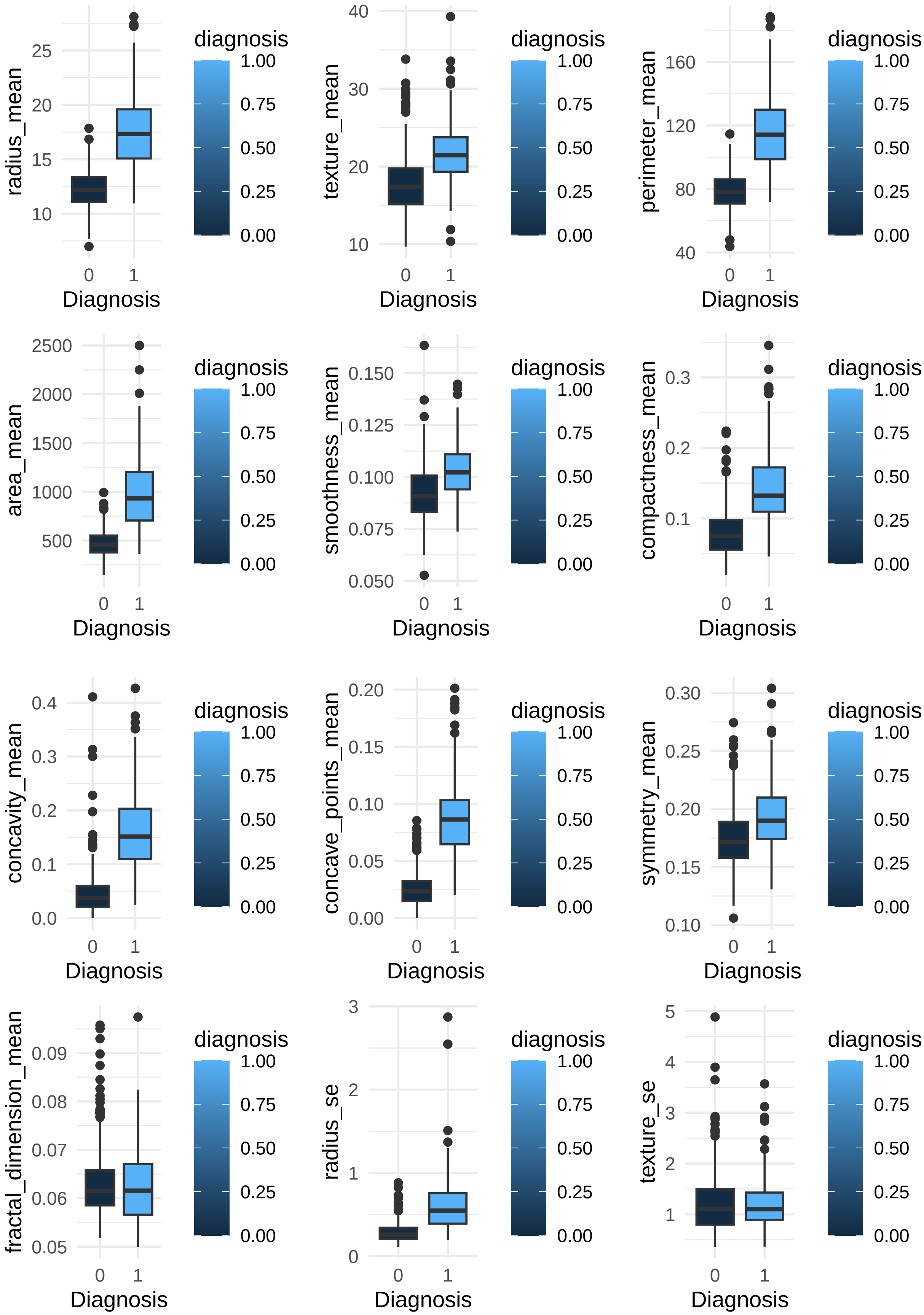
Read Data

```
data <- read.csv("data.csv", header = TRUE, sep = ",")
data <- dplyr::select(data, -c(X,id))
names(data) <- gsub("\\.", "_", names(data))
data$diagnosis <- ifelse(data$diagnosis == "M", 1, 0)
```

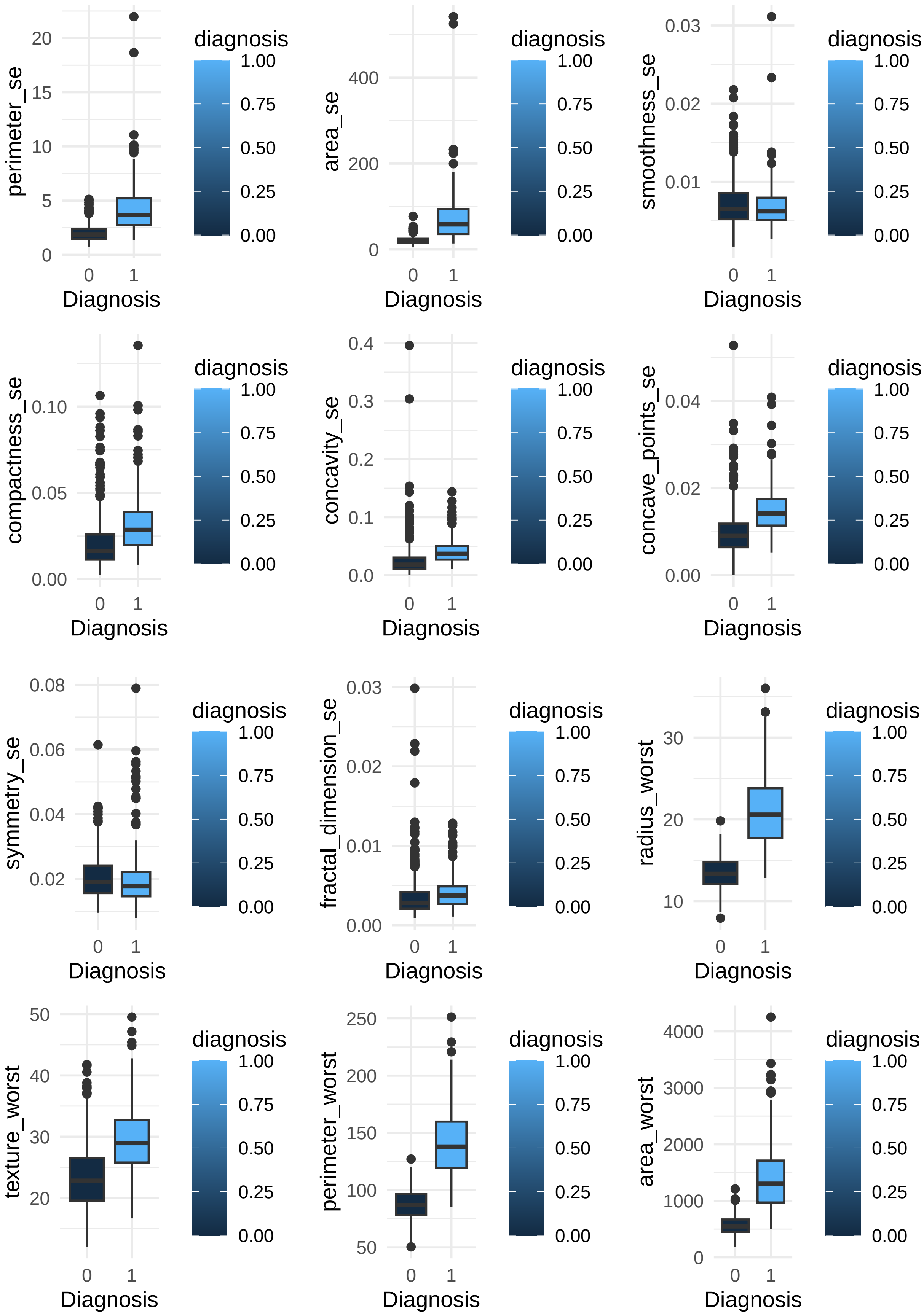
Exploratory Data Analysis

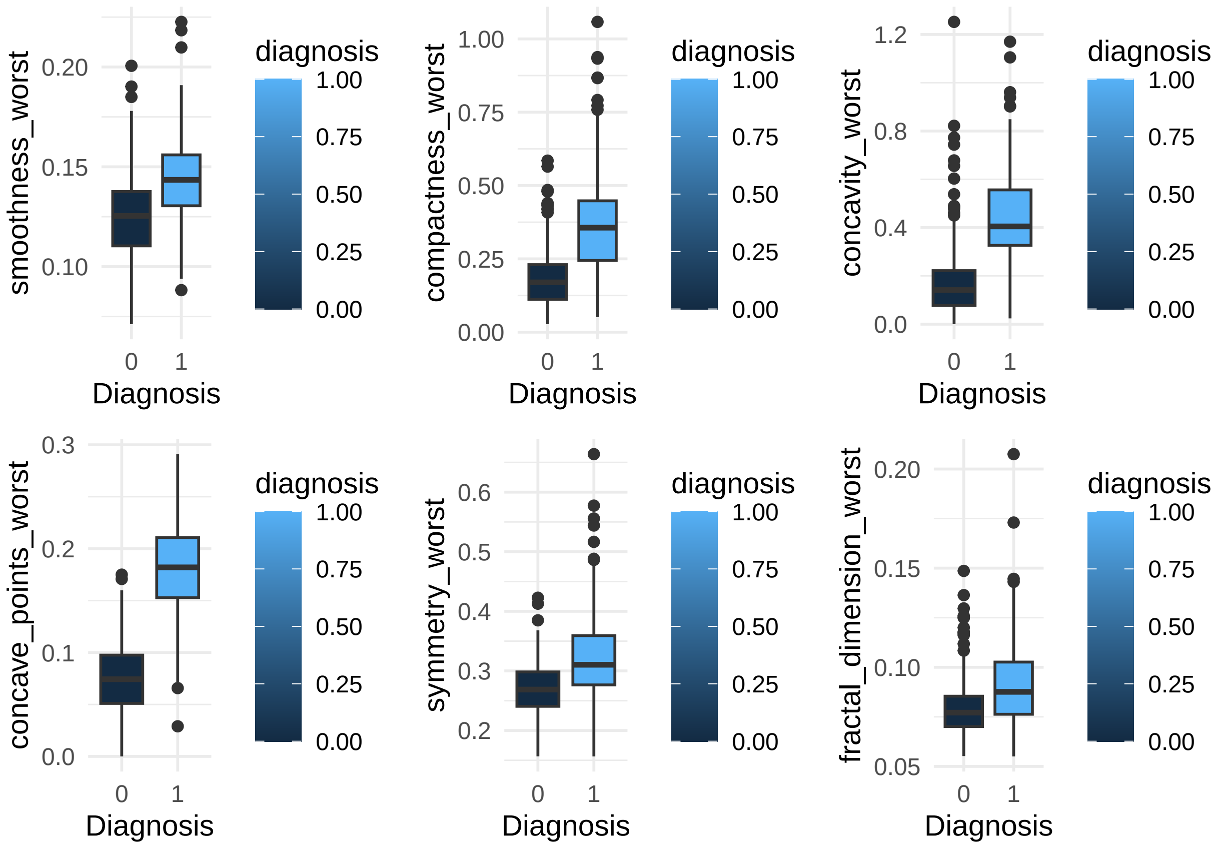
Relation with response var

```
numeric_vars <- data %>%select_if(is.numeric) %>% colnames()
numeric_vars <- setdiff(numeric_vars, "diagnosis")

plots <- lapply(numeric_vars, function(var) {
  ggplot(data, aes(x = factor(diagnosis), y = .data[[var]], fill = diagnosis)) +
    geom_boxplot() +
    labs(x = "Diagnosis", y = var) +
    theme_minimal()
})

# Print all plots

# Display plots in batches of 6 (2 rows × 3 columns)
num_plots <- length(plots)
batch_size <- 6

for(i in seq(1, num_plots, batch_size)) {
  end_idx <- min(i + batch_size - 1, num_plots)
  batch_plots <- plots[i:end_idx]
  grid.arrange(grobs = batch_plots, ncol = 3)
}
```



radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave_points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
Diagnosis
0
1
diagnosis
1.00
0.75
0.50
0.25
0.00


perimeter_se
0
5
10
15
20
area_se
0
200
400
smoothness_se
0.01
0.02
0.03
compactness_se
0.00
0.05
0.10
concavity_se
0.0
0.1
0.2
0.3
0.4
concave_points_se
0.00
0.02
0.04
symmetry_se
0.02
0.04
0.06
0.08
fractal_dimension_se
0.00
0.01
0.02
0.03
radius_worst
10
20
30
texture_worst
20
30
40
50
perimeter_worst
50
100
150
200
250
area_worst
0
1000
2000
3000
4000
0
1
Diagnosis
diagnosis
1.00
0.75
0.50
0.25
0.00



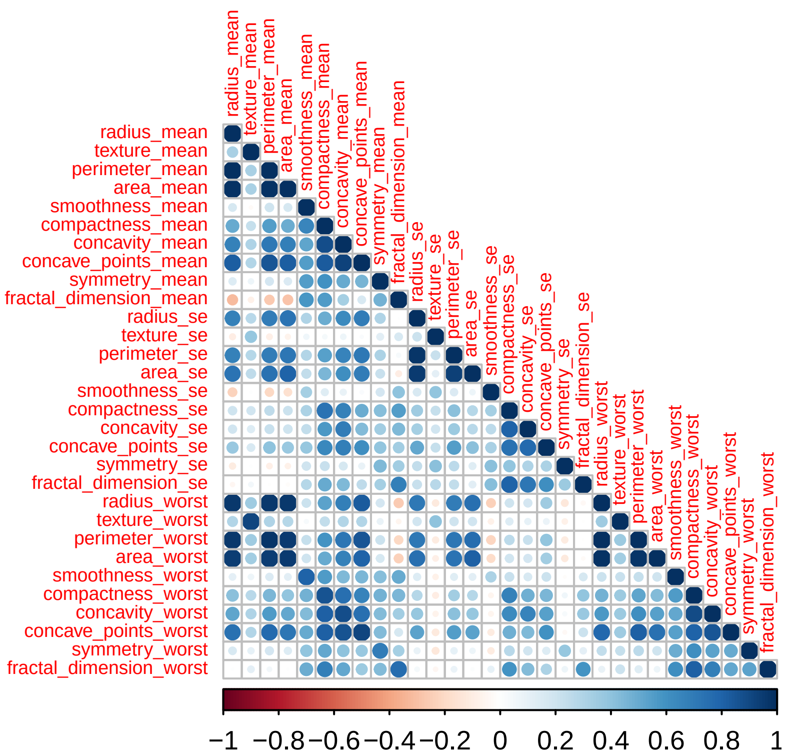

```r
summary_stat <- data %>%
  group_by(factor(diagnosis)) %>%
  summarise(across(all_of(numeric_vars),
                list(
                  mean = ~mean(.x, na.rm = TRUE),
                  sd = ~sd(.x, na.rm = TRUE),
                  median = ~median(.x, na.rm = TRUE),
                  min = ~min(.x, na.rm = TRUE),
                  max = ~max(.x, na.rm = TRUE)
                )
            ))


# For easier viewing, you can pivot longer
summary_long <- summary_stat %>%
  pivot_longer(cols = -`factor(diagnosis)`,
               names_to = c("variable", "stat"),
               names_pattern = "(.*)_(.*)")

summary_long
```

```
## # A tibble: 300 x 4
##    'factor(diagnosis)' variable    stat  value
##    <fct>               <chr>       <chr> <dbl>
##  1 0                   radius_mean mean  12.1
##  2 0                   radius_mean sd     1.78
```

```
##   3 0                     radius_mean  median 12.2
##   4 0                     radius_mean  min     6.98
##   5 0                     radius_mean  max    17.8
##   6 0                     texture_mean mean   17.9
##   7 0                     texture_mean sd      4.00
##   8 0                     texture_mean median 17.4
##   9 0                     texture_mean min     9.71
##  10 0                     texture_mean max    33.8
## # i 290 more rows
```

Correlation

```r
# Check correlation between numeric variables
cor_matrix <- cor(data[, numeric_vars])
corrplot(cor_matrix, method = "circle",type="lower",tl.cex = 0.6)
```




```r
# Or find highly correlated variables
high_cor <- findCorrelation(cor_matrix, cutoff = 0.8)
problematic_vars <- numeric_vars[high_cor]
print(problematic_vars)
```

```
##  [1] "concavity_mean"       "concave_points_mean"  "compactness_mean"
##  [4] "concave_points_worst" "concavity_worst"      "perimeter_worst"
##  [7] "radius_worst"         "perimeter_mean"       "compactness_worst"
## [10] "area_worst"           "radius_mean"          "perimeter_se"
## [13] "compactness_se"       "area_se"              "smoothness_mean"
## [16] "texture_mean"
```

Variable Selection

Frequentist Approach

Check VIF and remove variables with extremely high values

```
predictors <- setdiff(names(data), c("diagnosis"))
formula_str <- paste("diagnosis ~", paste(predictors, collapse = " + "))
formula <- as.formula(formula_str)

l_reg = lm(formula, data)
vif_values <- vif(l_reg)

vif_df <- data.frame(
  Variable = names(vif_values),
  VIF = vif_values
)
vif_df <- vif_df %>% arrange(desc(VIF))
print(head(vif_df,5))
```

```
##                        Variable       VIF
## radius_mean         radius_mean 3806.1153
## perimeter_mean   perimeter_mean 3786.4004
## radius_worst       radius_worst  799.1059
## perimeter_worst perimeter_worst  405.0233
## area_mean             area_mean  347.8787
```

```
vars_to_exclude <- c(head(vif_df,15)$Variable)
```

Check correlations after excluding x VIF, variables to pay attention if something does not work.

```
# Check correlation between numeric variables
cor_matrix_f <- cor(data[, setdiff(numeric_vars, vars_to_exclude)])
# corrplot(cor_matrix, method = "circle")

# Or find highly correlated variables
high_cor_f <- findCorrelation(cor_matrix_f, cutoff = 0.8)
problematic_vars_f <- setdiff(numeric_vars, vars_to_exclude)[high_cor_f]
print(problematic_vars_f)
```

```
## [1] "compactness_se"  "smoothness_mean" "texture_mean"
```

```
selected_freq <- setdiff(numeric_vars, vars_to_exclude)
```

Bayesian Approach

We obtain the posterior probability of including each beta, and also, some statistics for different models in order to select which one is the better. Then, we select the variables that compound the model with highest BF and lower logmarg.

```r
# Fit a Bayesian logistic regression with variable selection
model_bas <- bas.glm(diagnosis ~ .,
                     data = data,
                     family = binomial(),
                     method = "MCMC", # or "BAS" for deterministic sampling
                     MCMC.iterations = 10000,
                     modelprior = uniform()) # Prior over model space

# Summary of results
summary(model_bas)[30:36,]
```

```
##                          P(B != 0 | Y)     model 1     model 2   model 3
## symmetry_worst                  0.6746   1.0000000   1.0000000   1.00000
## fractal_dimension_worst         0.2648   0.0000000   0.0000000   0.00000
## BF                                  NA   0.3566076   0.3802395   1.00000
## PostProbs                           NA   0.0028000   0.0026000   0.00260
## R2                                  NA   0.9281000   0.9228000   0.91970
## dim                                 NA  11.0000000  10.0000000   9.00000
## logmarg                             NA -53.4100663 -53.3459012 -52.37895
##                             model 4     model 5
## symmetry_worst             0.000000   1.0000000
## fractal_dimension_worst    0.000000   0.0000000
## BF                         0.014816   0.0163772
## PostProbs                  0.002500   0.0023000
## R2                         0.914400   0.9307000
## dim                       10.000000  13.0000000
## logmarg                  -56.590995 -56.4908124
```

```r
# Posterior inclusion probabilities
pip <- model_bas$probne0
variable_names <- names(pip)
#pip_df <- data.frame(Variable = numeric_vars,
#                      InclusionProb = pip)
#pip_df <- pip_df[order(pip_df$InclusionProb, decreasing = TRUE),]
#print(pip_df)

selected_bayes <- c( "perimeter_mean", "concave_points_mean", "compactness_mean",
                     "concavity_mean", "area_se", "smoothness_se", "concave_points_se",
                     "fractal_dimension_se","radius_worst", "texture_worst",
                     "fractal_dimension_worst")
```

Logistic Models

Then we fit the models for the variable selected in each case, evaluate the autocorrelation and fix the thinning parameter and starting point. This second one is selected as the beta estimation of a regular linear model.

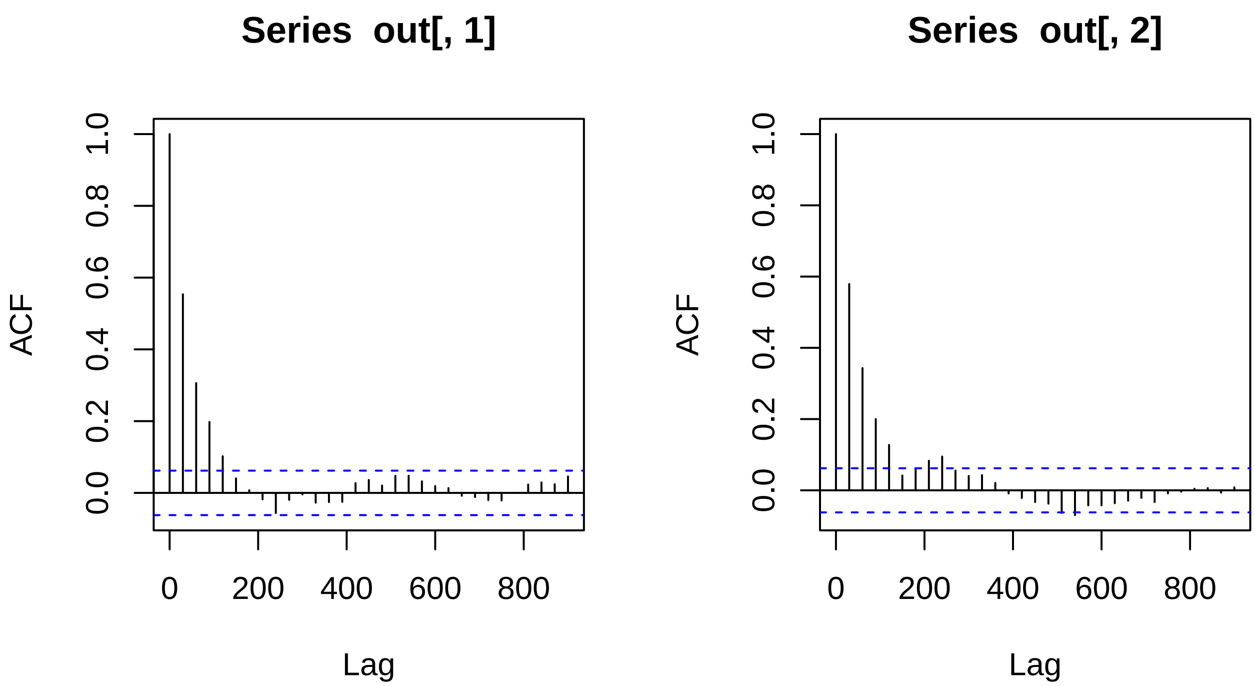
Freq var selection

```r
formula_str <- paste("diagnosis ~", paste(selected_freq, collapse = " + "))
formula <- as.formula(formula_str)

freq_model1<-lm(formula, data = data)
beta.start1 <- coef(freq_model1)

out = MCMClogit(formula, data, burnin=1000, mcmc=21000, beta.start = beta.start1)
# summary(out)
# acf(out[,1])
# acf(out[,2])

# Correct autocorrelation

out = MCMClogit(formula, data, burnin=5000, mcmc=30000, thin = 30,
                beta.start = beta.start1)
summary(out)
```

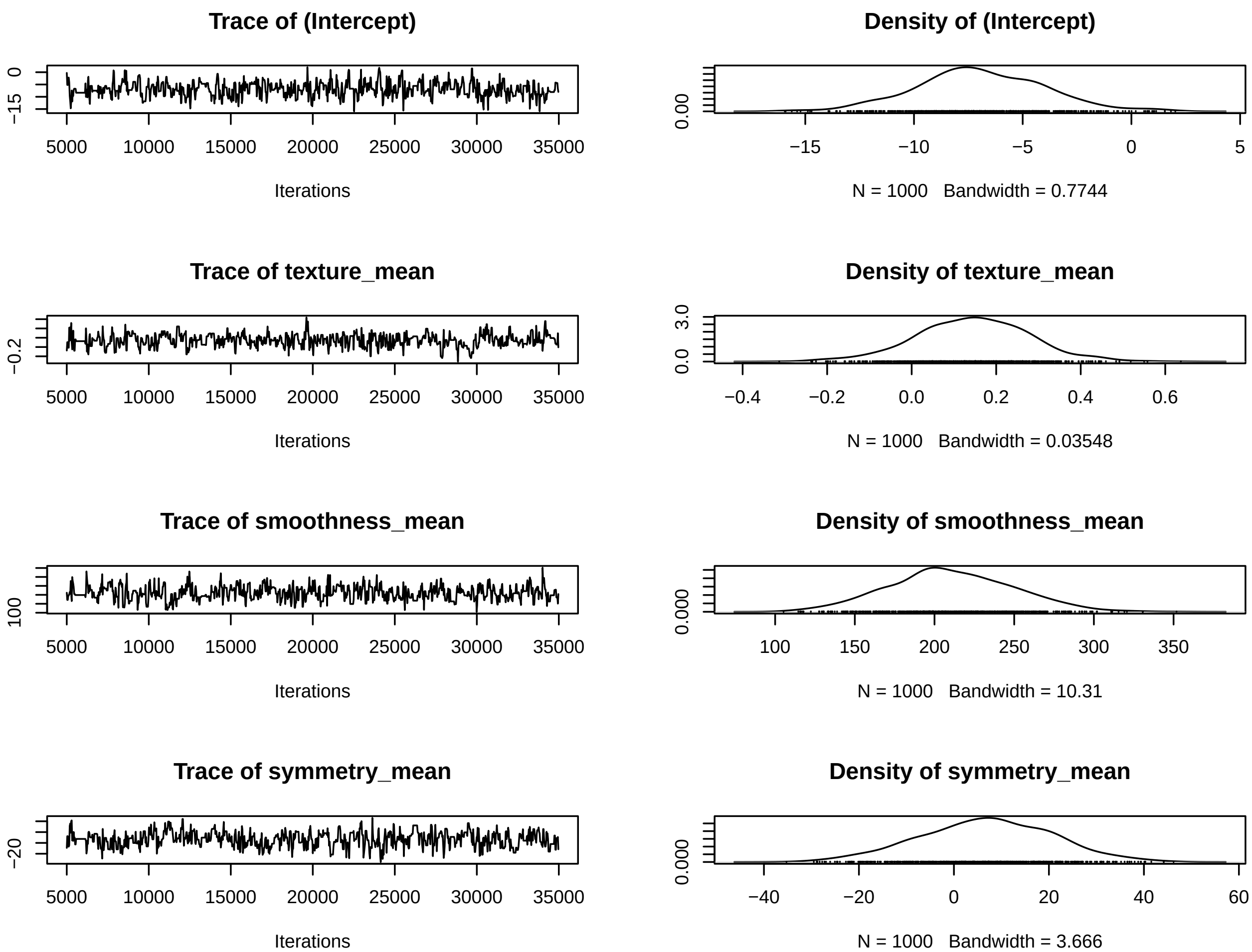
```
##
## Iterations = 5001:34971
## Thinning interval = 30
## Number of chains = 1
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                               Mean       SD Naive SE Time-series SE
## (Intercept)                -6.9619   2.9564 0.093488       0.174505
## texture_mean                0.1449   0.1338 0.004230       0.008603
## smoothness_mean           210.3856  38.8092 1.227254       2.364636
## symmetry_mean               6.7972  13.7695 0.435430       0.874829
## fractal_dimension_mean   -692.9451  85.9090 2.716683       5.109411
## texture_se                 -1.4312   0.7698 0.024345       0.043598
## smoothness_se            -245.3894 120.4758 3.809779       6.431210
## compactness_se             20.8806  28.9179 0.914463       1.881365
## concavity_se               -2.1651  11.8429 0.374504       0.841066
## concave_points_se         419.8216  68.7222 2.173187       4.788362
## symmetry_se                 2.8399  51.6192 1.632341       3.064199
## fractal_dimension_se     -235.3213 230.1363 7.277549      12.459032
## texture_worst               0.2585   0.1235 0.003904       0.008201
## smoothness_worst            0.4922  26.7191 0.844933       1.548509
## symmetry_worst             12.2236   8.5569 0.270594       0.595713
## fractal_dimension_worst   153.6350  38.6341 1.221717       2.154905
##
## 2. Quantiles for each variable:
##
##                               2.5%        25%       50%       75%      97.5%
## (Intercept)              -1.257e+01   -8.82359   -7.1256   -4.9263   -1.06303
## texture_mean             -1.264e-01    0.05609    0.1493    0.2347    0.42288
## smoothness_mean           1.346e+02  185.56985  207.4429  237.4691  285.25906
## symmetry_mean            -2.133e+01   -2.07081    7.2447   16.8415   33.46638
## fractal_dimension_mean   -8.685e+02 -748.06764 -687.5111 -638.0705 -535.08845
## texture_se               -3.011e+00   -1.90562   -1.3896   -0.9116   -0.01158
```

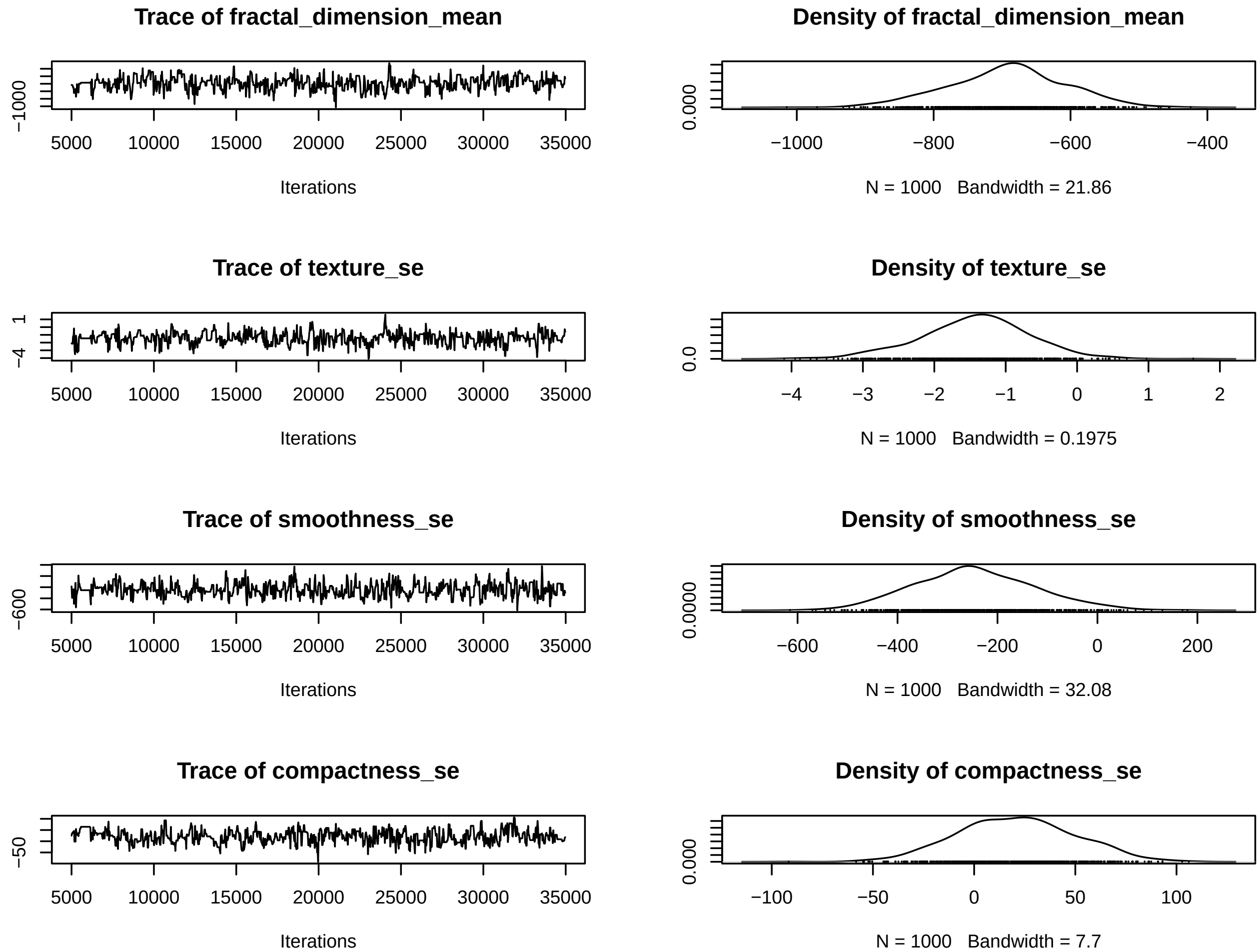
```
## smoothness_se          -4.695e+02 -329.76049 -253.4453 -166.7919    2.89369
## compactness_se         -3.356e+01    1.54953   21.3018   40.3877   76.28100
## concavity_se           -2.669e+01  -10.76965   -2.0245    6.0901   20.70983
## concave_points_se       2.861e+02  374.51992  416.7823  463.6432  566.28489
## symmetry_se            -1.014e+02  -29.30766    2.0131   37.1225   99.63610
## fractal_dimension_se   -7.050e+02 -385.97951 -235.5333  -66.6496  194.69516
## texture_worst           2.987e-03    0.17517    0.2570    0.3330    0.52035
## smoothness_worst       -5.199e+01  -17.05354   -0.2086   17.8640   54.16787
## symmetry_worst         -2.732e+00    6.80055   11.5672   17.4516   30.08029
## fractal_dimension_worst  7.957e+01  126.52236  155.9399  178.0914  231.92666
```

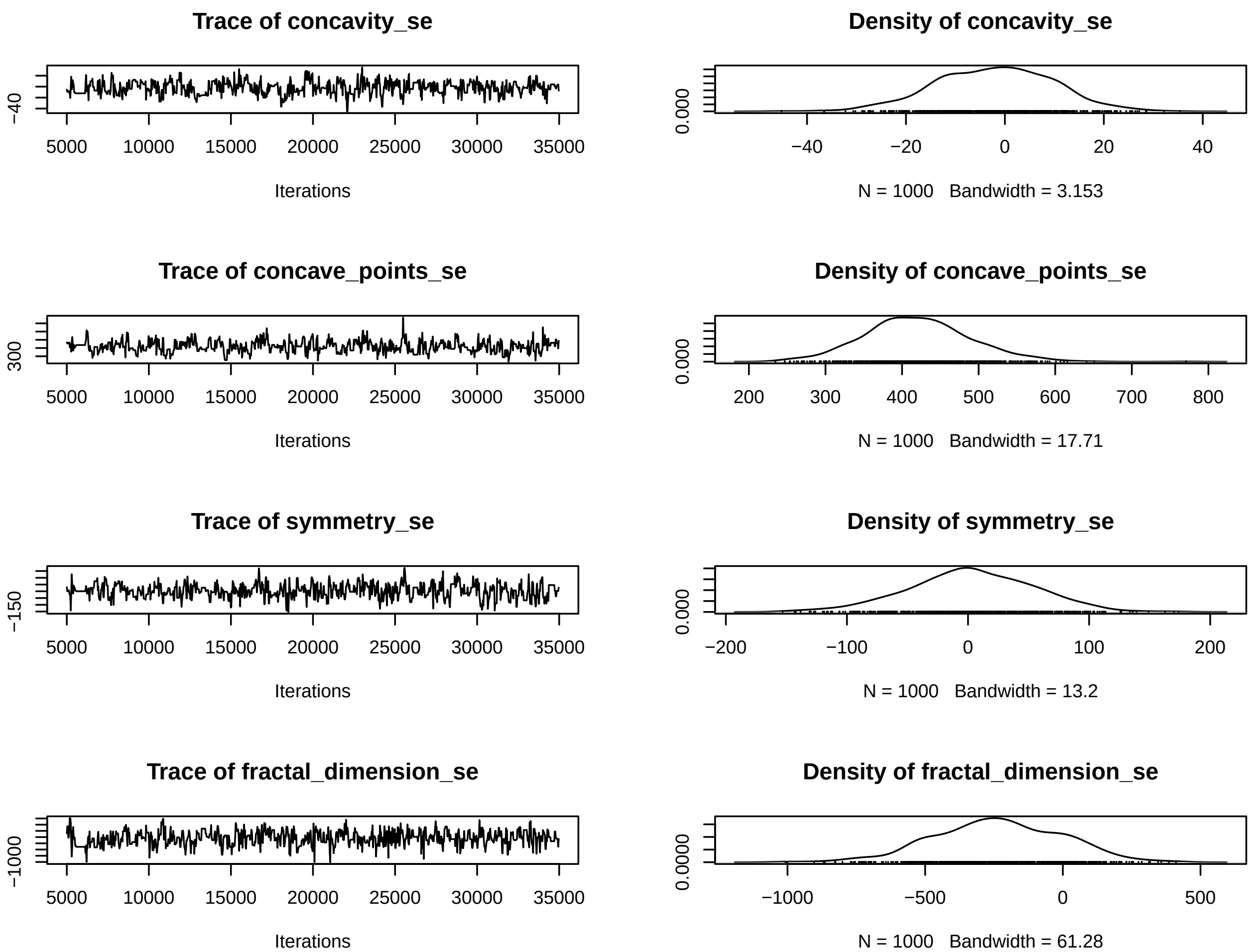
```r
par(mfrow=c(1,2))
acf(out[,1])
acf(out[,2])
```

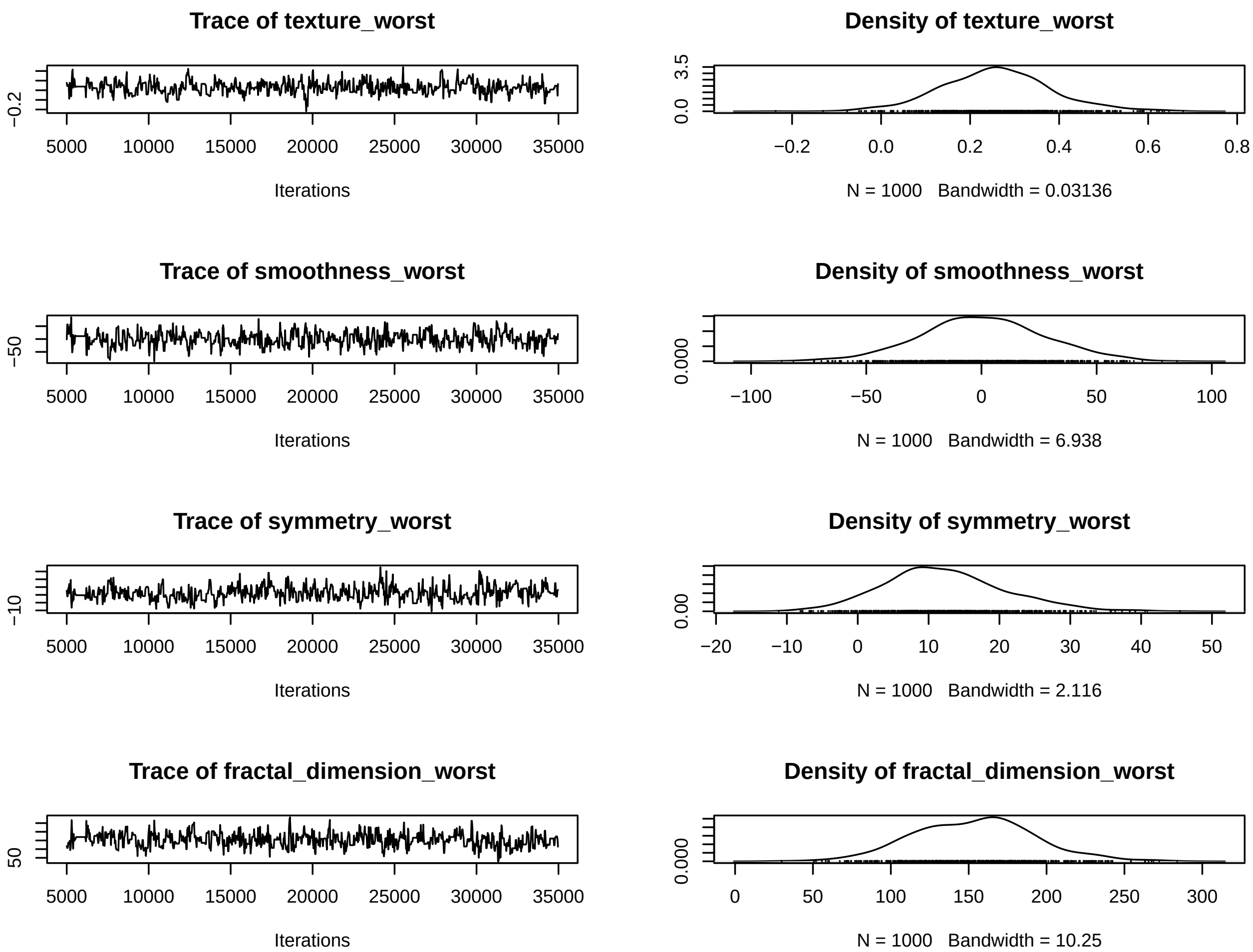



```r
plot(out)
```


Trace of (Intercept)
0
−15
5000
10000
15000
20000
25000
30000
35000
Iterations
Density of (Intercept)
0.00
−15
−10
−5
0
5
N = 1000 Bandwidth = 0.7744
Trace of texture_mean
−0.2
5000
10000
15000
20000
25000
30000
35000
Iterations
Density of texture_mean
3.0
0.0
−0.4
−0.2
0.0
0.2
0.4
0.6
N = 1000 Bandwidth = 0.03548
Trace of smoothness_mean
100
5000
10000
15000
20000
25000
30000
35000
Iterations
Density of smoothness_mean
0.000
100
150
200
250
300
350
N = 1000 Bandwidth = 10.31
Trace of symmetry_mean
−20
5000
10000
15000
20000
25000
30000
35000
Iterations
Density of symmetry_mean
0.000
−40
−20
0
20
40
60
N = 1000 Bandwidth = 3.666


Trace of fractal_dimension_mean
−1000
5000 10000 15000 20000 25000 30000 35000
Iterations
Density of fractal_dimension_mean
0.000
−1000 −800 −600 −400
N = 1000 Bandwidth = 21.86
Trace of texture_se
1
−4
5000 10000 15000 20000 25000 30000 35000
Iterations
Density of texture_se
0.0
−4 −3 −2 −1 0 1 2
N = 1000 Bandwidth = 0.1975
Trace of smoothness_se
−600
5000 10000 15000 20000 25000 30000 35000
Iterations
Density of smoothness_se
0.0000
−600 −400 −200 0 200
N = 1000 Bandwidth = 32.08
Trace of compactness_se
−50
5000 10000 15000 20000 25000 30000 35000
Iterations
Density of compactness_se
0.000
−100 −50 0 50 100
N = 1000 Bandwidth = 7.7


Trace of concavity_se
−40
5000 10000 15000 20000 25000 30000 35000
Iterations
Density of concavity_se
0.000
−40 −20 0 20 40
N = 1000 Bandwidth = 3.153
Trace of concave_points_se
300
5000 10000 15000 20000 25000 30000 35000
Iterations
Density of concave_points_se
0.000
200 300 400 500 600 700 800
N = 1000 Bandwidth = 17.71
Trace of symmetry_se
−150
5000 10000 15000 20000 25000 30000 35000
Iterations
Density of symmetry_se
0.000
−200 −100 0 100 200
N = 1000 Bandwidth = 13.2
Trace of fractal_dimension_se
−1000
5000 10000 15000 20000 25000 30000 35000
Iterations
Density of fractal_dimension_se
0.0000
−1000 −500 0 500
N = 1000 Bandwidth = 61.28



Bayes var selection

```
formula_str_b <- paste("diagnosis ~", paste(selected_bayes, collapse = " + "))
formula_b <- as.formula(formula_str_b)

# starting point
freq_model<-lm(formula_b, data = data)
beta.start <- coef(freq_model)

out_b = MCMClogit(formula_b, data, burnin=1000, mcmc=21000)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

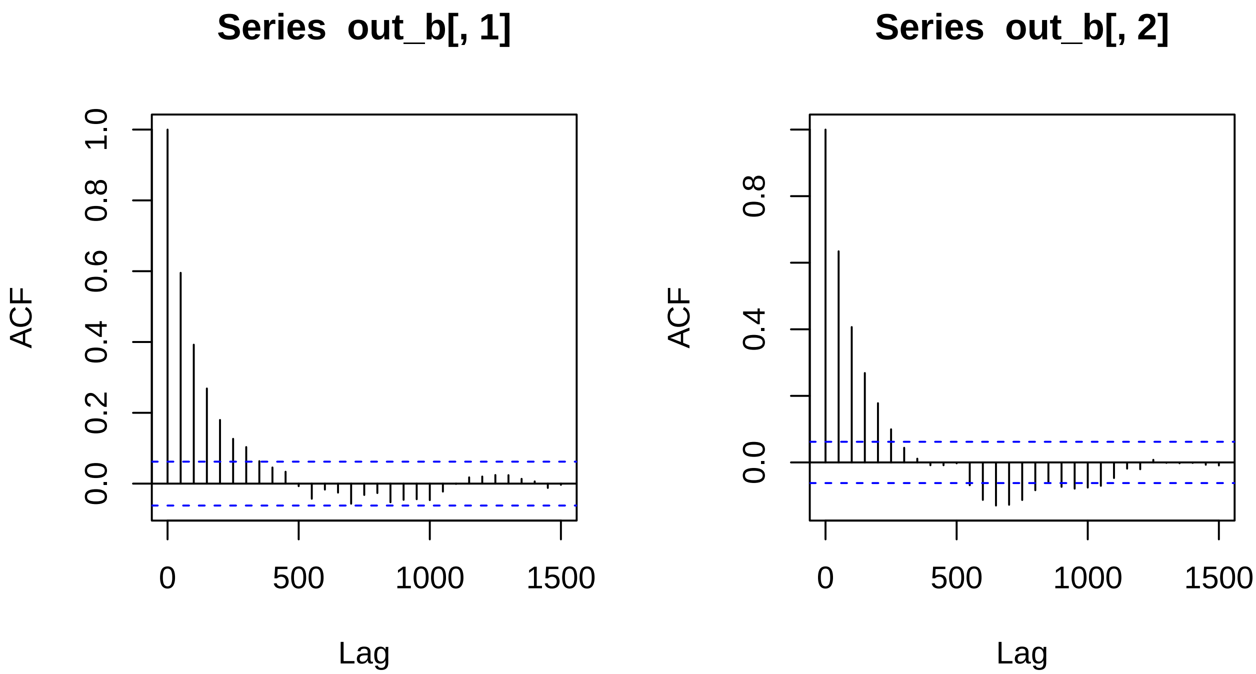
```
# summary(out_b)
# acf(out_b[,1])
# acf(out_b[,2])

out_b = MCMClogit(formula_b, data, burnin=5000, mcmc=50000,
                  beta.start = beta.start, thin = 50, tune=0.5)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

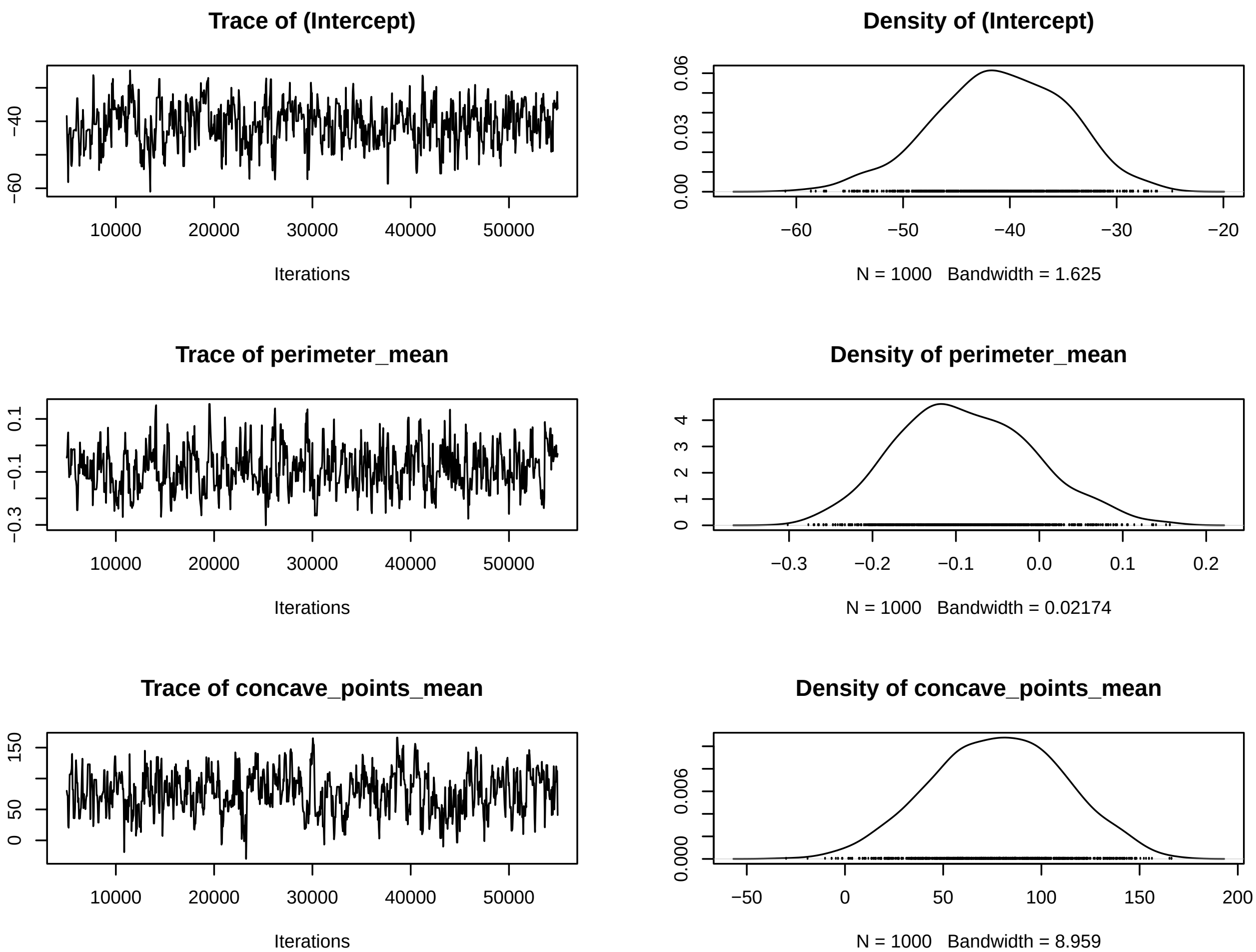
```r
summary(out_b)
```

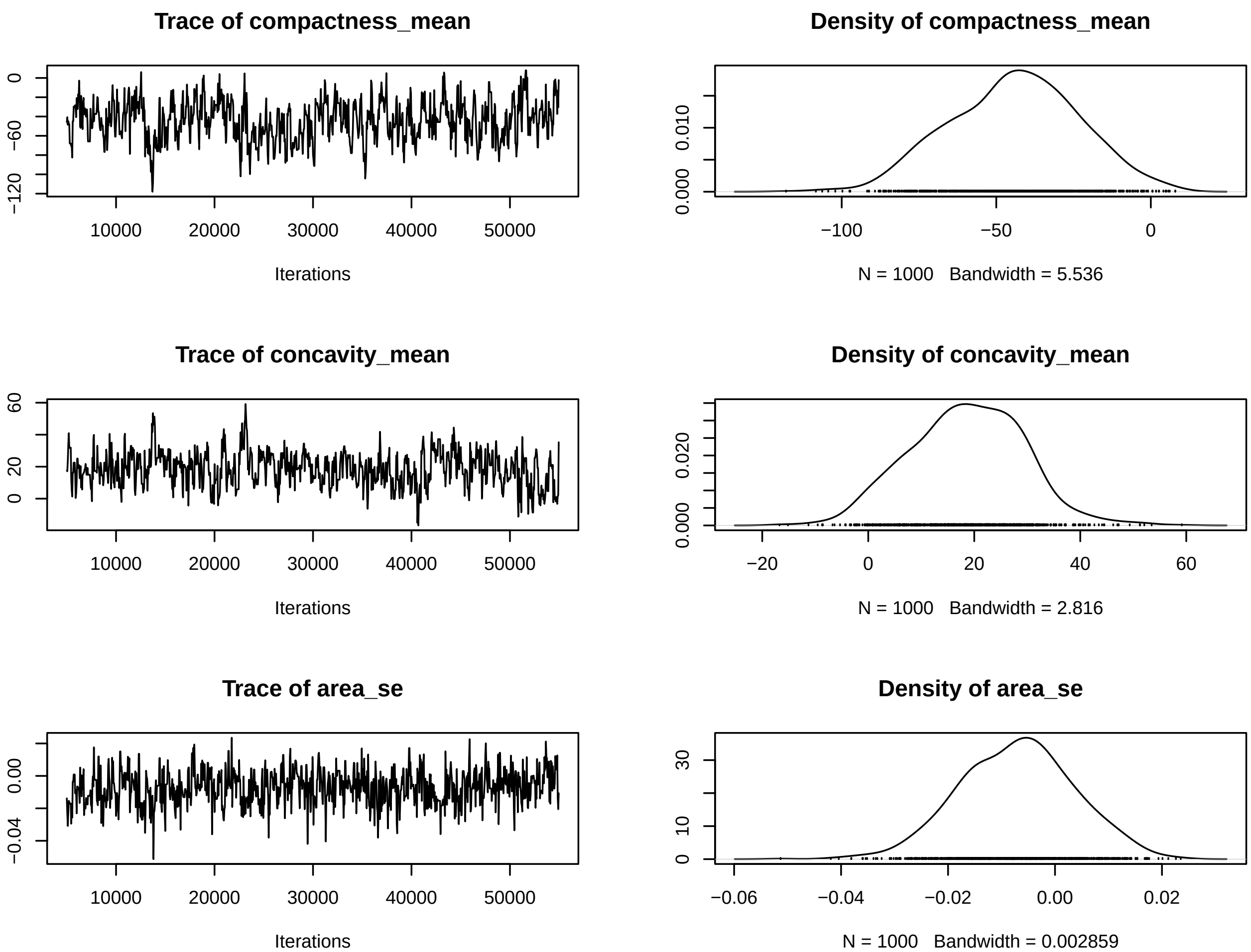
```
##
## Iterations = 5001:54951
## Thinning interval = 50
## Number of chains = 1
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                              Mean        SD  Naive SE Time-series SE
## (Intercept)            -4.089e+01   6.10325 1.930e-01      4.069e-01
## perimeter_mean         -8.767e-02   0.08165 2.582e-03      5.462e-03
## concave_points_mean     7.789e+01  33.64842 1.064e+00      2.578e+00
## compactness_mean       -4.411e+01  20.79019 6.574e-01      1.713e+00
## concavity_mean          1.849e+01  10.76886 3.405e-01      8.339e-01
## area_se                -7.147e-03   0.01074 3.396e-04      5.185e-04
## smoothness_se           2.135e+02 150.05698 4.745e+00      1.353e+01
## concave_points_se       2.921e+02 101.48434 3.209e+00      5.821e+00
## fractal_dimension_se   -1.196e+03 340.24976 1.076e+01      1.952e+01
## radius_worst            1.662e+00   0.44882 1.419e-02      2.645e-02
## texture_worst           2.749e-01   0.05205 1.646e-03      3.775e-03
## fractal_dimension_worst 1.559e+02  46.60596 1.474e+00      3.093e+00
##
## 2. Quantiles for each variable:
##
##                              2.5%        25%        50%        75%      97.5%
## (Intercept)            -5.348e+01 -4.504e+01 -4.084e+01 -3.637e+01  -29.41492
## perimeter_mean         -2.363e-01 -1.449e-01 -9.337e-02 -3.226e-02    0.08183
## concave_points_mean     1.352e+01  5.524e+01  7.861e+01  1.017e+02  141.59098
## compactness_mean       -8.455e+01 -5.933e+01 -4.349e+01 -2.957e+01   -4.49315
## concavity_mean         -2.063e+00  1.172e+01  1.872e+01  2.589e+01   39.71891
## area_se                -2.774e-02 -1.452e-02 -6.903e-03 -1.231e-04    0.01324
## smoothness_se          -8.381e+01  1.137e+02  2.063e+02  3.125e+02  512.29411
## concave_points_se       8.751e+01  2.252e+02  2.941e+02  3.573e+02  499.56987
## fractal_dimension_se   -1.905e+03 -1.411e+03 -1.177e+03 -9.488e+02 -589.19519
## radius_worst            7.393e-01  1.375e+00  1.711e+00  1.994e+00    2.41334
## texture_worst           1.752e-01  2.396e-01  2.744e-01  3.086e-01    0.37594
## fractal_dimension_worst 7.238e+01  1.237e+02  1.556e+02  1.866e+02  255.52220
```

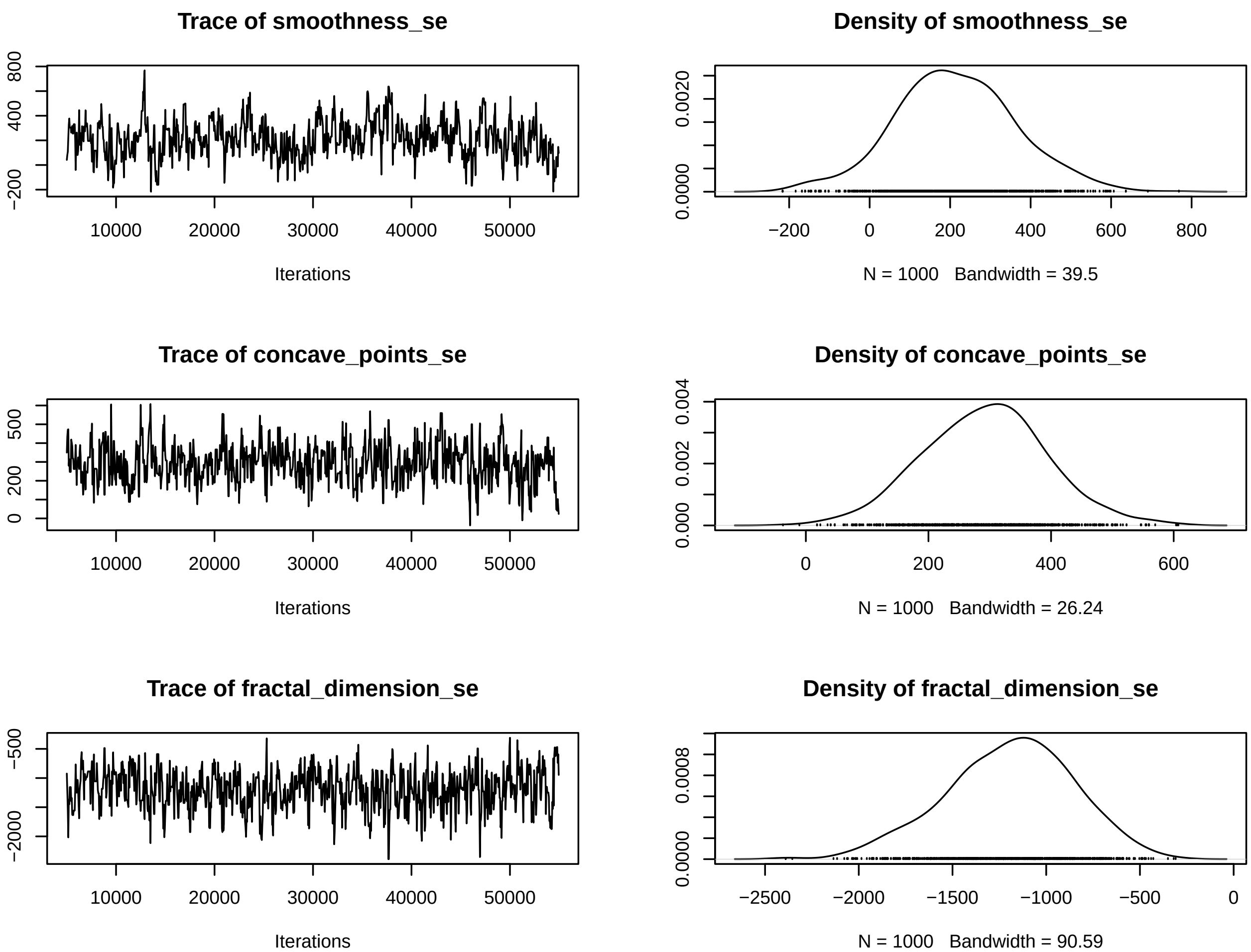
```r
par(mfrow=c(1,2))
acf(out_b[,1])
acf(out_b[,2])
```

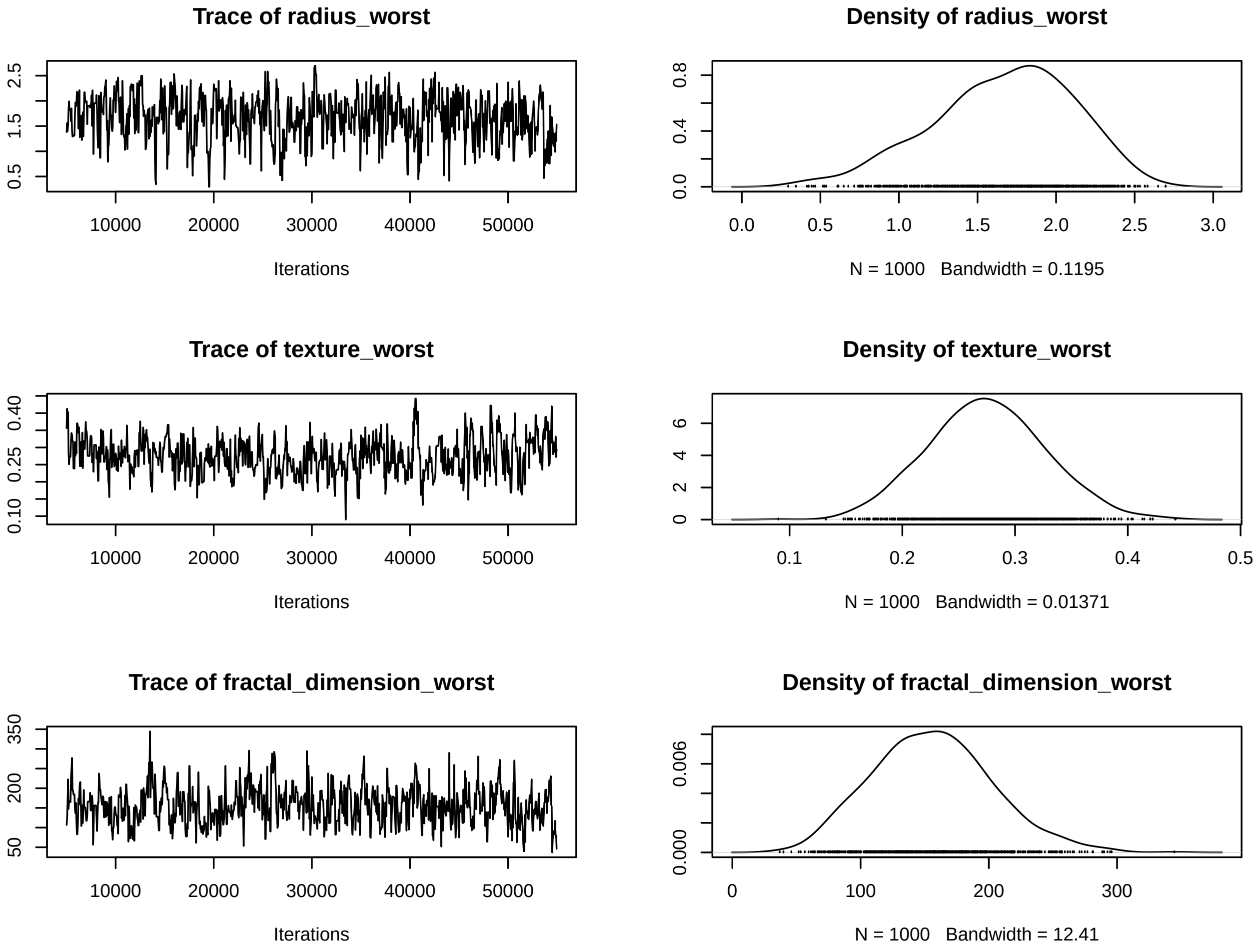

Series out_b[, 1]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
Lag
Series out_b[, 2]
ACF
0.0
0.4
0.8
0
500
1000
1500
Lag

```r
plot(out_b)
```


Trace of (Intercept)
−60
−40
10000
20000
30000
40000
50000
Iterations
Density of (Intercept)
0.00
0.03
0.06
−60
−50
−40
−30
−20
N = 1000 Bandwidth = 1.625
Trace of perimeter_mean
−0.3
−0.1
0.1
10000
20000
30000
40000
50000
Iterations
Density of perimeter_mean
0
1
2
3
4
−0.3
−0.2
−0.1
0.0
0.1
0.2
N = 1000 Bandwidth = 0.02174
Trace of concave_points_mean
0
50
150
10000
20000
30000
40000
50000
Iterations
Density of concave_points_mean
0.000
0.006
−50
0
50
100
150
200
N = 1000 Bandwidth = 8.959


Trace of compactness_mean
Iterations
Density of compactness_mean
N = 1000 Bandwidth = 5.536
Trace of concavity_mean
Iterations
Density of concavity_mean
N = 1000 Bandwidth = 2.816
Trace of area_se
Iterations
Density of area_se
N = 1000 Bandwidth = 0.002859


Trace of smoothness_se
Iterations
Density of smoothness_se
N = 1000 Bandwidth = 39.5
Trace of concave_points_se
Iterations
Density of concave_points_se
N = 1000 Bandwidth = 26.24
Trace of fractal_dimension_se
Iterations
Density of fractal_dimension_se
N = 1000 Bandwidth = 90.59



Evaluate Models with Deviance Information Criterion (DIC)

In the following code, we will calculate the Deviance Information Criterion (DIC) for both the Frequentist and Bayesian models. The DIC is a measure of model fit that penalizes the complexity of the model. Lower values of DIC indicate better model fit. The DIC is calculated as follows:

$$DIC = \bar{D} + p_D$$

where:
- $\bar{D}$ is the posterior mean deviance:

$$\bar{D} = \mathbb{E}[D(\theta) \mid \mathcal{D}]$$

with $D(\theta) = -2 \log p(\mathcal{D} \mid \theta)$, the deviance evaluated at parameter θ. - p_D a penalization term (effective number of parameters to penalize model complexity):

$$p_D = \bar{D} - D(\hat{\theta})$$

where $\hat{\theta}$ is the posterior mean of θ.

The R implementation of the DIC function is as follows and was developed with help of Prof Michael Wiper:

```r
# DIC Code
DIC = function(model, X, data, target) {
  dev = 0
  # Calculate Average Deviance of MCMC
  for (i in 1:nrow(model)) {
    params <- model[i,]
    p = inv.logit(X %*% params)
    p[data[target] == 0] = 1-p[data[target] == 0]
    dev = dev - 2 * sum(log(p))  # Negative log-likelihood
  }
  D_bar = dev / nrow(model)

  # D_theta: Deviance at the posterior mean (using the average parameter values)
  posterior_means <- colMeans(model)
  linear_predictor <- X %*% posterior_means
  p_post <- inv.logit(linear_predictor)
  p_post[data[target] == 0] = 1-p_post[data[target] == 0]

  D_theta = -2 * sum(log(p_post)) # Deviance at the posterior mean

  # p_D: Posterior deviance penalty
  p_D = D_bar - D_theta

  # DIC
  DIC = D_bar + p_D

  return(list(DIC=DIC, D_bar=D_bar, p_D=p_D))
}
```

We now continue with applying the DIC Score to the model derived from frequentist variable selection and the model derived from Bayesian variable selection. The straight forward conclusion is that the DIC is significantly better (lower) for the model that was set up with the Bayesian Variable Selection approach. Based on this result, we conclude this to be the best model and will use it for further analysis.

```r
# Frequentist
model = out
X <- model.matrix(~ texture_mean + smoothness_mean + symmetry_mean +
    fractal_dimension_mean + texture_se + smoothness_se + compactness_se +
    concavity_se + concave_points_se + symmetry_se + fractal_dimension_se +
    texture_worst + smoothness_worst + symmetry_worst + fractal_dimension_worst, data = data) # model m
target = "diagnosis"

print("Frequentist Variable Selection DIC Score")
```

```
## [1] "Frequentist Variable Selection DIC Score"
```

```r
DIC(model, X, data, target)
```

```
## $DIC
## [1] 245.5246
##
## $D_bar
```

```
## [1] 229.9737
##
## $p_D
## [1] 15.5509
```

```r
# Bayesian
model = out_b
X <- model.matrix(~ perimeter_mean + concave_points_mean + compactness_mean +
    concavity_mean + area_se + smoothness_se + concave_points_se +
    fractal_dimension_se + radius_worst + texture_worst + fractal_dimension_worst, data = data) # model
target = "diagnosis"

print("Bayesian Variable Selection DIC Score")
```

```
## [1] "Bayesian Variable Selection DIC Score"
```

```r
DIC(model, X, data, target)
```

```
## $DIC
## [1] 89.86894
##
## $D_bar
## [1] 80.33019
##
## $p_D
## [1] 9.538756
```

Prediction

Since we already have our posterior coefficients from the MCMC samples, predicting is fairly straightforward. We will just have to turn the log odds back into probability space and choose a suitable threshold probability for the two classes (1 = malignant (cancer), 0 = benign)).

The standard threshold is 0.5. However, in the light of classifying cancer, one might choose this threshold more carefully. Assuming we are performing an initial cancer screening, we would prefer having a false positive than a false negative. In simpler words, we would rather initially classify something as cancer that later turns out as no cancer than missing a cancer diagnosis that actually is one. By lowering the threshold, we reduce our exposure to false negatives and increase sensitivity.

```r
# posterior mean coefficients from MCMC samples
posterior_means <- colMeans(out_b)

# linear preds is in log odds space
linear_preds <- X %*% posterior_means

# inverse logit to get probabilities
prob_preds <- 1 / (1 + exp(-linear_preds))

# turn proba into binary prediction
test_preds <- ifelse(prob_preds >= 0.4, 1, 0)
```

Now that we have our predictions, let's analyze the outcome

```r
conf_matrix <- confusionMatrix(as.factor(test_preds),
                               as.factor(data$diagnosis),
                               positive = "1")
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 351   5
##          1   6 207
##
##                Accuracy : 0.9807
##                  95% CI : (0.9657, 0.9903)
##     No Information Rate : 0.6274
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9587
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9764
##             Specificity : 0.9832
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9860
##              Prevalence : 0.3726
##          Detection Rate : 0.3638
##    Detection Prevalence : 0.3743
##       Balanced Accuracy : 0.9798
##
##        'Positive' Class : 1
##
```

Overall, the model performs very well accurately classifying 98% of both malignant and benign cases. Our goal of achieving a high sensitivity was reached by correctly detecting 207 cancer cases out of 212 overall (97.6%). The specificity is also high with 98.3% of all benign cases correctly classified. Although false negatives might cause unnecessary medical procedures, we still prefer it this way. The model seems highly reliable as shown by the positive prediction value. When the model predicts cancer, it is correct 97.2% of the time. Overall, we have a more than solid classifier at hand.

Conclusion

This report applied bayesian logistic regression to predict breast cancer by using a breast mass cell dataset made available by the University of Irvine. The exploratory data analysis revealed significant correlations among predictors, which required variable selection. A comparison between frequentist and bayesian variable selection methods was conducted which proved the bayesian approach as far superior when assessed on the Deviance Information Criterion. The final model was evaluated via a confusion matrix which yielded a highly reliable classifier for cancer detection