# Project BDA

Anonymous

11/14/2020

## Contents

# Loaded packages

```
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = 3)
library(ggplot2)
library(aaltobda)
library(shinystan)
library(bayesplot)
library(loo)
library(dlookr)
library(corrplot)
library(rstanarm)
library(projpred)
library(GGally)
library(shiny)
library(gridExtra)
library(caret)
library(e1071)
library(pROC)
SEED <- 48927
```

# Introduction

Breast cancer most commonly presents as a lump that feels different from the rest of the breast tissue. More than 80% of cases are discovered when a person detects such a lump with the fingertips and there are various methods of assessing if the detected lump is a cyst, and if so, either benign or malignant.

One such method of detecting the dangerousness of the mass is Fine-needle Aspiration (FNA). This diagnostic procedure consists of a very safe and minor procedure, where a thin hollow needle is inserted into the mass for obtaining cell samples and analyzing them under a microscope. A major surgical biopsy can be avoided by performing FNA, which is safer and far less traumatic, and possibly eliminating the need for hospitalization and other complications.

The problem presented in this report deals with finding out which features of a FNA are more relevant in diagnosing a patient's mammary lump as benign or malignant. The data used is the Breast Cancer Wisconsin (Diagnostic) Data Set, a data set whose features are computed from digitized images of FNAs of breast mass.

### Main Modeling Idea

As a summary, our modeling idea has been to do the following analysis' with different types of models:

1. **Linear Model**, with each of the remaining 18 features, to see which individual variable might have more influence in predicting correct diagnosis.
2. **Multivariate Models**, done with 3 blocks of 6 variables from mean, se and worst, to see which variable block might have more influence in predicting correct diagnosis.
3. **Multivariate Model**, done with all 18 features in order to obtain proper posterior checking, and see the minimum optimal amount of variables needed, and which ones, for equal or better prediction as this model with all 18 features.
4. **Multivariate Model**, done with the best minimum optimal amount of variables, to check if the predictions obtained are equal or better than the model with all variables.
5. **Multivariate Model**, done with Regularized Horseshoe Prior with the best minimum optimal amount of variables, in order to check if the predictions obtained are equal or better than the model with all variables.
6. **Gaussian Model**, done with the four best variables obtained from applying the Multivariate model in order to compare the results of applying it to the dataset and check if it performs better with the selected variables than the considered alternatives.

## Exploratory Data Analysis

The breast cancer dataset consists of 569 FNA procedures each with 32 features. Ten real-valued features are computed for each cell nucleus:

a) radius
b) texture
c) perimeter
d) area
e) smoothness
f) compactness
g) concavity
h) concave points
i) symmetry
j) fractal dimension

From these, the mean, standard error and worst, are computed, thus making 10 features into 30. The other 2 features are the FNA ID number and the diagnosis, the target binary variable, which has values 'M' (malignant) or 'B' (benign).

```
cancer_data = read.csv('cancer.csv')
head(cancer_data)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
## 5         0.10030          0.13280         0.1980             0.10430
## 6         0.12780          0.17000         0.1578             0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave.points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1     0.03003             0.006193        25.38         17.33          184.60
## 2     0.01389             0.003532        24.99         23.41          158.80
## 3     0.02250             0.004571        23.57         25.53          152.50
## 4     0.05963             0.009208        14.91         26.50           98.87
## 5     0.01756             0.005115        22.54         16.67          152.20
## 6     0.02165             0.005082        15.47         23.75          103.40
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1     2019.0           0.1622            0.6656          0.7119
## 2     1956.0           0.1238            0.1866          0.2416
## 3     1709.0           0.1444            0.4245          0.4504
## 4      567.7           0.2098            0.8663          0.6869
## 5     1575.0           0.1374            0.2050          0.4000
## 6      741.6           0.1791            0.5249          0.5355
##   concave.points_worst symmetry_worst fractal_dimension_worst  X
## 1               0.2654         0.4601                0.11890 NA
## 2               0.1860         0.2750                0.08902 NA
## 3               0.2430         0.3613                0.08758 NA
## 4               0.2575         0.6638                0.17300 NA
## 5               0.1625         0.2364                0.07678 NA
## 6               0.1741         0.3985                0.12440 NA
```

```
nrows <- nrow(cancer_data)
ncols <- ncol(cancer_data)
cat("Breast cancer dataset size:", nrows, "rows x", ncols, "cols")
```

```
## Breast cancer dataset size: 569 rows x 33 cols
```

As an addendum when talking about the amount of features the dataset has, it can be noticed that instead of the 32 features mentioned, there are really 33 columns, this is because the last column, called 'X', is a pointless column present filled with 'N/A' values. This column will be dropped.

```
cancer_data$X <- NULL
cat("Dataset contains NULL values:", any(is.na(cancer_data)), '\n')
```

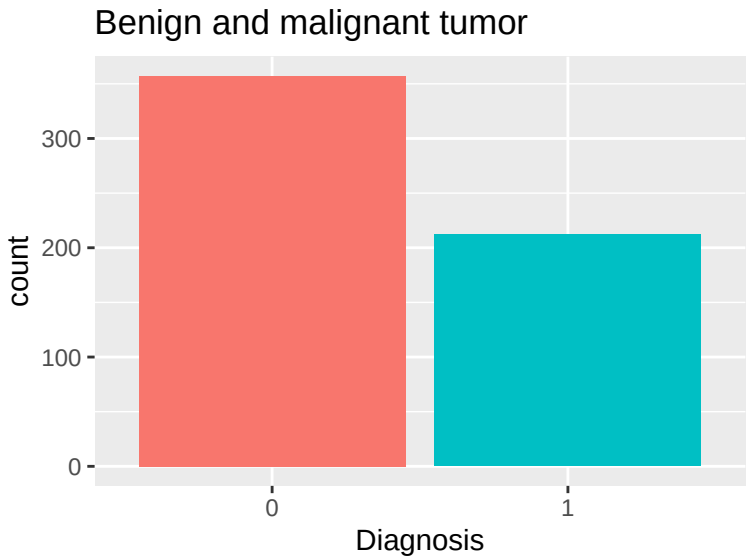
```
## Dataset contains NULL values: FALSE
```

In order to present our data into an numeric format, we convert the target output into binary values (B into '0' and M into '1')

```
cancer_data[cancer_data == "B"] <- as.numeric(0)
cancer_data[cancer_data == "M"] <- as.numeric(1)
cancer_data$diagnosis <- as.numeric(as.character(cancer_data$diagnosis))
```

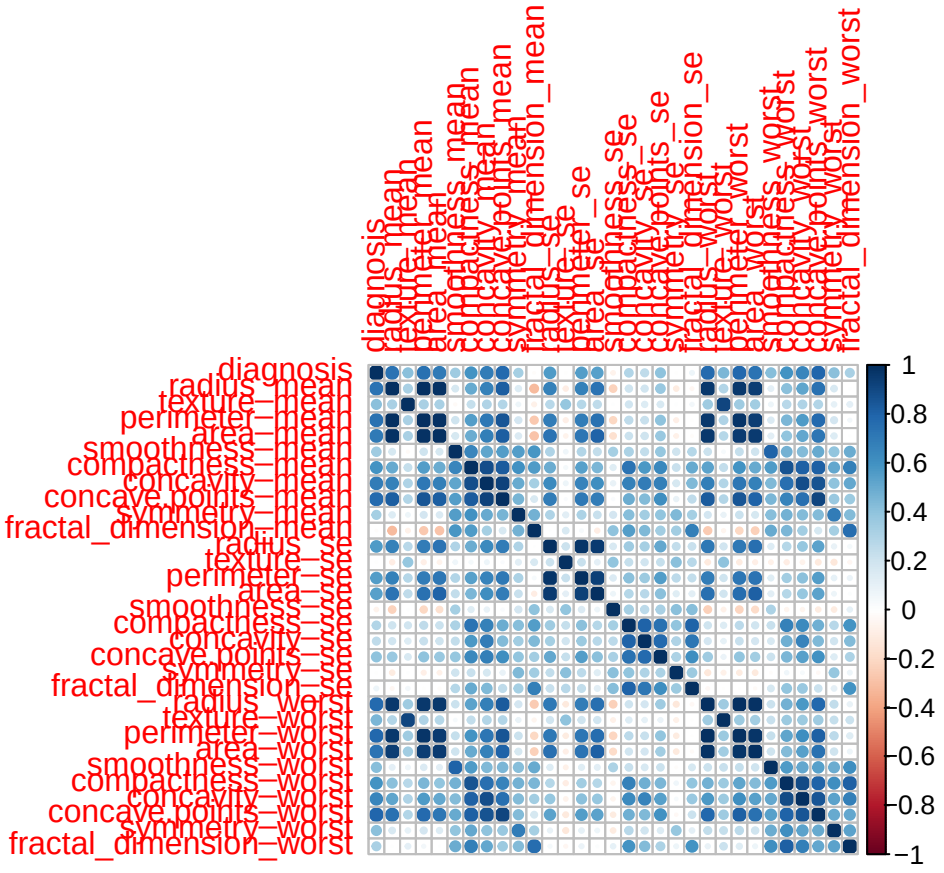
Is our data balanced? We would like to know more about how many benign and malignant tumors are contained in our dataset by visualizing a barplot.

```
ggplot(cancer_data, aes(factor(diagnosis), fill=as.factor(diagnosis))) +
      geom_bar() + theme(legend.position="none") + xlab("Diagnosis") +
      ggtitle("Benign and malignant tumor")
```




A correlation matrix is visualized to obtain correlation coefficients between variables.

```
corrplot(cor(cancer_data[,2:32]))
```

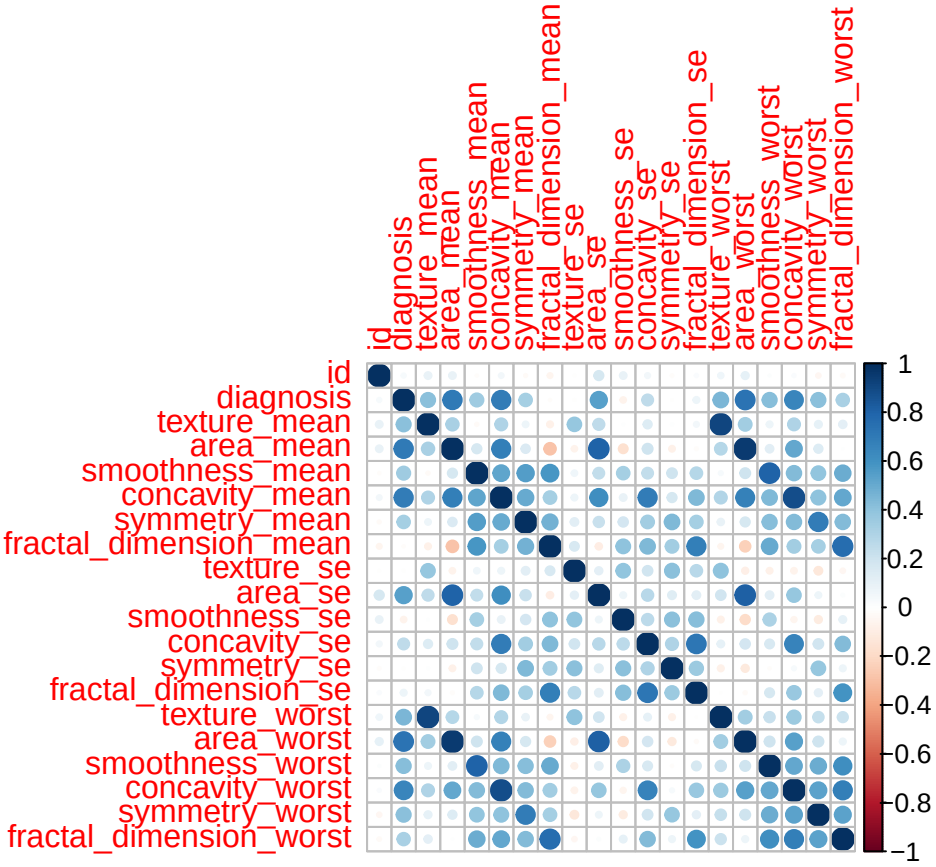



As seen in the correlation matrix, there are some variables that are *highly* correlated. Since they do not provide any new information, we could apply feature selection. For instance, `radius_mean`, `perimeter_mean` and `area_mean` are highly correlated, so we decide to keep `area_mean` in our dataset. Moreover, `compactness_mean`, `concavity_mean` and `concave.points_mean` are correlated, so we decide to keep `concavity_mean`. This process is also carried out for the standard deviation and worst related variables.

```
# Dropping columns using the subset function. '-' indicates dropping vars.
cancer_data = subset(cancer_data, select = -c(radius_mean, perimeter_mean,
                     compactness_mean, concave.points_mean, radius_se,
                     perimeter_se, compactness_se, concave.points_se,
                     radius_worst, perimeter_worst, compactness_worst,
                     concave.points_worst))

variables <- c("texture_mean", "area_mean", "smoothness_mean", "concavity_mean",
               "symmetry_mean", "fractal_dimension_mean", "texture_se",
               "area_se", "smoothness_se", "concavity_se", "symmetry_se",
               "fractal_dimension_se", "texture_worst", "area_worst",
               "smoothness_worst", "concavity_worst", "symmetry_worst",
               "fractal_dimension_worst")
```

By performing this feature selection, we drastically removed the number of variables, from 30 to 18. The final variables, along with the correlation, used for our models are as follows:

```
corrplot(cor(cancer_data))
```




There are still some variables that are *slightly* correlated, but a more curated feature selection will be made later on in order to create our models.

Now, we would like to **scale** the columns of our data into the range [0,1] for easier comparison, except the diagnosis and id column. Thus, each column has a mean of 0 with a standard deviation of 1.

```
scaled_cancer_data <- cancer_data
col_names <- colnames(cancer_data) # Keep column names

scaled_cancer_data[,1:2] <- cancer_data[,1:2] # Id and diagnosis without scaling
scaled_cancer_data[,3:18] <- scale(cancer_data[,3:18])
colnames(scaled_cancer_data) <- col_names # Retrieve column names
head(scaled_cancer_data)
```

```
##         id diagnosis texture_mean  area_mean smoothness_mean concavity_mean
## 1   842302         1   -2.0715123  0.9835095       1.5670875     2.65054179
## 2   842517         1   -0.3533215  1.9070303      -0.8262354    -0.02382489
## 3 84300903         1    0.4557859  1.5575132       0.9413821     1.36227979
## 4 84348301         1    0.2535091 -0.7637917       3.2806668     1.91421287
## 5 84358402         1   -1.1508038  1.8246238       0.2801253     1.36980615
## 6   843786         1   -0.8346009 -0.5052059       2.2354545     0.86554001
##   symmetry_mean fractal_dimension_mean texture_se    area_se smoothness_se
## 1   2.215565542              2.2537638 -0.5647681  2.4853907    -0.2138135
## 2   0.001391139             -0.8678888 -0.8754733  0.7417493    -0.6048187
```

```
## 3   0.938858720             -0.3976580 -0.7793976  1.1802975    -0.2967439
## 4   2.864862154              4.9066020 -0.1103120 -0.2881246     0.6890953
## 5  -0.009552062             -0.5619555 -0.7895490  1.1893103     1.4817634
## 6   1.004517928              1.8883435 -0.5921406 -0.2890039     0.1562093
##   concavity_se symmetry_se fractal_dimension_se texture_worst area_worst
## 1    0.7233897   1.1477468           0.90628565   -1.35809849  1.9994782
## 2   -0.4403926  -0.8047423          -0.09935632   -0.36887865  1.8888270
## 3    0.2128891   0.2368272           0.29330133   -0.02395331  1.4550043
## 4    0.8187979   4.7285198           2.04571087    0.13386631 -0.5495377
## 5    0.8277425  -0.3607748           0.49888916   -1.46548091  1.2196511
## 6    0.1598845   0.1340009           0.48641784   -0.31356043 -0.2441054
##   smoothness_worst concavity_worst symmetry_worst fractal_dimension_worst
## 1        1.3065367       2.1076718         0.4601                 0.11890
## 2       -0.3752817      -0.1466200         0.2750                 0.08902
## 3        0.5269438       0.8542223         0.3613                 0.08758
## 4        3.3912907       1.9878392         0.6638                 0.17300
## 5        0.2203623       0.6126397         0.2364                 0.07678
## 6        2.0467119       1.2621327         0.3985                 0.12440
```

# Priors

We have chosen to work with Weakly Informative Priors because:

1. A weakly informative prior means a reasonable representation of partial ignorance about the data, but that still includes a small amount of real-world information. Uniformity of distribution on an appropriate measurement scale means that the prior does not strongly favor particular values of the parameter.
2. A weakly informative prior does not contribute strongly to the posterior. With a weakly informative prior we say that this small contribution from the prior "lets the data speak for itself".

Because the target feature, the diagnosis, has binary values we will use a Bernoulli logistic regression approach:

$$y \sim Bernoulli(y \mid logit^{-1}(\alpha + \beta \times x)),$$

with $\alpha$ and $\beta$ are the intercept and regression coefficients, $x$ are the predictor variables and $y$ is the breast cancer diagnosis.

## Coefficients

For the $\alpha$ intercept coefficient and $\beta$ regression coefficient we have chosen a Normal and a Student T distribution respectively, to be more precise:

1. $\alpha \sim Normal(\mu_\alpha,\ \sigma_\alpha)$, with $\mu_\alpha = 0$ and $\sigma_\alpha = 1$.
2. $\beta \sim StudentT(df_\beta,\ location_\beta,\ scale_\beta)$, with $df_\beta = 3$, $location_\beta = 0$ and $scale_\beta = 1$.

The reason behind this choice in priors for the intercept coefficient $\alpha$ is because we scale our data, so it makes sense to select a normal distribution with mean centered in 0 and standard deviation 1. On the other hand, for the regression coefficient $\beta$ we chose a StudenT distribution with 3 degrees of freedom and scale 1 because we want to ensure a finite variance and also a finite mean, which will create thick tails that reflect our uncertainty in the prior scale.

```
beta <- student_t(df = 3, location = 0, scale = 1)
alpha <- normal(0, 1)
```

### Regularized Horseshoe Prior

We have also used a Weakly Regularized Horseshoe Prior, as *Piironen+Vehtari:RHS:2017*, where we included the prior assumption that a number of variables are more relevant that the rest. The assumption origin will be discussed in a future section, but as a summary, it has been assumed that there are 3 relevant variables, while the others are irrelevant.

$$\begin{aligned}
\beta_m &\sim N(0, \tau \times \tilde{\lambda}) \\
\tilde{\lambda}_m &\sim \frac{c \times \tilde{\lambda}_m}{\sqrt{c^2 + \tau^2 \times \tilde{\lambda}_m^2}} \\
\lambda_m &\sim HalfCauchy(0, 1) \\
c^2 &\sim InvGamma(\frac{v}{2}, \frac{v}{2} \times s^2) \\
\tau &\sim HalfCauchy(0, \tau_0)
\end{aligned}$$

```
p_0 <- 3 # number of relevant variables
tau_0 <- p_0 / (length(variables) - p_0) * 1 / sqrt(nrows)
rhs_prior <- hs(global_scale = tau_0)
```

### Gaussian Model

Gaussian processes are parameterized by a mean vector and a covariance matrix. The chosen kernel for implementing the covariance is the exponentiated quadratic kernel, which depends on two hyperpriors, the length scale, $\rho$, and the marginal standard deviation $\alpha$. The latter measures the average distance from the mean function, while the former represents the frequency of the function.

$$k(x|\alpha, \rho)_{ij} = \alpha^2 exp(\frac{-1}{2\rho^2} \sum_{d=1}^{D} (x_{i,d} - x_{j,d})^2)$$

Hence, the priors and hyperpriors are as following:

$$\rho \sim InvGamma(1, 1) \alpha \sim Normal(0, 1) a \sim Normal(0, 1) \eta \sim Normal(0, 1)$$

Where the vector eta is added to increase the efficiency of the sampling.

## Models

Throughout this section, the designed modeling ideas will be presented in its respective subsection: explanation of the model and code.

## Linear Models

Per each variable contained in the data set, a separate linear model is created in order to observe its efficiency and relevance.

### Stan code

The common **stan code** for each variable is designed as follows:

```
writeLines(readLines("models/linear_model_bernoulli.stan"))
```

```
## data {
##   int<lower=0> N;
##   vector[N] x;
##   int<lower=0,upper=1> y[N];
## }
## parameters {
##   real alpha;
##   real beta;
## }
## model {
##   alpha ~ normal(0, 1);
##   beta ~ student_t(3, 0, 1);
##   y ~ bernoulli_logit(alpha + beta * x);
## }
## generated quantities{
##   vector[N] log_lik;
##   int<lower=0,upper=1> y_sim[N];
##
##   for (i in 1:N){
##     log_lik[i] = bernoulli_logit_lpmf(y[i] | alpha + beta * x[i]);
##     y_sim[i] = bernoulli_logit_rng(alpha + beta * x[i]);
##   }
## }
```

```
linear_model <- rstan::stan_model(file = "models/linear_model_bernoulli.stan")
```

As seen in the stan model above, the priors are $\alpha \sim N(0,1)$ for the intercept coefficient and $\beta \sim StudentT(3,0,1)$ for the regression coefficient. The likelihood has been calculated using `bernoulli_logit` for the binary outcome. In order to compute the log-likelihood of the model `bernoulli_logit_lpmf` has been used as well as `bernoulli_logit_rng` to compute the predictions of the outcome.

### Sampling

The following code draws samples from the defined stan model. Each iteration pertains to a variable of the data set ('texture_mean', 'area_mean', etc...) contained in `variables` variable. Additionally to each of the variables, the sampling also receives as input the number `N` of observations of the dataset and the binary target value (diagnosis) for each of the observations.

```
linear_models <- c()
y <- c(scaled_cancer_data$diagnosis) # Binary outcomes
for (variable in variables){
  cat("Sampling with variable: ", variable, "\n")
  # Data
  linear_model_data <- list(N=nrows, y=y, x=c(scaled_cancer_data[,variable]))

  # Model sampling
  linear_sampling <- rstan::sampling(linear_model, data = linear_model_data,
                                     seed = SEED, refresh=0)

  # Model saved into a list for further analysis later on.
  linear_models <- c(linear_models, linear_sampling)
}
```

```
## Sampling with variable:  texture_mean
## Sampling with variable:  area_mean
## Sampling with variable:  smoothness_mean
## Sampling with variable:  concavity_mean
## Sampling with variable:  symmetry_mean
## Sampling with variable:  fractal_dimension_mean
## Sampling with variable:  texture_se
## Sampling with variable:  area_se
## Sampling with variable:  smoothness_se
## Sampling with variable:  concavity_se
## Sampling with variable:  symmetry_se
## Sampling with variable:  fractal_dimension_se
## Sampling with variable:  texture_worst
## Sampling with variable:  area_worst
## Sampling with variable:  smoothness_worst
## Sampling with variable:  concavity_worst
## Sampling with variable:  symmetry_worst
## Sampling with variable:  fractal_dimension_worst
```

## Multivariate Model - Variable Blocks

Followed by the analysis done for each of the variables, it was decided to create three separate multivariate models, one for each variable block (type of variable): **mean**, **se** and **worst**. This was done in order to observe the efficiency and relevance over the type of the variable and see if some of the measure blocks shouldn't be considered when making the diagnosis.

### Stan code

The common **stan code** used for Variable Blocks is designed as follows:

```
writeLines(readLines("models/linear_model_bernoulli_multivariate.stan"))
```

```
## data {
##   int<lower=0> N;
##   vector[N] x_1;
##   vector[N] x_2;
```

```
##   vector[N] x_3;
##   vector[N] x_4;
##   vector[N] x_5;
##   vector[N] x_6;
##   int<lower=0,upper=1> y[N];
## }
## parameters {
##   real alpha;
##   real beta_1;
##   real beta_2;
##   real beta_3;
##   real beta_4;
##   real beta_5;
##   real beta_6;
## }
## model {
##   alpha ~ normal(0, 1);
##   beta_1 ~ student_t(3, 0, 1);
##   beta_2 ~ student_t(3, 0, 1);
##   beta_3 ~ student_t(3, 0, 1);
##   beta_4 ~ student_t(3, 0, 1);
##   beta_5 ~ student_t(3, 0, 1);
##   beta_6 ~ student_t(3, 0, 1);
##   y ~ bernoulli_logit(alpha + beta_1*x_1 + beta_2*x_2 + beta_3*x_3
##                       + beta_4*x_4 + beta_5*x_5 + beta_6*x_6);
## }
## generated quantities{
##   vector[N] log_lik;
##   int<lower=0,upper=1> y_sim[N];
##
##   for (i in 1:N){
##     log_lik[i] = bernoulli_logit_lpmf(y[i] | alpha + beta_1*x_1[i] + beta_2*x_2[i]
##                                             + beta_3*x_3[i] + beta_4*x_4[i]
##                                             + beta_5*x_5[i] + beta_6*x_6[i]);
##     y_sim[i] = bernoulli_logit_rng(alpha + beta_1*x_1[i] + beta_2*x_2[i]
##                                      + beta_3*x_3[i] + beta_4*x_4[i]
##                                      + beta_5*x_5[i] + beta_6*x_6[i]);
##   }
## }
```

```
linear_model_bernoulli_multivariate <- rstan::stan_model(file = "models/linear_model_bernoulli_multivari
```

As seen in the stan model above, the priors are $\alpha \sim N(0,1)$ for the intercept coefficient and $\beta_n \sim StudentT(3,0,1)$ for each of the regression coefficients. The likelihood has been calculated using `bernoulli_logit` for the binary outcome. In order to compute the log-likelihood of the model `bernoulli_logit_lpmf` has been used as well as `bernoulli_logit_rng` to compute the predictions of the outcome.

For convenience we ended up sampling with `stan_glm`, whose code can be found in [!!!APPENDIX MV BLOCK CODE!!!], because there are many models and we needed to use functions that required `stan_glm` objects as input. For this reason, the sampling with the `.stan` code is commented.

### Sampling

The following code draws samples from the defined stan model. Each iteration pertains to a variable block ('mean', 'se' and 'worst'), comprised of 6 variables for each of them ('texture', 'area', 'smoothness', 'concavity', 'symmetry', 'fractal_dimension'), which are passed as predictors to the model. Additionally to each of the variable blocks, the sampling also receives as input the number `N` of observations of the dataset and the binary target value (diagnosis) for each of the observations.

Additionally to the `rstan::sampling` models obtained, the same models were fitted with `stan_glm`, in order to be able to make use of functions that need the input to be a stan_glm object. Both types of model objects have been saved for posteriority, for obtaining the needed diagnostics.

```
mv_mean_block <- list("texture_mean", "area_mean", "smoothness_mean",
                      "concavity_mean", "symmetry_mean",
                      "fractal_dimension_mean")

mv_se_block <- list("texture_se", "area_se", "smoothness_se", "concavity_se",
                       "symmetry_se", "fractal_dimension_se")

mv_worst_block <- list("texture_worst", "area_worst", "smoothness_worst",
                       "concavity_worst", "symmetry_worst",
                       "fractal_dimension_worst")

mv_model_names <- c("Mean Model", "SE Model", "Worst Model",
                    "Full Variables Model", "Three variables Model",
                    "RHS Model")

mv_var_block <- list(mv_mean_block, mv_se_block, mv_worst_block)

# multivariate_models <- c()
multivariate_glms <- list()

for (i in (1:3)){
  var_block <- mv_var_block[[i]]
  print(paste("diagnosis ~", paste(var_block, collapse = "+")))
  multivariate_block_data <- list(N=length(scaled_cancer_data$diagnosis),
                                  y=c(scaled_cancer_data$diagnosis),
                                  x_1=scaled_cancer_data[,var_block[[1]]],
                                  x_2=scaled_cancer_data[,var_block[[2]]],
                                  x_3=scaled_cancer_data[,var_block[[3]]],
                                  x_4=scaled_cancer_data[,var_block[[4]]],
                                  x_5=scaled_cancer_data[,var_block[[5]]],
                                  x_6=scaled_cancer_data[,var_block[[6]]])

  # multivariate_sampling <- rstan::sampling(linear_model_bernoulli_multivariate,
  #                                          data = multivariate_block_data,
  #                                          seed = SEED, refresh=0)
  #
  # multivariate_models <- c(multivariate_models, multivariate_sampling)

  multivariate_glm <- stan_glm(paste("diagnosis ~", paste(var_block, collapse = "+")),
                                  data = scaled_cancer_data,
                                  family = binomial(link = "logit"),
                                  prior = beta, prior_intercept = alpha, QR=TRUE,
```

```
                                seed = SEED, refresh=0)
  multivariate_glms <- append(multivariate_glms, list(multivariate_glm))
}
```

```
## [1] "diagnosis ~ texture_mean+area_mean+smoothness_mean+concavity_mean+symmetry_mean+fractal_dimensi
## [1] "diagnosis ~ texture_se+area_se+smoothness_se+concavity_se+symmetry_se+fractal_dimension_se"
## [1] "diagnosis ~ texture_worst+area_worst+smoothness_worst+concavity_worst+symmetry_worst+fractal_di
```

## Multivariate Model - Full Model

Additionally, we decided to carry out a more sensible approach, which was to start by creating and fitting a model which contains all of the 18 variables.

The reasoning behind this model is to carry out projection predictive variable by analyzing the coefficient marginal posteriors and the solution terms, in order to obtain possible information regarding the optimal number of variables that should be used for the data and which of the variables (and in which order) have more weight in the data.

### Sampling

For this model, we used `generalized linear model`, `stan_glm`, using the same prior distributions for the coefficients and intercept as the previous models, this is: $\alpha \sim N(0,1)$ for the intercept coefficient and $\beta_n \sim StudentT(3,0,1)$ for the regression coefficient.

In order for the model to compute the predictions of the diagnosis outcome we used binomial logistic regression, by making use of `stan_glm`'s `family` for which we chose `binomial` with `link = "logit"`.

The formula is the one stated above, and includes the predictive and target variables.

```
multivariate_full <- stan_glm(diagnosis ~ texture_mean + area_mean + smoothness_mean + concavity_mean
                              + symmetry_mean + fractal_dimension_mean
                              + texture_se + area_se + smoothness_se + concavity_se
                              + symmetry_se + fractal_dimension_se
                              + texture_worst + area_worst + smoothness_worst + concavity_worst
                              + symmetry_worst + fractal_dimension_worst,
                              data = scaled_cancer_data,
                              family = binomial(link = "logit"),
                              prior = beta, prior_intercept = alpha, QR=TRUE,
                              adapt_delta=0.9999, iter=5000,
                              seed=SEED, refresh=0)
multivariate_glms <- append(multivariate_glms, list(multivariate_full))
```

### Stan Code

CAMBIAR ESTO As seen in the stan model below, the priors are $\alpha \sim N(0,1)$ for the intercept coefficient and $\beta_n \sim StudentT(3,0,1)$ for each of the regression coefficients. The likelihood has been calculated using `bernoulli_logit` for the binary outcome. In order to compute the log-likelihood of the model `bernoulli_logit_lpmf` has been used as well as `bernoulli_logit_rng` to compute the predictions of the outcome.

```
#get_stanmodel(multivariate_full$stanfit)
```

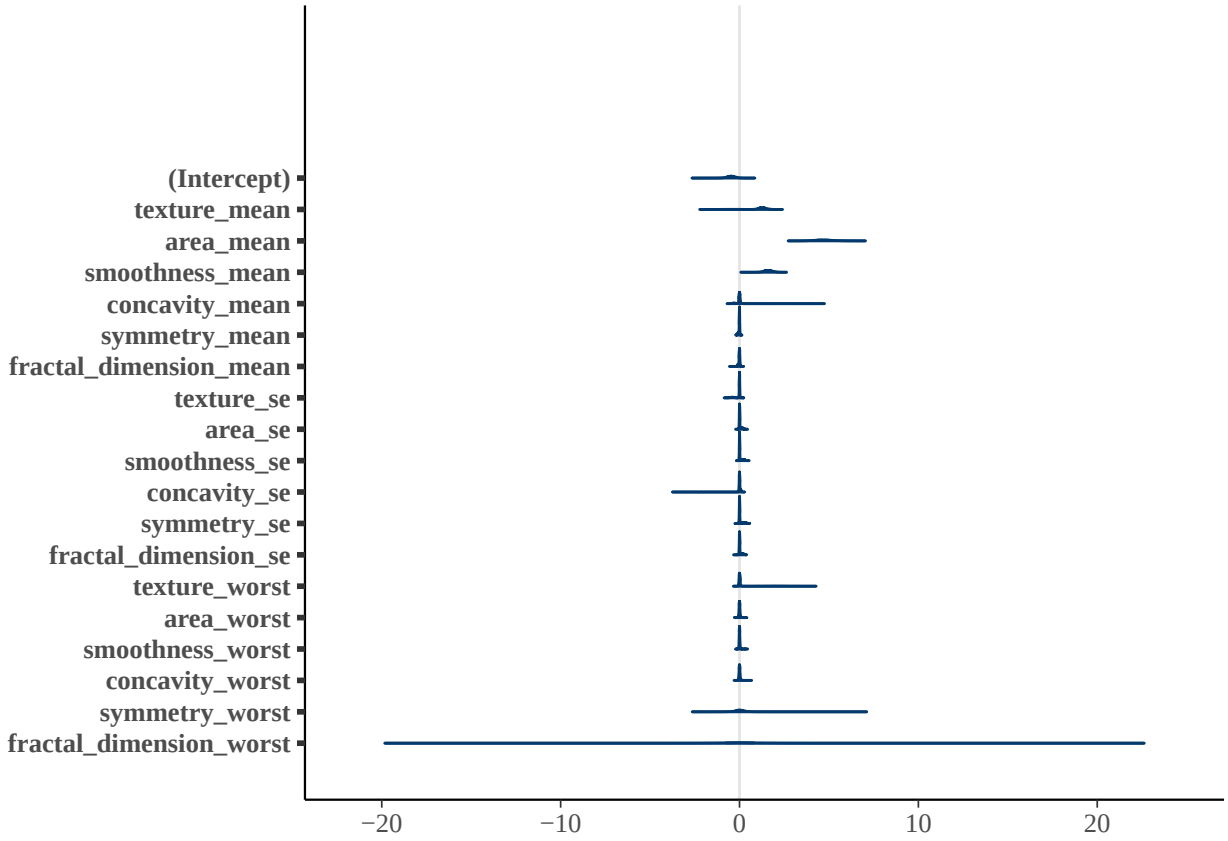
### Coefficient Marginal posteriors

For projection predictive variable selection we firstly analyzed the coefficient marginal posteriors by obtaining the plot below making use of `mcmc_areas`.

We can see that the posterior coefficient for a few variables is far away from zero or have wide margins, but it's not clear what variables should be included, or how many. Some, such as `texture_mean` and `smoothness_mean`, are hinted as potentially relevant variables with correlation in joint posterior.

Just looking at the marginals has the problem that it can perfectly occur that correlating variables have marginal posteriors overlapping zero or far from zero, but joint posterior typically does not include zero. Because of this, further projection predictive variable selection must be carried out.

```
mcmc_areas(as.matrix(multivariate_full))
```

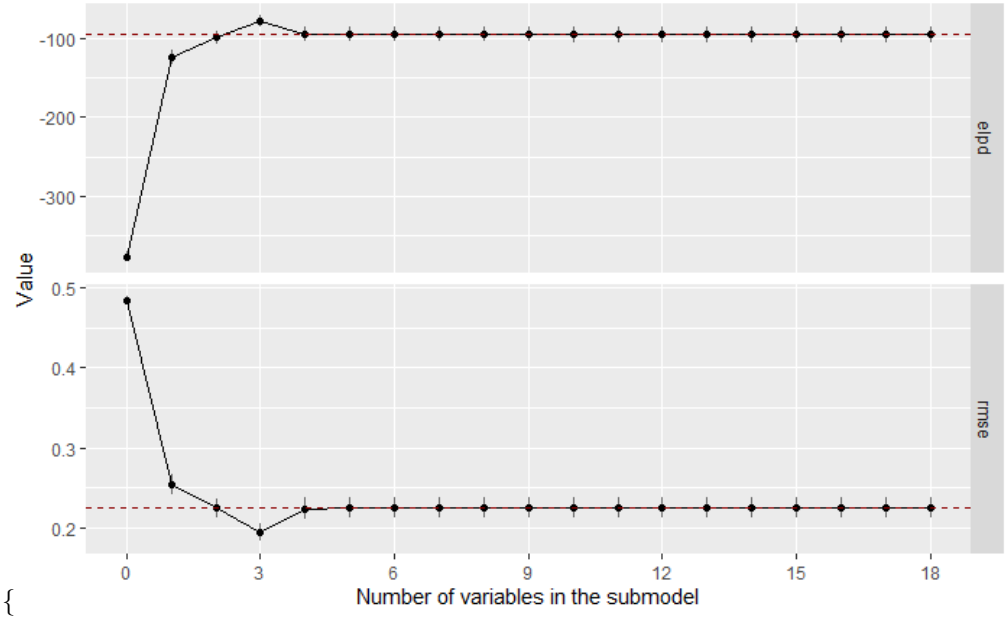



### Solution Terms

Because we want to subject all variables to selection, we continue with projection predictive variable selection which can be easily done with `cv_varsel` function. This also computes a LOO-CV estimate of the predictive performance for the best models with a certain number of variables.

As mentioned above, one of the reasonings behind using this full model is to obtain the optimal number of variables for which the predictive performance is the same as the full model or better. By looking at the plot we can see that we can achieve a higher predictive performance using only 3 variables instead of 18. This can be corroborated further by running the `suggest_size` function, which tells us that the optimal number of variables for equal performance is 2, but better results can be achieved with 3.

\begin{figure}[H]




{

}

\caption{cv_varsel function output} \end{figure}

The other reasoning behind the full model was which of the variables are the best for obtaining better predictive performance. We can obtain a ranking of best variables to be inserted in the model with `solution_terms`, and from this ranking we choose the first 3 best variables: `area_worst`, `smoothness_mean` and `texture_mean`.

\begin{figure}[H]

```
[1] "Computing LOOs..."

|================================================================================
 100%
[1] "Performing the selection using all the data.."
[1] "10% of terms selected."
[1] "20% of terms selected."
[1] "30% of terms selected."
[1] "40% of terms selected."
[1] "50% of terms selected."
[1] "60% of terms selected."
[1] "70% of terms selected."
[1] "80% of terms selected."
[1] "90% of terms selected."
[1] "100% of terms selected."
[1] "Done."
[1] 2
 [1] "area_worst"              "smoothness_mean"         "texture_mean"
"area_mean"               "texture_worst"
 [6] "symmetry_se"             "smoothness_worst"        "texture_se"
"smoothness_se"           "fractal_dimension_se"
[11] "fractal_dimension_worst" "concavity_se"            "concavity_mean"
"concavity_worst"         "symmetry_worst"
[16] "symmetry_mean"           "fractal_dimension_mean"  "area_se"
```

{

}

\caption{solution_terms function output} \end{figure}

Note: these functions are computationally complex and require quite a large amount of time to be executed and we needed to run the code many times to fix errors and other reasons. Because of this, the code block has been made to pass the execution, but photos of the correct execution have been inserted.

## Multivariate Model - Three Variables Model

Modelo resultante despues de haber ejecutado cv_varsel

```
multivariate_three <- stan_glm(diagnosis ~ area_worst + smoothness_mean + texture_mean,
                               data = scaled_cancer_data,
                               family = binomial(link = "logit"),
                               prior = beta, prior_intercept = alpha,
                               seed = SEED, refresh=0)
multivariate_glms <- append(multivariate_glms, list(multivariate_three))
```

### Stan Code

```
#get_stanmodel(multivariate_three$stanfit)
```

## Multivariate Model - Regularized Horseshoe Model

```
multivariate_rhsp <- stan_glm(diagnosis ~ texture_mean + area_mean + smoothness_mean + concavity_mean
                              + symmetry_mean + fractal_dimension_mean
                              + texture_se + area_se + smoothness_se + concavity_se
                              + symmetry_se + fractal_dimension_se
                              + texture_worst + area_worst + smoothness_worst + concavity_worst
                              + symmetry_worst + fractal_dimension_worst,
                              data = scaled_cancer_data,
                              family = binomial(link = "logit"),
                              prior=rhs_prior, QR=TRUE,
                              seed = SEED, refresh=0)
multivariate_glms <- append(multivariate_glms, list(multivariate_rhsp))
```

### Stan Code

```
#get_stanmodel(multivariate_rhsp$stanfit)
```

## Gaussian Model

The last model that was considered for this project was a Gaussian Process based model. Gaussian models provide a probability distribution over functions. They are useful for representing those datasets that cannot be easily adjusted to a line following a regression procedure. A gaussian process consists of generating higher degree polynomial functions and assigning to those functions a probability. Therefore, by taking the mean over the probability distribution, the most probable fit is obtained.

### Stan code

The common **stan code** for each variable is designed as follows:

```
writeLines(readLines("models/gaussian_process_bernoulli.stan"))
```

```
## data {
##   int<lower=1> N_obs;
##   real x_obs[N_obs];
##   int<lower=0, upper=1> y_obs[N_obs];
##   int<lower=1> N_predict;
##   real x_predict[N_predict];
## }
## transformed data {
##   real delta = 1e-9;
##   int<lower=1> N = N_obs + N_predict;
##   real x[N];
##   for (n_obs in 1:N_obs) x[n_obs] = x_obs[n_obs];
##   for (n_predict in 1:N_predict) x[N_obs + n_predict] = x_predict[n_predict];
## }
## parameters {
##   real<lower=0> rho;
##   real<lower=0> alpha;
##   real a;
##   vector[N] eta;
## }
## transformed parameters {
##   vector[N] f;
##   {
##     matrix[N, N] L_K;
##     matrix[N, N] K = cov_exp_quad(x, alpha, rho);
##
##     // diagonal elements
##     for (n in 1:N)
##       K[n, n] = K[n, n] + delta;
##
##     L_K = cholesky_decompose(K);
##     f = L_K * eta;
##   }
## }
## model {
##   rho ~ inv_gamma(1, 1);
##   alpha ~ std_normal();
##   a ~ std_normal();
##   eta ~ std_normal();
##
##   y_obs ~ bernoulli_logit(a + f[1:N_obs]);
## }
## generated quantities {
##   int y_predict[N_predict];
##   vector[N_predict] log_lik;
##   for (n_predict in 1:N_predict){
##     y_predict[n_predict] = bernoulli_logit_rng(a + f[N_obs + n_predict]);
##     log_lik[n_predict] = bernoulli_logit_lpmf(y_obs[n_predict] | a + f[N_obs + n_predict]);
##   }
## }
```

In order to later assess the performance of the gaussian model based on the quality of its predictions, the data set is divided into two subsets, one for "training" the model, containing a higher number of observations, and another one for "testing" the model.

```
train_index <- sample(1:nrow(scaled_cancer_data), 0.8*nrow(scaled_cancer_data))
test_index <- setdiff(1:nrow(scaled_cancer_data), train_index)

train_set <-scaled_cancer_data[train_index, ]
test_set <- scaled_cancer_data[test_index, ]
```

The gaussian model was carried out over the four best variables contained in the set of optimal variables obtained using the previous models.

From the training set, 200 randomly selected observations are considered. Gaussian models are computationally slow, hence, using the total number of observations would take much time. From the test set, 100 observations are chosen, also randomly.

# Convergence Diagnostics

## Diagnostic Metrics

The aim of this section is to assess the convergence of our models using a variety of diagnostics:

$\hat{R}$ can be used to monitor if a group of chains has converged to the target distribution. This can be done by comparing the between and within-chain estimate for model parameters. If chains have not converged, meaning that they have not mixed well, $\hat{R}$ will be greater than one. If chains have converged, the $\hat{R}$ value will be virtually one.

We would like to have a diagnostic that tells us when the weights are problematic, as we could have vastly imbalanced weights. In this case, we could use **effective sample size ($S_{eff}$)**, a quantitative measure of efficiency in **importance sampling**. $S_{eff}$ represents the number of independent samples required to obtain an importance sampling estimate with about the same efficiency as if we would have used all the samples.

A divergence arises when the simulated Hamiltonian trajectory departs from the true trajectory as measured by departure of the Hamiltonian value from its initial value. To evaluate this, we can use the function `check_hmc_diagnostics`.

## Linear Model Diagnostics

As seen in the `Appendix A`Appendix A - Linear models `monitor` outputs, all $\hat{R}$ values were 1 or close to 1, which means that the linear models' chains have converged.

The `monitor` function gives us two crude measures of effective sample size called Bulk_ESS and Tail_ESS (an $ESS > 100$ per chain is considered good). Additionally, our ESS values are way higher than 100, which established that our values are good.

Finally, we have obtained 0 divergences for all of our linear models.

## Multivariate Model Diagnostics

ESCRIBIR AQUI SOBRE TODO EN UN MISMO APARTADO

Escribir que se ha aumentado el numero de cadenas,iteraciones y adapt_delta. Escribir el porque

### Gaussian Model Diagnostics

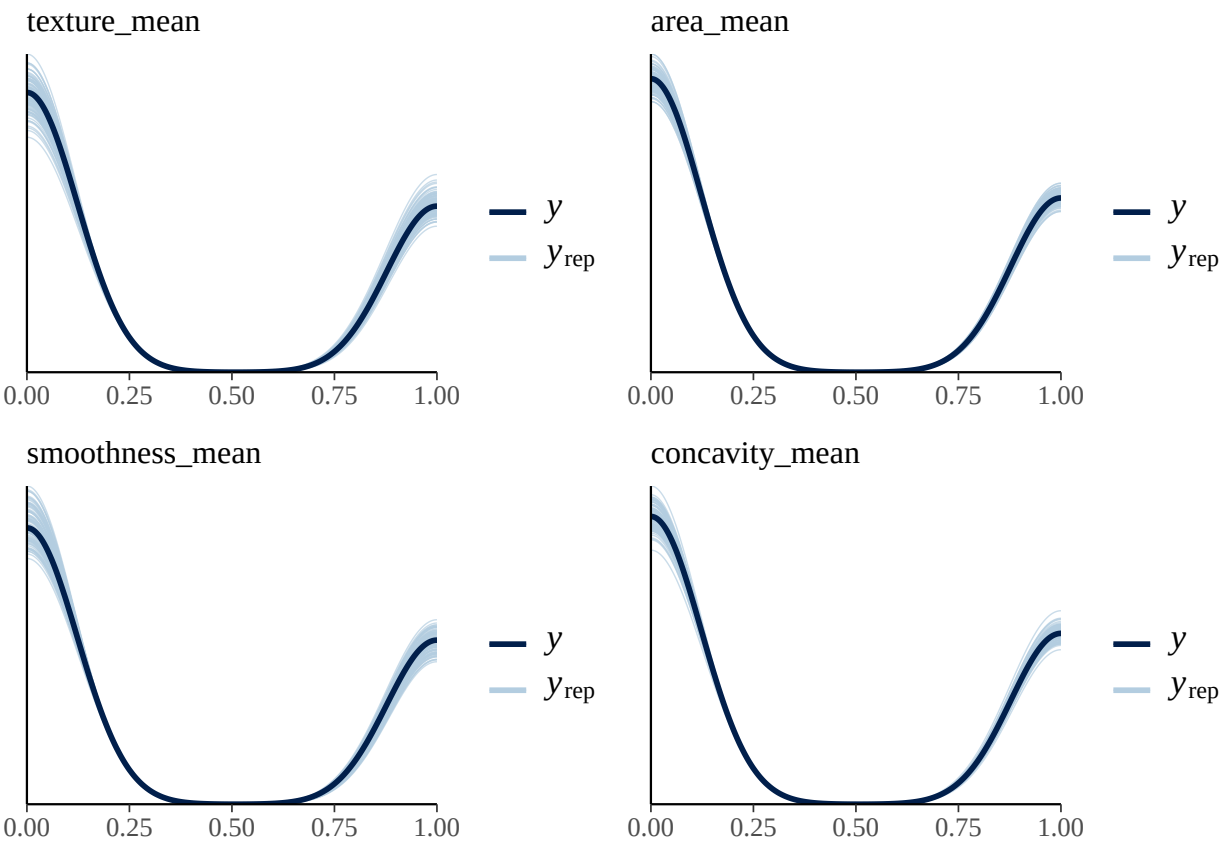
In order to visually assess the convergence of the gaussian model, the chains for the scalar parameters are plotted.

As in previous cases, the values obtained from applying the function `monitor`are satisfactory. $\hat{R}$ provides a valid scalar metric for convergence checking based on computing the division between the pooled variance of the chains and the individual variance of each of them. Obtaining values close to 1, as in the current case, means that the chains happen to have very similar distributions. Finally, the results yielded for the effective sample size are higher than 100.

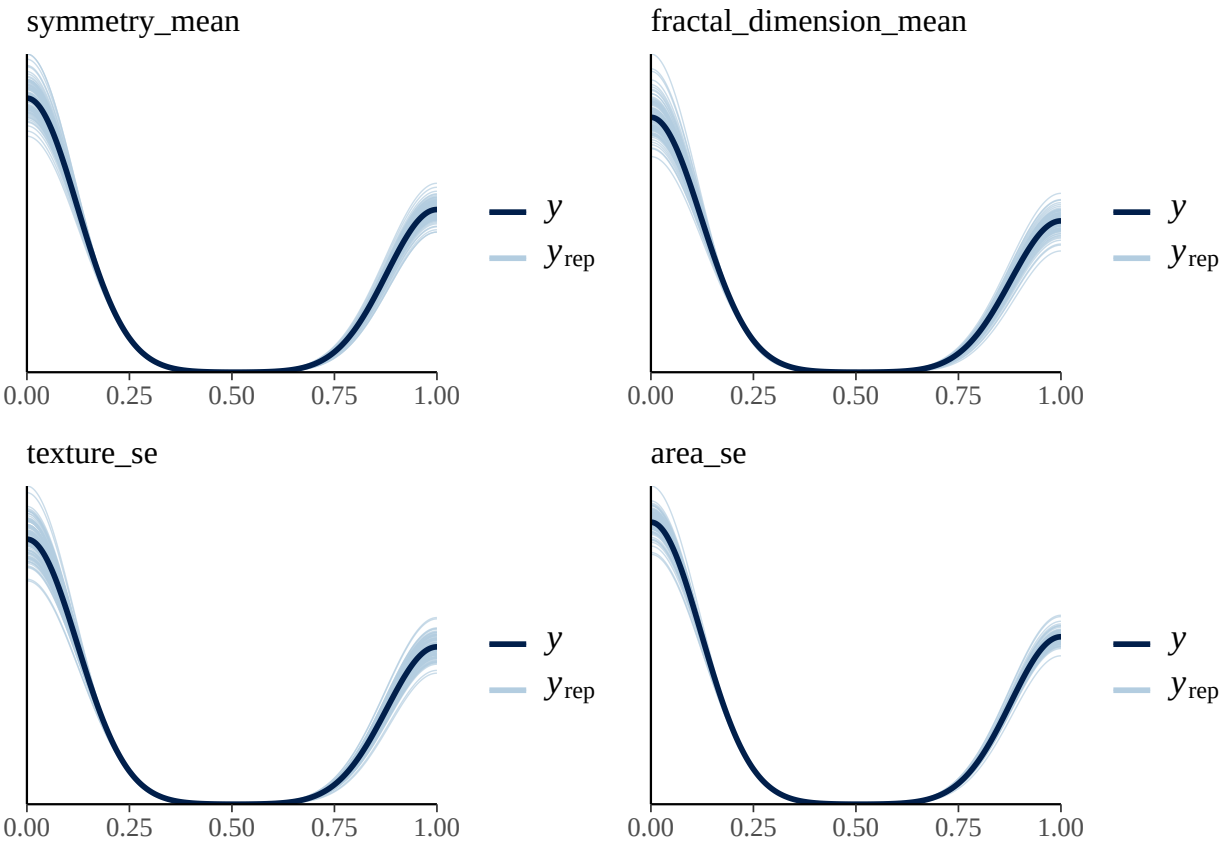
## Posterior Predictive Checking

### Linear Models

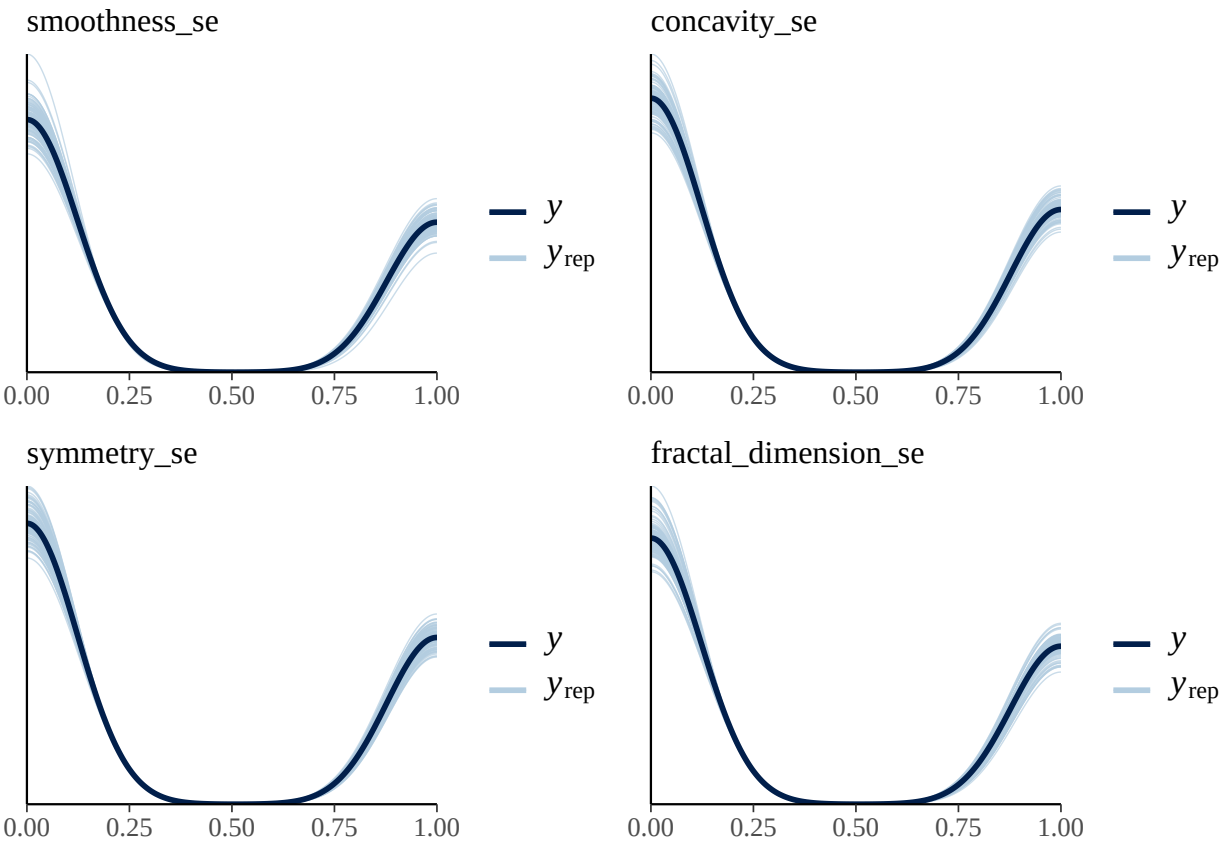
```
y_sims <- list()
for (model in linear_models){
  params <- extract(model)
  y_sims <- append(y_sims, list(params$y_sim[1:100,]))
}

ppc1 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[1]])
ppc2 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[2]])
ppc3 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[3]])
ppc4 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[4]])
ppc5 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[5]])
ppc6 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[6]])
ppc7 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[7]])
ppc8 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[8]])
ppc9 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[9]])
ppc10 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[10]])
ppc11 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[11]])
ppc12 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[12]])
ppc13 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[13]])
ppc14 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[14]])
ppc15 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[15]])
ppc16 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[16]])
ppc17 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[17]])
ppc18 <- ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[18]])

bayesplot_grid(ppc1, ppc2, ppc3, ppc4,
               subtitles = c("texture_mean", "area_mean", "smoothness_mean",
                             "concavity_mean"))
```



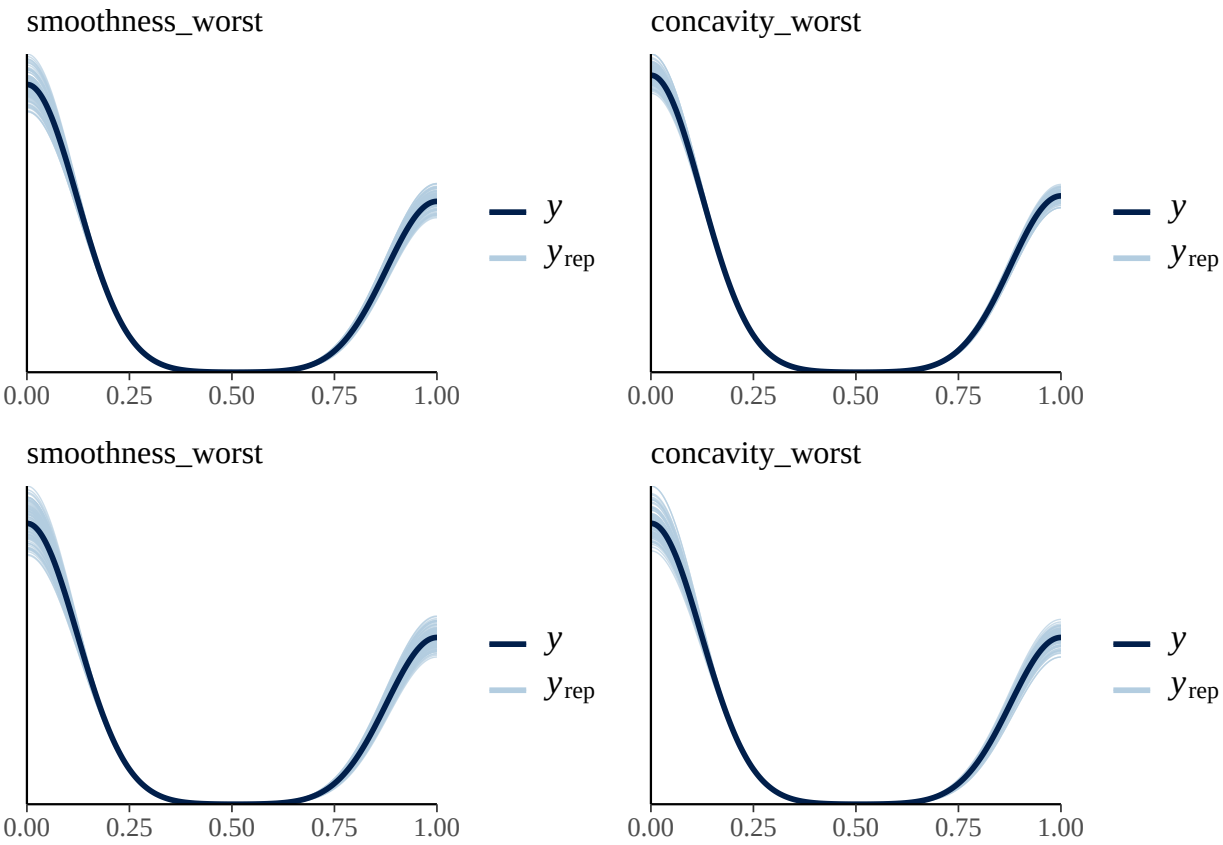

```
bayesplot_grid(ppc5, ppc6, ppc7, ppc8,
               subtitles = c("symmetry_mean", "fractal_dimension_mean",
                             "texture_se", "area_se"))
```



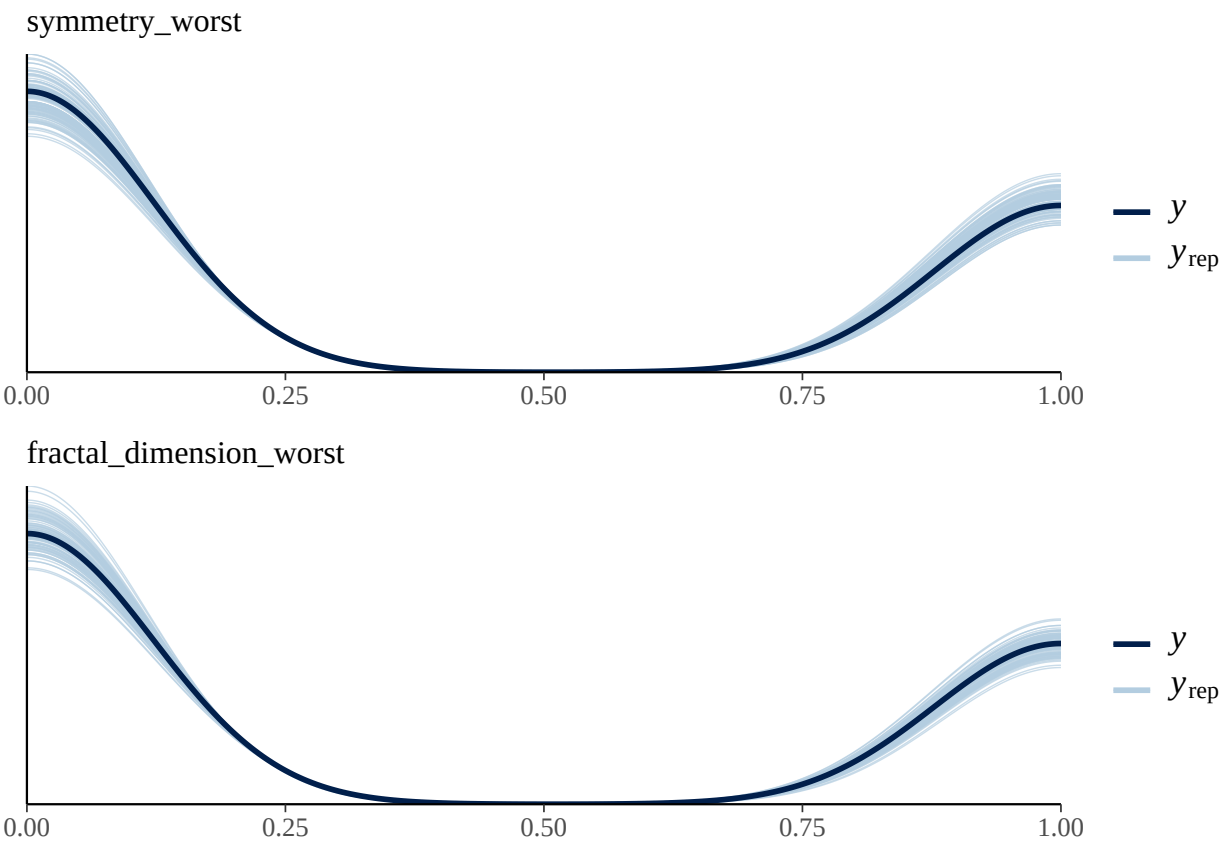

```
bayesplot_grid(ppc9, ppc10, ppc11, ppc12,
               subtitles = c("smoothness_se", "concavity_se", "symmetry_se",
                             "fractal_dimension_se"))
```




```
bayesplot_grid(ppc13, ppc14, ppc15, ppc16, subtitles =
                 c("smoothness_worst", "concavity_worst", "smoothness_worst",
                   "concavity_worst"))
```



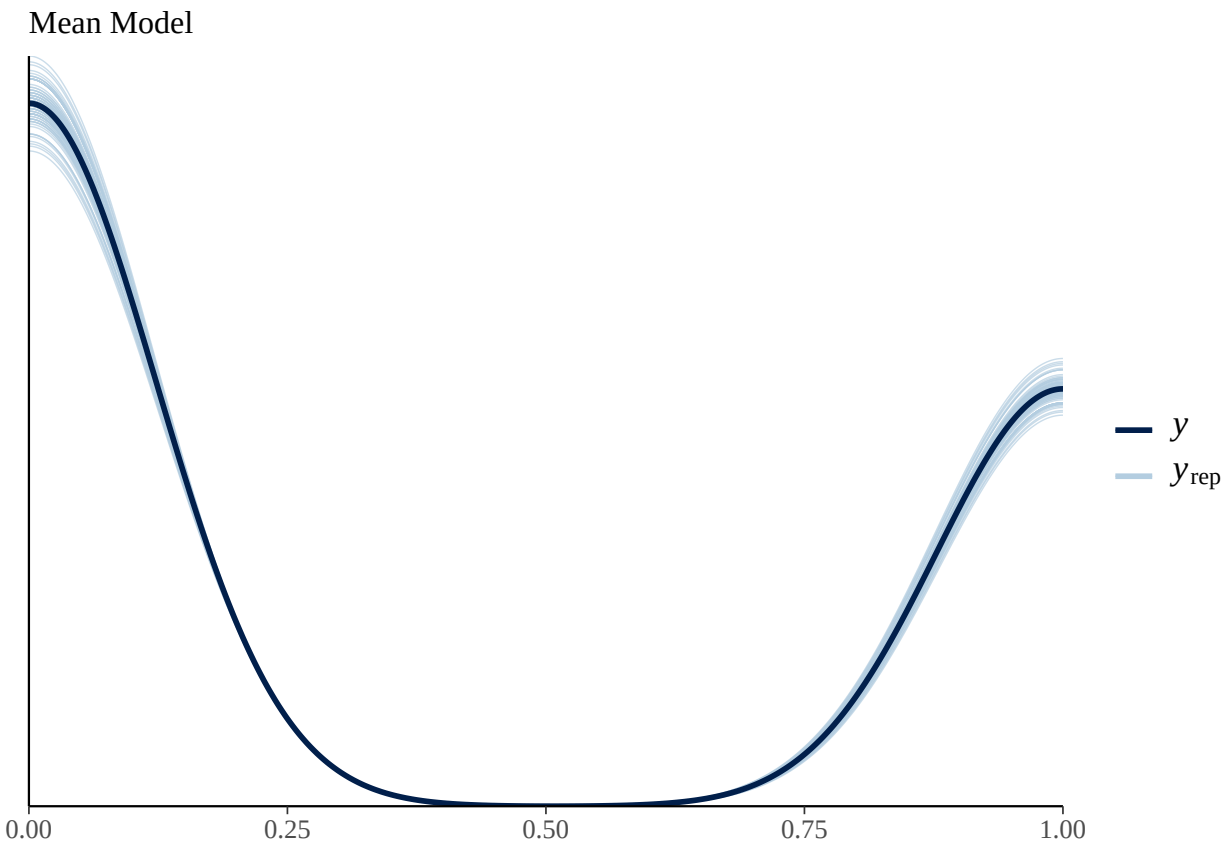

```
bayesplot_grid(ppc17, ppc18, subtitles = c("symmetry_worst",
                                           "fractal_dimension_worst"))
```



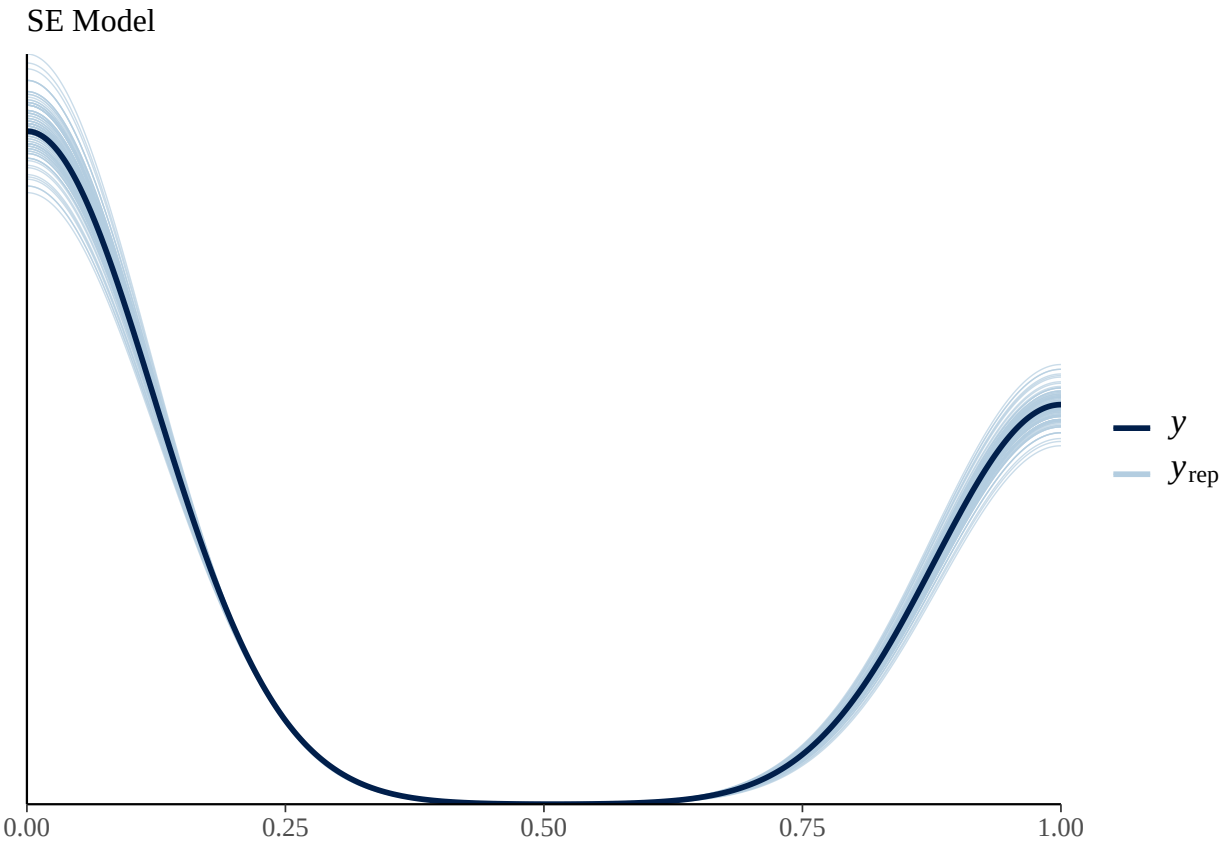

## Multivariate Models

In the posterior predictive checking for each of the variable blocks, it is noticeable how much less the `se` block can influence the fitting of the model and thus getting better predictive results with these variables might not be a viable option.

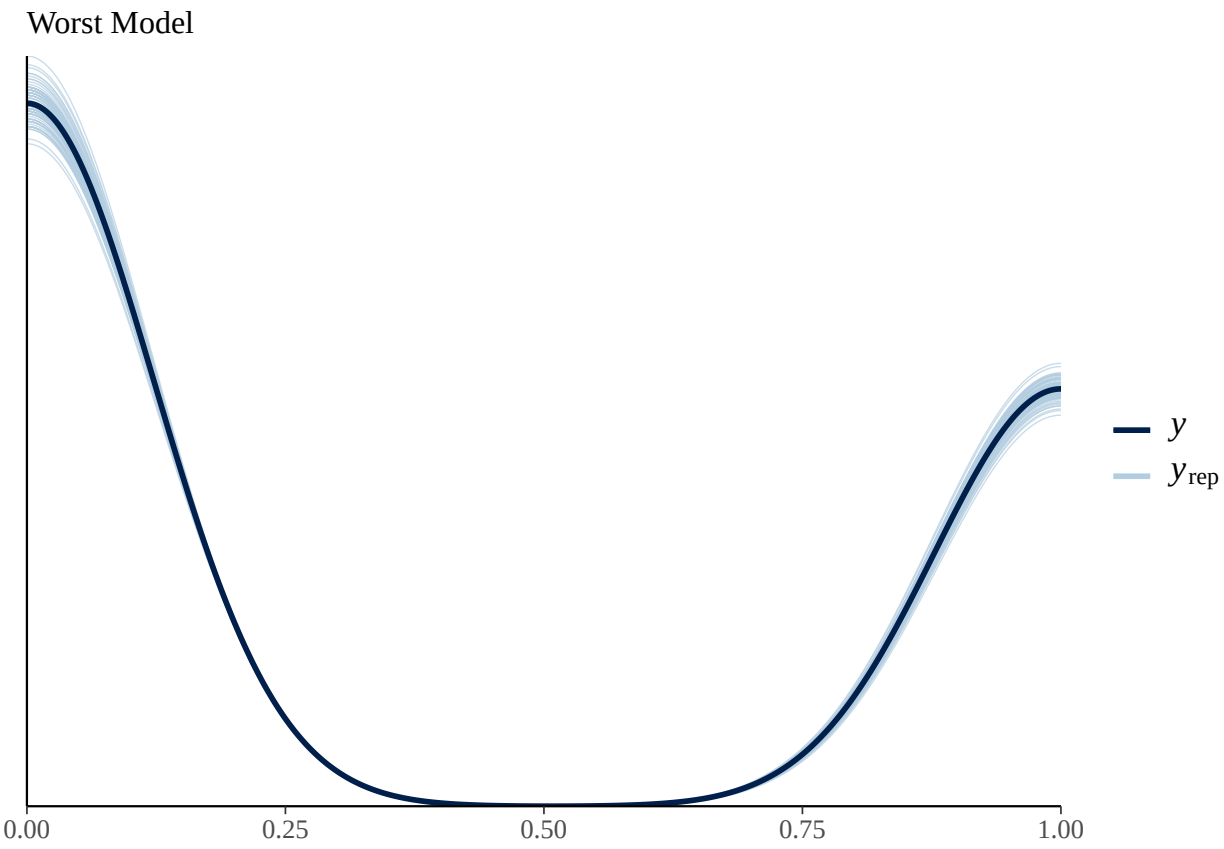
```
y_sims <- list()
for (model in multivariate_glms){
  params <- posterior_predict(model, draws = 100)
  head(params)
  y_sims <- append(y_sims, list(params))
}

bayesplot_grid(ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[1]]),
               subtitles = c("Mean Model"))
```



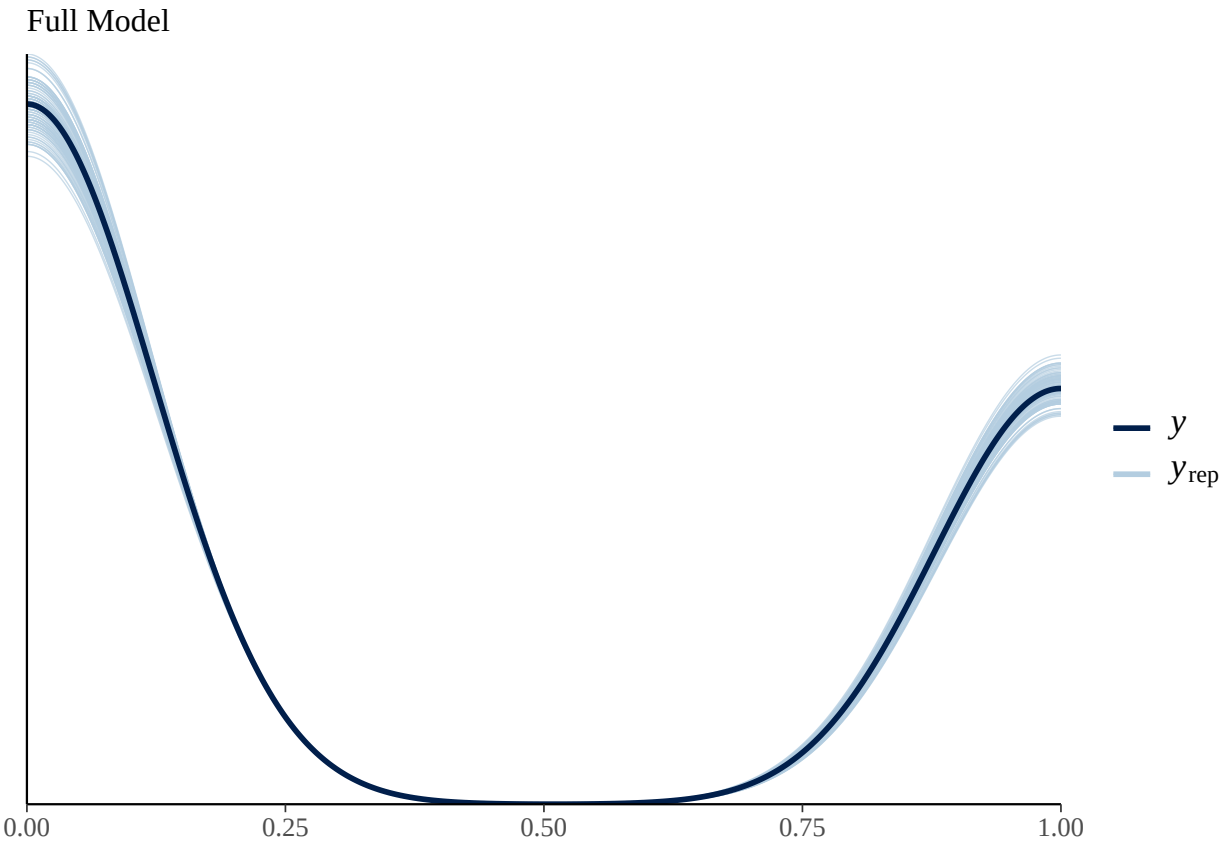

```
bayesplot_grid(ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[2]]),
               subtitles = c("SE Model"))
```



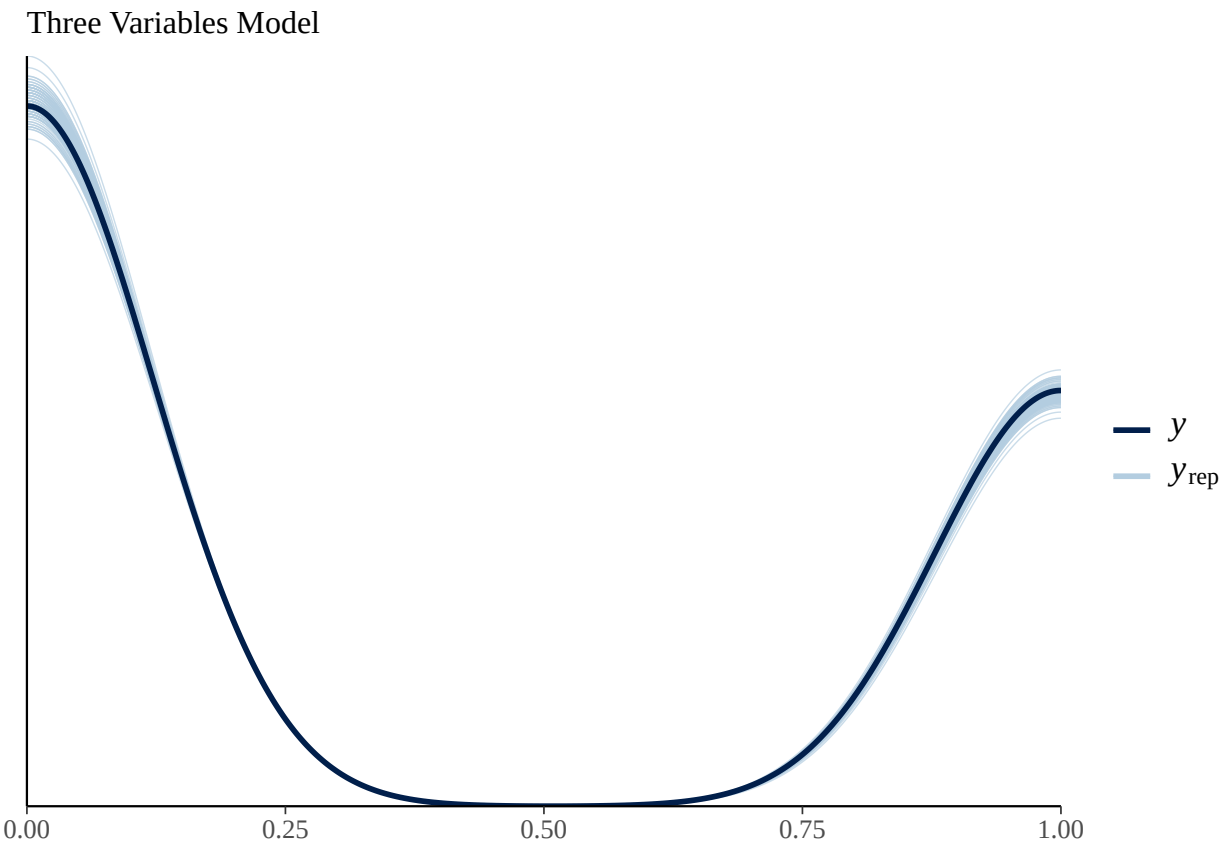

```
bayesplot_grid(ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[3]]),
               subtitles = c("Worst Model"))
```



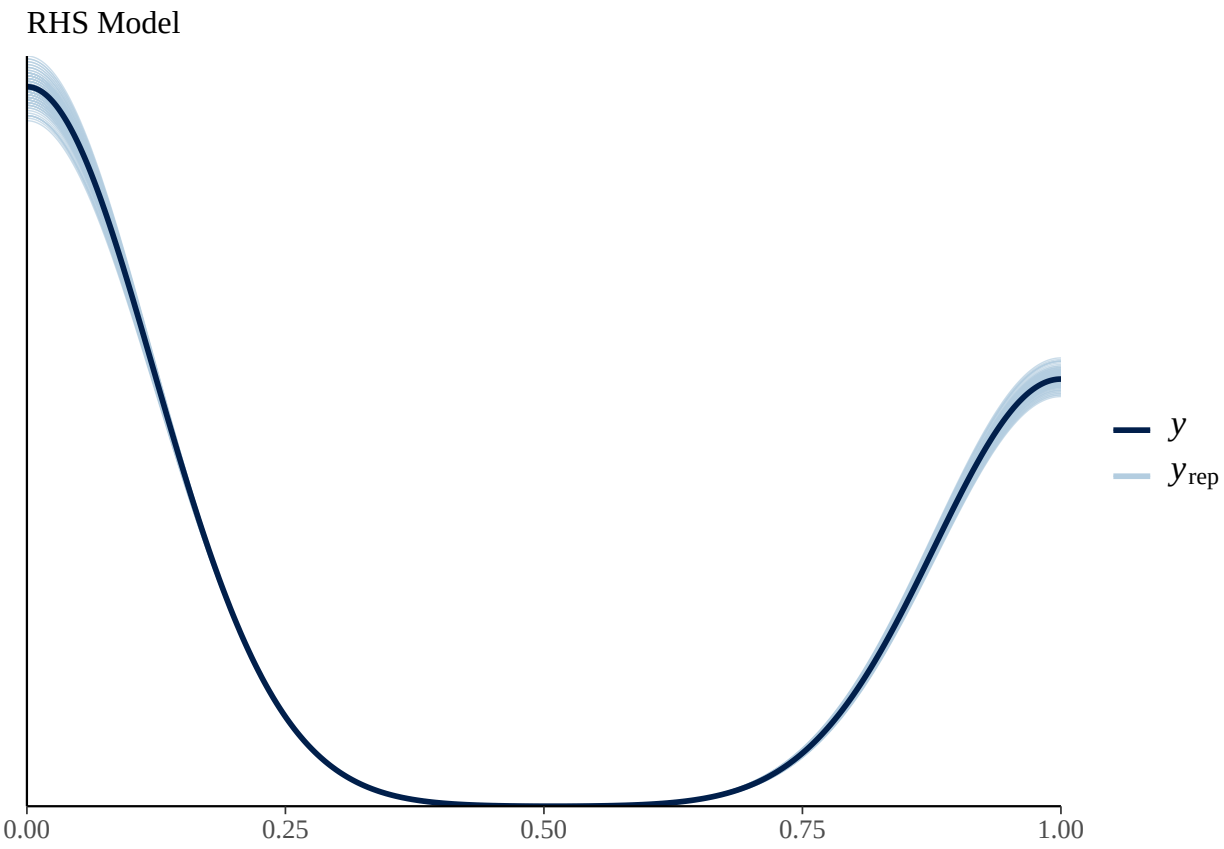

```
bayesplot_grid(ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[4]]),
               subtitles = c("Full Model"))
```




```
bayesplot_grid(ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[5]]),
               subtitles = c("Three Variables Model"))
```




```
bayesplot_grid(ppc_dens_overlay(scaled_cancer_data$diagnosis, y_sims[[6]]),
               subtitles = c("RHS Model"))
```




### Gaussian Model

The density of the diagnosis in the test set is printed in black color. Overlapping, the densities for the predictions of the first 50 iterations. This graph enables visual assessment of the quality of the predictions. Preliminarly, the results seem to be satisfactory.

Another way to check if the predicted samples represent the reality is by directly plotting them (orange) and comparing with the target ones (grey). This is possible because the gaussian model is applied on values that are contained in the test set for the chosen variable. The concentration of points at 0 or 1 $y$ values should be similar for the same values of $x$. In the given cases, by looking at the graphs, there is a very similar tendency between the real values and the simulated ones, meaning that the result is good.

## Model comparisons

### Linear Models

By using **Leave-One-Out Cross Validation**, we obtain the PSIS-LOO estimates for each of the variables for the Linear Model for posterior model comparison.

Most of the models have exceptional k-values (lower than 0.7), which means that the results obtained by the models are reliable. Additionally by looking at the `elpd_loo` and `p_loo` values for each of the models, we can deduce that the best variables to be used in order for diagnosis are as follows (in order of best): `area_worst`, `area_mean`, `area_se`, `concavity_mean`.

```r
pareto_k_values <- list()
linear_loo_estimates_df <- data.frame()

for (i in (1:length(linear_models))){
  cat("- Monitoring variable: ", variables[i], "\n")
  model <- linear_models[[i]]
  model_log_lik <- extract_log_lik(model, parameter_name = "log_lik",
                                   merge_chains = FALSE)
  model_r_eff <- relative_eff(exp(model_log_lik))
  model_loo <- loo(model_log_lik, r_eff = model_r_eff)
  pareto_k_values <- append(pareto_k_values,
                            list(model_loo$diagnostics$pareto_k))

  y_pred <- as.vector(extract(linear_models[[i]], pars="y_sim"))
  preds <- colMeans(y_pred$y_sim)
  pr <- as.integer(preds >= 0.5)
  acc <- round(mean(xor(pr, as.integer(scaled_cancer_data$diagnosis==0))), 4)

  aux_df <- data.frame("variable" = variables[i],
                       "elpd_loo" = model_loo$estimates[1],
                       "p_loo" = model_loo$estimates[2],
                       "accuracy" = acc)

  linear_loo_estimates_df <- rbind(linear_loo_estimates_df, aux_df)
}
```

```
## - Monitoring variable:  texture_mean
## - Monitoring variable:  area_mean

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

## - Monitoring variable:  smoothness_mean
## - Monitoring variable:  concavity_mean
## - Monitoring variable:  symmetry_mean
## - Monitoring variable:  fractal_dimension_mean
## - Monitoring variable:  texture_se
## - Monitoring variable:  area_se

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

## - Monitoring variable:  smoothness_se
## - Monitoring variable:  concavity_se

## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta

## - Monitoring variable:  symmetry_se
## - Monitoring variable:  fractal_dimension_se
## - Monitoring variable:  texture_worst
## - Monitoring variable:  area_worst

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

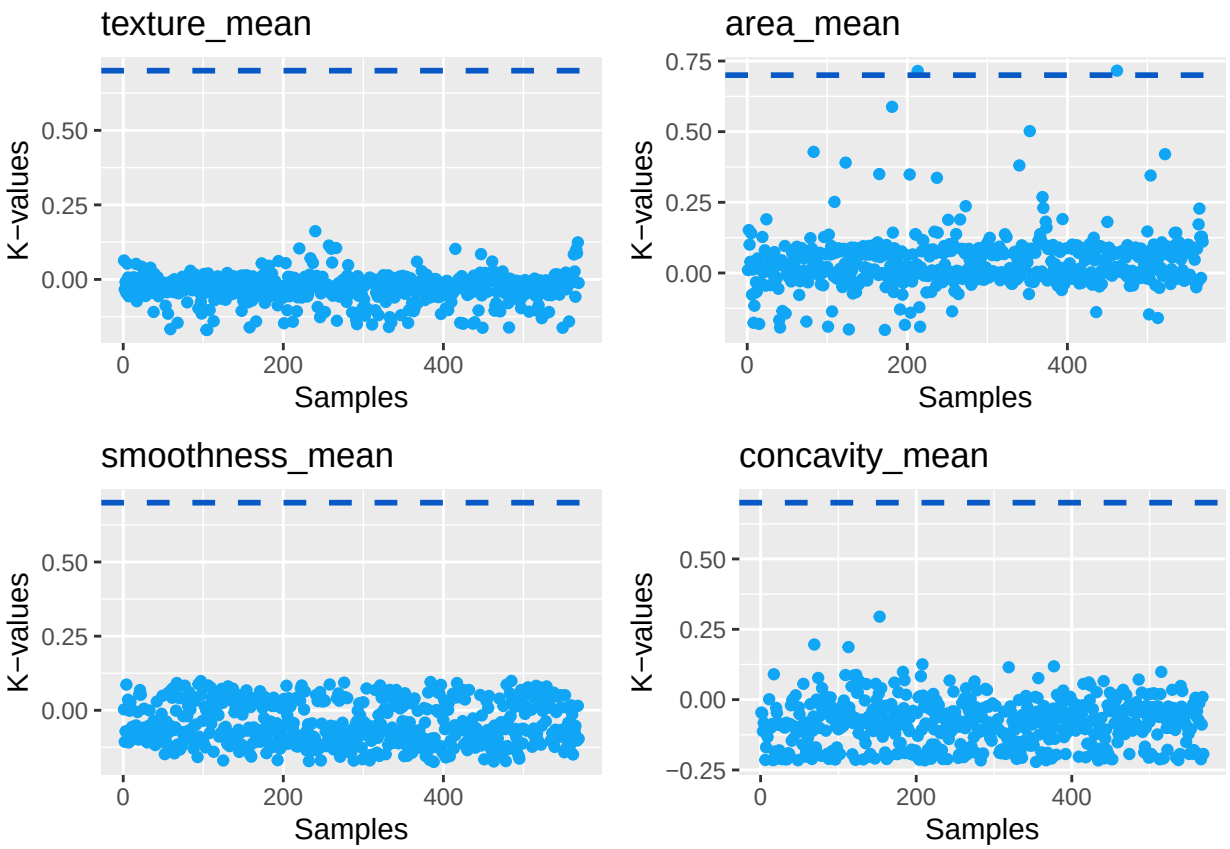
```
## - Monitoring variable:  smoothness_worst
## - Monitoring variable:  concavity_worst
## - Monitoring variable:  symmetry_worst
## - Monitoring variable:  fractal_dimension_worst
```

```
print(linear_loo_estimates_df)
```

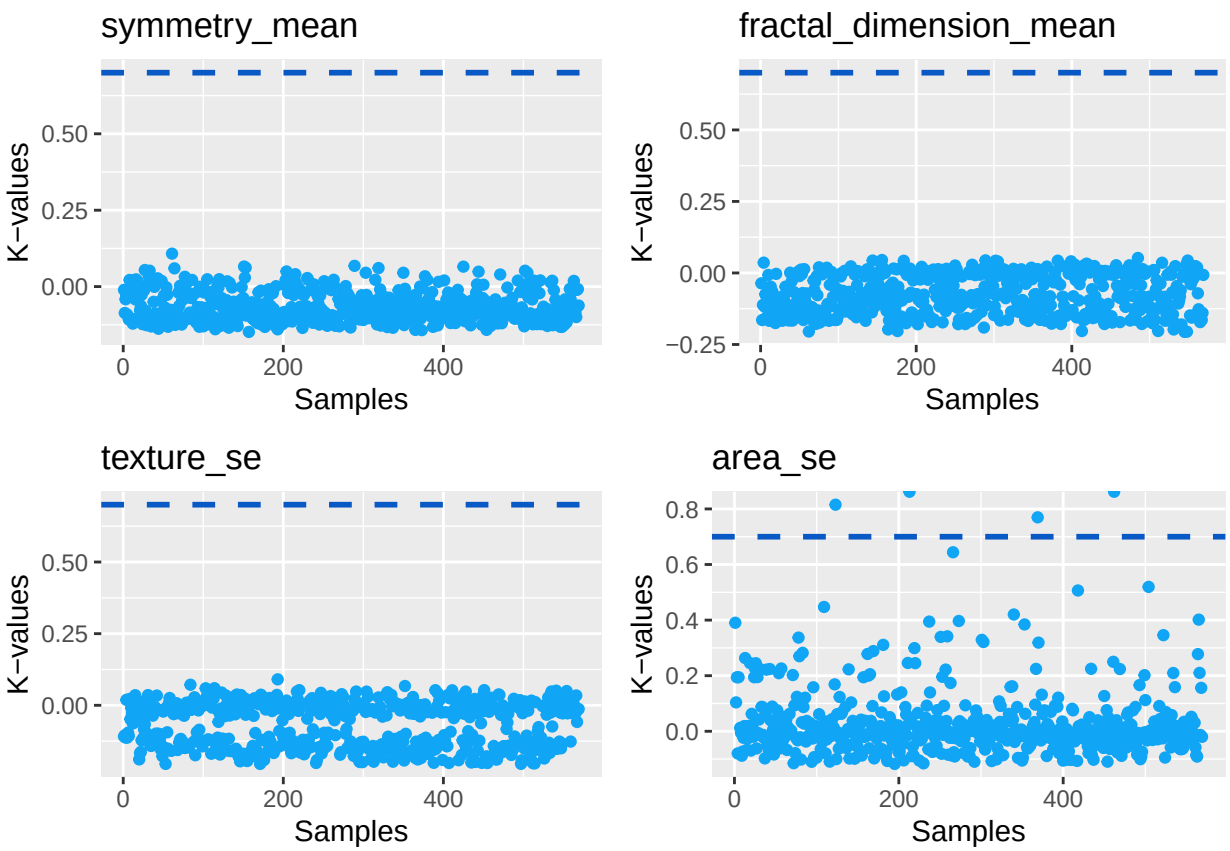
```
##                   variable  elpd_loo    p_loo accuracy
## 1             texture_mean -325.3444 2.230660   0.6977
## 2                area_mean -164.8824 2.000459   0.8822
## 3          smoothness_mean -339.0114 2.107357   0.6678
## 4           concavity_mean -194.4851 3.579987   0.8822
## 5            symmetry_mean -345.4021 2.037845   0.6837
## 6   fractal_dimension_mean -377.6696 2.027334   0.6274
## 7               texture_se -377.7236 1.975586   0.6274
## 8                  area_se -181.7876 2.025407   0.8682
## 9            smoothness_se -376.4835 2.183166   0.6274
## 10            concavity_se -357.6228 5.913835   0.6555
## 11             symmetry_se -377.8206 2.220651   0.6274
## 12    fractal_dimension_se -376.3487 2.481702   0.6186
## 13           texture_worst -313.1982 2.233350   0.7223
## 14              area_worst -117.2949 1.886005   0.9104
## 15        smoothness_worst -322.7422 2.060363   0.7153
## 16         concavity_worst -221.6281 3.419314   0.8471
## 17          symmetry_worst -326.3961 1.568806   0.7118
## 18 fractal_dimension_worst -348.6119 1.795104   0.6907
```

The following output visualizes the Pareto-k values for each linear model.

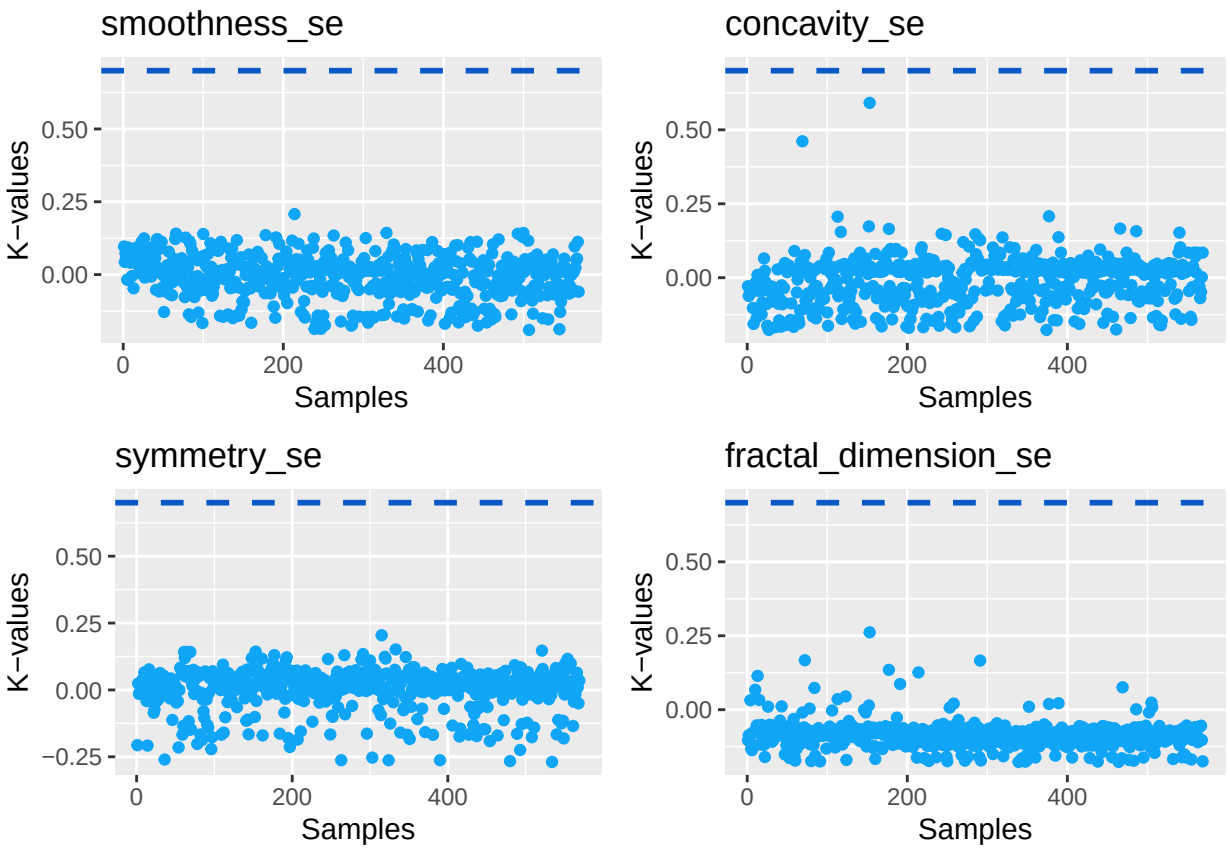
```
p <- list()
for (i in 1:length(pareto_k_values)){
  df <- data.frame(pareto_k_values[[i]])
  colnames(df) <- c("values")
  p[[i]] <- ggplot(data=df, aes(y=values, x=1:nrow(cancer_data))) +
    geom_point(color="#10A5F5") +
    geom_hline(yintercept=0.7, linetype="dashed", size=1, color="#0859C6") +
    ggtitle(variables[i]) +  xlab("Samples") + ylab("K-values")
}
do.call(grid.arrange, c(p[1:4], ncol=2))
```



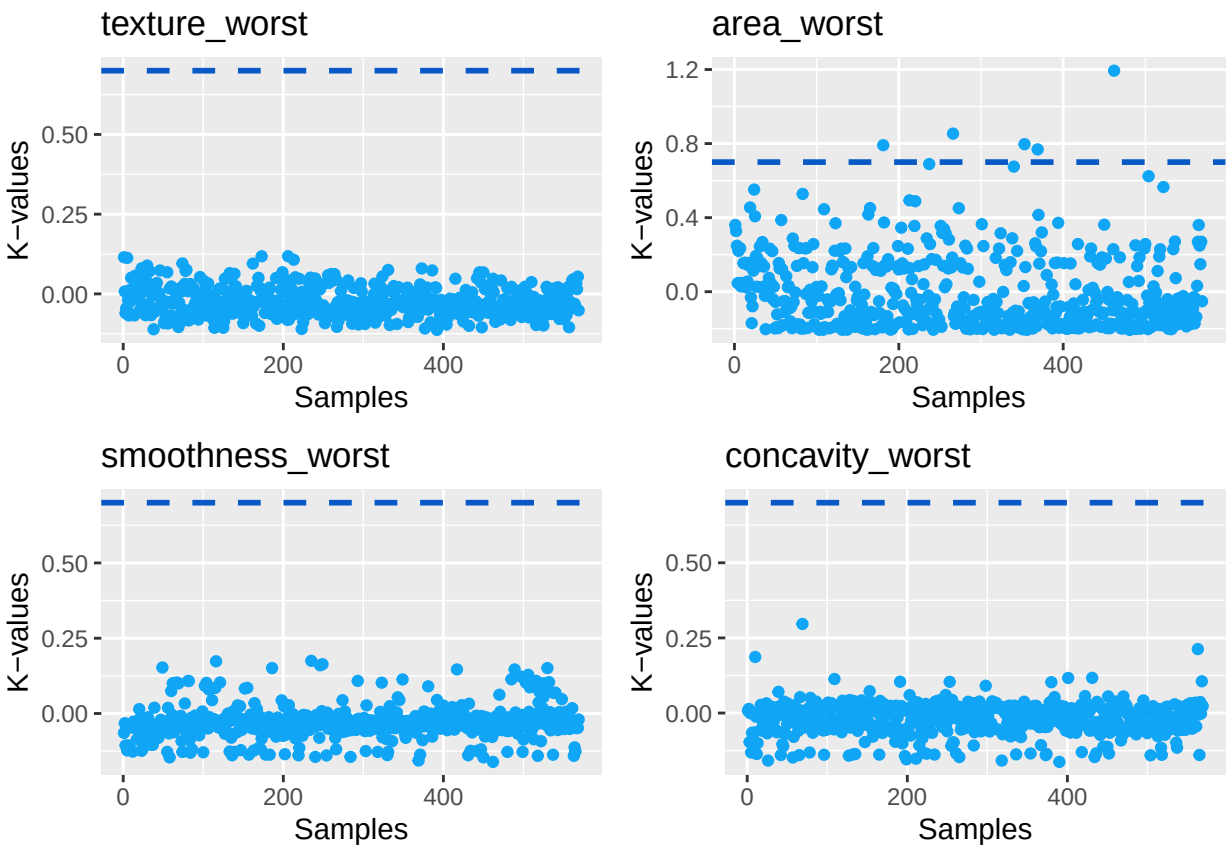

```
do.call(grid.arrange, c(p[5:8], ncol=2))
```



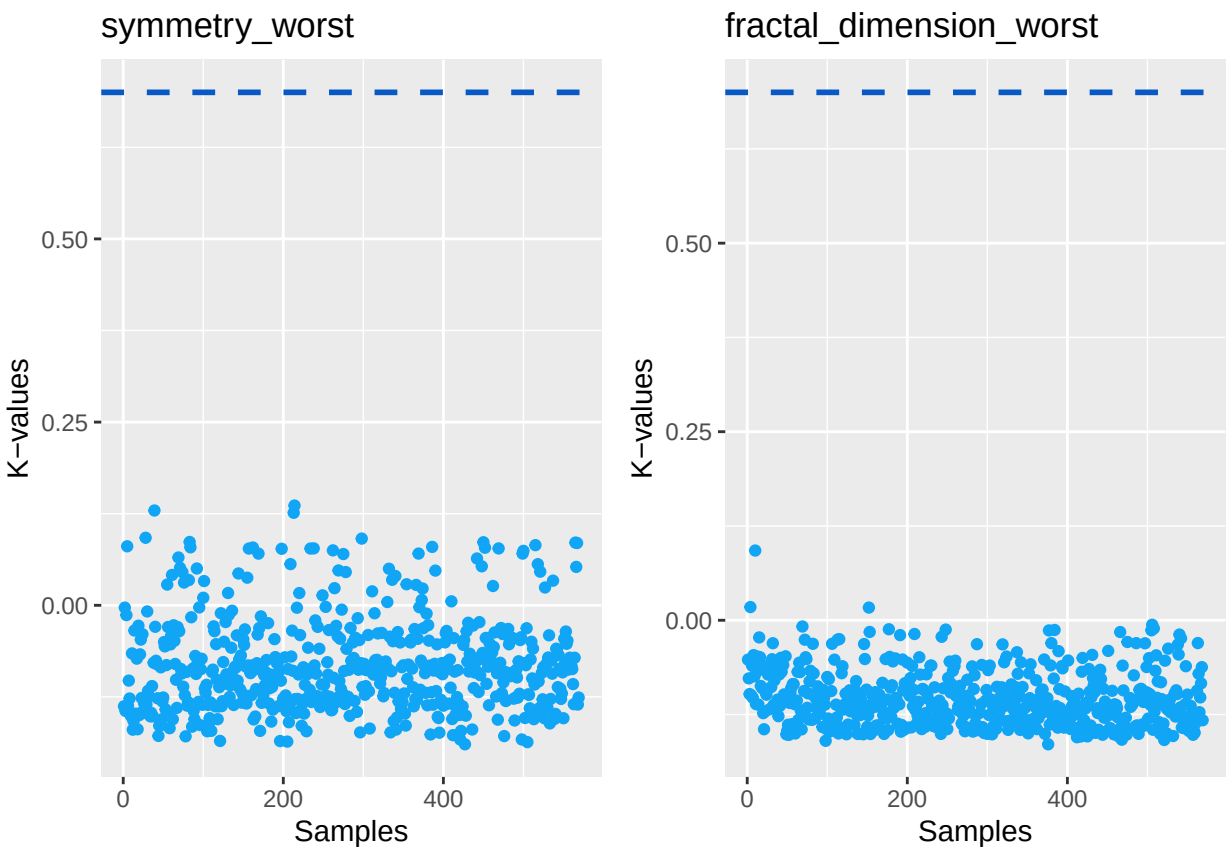

```
do.call(grid.arrange, c(p[9:12], ncol=2))
```




```
do.call(grid.arrange, c(p[13:16], ncol=2))
```




```
do.call(grid.arrange, c(p[17:18], ncol=2))
```




## Multivariate Models

By using **Leave-One-Out Cross Validation**, we obtain the PSIS-LOO estimates for each of the variable blocks for the Multivariate Model for posterior model comparison.

All of the models have relatively exceptional k-values (lower than 0.7), with a few high exceptions (equal or higher than 0.7), which means that the results obtained by the models are reliable. Additionally by looking at the `elpd_loo` and `p_loo` values for each of the models, we can deduce that the best variables to be used in order for diagnosis are as follows (in order of best): `worst block`, `mean block` and `se block`.

What we take out of this analysis by variable blocks is that, in general, the `se` variables might not be as influential for predictions as the other blocks. Although this analysis did not bring a more specific result, it was an interesting experiment to carry out in order to check if any of the variable blocks might influence too badly the final model.

Finally, it **must** be mentioned that the difference in prediction accuracy of the `se block` and the `mean block` are ... .

EXPLICAR ESTA FUNCTION AUXILIAR Y PONERLA EN OTRO SITIO

```
# Auxiliary function to calculate the accuracy of a model
model_accuracies <- function(model){
  y_pred <- posterior_epred(model)
  preds <- colMeans(y_pred)
  pr <- as.integer(preds >= 0.5)
  acc <- round(mean(xor(pr, as.integer(scaled_cancer_data$diagnosis==0))), 4)
```

```
  conf_matrix <- confusionMatrix(factor(round(pr)), factor(scaled_cancer_data$diagnosis))
  return (list(acc, list(conf_matrix)))
}
```

```
pareto_k_values <- list()
multivariate_loo_estimates_df <- data.frame()

for (i in (1:length(multivariate_glms))){
  model <- multivariate_glms[[i]]

  model_log_lik <- log_lik(model)
  model_loo <- loo(model_log_lik)
  pareto_k_values <- append(pareto_k_values, list(model_loo$diagnostics$pareto_k))
  cat("\n-- Monitoring Model: ", mv_model_names[i], "-- \n")
  print(model_loo)

  accs <- model_accuracies(model = multivariate_glms[[i]])

  aux_df <- data.frame("variable group" = mv_model_names[i],
                       "elpd_loo" = model_loo$estimates[1],
                       "p_loo" = model_loo$estimates[2],
                       "accuracy" = accs[[1]])

  multivariate_loo_estimates_df <- rbind(multivariate_loo_estimates_df, aux_df)
}
```

```
##
## -- Monitoring Model:  Mean Model --
##
## Computed from 4000 by 569 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -83.8 10.1
## p_loo         5.7  0.9
## looic       167.7 20.1
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                            Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     558   98.1%   1276
##  (0.5, 0.7]   (ok)         6    1.1%   4000
##    (0.7, 1]   (bad)        3    0.5%   4000
##    (1, Inf)   (very bad)   2    0.4%   4000
## See help('pareto-k-diagnostic') for details.
##
## -- Monitoring Model:  SE Model --
##
## Computed from 4000 by 569 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -173.0 14.4
## p_loo        12.2  3.5
```

```
## looic       346.1 28.8
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     553   97.2%   1509
##  (0.5, 0.7]   (ok)        11    1.9%   286
##    (0.7, 1]   (bad)        3    0.5%   73
##    (1, Inf)   (very bad)   2    0.4%   4000
## See help('pareto-k-diagnostic') for details.
##
## -- Monitoring Model:  Worst Model --
##
## Computed from 4000 by 569 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -57.2 10.0
## p_loo         2.8  0.7
## looic       114.3 20.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     550   96.7%   2401
##  (0.5, 0.7]   (ok)         7    1.2%   4000
##    (0.7, 1]   (bad)       10    1.8%   4000
##    (1, Inf)   (very bad)   2    0.4%   4000
## See help('pareto-k-diagnostic') for details.
##
## -- Monitoring Model:  Full Variables Model --
##
## Computed from 10000 by 569 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -95.8  9.5
## p_loo         5.8  1.0
## looic       191.5 19.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     553   97.2%   1715
##  (0.5, 0.7]   (ok)         8    1.4%   9984
##    (0.7, 1]   (bad)        2    0.4%   10000
##    (1, Inf)   (very bad)   6    1.1%   5579
## See help('pareto-k-diagnostic') for details.
##
## -- Monitoring Model:  Three variables Model --
##
## Computed from 4000 by 569 log-likelihood matrix
##
```

```
##          Estimate   SE
## elpd_loo    -63.9  9.3
## p_loo         3.3  0.6
## looic       127.8 18.6
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     526   92.4%   2895
##  (0.5, 0.7]   (ok)        19    3.3%   4000
##    (0.7, 1]   (bad)       10    1.8%   4000
##    (1, Inf)   (very bad)  14    2.5%   4000
## See help('pareto-k-diagnostic') for details.
##
## -- Monitoring Model:  RHS Model --
##
## Computed from 4000 by 569 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo    -48.7 10.1
## p_loo        14.4  3.2
## looic        97.4 20.2
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     416   73.1%   971
##  (0.5, 0.7]   (ok)        67   11.8%   260
##    (0.7, 1]   (bad)       44    7.7%   519
##    (1, Inf)   (very bad)  42    7.4%   22
## See help('pareto-k-diagnostic') for details.
```

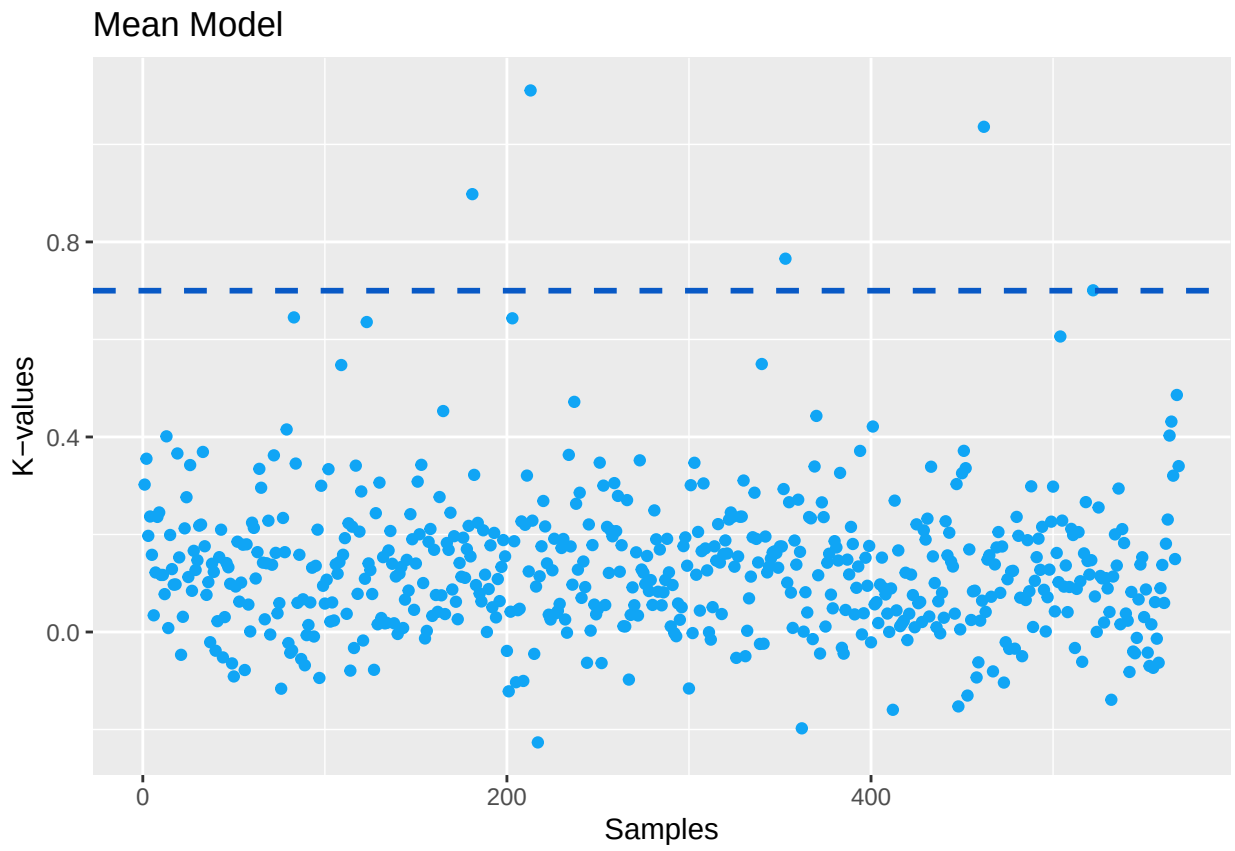
```
print(multivariate_loo_estimates_df)
```

```
##          variable.group   elpd_loo     p_loo accuracy
## 1            Mean Model  -83.84899  5.676937   0.9438
## 2              SE Model -173.02859 12.218856   0.8910
## 3           Worst Model  -57.17492  2.763165   0.9719
## 4  Full Variables Model  -95.77037  5.779921   0.9332
## 5 Three variables Model  -63.91804  3.344444   0.9543
## 6             RHS Model  -48.70333 14.446112   0.9859
```

```
p <- list()
for (i in 1:length(pareto_k_values)){
  df <- data.frame(pareto_k_values[[i]])
  colnames(df) <- c("values")
  p[[i]] <- ggplot(data=df, aes(y=values, x=1:nrow(cancer_data))) +
    geom_point(color="#10A5F5") +
    geom_hline(yintercept=0.7, linetype="dashed", size=1, color="#0859C6") +
    ggtitle(mv_model_names[i]) +  xlab("Samples") + ylab("K-values")
}
```

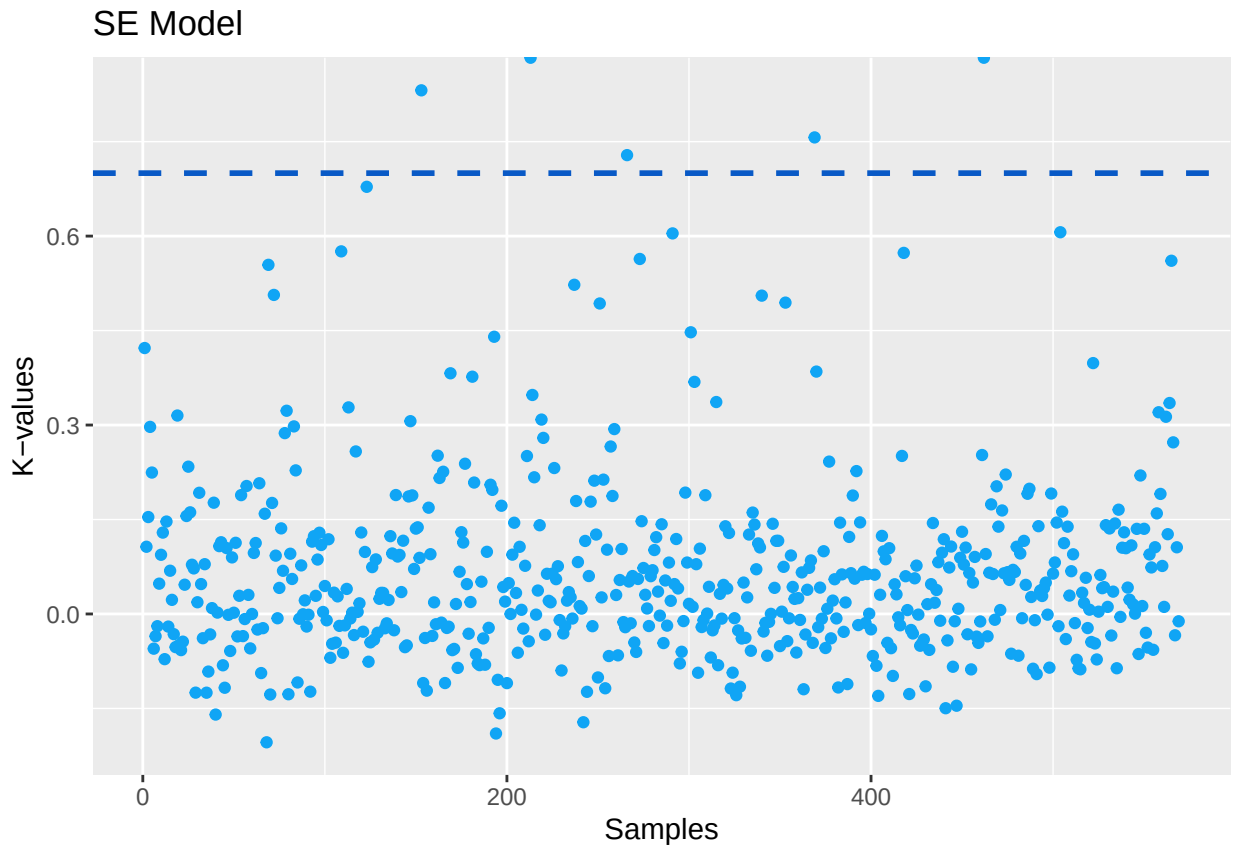
```
p[1]
```

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



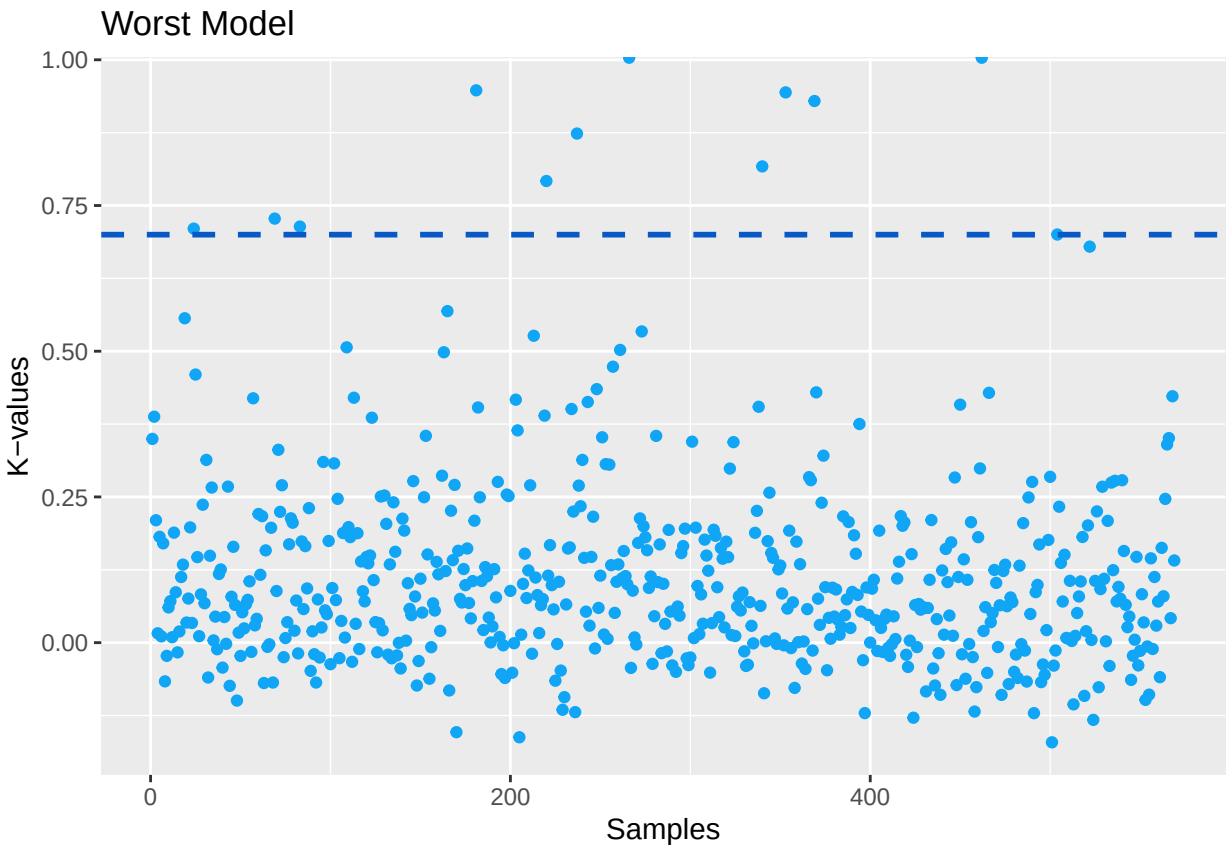

```
p[2]
```

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



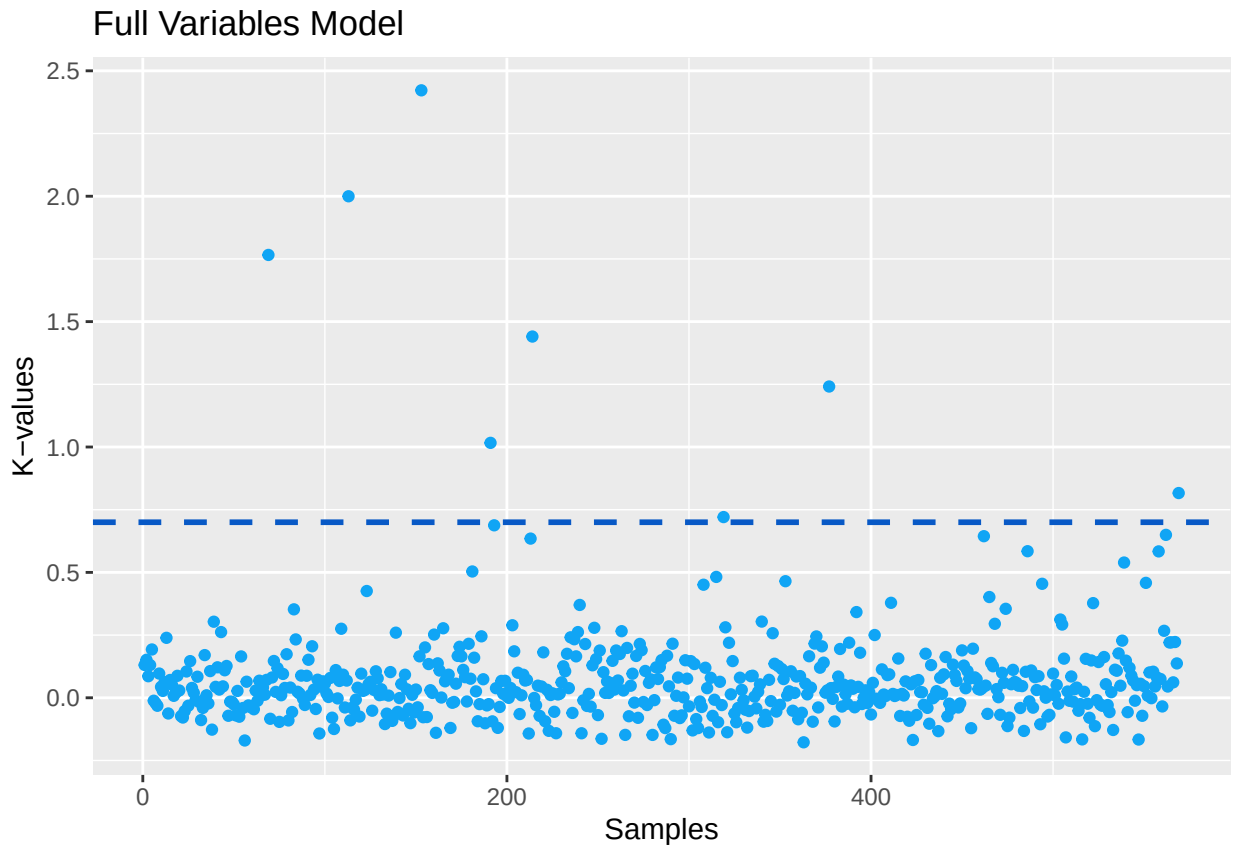

```
p[3]
```

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



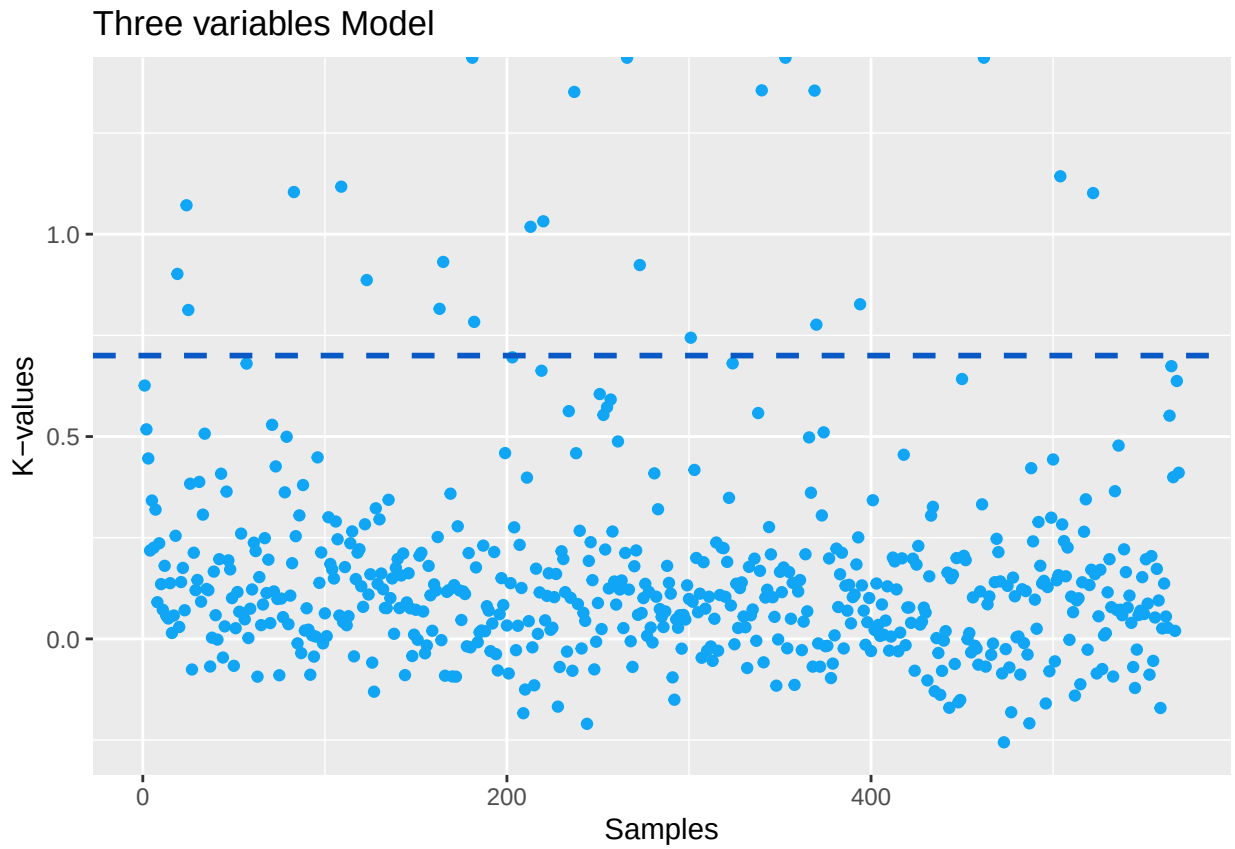

```
p[4]
```

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


Full Variables Model
2.5
2.0
1.5
1.0
0.5
0.0
K−values
0
200
400
Samples

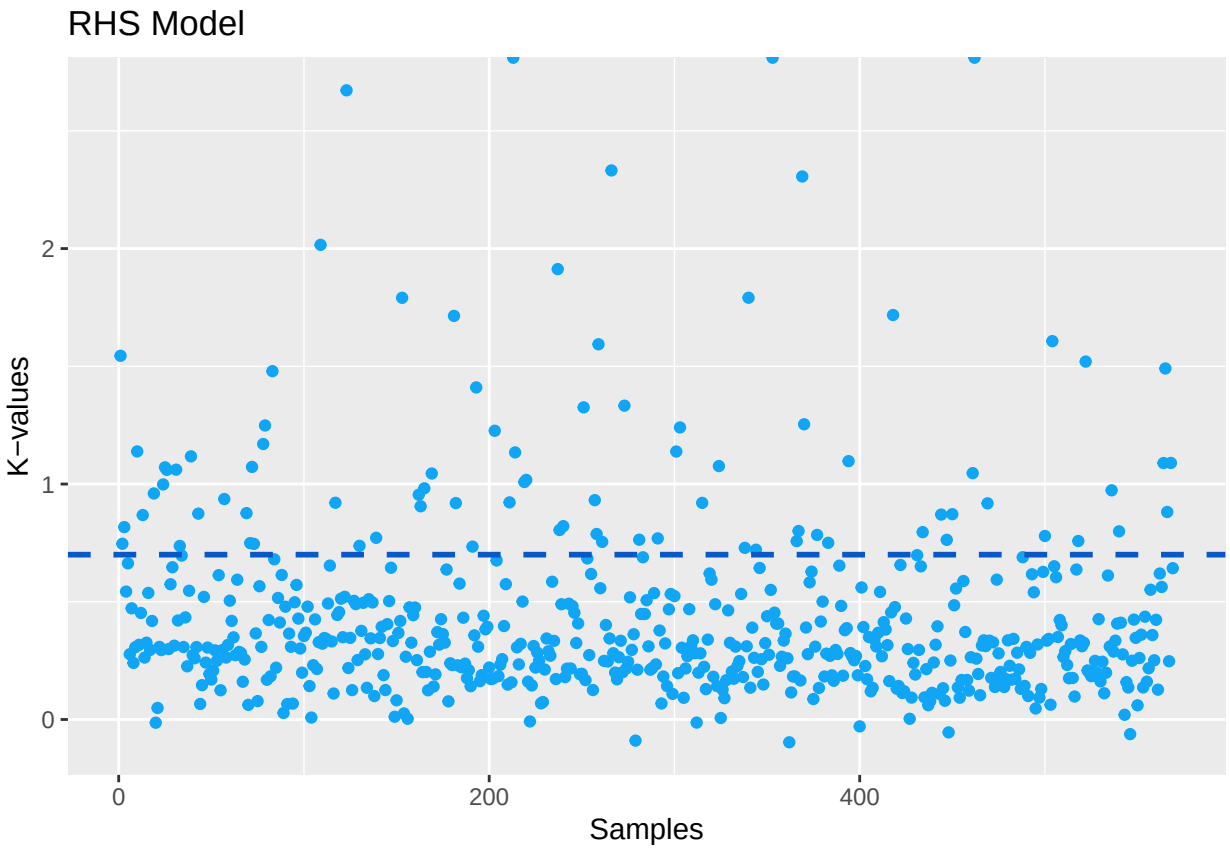
```
p[5]
```

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




```
p[6]
```

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




### Gaussian Model

When fitting the gaussian model with the four "optimal" variables, the results of the Pareto k values are, in general, good. However, for those variables for which the model yields a higher accuracy, the k values are larger, in some occasions, over 0.7. This might be representative of a model which is optimistically biased.

Finally, the auxiliary function for computing the accuracy of the model is implemented. The following function considers as accuracy the fraction of correctly labeled examples among all the predictions.

## Predictive Performance Assessment

In order to decide whether the overall diagnosis ability of our model is reliable or not, we can take a look at the accuracy. Given the nature of the diagnosis, that a negligence in this medical diagnostic could be harmful for a subject. Thus, we take into account the ratio of false positives and negatives. To accomplish this we visualize the confusion matrix produced by the function `confusionMatrix`. The output provides insightful measures for our data: sensitivity and specificity are statistical measures of the performance of a binary classification test that are widely used in medicine: Sensitivity measures the proportion of true positives that are correctly identified whereas specificity measures the proportion of true negatives.

[Decir si los valores predicción, sensitvity y specificity son buenos] [ESCRIBIR SOBRE LO DE LA PAGINA ESA DE SENSITIVITY, SPECIFICITY, ETCCCCC]

```
for (i in 1:length(multivariate_glms)){
  accs <- model_accuracies(model = multivariate_glms[[i]])
```

```
  print(mv_model_names[i])
  print(accs[[2]])
}
```

```
## [1] "Mean Model"
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 345  20
##          1  12 192
##
##                Accuracy : 0.9438
##                  95% CI : (0.9215, 0.9612)
##     No Information Rate : 0.6274
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8788
##
##  Mcnemar's Test P-Value : 0.2159
##
##             Sensitivity : 0.9664
##             Specificity : 0.9057
##          Pos Pred Value : 0.9452
##          Neg Pred Value : 0.9412
##              Prevalence : 0.6274
##          Detection Rate : 0.6063
##    Detection Prevalence : 0.6415
##       Balanced Accuracy : 0.9360
##
##        'Positive' Class : 0
##
##
## [1] "SE Model"
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 342  47
##          1  15 165
##
##                Accuracy : 0.891
##                  95% CI : (0.8625, 0.9154)
##     No Information Rate : 0.6274
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7596
##
##  Mcnemar's Test P-Value : 8.251e-05
##
##             Sensitivity : 0.9580
```

```
##             Specificity : 0.7783
##          Pos Pred Value : 0.8792
##          Neg Pred Value : 0.9167
##              Prevalence : 0.6274
##          Detection Rate : 0.6011
##    Detection Prevalence : 0.6837
##       Balanced Accuracy : 0.8681
##
##        'Positive' Class : 0
##
##
## [1] "Worst Model"
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 353  12
##          1   4 200
##
##                Accuracy : 0.9719
##                  95% CI : (0.9547, 0.9838)
##     No Information Rate : 0.6274
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9394
##
##  Mcnemar's Test P-Value : 0.08012
##
##             Sensitivity : 0.9888
##             Specificity : 0.9434
##          Pos Pred Value : 0.9671
##          Neg Pred Value : 0.9804
##              Prevalence : 0.6274
##          Detection Rate : 0.6204
##    Detection Prevalence : 0.6415
##       Balanced Accuracy : 0.9661
##
##        'Positive' Class : 0
##
##
## [1] "Full Variables Model"
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 343  24
##          1  14 188
##
##                Accuracy : 0.9332
##                  95% CI : (0.9095, 0.9523)
##     No Information Rate : 0.6274
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                  Kappa : 0.8558
##
##  Mcnemar's Test P-Value : 0.1443
##
##             Sensitivity : 0.9608
##             Specificity : 0.8868
##          Pos Pred Value : 0.9346
##          Neg Pred Value : 0.9307
##              Prevalence : 0.6274
##          Detection Rate : 0.6028
##    Detection Prevalence : 0.6450
##       Balanced Accuracy : 0.9238
##
##        'Positive' Class : 0
##
##
## [1] "Three variables Model"
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 347  16
##          1  10 196
##
##                Accuracy : 0.9543
##                  95% CI : (0.9338, 0.9699)
##     No Information Rate : 0.6274
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9017
##
##  Mcnemar's Test P-Value : 0.3268
##
##             Sensitivity : 0.9720
##             Specificity : 0.9245
##          Pos Pred Value : 0.9559
##          Neg Pred Value : 0.9515
##              Prevalence : 0.6274
##          Detection Rate : 0.6098
##    Detection Prevalence : 0.6380
##       Balanced Accuracy : 0.9483
##
##        'Positive' Class : 0
##
##
## [1] "RHS Model"
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 355   6
```

```
##          1   2 206
##
##                Accuracy : 0.9859
##                  95% CI : (0.9725, 0.9939)
##     No Information Rate : 0.6274
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9698
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.9944
##             Specificity : 0.9717
##          Pos Pred Value : 0.9834
##          Neg Pred Value : 0.9904
##              Prevalence : 0.6274
##          Detection Rate : 0.6239
##    Detection Prevalence : 0.6344
##       Balanced Accuracy : 0.9830
##
##        'Positive' Class : 0
##
```

In the case of the gaussian processes, making use of the implemented function, the obtained accuracies are computed as:

Yielding the following results:

The results for the intermediate variables are not good at all. They improve for the first and last variable. However, they are far from the good results obtained with the other implemented models.

# Conclusion

## Issues and Improvements

- hablar sobre mejorar lo de divergencias
- no poder usar stan::sampling con metodos de stan_glm sin hacerlo super lioso
- super incomodo utilizar R para trabajar con datos
- ...

## Group Self-Reflection

# References

# Appendices

## Appendix A - Linear models

### Monitor

Monitor output has been omitted for convenience, only relevant variables (Rhat, Bulk_ESS and Tail_ESS) are shown.

```r
for (i in 1:length(linear_models)){
  cat("\nMonitor for model variable:", variables[i], '\n')
  cat("-------------------------\n")
  model_mon <- monitor(linear_models[[i]], print=FALSE)
  rhat_alpha <- round(model_mon$Rhat[1], 3)
  rhat_beta <- round(model_mon$Rhat[2], 3)
  bulk_alpha <- round(model_mon$Bulk_ESS[1], 3)
  bulk_beta <- round(model_mon$Bulk_ESS[2], 3)
  tail_alpha <- round(model_mon$Tail_ESS[1], 3)
  tail_beta <- round(model_mon$Tail_ESS[2], 3)
  cat("Rhat of alpha and beta:", rhat_alpha, rhat_beta, '\n')
  cat("Bulk_ESS for alpha and beta:", bulk_alpha, bulk_beta, '\n')
  cat("Tail_ESS for alpha and beta:", tail_alpha, tail_beta, '\n')

}
```

```
##
## Monitor for model variable: texture_mean
## -------------------------
## Rhat of alpha and beta: 1.003 1.002
## Bulk_ESS for alpha and beta: 3503 3083
## Tail_ESS for alpha and beta: 2400 2521
##
## Monitor for model variable: area_mean
## -------------------------
## Rhat of alpha and beta: 1.002 1.003
## Bulk_ESS for alpha and beta: 2681 2682
## Tail_ESS for alpha and beta: 2430 2060
##
## Monitor for model variable: smoothness_mean
## -------------------------
## Rhat of alpha and beta: 1.001 1.001
## Bulk_ESS for alpha and beta: 3359 2938
## Tail_ESS for alpha and beta: 2836 2493
##
## Monitor for model variable: concavity_mean
## -------------------------
## Rhat of alpha and beta: 1.003 1.001
## Bulk_ESS for alpha and beta: 2957 3170
## Tail_ESS for alpha and beta: 2203 2304
##
## Monitor for model variable: symmetry_mean
## -------------------------
## Rhat of alpha and beta: 1.001 1.001
## Bulk_ESS for alpha and beta: 3588 2950
## Tail_ESS for alpha and beta: 2654 2400
##
## Monitor for model variable: fractal_dimension_mean
## -------------------------
## Rhat of alpha and beta: 1.001 1
## Bulk_ESS for alpha and beta: 3117 3501
## Tail_ESS for alpha and beta: 2287 2643
##
```

```
## Monitor for model variable: texture_se
## -------------------------
## Rhat of alpha and beta: 1.002 1.001
## Bulk_ESS for alpha and beta: 3574 3527
## Tail_ESS for alpha and beta: 2635 2654
##
## Monitor for model variable: area_se
## -------------------------
## Rhat of alpha and beta: 1.001 1.002
## Bulk_ESS for alpha and beta: 1202 1301
## Tail_ESS for alpha and beta: 1974 1731
##
## Monitor for model variable: smoothness_se
## -------------------------
## Rhat of alpha and beta: 1.002 1.001
## Bulk_ESS for alpha and beta: 3516 3329
## Tail_ESS for alpha and beta: 2540 2635
##
## Monitor for model variable: concavity_se
## -------------------------
## Rhat of alpha and beta: 1 1
## Bulk_ESS for alpha and beta: 3871 3680
## Tail_ESS for alpha and beta: 2650 2813
##
## Monitor for model variable: symmetry_se
## -------------------------
## Rhat of alpha and beta: 1.001 1.002
## Bulk_ESS for alpha and beta: 2923 3384
## Tail_ESS for alpha and beta: 2501 2815
##
## Monitor for model variable: fractal_dimension_se
## -------------------------
## Rhat of alpha and beta: 1 1.004
## Bulk_ESS for alpha and beta: 3752 3463
## Tail_ESS for alpha and beta: 2639 2800
##
## Monitor for model variable: texture_worst
## -------------------------
## Rhat of alpha and beta: 1 1
## Bulk_ESS for alpha and beta: 3026 3278
## Tail_ESS for alpha and beta: 2808 2509
##
## Monitor for model variable: area_worst
## -------------------------
## Rhat of alpha and beta: 1.001 1.001
## Bulk_ESS for alpha and beta: 1785 1653
## Tail_ESS for alpha and beta: 2216 2062
##
## Monitor for model variable: smoothness_worst
## -------------------------
## Rhat of alpha and beta: 1 1.001
## Bulk_ESS for alpha and beta: 2750 2222
## Tail_ESS for alpha and beta: 2436 2329
##
```

```
## Monitor for model variable: concavity_worst
## -------------------------
## Rhat of alpha and beta: 1.001 1.002
## Bulk_ESS for alpha and beta: 3461 3318
## Tail_ESS for alpha and beta: 2761 2470
##
## Monitor for model variable: symmetry_worst
## -------------------------
## Rhat of alpha and beta: 1.007 1.006
## Bulk_ESS for alpha and beta: 491 497
## Tail_ESS for alpha and beta: 627 621
##
## Monitor for model variable: fractal_dimension_worst
## -------------------------
## Rhat of alpha and beta: 1.014 1.015
## Bulk_ESS for alpha and beta: 420 417
## Tail_ESS for alpha and beta: 617 574
```

**Divergences**

```
for (i in (1:length(linear_models))){
  cat("\n\n- Checking divergence for variable: ", variables[i], "\n")
  check_hmc_diagnostics(linear_models[[i]])
}
```

```
##
##
## - Checking divergence for variable:  texture_mean
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  area_mean
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.
```

```
##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  smoothness_mean
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  concavity_mean
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  symmetry_mean
##
## Divergences:
```

```
## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  fractal_dimension_mean
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  texture_se
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

```
##
##
## - Checking divergence for variable:  area_se
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  smoothness_se
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  concavity_se
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:
```

```
## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  symmetry_se
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  fractal_dimension_se
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  texture_worst
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.
```

```
##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  area_worst
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  smoothness_worst
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  concavity_worst
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:
```

```
## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  symmetry_worst
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
##
## - Checking divergence for variable:  fractal_dimension_worst
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 10.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

## Appendix B - Multivariate models

### Monitor

Monitor output has been omitted for convenience, only relevant variables (Rhat, Bulk_ESS and Tail_ESS) are shown.

```r
for (i in 1:length(multivariate_glms)){
  cat("\nMonitor for multivariate model:", mv_model_names[i], '\n')
  cat("-------------------------\n")
  model_mon <- monitor(multivariate_glms[[i]]$stanfit, print=FALSE)
  rhat_alpha <- round(model_mon$Rhat[1], 3)
  rhat_beta <- round(model_mon$Rhat[2], 3)
  bulk_alpha <- round(model_mon$Bulk_ESS[1], 3)
  bulk_beta <- round(model_mon$Bulk_ESS[2], 3)
  tail_alpha <- round(model_mon$Tail_ESS[1], 3)
  tail_beta <- round(model_mon$Tail_ESS[2], 3)
  cat("Rhat of alpha and beta:", rhat_alpha, rhat_beta, '\n')
  cat("Bulk_ESS for alpha and beta:", bulk_alpha, bulk_beta, '\n')
  cat("Tail_ESS for alpha and beta:", tail_alpha, tail_beta, '\n')

}
```

```
##
## Monitor for multivariate model: Mean Model
## -------------------------
## Rhat of alpha and beta: 1.001 1.001
## Bulk_ESS for alpha and beta: 3640 2834
## Tail_ESS for alpha and beta: 2867 2928
##
## Monitor for multivariate model: SE Model
## -------------------------
## Rhat of alpha and beta: 1.001 1.001
## Bulk_ESS for alpha and beta: 3206 4145
## Tail_ESS for alpha and beta: 2872 3010
##
## Monitor for multivariate model: Worst Model
## -------------------------
## Rhat of alpha and beta: 1.002 1.002
## Bulk_ESS for alpha and beta: 3360 3169
## Tail_ESS for alpha and beta: 2993 3232
##
## Monitor for multivariate model: Full Variables Model
## -------------------------
## Rhat of alpha and beta: 1.002 1.017
## Bulk_ESS for alpha and beta: 9223 289
## Tail_ESS for alpha and beta: 6923 428
##
## Monitor for multivariate model: Three variables Model
## -------------------------
## Rhat of alpha and beta: 1 1.002
## Bulk_ESS for alpha and beta: 2131 1517
## Tail_ESS for alpha and beta: 2517 2163
##
## Monitor for multivariate model: RHS Model
## -------------------------
## Rhat of alpha and beta: 1 1
## Bulk_ESS for alpha and beta: 2768 3833
## Tail_ESS for alpha and beta: 3023 3732
```

### Divergences

```
for (i in 1:length(multivariate_glms)){
 cat("Checking divergences for multivariate model:", mv_model_names[i], '\n')
 cat("-------------------------\n")
 check_hmc_diagnostics(multivariate_glms[[i]]$stanfit)
}
```

```
## Checking divergences for multivariate model: Mean Model
## -------------------------
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 15.

##
## Energy:

## E-BFMI indicated no pathological behavior.

## Checking divergences for multivariate model: SE Model
## -------------------------
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 15.

##
## Energy:

## E-BFMI indicated no pathological behavior.

## Checking divergences for multivariate model: Worst Model
## -------------------------
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.
```

```
##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 15.

##
## Energy:

## E-BFMI indicated no pathological behavior.

## Checking divergences for multivariate model: Full Variables Model
## -------------------------
##
## Divergences:

## 11 of 10000 iterations ended with a divergence (0.11%).
## Try increasing 'adapt_delta' to remove the divergences.

##
## Tree depth:

## 0 of 10000 iterations saturated the maximum tree depth of 15.

##
## Energy:

## E-BFMI indicated no pathological behavior.

## Checking divergences for multivariate model: Three variables Model
## -------------------------
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 15.

##
## Energy:

## E-BFMI indicated no pathological behavior.

## Checking divergences for multivariate model: RHS Model
## -------------------------
##
## Divergences:
```

```
## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 15.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

## Appendix C - Gaussian model

### Monitor

Monitor output has been omitted for convenience, only relevant variables (Rhat, Bulk_ESS and Tail_ESS) are shown.