

# Credit Risk Analysis

Hung Nguyen (1022029) - Hiep Nguyen (1022799)

November, 2022

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Data description</b>	<b>2</b>
2.1	Data preprocessing and cleaning . . . . .	3
2.2	Explanatory variables analysis . . . . .	4
<b>3</b>	<b>Models description</b>	<b>5</b>
3.1	Pooled Logistic Regression model . . . . .	5
3.2	Hierarchical Logistic Regression model . . . . .	6
<b>4</b>	<b>Results analysis</b>	<b>8</b>
4.1	Covergence diagnostics . . . . .	8
4.2	Posterior predictive checks . . . . .	11
4.3	Model comparison . . . . .	11
4.4	Predictive performance assessment . . . . .	13
4.5	Prior sensitivity analysis . . . . .	14
<b>5</b>	<b>Conclusion and potential improvements</b>	<b>15</b>
5.1	Conclusion . . . . .	15
5.2	Issues and potential improvements . . . . .	15
<b>6</b>	<b>Self-reflection</b>	<b>16</b>
<b>7</b>	<b>Appendices and references</b>	<b>16</b>
7.1	Stan code appendices . . . . .	16
7.2	References . . . . .	19

# 1 Introduction

This project is part of the Aalto University Bayesian Data Analysis 2022 Course.

One of the most important services of banks that attract their customers is providing credit. However, when lenders offer home loans, auto loans, or business loans, there is an inherent risk that borrowers will default on their payments, and this is termed credit risk. Credit risk is universally known as the possibility of a loss for a lender due to a borrower's failure to repay a loan. When the credit risk is mishandled by lenders, the consequences can be catastrophic. The collapse of the housing market in 2008 and the ensuing recession were one of the best illustrations of how severe the outcome can be: in just a few months, the banking sector nearly collapsed due to a significant overexposure to credit defaults. Therefore, it is essential to determine the borrowers' ability to meet debt obligations as well as the risks involved, and this process is known as credit risk analysis.

The reason this topic is chosen is because of its meaningfulness and significance to the credit market, and the fact that there has been limited Bayesian data analysis project performed on the topic. Our goal is to use Logistic Regression to identify the riskiness of a loan by classifying them into good and bad loans. A pooled and hierarchical model will be applied to this problem, and through results analysis, we can compare and choose a stronger-performed model for this problem.

The report consists of the following parts: Introduction, Data Description, Models, Results, Discussion, Conclusion, and Appendices. First, we formulate the problem and show how we handle the data through pre-processing and feature selection processes. In the Models section, we describe the two Stan models used and justify their likelihood and justification of their choice. In the Results section, we analyze the performance of the models through convergence analysis, posterior predictive checks, predictive performance assessment, model selection, and prior sensitivity analysis. Finally, in the Discussion and Conclusion section, we will address the issues and potential improvements for our models, as well as provide some interesting insights that we have learned while doing the project. The Appendices part will include all the Stan models being used in this report. The Stan models, as well as complete R code for this project, can be found [here](#).

## 2 Data description

The data set is obtained through Kaggle. It contains 1000 data points with 20 categorical attributes created by Professor Hofmann. In this data set, each entry represents a person taking credit from a bank, and each person is classified as having good or bad credit risk according to the set of attributes. The full data set can be found [here](#).

The data consists of 10 columns, 9 explanatory variables, and 1 target variable, defined as follows:

Explanatory variables:

- Age (numerical/continuous): The age of the subject.
- Duration (numerical/continuous): The contracted loan due time (in months).
- Credit.amount (numerical/continuous): The amount of credit of the subject's loan.
- Sex (textual/categorical): The gender of the subject. This includes **"male"** and **"female"**.
- Job (numerical/categorical): The level of employment of the subject. This includes **"0"** - unskilled and non-resident, **"1"** - unskilled and resident, **"2"** - skilled, **"3"** - highly skilled.
- Housing (textual/categorical): The type of housing of the subject. This includes **"own"**, **"rent"** and **"free"**.
- Saving.accounts (textual/categorical): The level of wealth of subject's saving account. This includes **"little"**, **"moderate"**, **"quite rich"**, **"rich"**.
- Checking.account (textual/categorical): The level of wealth of subject's checking account. This includes **"little"**, **"moderate"**, **"quite rich"**, **"rich"**.
- Purpose (textual/categorical): The purpose of the subject's loan. This includes **"business"**, **"car"**, **"domestic appliances"**, **"education"**, **"furniture/equipemt"**, **"radio/TV"**, **"repairs"**, **"vacation/others"**.

Target variable:

- Risk (textual/categorical): The fact that the subject has defaulted (not repaying the loan). This includes “good” - good loan (the subject has repaid on time), “bad” - bad loan (the subject has defaulted).

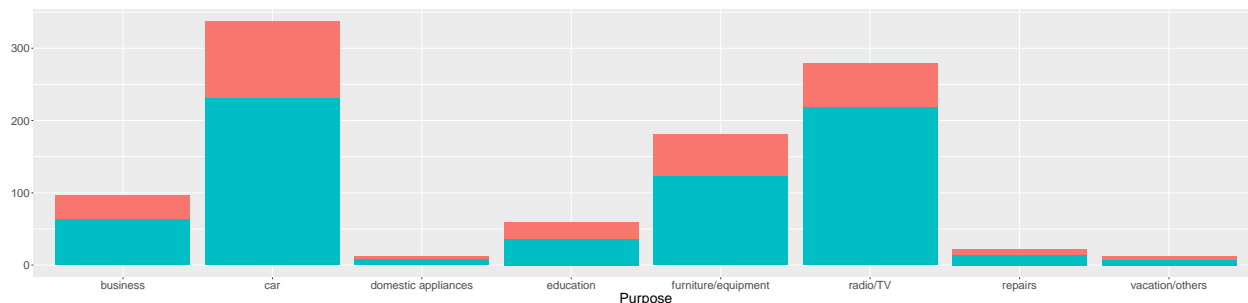
The data set have missing data: the columns Saving.accounts and Checking account both contain NA values (missing variables), which takes 18.3% and 39.4% of the total observations, respectively.

Age	Sex	Job	Housing
0.0	0.0	0.0	0.0
Saving.accounts	Checking.account	Credit.amount	Duration
18.3	39.4	0.0	0.0
Purpose	Risk		
0.0	0.0		

## 2.1 Data preprocessing and cleaning

We can explore the data by plotting histograms for the features corresponding with their target values. The histograms can provide insights regarding the correlation between the explanatory variables (features) with the target variables Risk.





There are several issues that can be observed from the histograms:

- The variables Saving.accounts and Checking.account have NA values (which is also suggested in the Data set description section above).
- The continuous variables Credit.amount and Duration have single-point outliers, as their histograms both show single significantly larger values clearly separating themselves from the rest of the population.
- The discrete variables Saving.accounts and Purpose have a high scatter level. That is, their histograms show that the data was categorized into too many groups while some groups are relatively low in population.

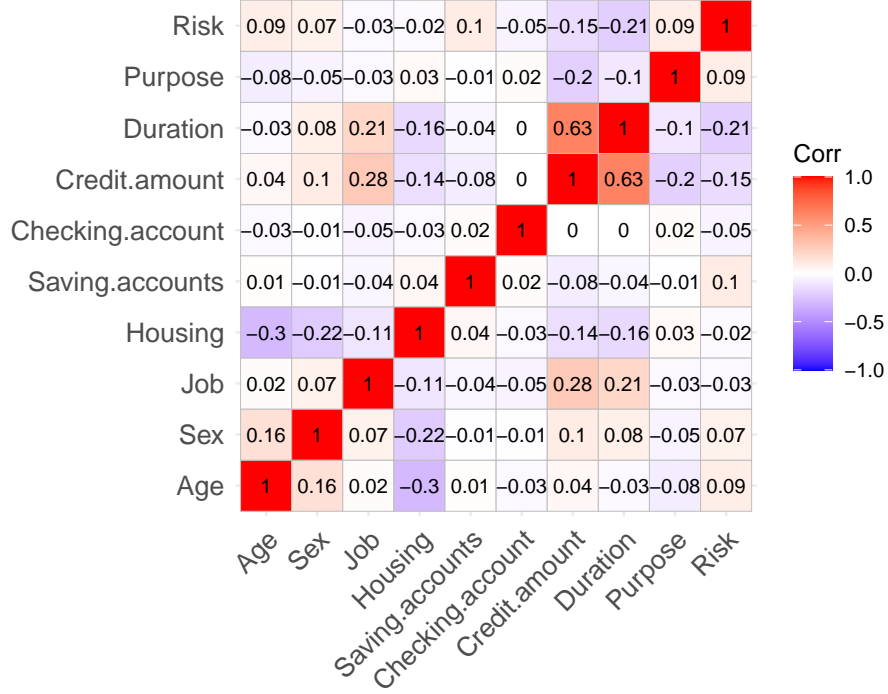
We have provided the following solutions to these issues:

- In order to resolve the high NA values presented in two of the features, we decided to impute the missing data with the mode (most frequent value) of data, which, from the plots, is the “little” value. Mode imputation is a popular method for filling in missing data of categorical data, especially when the data is unbalanced and highly skewed towards the most frequent value like Saving.accounts and Checking.accounts.
- To resolve the outliers presented in the Credit.amount and Duration, because they are very little in population, we can just remove them from the data. For Credit.amount, we remove the maximum data point; for Duration, the outlier is also just a single data point so we just delete the maximum of the column from the data.
- To resolve the high sparse level presented in Saving.accounts and Purpose, a possible solution could be merging the less populated groups with the relevant counterparts. With Saving.accounts, we merge the “rich” and “quite rich” values into “rich”; with Purpose, we merge “domestic appliances” with “furniture/equipment” into “furniture/equipment”, “repairs” with “vacation/others” into “others”. This fix will help to data to be less scattered and make the models constructed later to predict more robust results.

One additional but important step that could be made to our data is standardization. The input consists of different variables (categorical, numerical, textual) with distinct scales so standardizing them is vital for the models to perform better. In addition, standardizing can facilitate the process of prior choice for the features, since all variables are on the same scale. To standardize, we transform all variables (including the target) to numerical. Then, we scale the explanatory variables them so that their mean = 0 and standard deviation = 1.

## 2.2 Explanatory variables analysis

We can select the final explanatory variables for the models by using a correlation matrix to visualize the correlated level of each variable with each other.



From the correlation plot, we observe no correlation greater than 0.9 between the features so no significant multicollinearity is presented. However, it is noticeable that the Housing variable shows the lowest correlation with the target variable (only  $-0.02$ ), indicating that it is quite insignificant to the sampling process. This should be noted, as later on, some actions will be taken to diminish its effect on the results.

In conclusion, for the pooled model, the chosen explanatory variables are Age, Sex, Job, Housing, Saving.accounts, Checking.account, Duration, and Purpose, with Housing being less correlated with the target values. For the hierarchical model, the features remain the same but the variable Purpose will instead be used as a hierarchical grouping variable and will be excluded from the explanatory variables. The chosen target for both models is the Risk variable.

### 3 Models description

#### 3.1 Pooled Logistic Regression model

The Logistic Regression model is used to classify the risk  $y$  in lending a loan based on the observation  $x$ . Its predictor  $p$  is parameterized with an intercept  $\beta_0$  and explanatory coefficients  $\beta_1, \beta_2, \dots, \beta_k$  as follows:

$$p = \frac{e^{\beta_0 + \beta_1 x_{n,1} + \dots + \beta_k x_{n,k}}}{1 + e^{\beta_0 + \beta_1 x_{n,1} + \dots + \beta_k x_{n,k}}} = \text{logit}^{-1}(\beta_0 + \beta_1 x_{n,1} + \dots + \beta_k x_{n,k})$$

The target variable  $y$  for the observations  $x$  follows the distribution:

$$y_n \sim \text{Bernoulli}(p) = \text{Bernoulli}(\text{logit}^{-1}(\beta_0 + \beta_1 x_{n,1} + \dots + \beta_k x_{n,k}))$$

where  $\beta_0$  is the intercept and each  $\beta_1, \beta_2, \dots, \beta_k$  models the regression coefficient for each feature.

##### 3.1.1 Prior justification

Because of the standardized data, we may assume a commonly-used generic weakly informative prior for the explanatory coefficients:

$$\beta_0, \beta_1, \dots, \beta_k \sim \text{normal}(0, 1)$$

Normal distribution with a location parameter of 0 expresses our little knowledge of how a change in input variables can affect the predicted target, while a standard deviation of 1 widens the shape of the distribution so that it, softly concentrates below the scale, which is weakly informative enough for our standardized data.

However, as noted, for the potentially irrelevant variable Housing, which corresponds to  $\beta_3$ , we assume an informative prior:

$$\beta_3 \sim \text{normal}(0, 0.01)$$

This informative prior serves as an powerful *regularization prior* scaling the insignificant coefficient  $\beta_3$  to nearly zero, but not completely disregard them from the model, therefore shrinking its effect on the results. This effect shrinkage idea is to justify model selection, according to [Hahn et al. \(2014\)](#).

### 3.1.2 Running the model

Before running, the input data was separated into training data and testing data with equal proportions (half) of the data. While the training set is used for the sampling process of the mode, the test set is utilized for predictions and accuracy testing purposes.

```
train_size <- round(nrow(data_pooled)/2, 0)
test_size <- nrow(data_pooled) - train_size
cat("Train size:", train_size, "/ Test size:", test_size)
```

Train size: 499 / Test size: 499

The model is then run with 4 chains, 2000 iterations and 1000 warm-ups.

```
# Create train and test sets
data_train <- head(data_pooled, train_size)
data_test <- tail(data_pooled, test_size)

# Create train and test target
y_train_pooled = data_train$Risk
y_test_pooled = data_test$Risk

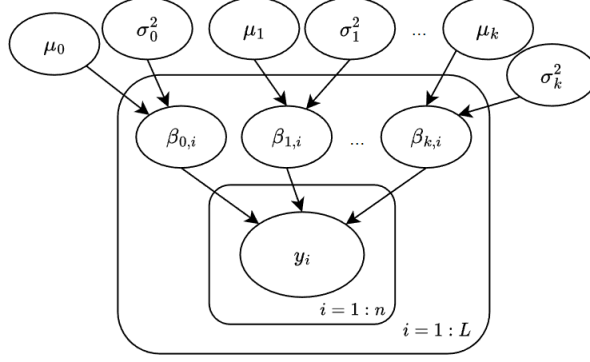
# Create train and test observations
X_train_pooled = subset(data_train, select=-c(`Risk`))
X_test_pooled = subset(data_test, select=-c(`Risk`))

# Create combined data
credit_data_pooled <- list(N_train=nrow(X_train_pooled),
                           K=ncol(X_train_pooled),
                           X_train=X_train_pooled,
                           y_train=y_train_pooled,
                           N_test=nrow(X_test_pooled),
                           X_test=X_test_pooled)

# Load and run data
pooled_fit <- stan(file="pooled.stan", data=credit_data_pooled,
                  chains=4, iter=2000, warmup=1000)
```

## 3.2 Hierarchical Logistic Regression model

The hierarchical model, though resembles the pooled model in using Logistic Regression, is a relatively more complex model. As mentioned in the Explanatory variables analysis section, the data is categorized into  $L$  groups using the categorical data in the Purpose variable. The coefficients  $\beta_0, \beta_1, \dots, \beta_k$  is then sampled separately for each of  $L$  groups using their own *hyper-parameters*  $\mu$  and  $\sigma^2$ . The structuring of the model can be given as below:



With each of  $L$  groups having their own  $\beta_0, \beta_1, \dots, \beta_k$  sampled separately, we can build a Logistic Regression equation for the target variable:

$$y_n \sim \text{Bernoulli}(\text{logit}^{-1}(\beta_{0,L(n)} + \beta_{1,L(n)}x_{n,1} + \dots + \beta_{k,L(n)}x_{n,k})),$$

with  $L(n)$  being the group of the  $n^{\text{th}}$  data

### 3.2.1 Prior justification

We can estimate the coefficients  $\beta_{0,L(n)}, \beta_{1,L(n)}, \dots, \beta_{k,L(n)}$  using a Normal prior with the hyper-parameters  $\mu$  and  $\sigma^2$ :

$$\beta_{i,L(n)} \sim \mathcal{N}(\mu_i, \sigma_i^2), \text{ with } i = 1 : k$$

For the hyper-parameter  $\mu_i$ , we assume a the same generic weakly-informative prior:

$$\mu_i \sim \mathcal{N}(0, 1)$$

Note that because of the standardization step, this prior is considered to be weakly informative enough to not create biases. We have explained the choice of this prior above in the pooled model.

For the hyper-parameter  $\sigma^2$ , we assume a weakly-informative inverse-gamma prior:

$$\sigma_i^2 \sim \text{InvGamma}(0.5, 1)$$

In hierarchical modeling, the inverse-gamma distribution is a common prior for hyper-variance estimating. The shape of the distribution  $\text{InvGamma}(0.5, 1)$  has a wide and heavy right tail, indicating that it is positive and weakly informative, which is suitable for our estimation for  $\sigma^2$ .

For the weakly significant Housing variable, similar to the previous model, we assume informative regularization priors for the hyperparameters:

$$\begin{aligned} \mu_3 &\sim \mathcal{N}(0, 0.01) \\ \sigma_3 &\sim \text{InvGamma}(1, 0.01) \end{aligned}$$

The priors  $\mathcal{N}(0, 0.01)$  and  $\text{InvGamma}(1, 0.01)$  are strongly informative, they both scales the hyper means  $\sigma_3$  and variance  $\sigma_3$  of the Housing variable close to zero. As explained in the previous model, this is to shrink the effect of the potentially irrelevant variable Housing onto the sampling process of the model.

### 3.2.2 Running the model

Before running, the input data was separated into training data and testing data with equal proportions (half) of the data. While the training set is used for the sampling process of the mode, the test set is utilized for predictions and accuracy testing purposes.

```
train_size <- round(nrow(data_hierarchical)/2, 0)
test_size <- nrow(data_hierarchical) - train_size
cat("Train size:", train_size, "/ Test size:", test_size)
```

Train size: 499 / Test size: 499

The model is then run with 4 chains, 2000 iterations and 1000 warm-ups.

```
# Create train and test sets
data_train <- head(data_hierarchical, train_size)
data_test <- tail(data_hierarchical, test_size)

# Create train and test target
y_train_hierarchical = data_train$Risk
y_test_hierarchical = data_test$Risk

# Create train and test observations
X_train_hierarchical = subset(data_train, select=-c(`Risk`, `Purpose`))
X_test_hierarchical = subset(data_test, select=-c(`Risk`, `Purpose`))

# Create train and test hierarchical groupings
ll_train = data_train$Purpose
ll_test = data_test$Purpose

# Create combined data
credit_data_hierarchical <- list(N_train=nrow(X_train_hierarchical),
                                K=ncol(X_train_hierarchical),
                                L_train=length(unique(ll_train)),
                                ll_train=ll_train,
                                X_train=X_train_hierarchical,
                                y_train=y_train_hierarchical,
                                N_test=nrow(X_test_hierarchical),
                                L_test=length(unique(ll_test)),
                                ll_test=ll_test,
                                X_test=X_test_hierarchical)

# Load and run data
hierarchical_fit <- stan(file="hierarchical.stan", data=credit_data_hierarchical,
                        chains=4, iter=2000, warmup=1000)
```

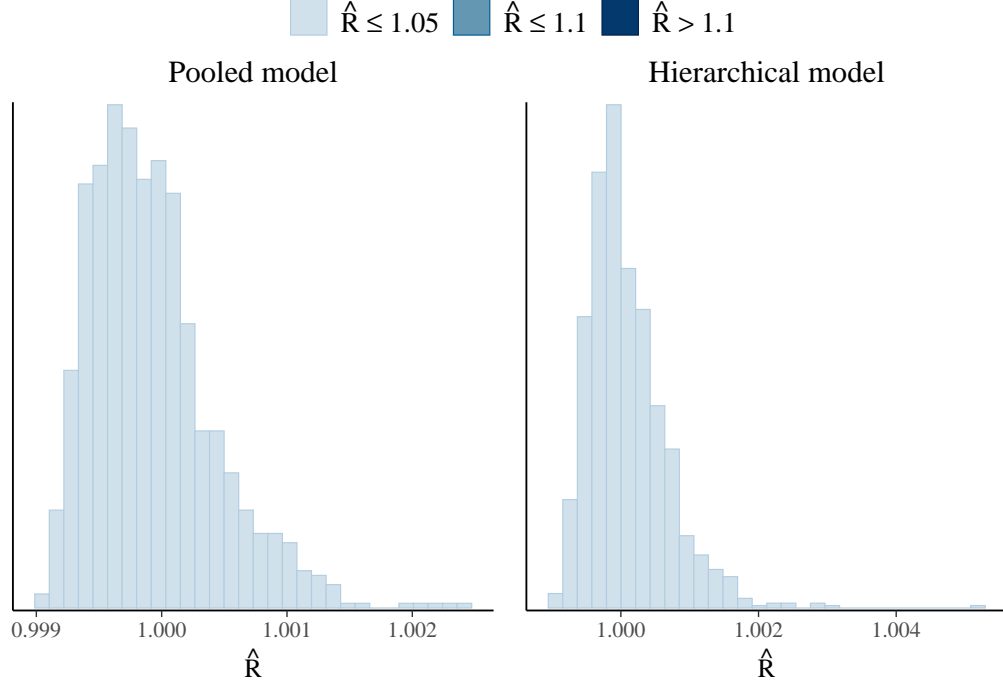
## 4 Results analysis

### 4.1 Coverage diagnostics

#### 4.1.1 $\hat{R}$ diagnostics

The  $\hat{R}$  values for the the models are plotted into histograms:

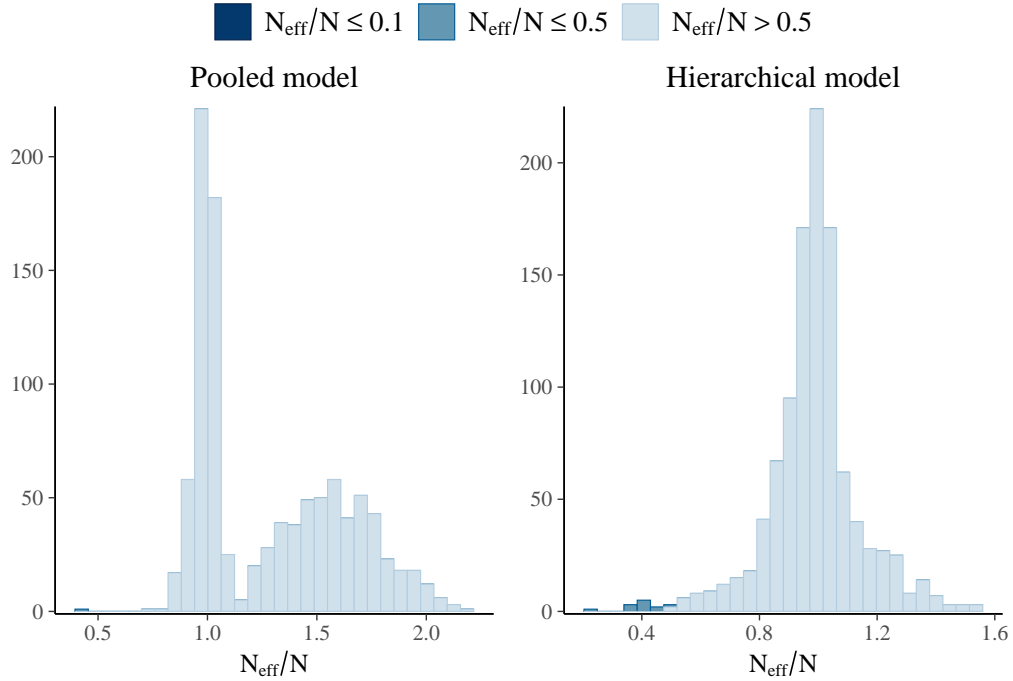




The histograms clearly show that all  $\hat{R}$  values are smaller than 1.05, and their distributions are relatively focused at 1.00. This suggests that the MCMC chains for both models have converged well.

#### 4.1.2 Effective sample size (ESS) diagnostics

Instead of directly plotting the ESS, we instead plot another metric called ESS ratio, which simply equals to  $N_{\text{eff}}/N_{\text{sample}}$  - the ratios of an effective sample size to total sample size.



From the histogram of the pooled model, although the  $N_{\text{eff}}$  has a relatively odd distribution for the effective sample size, it is nothing to worry about. More importantly, it can be observed that most of the ratios are populated around 1.00, and no ratio is below 0.1, which is a critical threshold for detecting instability in

the sampling process according to [Gelman et al. \(2013\)](#). This indicates that the pooled and the hierarchical models are stable and the results are reliable.

### 4.1.3 Divergences and tree depth diagnostics

We can access the parameters of the sampler for both models using `get_sampler_params`:

Pooled model:

```
summary(do.call(rbind, get_sampler_params(pooled_fit, inc_warmup=FALSE)))
```

```
##  accept_stat__      stepsize__      treedepth__      n_leapfrog__
##  Min.      :0.3205    Min.      :0.4737    Min.      :2.000    Min.      : 3.000
##  1st Qu.:0.8493    1st Qu.:0.5200    1st Qu.:3.000    1st Qu.: 7.000
##  Median :0.9356    Median :0.5475    Median :3.000    Median : 7.000
##  Mean   :0.9014    Mean   :0.5341    Mean   :2.959    Mean   : 7.091
##  3rd Qu.:0.9837    3rd Qu.:0.5616    3rd Qu.:3.000    3rd Qu.: 7.000
##  Max.   :1.0000    Max.   :0.5679    Max.   :4.000    Max.   :15.000
##  divergent__      energy__
##  Min.      :0      Min.      :290.1
##  1st Qu.:0      1st Qu.:295.5
##  Median :0      Median :297.5
##  Mean   :0      Mean   :297.8
##  3rd Qu.:0      3rd Qu.:299.7
##  Max.   :0      Max.   :314.2
```

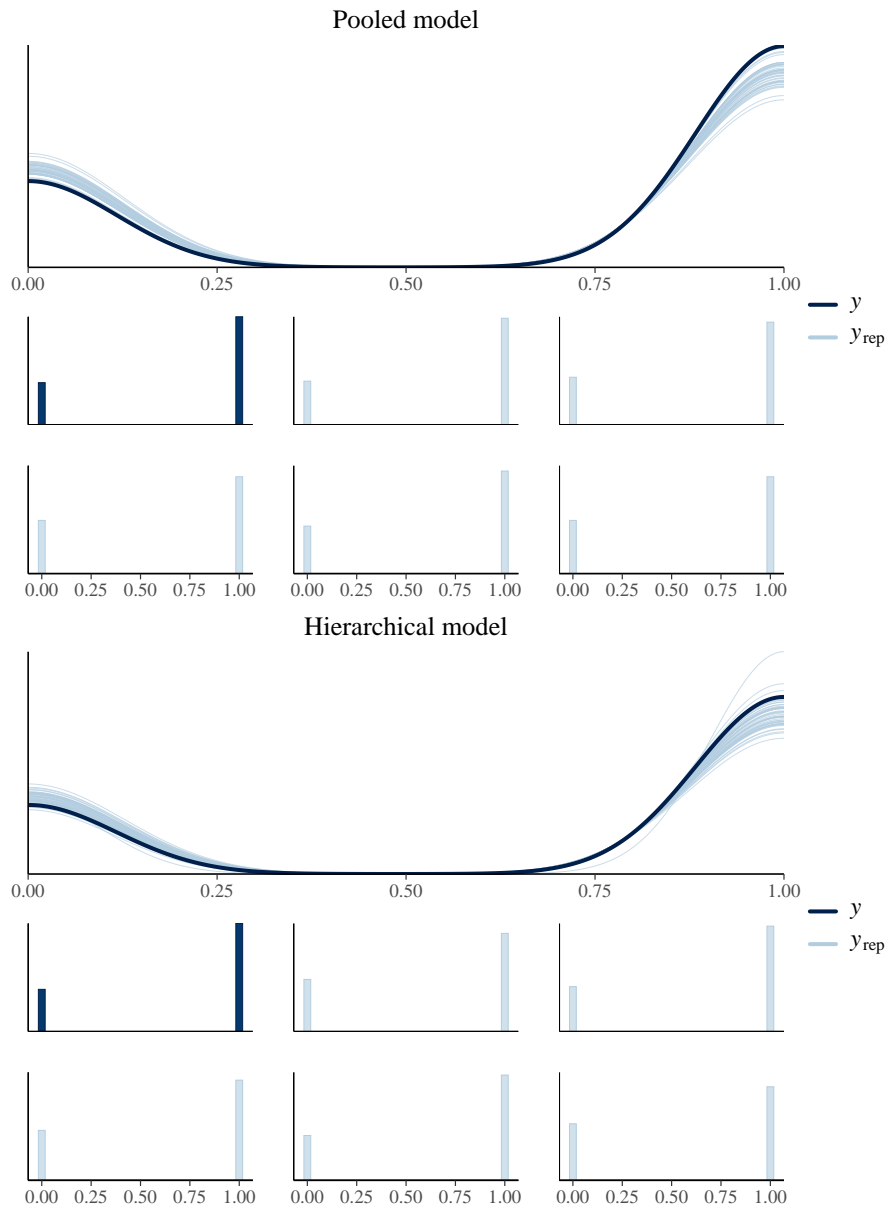
Hierarchical model:

```
summary(do.call(rbind, get_sampler_params(hierarchical_fit, inc_warmup=FALSE)))
```

```
##  accept_stat__      stepsize__      treedepth__      n_leapfrog__
##  Min.      :0.007168    Min.      :0.1977    Min.      :4.000    Min.      :15.00
##  1st Qu.:0.857444    1st Qu.:0.2155    1st Qu.:4.000    1st Qu.:15.00
##  Median :0.934623    Median :0.2256    Median :4.000    Median :15.00
##  Mean   :0.896838    Mean   :0.2235    Mean   :4.115    Mean   :19.04
##  3rd Qu.:0.975416    3rd Qu.:0.2335    3rd Qu.:4.000    3rd Qu.:15.00
##  Max.   :1.000000    Max.   :0.2450    Max.   :5.000    Max.   :63.00
##  divergent__      energy__
##  Min.      :0      Min.      :277.7
##  1st Qu.:0      1st Qu.:308.6
##  Median :0      Median :315.0
##  Mean   :0      Mean   :315.4
##  3rd Qu.:0      3rd Qu.:321.9
##  Max.   :0      Max.   :349.9
```

The NUTS sampler from both models encountered no divergences, as all the divergent statistics remain at 0. In addition, we can see that the tree depth maximum for both models does not exceed the threshold 10, indicating that the samplers are very stable.

## 4.2 Posterior predictive checks

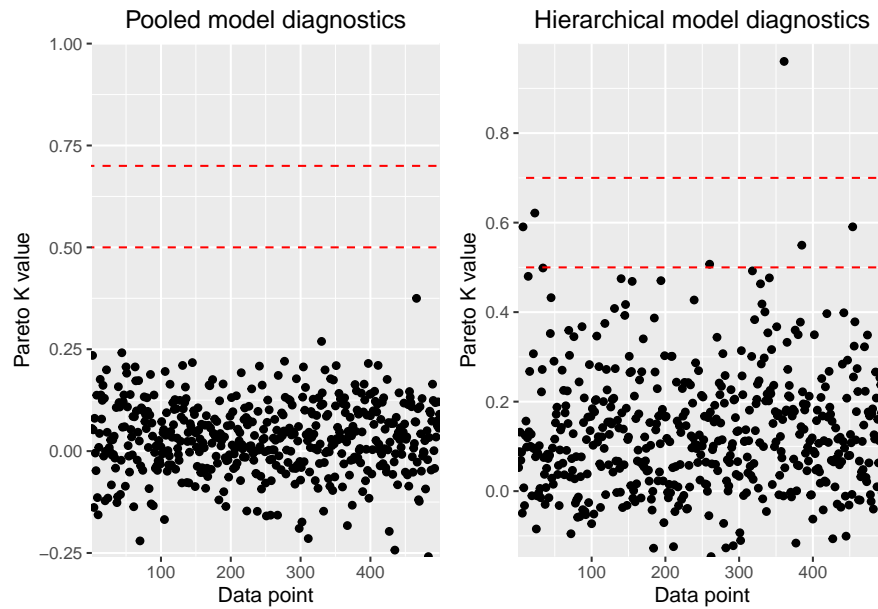


The density plots and the histograms draw the predictions from the testing data of both models and compared them with the original testing data. Although there is a small amount of noise, it is clear that the distribution of the predictions resembles that of the original data, implying that the pooled and hierarchical model has generated robust posterior predictions. In the predictive performance assessment section, we will provide numerical metrics to evaluate the true performance of both models' predicting ability.

## 4.3 Model comparison

Model comparison is performed using Leave-one-out cross-validation (LOO-CV), which will estimate the models' generalization performance.

Firstly, the Pareto  $k$  values are plotted in order to assess the reliability of the `elpd` estimates.



While all Pareto  $k$  values of the pooled model are below 0.5, indicating that the results are very reliable, one of the  $k$  values of the hierarchical model exceeds 0.7, which is usually the threshold for bad values, showing that the estimate for the hierarchical model might be too optimistic.

The LOO-CV estimation of `elpd` for both models is given and compared as follows:

Pooled model:

```
pooled_loo
```

Computed from 4000 by 499 log-likelihood matrix

```

      Estimate   SE
elpd_loo  -296.4 10.2
p_loo       9.5  0.6
looic      592.8 20.5
-----
```

Monte Carlo SE of `elpd_loo` is 0.0.

All Pareto  $k$  estimates are good ( $k < 0.5$ ).  
See `help('pareto-k-diagnostic')` for details.

Hierarchical model:

```
hierarchical_loo
```

Computed from 4000 by 499 log-likelihood matrix

```

      Estimate   SE
elpd_loo  -306.4 11.6
p_loo      41.7  3.0
looic      612.8 23.2
-----
```

Monte Carlo SE of `elpd_loo` is NA.

Pareto  $k$  diagnostic values:

		Count	Pct.	Min.	n_eff
(-Inf, 0.5]	(good)	493	98.8%	566	
(0.5, 0.7]	(ok)	5	1.0%	489	
(0.7, 1]	(bad)	1	0.2%	88	
(1, Inf)	(very bad)	0	0.0%	<NA>	

See `help('pareto-k-diagnostic')` for details.

```
loo_compare(pooled_loo, hierarchical_loo)
```

	elpd_diff	se_diff
model1	0.0	0.0
model2	-10.0	4.9

The `elpd` estimation (scaled by standard error) of the pooled model yields a higher value than that of the hierarchical model. Taking into account the fact that the hierarchical model also has instability resulting in a high Pareto  $k$  value, this suggests that the pooled model has a better performance and should be preferred.

## 4.4 Predictive performance assessment

To have a better insight into the models' predictive power, we compute a confusion matrix for them. For each data point of the testing set, we compute the mean of the predictions from 4000 chains. If the mean is larger than 0.5, the prediction is recorded as 1; otherwise, the prediction is rounded down to 0. The predictions are then compared with the original test set in order to compute the true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

Pooled model:

	Actual True	Actual False
Predicted True	320	39
Predicted False	101	39

Hierarchical model:

	Actual True	Actual False
Predicted True	308	51
Predicted False	100	40

Two metrics that we are using to assess the models' predictive performance are Accuracy - which measures the rate of correct predictions in all predictions - and Specificity - the rate of correct negative predictions in all negative predictions (bad loans). While the accuracy assesses the predictions of the models overall, the Specificity can measure the ability to actually detect a bad loan, which is more economically meaningful. The main target of banks when assessing the ability to repay a loan would be to avoid bad loans and losing money. Therefore, it is also vital to take Specificity, besides Accuracy, to evaluate the predicting power of our models.

These metrics can be obtained from the confusion matrix as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Pooled model:

Accuracy: 0.7194389 / Specificity: 0.5

Hierarchical model:

Accuracy: 0.6973948 / Specificity: 0.4395604

From the metrics obtained above, it is clear that the pooled model's predictions for all types of loans, and bad loans in particular are noticeably more robust than the hierarchical model.

Practically speaking, the hierarchical model, does not predict bad loans with enough accuracy (lower than 50%) to be taken into use. Even with a relatively good overall accuracy of 71.9%, the pooled model only has a 50% chance of correctly identifying bad loans, which is not economically meaningful. Therefore, both models cannot be applicable to real-life use cases.

## 4.5 Prior sensitivity analysis

For both models, we will use the regression intercept  $\beta_0$  and analyze the changes in its posteriors with different priors applied.

The  $\beta_0$  in the pooled model will assume a relatively more informative prior  $\mathcal{N}(2, 0.3)$  instead of the original  $\mathcal{N}(0, 1)$ .

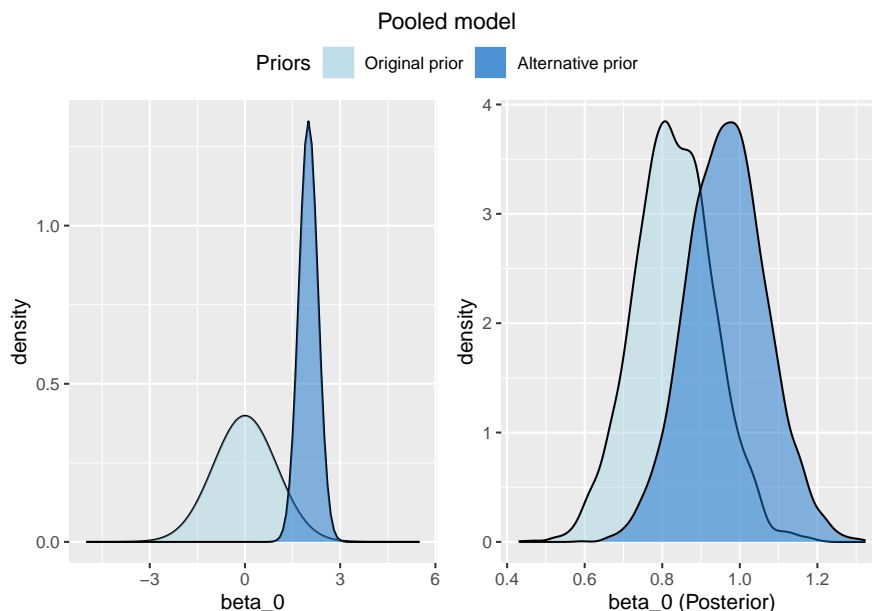
The hierarchical model, on the other hand, will still assume the prior  $\mathcal{N}(\mu_0, \sigma_o)$  for the intercept  $\beta_{0,1}$  but take in a more informative prior for the hyper-parameters:  $\sigma_0 \sim \mathcal{N}(2, 0.3)$  and  $\mu_0 \sim \text{InvGamma}(1, 0.1)$  instead of the original  $\mathcal{N}(0, 1)$ , and  $\text{InvGamma}(0.5, 1)$ , respectively.

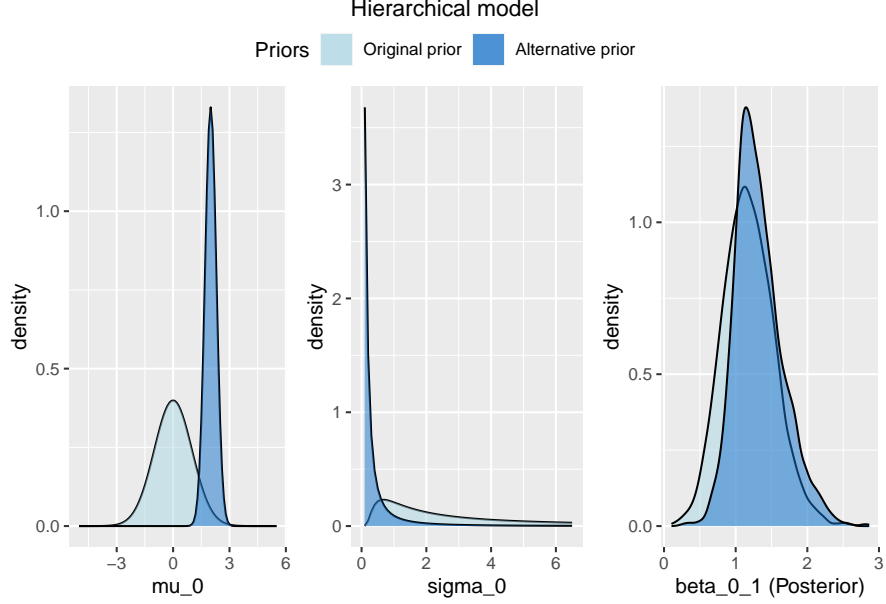
This is done by fitting another model, with the only differences being the priors being applied for  $\beta_0$  for the pooled, and  $\sigma_0, \mu_0$  for the hierarchical model:

```
prior_pooled_fit <- stan(file="prior_check_pooled.stan", data=credit_data_pooled,
                        chains=4, iter=2000, warmup=1000)
prior_hierarchical_fit <- stan(file="prior_check_hierarchical.stan", data=credit_data_hierarchical,
                             chains=4, iter=2000, warmup=1000)
```

All the chains from the prior-check models have converged and no divergence warnings are given. This may suggest that our alternative prior choices are reasonable enough to be taken as references for the models' sensitivity assessing.

The obtained posterior is compared with the original posterior along with the new and old priors as follows:





For the pooled model, it is anticipated that the posterior will be shifted away due to the change in location parameter from the prior, but we can see that it only shifts the distribution slightly. Even with a considerably smaller scale parameter for the prior, the shape of the distribution hardly changes. For the hierarchical model, even with a significantly more informative hyperprior being applied, the resulting shape of the posterior does not vary too much. Therefore, it is concludable that both of our models are relatively robust to prior changes.

## 5 Conclusion and potential improvements

### 5.1 Conclusion

This project aimed to conduct an analysis of two different Logistic Regression models, and the one that come out more reliable is the pooled model. We achieve relatively good accuracy for both models: hierarchical model with approximately 70% and pooled model with 72%. However, with such a poor performance in detecting bad loans, which is the main target of any credit lender, both models are not economically meaningful and would not be appropriate in real-life use cases. Nonetheless, the models produce robust results in general and are useful for research purposes, as there has not been many credit risk analysis using Bayesian methods available out there.

### 5.2 Issues and potential improvements

At the beginning of the project, we were having problems with finding the appropriate priors for our models. As we have limited knowledge regarding the credit market field, it is challenging to assume anything concrete and we have to use the generic priors instead. In addition, we also have a lot of issues with Stan. We initially started the project using `cmdstan`, but it has a very limited use case with other libraries as they only accept a `rstan` fit object as a parameter. Therefore, we have to switch everything to `rstan` and work from the beginning.

There is a lot of room for improvements that could be made to our project. One could be the imputation of missing data. The column `Checking.account` has a very high proportion of NA values (up to 39.4%) and imputing with mode was not the best option. Instead, we could use  $k$  nearest neighbor algorithm to impute the data better and avoid creating biases in the model. Another could be improving the prior estimation process. We could research more in the credit market field and introduce more relevant and robust prior distributions for our models. These measures may help improve the Specificity as well as the accuracy of the model and be more economically practical.

## 6 Self-reflection

Throughout the group project, we improved our understanding of Bayesian data analysis working flow. Unlike the assignments, where all the data, the priors, and the knowledge are given to us, we have to work very hard in order to gather data, try to find our own solutions for problems and build our own models and create the project in a duration of 1 week. We initially chose the topic as we have both had Economics courses and become particularly interested in how the credit market works, and we found this great opportunity to understand more about it. This project benefits us not only in knowledge but also in time management, task management, and problem-solving skill. We are satisfied we have created this project on our very own and learned so many valuable things from it.

## 7 Appendices and references

### 7.1 Stan code appendices

**Pooled model:**

```
data {
  int<lower=0> N_train;           // Number of observation in train set
  int<lower=0> K;                 // Number of columns
  matrix[N_train, K] X_train;    // Observations in train set
  int<lower=0, upper=1> y_train[N_train]; // Target value in train set

  int<lower=0> N_test;           // Number of observation in test set
  matrix[N_test, K] X_test;     // Observations in test set
}

parameters {
  real beta_0;
  vector[K] beta;
}

model {
  // Set priors
  beta_0 ~ normal(0, 1);
  beta ~ normal(0, 1);

  // Set regularization priors
  beta[3] ~ normal(0, 0.01);

  // Compute likelihood
  y_train ~ bernoulli_logit(beta_0 + X_train * beta);
}

generated quantities {
  // Compute predictions
  int<lower=0, upper=1> y_pred[N_test] = bernoulli_logit_rng(beta_0 + X_test * beta);

  // Compute log likelihood
  vector[N_train] log_lik;
  for(i in 1:N_train)
    log_lik[i] = bernoulli_logit_lpmf(y_train[i] | beta_0 + X_train[i] * beta);
}
```

**Hierarchical model:**

```
data {
  int<lower=0> N_train;           // Number of observation in train set
```



```

int<lower=0> K; // Number of columns
int<lower=0> L_train; // Number of groups in train set
matrix[N_train, K] X_train; // Observations in train set
int<lower=0, upper=L_train> ll_train[N_train]; // Group level in train set
int<lower=0, upper=1> y_train[N_train]; // Target value in train set

int<lower=0> N_test; // Number of observation in test set
int<lower=0> L_test; // Number of groups in test set
matrix[N_test, K] X_test; // Observations in test set
int<lower=0, upper=L_test> ll_test[N_test]; // Group level in test set
}

parameters {
  real mu_0;
  real<lower=0> sigma_0;
  real beta_0[L_train];

  real mu[K];
  real<lower=0> sigma[K];
  vector[K] beta[L_train];
}

model {
  // Set priors
  mu_0 ~ normal(0, 1);
  sigma_0 ~ inv_gamma(0.5, 1);

  beta_0 ~ normal(mu_0, sigma_0);

  mu ~ normal(0, 1);
  sigma ~ inv_gamma(0.5, 1);

  // Set regularization priors
  mu[3] ~ normal(0, 0.01);
  sigma[3] ~ inv_gamma(1, 0.01);

  // Set priors
  for(i in 1:L_train) {
    beta[i] ~ normal(mu, sigma);
  }

  // Compute likelihood
  for(i in 1:N_train)
    y_train[i] ~ bernoulli_logit(beta_0[ll_train[i]] + X_train[i] * beta[ll_train[i]]);
}

generated quantities {
  int<lower=0, upper=1> y_pred[N_test];
  vector[N_train] log_lik;

  // Compute predictions
  for(i in 1:N_test)
    y_pred[i] = bernoulli_logit_rng(beta_0[ll_test[i]] + X_test[i] * beta[ll_test[i]]);
}

```

```

// Compute log likelihood
for(i in 1:N_train)
  log_lik[i] = bernoulli_logit_lpmf(y_train[i] | beta_0[ll_train[i]] + X_train[i] * beta[ll_train[i]]);
}

```

#### Prior-testing pooled model:

```

data {
  int<lower=0> N_train;
  int<lower=0> K;
  matrix[N_train, K] X_train;
  int<lower=0, upper=1> y_train[N_train];

  int<lower=0> N_test;
  matrix[N_test, K] X_test;
}

parameters {
  real beta_0;
  vector[K] beta;
}

model {
  beta_0 ~ normal(2, 0.3); // Change prior
  beta ~ normal(0, 1);
  beta[3] ~ normal(0, 0.01);
  y_train ~ bernoulli_logit(beta_0 + X_train * beta);
}

generated quantities {
  int<lower=0, upper=1> y_pred[N_test] = bernoulli_logit_rng(beta_0 + X_test * beta);

  vector[N_train] log_lik;
  for(i in 1:N_train)
    log_lik[i] = bernoulli_logit_lpmf(y_train[i] | beta_0 + X_train[i] * beta);
}

```

#### Prior-testing hierarchical model:

```

data {
  int<lower=0> N_train;
  int<lower=0> K;
  int<lower=0> L_train;
  matrix[N_train, K] X_train;
  int<lower=0, upper=L_train> ll_train[N_train];
  int<lower=0, upper=1> y_train[N_train];

  int<lower=0> N_test;
  int<lower=0> L_test;
  matrix[N_test, K] X_test;
  int<lower=0, upper=L_test> ll_test[N_test];
}

parameters {
  real mu_0;

```

```

    real<lower=0> sigma_0;
    real beta_0[L_train];

    real mu[K];
    real<lower=0> sigma[K];
    vector[K] beta[L_train];
}

model {
    mu_0 ~ normal(2, 0.3);          // Change prior
    sigma_0 ~ inv_gamma(1, 0.1); // Change prior

    beta_0 ~ normal(mu_0, sigma_0);

    mu ~ normal(0, 1);
    sigma ~ inv_gamma(0.5, 1);

    mu[3] ~ normal(0, 0.01);
    sigma[3] ~ inv_gamma(1, 0.01);

    for(i in 1:L_train) {
        beta[i] ~ normal(mu, sigma);
    }

    for(i in 1:N_train)
        y_train[i] ~ bernoulli_logit(beta_0[l1_train[i]] + X_train[i] * beta[l1_train[i]]);
}

generated quantities {
    int<lower=0, upper=1> y_pred[N_test];
    vector[N_train] log_lik;

    for(i in 1:N_test)
        y_pred[i] = bernoulli_logit_rng(beta_0[l1_test[i]] + X_test[i] * beta[l1_test[i]]);

    for(i in 1:N_train)
        log_lik[i] = bernoulli_logit_lpmf(y_train[i] | beta_0[l1_train[i]] + X_train[i] * beta[l1_train[i]]);
}

```

## 7.2 References

- Hahn, Paul & Carvalho, Carlos. (2014). Decoupling Shrinkage and Selection in Bayesian Linear Models: A Posterior Summary Perspective. *Journal of the American Statistical Association*. 110. 10.1080/01621459.2014.993077.
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. Chapman & Hall/CRC Press, London, third edition