COVID-19 STOPGAP

The Search for a Practical Solution to an Overwhelming Problem Using Bayesian Statistics

Anup Sakpal (S3801788)
Cha Riofloriodo (S3789114)
Dilanka Wijeratne (S3808496)
Keavattey Srun (S3767615)
Millie Woo (S3806940)

MATH 2269: Applied Bayesian Statistics

Table of Contents

List of Figures

List of Tables

1. Introduction

During the early stages of the COVID-19 pandemic, the healthcare systems became overwhelmed due to the uncontrollable surge capacity and adverse effects experienced by our medical professionals. Some countries were able to manage the outbreak and flatten the curve successfully. However, other countries like the U.S. and Brazil are still unable to contain the virus's spread.

Six months ago, Albert Einstein Data4u posed a challenge on Kaggle to support hospitals in minimizing the risk of an overwhelmed health system and avoiding impractical testing of every COVID-19 case, especially in places with limited access to health services. The challenge focused on Brazil, a third-world country with some of the biggest and hardest-hit indigenous communities globally. It is believed that Brazil is not yet fully equipped to handle such a pandemic, financially and economically, as it is still recuperating from the destruction in the Amazon Rainforest.

In this report, we aim to help speed-up the screening tests on arrival to hospitals by predicting the presence of COVID-19 among the suspected cases using Bayesian statistics. This analysis is limited to the lab results at point-of-presentation or patient data collected during their visit to the emergency room.

2. Methodology

2.1 Dataset Description

The dataset provided by Einstein Data4u contained anonymized patient data from Hospital Israelita Albert Einstein in São Paulo, Brazil. It can be categorized according to the suspected cases, which indicated the COVID-19 test outcome, or according to the confirmed cases, which designated the admission to the general ward, semi-intensive, and intensive care unit.

It covers routinely collected data from various laboratory exams commonly ordered by the doctors during a visit to the emergency room. The tests depend on the onset of symptoms or type of complaints, which include some of the following:

- *Complete Blood Count with or without differential* – evaluates overall health and detects a wide range of disorders, including anemia, infection, and leukemia
- *Immuno-Serological test* – evaluates the performance of commercial antibody tests and detects any infections in the body
- *Complete or Basic Metabolic Panel* – assesses organ dysfunction, measures sugar level, kidney function, and electrolyte and fluid balance
- *Blood Gas Analysis (Arterial or Peripheral Venous)* – determines the acidity of the blood, indicating the presence of certain medical conditions such as kidney, heart, or lung failure
- *Urinalysis* – detects disorders such as urinary tract infections, kidney disease, and diabetes

The tests are not mandatory but may help detect the possibility of a favourable or unfavourable COVID-19 prognosis. The data also comprised the patient age quantile from 0 to 19, with the youngest patients at the lower end of the spectrum opposite the older patients at the higher end of the spectrum.

2.2 Dataset Specification

Medical literature published on medRxiv, a preprint server for health sciences, suggested that features routinely collected on presentation were most predictive of COVID-19 (Soltan et al., 2020). The clinical parameters encompassed complete blood count with differential, complete metabolic panel, blood gas, and vital signs all collected within one hour of presentation at the emergency room. An additional clinical journal from the Royal College of Physicians expressly implied that complete blood count has a high probability of identifying patients positive for SARS-COV-2. Information on this journal also mentioned that mean corpuscular volume, age, platelets, and eosinophils gave the highest contribution to a COVID-19 positive outcome (Formica et al., 2020).

Given the aim of this analysis, the dependent variable (y) focused only on the suspected cases, whether a patient is COVID-19 positive or negative. Age and complete blood count with differential measures were assigned as the predictor or independent variables (x_i):

- *Red Blood Cells (RBC)* – Hematocrit, Hemoglobin, Red Blood Cell Distribution Width, Red Blood Cell Count, Mean Corpuscular Hemoglobin, Mean Corpuscular Hemoglobin Concentration, Mean Corpuscular Volume
- *White Blood Cells (WBC)* – Leukocytes, Lymphocytes, Basophils, Eosinophils, Monocytes. Note that Neutrophils were excluded as there were inconsistencies with the collection and measurements on mature and immature neutrophils.
- *Platelets* – Platelets, Mean Platelet Volume

2.3 Data Preprocessing

When patients were asked to take a complete blood count test with differential, all the clinical parameters were analyzed, and all the results were provided accordingly. However, the dataset had missing values, most likely due to data entry or extraction. As the missing values were below 5%, the decision was to delete it. The final dataset now has a total sample size of n = 598 patients, containing one dependent variable and 15 independent variables discussed in Section 2.2.

According to Einstein Data4u, all clinical data were standardized to have a mean of zero and a unit standard deviation ("Diagnosis of COVID-19 and Its Clinical Spectrum", 2020). Therefore, no additional transformation was conducted.

2.4 Approach to Bayesian Statistics

a) Data Analysis:

i. We studied the descriptive statistics of the sampled data set. The summary of the data has been mentioned in Table 2.

ii. We studied the correlation of the chosen predictors and found that some 3 variables had a high correlation and decided to drop those variables. The Correlation matrix is shown in Table 3.

iii. We have studied the scatter plot to give a fair idea of the nature of the outcome of the test result, depending on the values of the hematology tests.

b) Choosing Model: We decided to choose **Logistic Regression for Binary Classification** as we want to predict the outcome of the test of the patient being positive or negative depending on the complete blood count test. Binary variables were assigned as Positive = 1, Negative = 0 . We have also taken into consideration the outcome of the tests on different groups of age quantile and did our analysis using **hierarchical modelling.**

c) Logistic Regression:

i. **Splitting Train and Test Data:** As the data has a highly imbalanced class, stratified sampling was used to split data into train and test based on 70:30 ratio.

ii. **Implement Logistic Regression:** Fit model on train data with non-informative priors with the 12 betas and guess. Guess was used for robustness in taking care of outliers and small values.

iii. **Analysis of Logistic Runs:** Check posterior distributions of a logistic regression model. In the Bayesian estimates, some betas are capturing 0 within HDI interval. Performance of the chains were analysed using accuracy and predictive checks.

iv. **Model Comparison:** 4 models are proposed for our findings considering different aspects of intercept and betas and analysed their respective predictive checks. Finally, we chose the best model, which was model 4 in this case, to reach our conclusions.

d) Hierarchical Modelling:

i. **Group Data :** We have grouped the data into 4-age quantile groups present in the data set. The 4 groups contain a group of 5 age quantiles, and each group accounts for 25 percent of the total data. We also discussed the properties of each quantile with respect to their test results.

ii. **Implement Hierarchical Model:** We defined the model with **Bernoulli distribution** and **non-informative** priors for MCMC analysis.

iii. **Analysis of Hierarchical Runs:** We analysed the posterior distributions of all the age quantile groups followed by their sensitivity analysis to reach our findings and conclusion.

3. Descriptive Statistics

3.1 Summary Statistics

Table 1: Count and Proportion of SARS COV2 Test Result

SARS Cov2 exam result	Positive	Negative
Count	81	517
Proportion	0.1354	0.8646

Table 2: Summary Statistics (Independent Variables)

Paramters	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Age.quantile	11.14214	5.69123	12	11.425	7.413	0	19	19	-0.33	-1.0157	0.23273
Hematocrit	0.003075	1.00243	0.05341	0.06261	0.95	-4.501	2.663	7.1641	-0.74	1.4215	0.04099
Hemoglobin	0.005635	1.00125	0.04032	0.05624	0.929	-4.346	2.672	7.0175	-0.64	1.1798	0.04094
Platelets	0.00657	0.99475	-0.1092	-0.0496	0.829	-2.552	9.532	12.084	1.84	13.714	0.04068
MPV	-1.78E-05	1.00167	-0.1015	-0.0468	0.998	-2.458	3.713	6.1706	0.422	-0.0396	0.04096
RBC	-0.00944	0.98784	0.01385	0.02318	0.915	-3.971	3.646	7.6163	-0.44	1.2211	0.0404
Lymphocytes	-0.00082	1.00254	-0.0143	-0.0471	0.986	-1.865	3.764	5.6292	0.473	0.1352	0.041
MCHC	0.010023	0.99196	-0.0546	0.02321	0.886	-5.432	3.331	8.7629	-0.49	2.2663	0.04056
Leukocytes	0.006673	1.00031	-0.2087	-0.1135	0.72	-2.02	4.522	6.5423	1.423	2.898	0.04091
Basophils	0.000986	1.00296	-0.2238	-0.1079	0.906	-1.14	11.08	12.218	2.913	24.674	0.04101
MCH	0.022934	0.95054	0.1259	0.06862	0.775	-5.938	4.099	10.036	-0.79	3.8432	0.03887
Eosinophils	0.003968	1.00278	-0.3298	-0.1789	0.625	-0.836	8.351	9.1864	2.74	12.904	0.04101
MCV	0.021765	0.95499	0.07606	0.04514	0.846	-4.581	3.411	7.9918	-0.33	1.62	0.03905
Monocytes	-0.00368	0.99805	-0.1152	-0.073	0.818	-2.164	4.533	6.6971	0.957	1.9864	0.04081
RDW	-0.01934	0.96962	-0.1828	-0.1483	0.656	-1.598	6.982	8.5803	2.376	9.6214	0.03965

From the 598 records of the patients, 81 of the patients have tested positive for SARS-COV-2, which account for 13.54%, while 517 have tested negative, which account for 86.46%.

The following are the interpretations for each of the variable in the Complete Blood Count test results:

i) **Hematocrit** is the volume percentage of RBC. It has a standard deviation of around 1 and ranges from -4.501 to 2.663 in the dataset.

ii) **Hemoglobin** is the protein that carries oxygen to the blood. It almost has the same range of reading as hematocrit.

iii) **Platelets** are tiny blood cells that help the body to stop bleeding. They have median of -0.1092 and kurtosis of 13.714, which means that the data have heavier tails and are more concentrated around the mean than a normal distribution.

iv) **Mean Platelet Volume (MPV)** is a machine-calculated measurement of the average size of platelets found in blood and is typically included in blood tests as part of the CBC. The mean of this variable is very low as -1.78E-05, whereas the median is about -0.1015.

v) **Red Blood Cells** give information on the hemoglobin content and the size of RBC. The median of the RBC is about 0.013852.

vi) **Lymphocytes** are WBC, and also one of the body's main types of immune cells. It has a very low mean of -0.00082 and a median of -0.01427.

vii) **Mean Corpuscular Hemoglobin Concentration (MCHC)** is a measure of the concentration of hemoglobin in each volume of a packed red blood cell. It is calculated by dividing the hemoglobin by the hematocrit. The median of this variable is -0.0546.

viii) **Leukocytes** are the cells of the immune system involved in protecting the body against both infectious disease and foreign invaders. It has a low mean of 0.006673 and a median of -0.2087.

ix) **Basophils** are white blood cells from the bone marrow that play a role in keeping the immune system functioning correctly. It has a high kurtosis of 24.674, which indicates a heavy-tailed distribution.

x) **Mean Corpuscular Hemoglobin (MCH)** is the average mass of hemoglobin per red blood cell in a sample of blood. It has a median of 0.1259.

xi) **Eosinophil** is a type of disease-fighting white blood cell. This condition most often indicates a parasitic infection, an allergic reaction, or cancer. The mean of its measurement is 0.003968 and has a leptokurtic distribution (i.e., kurtosis statistic of 12.904).

xii) **Mean corpuscular volume (MCV)** measures the average size and volume of RBC. The median of the measurement is 0.021765.

xiii) **Monocyte** is a type of leukocyte or white blood cell. As a part of the vertebrate innate immune system, monocytes also influence the process of adaptive immunity. It has a median of -0.1152.

xiv) **Red Blood Cell Distribution Width (RDW)** measures the amount of red blood cell variation in volume and size. The median of RDW is -0.1828.

Overall, we can see that the mean and median of the parameters, except for the Age Quantile. Note that the variables' standard deviation is about 1, as the dataset has already been standardized (See section 2.3). We can say these microscopic readings of the variables need to be carefully analyzed, and we need a thorough understanding of the CBC terms to do Bayesian analysis.

3.2 Correlation Analysis

Table 3: Correlation Matrix (Independent Variables)

Paramter	Age.quantile	Hematocrit	Hemoglobin	Platele	MPV	RBC	Lymphocytes	MCHC	Leukocytes	Basophils	MCH	Eosinophils	MCV	Monocytes	RDW
Age.quantile	1	0.102	0.066	-0.148	0.12	-0.04	-0.122	-0.122	-0.163	0.103	0.221	0.022	0.308	0.045	0.16
Hematocrit	0.102	1	0.969	-0.089	0.084	0.891	0.003	0.129	-0.095	0.129	0.07	0.027	0.017	0.085	-0.263
Hemoglobin	0.066	0.969	1	-0.129	0.079	0.865	-0.003	0.368	-0.11	0.116	0.175	0.015	0.01	0.1	-0.338
Platelets	-0.148	-0.089	-0.129	1	-0.36	-0.05	0.086	-0.17	0.439	-0.02	-0.132	0.168	-0.06	-0.2	0.02
MPV	0.12	0.084	0.079	-0.357	1	0.043	0.079	-0.004	-0.155	0.129	0.07	-0.047	0.079	0.038	0.046
RBC	-0.043	0.891	0.865	-0.046	0.043	1	-0.009	0.116	-0.025	0.08	-0.336	0.001	-0.43	0.039	-0.184
Lymphocytes	-0.122	0.003	-0.003	0.086	0.079	-0.01	1	-0.027	-0.333	0.238	0.016	0.201	0.028	0.066	-0.08
MCHC	-0.122	0.129	0.368	-0.17	-0	0.116	-0.027	1	-0.078	-0.03	0.459	-0.048	-0.01	0.084	-0.383
Leukocytes	-0.163	-0.095	-0.11	0.439	-0.16	-0.03	-0.333	-0.078	1	-0.31	-0.159	-0.096	-0.13	-0.293	0.157
Basophils	0.103	0.129	0.116	-0.023	0.129	0.08	0.238	-0.027	-0.306	1	0.067	0.335	0.089	0.098	0.039
MCH	0.221	0.07	0.175	-0.132	0.07	-0.34	0.016	0.459	-0.159	0.067	1	0.017	0.884	0.115	-0.246
Eosinophils	0.022	0.027	0.015	0.168	-0.05	0.001	0.201	-0.048	-0.096	0.335	0.017	1	0.043	0.01	0.004
MCV	0.308	0.017	0.01	-0.059	0.079	-0.43	0.028	-0.007	-0.134	0.089	0.884	0.043	1	0.084	-0.085
Monocytes	0.045	0.085	0.1	-0.2	0.038	0.039	0.066	0.084	-0.293	0.098	0.115	0.01	0.084	1	-0.026
RDW	0.16	-0.263	-0.338	0.02	0.046	-0.18	-0.08	-0.383	0.157	0.039	-0.246	0.004	-0.09	-0.026	1

From Table 3, we can see four variables have high multicollinearity as highlighted in red. Collinearity between hematocrit and hemoglobin is at 0.969, hematocrit and RBC is at 0.891, hemoglobin and RBC is at 0.865, and MCH and MCV is at 0.884. As a result, hematocrit, RBC Count, and MCV were removed to minimize the effects on the standard errors and precision of the regression coefficient models.

3.3 Scatter Plots

The value of zero on the y-axis corresponds to a negative outcome of the result, while 1 corresponds to a positive outcome.

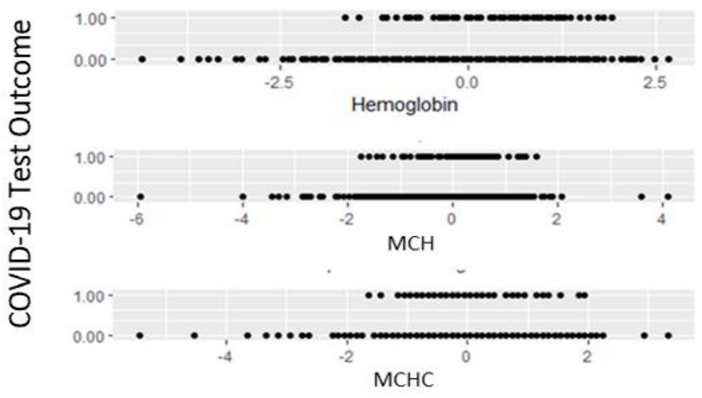


Figure 1: Scatter Plots of the Test Outcome against Hemoglobin, MCH, and MCHC

It is evident that the lower values of hemoglobin, MCH, and MCHC are mostly associated with the test's negative outcome (Figure 1).

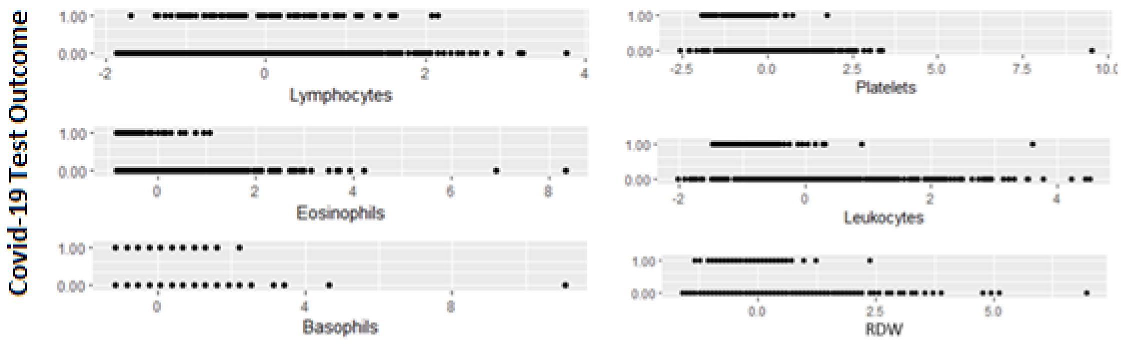


Figure 2: Scatter Plots of the Test Outcome against Lymphocytes, Platelets, Eosinophils, Leukocytes, Basophils, and RDW

On the contrary, it is visible from the scatter plots in Figure 2 that higher values of Lymphocytes, Platelets, Eosinophils, Leukocytes, Basophils, and RDW are mostly associated with the negative test outcome.

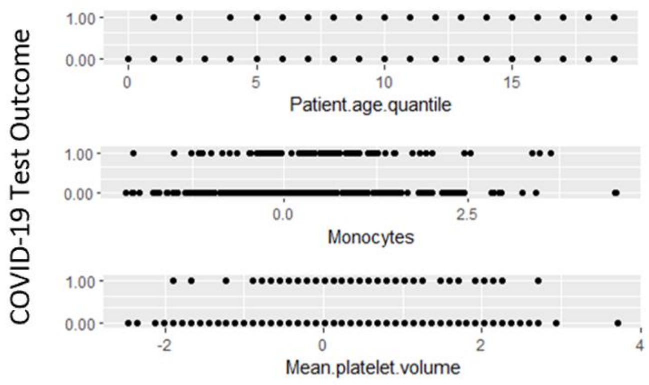


Figure 3: Scatter Plots of the Test Outcome vs. Patient Age Quantile, Monocytes, and MPV

The scatter plots in Figure 3 suggest that the patient age quantile, monocytes, and MPV are unlikely to support the evidence of the patient being tested positive or negative.

4. Logistic Regression

4.1 Mathematical Model

From the descriptive statistics in section 3, the dependent feature (COVID test result) has two classes—negative and positive. Thus, Bernoulli distribution was considered a likelihood distribution.

The mathematical of the regression model in this analysis is stated below.

$$Y \sim Bernoulli\ (\mu)$$

$$\boxed{\mu = logistic\ (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k)}$$

To reduce the impact of outliers on the model, an additional predictor called "guessing parameter" is included in the model.

$$\boxed{\mu = \alpha \cdot 1/2 + (1-\alpha) \cdot logistic\ (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k)}$$

Where $\alpha \sim dbeta\ (a,b)$, x are the 12 predictors, and β are the coefficient of each predictor.

Therefore,

$$\boxed{\begin{aligned}\mu = \alpha \cdot 1/2 + (1-\alpha) \cdot logistic\ (&\beta_0 + \beta_1\ Age\ quantile + \beta_2 Hemoglobin + \beta_3 Platelets \\ &+ \beta_4 Mean\ platelet\ volume + \beta_5 Lymphocytes + \beta_6 MCHC + \beta_7 Leukocytes \\ &+ \beta_8 Basophils + \beta_9\ MCH + \beta_{10} Eosinophils + \beta_{11} Monocytes + \beta_{12} RDW\end{aligned}}$$

4.2 Specification of the Prior Distribution

4.2.1 JAGS Model Diagram

JAGS model diagrams below explain the hierarchical model of this Bayesian logistic regression. As shown in Figure 4, the diagrams themselves are self-explanatory. The data is distributed Bernoulli with the success probability μ_i for ith object. The μ_i equal to logistic regression model $logistic\ (\beta_0 + \Sigma_j \beta_j x_j)$, each beta parameter has a normal distribution, and guess parameter has a beta distribution.

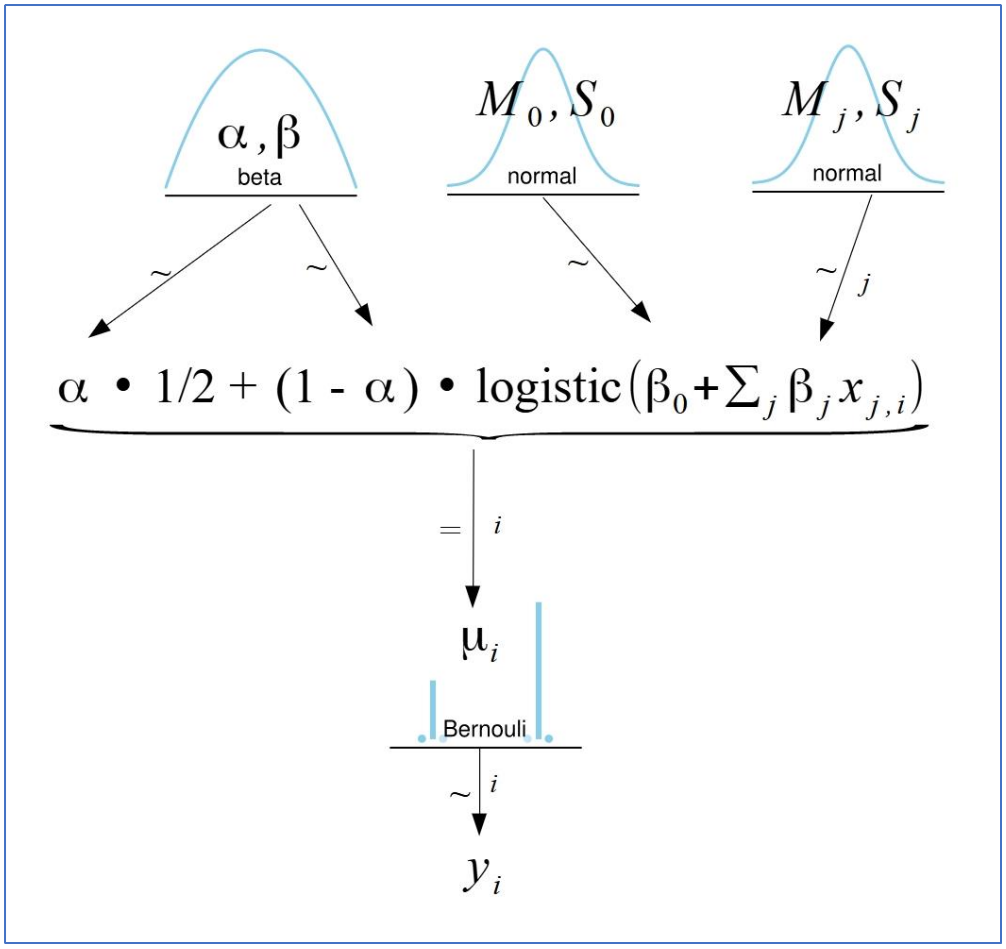


Figure 4: JAGS Model Diagram for Logistic Regression

4.2.2 Prior Information

For every independent variable, we used the values given as the mean and chose values of variance to reflect the degree of belief as given above. There is "no expert knowledge" on each independent variable and intercept, the variance of prior distributions was determined to be a big value to have distribution a dispersed one. We selected 0 and 4 as mean and variance, respectively, to make its distribution less concentrated.

Furthermore, there is no prior information on the guess parameter, either. Hence, the mean and variance were specified at 1 and 9 of the guess parameter's distribution to indicate a vague prior.

The sensitivity analysis of prior distributions of independent variables will be discussed in a section [4.6].

4.3 MCMC Settings and Diagnostic Checks

Since there are many parameters of interest to be found, the joint priors are not conjugate. Therefore, we used the Markov Chain Monte Carlo (MCMC) method, with the assistance of JAGS to run the models in R studio to find the posterior distributions of all parameters as well as the diagnostics checking.

Generally, there are three requirements to assess the appropriateness of MCMC diagnostics.

- **Representativeness:** shrink factor is well below 1.2, desirably around 1. Chains converge by overlapping in the trace plots and fluctuate around the mean. Density plots and the HDI intervals of all chains overlap each other.
- **Accuracy:** There is no significant autocorrelation. ESS is high, and MCSE is low.
- **Efficiency:** Run time of running the whole MCMC procedure is checked if it is a reasonable duration.

To improve the efficiency and accuracy of the MCMC process, we went straight with the parallel run. As the dataset contains standardized values, no scaling was needed.

Several trials and errors were conducted to find a suitable MCMC setting to produce acceptable diagnostics plots of all parameters and find ways to improve efficiency.

Starting off the run without specifying the initial list, the model could run fine. Thus, we just relied on MCMC to generate the initial list automatically.

There were 4 trials in total. Table 4 is the summary of all trials with different parameters of the MCMC setting.

Table 4: Summary List of Trials for Logistic Regression

Trial No.	Adapt Steps	Burn-in	# Chains	Thinning Steps	# Saved Steps	# of Iterations	Elapsed Time (second)
1	1,000	1,000	3	10	1,000	3,333.33	84.06
2	1,000	1,000	3	20	3,000	20,000	328.49
3	1,000	1,000	3	30	3,000	30,000	501.92
4	1,000	1,000	3	40	3,000	40,000	751.30

Trial No. 1 started with thinning steps of 10, other parameters as listed in Table 4. It ran on the full dataset, and the process took only 84.06 seconds to run and resulted in undesirable diagnostic plots of all parameters. Diagnostic plots of beta0 and beta1 parameters were the most problematic (See Figure 5). The shrink factor of beta0 started at 1.15, which is a bit of a concern, but it went down to 1 in later iterations. In comparison, the shrink factor of beta1 is good as it fluctuates around 1. Trace plot of both betas fluctuates around the mean, but the chains do not converge very well. More importantly, there were still some high autocorrelations, especially in beta0.

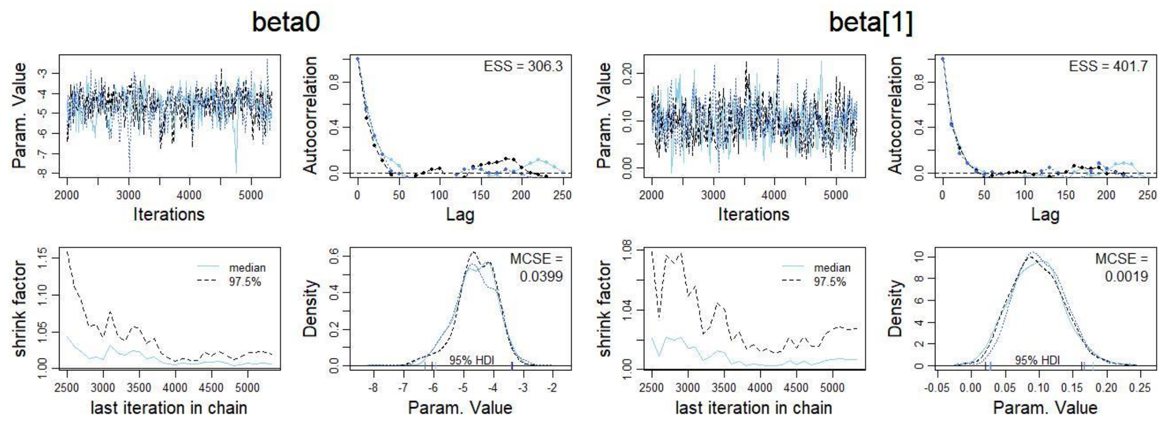


Figure 5: Diagnostics Plots of beta0 and beta1 in Trial No. 1

The diagnostics plots of other parameters were better than these two betas (see Appendix A). However, since the diagnostics of these two betas did not meet the requirement, we increased the number of saved steps to 3000, hoping to improve the shrink factor further, and increased the number of thin steps to 20 to get rid of the autocorrelation in Trial No. 2. It took 328.49 seconds, about 4 times longer than Trial No.1

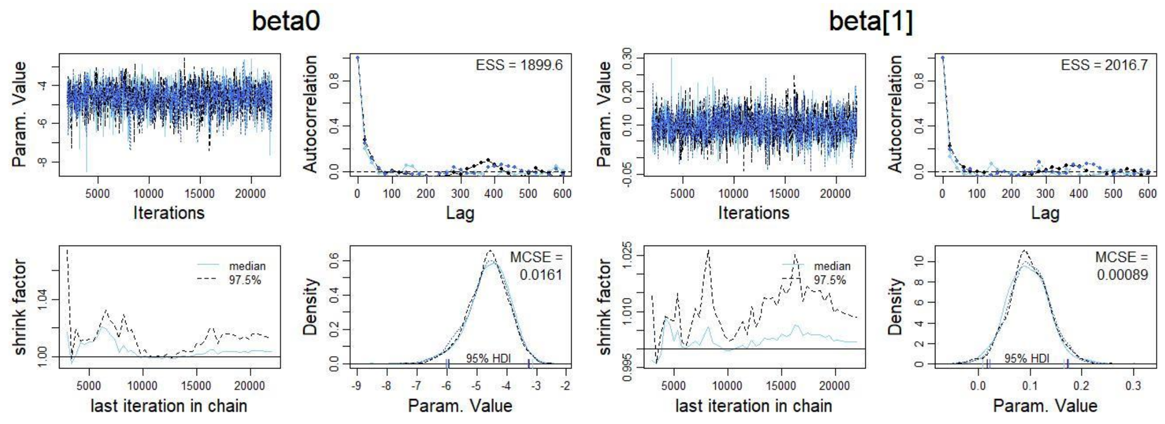


Figure 6: Diagnostics Plots of beta0 and beta1 in Trial No. 2

As noticed in Figure 6, the result of diagnostics had improved. The shrink factor of beta0 went to around 1 and was not seen anywhere near 1.2; hence, it was no longer a concern. However, we still could see some high significant correlations, which is higher than 0.2 there. To further improve the autocorrelation, we increased the thin steps to 30 in Trial No. 3.

Trial No. 3 ran for 501.92 seconds (8.36 minutes), and the results show that the autocorrelation of both parameters went down to below 0.2. It was optimistic that we could get rid of high autocorrelation entirely if we keep increasing the thinning steps. Therefore, we decided to set the number of thinning steps at 40 in Trial No. 4.

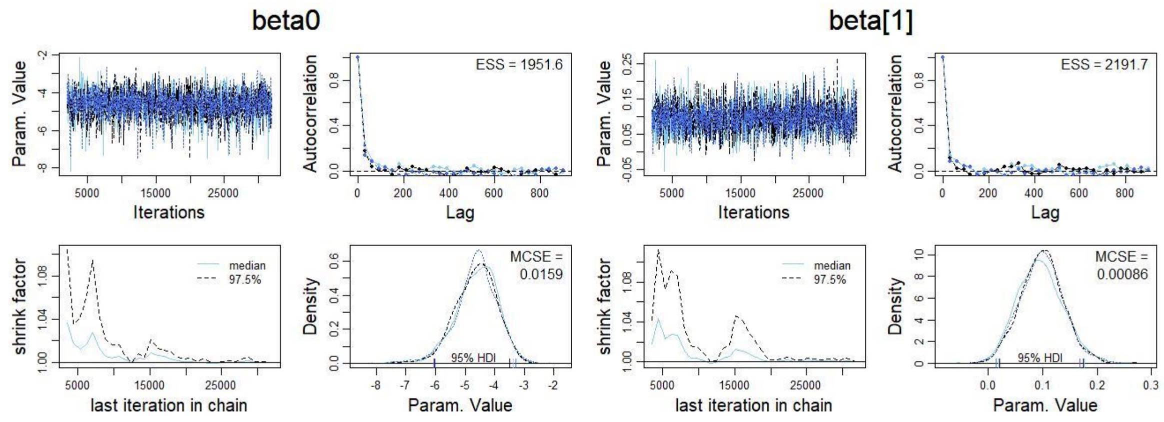


Figure 7: Diagnostics Plots of beta0 and beta1 in Trial No. 3

As a result, diagnostics of all parameters passed the requirements with the MCMC settings of this trial. The diagnostic plots of all parameters are illustrated in Figure 8 (See pages 14-16). In summary:

- The shrink factor is well below 1.2, suggesting the chains are fully converged. The three chains are superimposed and fluctuate around the mean in the trace plots, which indicates representativeness.
- Additionally, the density plots of the three chains are overlapping each other very well, while the 95% HDI is slightly different for each chain yet acceptable. These results signify the chains are producing representative values for the posterior distributions.
- The autocorrelation plots show no significant autocorrelation. The Effective Sample Size (ESS) are all over 2,500, which is acceptable.
- Monte Carlo Standard Error (MCSE) resulted in very small values (less than 1 and close to 0), so we can infer the accuracy of the parameter values.
- The elapsed time of Trial No. 4 is 751.30 seconds (approx. 13mns), which is not too long.

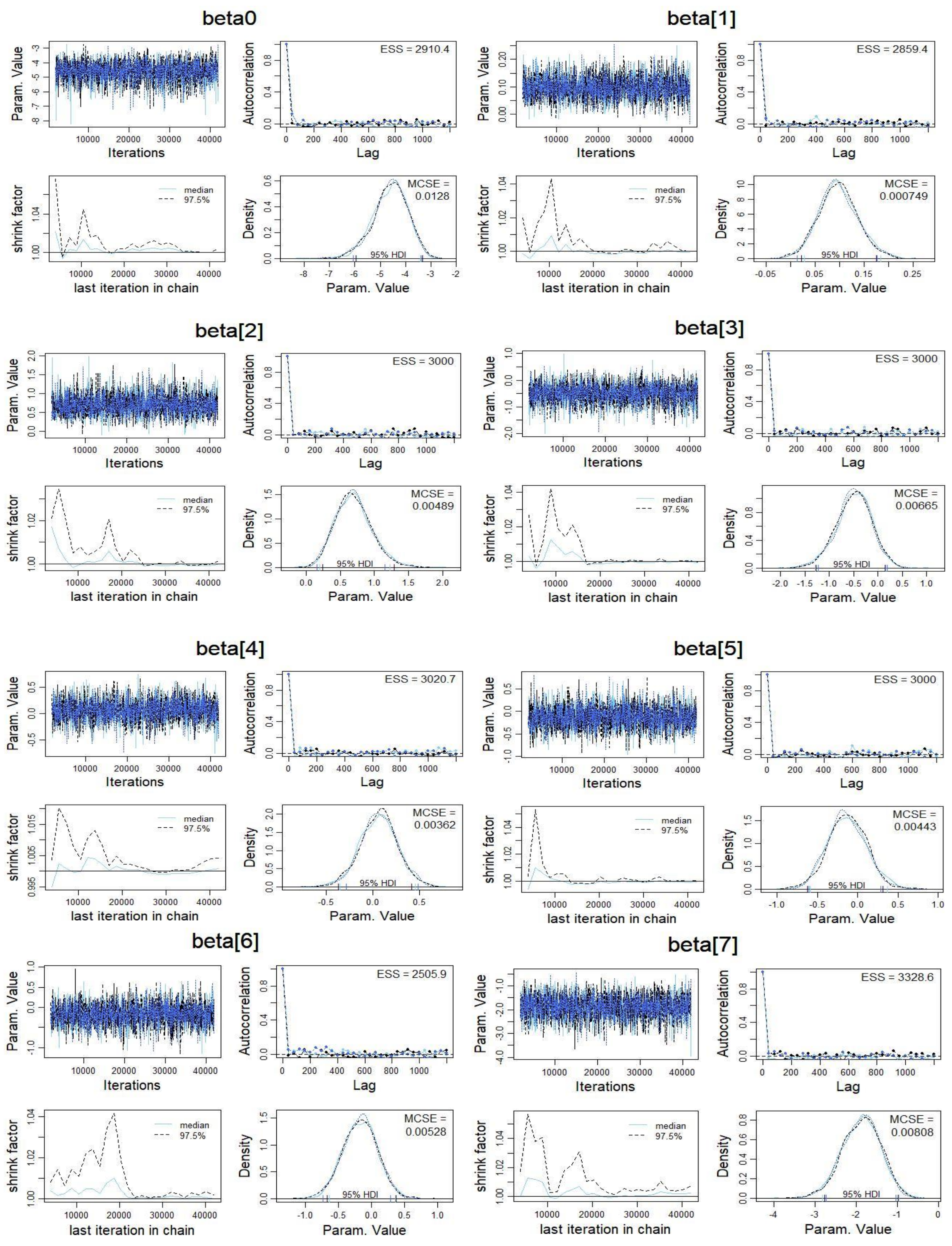


Figure 8: Diagnostics Plots of All Parameters in Trial No. 4

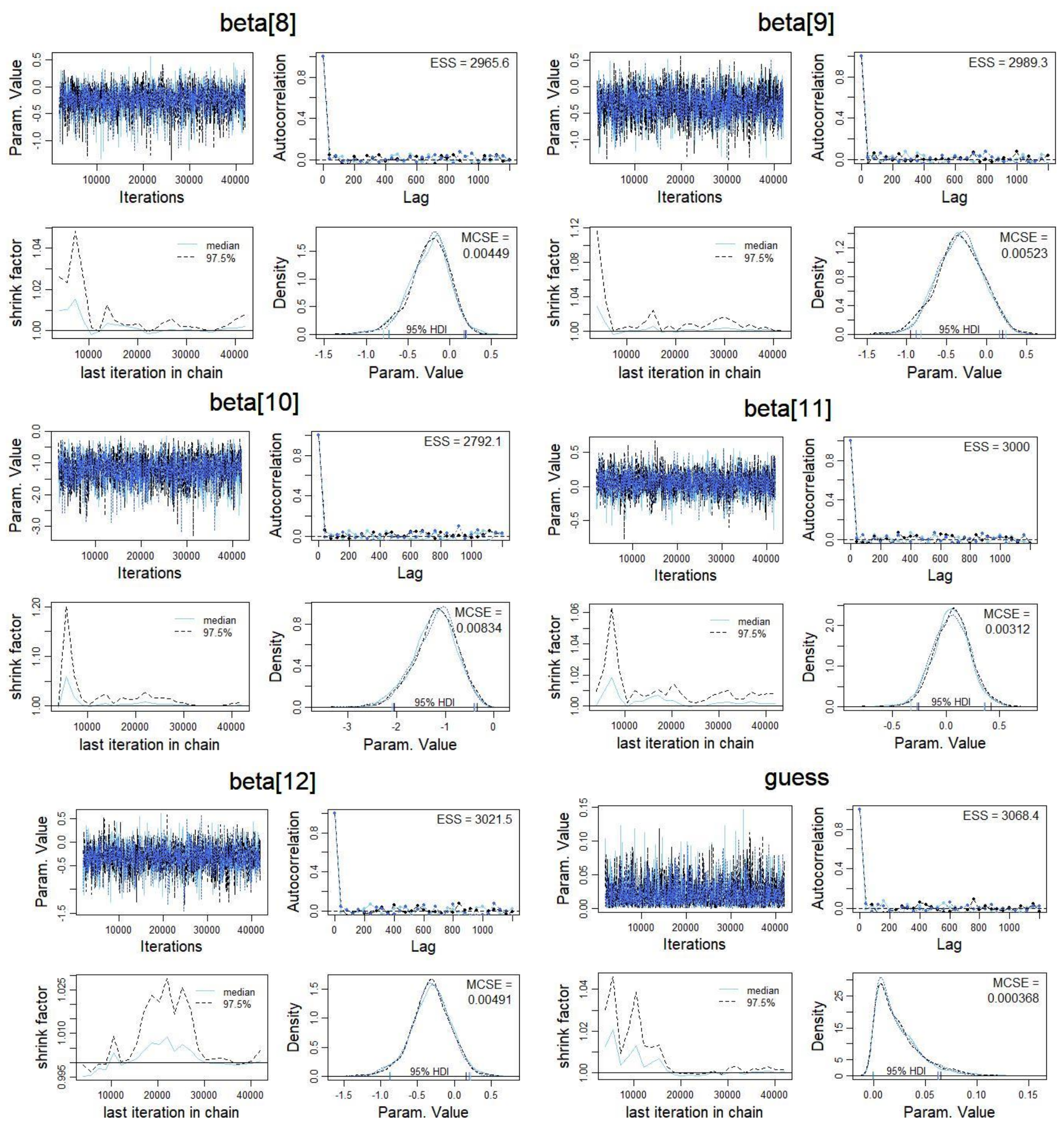


Figure 8: Diagnostics Plots of All Parameters in Trial No. 4, continued

4.4 Posterior Distributions

After the MCMC diagnostics of all parameters have been validated, we got the Bayesian estimate of all parameters with 95% HDI intervals as follows.

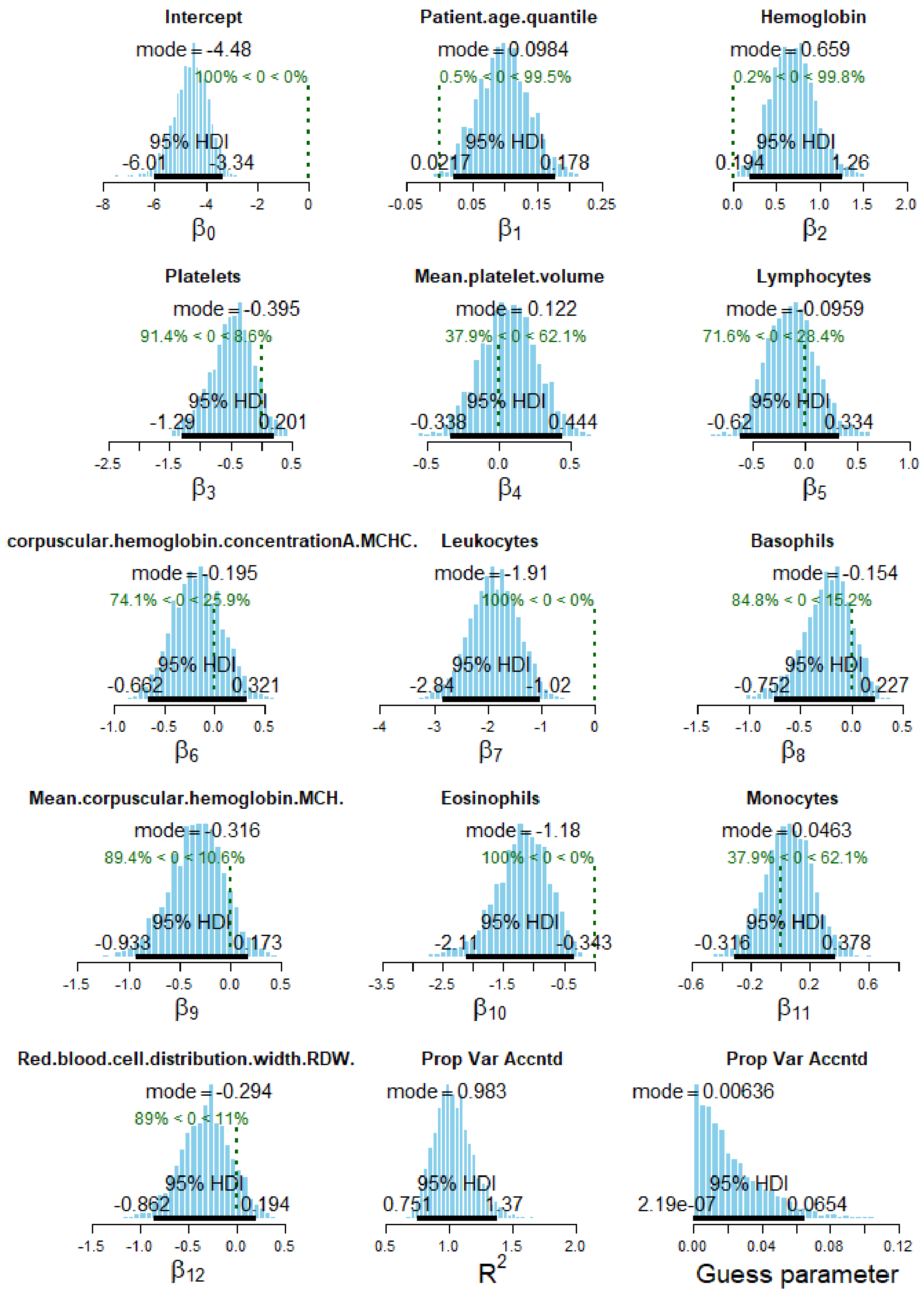


Figure 9: Posterior Distributions of All Parameters of Logistic Regression

Table 5: Bayesian Parameter Estimates for Logistic Regression Model

Parameters	Mean	Median	Mode	Exp(Mode)	1/Exp(Mode)	HDIlow	HDIhigh
beta0	-4.62141	-4.56881	-4.4847			-6.00946	-3.34039
beta[1]	0.097678	0.09736	0.0984	1.1034072	0.906283754	0.021672	0.177681
beta[2]	0.695636	0.684406	0.6586	1.9321552	0.517556776	0.193605	1.26356
beta[3]	-0.48811	-0.46772	-0.3949	0.6737791	1.484165853	-1.2913	0.201292
beta[4]	0.059872	0.062257	0.1217	1.1294605	0.88537844	-0.33843	0.443931
beta[5]	-0.13151	-0.13606	-0.0959	0.9085925	1.100603447	-0.61975	0.33357
beta[6]	-0.16746	-0.17066	-0.1954	0.8225257	1.215767496	-0.66214	0.320641
beta[7]	-1.88999	-1.88313	-1.9110	0.147929	6.759998143	-2.83571	-1.01754
beta[8]	-0.24316	-0.22261	-0.1539	0.8573208	1.166424579	-0.75213	0.227366
beta[9]	-0.33619	-0.32655	-0.3163	0.7288682	1.371990206	-0.93298	0.172806
beta[10]	-1.21831	-1.18098	-1.1775	0.3080442	3.246287847	-2.10975	-0.3435
beta[11]	0.049595	0.051844	0.0463	1.0474004	0.954744746	-0.31625	0.377692
beta[12]	-0.3298	-0.31842	-0.2940	0.7452852	1.341768149	-0.86186	0.193736

Table 5 summarizes the parameter estimates. It is worth to mention that all parameters are in the unit of a standardized scale. The corresponding interpretations are explained below:

Significant coefficients were observed for the following variables: Intercept (β_0), Patient age quantile (β_1), Hemoglobin (β_2), Platelets (β_3), Leukocytes (β_7), Basophils (β_8), MCH (β_9), Eosinophils (β_{10}), and RDW (β_{12}). The model coefficients are far from 0 or HDIs include 0, but far from the mid-point.

- Intercept (β_0) has a mode of -4.48; its 95% HDI limits does not capture 0.
- Patient age quantile (β_1) has a mode of 0.0984; its 95% HDI limit includes 0 on the edge, but 99.5% of the values are bigger than 0. The results of β_1 suggest that a unit increase of patient age quantile increases the odds of a COVID-19 positive outcome by 1.1 times.
- Hemoglobin (β_2) has a mode of 0.659; its 95% HDI limit also captures 0 on the edge. The results of β_2 indicate that a unit increase in Hemoglobin increases the odds by 1.93 times. Across all variables, Hemoglobin has the highest effect on increasing the odds of a COVID-19 positive outcome as indicated in the cell highlighted in yellow.
- Platelets (β_3) has a mode of -0.3949; its 95% HDI limit includes 0 but not in the middle. The results of β_3 highlight that a unit increase of Platelets increases the odds of a COVID-19 negative outcome by 0.67 times.
- Leukocytes (β_7) has a mode of -1.91; its 95% HDI limit does not capture 0. A unit increase in Leukocytes increases the odds of getting a negative result by 6.76 as indicated in the cell highlighted in blue. Therefore, Leukocytes has the highest effect on increasing odds of a negative result.
- Basophils (β_8) has a mode of -0.154; its 95% HDI limit includes 0 but not in the middle. A unit increase of Basophils increases the odds of a COVID-19 positive outcome by 0.85 times.
- MCH (β_9) has a mode of -0.316; its 95% HDI limit includes 0 but not in the middle. A unit increase of MCH increases the odds of a COVID-19 positive outcome by 0.73 times.

- Eosinophils (β_{10}) has a mode of -1.18; its 95% HDI limit does not capture 0. A unit increase in Eosinophils increases the odds of getting a negative result as indicated in the cell highlighted in green. It is the second highest, which gives an increase of 3.25 in odds for a unit increase.
- RDW (β_{12}) has a mode of -0.294; its 95% HDI limit includes 0 but not in the middle. A unit increase of MCH increases the odds of a COVID-19 positive outcome by 0.75 times.

Meanwhile, **insignificant coefficients** were noted for Mean Platelet Volume (β_4), Lymphocytes (β_5), MCHC (β_6), and Monocytes (β_{11}).

- Mean platelet volume (β_4) has a mode of 0.122; zero is close to the middle of its 95% HDI limit.
- Lymphocytes (β_5) has a mode of -0.0959; captures zero in its 95% HDI limit, almost close to the middle.
- MCHC (β_6) has a mode of –0.195; captures zero in its 95% HDI limit, and relatively far from the edge.
- Monocytes (β_{11}) has a mode of 0.0463; zero is close to the middle of its 95% HDI limit.

Lastly, the **guess parameter** has a mode of 0.00. It is very low, indicating that the train data does not have many outliers.

Overall, there are 9 significant coefficients and 4 insignificant coefficients in the dataset. Of the 9 significant coefficients, Hemoglobin (β_2) has the highest impact on increasing the odds of a COVID-19 positive test outcome. In contrast, Leukocytes (β_7) and Eosinophils (β_{10}) have the highest effects on increasing the odds of a COVID-19 negative test outcome.

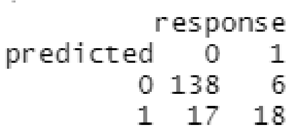
4.5 Predictive Check

Since the dependent variable of the regression model is highly imbalanced, the predictive check was conducted using AUC and accuracy metrics. For the different threshold probabilities, AUC and accuracy metrics have been obtained, and this is shown in Table 6. We obtained the highest AUC when the threshold is set to 0.2 and we obtained the highest accuracy when the threshold is set to 0.5.

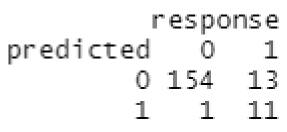
Figure 10 showcases the confusion matrices for the threshold probabilities 0.2 and 0.5. It is interesting to note that the threshold probability with highest AUC predicted 18 out of 24 positive cases, and threshold probability with highest accuracy predicted 154 out of 155 negative cases. Overall, from these confusion matrices, it can be said that threshold probability of 0.2 is better at predicting positive cases; similarly, threshold probability of 0.5 is better at predicting negative cases.

Table 6: AUC and Accuracy Metrics for Different Threshold Probabilities

Threshold	AUC	Accuracy
0.1	0.78293	0.776536
0.15	0.797581	0.832402
0.2	0.820161	0.871508
0.25	0.80578	0.877095
0.3	0.770565	0.877095
0.35	0.783468	0.899441
0.4	0.793145	0.916201
0.45	0.761156	0.921788
0.5	0.725941	0.921788
0.55	0.705108	0.916201
0.6	0.666667	0.910615
0.65	0.645833	0.905028
0.7	0.604167	0.893855
0.75	0.5625	0.882682
0.8	0.5625	0.882682
0.85	0.541667	0.877095
0.9	0	0
0.95	0	0



```
         response
predicted   0   1
        0 138   6
        1  17  18
```



```
         response
predicted   0   1
        0 154  13
        1   1  11
```

Figure 10:(Left) Confusion Matrix for Threshold Probability of 0.2 and (Right) Confusion Matrix for Threshold Probability of 0.5

4.6 Sensitivity Analysis

In this section, we are investigating the impact on the logistic regression model when different values are applied to prior variances of β parameters.

First, to make it more informative, we adjusted β_1 prior variance to 0.0001, while holding the remaining β prior variances constant at 4 and obtained the β_1 estimated value.

Then, we adjusted β_2 prior variance to 0.0001 while keeping the remaining β prior variances at 4 and obtained the estimated value for β_2.

For all the β 's, the above steps were iterated, and we obtained their estimated values.

The above-mentioned scenarios are shown in Table 7.

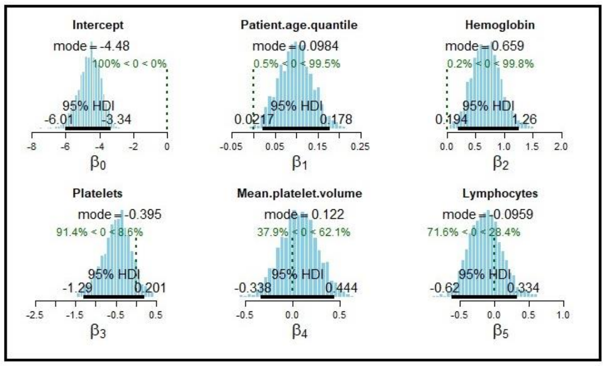
Table 7: Different Scenarios for Sensitivity Analysis

	Prior Variances of											
	beta[1]	beta[2]	beta[3]	beta[4]	beta[5]	beta[6]	beta[7]	beta[8]	beta[9]	beta[10]	beta[11]	beta[12]
Trial 1	0.0001	4	4	4	4	4	4	4	4	4	4	4
Trial 2	4	0.0001	4	4	4	4	4	4	4	4	4	4
Trial 3	4	4	0.0001	4	4	4	4	4	4	4	4	4
Trial 4	4	4	4	0.0001	4	4	4	4	4	4	4	4
Trial 5	4	4	4	4	0.0001	4	4	4	4	4	4	4
Trial 6	4	4	4	4	4	0.0001	4	4	4	4	4	4
Trial 7	4	4	4	4	4	4	0.0001	4	4	4	4	4
Trial 8	4	4	4	4	4	4	4	0.0001	4	4	4	4
Trial 9	4	4	4	4	4	4	4	4	0.0001	4	4	4
Trial 10	4	4	4	4	4	4	4	4	4	0.0001	4	4
Trial 11	4	4	4	4	4	4	4	4	4	4	0.0001	4
Trial 12	4	4	4	4	4	4	4	4	4	4	4	0.0001

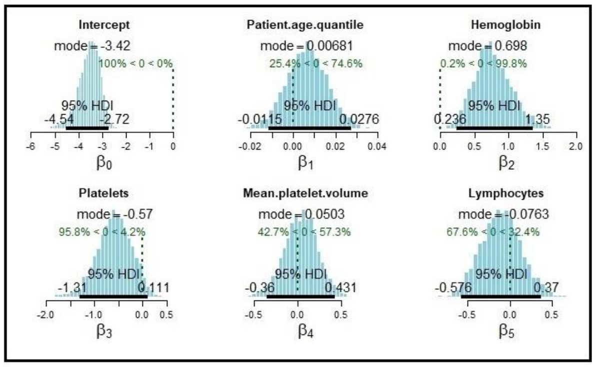
First, let's look at trial 1, refer to Figure 11, setting β_1 prior variance to 0.0001, β_1 estimate changed drastically from 0.0984 to 0.00681. β_1 significance changed drastically too. In trial 1, β1 was considered to be somewhat significant (HDI interval included 0, but somewhat far from the mid-point), whereas using the vague prior, it was considered significant.

β_2 , β_7 and β_{10} behave the same way as before. Significance of β_4, β_5, β_9, and β_{12} decreases in trial 1, and significance of β_3, β_6, β_8 and β_{11} increases in trial 1.

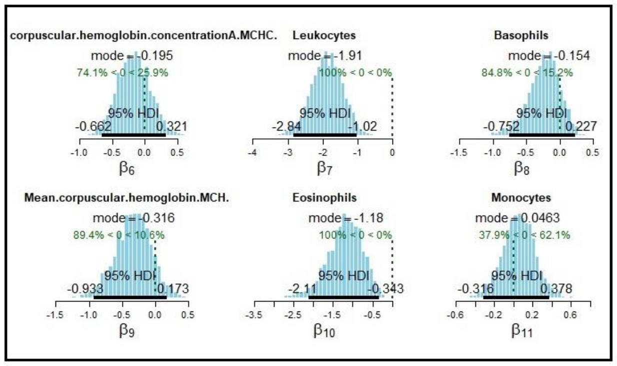
vague prior



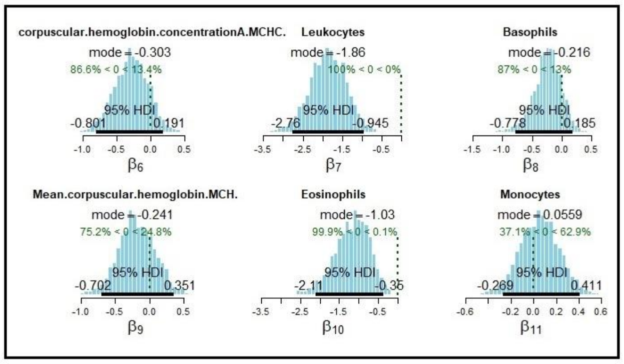
Beta1's variance=0.0001



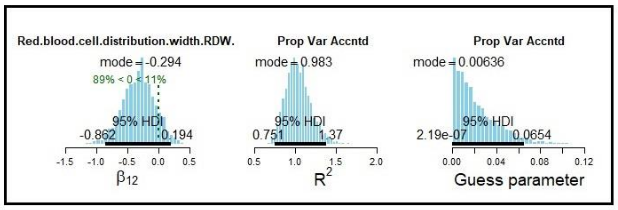
vague prior



Beta1's variance=0.0001



vague prior



Beta1's variance=0.0001

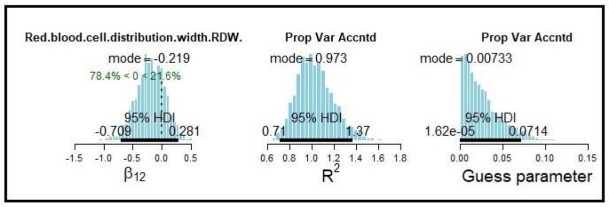


Figure 11: Posterior Distribution Comparison

The Bayesian estimates for β coefficients are shown in Table 8 for each trial. When the prior variance of a β is changed from 4 to 0.0001, the estimated value of that β changes drastically. In Table 8, these values are highlighted in yellow.

Table 8: Estimated Beta Coefficients for the 12 Trials

	Vague Prior	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Trial 11	Trial 12
beta[1]	0.0984	0.0068	0.0990	0.1002	0.0893	0.0889	0.0954	0.0979	0.0943	0.0876	0.0933	0.0917	0.0889
beta[2]	0.6586	0.6982	0.0026	0.6434	0.6447	0.7208	0.5885	0.6455	0.5580	0.6711	0.7017	0.7108	0.7681
beta[3]	-0.3949	-0.5700	-0.4050	-0.0011	-0.5080	-0.4821	-0.3474	-1.3857	-0.4171	-0.3847	-0.5650	-0.4462	-0.4115
beta[4]	0.1217	0.0503	0.0447	0.1749	0.0012	0.0670	0.0776	0.0099	0.0591	0.1078	0.0231	0.0455	0.1285
beta[5]	-0.0959	-0.0763	-0.2351	-0.2140	-0.1627	0.0006	-0.1160	0.3933	-0.1927	-0.0983	-0.3328	-0.1907	-0.1436
beta[6]	-0.1954	-0.3031	0.1002	-0.1453	-0.2048	-0.1329	0.0004	-0.1100	-0.1529	-0.2899	-0.1756	-0.1844	-0.1443
beta[7]	-1.911	-1.8620	-1.4734	-1.9874	-1.8425	-1.8078	-1.8053	-0.0022	-1.8367	-1.8306	-2.0931	-1.7763	-1.8332
beta[8]	-0.1539	-0.2163	-0.1212	-0.1979	-0.1539	-0.1902	-0.1752	-0.1533	-0.0020	-0.2054	-0.3896	-0.1865	-0.2195
beta[9]	-0.3163	-0.2408	-0.4111	-0.2490	-0.3497	-0.3103	-0.4038	-0.2935	-0.2617	-0.0002	-0.4336	-0.3177	-0.2668
beta[10]	-1.1775	-1.0324	-1.0713	-1.1144	-1.1038	-1.2030	-1.1472	-1.7752	-1.1629	-1.2116	-0.0021	-1.0996	-1.1598
beta[11]	0.0463	0.0559	0.0891	0.0797	0.0409	0.0603	0.0563	0.1989	0.0362	0.0953	0.0038	0.0013	0.0436
beta[12]	-0.294	-0.2190	-0.5196	-0.2156	-0.2786	-0.2401	-0.2715	-0.3204	-0.3625	-0.2242	-0.3010	-0.2873	-0.0020

From the above table, we realized that the Bayesian estimate of the coefficients is altered by the variance of its corresponding prior in the Bayesian logistics regression model. The change of one coefficient would impact other coefficient values in logistic regression. Odds will change subsequently.

The change in odds for a unit increase of β, for trial 1 is provided in Table 9 below.

A unit increase in patient age quantile lowers the increase in odds of getting COVID-19 from 1.1 times (when prior variance= 4) to 1.01 times in trial 1, where the prior variance of β1 is set to 0.0001.

A unit increase in Hemoglobin, raises the increase in odds of getting COVID-19 from 1.93 times to 2.01 times. Therefore, the effect of Hemoglobin on increasing the odds, is increased when the prior variance of β1 is reduced. Similarly, the effect of Lymphocytes, Leokocytes, MCH, Eosinophils, Monocytes, and RDW is increased.

The effect of Plateletes, Mean Platelet volume, MCHC, and Basophils on increasing the odds, is reduced when the prior variance of β1 is reduced.

Now, for each trial, let's compare the odds of getting a positive COVID-19 result. The evaluation of the increase in odds for the 12 trials is provided in Table 10.

Table 9: Comparison of the Odds

	Vague Prior	Exp(mode) for vague prior	Trial 1	Exp(mode) for Trial 1
beta[1]	0.0984	1.1034	0.0068	1.0068
beta[2]	0.6586	1.9321	0.6982	2.0100
beta[3]	-0.3949	0.6737	-0.5700	0.5655
beta[4]	0.1217	1.1294	0.0503	1.0516
beta[5]	-0.0959	0.9086	-0.0763	0.9265
beta[6]	-0.1954	0.8225	-0.3031	0.7385
beta[7]	-1.9110	0.1479	-1.8620	0.1554
beta[8]	-0.1539	0.8574	-0.2163	0.8055
beta[9]	-0.3163	0.7288	-0.2408	0.7860
beta[10]	-1.1775	0.3080	-1.0324	0.3562
beta[11]	0.0463	1.0474	0.0559	1.0575
beta[12]	-0.2940	0.7453	-0.2190	0.8033

Table 10: Comparison of the Increase in Odds for the 12 trials

Trial	Parameters	Prior Variance 0.0001	Prior Variance 4
		Exp(Mode)	Exp(Mode)
Trial 1	beta[1]	1.0068	1.1034
Trial 2	beta[2]	1.0026	1.9322
Trial 3	beta[3]	0.9989	0.6738
Trial 4	beta[4]	1.0012	1.1295
Trial 5	beta[5]	1.0006	0.9086
Trial 6	beta[6]	1.0004	0.8225
Trial 7	beta[7]	0.9978	0.1479
Trial 8	beta[8]	0.9980	0.8573
Trial 9	beta[9]	0.9998	0.7289
Trial 10	beta[10]	0.9979	0.3080
Trial 11	beta[11]	1.0013	1.0474
Trial 12	beta[12]	0.9980	0.7453

The above table can be interpreted as follows:

- Effect of β2 in Trial 2 → The effect of Hemoglobin in the odds of getting COVID-19 is reduced (Increase in odds of getting COVID-19 reduced from 1.93 to 1.002).
- Effect of β3 in Trial 3 → The effect of Plateletes in the odds of getting COVID-19 is increased. (Increase in odds of getting COVID-19 increased from 0.674 to 0.999).
- Effect of Mean platelet volume and Monocytes reduced for Trial 4 and Trial 11, respectively.
- Effect of Lymphocytes, MCHC, Leukocytes, Basophiles, MCH, Eosinophiles and RDW increased for Trials 5, 6, 7, 8, 9,10 and 12 respectively.

Now let’s look at the threshold probabilities and best AUC and accuracy metrics for the 12 trials.

Table 11: Threshold probabilities with best AUC and Accuracy metrics for the 12 trials

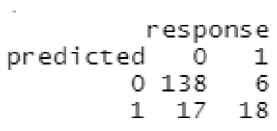
	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Trial 11	Trial 12
Threshold with best AUC	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.25	0.15	0.15	0.15
Threshold with best Accuracy	0.4	0.4	0.4	0.4	0.4	0.4	0.45	0.4	0.45	0.6	0.4	0.4

Table 12: Best AUC and Accuracy metrics for the 12 trials

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Trial 11	Trial 12
Best AUC	0.8038	0.8234	0.8234	0.8058	0.8058	0.8137	0.8137	0.8234	0.8202	0.8199	0.8008	0.8008
Best Accuracy	0.9162	0.9274	0.9274	0.9218	0.9218	0.9274	0.9218	0.9106	0.9274	0.9050	0.9218	0.9218

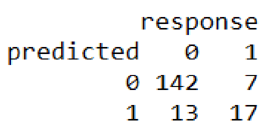
From Table 12, we can see that trial 2, trial 3, and trial 8 have the best AUC value (0.8234), and it is an improvement from 0.8202 (refers to Table 6) , and trial 2, trial 3, trial 6, and trial 9 have the best accuracy metric (0.9274), which is an improvement from 0.9218 (refer to Table 6).

Vague Prior



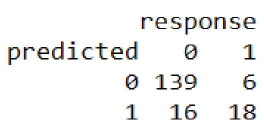
```
         response
predicted   0   1
        0 138   6
        1  17  18
```

Trial 2 & 3



```
         response
predicted   0   1
        0 142   7
        1  13  17
```

Trial 8



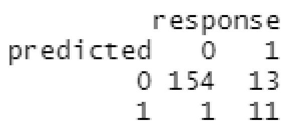
```
         response
predicted   0   1
        0 139   6
        1  16  18
```

Figure 12: Comparison of Confusion Matrices with Best AUC Values

Figure 12 showcases the confusion matrices we obtained for the best AUC values. It is important to note that trials 2 and 3 predicted 142 out of 155 negative cases in the test data, which means that the possibility of correctly identifying the negative cases increases when you change the prior variance of β2 or β3 from 4 to 0.0001, and setting the threshold probability at 0.2.

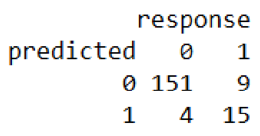
The logistic regression model with vague priors for β parameters is better at predicting the positive cases of COVID-19 in the test data, at threshold probability 0.2.

Vague Prior



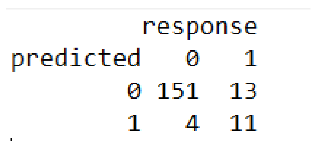
```
         response
predicted   0   1
        0 154  13
        1   1  11
```

Trials 2,3 & 6



```
         response
predicted   0   1
        0 151   9
        1   4  15
```

Trial 9



```
         response
predicted   0   1
        0 151  13
        1   4  11
```

Figure 13: Comparison of Confusion Matrices with Best Accuracy Values

Figure 13 showcases the confusion matrices we obtained for the best accuracy values. It is interesting to note that Trials 2,3 and 6 predicted 15 out of 24 positive cases in the test data, which means that the possibility of correctly identifying the positive cases increases when you change the prior variances of either β2 , β3 or β6 from 4 to 0.0001, and setting the threshold probability at 0.4 .

The logistic regression model with vague priors for β parameters is better at predicting the negative cases of COVID-19 in the test data, at threshold probability 0.5.

5. Hierarchical Model

5.1 Mathematical Model and JAGS Diagram

A hierarchical model can be used to analyse further the likelihood of a positive class distribution respective to any categorical attribute. In our case, it would be interesting to explore the relevance of age in the likelihood of positive test results and observe the impact of prior information on this relevance with Bayesian analysis.

Age has been classified into 20 quantiles in the given dataset. We would further group these quantiles as below, such that each age group would represent 25% of the age tier:

Table 13: Grouping of Age Quantile

Age Quantile	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
% in each quantile	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%
Total % in each group	25%					25%					25%					25%				
Group	Age Group 1					Age Group 2					Age Group 3					Age Group 4				

The frequency count and proportion of patients in each age group with covid test results were distributed as shown in Table 14 and Table 15:

Table 14: Frequency table of Different Age Groups

	Age Group 1	Age Group 2	Age Group 3	Age Group 4	Total
0 (negative)	98	115	137	167	517
1 (positive)	3	12	32	34	81
Total	101	127	169	201	598

Table 15: Proportion table of Different Age Groups

	Age Group 1	Age Group 2	Age Group 3	Age Group 4	Total
0 (negative)	0.97030	0.90551	0.81065	0.83085	0.86455
1 (positive)	0.02970	0.09449	0.18935	0.16915	0.13545
Total	1.00000	1.00000	1.00000	1.00000	1.00000

The hierarchical model can be formulated mathematically as provided below. Each equation is labelled from numbers 1 to 5 and are explained in detail. These then translate into the JAGS model diagram in Figure 14.

$$\boxed{Y_{i|s,c} \sim Bernoulli\ (\theta_{s|c})}$$

Where $Y_{i|s,c}$ is the test instance (i) within each patient (subject s), and age group (category c), $\theta_{s|c}$ denotes the probability of being positive for the patient in his/her respective age group. ①

We then induced a beta prior on $\theta_{s|c}$, which would be reparametrized with mode ω_c and concentration κ_c as follows:

$$\boxed{\theta_{s|c} \sim dbeta\ (\omega_c(\kappa_c - 2) + 1, (1 - \omega_c)(\kappa_c - 2) + 1)}$$

This could be read as the biases of patients (subject s) within age group (category c) are assumed to be distributed as a beta density with mode ω_c and concentration κ_c. Each Age group has its own modal bias of ω_c, from which all patient biases in the age group are assumed to be drawn. ②

The model assumes that all the age group modes ω_c come from a higher level beta distribution that describes the variation across age groups. The modal bias across age groups is denoted ω (without subscript), and the concentration of the age group biases is denoted κ (without subscript). ③

For simplicity, the age group concentration κ_c is also fixed by the same prior constants of κ, and do not mutually inform others for different age groups. ④

In order to estimate ω (without subscript) and κ (without subscript), we need to specify prior settings for the overall positive testing probability. ω (without subscript) can be estimated using a beta distribution, as the range of the modal bias is within [0, 1]. κ (without subscript) can be estimated using a gamma distribution, as the range of concentration is between [0, ∞). ⑤

As mentioned, all the above mathematical formulations can be better illustrated by the JAGS model diagram with its corresponding numeric symbols in Figure 14.

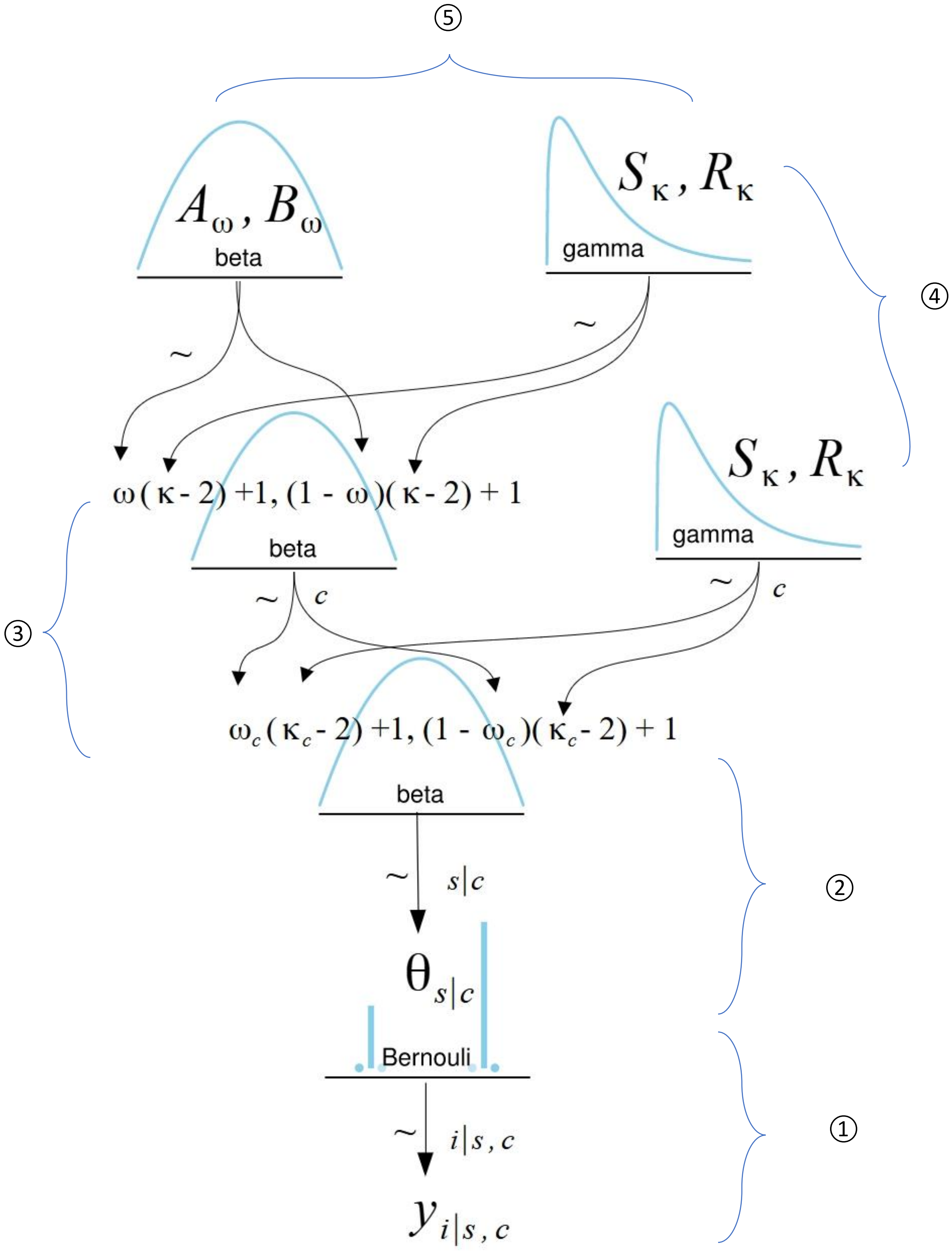


Figure 14: JAGS Model Diagram for the Hierarchical Model

5.2 Prior Information

As there is "no expert knowledge" on the probability of being positive in the test for each age group or the entire sampled population, we would initially set ω with a vague prior of uniform distribution (dbeta(1.0 , 1.0), where A_ω=1.0, B_ω=1.0, ⑤ in the JAGS model diagram, and that is equivalent to Mode_ω = 0, $\text{Conentration}_\omega$ =2).

To set a vague prior for κ, we would set a gamma distribution with a small value of mode and board standard deviation (Mode_κ=1, S.D_κ=100 ➔ dgamma(1.01005 , 0.01005012), where S_κ (Shape of Kappa)= 1.01005 and R_κ (Scale of Kappa)=0.01005012, as shown in ④ in the JAGS model diagram) to indicate no information on how far $\theta_{s|c}$ is away from ω_c.

5.3 Posterior Distributions and Interpretation

After running MCMC in JAGS (refer to the coding in [Appendix B2]) and passing through all the diagnostic checks (refer to section 6.5 for further details), we could obtain the following posterior distributions:

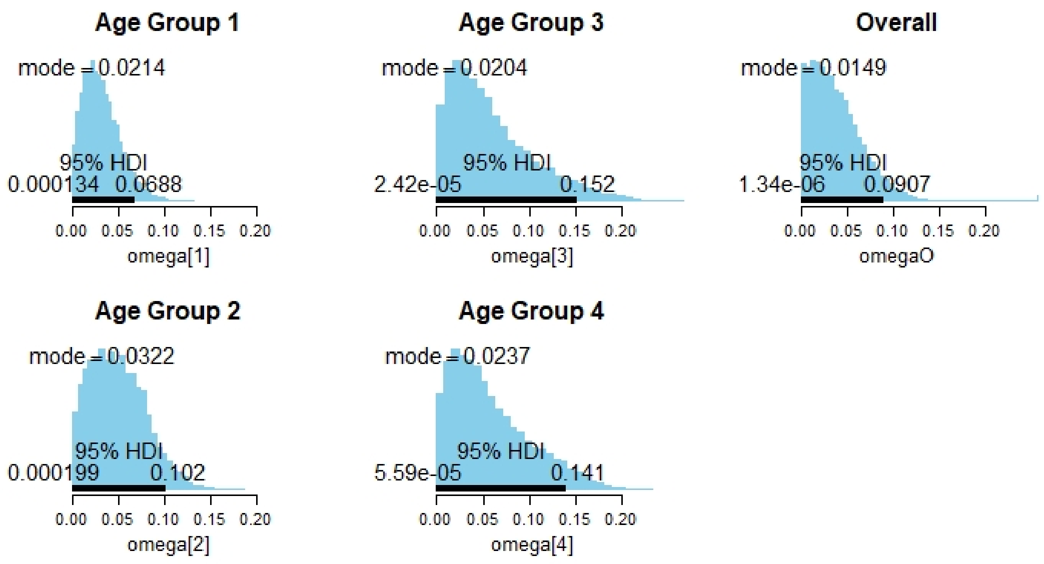


Figure 15: Marginal Prior Distributions of Positive Probability for Different Age Groups

Refer to Figure 15:

1. Using the mode of the posterior distribution, the positive probability of age group 1 is estimated to be 0.0214, i.e. $\tilde{\omega}_{age\_group\_1}$ = 0.0214, similarly, $\tilde{\omega}_{age\_group\_2}$ = 0.0322, $\tilde{\omega}_{age\_group\_3}$ = 0.0204, $\tilde{\omega}_{age\_group\_4}$ = 0.0237, $\tilde{\omega}_{overall}$ = 0.0149. $\tilde{\omega}_{age\_group\_2}$ has the highest estimated positive probability.
2. 97.5% (since we are only counting positive distribution, 95% HDI is indicating 97.5% of the distribution) of the patients in age group 1 have less than 0.0688 positive probability, which is the lowest among all age groups. A similar reading method applies to other age groups. 97.5% of the patients in each group with positive probability would go with ascending order as follows:
   Age group 1 < Age group 2 < Age group 4 < Age group 3
3. Age group 3 has the longest right tail while age group 1 has the shortest right tail, which indicates that patients in age group 1 are less likely to have a high positive probability compare

to age group 3. Some patients in the age group 3 would have a positive probability higher than 0.20.

4. 97.5% of the patients have less than 0.0907 positive probability in $\omega_{overall}$, which is higher than group 1, but lower than group 2, 4, and 3.

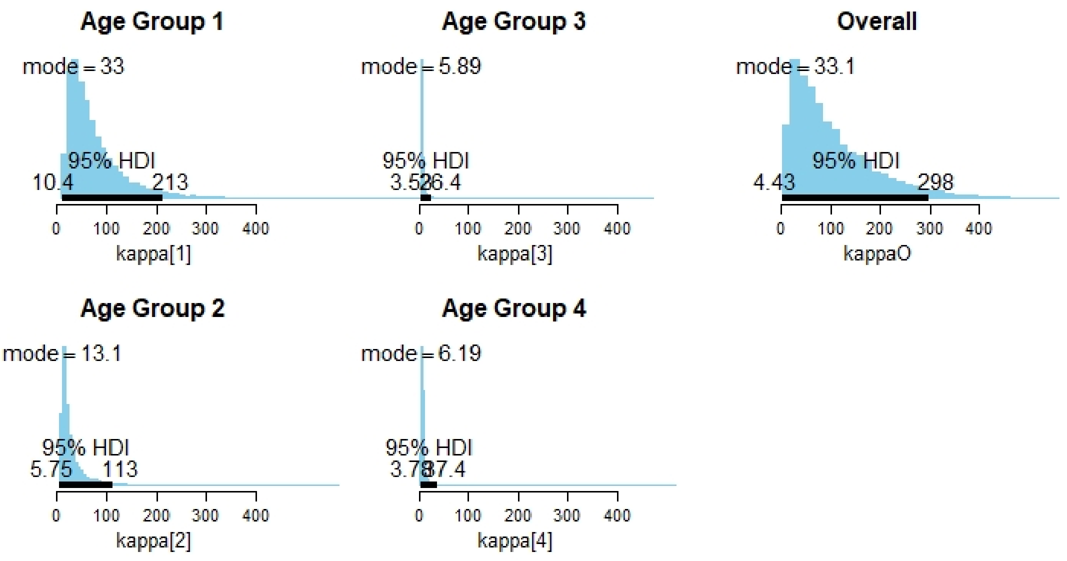


Figure 16: Marginal Prior Distributions of the Concentration of Positive Probability for Different Age Groups

Refer to Figure 16:

1. Using the mode of the posterior distribution, the concentration of positive probability of age group 1 is estimated to be 33, i.e. $\tilde{\kappa}_{age\_group\_1}$ = 33, similarly, $\tilde{\kappa}_{age\_group\_2}$ = 13.1, $\tilde{\kappa}_{age\_group\_3}$ = 5.89, $\tilde{\kappa}_{age\_group\_4}$ = 6.19, $\tilde{\kappa}_{overall}$ = 33.1.
2. $\tilde{\kappa}_{age\_group\_1}$ has the highest estimated value (which implies $\theta_{s|c}$ is closer to ω_c in age group 1), but the largest variation as compared to other age groups. $\tilde{\kappa}_{overall}$ also has similarly high estimated value and high variation. (which implies $\theta_{s|c}$ is close to $\omega_{overall}$, but there is a high variation)

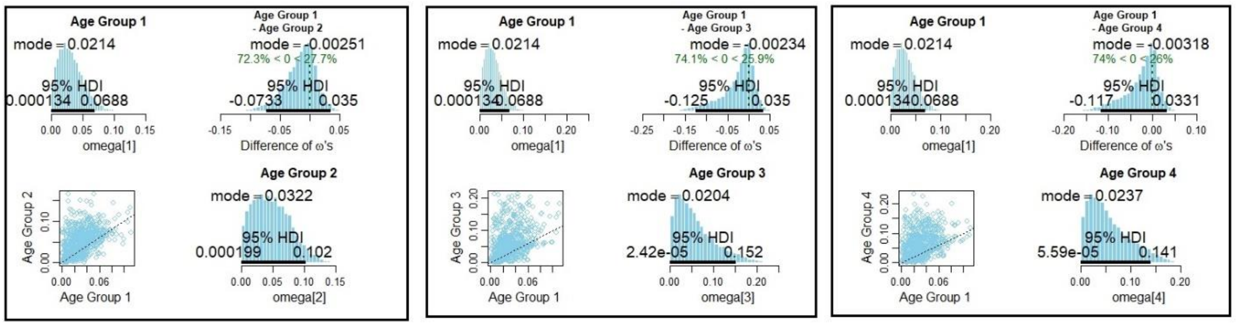


Figure 17: Posterior Distributions of the Positive Probability of Age Group 1-4, and the Difference of Age Group 1 vs Age Group 2, Age Group 1 vs Age Group 3, Age Group 1 vs Age Group 4

Refer to Figure 17: In the top right plot of the comparison diagram between Age Group 1 vs Age Group 2, the HDI of the difference of ω's plot capture 0, 27.7% of the difference of ω's will be greater than 0, 72.3% of the difference of ω's will be less than 0, there might be a difference, but we are not super confident in that. A similar interpretation applies to the comparison between Age Group 1 vs Age Group 3 and Age Group 1 vs Age Group 4.

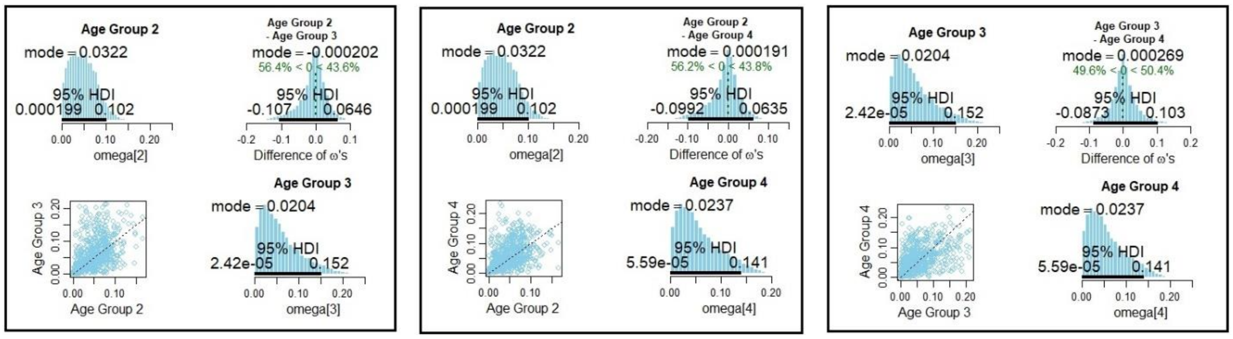


Figure 18: Posterior Distributions of the Positive Probability of Age Group 2-4, and Difference of Age Group 2 vs Age Group 3, Age Group 2 vs Age Group 4, Age Group 3 vs Age Group 4

Refer to Figure 18: In the top-right plot of the comparison diagram between Age Group 2 vs Age Group 3, the HDI of the difference of ω's plot capture 0 right in the middle of the distribution. We are confident to say there would be no difference of ω between Age group 2 and Age group 3. A similar interpretation applies to the comparison between Age Group 2 vs Age Group 4 and Age Group 3 vs Age Group 4.

5.4 Sensitivity Analysis

In order to observe the impact of changing prior distribution on posterior distribution, we have performed the following different trials with different Mode$_\omega$, Conentration$_\omega$, Mode$_\kappa$ and S.D$_\kappa$.

Horizontally, we would like to observe the effect of the beta distribution of Omega from non-informative to informative(changing Conentration$_\omega$ from 2 (trial 1) to 100(trial 2), while Mode$_\omega$ remains to 0) and then the effect of location change (from 0 to 0.1 and 0.5).

Vertically, we would like to observe the effect of gamma distribution of Kappa from non-informative to informative(changing S.D$_\kappa$ from 100 (trial 1) to 2(trial 5), while Mode$_\kappa$ remains to 1) and then the effect of location change (from 1 to 10 and 40).

Table 16: Trials for the Sensitivity Analysis

Omega / Kappa	dbeta(1.0 , 1.0) vague Modeω = 0, Conentrationω =2	Modeω = 0, Conentrationω = 100	Modeω = 0.1, Conentrationω = 100	Modeω = 0.5, Conentrationω = 100
Modeκ=1, S.Dκ =100 (vague)	trial 1	trial 2	trial 3	trial 4
Modeκ=1, S.Dκ =2	trial 5	trial 6	trial 7	trial 8
Modeκ=10, S.Dκ =2	trial 9	trial 10	trial 11	trial 12
Modeκ=40, S.Dκ =2	trial 13	trial 14	trial 15	trial 16

We have observed some interesting patterns, first let's compare the posterior distribution of trial 1 (vague prior) with trial 16 (Mode$_\omega$=0.5 Conentration$_\omega$ =100, would imply prior knowledge of 0.5 positive probability with high degree of belief; Mode$_\kappa$ = 40 and S.D$_\kappa$ =2, would imply the prior information of $\theta_{s|c}$ is closer to ω_c (as compare to Mode$_\kappa$ =1) with high degree of belief), if we have more time to explore, we would go for higher value of Mode_ω and Mode_κ to see if there is any further difference. As for now, these 2 trials would be at the extreme ends of all our trials.

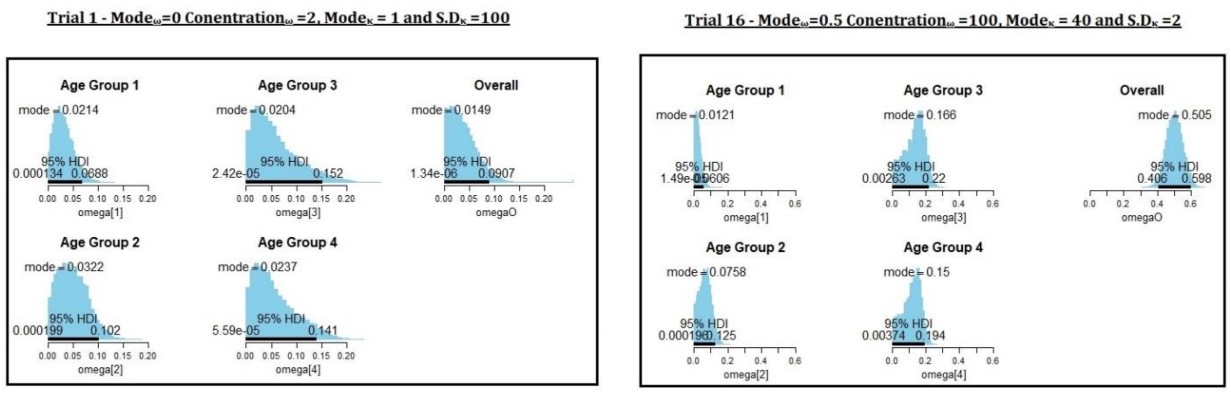


Figure 19: Comparison of Marginal Prior Distributions of Positive Probability for Different Age Groups (Trial 1 vs Trial 16)

Refer to Figure 19:

1. $\tilde{\omega}_{age\_group\_1}$ has dropped from 0.0214 to 0.0121, $\tilde{\omega}_{age\_group\_2}$ has increased from 0.0322 to 0.0758, $\tilde{\omega}_{age\_group\_3}$ has increased from 0.0204 to 0.166, $\tilde{\omega}_{age\_group\_4}$ has increased from 0.0237 to 0.15, $\tilde{\omega}_{overall}$ has increased from 0.0149 to 0.0505. $\tilde{\omega}$ for all age groups in trial 16 would be estimated much closer to the proportion of the dataset (closer to likelihood) compare to trial1. $\tilde{\omega}_{overall}$ is close to Mode_ω (which is set to 0.5) and further apart from $\tilde{\omega}$ for all age groups in trial 16.
2. The upper limit of positive probability of 97.5% of the patients in the different age group has changed as below:

Table 17: Comparison of the Upper Limit Of HDI (Prior Distributions Of Positive Probability) in Trial1 and Trial 16 For Different Age Group

	Trial 1	Trial 16
Age Group 1	0.0688	0.0606
Age Group 2	0.102	0.125
Age Group 3	0.152	0.22
Age Group 4	0.141	0.194

The order still remains the same as follows:

Age group 1 < Age group 2 < Age group 4 < Age group 3

Only Age Group 1 has gone down; all the other age groups have gone up.

3. The scale of the probability axis has changed, but Age group 3 still has the longest right tail while age group 1 has the shortest right tail, tail of Age group 3 has extended beyond 0.3, while the tail of Age group 1 has stretched from less than 0.15 to over 0.2 in trial 16.
4. In trial 1, 97.5% of the patients have less than 0.0907 positive probability in $\omega_{overall}$; In trial 16, 95% of the patients have positive probability between 0.406 and 0.598 in $\omega_{overall}$, both of these ranges are close to Mode_ω being set for the respective trial.

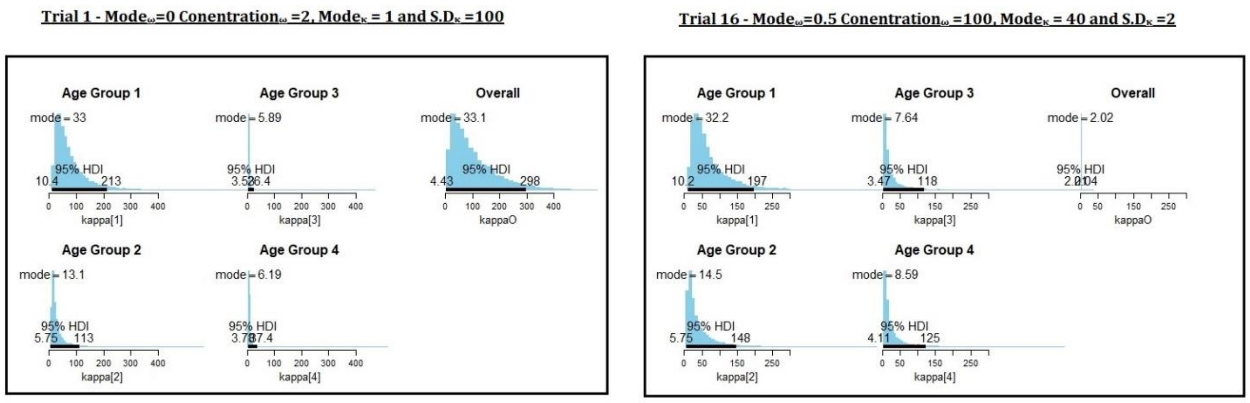


Figure 20: Comparison of Marginal Prior Distributions of the Concentration of Positive Probability for Different Age Groups (Trial 1 vs Trial 16)

Refer to Figure 20:

1. $\tilde{\kappa}$ for all age group have a small increase (except $\tilde{\kappa}_{age\_group\_1}$ has slightly decreased), but variations have certainly increased.
2. In trial 1, $\tilde{\kappa}_{overall}$ =33.1, while in trial 16, $\tilde{\kappa}_{overall}$ =2.02, which implies $\theta_{s|c}$ is much close to $\omega_{overall}$ (0.0149) in trial 1 compare to trial 16 (0.505), but there is a high variation in trial 1 (i.e. not all $\theta_{s|c}$ is close to 0.0149) and extremely low variation in trial 16 (i.e. all $\theta_{s|c}$ is far away from 0.505).

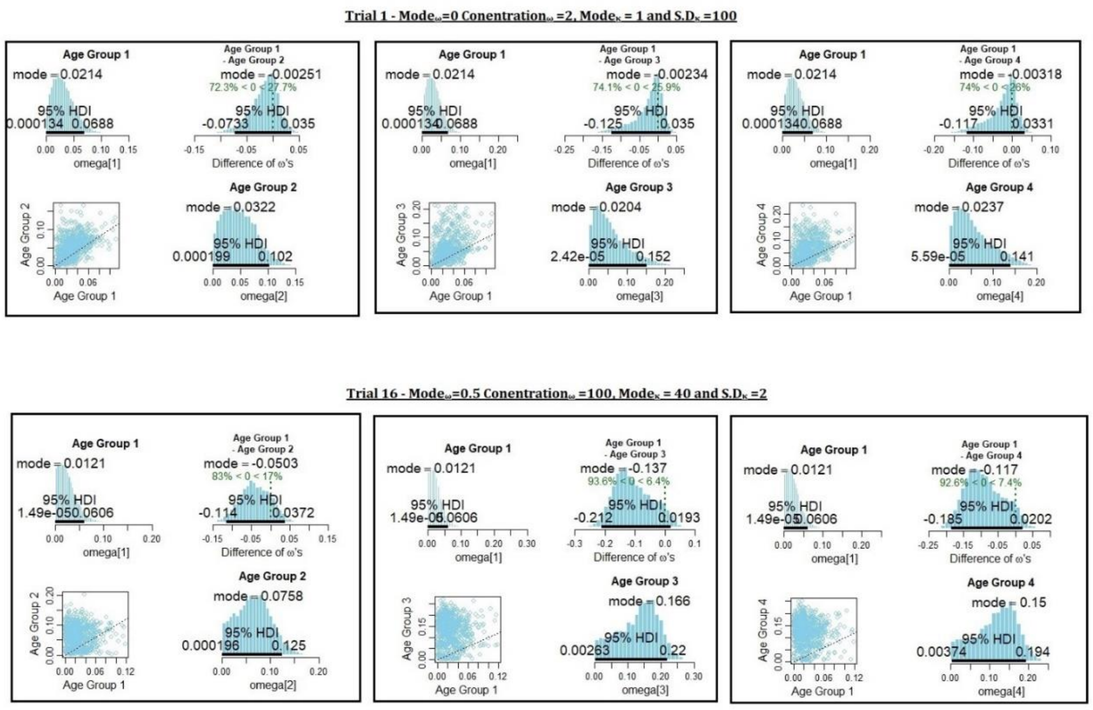


Figure 21: Comparison of Trial 1 vs Trial 16 in the Posterior Distributions of the Positive Probability of Age Group 1-4, and Difference of Age Group 1 vs Age Group 2, Age Group 1 vs Age Group 3, Age Group 1 vs Age Group 4

Refer to Figure 21: In trial 16, the HDI of the difference of ω's still capture 0 for all the top right plot (Age Group 1 vs all other age groups), however, the zero-compare-lines are leaning more toward the HDI upper limits compare to trial 1. (In Age Group 1 vs Age Group 2, the difference of ω's which is greater than 0 has dropped from 27.7% to 17%, in Age Group 1 vs Age Group 3, has dropped from 25.9% to 6.4%, and in Age Group 1 vs Age Group 4, has dropped from 26% to 7.4%), thus we are more confident to address there would be a difference in Age Group 1 vs Age Group 3 and Age Group 1 vs Age Group 4 in trial 16.

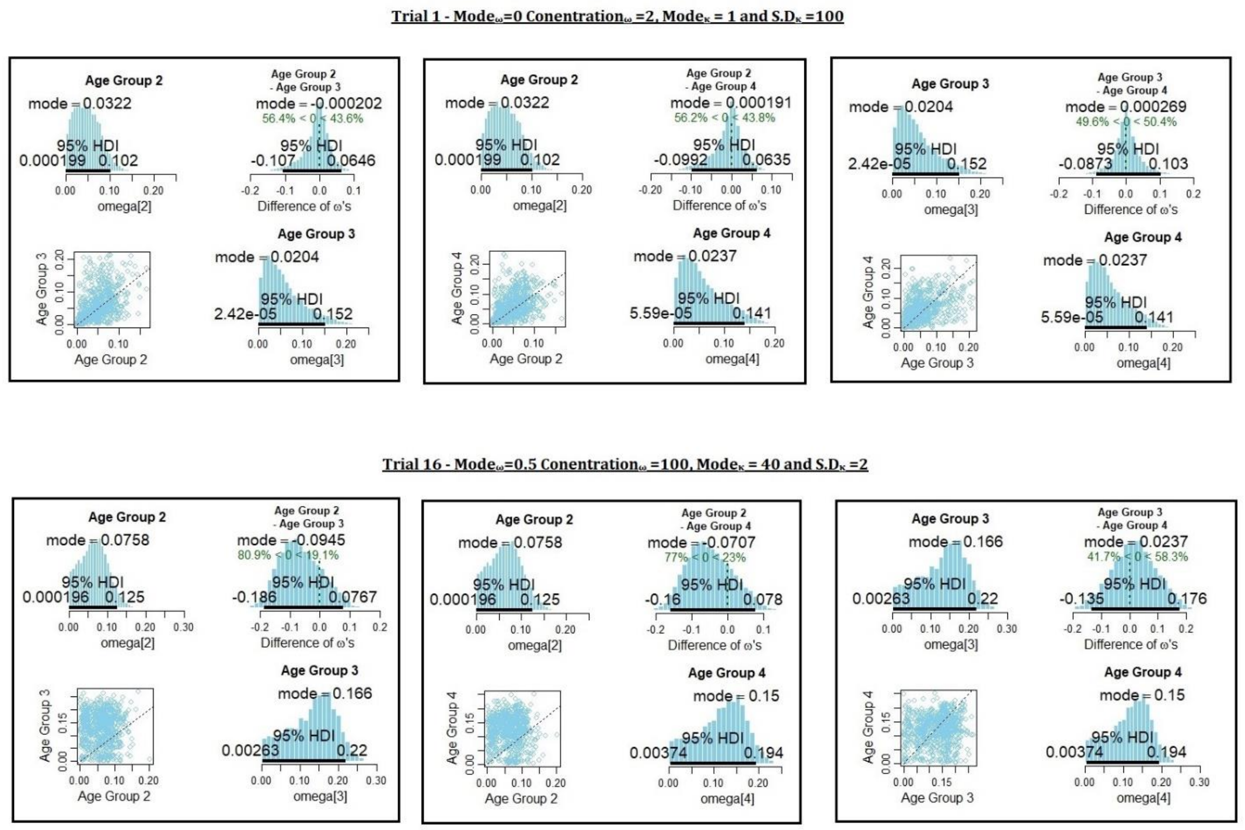


Figure 22: Comparison of Trial 1 vs Trial 16 in the Posterior Distributions of the Positive Probability of Age Group 2-4, and Difference of Age Group 2 vs Age Group 3, Age Group 2 vs Age Group 4, Age Group 3 vs Age Group 4

Refer to Figure 22:

1) In Age Group 2 vs Age Group 3, the difference of ω's which is greater than 0 has dropped from 43.6% to 19.1%, in Age Group 2 vs Age Group 4, has dropped from 43.9% to 23%, the figures 19.1% and 23% both indicate there might be a difference between these groups, but we are not super confident about that. Thus, Age Group 2 vs Age Group 3 and Age Group 2 vs Age Group 4 have turned from confident with no difference to not super confident with the difference from trial 1 to trial 16.
2) For Age Group 3 vs Age Group 4, the HDI of the difference of ω's plot capture 0 right in the middle of the distribution in both trials, thus we are confident there would be no difference of ω between Age group 3 and Age group 4 in trial 1 and 16.

So now, the question comes up: Would the distribution of $\tilde{\theta}_{s|c}$ (where s=patient, c=age group) be impacted by prior distribution? Could this be illustrated?

For example, each individual patient would have his/her own posterior distribution on positive probability ($\theta_{s|c}$). Refer to Figure 23 using the mode of the posterior distribution, the positive probability of patient 8 is estimated to be 0.0572, i.e. $\tilde{\theta}_{patient\_8|age\_group\_2} = 0.0572$, similarly, $\tilde{\theta}_{patient\_9|age\_group\_3} = 0.19$, would this be changed with different prior settings?

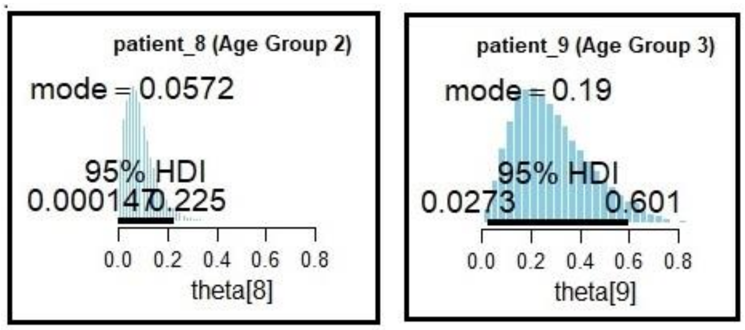


Figure 23: Posterior Distributions of Positive Probability of Patient 8 and 9 in Trial 1

It would be compulsive to see the impact of different prior settings on $\tilde{\theta}_{patient|age\_group}$.

Table 18: Trials for Sensitivity Analysis

Omega Kappa	dbeta(1.0 , 1.0) vague Modeω = 0, Conentrationω =2	Modeω = 0, Conentrationω = 100	Modeω = 0.1, Conentrationω = 100	Modeω = 0.5, Conentrationω = 100
Modeκ=1, S.Dκ =100 (vague)	trial 1	trial 2	trial 3	trial 4
Modeκ=1, S.Dκ =2	trial 5	trial 6	trial 7	trial 8
Modeκ=10, S.Dκ =2	trial 9	trial 10	trial 11	trial 12
Modeκ=40, S.Dκ =2	trial 13	trial 14	trial 15	trial 16

Refer to Table 18, using trial 1 as control, we would be comparing all the trials highlighted in blue in Figure 24 and comparing all the trials highlighted in orange in Figure 25 to observe the change in $\tilde{\theta}_{patient\_9|age\_group}$ associated with the covid test result.

Refer to Figure 24 , in trial 1, we could observe that Age Group 3 and 4 with actual positive results are far away from other groups. Age Group 1 with actual negative results has low variation with low positive probability. If we set the probability threshold to 0.11, we could correctly classify all the actual negative test results, but Age Group 1 and 2 with actual positive results would be misclassified. However, that is already the optimal threshold in this trial.

In trial 2, all the settings would be the same as trial 1, except Conentration$_{\omega}$ has increased to 100. From the plot, it shows that all age groups with actual negative results would shift to the lower end and the variations also decrease, they are becoming highly concentrated. If we set the probability threshold to 0.06 (the optimal threshold), we could correctly classify all the actual negative test results, and only Age Group 1 with actual positive results would be misclassified in this trial.

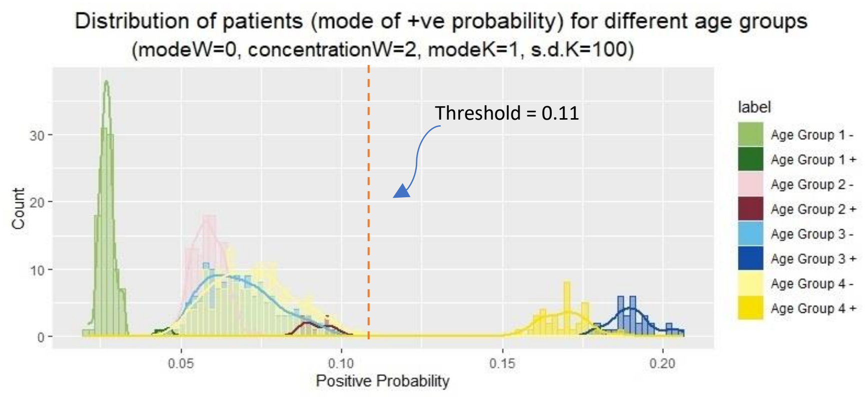
In trial 3 and 4, all the settings would be the same as trial 2, except Mode$_{\omega}$ (location) is changed to 0.1 and 0.5, respectively. From the plots, we could observe that age groups 2, 3, and 4 with actual negative results would have a higher impact on the change. These groups would shift towards the higher end of the axis, in trial 4, the shift for age group 3 and 4 with actual negative results is more obvious, and the distribution of age group 3 with negative results would overlap with age group 4 with positive results.

Thus, setting the probability threshold to optimal thresholds, in trial 3 (optimal at 0.15), only 1 patient in age group 3 with actual negative result would be misclassified, but in trial 4 (optimal at 0.18), 4 patients in age group 3 with actual negative result would be misclassified, and 26 (41 – 15 =26, 15 are the patients with negative result in age group 1 and 2) patients in age group 4 with positive results would be misclassified in trial 4, while no patient in age group 4 with positive results would be misclassified.

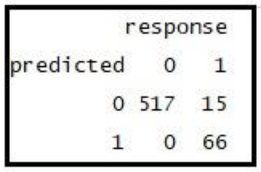
Refer to Figure 25, in trial 5, all the settings would be the same as trial 1, except S.D$_\kappa$ has gone down to 2 (more concentrated), from the plot, it shows that age group 3 and 4 with actual negative results would again shift towards to the higher end, the distribution of age group 3 with negative results would overlap with age group 4 with positive results. Thus misclassification of age group 3 with actual negative result and age group 4 with positive results would be introduced.

In trial 9 and 13, all the settings would be the same as trial 5, except Mode$_\kappa$ (location) is changed to 10 and 40, respectively. There would be a slight change in the variations (shown from the shape of the density curves), but overall the distributions for trial 5, 9 and 13 are very similar.

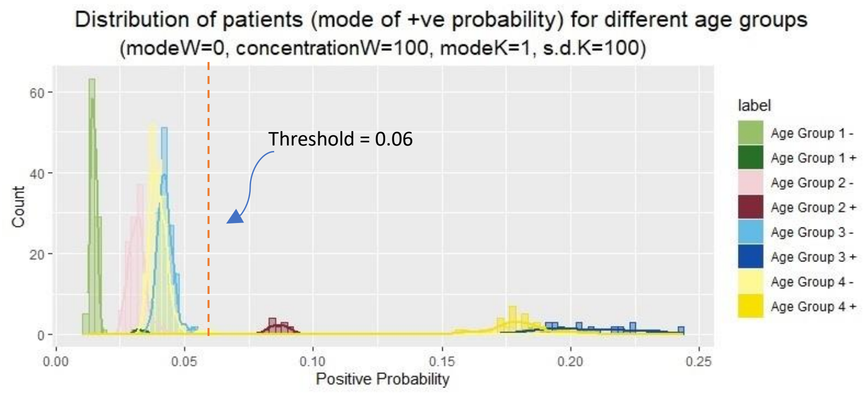
Thus, from Figure 24 and Figure 25, Conentration$_\omega$, S.D$_\kappa$ and Mode$_\omega$ of prior distribution will have a high impact on $\tilde{\theta}_{\text{patient|age\_group,}}$ while Mode$_\kappa$ has lower impact in our hierarchical model.



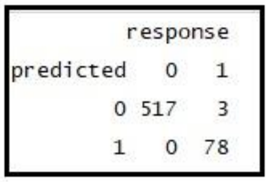
Trial 1 - Mode$_\omega$=0 Conentration$_\omega$=2; Mode$_\kappa$ = 1 and S.D$_\kappa$ = 100



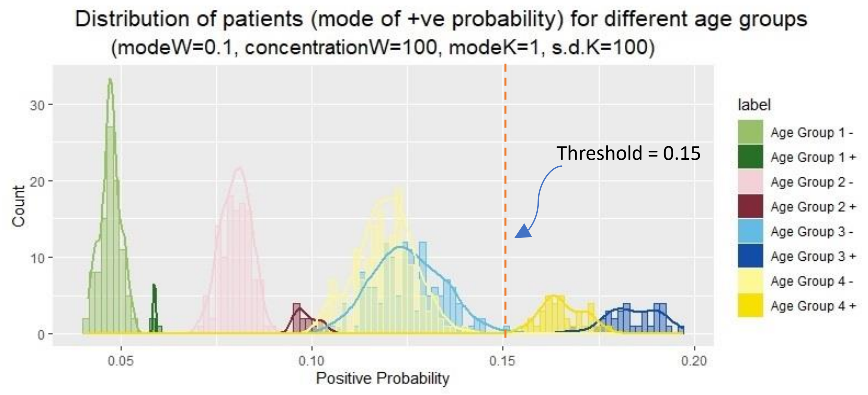
```
          response
predicted   0   1
        0 517  15
        1   0  66
```



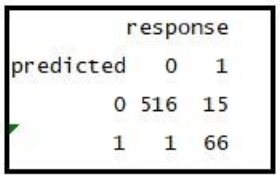
Trial 2 - Mode$_\omega$=0 Conentration$_\omega$=100; Mode$_\kappa$ = 1 and S.D$_\kappa$ = 100



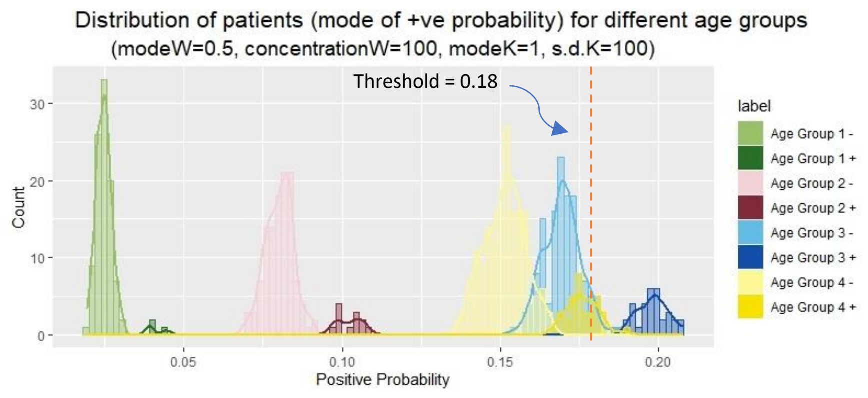
```
          response
predicted   0   1
        0 517   3
        1   0  78
```



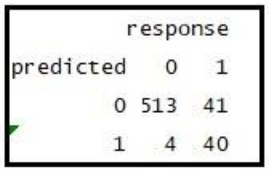
Trial 3 - Mode$_\omega$=0.1 Conentration$_\omega$=100; Mode$_\kappa$ = 1 and S.D$_\kappa$ = 100



```
          response
predicted   0   1
        0 516  15
        1   1  66
```

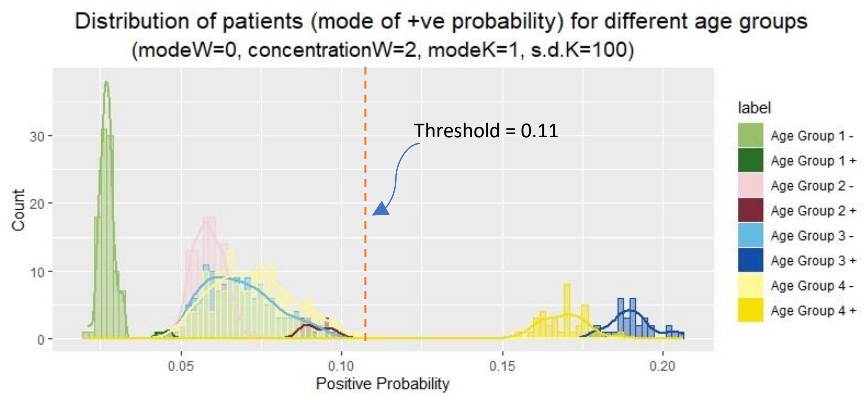


Trial 4 - Mode$_\omega$=0.5 Conentration$_\omega$=100; Mode$_\kappa$ = 1 and S.D$_\kappa$ = 100

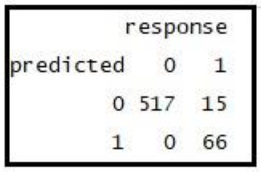


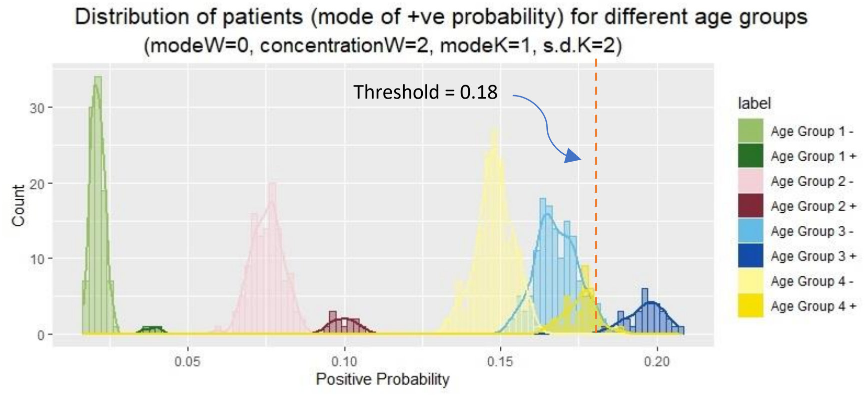
```
          response
predicted   0   1
        0 513  41
        1   4  40
```

Figure 24: Comparison of Probability Distribution with Different Omega Priors

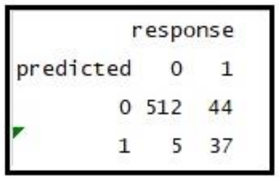


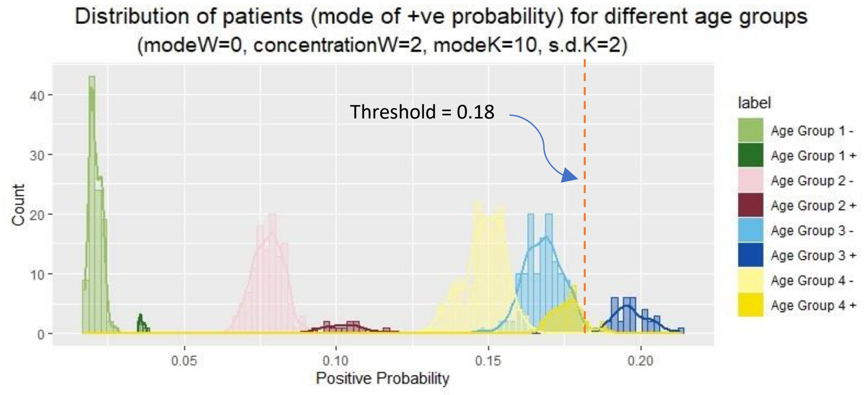
Trial 1 - Mode$_\omega$=0 Conentration$_\omega$=2; Mode$_\kappa$ = 1 and S.D$_\kappa$ = 100


```
         response
predicted   0   1
        0 517  15
        1   0  66
```

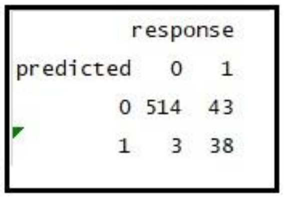


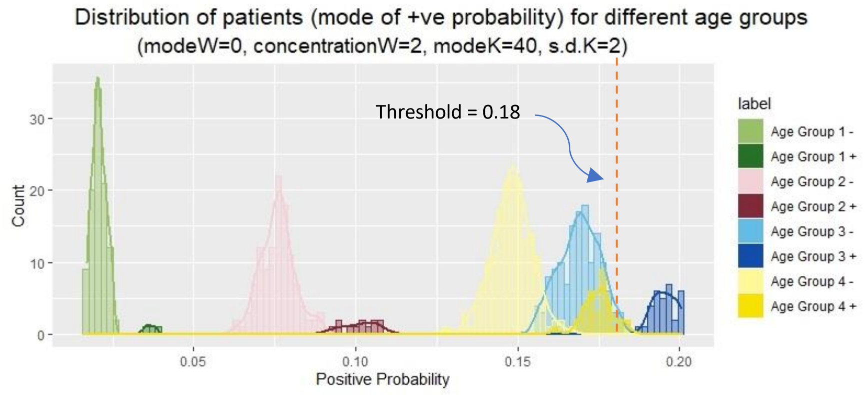
Trial 5 - Mode$_\omega$=0 Conentration$_\omega$=2; Mode$_\kappa$ = 1 and S.D$_\kappa$ = 2


```
         response
predicted   0   1
        0 512  44
        1   5  37
```



Trial 9 - Mode$_\omega$=0 Conentration$_\omega$=2; Mode$_\kappa$ = 10 and S.D$_\kappa$ = 2


```
         response
predicted   0   1
        0 514  43
        1   3  38
```



Trial 13 - Mode$_\omega$=0 Conentration$_\omega$=2; Mode$_\kappa$ = 40 and S.D$_\kappa$ = 2

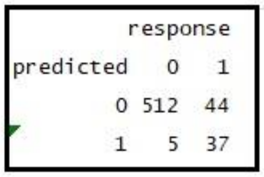

```
         response
predicted   0   1
        0 512  44
        1   5  37
```

Figure 25: Comparison of Probability Distribution with Different Kappa Priors

5.5 Diagnostic Check, Representativeness, Accuracy, and Efficiency

In this section, we would thoroughly discuss how to examine good diagnostic plots of all MCMC chains for the hierarchical model to provide the above analysis on the posterior distributions. Posterior distributions are only valid if they are from the MCMC with good diagnostic plots. We initially started running JAGS with the following settings:

Adapt	Burn-in	Saved steps	Thinning
1000	1000	1000	10

in our first ever attempt on the sample dataset and gradually increased to the following values for trial 1 with vague prior.

Adapt	Burn-in	Saved steps	Thinning
8000	8000	12000	2000

Figure 26 shows how the diagnostic plots have been improved from the initial settings (left) to our satisfactory level (right) on kappa2 as an example.

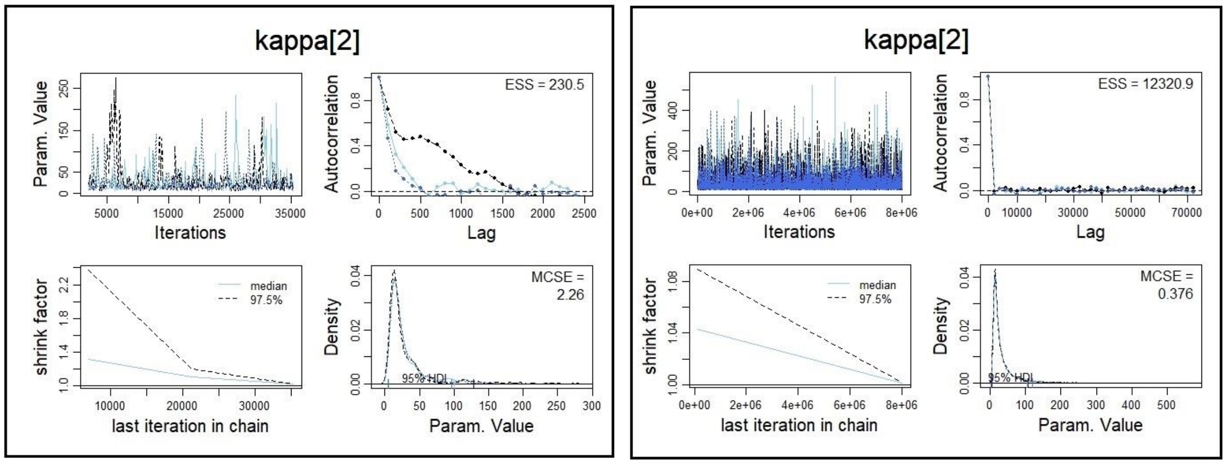


Figure 26: Comparison of Diagnostic Plots of Kappa[2] (Initial Settings (left), Final Settings (right))

On the right (Figure 26 – Final Settings):

1) 3 chains are all overlapping for the kappa[2] in the later iterations (top-left plot)
2) there are almost no autocorrelations in the chains (from the top-right plot), ESS is very high, which implies we do not have to change the number of thinning steps.
3) shrink factor is always less than 1.2 (lower-left plot), which implies there are no orphaned or stuck chains
4) In the density plot (lower-right plot), the shape and HDI interval overlap very well. There is no need to adjust any burn-in or saved steps. MCSE is relatively low.

On the left (Figure 26 – Initial Settings):

1) 3 chains do not overlap at all (top-left plot), which indicates we need more saved steps and burn-in steps,

2) there are high autocorrelations in the chains (from the top-right plot), ESS is extremely low, which implies we need to increase the number of thinning steps.
3) shrink factor is higher than 1.2 (lower-left plot), which implies there is orphaned or stuck chains
4) In the density plot (lower-right plot), the shape and HDI interval overlap, however MCSE is larger than 1. We need to adjust the burn-in or saved steps.

We put burn-in and adapt steps to 8000 in the tuned model, as we found that some of the parameters have a high shrink factor until we put 8000 steps for burn-in and adapt. Autocorrelation of kappa[3] and kappa[4] are finally reduced when thinning reached 2000, and saved steps reached 12,000.

With this configuration, all the parameters would have similar satisfactory diagnostic plots, however, JAGS took around 6 hours to run with these settings. Table 28 shows the runtime of the hierarchical model on the dataset.

We found that when $S.D_{\kappa} = 2$, we could reduce the thinning steps to 800 and saved steps to 8,000, and we could still obtain satisfactory diagnostic plots.

As the total steps being run would be multiped by the number of thinning, for example, with the satisfactory diagnostic plots, the total steps highlighted in yellow would be: adaptSteps: 8000 + burnInSteps: 8000 + (numSavedSteps: 12000 * thinSteps: 2000) = 24,016,000 steps.

If we reduced thinning steps to 800 and saved steps to 8,000 (highlighted in blue in Table 19) , total steps = 6,416,000 steps, thus we are saving 17,600,000 steps for trial 5 to 16.

However, we have observed some of the parameters do not have good diagnostic plots when thinning and saved steps are specified as the row highlighted in yellow in Table 28 for trial 3 and 4 (Modeω=0.1 or Modeω=0.5, Conentrationω =100; Modeκ = 1 and S.Dκ = 100). We need to increased thinning to 3,000 to get past the diagnostic checks. This inspired us to check the diagnostic plots, whenever there are changes in the prior settings. Refer to [Appendix A4] for all the diagnostic plots for hierarchical models of trial 1.

Table 19: Run Time of the Hierarchical Model

adapt	Burn-in	Saved steps	thinning	user	system	elapsed	Total time (in minutes)
1000	1000	1000	10	0.94	0.3	47.28	0.0131
1000	1000	1000	30	1.14	0.27	107.14	0.0298
1000	1000	1000	100	1.2	0.47	638.03	0.1772
4000	4000	4000	300	7.86	1.9	3869.28	1.0748
8000	8000	8000	600	11.98	2.5	5096.19	1.4156
8000	8000	8000	800	8.44	1.59	7746.44	2.1518
8000	8000	12000	1200	41.61	11	14916.93	4.1436
8000	8000	8000	2000	23.01	5.96	16098.68	4.4719
8000	8000	12000	2000	23.1	7.95	22330.59	6.2029
8000	8000	12000	3000	27.39	6.45	34078.08	9.4661

6. Summary and Conclusion

In this project, we have used JAGS to run MCMC to perform logistic regression, model comparison, and apply the hierarchical model on the given dataset.

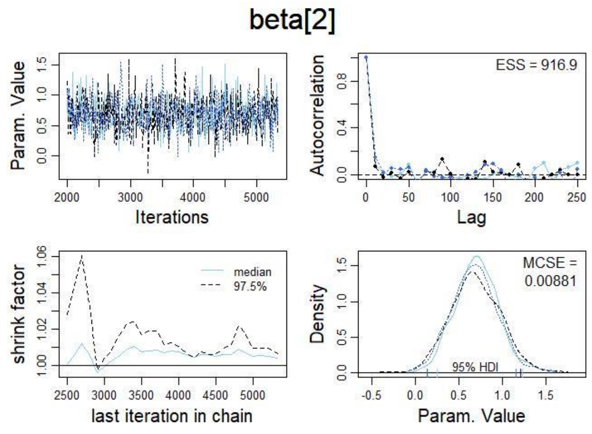
From logistic regression, without any prior knowledge, we learned that Patient age quantile (β_1), Hemoglobin (β_2), Platelets (β_3), Leukocytes (β_7), Basophils (β_8), MCH (β_9), Eosinophils (β_{10}), and RDW (β_{12}) are significant to classify covid test results.

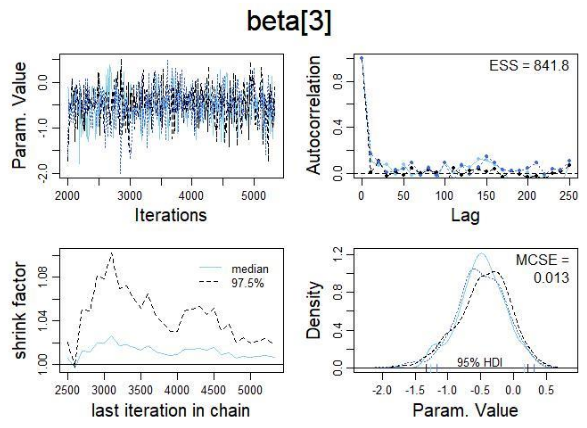
Hemoglobin (β_2), Platelets (β_3), MCHC (β_6), Basophils (β_8), MCH (β_9) are sensitive to prior variance in order to classify the test sets better.

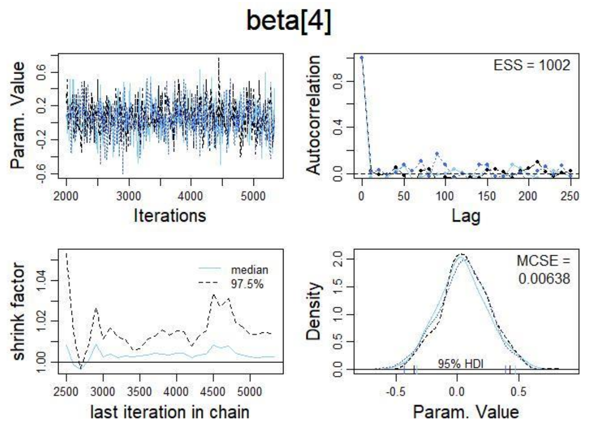
From the hierarchical model, age groups 3 and 4 are more sensitive to prior knowledge than age groups 1 and 2. The positive probability distribution of age groups 1 and 2 would become more different from age groups 3 and 4 if we have been informed there is a higher chance of getting the virus.

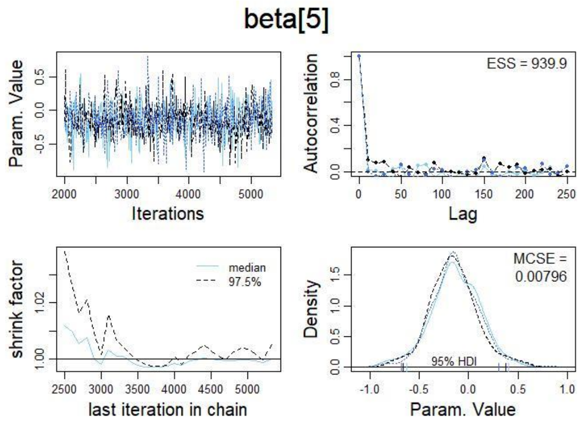
Appendix A: Diagnostic Plots of the Trials for the Logistic Regression and Hierarchical Model

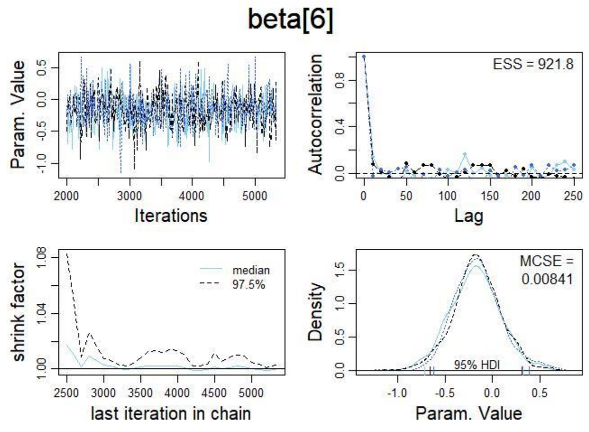
[A1] – Diagnostic plots of trial no. 1

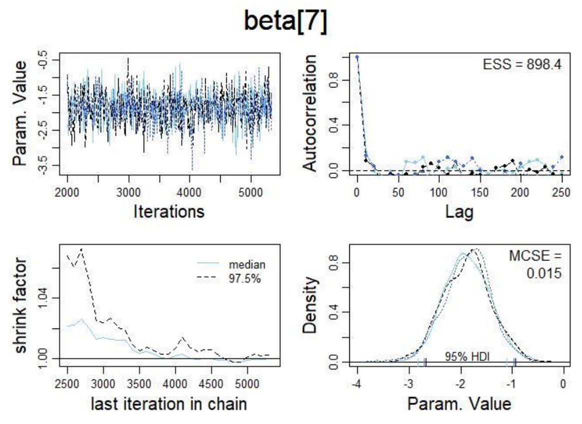


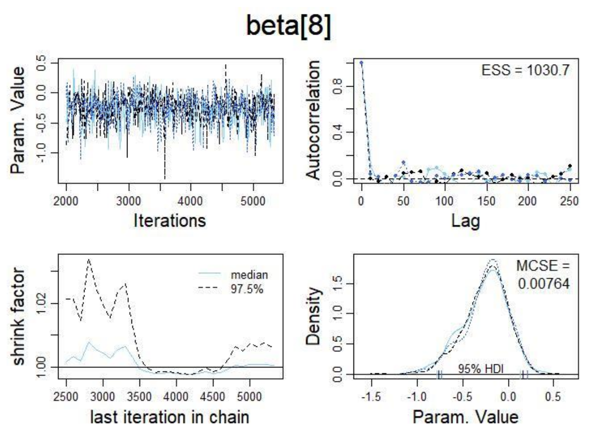


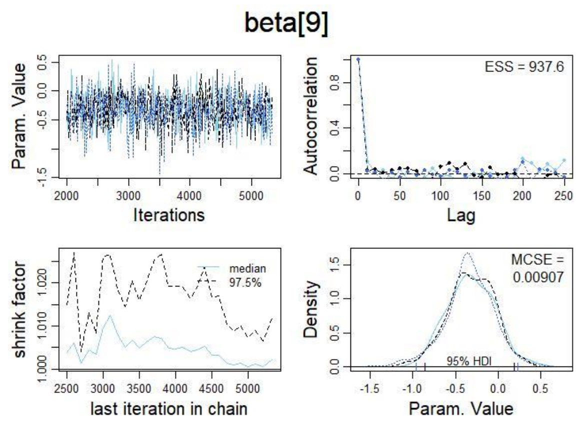


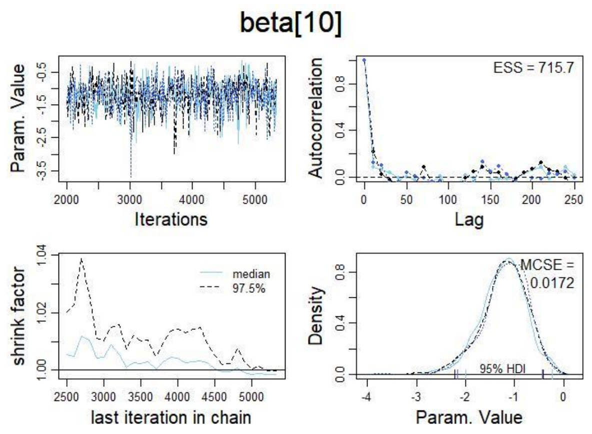


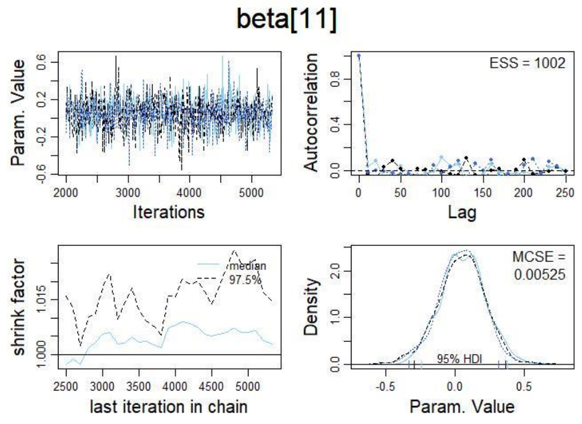


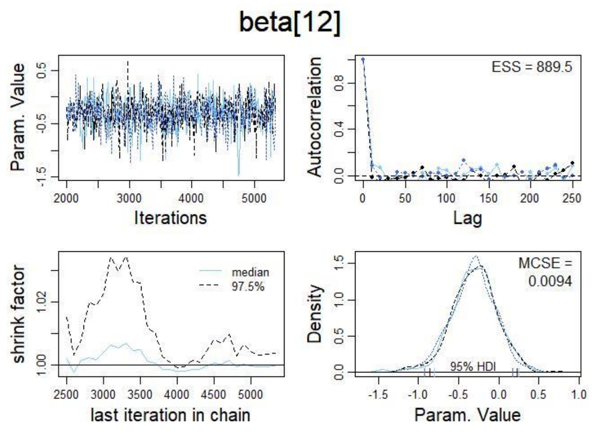


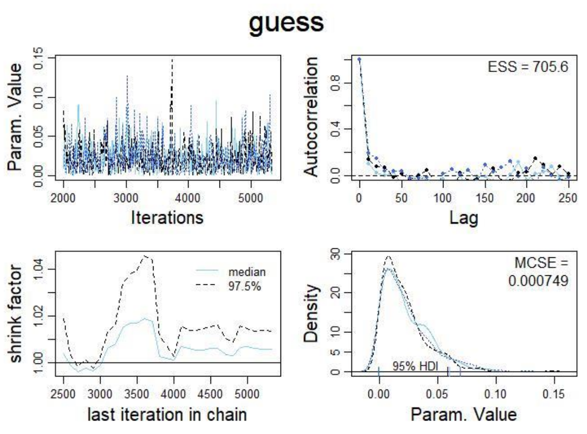

beta[8]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 1030.7
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.00764
95% HDI


beta[9]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 937.6
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.00907
95% HDI

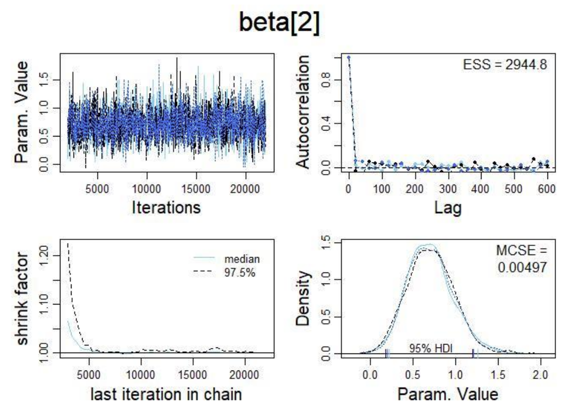

beta[10]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 715.7
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.0172
95% HDI

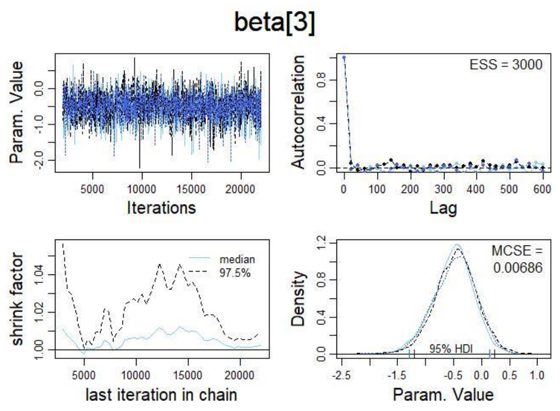

beta[11]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 1002
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.00525
95% HDI

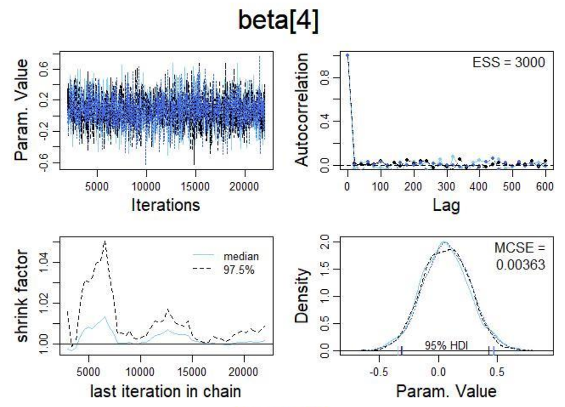

beta[12]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 889.5
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.0094
95% HDI

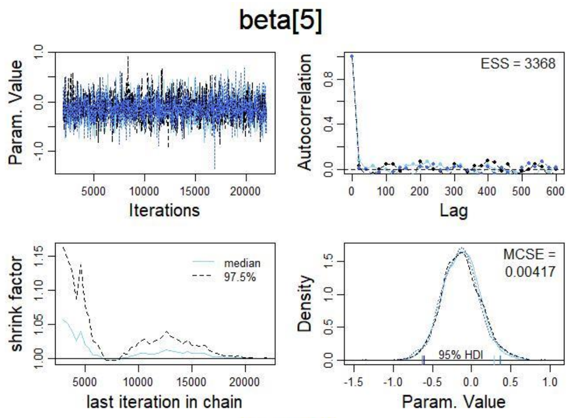

guess
Param. Value
Iterations
Autocorrelation
Lag
ESS = 705.6
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.000749
95% HDI

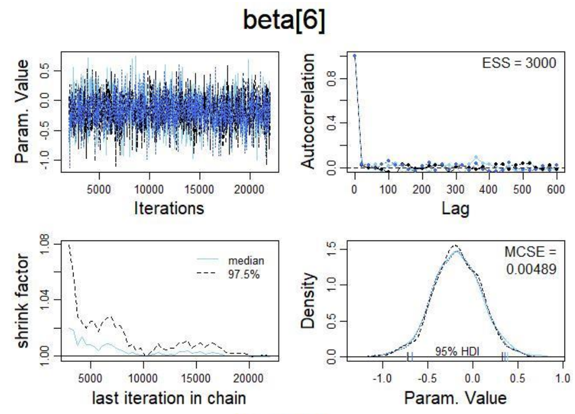
[A2] – Diagnostic plots of trial no. 2

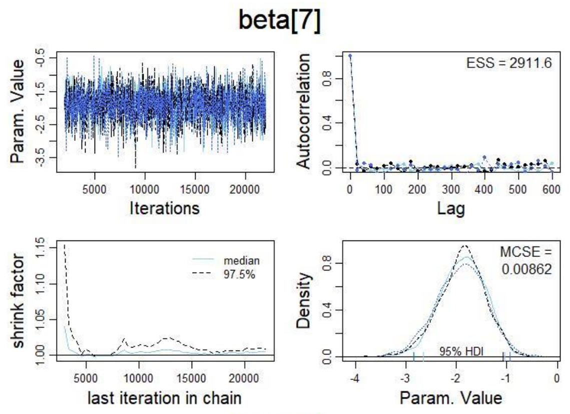


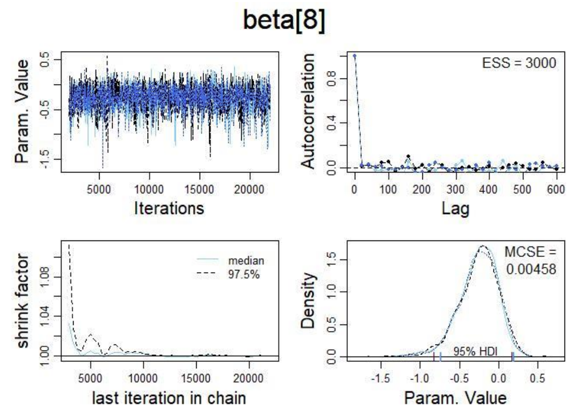


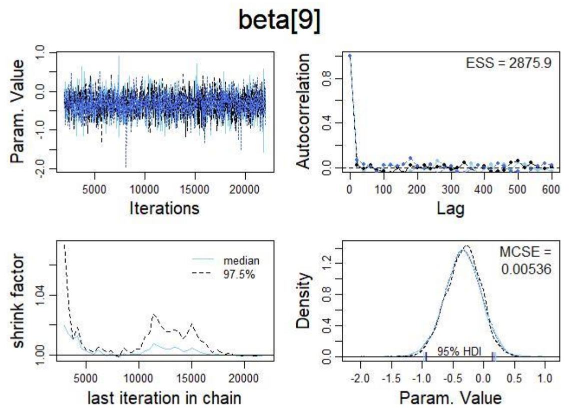


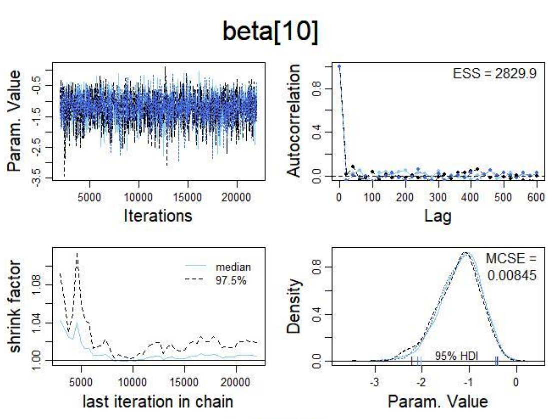


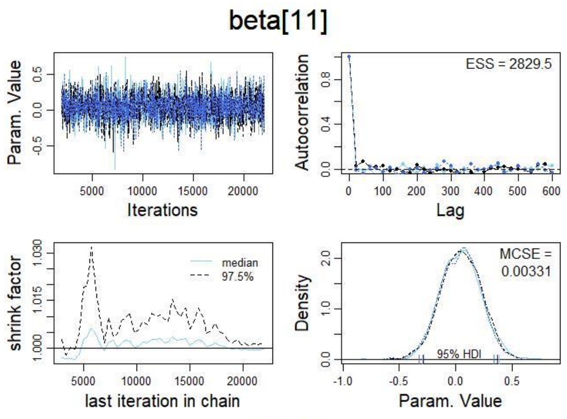


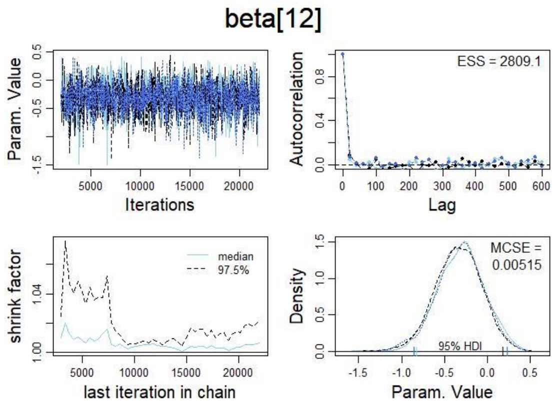


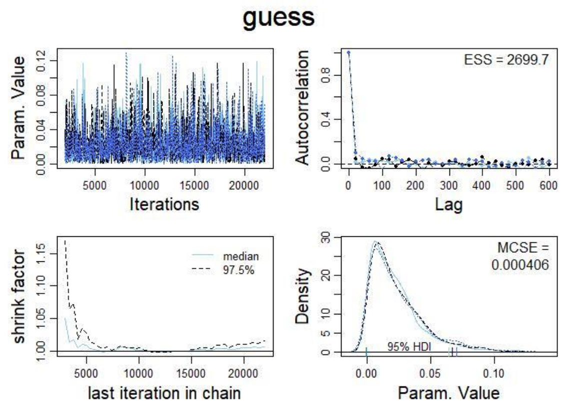




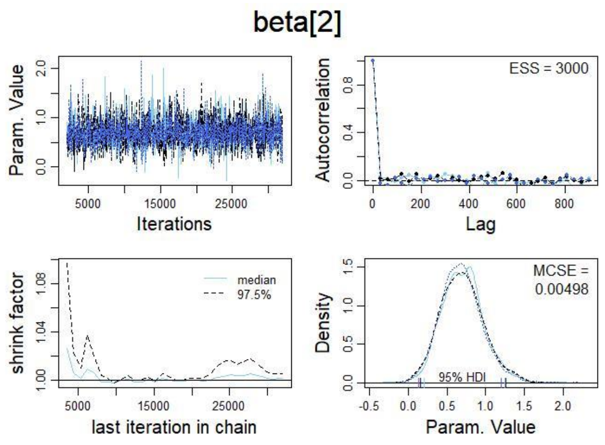


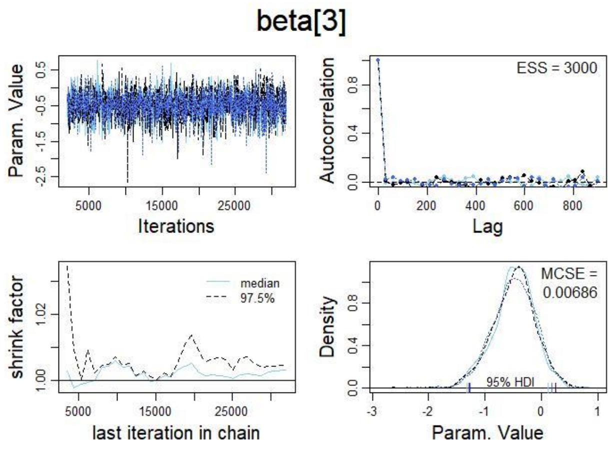


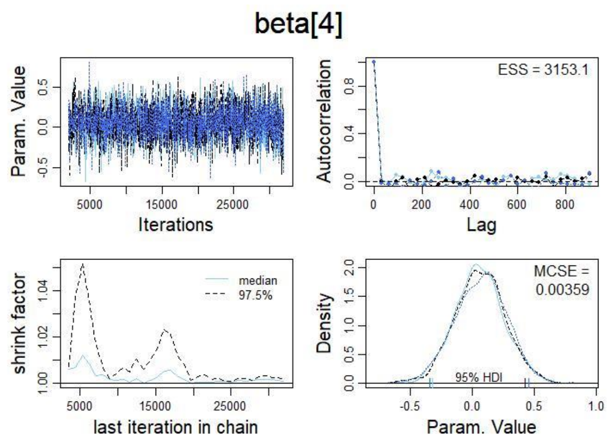


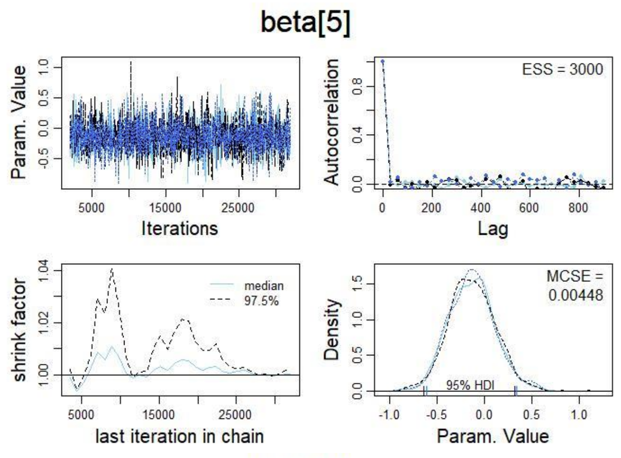


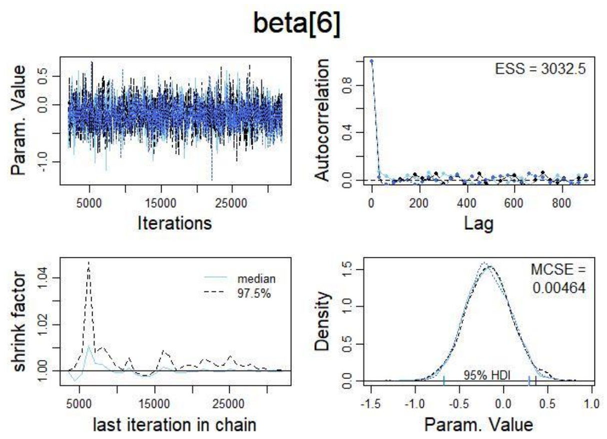
[A3] – Diagnostic plots of trial no. 3

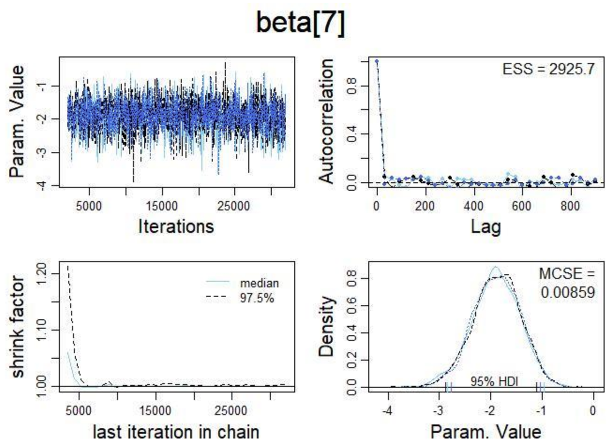


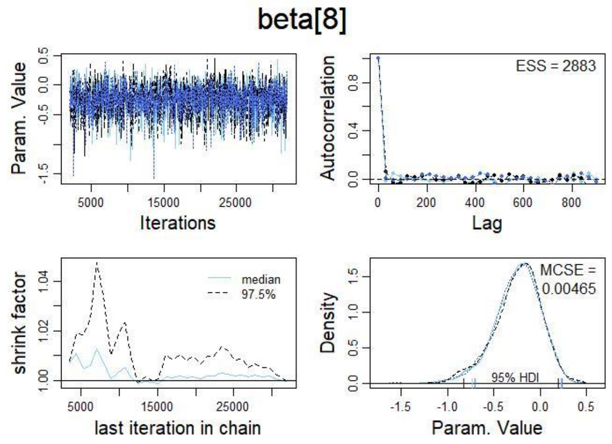


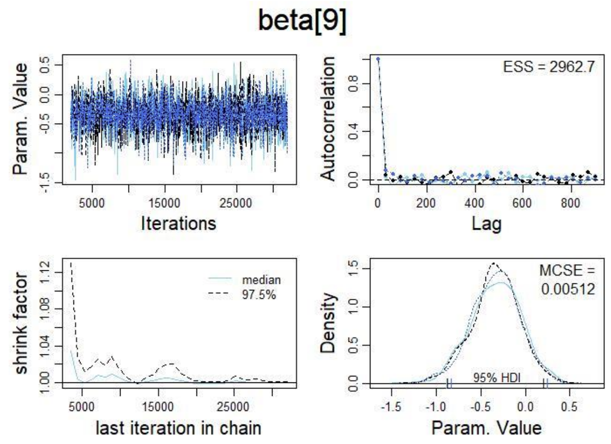

beta[4]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 3153.1
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.00359
95% HDI

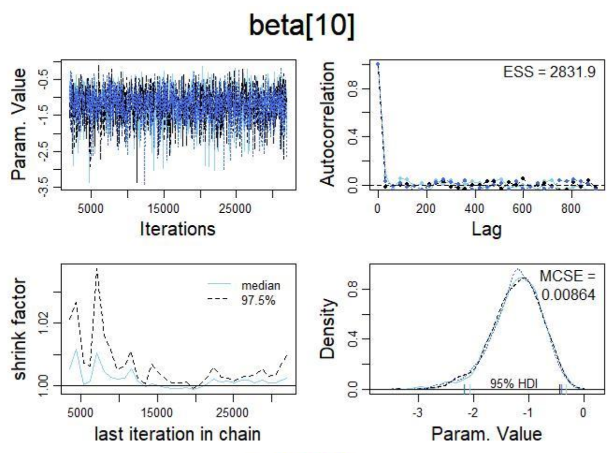

beta[5]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 3000
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.00448
95% HDI

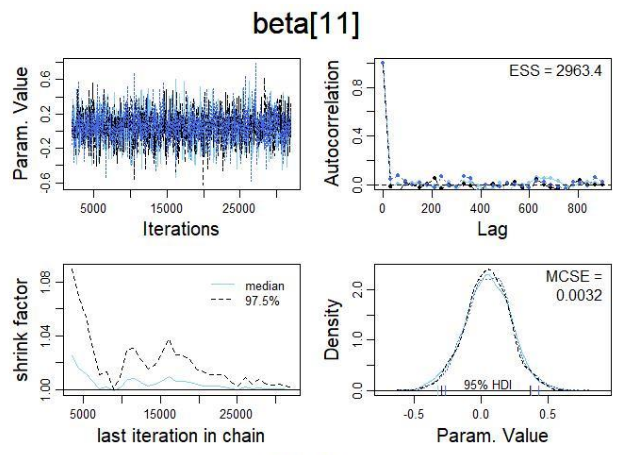

beta[6]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 3032.5
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.00464
95% HDI

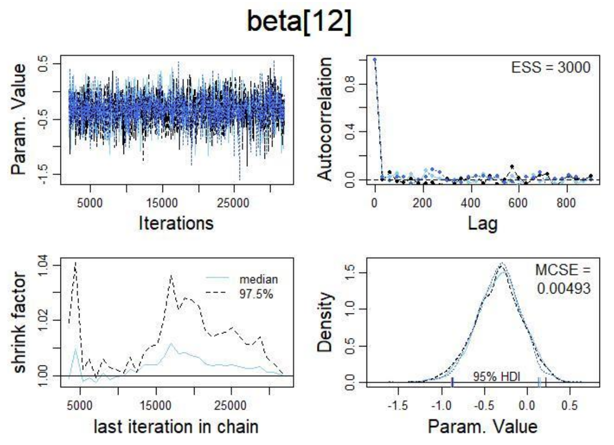

beta[7]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 2925.7
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.00859
95% HDI

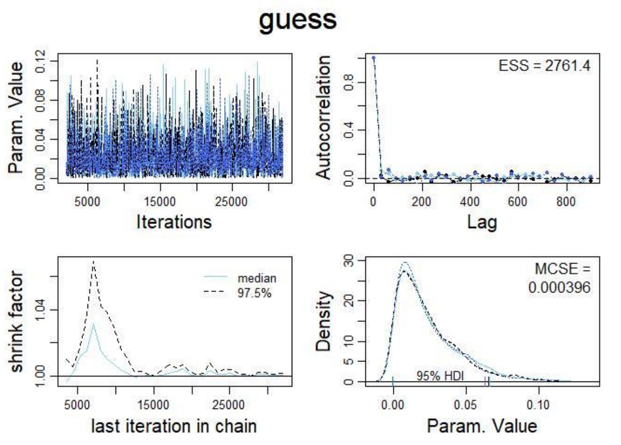

beta[8]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 2883
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.00465
95% HDI


beta[9]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 2962.7
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE = 0.00512
95% HDI

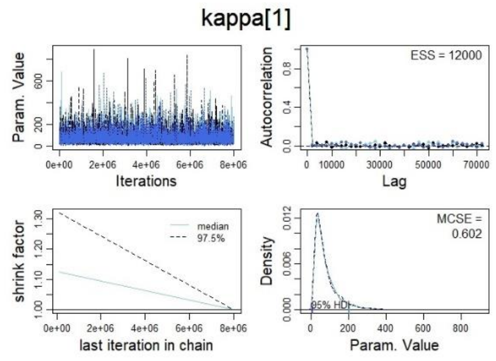


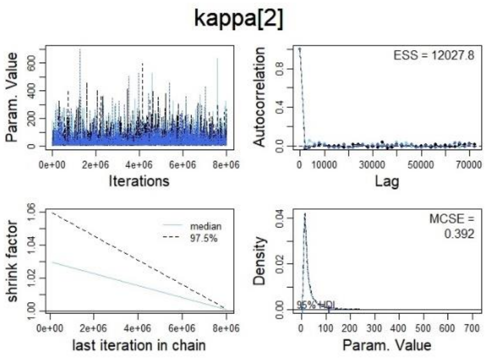


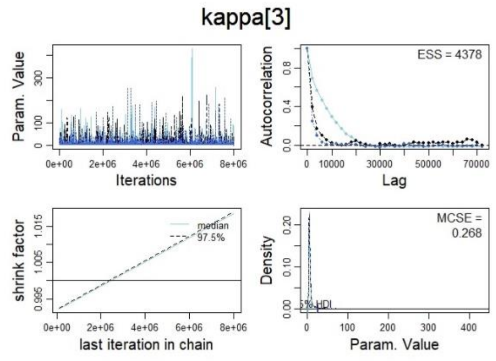


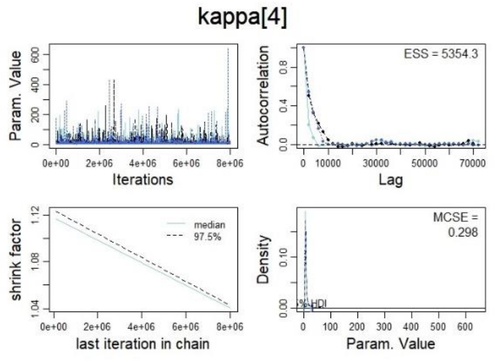


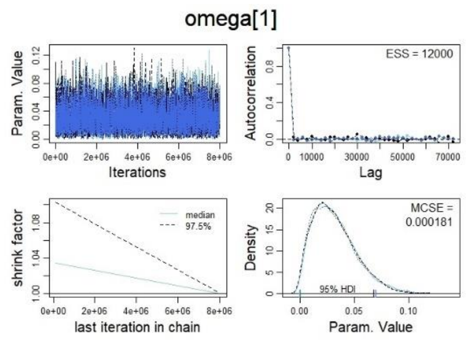
[A4] – Diagnostic plots of hierarchical model

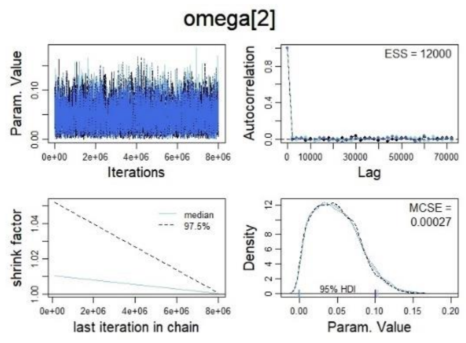

kappa[1]
Param. Value
Iterations
Autocorrelation
ESS = 12000
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 0.602
95% HDI
Param. Value

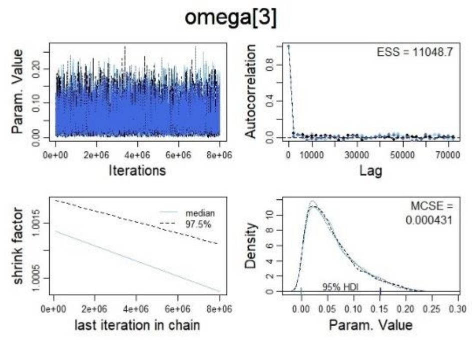

kappa[2]
Param. Value
Iterations
Autocorrelation
ESS = 12027.8
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 0.392
95% HDI
Param. Value

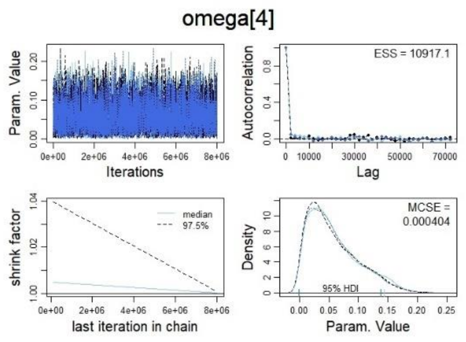

kappa[3]
Param. Value
Iterations
Autocorrelation
ESS = 4378
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 0.268
5% HDI
Param. Value

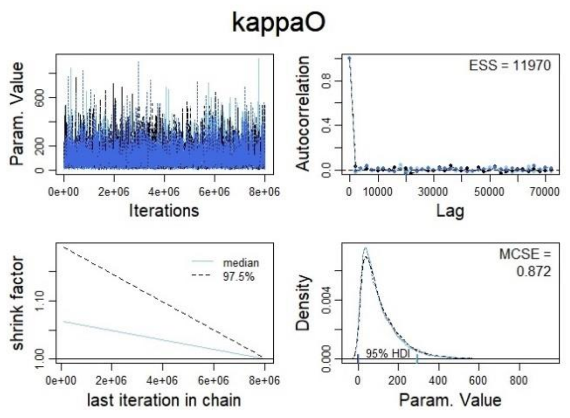

kappa[4]
Param. Value
Iterations
Autocorrelation
ESS = 5354.3
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 0.298
% HDI
Param. Value

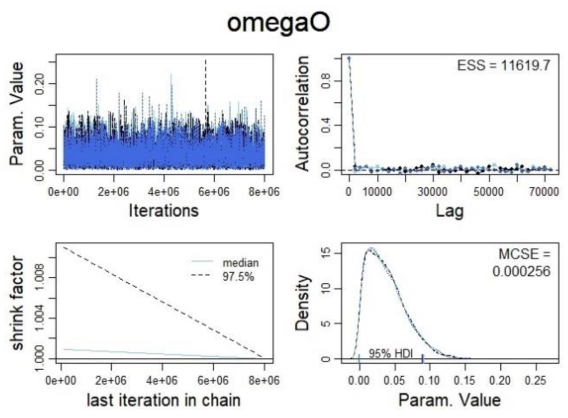

omega[1]
Param. Value
Iterations
Autocorrelation
ESS = 12000
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 0.000181
95% HDI
Param. Value

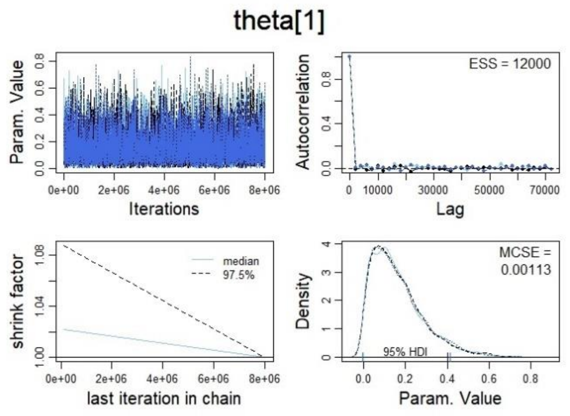

omega[2]
Param. Value
Iterations
Autocorrelation
ESS = 12000
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 0.00027
95% HDI
Param. Value

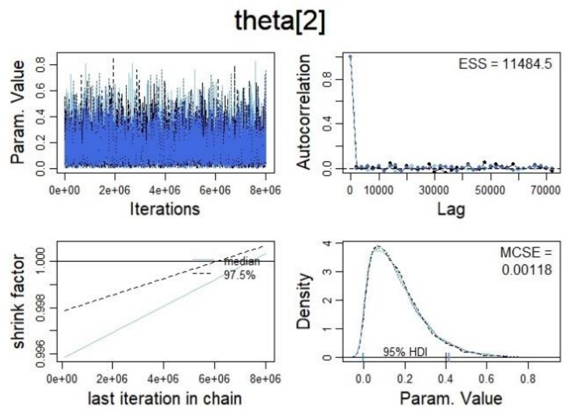

omega[3]
Param. Value
Iterations
Autocorrelation
ESS = 11048.7
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 0.000431
95% HDI
Param. Value


omega[4]
Param. Value
Iterations
Autocorrelation
ESS = 10917.1
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 0.000404
95% HDI
Param. Value









Appendix B: R Codes

[B1] – R codes of the Logistic Regression Model

```
graphics.off() # This closes all of R's graphics windows.
rm(list=ls())  # Careful! This clears all of R's memory!

# INSTALLATION AND LOADING OF THE PACKAGES
packages <- c('ggplot2', 'ggpubr', 'ks', 'rjags', 'runjags', 'nimble', '"PerformanceA
nalytics"', 'psych', 'GGally', 'summarytools', 'knitr', 'dplyr', 'data.table', 'split
stackshape')

for (pkg in packages) {
  if (pkg %in% rownames(installed.packages()) == FALSE)
  {install.packages(pkg)}
  if (pkg %in% rownames(.packages()) == FALSE)
  {library(pkg, character.only = TRUE)}
}

setwd("D:/RMIT/Sem 3/Applied Bayesian")
source("DBDA2E-utilities.R")
```

```
#============================================================================
#===========================DATA PREPARATION================================
#============================================================================

myData <- read.csv("covid_brazil_final.csv")
describe(myData)
x = as.matrix(myData[,c("Patient.age.quantile","Hematocrit", "Hemoglobin","Pl
atelets", "Mean.platelet.volume", "Red.blood.Cells", "Lymphocytes", "Mean.cor
puscular.hemoglobin.concentrationA.MCHC.", "Leukocytes", "Basophils", "Mean.c
orpuscular.hemoglobin.MCH.", "Eosinophils", "Mean.corpuscular.volume.MCV.", "
Monocytes", "Red.blood.cell.distribution.width.RDW.")])

# Some more descriptives
cat("\nCORRELATION MATRIX OF PREDICTORS:\n ")
show( round(cor(x),3) )
cat("\n")

x_correlation <- round(cor(x),3)
write.csv(x_correlation, "x_correlation.csv" )

#Pairplots for all variables
chart.Correlation(x, histogram=TRUE, pch=19)

#Remove Hematocrit, Red.blood.Cells and Mean.corpuscular.volume.MCV. due to h
igh correlation

myData2 = myData[,c("Patient.ID", "Patient.age.quantile", "SARS.Cov2.exam.res
ult", "Hemoglobin","Platelets", "Mean.platelet.volume", "Lymphocytes", "Mean.
corpuscular.hemoglobin.concentrationA.MCHC.", "Leukocytes", "Basophils", "Mea
n.corpuscular.hemoglobin.MCH.", "Eosinophils", "Monocytes", "Red.blood.cell.d
istribution.width.RDW.")]

names(myData2)[3] <- "covid.results"
describe(myData2)

myData2$covid.results <- as.numeric(as.factor(myData2$covid.results)) - 1 # T
o get 0/1 instead of 1/2; positive = 1; negative = 0

#check for missing values
sum(is.na(myData2))

#Descriptive look
p1 <- ggplot(myData2, aes(x=Patient.age.quantile, y = covid.results)) +
  geom_point()

p2 <- ggplot(myData2, aes(x=Hemoglobin, y = covid.results)) +
  geom_point()

p3 <- ggplot(myData2, aes(x=Platelets, y = covid.results)) +
  geom_point()

p4 <- ggplot(myData2, aes(x=Mean.platelet.volume, y = covid.results)) +
  geom_point()

p5 <- ggplot(myData2, aes(x=Lymphocytes, y = covid.results)) +
  geom_point()

p6 <- ggplot(myData2, aes(x=Mean.corpuscular.hemoglobin.concentrationA.MCHC.,
y = covid.results)) +
  geom_point()

p7 <- ggplot(myData2, aes(x=Leukocytes, y = covid.results)) +
  geom_point()

p8 <- ggplot(myData2, aes(x=Basophils, y = covid.results)) +
  geom_point()

p9 <- ggplot(myData2, aes(x=Mean.corpuscular.hemoglobin.MCH., y = covid.resul
ts)) +
  geom_point()

p10 <- ggplot(myData2, aes(x=Eosinophils, y = covid.results)) +
  geom_point()

p11 <- ggplot(myData2, aes(x=Monocytes, y = covid.results)) +
  geom_point()

p12 <- ggplot(myData2, aes(x=Red.blood.cell.distribution.width.RDW., y = covi
d.results)) +
  geom_point()


figure <- ggarrange(p1, p2, p3, p4, p5, p10, p7, p8, nrow = 4, ncol = 2)
figure <- annotate_figure(figure,
                          top = text_grob("Covid test results vs independent
variables", face = "bold", size = 14))

figure

figure2 <- ggarrange(p9, p6, p11, p12, nrow = 4, ncol = 1)
figure2 <- annotate_figure(figure2,
                           top = text_grob("Covid test results vs independent
variables", face = "bold", size = 14))

figure2

myData3 = myData[,c("Hemoglobin","Platelets",
                    "Mean.platelet.volume", "Lymphocytes",
                    "Mean.corpuscular.hemoglobin.concentrationA.MCHC.", "Leuk
ocytes",
                    "Basophils", "Mean.corpuscular.hemoglobin.MCH.", "Eosinop
hils",
                    "Monocytes", "Red.blood.cell.distribution.width.RDW.")]
names(myData3)[5] <- "MCHC"
names(myData3)[8] <- "MCH"
names(myData3)[11] <- "Red Blood cell D.W"

boxplot(myData3, xaxt = "n")
text(x = 1:length(myData3), y = par("usr")[3] - 0.45, labels = names(myData3)
, xpd = NA, srt = 35, cex = 1.2)


# THE DATA
set.seed(999)
trainData <- stratified(myData2, "covid.results", .7)
testData <- setdiff(myData2, trainData)


x = as.matrix(trainData[,c("Patient.age.quantile", "Hemoglobin","Platelets",
                           "Mean.platelet.volume", "Lymphocytes",
                           "Mean.corpuscular.hemoglobin.concentrationA.MCHC."
, "Leukocytes",
                           "Basophils", "Mean.corpuscular.hemoglobin.MCH.", "
Eosinophils",
                           "Monocytes", "Red.blood.cell.distribution.width.RD
W.")])


y = unlist(trainData[, "covid.results"])

#============================================================================
#------------------------------ LOGISTIC REGRESSION ------------------------
#============================================================================

#===============PRELIMINARY FUNCTIONS FOR POSTERIOR INFERENCES==============
=====
genMCMC = function( x, y, numAdaptSteps=500, numBburnInSteps = 500,
                    numSavedSteps=500 ,  thinSteps=1 , saveName=NULL ,
                    runjagsMethod=runjagsMethodDefault ,
                    nChains=nChainsDefault, beta1Sens="4", beta2Sens="4", bet
a3Sens="4", beta4Sens="4", beta5Sens="4",
                    beta6Sens="4", beta7Sens="4", beta8Sens="4", beta9Sens="4
", beta10Sens="4",
                    beta11Sens="4", beta12Sens="4"
) {
  require(runjags)
  #-------------------------------------------------------------------------
---
  # THE DATA.
  # Do some checking that data make sense:
  if ( any( !is.finite(y) ) ) { stop("All y values must be finite.") }
  if ( any( !is.finite(x) ) ) { stop("All x values must be finite.") }
  cat("\nCORRELATION MATRIX OF PREDICTORS:\n ")
  show( round(cor(x),3) )
  cat("\n")
  flush.console()
  # Specify the data in a list, for later shipment to JAGS:
  dataList = list(
    x = x ,
    y = y ,
    Nx = dim(x)[2] ,
    Ntotal = dim(x)[1]
  )
  #-----------------------------------------------------------------------------
---
  # THE MODEL.
  modelString = "
#  Specify the model for standardized data:
  model {
    for ( i in 1:Ntotal ) {
      # In JAGS, ilogit is logistic:
      y[i] ~ dbern( mu[i] )
      mu[i] <- ( guess*(1/2)
                 + (1.0-guess)*ilogit(beta0+sum(beta[1:Nx]*x[i,1:Nx])) )
    }
    "
  priorString0=paste("beta0 ~ dnorm( 0 , 1/2^2 )", "\n")
  priorString1= paste("beta[1] ~ dnorm(0 , 1/", beta1Sens, ")", "\n")
  priorString2= paste("beta[2] ~ dnorm(0 , 1/", beta2Sens, ")", "\n")
  priorString3= paste("beta[3] ~ dnorm(0 , 1/", beta3Sens, ")", "\n")
  priorString4= paste("beta[4] ~ dnorm(0 , 1/", beta4Sens, ")", "\n")
  priorString5= paste("beta[5] ~ dnorm(0 , 1/", beta5Sens, ")", "\n")
  priorString6= paste("beta[6] ~ dnorm(0 , 1/", beta6Sens, ")", "\n")
  priorString7= paste("beta[7] ~ dnorm(0 , 1/", beta7Sens, ")", "\n")
  priorString8= paste("beta[8] ~ dnorm(0 , 1/", beta8Sens, ")", "\n")
  priorString9= paste("beta[9] ~ dnorm(0 , 1/", beta9Sens, ")", "\n")
  priorString10= paste("beta[10] ~ dnorm(0 , 1/", beta10Sens, ")", "\n")
  priorString11= paste("beta[11] ~ dnorm(0 , 1/", beta11Sens, ")", "\n")
  priorString12= paste("beta[12] ~ dnorm(0 , 1/", beta12Sens, ")", "\n")

  priorString = paste(priorString0, priorString1,priorString2, priorString3, 
priorString4, priorString5, priorString6, 
                      priorString7, priorString8,priorString9,priorString10,p
riorString11, priorString12)

  # Priors vague on standardized scale:
  guessString="

    guess ~ dbeta(1,9)
    # Transform to original scale:
    # beta[1:Nx] <- zbeta[1:Nx] / xsd[1:Nx]
    # beta0 <- zbeta0 - sum( zbeta[1:Nx] * xm[1:Nx] / xsd[1:Nx] )
  }
  " # close quote for modelString
  # Write out modelString to a text file
  finalString = paste(modelString, priorString, guessString)
  writeLines( finalString , con="TEMPmodel.txt" )
  #-----------------------------------------------------------------------------
---
  # INTIALIZE THE CHAINS.
  # Let JAGS do it...

  #-----------------------------------------------------------------------------
---
  # RUN THE CHAINS
  parameters = c( "beta0" ,  "beta" ,
                  "guess" )
  runJagsOut <- run.jags( method=runjagsMethod ,
                          model="TEMPmodel.txt" ,
                          monitor=parameters ,
                          data=dataList ,
                          #inits=initsList ,
                          n.chains=nChains ,
                          adapt=numAdaptSteps ,
                          burnin=numBburnInSteps ,
                          sample=ceiling(numSavedSteps/nChains) ,
                          thin=thinSteps ,
                          summarise=FALSE ,
                          plots=FALSE )
  codaSamples = as.mcmc.list( runJagsOut )
  # resulting codaSamples object has these indices:
  #   codaSamples[[ chainIdx ]][ stepIdx , paramIdx ]
  if ( !is.null(saveName) ) {
    save( codaSamples , file=paste(saveName,"Mcmc.Rdata",sep="") )
  }
  return( codaSamples )
} # end function

#===============================================================================
==========

smryMCMC_HD = function(  codaSamples , compVal = NULL,  saveName=NULL) {
  summaryInfo = NULL
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  paramName = colnames(mcmcMat)
  for ( pName in paramName ) {
    if (pName %in% colnames(compVal)){
      if (!is.na(compVal[pName])) {
        summaryInfo = rbind( summaryInfo , summarizePost( paramSampleVec = mc
mcMat[,pName] ,
                                                           compVal = as.numeri
c(compVal[pName]) ))
      }
      else {
        summaryInfo = rbind( summaryInfo , summarizePost( paramSampleVec = mc
mcMat[,pName] ) )
      }
    } else {
      summaryInfo = rbind( summaryInfo , summarizePost( paramSampleVec = mcmc
Mat[,pName] ) )
    }
  }
  rownames(summaryInfo) = paramName

  # summaryInfo = rbind( summaryInfo ,
  #                      "tau" = summarizePost( mcmcMat[,"tau"] ) )
  if ( !is.null(saveName) ) {
    write.csv( summaryInfo , file=paste(saveName,"SummaryInfo.csv",sep="") )
  }
  return( summaryInfo )
}

#=============================================================================
===

plotMCMC_HD = function( codaSamples , data , xName="x" , yName="y", preds = F
ALSE ,
                        showCurve=FALSE ,  pairsPlot=FALSE , compVal = NULL,
                        saveName=NULL , saveType="jpg" ) {
  # showCurve is TRUE or FALSE and indicates whether the posterior should
  #   be displayed as a histogram (by default) or by an approximate curve.
  # pairsPlot is TRUE or FALSE and indicates whether scatterplots of pairs
  #   of parameters should be displayed.
  #-----------------------------------------------------------------------------
---
  y = data[,yName]
  x = as.matrix(data[,xName])
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  chainLength = NROW( mcmcMat )
  # zbeta0 = mcmcMat[,"zbeta0"]
  # zbeta  = mcmcMat[,grep("^zbeta$|^zbeta\\[",colnames(mcmcMat))]
  # if ( ncol(x)==1 ) { zbeta = matrix( zbeta , ncol=1 ) }
  beta0 = mcmcMat[,"beta0"]
  beta  = mcmcMat[,grep("^beta$|^beta\\[",colnames(mcmcMat))]
  if ( ncol(x)==1 ) { beta = matrix( beta , ncol=1 ) }
  if (preds){
    pred = mcmcMat[,grep("^pred$|^pred\\[",colnames(mcmcMat))]
  } # Added by Demirhan
  guess = mcmcMat[,"guess"]
  #------------------------------------------------------------------------
---
  # Compute R^2 for credible parameters:
  YcorX = cor( y , x ) # correlation of y with each x predictor
  Rsq = beta %*% matrix( YcorX , ncol=1 )
  #------------------------------------------------------------------------
---
  if ( pairsPlot ) {
    # Plot the parameters pairwise, to see correlations:
    openGraph()
    nPtToPlot = 1000
    plotIdx = floor(seq(1,chainLength,by=chainLength/nPtToPlot))
    panel.cor = function(x, y, digits=2, prefix="", cex.cor, ...) {
      usr = par("usr"); on.exit(par(usr))
      par(usr = c(0, 1, 0, 1))
      r = (cor(x, y))
      txt = format(c(r, 0.123456789), digits=digits)[1]
      txt = paste(prefix, txt, sep="")
      if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
      text(0.5, 0.5, txt, cex=1.25 ) # was cex=cex.cor*r
    }
    pairs( cbind( beta0 , beta )[plotIdx,] ,
           labels=c( "beta[0]" ,
                     paste0("beta[",1:ncol(beta),"]\n",xName) ,
                     expression(tau) ) ,
           lower.panel=panel.cor , col="skyblue" )
    if ( !is.null(saveName) ) {
      saveGraph( file=paste(saveName,"PostPairs",sep=""), type=saveType)
    }
  }
  #------------------------------------------------------------------------
---
  # Marginal histograms:

  decideOpenGraph = function( panelCount , saveName , finished=FALSE ,
                              nRow=2 , nCol=3 ) {
    # If finishing a set:
    if ( finished==TRUE ) {
      if ( !is.null(saveName) ) {
        saveGraph( file=paste0(saveName,ceiling((panelCount-1)/(nRow*nCol))),
                   type=saveType)
      }
      panelCount = 1 # re-set panelCount
      return(panelCount)
    } else {
      # If this is first panel of a graph:
      if ( ( panelCount %% (nRow*nCol) ) == 1 ) {
        # If previous graph was open, save previous one:
        if ( panelCount>1 & !is.null(saveName) ) {
          saveGraph( file=paste0(saveName,(panelCount%/%(nRow*nCol))),
                    type=saveType)
        }
        # Open new graph
        openGraph(width=nCol*7.0/3,height=nRow*2.0)
        layout( matrix( 1:(nRow*nCol) , nrow=nRow, byrow=TRUE ) )
        par( mar=c(4,4,2.5,0.5) , mgp=c(2.5,0.7,0) )
      }
      # Increment and return panel count:
      panelCount = panelCount+1
      return(panelCount)
    }
  }

  # Original scale:
  panelCount = 1
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMa
rg") )
  histInfo = plotPost( beta0 , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(beta[0]) , main="Intercept", compVal = as.
numeric(compVal["beta0"] ))
  for ( bIdx in 1:ncol(beta) ) {
    panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"Post
Marg") )
    if (!is.na(compVal[paste0("beta[",bIdx,"]")])){
      histInfo = plotPost( beta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve
,
                           xlab=bquote(beta[.(bIdx)]) , main=xName[bIdx],
                           compVal = as.numeric(compVal[paste0("beta[",bIdx,"
]")]))
    } else{
      histInfo = plotPost( beta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve
,
                           xlab=bquote(beta[.(bIdx)]) , main=xName[bIdx])
    }
  }
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMa
rg") )
  histInfo = plotPost( Rsq , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(R^2) , main=paste("Prop Var Accntd") )

  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMa
rg") )
  histInfo = plotPost( guess , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab="Guess parameter" , main=paste("Prop Var Accntd")
)

  panelCount = 1
  if ( preds){

    for ( pIdx in 1:ncol(pred) ) {
      panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"Po
stMarg") )
      histInfo = plotPost( pred[,pIdx] , cex.lab = 1.75 , showCurve=showCurve
,
                                   xlab=bquote(pred[.(pIdx)]) , main=paste0("Predicti
on ",pIdx) )
    }
  }
  # Standardized scale:
  panelCount = 1
  # panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"Post
MargZ") )
  # histInfo = plotPost( zbeta0 , cex.lab = 1.75 , showCurve=showCurve ,
  #                      xlab=bquote(z*beta[0]) , main="Intercept" )
  # for ( bIdx in 1:ncol(beta) ) {
  #   panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"Po
stMargZ") )
  #   histInfo = plotPost( zbeta[,bIdx] , cex.lab = 1.75 , showCurve=showCurv
e ,
  #                        xlab=bquote(z*beta[.(bIdx)]) , main=xName[bIdx] )
  # }
  # panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"Post
MargZ") )
  # histInfo = plotPost( Rsq , cex.lab = 1.75 , showCurve=showCurve ,
  #                      xlab=bquote(R^2) , main=paste("Prop Var Accntd") )
  # panelCount = decideOpenGraph( panelCount , finished=TRUE , saveName=paste
0(saveName,"PostMargZ") )

  #---------------------------------------------------------------------------
---
}

#===============PRELIMINARY FUNCTIONS FOR POSTERIOR INFERENCES===============
=====



numAdaptSteps = 1000 ; numBburnInSteps=1000; numSavedSteps=3000 ; thinSteps=4
0; nChains = 3

startTime = proc.time()
mcmcCoda = genMCMC( x, y, numAdaptSteps=numAdaptSteps, numBburnInSteps= numBb
urnInSteps,
                    numSavedSteps=numSavedSteps , thinSteps=thinSteps, beta12
Sens=0.00000001,
                    nChains = nChains )



stopTime = proc.time()
```

```
duration = stopTime - startTime
show(duration)

#save.image(file="rEnvironment_40Thin_3000_logistic_b12_try2.RData")
load(file="rEnvironment_40Thin_3000_logistic.RData")

#-----------------------------------------------------------------------------
---
# Display diagnostics of chain, for specified parameters:
parameterNames = c("beta0"  ,  "beta[1]",  "beta[2]" ,  "beta[3]" ,  "beta[4]
" ,  "beta[5]"  , "beta[6]"  ,
                   "beta[7]"  , "beta[8]", "beta[9]", "beta[10]", "beta[11]",
"beta[12]", "guess") #varnames(mcmcCoda) # get all parameter names
for ( parName in parameterNames ) {
  diagMCMC( codaObject=mcmcCoda , parName=parName)
}
#-----------------------------------------------------------------------------
---
# Get summary statistics of chain:

compVal <- data.frame("beta0" = 0, "beta[1]" = 0, "beta[2]" = 0, "beta[3]" =
0, "beta[4]" =  0,  "beta[5]" =  0,
                      "beta[6]" =  0, "beta[7]" = 0, "beta[8]" = 0, "beta[9]"
= 0,"beta[10]" = 0,
                      "beta[11]" = 0, "beta[12]" = 0,check.names=FALSE)

summaryInfo <- smryMCMC_HD( codaSamples = mcmcCoda , compVal = compVal )
print(summaryInfo)

plotMCMC_HD( codaSamples = mcmcCoda , data = myData2, xName=c("Patient.age.qu
antile", "Hemoglobin","Platelets",
                                                              "Mean.platelet.
volume", "Lymphocytes",
                                                              "Mean.corpuscul
ar.hemoglobin.concentrationA.MCHC.", "Leukocytes",
                                                              "Basophils", "M
ean.corpuscular.hemoglobin.MCH.", "Eosinophils",
                                                              "Monocytes", "R
ed.blood.cell.distribution.width.RDW."),
             yName="covid.results", compVal = compVal, preds= FALSE, pairsPlo
t=TRUE )

write.csv(summaryInfo, "summaryInfoLogistic_b12_try2.csv" )



# ============ Predictive check ============

coeffs <- as.vector(summaryInfo[2:14,3])
X <- as.matrix(cbind(rep(1,nrow(testData)),testData[, c(2,4:14)]))
```

```
#X <- as.matrix(cbind(rep(1,nrow(trainData)),trainData[, c(1,3:13)]))
predProbs <- 1/(1+exp(-(X%*%coeffs)))


trueClass <- unlist(testData[,3])
#trueClass <- unlist(trainData[,3])

confusionMatrix <- function(resp, pred){
  #if only one class is predicted, there is no accuracy at all
  if(dim(table(pred))==1)
  {
    return(list(accuracy=0, AUC = 0))
  }
  else
  {

    classRes <- data.frame(response = resp , predicted = pred)
    conf = xtabs(~ predicted + response, data = classRes)

    accuracy = sum(diag(conf))/sum(conf)
    accuracy
    precision = conf[1,1]/(conf[1,1]+conf[1,2])
    precision
    recall = conf[1,1]/(conf[1,1]+conf[2,1])
    recall
    Fscore = 2*((precision*recall)/(precision+recall))
    Fscore
    tpr = conf[1,1]/(conf[1,1]+conf[2,1])
    tnr = conf[2,2]/(conf[2,2]+conf[1,2])
    AUC = (tpr + tnr) /2
    return(list(accuracy = accuracy, precision = precision, recall = recall,
Fscore = Fscore, AUC=AUC,conf = conf ))

  }
}

thresholds <- seq(0.05, 0.95, 0.05)
cf <- array(NA,dim =c(length(thresholds),3))
for (i in 1:(length(thresholds))){
  predClass <- as.numeric(predProbs>thresholds[i])
  cf[i,3] <- confusionMatrix(resp = trueClass, pred = predClass)$accuracy
  cf[i,2] <- confusionMatrix(resp = trueClass, pred = predClass)$AUC
  cf[i,1] <- thresholds[i]
}

colnames(cf) <- c("Threshold", "AUC", "Accuracy")
cf
write.csv(cf, "cf_b12_try2.csv" )
```

```
# Best performance using AUC
threshold <- 0.2
predClass <- as.numeric(predProbs>threshold)
confusionMatrix(resp = trueClass, pred = predClass)
preds <- data.frame(name = testData$Patient.ID, probPositive = predProbs, res
p = trueClass, pred = predClass)
write.csv(preds, "preds_logistic_AUC_b12_try2.csv" )

# Best performance using Accuracy
threshold <- 0.5
predClass <- as.numeric(predProbs>threshold)
confusionMatrix(resp = trueClass, pred = predClass)
preds <- data.frame(name = testData$Patient.ID, probPositive = predProbs, res
p = trueClass, pred = predClass)
write.csv(preds, "preds_logistic_Accuracy_b12_try2.csv" )

a <- ggplot(preds, aes(x = probPositive))
a + geom_histogram(aes(color = as.factor(pred), fill = as.factor(pred)),bins
= 100,
                   alpha = 0.4, position = "identity")
```

[B2] – R codes of the Hierarchical Model

```
# Note that the codes for the Hierarchical Model was ran separately for effic
iency purposes. It means that the codes for data preprocessing are repeated a
nd edited as deemed fit in this version.



#===============PRELIMINARY FUNCTIONS FOR POSTERIOR INFERENCES===============
=====
genMCMC = function( data , zName="z" , NName="N" , sName="s" , cName="c" ,
                    numSavedSteps=5000 , saveName=NULL , thinSteps=1 ,
                    runjagsMethod=runjagsMethodDefault ,   useRunjags = TRUE,
                    nChains=nChainsDefault,   burnInSteps = 500 , adaptSteps
= 500 , forInits = NULL    ) {
  require(rjags)
  require(runjags)
  #-------------------------------------------------------------------------
---
  # THE DATA.
  # N.B.: This function expects the data to be a data frame,
  # with one component z being a vector of integer # successes,
  # one component N being a vector of integer # attempts,
  # one component s being a factor of subject identifiers,
  # and one component c being a factor of category identifiers, with
  # subjects nested in categories.
  print(cName)
  z = data[[zName]]
  N = data[[NName]]
  s = data[[sName]]
  c = data[[cName]]
  Nsubj = length(unique(s))
  Ncat =  length(unique(c))
  # Specify the data in a list, for later shipment to JAGS:
  dataList = list(
    z = z ,
    # N = N ,
    c = as.numeric(c) , # c in JAGS is numeric, in R is possibly factor
    Nsubj = Nsubj ,
    Ncat = Ncat
  )
  #-----------------------------------------------------------------------------
---
  # THE MODEL.
  modelString = "
  model {
    for ( sIdx in 1:Nsubj ) {
      z[sIdx] ~ dbern( theta[sIdx])
      theta[sIdx] ~ dbeta( omega[c[sIdx]]*(kappa[c[sIdx]]-2)+1 ,
                           (1-omega[c[sIdx]])*(kappa[c[sIdx]]-2)+1 )
    }


    for ( cIdx in 1:Ncat ) {
      omega[cIdx] ~ dbeta( omegaO*(kappaO-2)+1 ,
                           (1-omegaO)*(kappaO-2)+1 )
      kappa[cIdx] <- kappaMinusTwo[cIdx] + 2
      kappaMinusTwo[cIdx] ~ dgamma( 0.01 , 0.01 ) # mean=1 , sd=10 (generic v
ague)
    }
    #omegaO ~ dbeta( 1.0 , 1.0 )
    #omegaO ~ dbeta( 1 , 99 ) # mode=0 , concentration=100
    #omegaO ~ dbeta( 10.8 , 89.2 ) # mode=0.1 , concentration=100
    #omegaO ~ dbeta( 50 , 50 ) # mode=0.5 , concentration=100
    omegaO ~ dbeta( 89.2 , 10.8 ) # mode=0.9 , concentration=100

    kappaO <- kappaMinusTwoO + 2
    #kappaMinusTwoO ~ dgamma( 0.01 , 0.01 )  # mean=1 , sd=10 (generic vague)
    kappaMinusTwoO ~ dgamma( 1.01005 , 0.01005012 )  # mode=1 , sd=100
    # kappaMinusTwoO ~ dgamma( 2.6 , 26.96 )  # mode=10 , sd=2
    #kappaMinusTwoO ~ dgamma( 1.105125 , 0.1051249 )  # mode=1 , sd=10
    #kappaMinusTwoO ~ dgamma( 1.105125 , 0.01051249 )  # mode=10 , sd=100
    #kappaMinusTwoO ~ dgamma( 5.05 , 101.99 )  # mode=20 , sd=2
    #kappaMinusTwoO ~ dgamma( 0.01 , 1.22)  # mode=20 , sd=100

    #kappaMinusTwoO ~ dgamma( 10.02 , 402 )  # mode=40 , sd=2
    #kappaMinusTwoO ~ dgamma( 0.01 , 1.49 )  # mode=40 , sd=100
  }
  " # close quote for modelString
  writeLines( modelString , con="TEMPmodel.txt" )
  #-----------------------------------------------------------------------------
---
  # INTIALIZE THE CHAINS.
  # Initial values of MCMC chains based on data:
  # initsList = function() {
  #   thetaInit = rep(NA,Nsubj)
  #   for ( sIdx in 1:Nsubj ) { # for each subject
  #     resampledZ = rbinom(1, size=N[sIdx] , prob=z[sIdx]/N[sIdx] )
  #     thetaInit[sIdx] = resampledZ/N[sIdx]
  #   }
  #   thetaInit = 0.001+0.998*thetaInit # keep away from 0,1
  #   kappaInit = 100 # lazy, start high and let burn-in find better value
  #   return( list( theta=thetaInit ,
  #                 omega=aggregate(thetaInit,by=list(c),FUN=mean)$x ,
  #                 omegaO=mean(thetaInit) ,
  #                 kappaMinusTwo=rep(kappaInit-2,Ncat) ,
  #                 kappaMinusTwoO=kappaInit-2 ) )
  # }
  initsList = list( theta=rep(0.5,891) ,
                    omega=rep(0.5,3),
                    omegaO=0.5 ,
                    kappaMinusTwoO=3,
                    kappaMinusTwo=rep(3,3))

  # initsList = list( theta=forInits[9:899,3],
  #                   omega=forInits[1:3,3],
  #                   omegaO=forInits[4,3] ,
  #                   kappaMinusTwoO=forInits[8,3],
  #                   kappaMinusTwo=forInits[5:7,3])
  #-----------------------------------------------------------------------------
---
  # RUN THE CHAINS
  parameters = c( "theta","omega","kappa","omegaO","kappaO")


  if ( useRunjags ) {
    runJagsOut <- run.jags( method=runjagsMethod ,
                            model="TEMPmodel.txt" ,
                            monitor=parameters ,
                            data=dataList ,
                            # inits=initsList ,
                            n.chains=nChains ,
                            adapt=adaptSteps ,
                            burnin=burnInSteps ,
                            sample=ceiling(numSavedSteps/nChains) ,
                            thin=thinSteps ,
                            summarise=FALSE ,
                            plots=FALSE )
    codaSamples = as.mcmc.list( runJagsOut )
  } else {
    # Create, initialize, and adapt the model:
    jagsModel = jags.model( "TEMPmodel.txt" , data=dataList , inits=initsList
,
                            n.chains=nChains , n.adapt=adaptSteps )
    # Burn-in:
    cat( "Burning in the MCMC chain...\n" )
    update( jagsModel , n.iter=burnInSteps )
    # The saved MCMC chain:
    cat( "Sampling final MCMC chain...\n" )
    codaSamples = coda.samples( jagsModel , variable.names=parameters ,
                                n.iter=ceiling(numSavedSteps*thinSteps/nChain
s),
                                thin=thinSteps )
  }

  # resulting codaSamples object has these indices:
  #   codaSamples[[ chainIdx ]][ stepIdx , paramIdx ]
  if ( !is.null(saveName) ) {
    save( codaSamples , file=paste(saveName,"Mcmc.Rdata",sep="") )
  }
  return( codaSamples )
} # end function

#==============================================================================
==========

smryMCMC = function(  codaSamples , compVal=0.5 , rope=NULL ,
                      diffSVec=NULL , diffCVec=NULL ,
                      compValDiff=0.0 , ropeDiff=NULL ,
                      saveName=NULL ) {
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  summaryInfo = NULL
  rowIdx = 0
  # omega:
  for ( parName in grep("omega",colnames(mcmcMat),value=TRUE) ) {
    summaryInfo = rbind( summaryInfo ,
                         summarizePost( mcmcMat[,parName] ,
                                        compVal=compVal , ROPE=rope ) )
    rowIdx = rowIdx+1
    rownames(summaryInfo)[rowIdx] = parName
  }
  # kappa:
  for ( parName in grep("kappa",colnames(mcmcMat),value=TRUE) ) {
    summaryInfo = rbind( summaryInfo ,
                         summarizePost( mcmcMat[,parName] ,
                                        compVal=NULL , ROPE=NULL ) )
    rowIdx = rowIdx+1
    rownames(summaryInfo)[rowIdx] = parName
  }
  # theta:
  for ( parName in grep("theta",colnames(mcmcMat),value=TRUE) ) {
    summaryInfo = rbind( summaryInfo ,
                         summarizePost( mcmcMat[,parName] ,
                                        compVal=compVal , ROPE=rope ) )
    rowIdx = rowIdx+1
    rownames(summaryInfo)[rowIdx] = parName
  }
  # differences of theta's:
  if ( !is.null(diffSVec) ) {
    Nidx = length(diffSVec)
    for ( t1Idx in 1:(Nidx-1) ) {
      for ( t2Idx in (t1Idx+1):Nidx ) {
        parName1 = paste0("theta[",diffSVec[t1Idx],"]")
        parName2 = paste0("theta[",diffSVec[t2Idx],"]")
        summaryInfo = rbind( summaryInfo ,
                             summarizePost( mcmcMat[,parName1]-mcmcMat[,parNa
me2] ,
                                            compVal=compValDiff , ROPE=ropeDi
ff ) )
        rowIdx = rowIdx+1
        rownames(summaryInfo)[rowIdx] = paste0(parName1,"-",parName2)
      }
    }
  }
  # differences of omega's:
  if ( !is.null(diffCVec) ) {
    Nidx = length(diffCVec)
    for ( t1Idx in 1:(Nidx-1) ) {
      for ( t2Idx in (t1Idx+1):Nidx ) {
        parName1 = paste0("omega[",diffCVec[t1Idx],"]")
        parName2 = paste0("omega[",diffCVec[t2Idx],"]")
        summaryInfo = rbind( summaryInfo ,
                             summarizePost( mcmcMat[,parName1]-mcmcMat[,parNa
me2] ,
                                            compVal=compValDiff , ROPE=ropeDi
ff ) )
        rowIdx = rowIdx+1
        rownames(summaryInfo)[rowIdx] = paste0(parName1,"-",parName2)
      }
    }
  }
  # save:
  if ( !is.null(saveName) ) {
    write.csv( summaryInfo , file=paste(saveName,"SummaryInfo.csv",sep="") )
  }
  show( summaryInfo )
  return( summaryInfo )
}
```

```
#==============================================================================
===

plotMCMC = function( codaSamples ,
                     data , zName="z" , NName="N" , sName="s" , cName="c" ,
compVal=0.5 , rope=NULL ,
                     diffSList=NULL , diffCList=NULL ,
                     compValDiff=0.0 , ropeDiff=NULL ,
                     saveName=NULL , saveType="jpg" ) {
  #----------------------------------------------------------------------------
---
  # N.B.: This function expects the data to be a data frame,
  # with one component z being a vector of integer # successes,
  # one component N being a vector of integer # attempts,
  # one component s being a factor of subject identifiers,
  # and one component c being a factor of category identifiers, with
  # subjects nested in categories.
  z = data[[zName]]
  N = data[[NName]]
  s = data[[sName]]
  c = data[[cName]]
  Nsubj = length(unique(s))
  Ncat =  length(unique(c))
  # Now plot the posterior:
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  chainLength = NROW( mcmcMat )

  # kappa:
  parNames = sort(grep("kappa",colnames(mcmcMat),value=TRUE))
  nPanels = length(parNames)
  nCols = 3
  nRows = ceiling(nPanels/nCols)
  openGraph(width=2.5*nCols,height=2.0*nRows)
  par( mfcol=c(nRows,nCols) )
  par( mar=c(3.5,4,3.5,4) , mgp=c(2.0,0.7,0) )
  #xLim = range( mcmcMat[,parNames] )
  xLim=quantile(mcmcMat[,parNames],probs=c(0.000,0.995))
  mainLab = c(paste( levels(factor(data[[cName]]))),"Overall")
  print(paste("levels=", levels(factor(data[[cName]]))))

  mainIdx = 0
  for ( parName in parNames ) {
    mainIdx = mainIdx+1
    print(paste("Kappa Title=", mainLab[mainIdx]))

    postInfo = plotPost( mcmcMat[,parName] , compVal=compVal , ROPE=rope ,
                         xlab=bquote(.(parName)) , cex.lab=1.25 ,
                         main=mainLab[mainIdx] , cex.main=1.5 ,
                         xlim=xLim , border="skyblue" )
  }
  if ( !is.null(saveName) ) {
    saveGraph( file=paste(saveName,"Kappa",sep=""), type=saveType)
  }

  # omega:
  parNames = sort(grep("omega",colnames(mcmcMat),value=TRUE))
  nPanels = length(parNames)
  nCols = 3
  nRows = ceiling(nPanels/nCols)
  openGraph(width=2.5*nCols,height=2.0*nRows)
  par( mfcol=c(nRows,nCols) )
  par( mar=c(3.5,4,3.5,4) , mgp=c(2.0,0.7,0) )
  #xLim = range( mcmcMat[,parNames] )
  xLim=quantile(mcmcMat[,parNames],probs=c(0.001,0.999))
  mainLab = c(paste(levels(factor(data[[cName]]))),"Overall")
  mainIdx = 0
  for ( parName in parNames ) {

    mainIdx = mainIdx+1
    print(paste("Omega Title=", mainLab[mainIdx]))

    postInfo = plotPost( mcmcMat[,parName] , compVal=compVal , ROPE=rope ,
                         xlab=bquote(.(parName)) , cex.lab=1.25 ,
                         main=mainLab[mainIdx] , cex.main=1.5 ,
                         xlim=xLim , border="skyblue" )
  }
  if ( !is.null(saveName) ) {
    saveGraph( file=paste(saveName,"Omega",sep=""), type=saveType)
  }

  # Plot individual omega's and differences:
  if ( !is.null(diffCList) ) {
    for ( compIdx in 1:length(diffCList)) {
      diffCVec = diffCList[[compIdx]]
      Nidx = length(diffCVec)
      temp=NULL
      mainLab = c(paste(levels(factor(data[[cName]]))),"Overall")

      for ( i in 1:Nidx ) {
        temp = c( temp , which(levels(factor((data[[cName]])))==diffCVec[i])
)
      }
      #diffCVec = temp
      print(paste("Nidx=", Nidx, "diffCList", diffCList, "diffCVec", diffCVec
))

      openGraph(width=2.5*Nidx,height=2.0*Nidx)
      par( mfrow=c(Nidx,Nidx) )
      xLim = range(c( compVal, rope,
                      mcmcMat[,paste0("omega[",diffCVec,"]")] ))
      for ( t1Idx in 1:Nidx ) {
        for ( t2Idx in 1:Nidx ) {
          parName1 = paste0("omega[",diffCVec[t1Idx],"]")
          parName2 = paste0("omega[",diffCVec[t2Idx],"]")
          if ( t1Idx > t2Idx) {
            # plot.new() # empty plot, advance to next
            par( mar=c(3,1,3,1) , mgp=c(2.0,0.7,0) , pty="s" )
            nToPlot = 700
            ptIdx = round(seq(1,chainLength,length=nToPlot))

            print(paste("scatter plot x=", diffCVec[t2Idx]))
            print(paste("scatter plot y=", diffCVec[t1Idx]))

            print(paste("scatter plot x=", levels(factor(data[[cName]]))[diff
CVec[t2Idx]]))
            print(paste("scatter plot y=", levels(factor(data[[cName]]))[diff
CVec[t1Idx]]))

            plot ( mcmcMat[ptIdx,parName2] , mcmcMat[ptIdx,parName1] ,
                   cex.main=1.25 , cex.lab=1.25 ,
                     xlab=paste(levels(factor(data[[cName]]))[diffCVec[t2Idx]
]) ,
                     ylab=paste(levels(factor(data[[cName]]))[diffCVec[t1Idx]
]) ,
                   col="skyblue" )
            abline(0,1,lty="dotted")
          } else if ( t1Idx == t2Idx ) {
            par( mar=c(3,3.5,3,1.5) , mgp=c(2.0,0.7,0) , pty="m" )
            postInfo = plotPost( mcmcMat[,parName1] ,
                                 compVal=compVal , ROPE=rope ,
                                 cex.main=1.25 , cex.lab=1.25 ,
                                 xlab=bquote(.(parName1)) ,
                                 main=paste(levels(factor(data[[cName]]))[dif
fCVec[t1Idx]]) ,
                                 xlim=xLim )
          } else if ( t1Idx < t2Idx ) {
            par( mar=c(3,1.5,3,1.5) , mgp=c(2.0,0.7,0) , pty="m" )
            postInfo = plotPost( mcmcMat[,parName1]-mcmcMat[,parName2] ,
                                 compVal=compValDiff , ROPE=ropeDiff ,
                                 cex.main=1.0 , cex.lab=1.25 ,
                                 xlab=bquote("Difference of "*omega*"'s"),
                                  main=paste0(
                                     levels(factor(data[[cName]]))[diffCVec[t
1Idx]] ,
                                     "\n - ",
                                     levels(factor(data[[cName]]))[diffCVec[t
2Idx]]))
          }
        }
      }
      if ( !is.null(saveName) ) {
        saveGraph( file=paste0(saveName,"OmegaDiff",compIdx), type=saveType)
      }
    }
  }

  # Plot individual theta's and differences:
  if ( !is.null(diffSList) ) {
    for ( compIdx in 1:length(diffSList) ) {
      diffSVec = diffSList[[compIdx]]
      Nidx = length(diffSVec)
      temp=NULL
      for ( i in 1:Nidx ) {

            temp = c( temp , data[[sName]]==diffSVec[i])
      }
      #diffSVec = temp
      openGraph(width=2.5*Nidx,height=2.0*Nidx)
      par( mfrow=c(Nidx,Nidx) )
      xLim = range(c(compVal,rope,mcmcMat[,paste0("theta[",diffSVec,"]")],
                     z[diffSVec]/N[diffSVec]))
      for ( t1Idx in 1:Nidx ) {
        for ( t2Idx in 1:Nidx ) {
          parName1 = paste0("theta[",diffSVec[t1Idx],"]")
          parName2 = paste0("theta[",diffSVec[t2Idx],"]")
          if ( t1Idx > t2Idx) {
            # plot.new() # empty plot, advance to next
            par( mar=c(3,3,3,1) , mgp=c(2.0,0.7,0) , pty="s" )
            nToPlot = 700
            ptIdx = round(seq(1,chainLength,length=nToPlot))
            plot ( mcmcMat[ptIdx,parName2] , mcmcMat[ptIdx,parName1] , cex.la
b=1.25 ,
                   xlab=s[diffSVec[t2Idx]] ,
                   ylab=s[diffSVec[t1Idx]] ,
                   col="skyblue" )
            abline(0,1,lty="dotted")
          } else if ( t1Idx == t2Idx ) {
            par( mar=c(3,3.5,3,1.5) , mgp=c(2.0,0.7,0) , pty="m" )
            postInfo = plotPost( mcmcMat[,parName1] ,
                                 compVal=compVal , ROPE=rope ,
                                 cex.main=1.0 , cex.lab=1.25 ,
                                 xlab=bquote(.(parName1)) ,
                                 main=paste0( s[diffSVec[t1Idx]],
                                              " (",c[diffSVec[t1Idx]],")") ,
                                 xlim=xLim )
            # points( z[diffSVec]/N[diffSVec] , 0 ,
            #         pch="+" , col="red" , cex=3 )
            # text( z[diffSVec[t1Idx]]/N[diffSVec[t1Idx]] , 0 ,
            #       bquote(list( z==.(z[diffSVec[t1Idx]]) ,
            #                    N==.(N[diffSVec[t1Idx]]) )) ,
            #        adj=c( (z[diffSVec[t1Idx]]/N[diffSVec[t1Idx]]-xLim[1])/
            #                (xLim[2]-xLim[1]),-3.25) , col="red" )
          } else if ( t1Idx < t2Idx ) {
            par( mar=c(3,1.5,3,1.5) , mgp=c(2.0,0.7,0) , pty="m" )
            postInfo = plotPost( mcmcMat[,parName1]-mcmcMat[,parName2] ,
                                 compVal=compValDiff , ROPE=ropeDiff ,
                                 cex.main=1.0 , cex.lab=1.25 ,
                                 xlab=bquote("Difference of "*theta*"'s") ,
                                 main=paste(
                                   s[diffSVec[t1Idx]] ,
                                   " (",c[diffSVec[t1Idx]],")" ,
                                   "\n -",
                                   s[diffSVec[t2Idx]] ,
                                   " (",c[diffSVec[t2Idx]],")" ) )
            # points( z[diffSVec[t1Idx]]/N[diffSVec[t1Idx]]
            #         - z[diffSVec[t2Idx]]/N[diffSVec[t2Idx]] , 0 ,
            #         pch="+" , col="red" , cex=3 )
          }
        }
      }
      if ( !is.null(saveName) ) {
        saveGraph( file=paste0(saveName,"ThetaDiff",compIdx), type=saveType)
      }
    }
  }
}

#==============PRELIMINARY FUNCTIONS FOR POSTERIOR INFERENCES==============
=====

myData <- read.csv("covid_brazil_final.csv")
describe(myData)

names(myData)[3] <- "covid.results"
myData$covid.results <- as.numeric(as.factor(myData$covid.results)) - 1 # To
get 0/1 instead of 1/2; positive = 1; negative = 0

#check for missing values
sum(is.na(myData))

## [1] 0

zName = "covid.results" # column name for 0,1 values
sName = "Patient.ID" # column name for subject ID
cName = "Age.group"

myData$Age.group= 0
myData$Age.group[which(myData$Patient.age.quantile < 5)] <- 1
myData$Age.group[which(myData$Patient.age.quantile > 4)] <- 2
```

```
myData$Age.group[which(myData$Patient.age.quantile > 9)] <- 3
myData$Age.group[which(myData$Patient.age.quantile > 14)] <- 4

myData$Age.group <- factor(myData$Age.group,
                    levels = c(1,2,3, 4),
                    labels = c("Age Group 1", "Age Group 2", "Age Group 3", "
Age Group 4"))

write.csv(myData, "myData.csv" )

tab <- table(myData$Age.group, myData$covid.results)
tab <- cbind(tab, Total = rowSums(tab))


numAdaptInSteps = 8000 ; numBburnInSteps=8000; numSavedSteps=8000 ; thinSteps
=800; nChains = 3


startTime = proc.time()


mcmcCoda = genMCMC( data=myData ,
                    zName=zName, sName=sName, cName=cName,
                    numSavedSteps=numSavedSteps ,   useRunjags = TRUE,
                    thinSteps=thinSteps , burnInSteps = numBburnInSteps , ada
ptSteps = numAdaptInSteps ,nChains = nChains)

stopTime = proc.time()
duration = stopTime - startTime
show(duration)

#save.image(file="rEnvironment_800Thin_8thousands_Age_trial3.RData")
#load(file="rEnvironment_800Thin_8thousands_Age_trial3.RData")

#-----------------------------------------------------------------------------
---
# Display diagnostics of chain, for specified parameters:
#
#-------------------------
parameterNames = varnames(mcmcCoda) # get all parameter names for reference
for ( parName in c("omega[1]","omega[2]","omega[3]","omega[4]","omegaO","kapp
a[1]","kappa[2]","kappa[3]","kappa[4]","kappaO","theta[1]","theta[2]","theta[
3]") ) {
  diagMCMC( codaObject=mcmcCoda , parName=parName)
}

# Get summary statistics of chain:

summaryInfo = smryMCMC( mcmcCoda , compVal=NULL ,
                        diffSVec=c(9,19, 29,39) ,
                        diffCVec=c(1,2,3,4) ,
                        compValDiff=0.0)

# Display posterior information:
plotMCMC( mcmcCoda , data=myData ,
          zName=zName, sName=sName, cName=cName,
          compVal=NULL ,
          diffCList=list( c(1,2) ,
                          c(1,3) ,
                          c(1,4),
                          c(2,3),
                          c(2,4),
                          c(3,4)) ,
          diffSList=list( c(8,7),
                          c(8,9) ,
                          c(9,11)) ,
                          compValDiff=0.0) #ropeDiff = c(-0.05,0.05))

write.csv(summaryInfo, "summaryInfoHierarchicalAge_trial16.csv" )
```

```
# ============ Predictive check ============
confusionMatrix <- function(resp, pred){
  #if only one class is predicted, there is no accuracy at all
  if(dim(table(pred))==1)
  {
    return(list(accuracy=0, AUC = 0))
  }
  else
  {

    classRes <- data.frame(response = resp , predicted = pred)
    conf = xtabs(~ predicted + response, data = classRes)

    accuracy = sum(diag(conf))/sum(conf)
    accuracy
    precision = conf[1,1]/(conf[1,1]+conf[1,2])
    precision
    recall = conf[1,1]/(conf[1,1]+conf[2,1])
    recall
    Fscore = 2*((precision*recall)/(precision+recall))
    Fscore
    tpr = conf[1,1]/(conf[1,1]+conf[2,1])
    tnr = conf[2,2]/(conf[2,2]+conf[1,2])
    AUC = (tpr + tnr) /2
    return(list(accuracy = accuracy, precision = precision, recall = recall,
Fscore = Fscore, AUC=AUC,conf = conf ))

  }
}
```

```
# Searching threshold between 0.1 to 0.95
thresholds <- seq(0.1, 0.95, 0.05)
cf <- array(NA,dim =c(length(thresholds),3))
for (i in 1:(length(thresholds))){
  predProbs <- summaryInfo[11:608,3]#summaryInfo[9:208,3]
  predcovidResults <- array(1, 598)#array(1, 200)
  preds <- data.frame(name = myData$Patient.ID, probPositive = predProbs, cov
idResultPred = predcovidResults, covidResultActual = myData$covid.results)
  preds$covidResultPred[which(preds$probPositive < thresholds[i])] <- 0 # App
ly each threshold to estimate those not survived
  #write.csv(preds, paste("pred_", i, ".csv" ))
  print(paste("i=", i, "thresholds", thresholds[i], "dimension", dim(table(pr
eds$covidResultPred))))
  confusionMatrix(resp = preds$covidResultActual, pred = preds$covidResultPre
d)
  cf[i,3] <- confusionMatrix(resp = preds$covidResultActual, pred = preds$cov
idResultPred)$accuracy
  cf[i,2] <- confusionMatrix(resp = preds$covidResultActual, pred = preds$cov
idResultPred)$AUC
  cf[i,1] <- thresholds[i]
}

# Searching threshold between 0.01 to 0.2
# Searching threshold between 0.01 to 0.15 - for Omega prior

thresholds <- seq(0.01, 0.2, 0.01)
thresholds <- seq(0.01, 0.15, 0.005)

cf <- array(NA,dim =c(length(thresholds),3))
for (i in 1:(length(thresholds))){
  predProbs <- summaryInfo[11:608,3]#summaryInfo[9:208,3]
  predcovidResults <- array(1, 598)#array(1, 200)
  preds <- data.frame(name = myData$Patient.ID, probPositive = predProbs, cov
idResultPred = predcovidResults, covidResultActual = myData$covid.results)
  preds$covidResultPred[which(preds$probPositive < thresholds[i])] <- 0 # App
ly each threshold to estimate those not survived
  #write.csv(preds, paste("pred_", i, ".csv" ))
  print(paste("i=", i, "thresholds", thresholds[i], "dimension", dim(table(pr
eds$covidResultPred))))
  confusionMatrix(resp = preds$covidResultActual, pred = preds$covidResultPre
d)
  cf[i,3] <- confusionMatrix(resp = preds$covidResultActual, pred = preds$cov
idResultPred)$accuracy
  cf[i,2] <- confusionMatrix(resp = preds$covidResultActual, pred = preds$cov
idResultPred)$AUC
  cf[i,1] <- thresholds[i]
}
```

```r
predProbs <- summaryInfo[11:608,3]#summaryInfo[9:208,3]
predcovidResults <- array(1, 598)#array(1, 200)
preds <- data.frame(name = myData$Patient.ID, Age.group=myData$Age.group,   p
robPositive = predProbs, covidResultPred = predcovidResults, covidResultActua
l = myData$covid.results)
preds$covidResultPred[which(preds$probPositive < 0.18)] <- 0 # Apply threshol
d of 0.135 to estimate those not survived
confusionMatrix(resp = preds$covidResultActual, pred = preds$covidResultPred)

preds$label= 0
preds$label[which(preds$Age.group=="Age Group 1" & preds$covidResultActual ==
0)] <- 1
preds$label[which(preds$Age.group=="Age Group 1" & preds$covidResultActual ==
1)] <- 2
preds$label[which(preds$Age.group=="Age Group 2" & preds$covidResultActual ==
0)] <- 3
preds$label[which(preds$Age.group=="Age Group 2" & preds$covidResultActual ==
1)] <- 4
preds$label[which(preds$Age.group== "Age Group 3" & preds$covidResultActual =
= 0)] <- 5
preds$label[which(preds$Age.group=="Age Group 3" & preds$covidResultActual ==
1)] <- 6
preds$label[which(preds$Age.group=="Age Group 4" & preds$covidResultActual ==
0)] <- 7
preds$label[which(preds$Age.group=="Age Group 4" & preds$covidResultActual ==
1)] <- 8


for(i in 1:8)
{
  minNum=min(preds[preds$label==i,]$probPositive)
  maxNum=max(preds[preds$label==i,]$probPositive)
  print(paste(minNum, maxNum))

}

preds$label <- factor(preds$label,
                      levels = c(1,2,3,4,5,6,7,8),
                      labels = c("Age Group 1 -", "Age Group 1 +",
                           "Age Group 2 -", "Age Group 2 +",
                           "Age Group 3 -", "Age Group 3 +",
                           "Age Group 4 -", "Age Group 4 +"))

write.csv(preds, paste("pred_final_trial16", ".csv" ))

a <- ggplot(preds, aes(x = probPositive))
a + geom_histogram(aes(color = label, fill = label),bins = 100,
                   alpha = 0.4, position = "identity") +
  geom_density(aes(y=..count../500,color= label), size = 1) +
  scale_fill_manual(values = c("#98c068", "#287028", "#f4d1d7", "#802939",
                               "#63bce5", "#114da9", "#fff897", "#f8e000")) +
  scale_color_manual(values = c("#98c068", "#287028", "#f4d1d7", "#802939",
                                "#63bce5", "#114da9", "#fff897", "#f8e000"))
+
  ggtitle(label="Distribution of patients (mode of +ve probability) for diffe
rent age groups", subtitle="(modeW=0.1, concentrationW=100, modeK=10, s.d.K=2
)") +
  xlab("Positive Probability") + ylab("Count") +
  theme(plot.title = element_text(hjust = -0.3, size=16),plot.subtitle = elem
ent_text(hjust = 0.5, size=14))
```

References

Formica, V., Minieri, M., Bernardini, S., Ciotti, M., D'Agostini, C., Roselli, M., Andreoni, M., Morelli, C., Parisi, G., Federici, M., Paganelli, C. and Legramante, J., 2020. Complete blood count might help to identify subjects with high probability of testing positive to SARS-CoV-2. *Clinical Medicine*, 20(4), pp.e114-e119.

Kaggle.com. 2020. *Diagnosis Of COVID-19 And Its Clinical Spectrum*. [online] Available at: <https://www.kaggle.com/einsteindata4u/covid19> [Accessed 21 September 2020].

Soltan, A.A., Kouchaki, S., Zhu, T., Kiyasseh, D., Taylor, T., Hussain, Z.B., Peto, T., Brent, A.J., Eyre, D.W. and Clifton, D., 2020. Artificial intelligence driven assessment of routinely collected healthcare data is an effective screening test for COVID-19 in patients presenting to hospital. *medRxiv*.