Project_v3

2023-04-26

```
## Warning: package 'stableGR' was built under R version 4.2.3
```

```
## Warning: package 'mcmcse' was built under R version 4.2.3
```

Introduction

We have been provided with data from 106 experimental runs where various amounts of reactants were subjected to heat and pressure to produce a hydroformylated aldehydes. The final product comes in two structures, linear (A) and branched (B). Predetermining conditions required to create desired product composition would greatly benefit researchers. A robust model would aid researchers in increasing the efficiency of their experiments.

The data itself consists of 7 explanatory variables (Carbon Monoxide and Hydrogen flow rates, Temperature, Pressure, solvent ratio, ligand-to-rhodium ratio, and olefin-to-rhodium ratio) and two response variables (total yield and selectivity of A). Exploratory data analysis revealed significant correlation between a number of variables, more specifically, between the two response variables. It was found that yield decreases with increasing selectivity for product A. Principal component analysis showed some clustering based on desired outcomes which indicates a robust model may exist.

Reaction Data

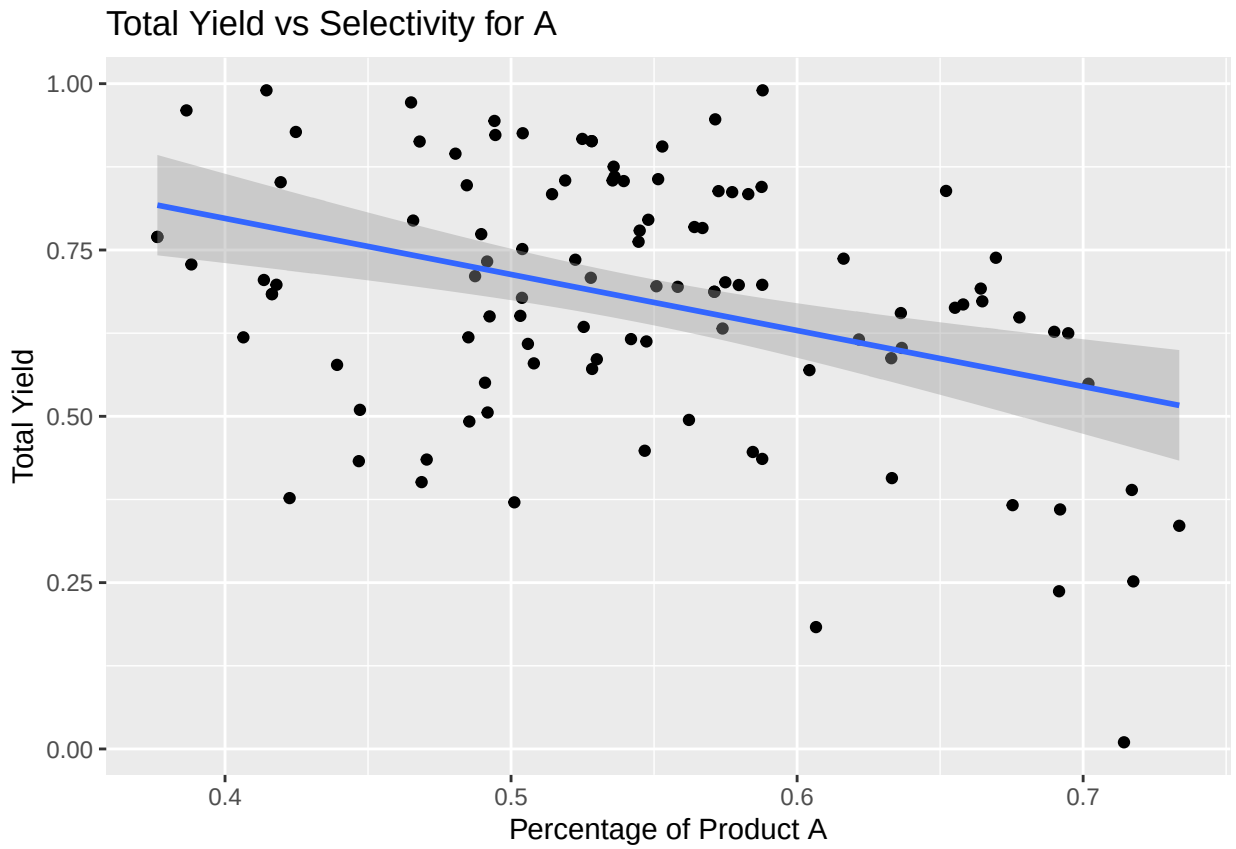
In each run, at least one of the following input parameters were varied:

Variable	Variable description
CO	Carbon Monoxide flow rate
H	Hydrogen Flow Rate
Pr	Pressure
T	Temperature
SolR	Solvent ratio
LigRhoR	Ligand-to-Rhodium Ratio
OctRhoR	Olefin-to-Rhodium Ratio

The output for each experimental run will be denoted by `Yield` for Total Yield and `ASel` for Selectivity for Product A.

Initial exploratory analysis indicate that there is a negative correlation between Selectivity for A and total yield. producing the following linear relationship:

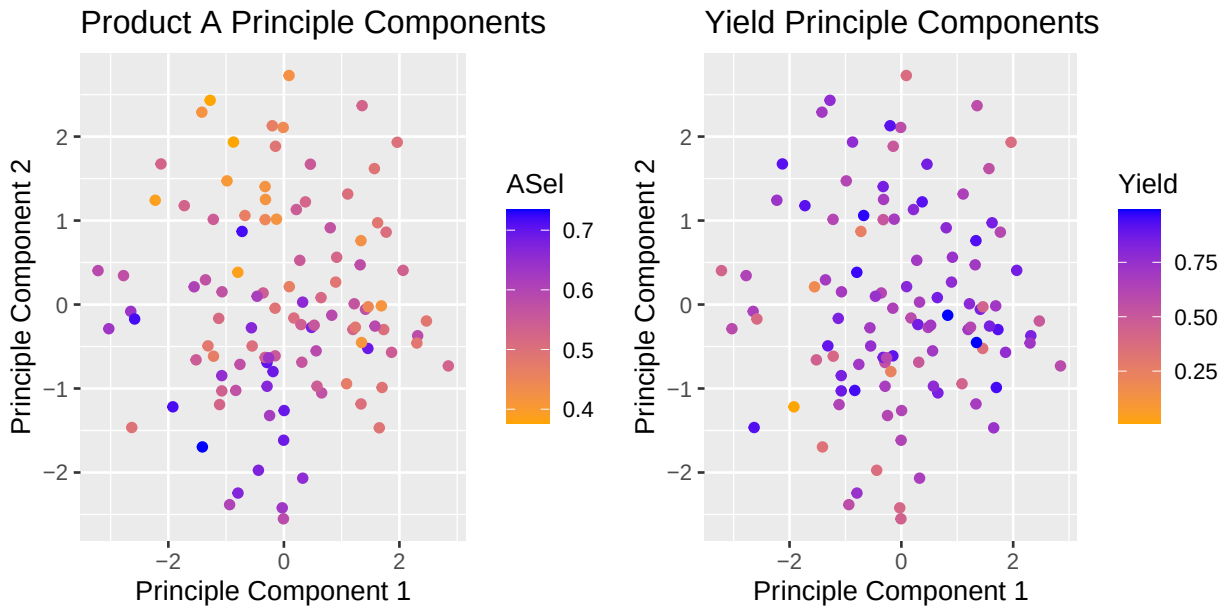
$Yield = 1.30 - 1.03\ ASel$



The following correlation matrix indicates, not only the initial correlation between the dependent variables, autocorrelation between the input variables. This observation highlights to the complexity of this chemical reaction and must be accounted for in the final model.

	H	Pr	T	SolR	LigRhoR	OctRhoR	Yield	ASel
H	1.000	0.161	-0.075	0.207	0.108	-0.011	-0.052	0.257
Pr	0.161	1.000	0.123	0.225	-0.071	-0.102	0.383	-0.057
T	-0.075	0.123	1.000	0.041	-0.132	-0.088	-0.024	-0.672
SolR	0.207	0.225	0.041	1.000	-0.153	-0.053	-0.179	0.165
LigRhoR	0.108	-0.071	-0.132	-0.153	1.000	0.373	-0.140	0.259
OctRhoR	-0.011	-0.102	-0.088	-0.053	0.373	1.000	-0.250	0.331
Yield	-0.052	0.383	-0.024	-0.179	-0.140	-0.250	1.000	-0.379
ASel	0.257	-0.057	-0.672	0.165	0.259	0.331	-0.379	1.000

Due to the large number of predictor variables we plotted the outputs against the first two principle components of a PCA. This dimension reduction technique allows us to look for any possible groupings between the experimental runs in a concise plot.



Observing the PCA plots, there is some evidence of grouping based on both selectivity and yield. This indicates that once an adequate model has been developed, parameter optimization can lead us to experimental settings that maximize selectivity for A or B, while maintaining high yields.

Bayesian Linear Analysis

The general form of the model used for linear regression is:

$$Y_i = \alpha_j + \sum_{j=1}^{p} X_i \beta_j + \epsilon_i$$

Where i is 1 = 106, Y is Yield or Selectivity or Yield A or Yield B and j = 1 to 7 for each experimental condition

Conducting a simple Bayesian linear analysis the following coefficient values are derived;

Bayesian Lasso Model

The second model incorporates the LASSO method to penalize the size of the coefficients and address any autocorrelation that might exist.

Baysien LASSO with Random Effects

BLASSO Regression using Expanded Dataset

In this model we introduce quadratic and interaction elements into the model. The additional variables are CO^2, H^2, Pr^2, T^2, $OctRhoR^2$, $Pr:CO$, $T:CO$, $Pr:H$, $T:H$, and $Pr:T$. It is suspected the model surface contains multiple local maxima and minima locations and the quadratic components will give the model more flexibility. The interation compoents were added to account for physical phenomenon of the reactions conditions. For example, pressure and temperature effect the solubility of the two gases (CO and H_2) which directly impact the amounts of those reactants that will reach the olefin. Also, pressure and temperture have an effect on each other which will be represented by their respective interaction term.

BLASSO Regression using Fully Expanded Dataset and tranformed outputs

BLASSO Regression using Fully Expanded Dataset and tranformed outputs with correlation priors

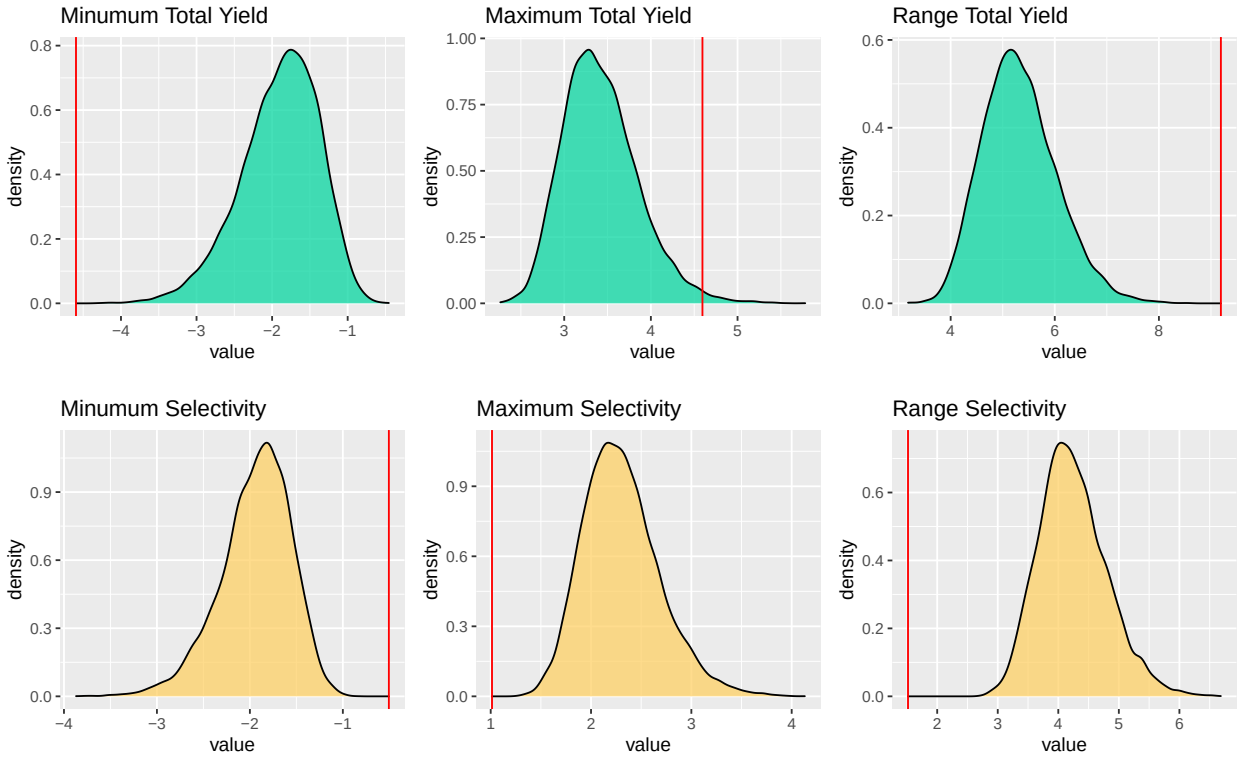
Model Comparisons

To compare the simple versus LASSO linear regression models we can take a look at the calculated DIC and WAIC values. The expanded dataset model showed the lowest WAIC and penalized DIC values.

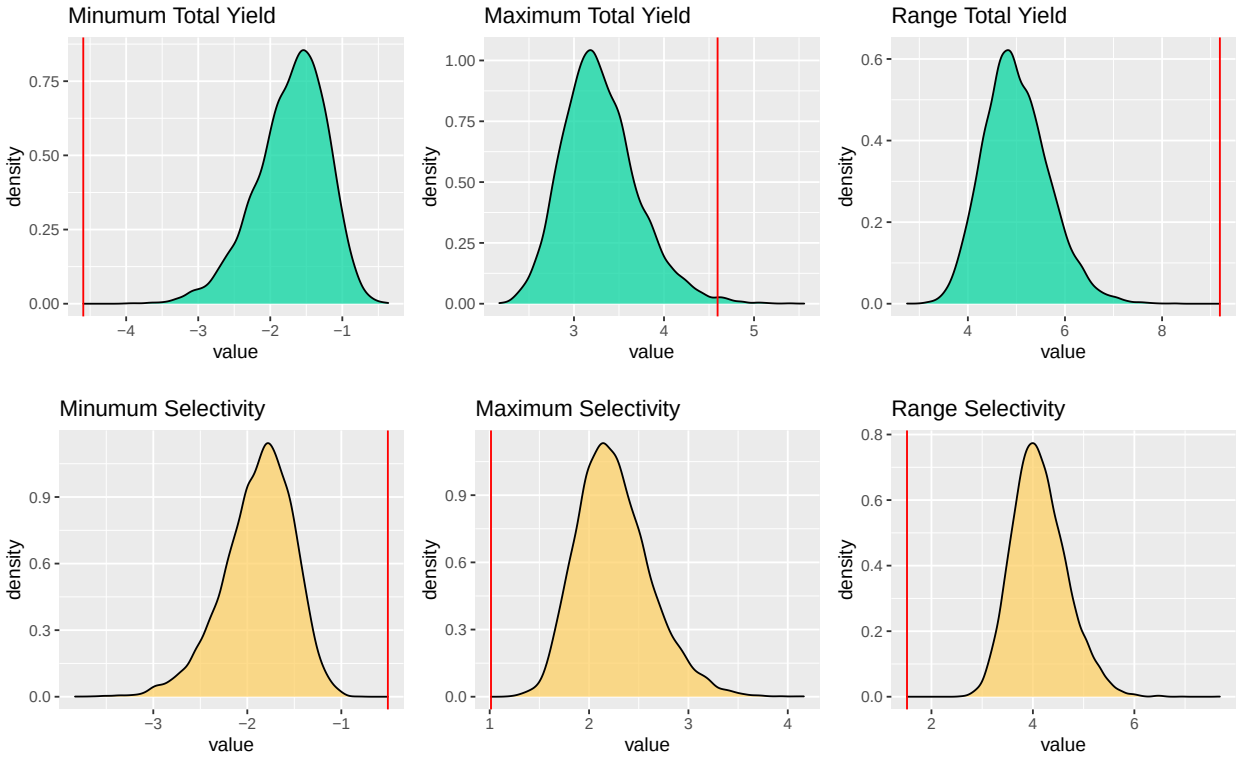
Model	WAIC	DIC
Simple	510.0655423	501.6714762
BLASSO	509.1106364	500.4578875
BLASSO with RE	508.6528722	506.1302029
BLASSO Expanded	503.5721781	491.3630566
BLASSO Fully Expanded	387.2356009	357.5594918
BLASSO Fully Expanded with Correlation	253.9805922	237.5108891

Model Fit

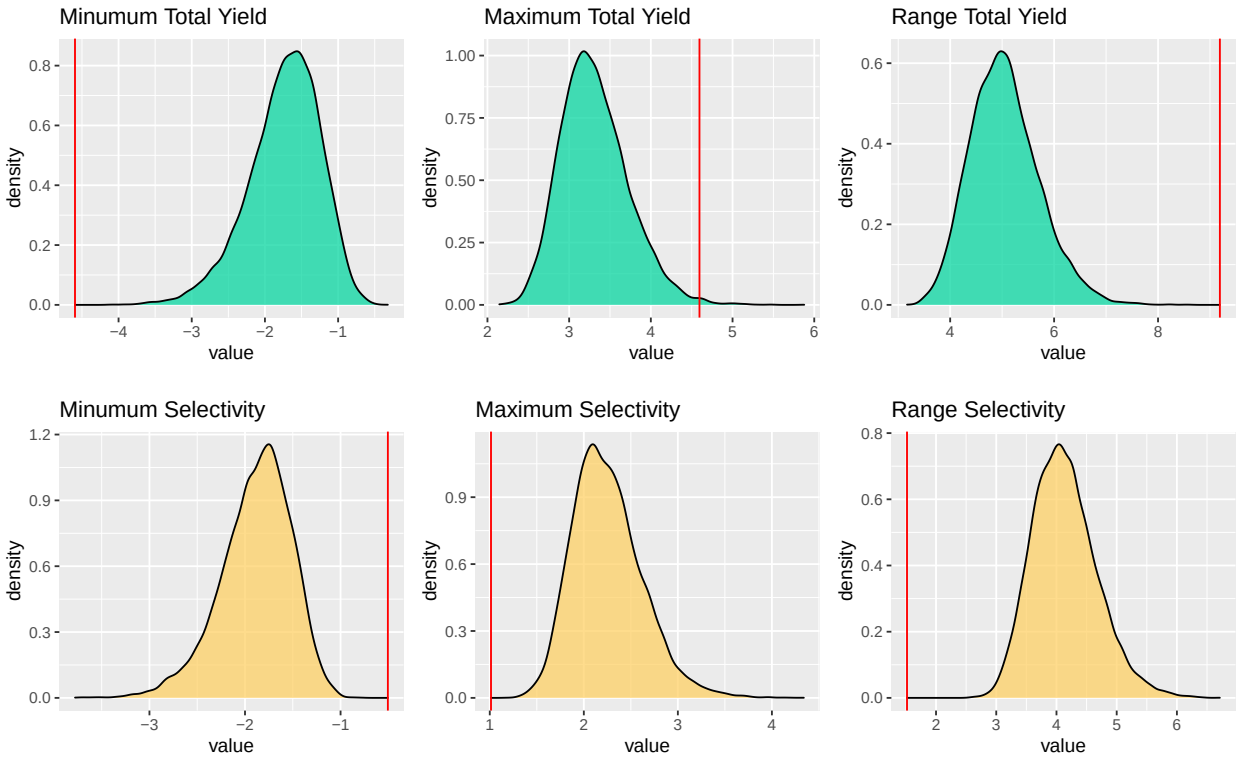
Model 1



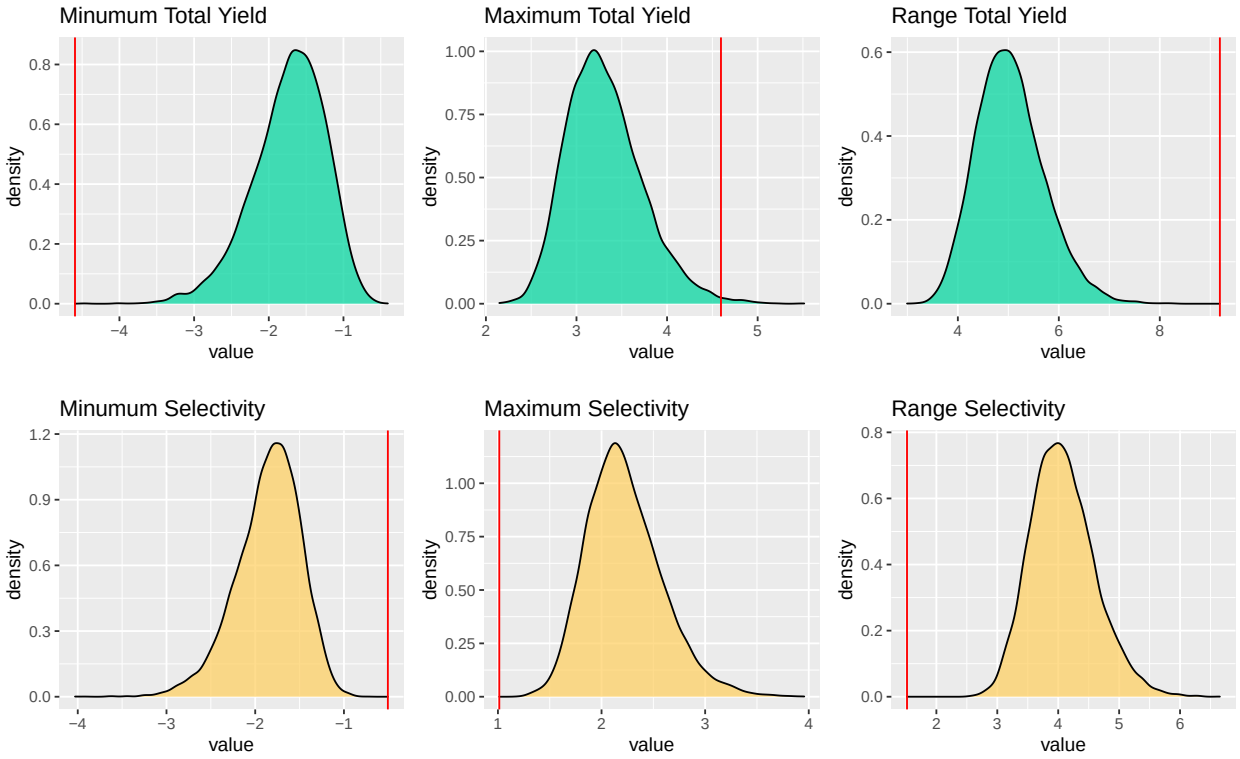
Model 2



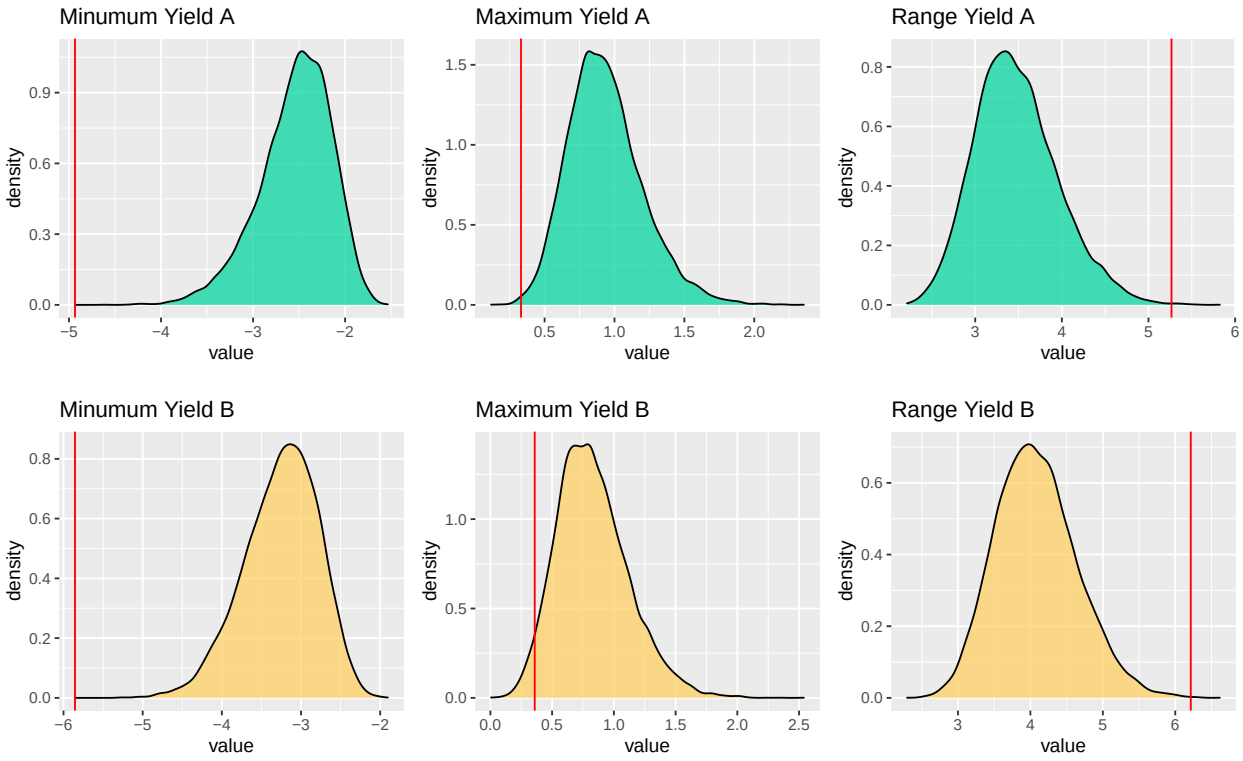
Model 3



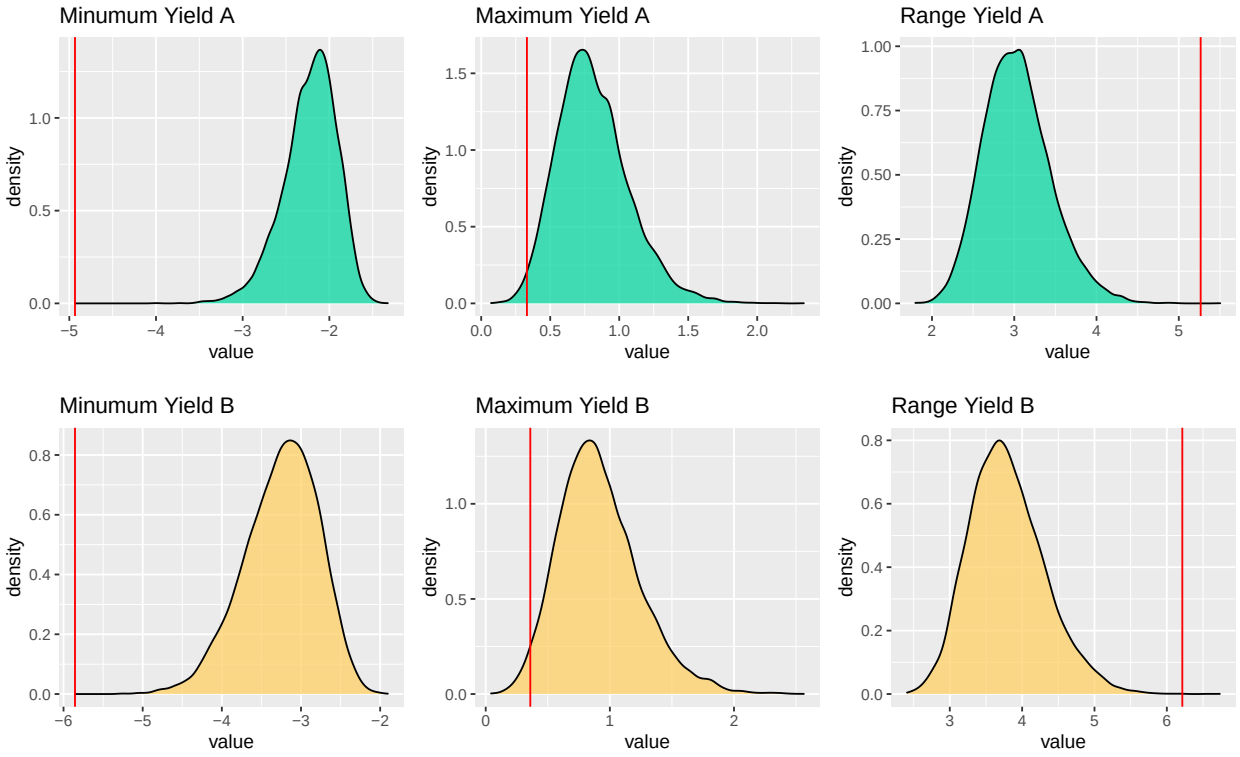
Model 4



Model 5

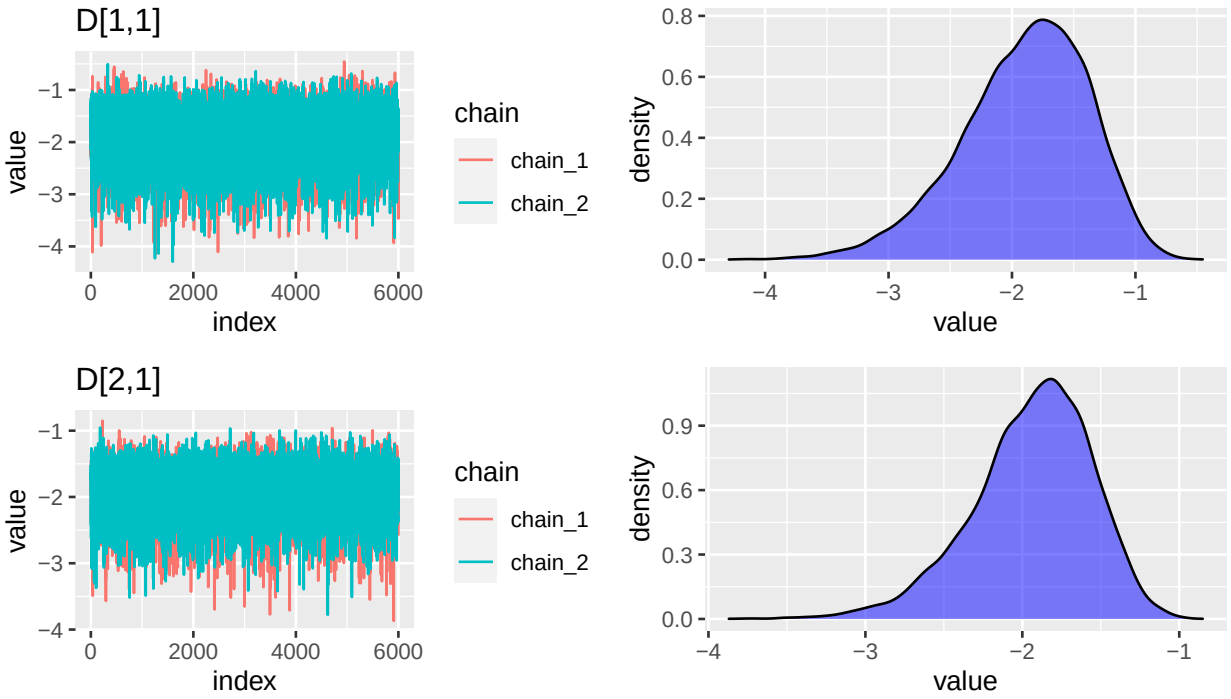


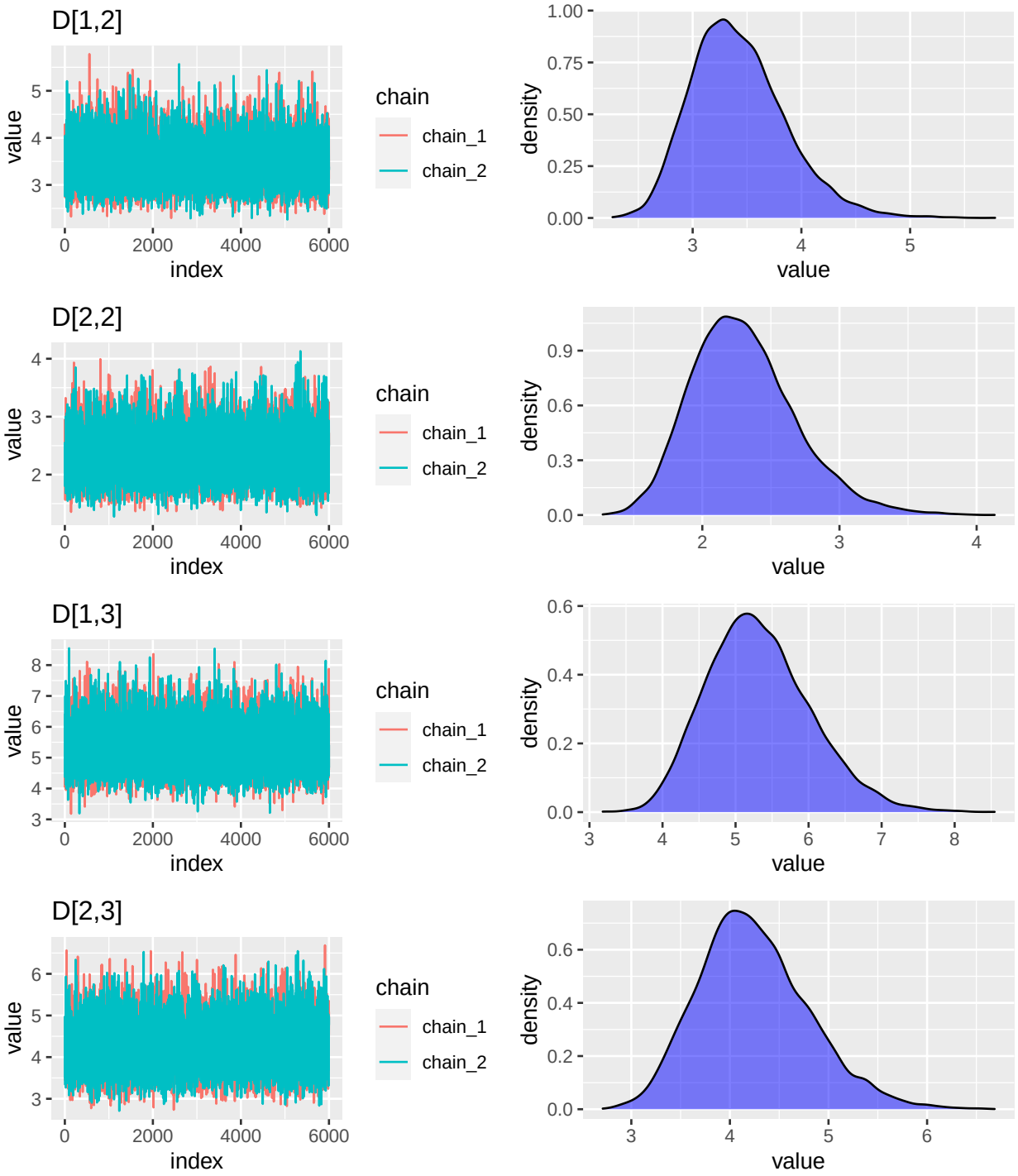
Model 6

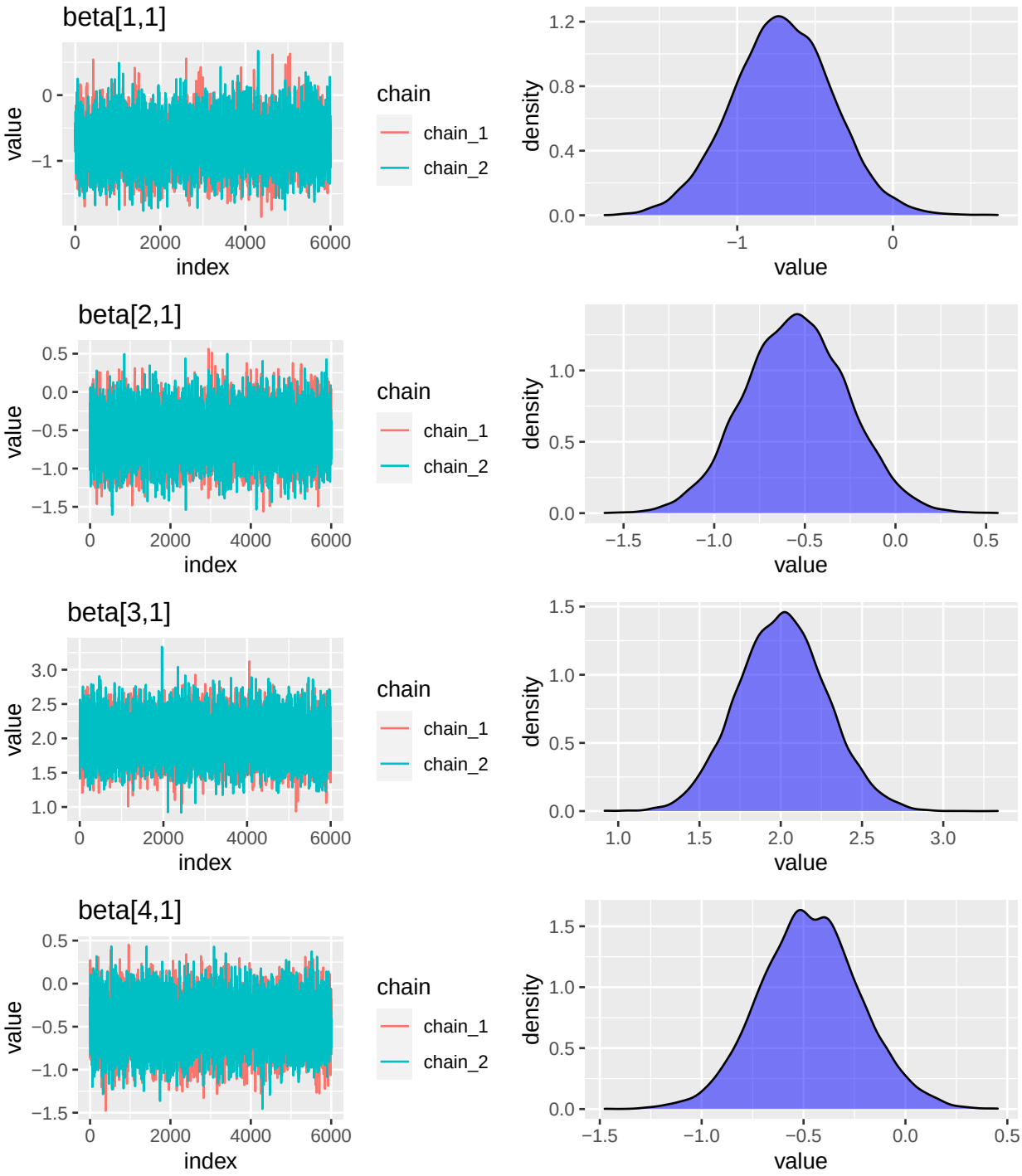


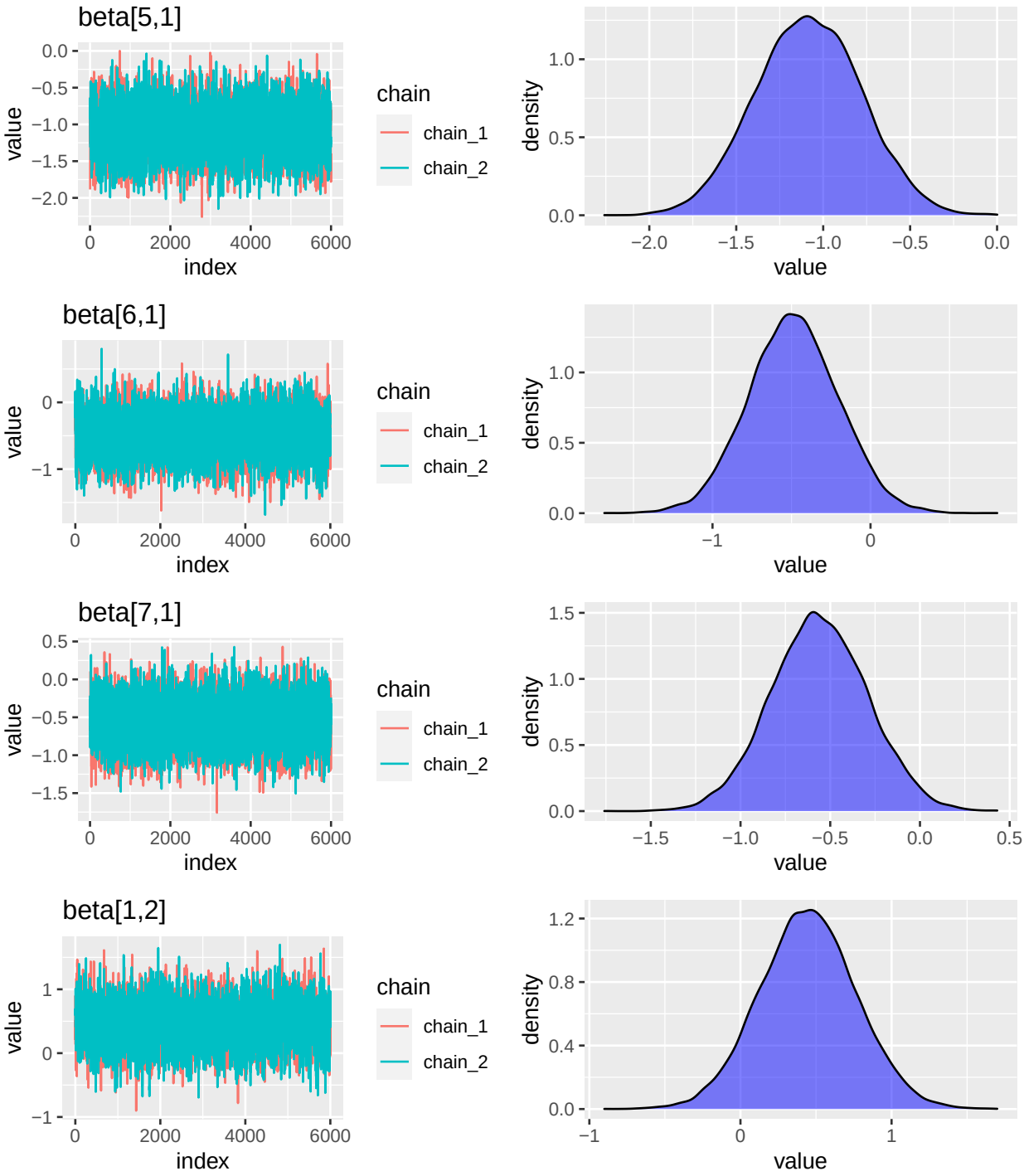
Trace Plots

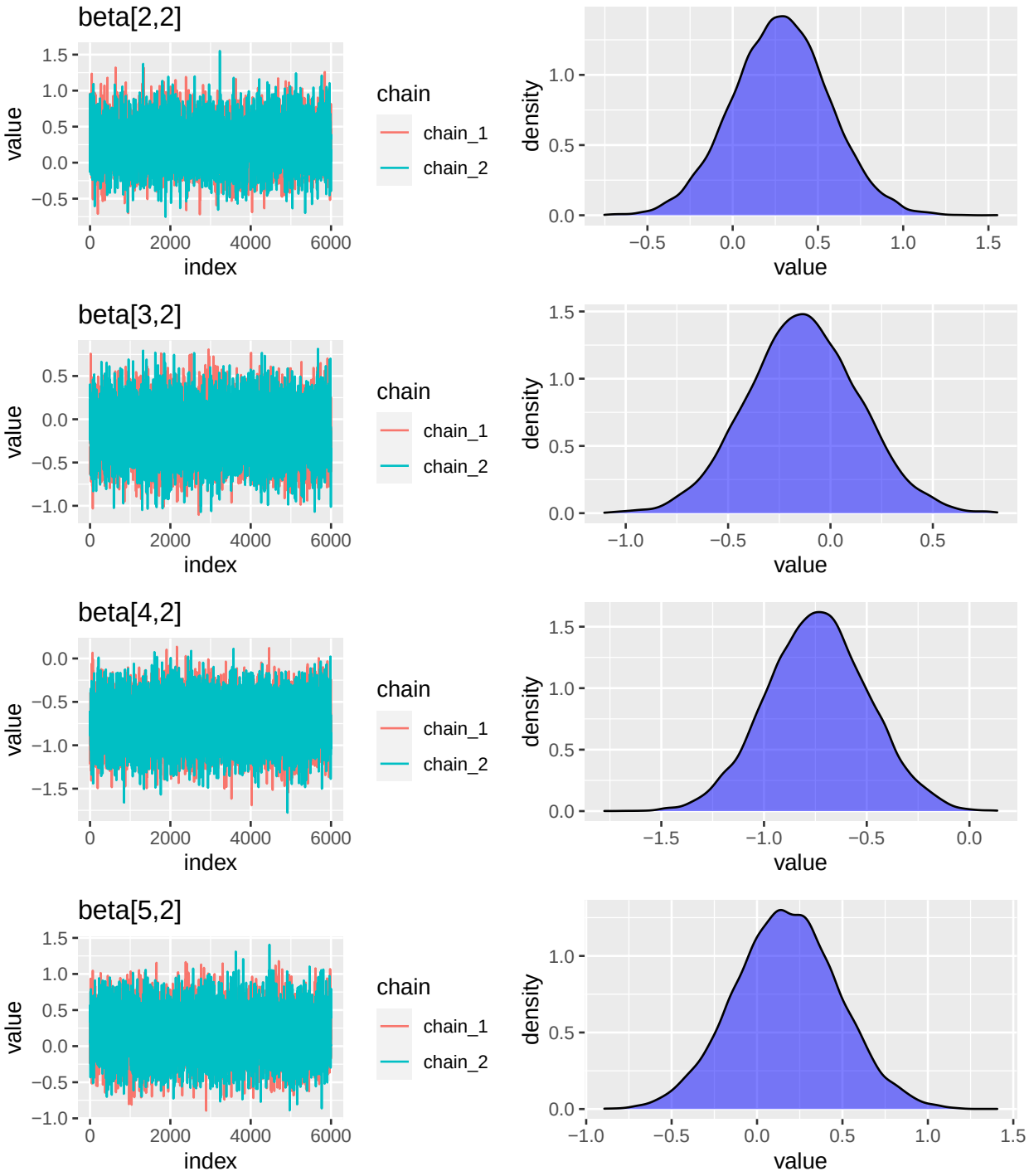
Bayesian Simple Linear Regression

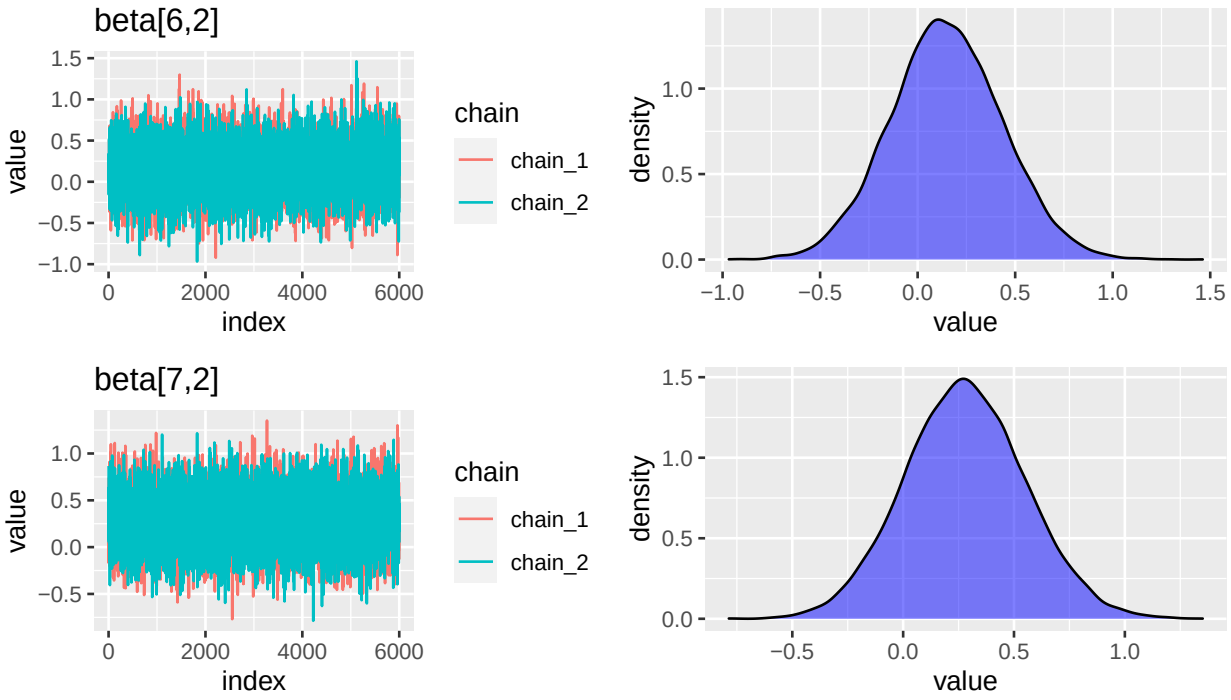



D[1,2]
value
index
chain
chain_1
chain_2
density
value
D[2,2]
value
index
chain
chain_1
chain_2
density
value
D[1,3]
value
index
chain
chain_1
chain_2
density
value
D[2,3]
value
index
chain
chain_1
chain_2
density
value


beta[1,1]
value
index
chain
chain_1
chain_2
density
value
beta[2,1]
value
index
chain
chain_1
chain_2
density
value
beta[3,1]
value
index
chain
chain_1
chain_2
density
value
beta[4,1]
value
index
chain
chain_1
chain_2
density
value

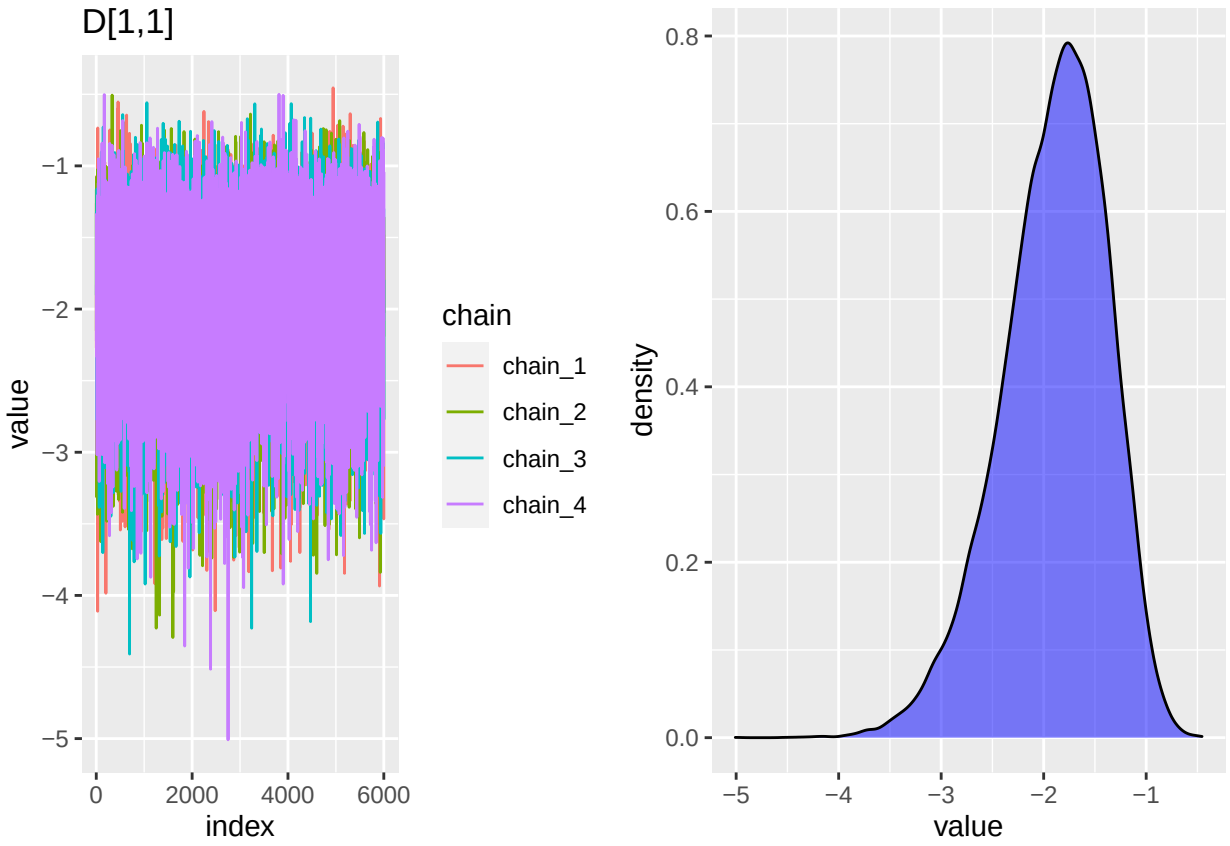

beta[5,1]
value
index
chain
chain_1
chain_2
density
value
beta[6,1]
value
index
chain
chain_1
chain_2
density
value
beta[7,1]
value
index
chain
chain_1
chain_2
density
value
beta[1,2]
value
index
chain
chain_1
chain_2
density
value

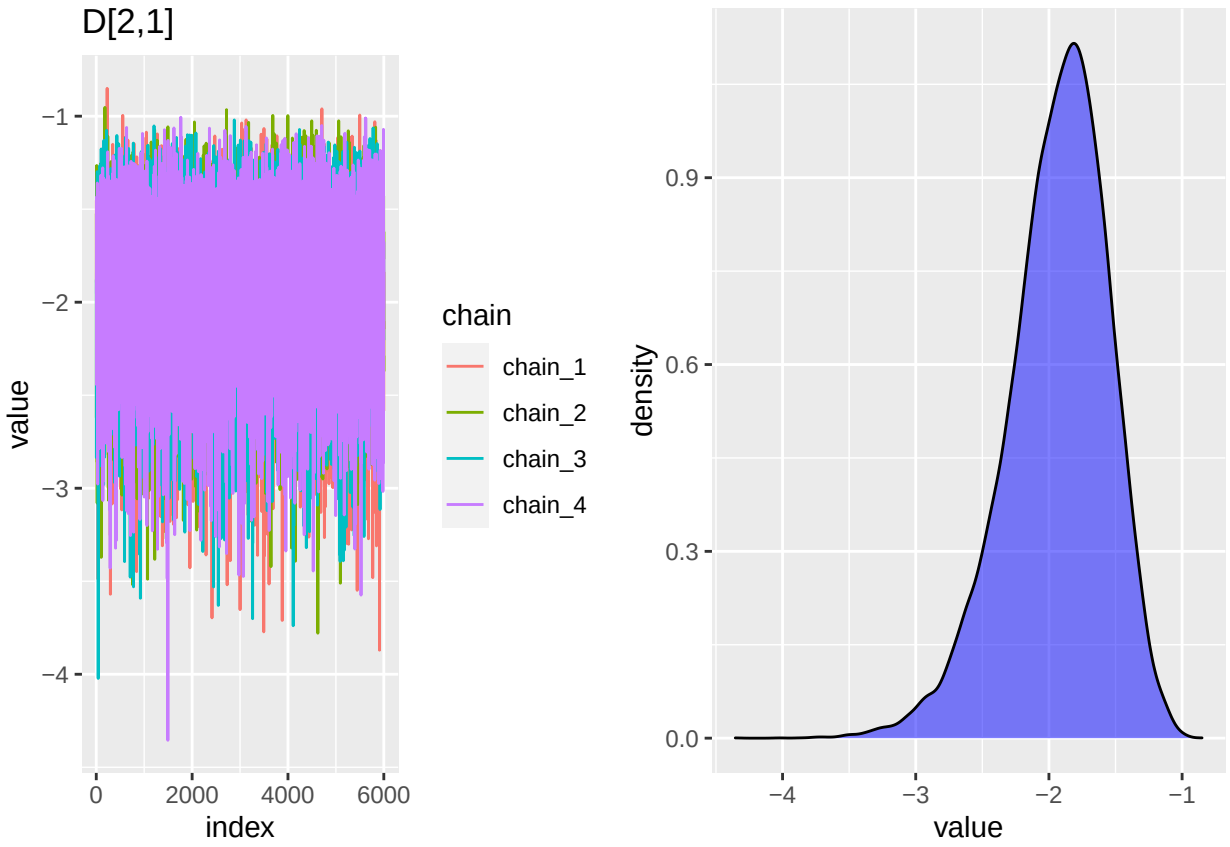

beta[2,2]
value
index
chain
chain_1
chain_2
density
value
beta[3,2]
value
index
chain
chain_1
chain_2
density
value
beta[4,2]
value
index
chain
chain_1
chain_2
density
value
beta[5,2]
value
index
chain
chain_1
chain_2
density
value

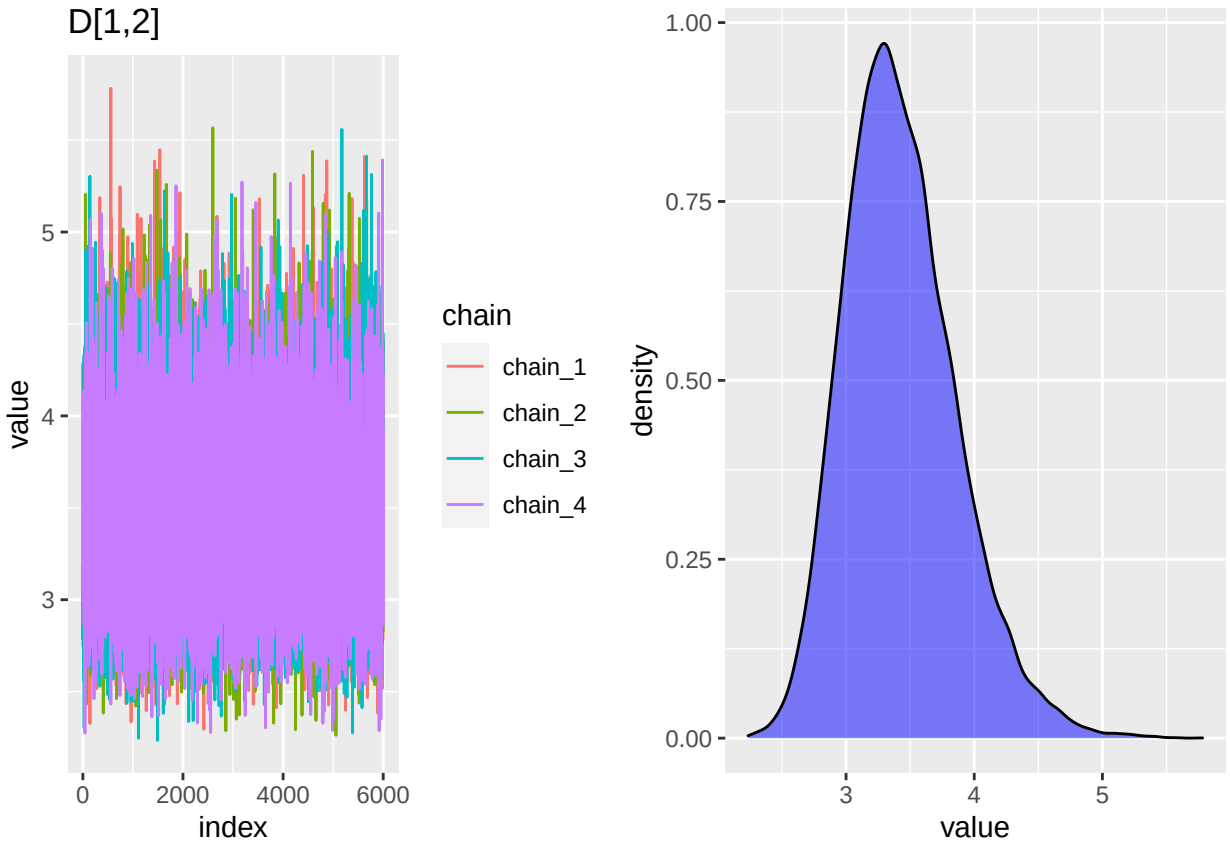


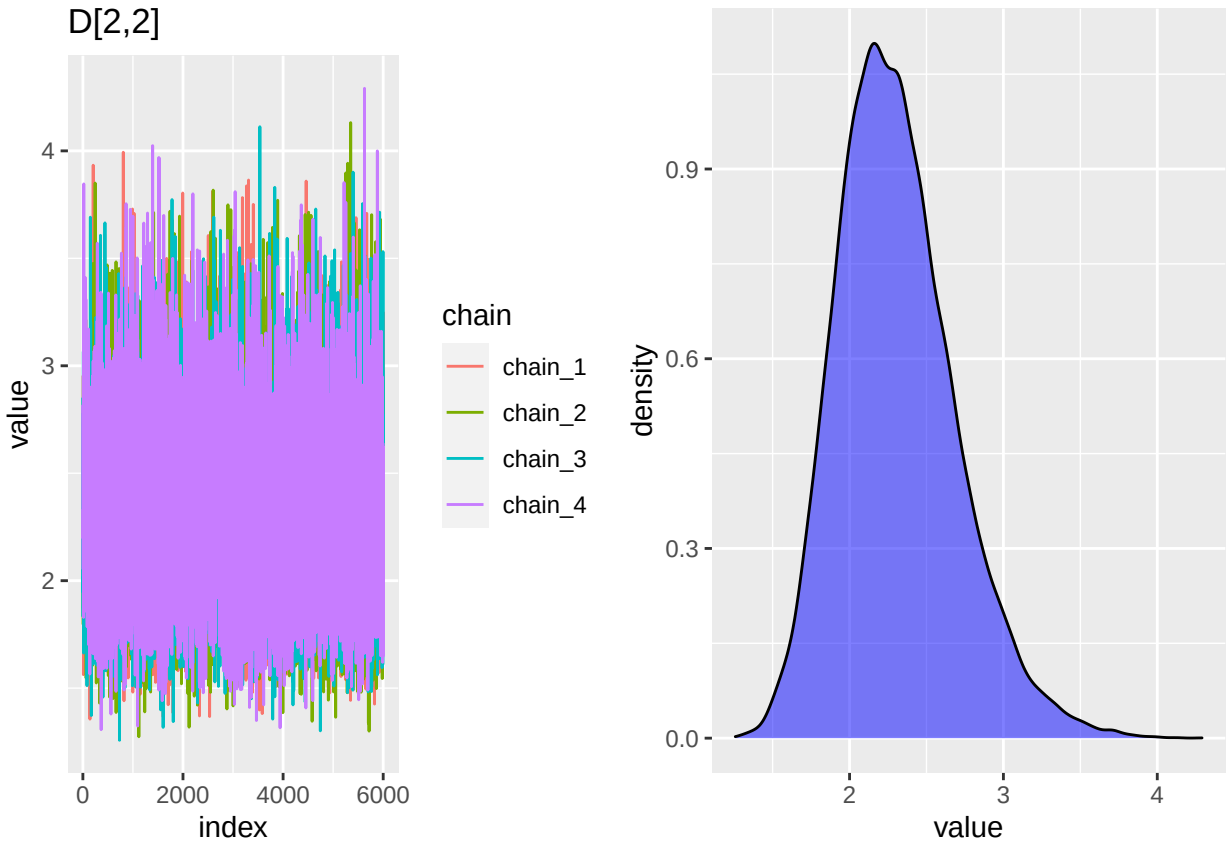

```
param_names <- rownames(summary(samples1[[1]])[[1]])

n <- length(param_names)
n_index <- 1:nrow(summary(samples1)[[1]])

for(i in 1:n){

    j <- n_index[i]

    df_temp <- cbind(1:length(unlist(samples1[1][,j])),
                     unlist(samples1[1][,j]),
                     unlist(samples1[2][,j]),
                     unlist(samples1[3][,j]),
                     unlist(samples1[4][,j]))

    colnames(df_temp) <- c('index', 'chain_1', 'chain_2', 'chain_3', 'chain_4')

    df_temp <- as_tibble(df_temp)

    plot1 <-

    df_temp %>% pivot_longer(c(chain_1, chain_2, chain_3, chain_4), values_to = 'value', names_to = 'cha
        ggplot() + geom_line(aes(x = index, y = value, col = chain)) + labs(title = param_names[i])

    plot2 <-

    df_temp %>% pivot_longer(c(chain_1, chain_2, chain_3, chain_4), values_to = 'value', names_to = 'cha
        ggplot() + geom_density(aes(value), fill = 'blue', alpha = 0.5)

    print(plot_grid(plot1, plot2))
```

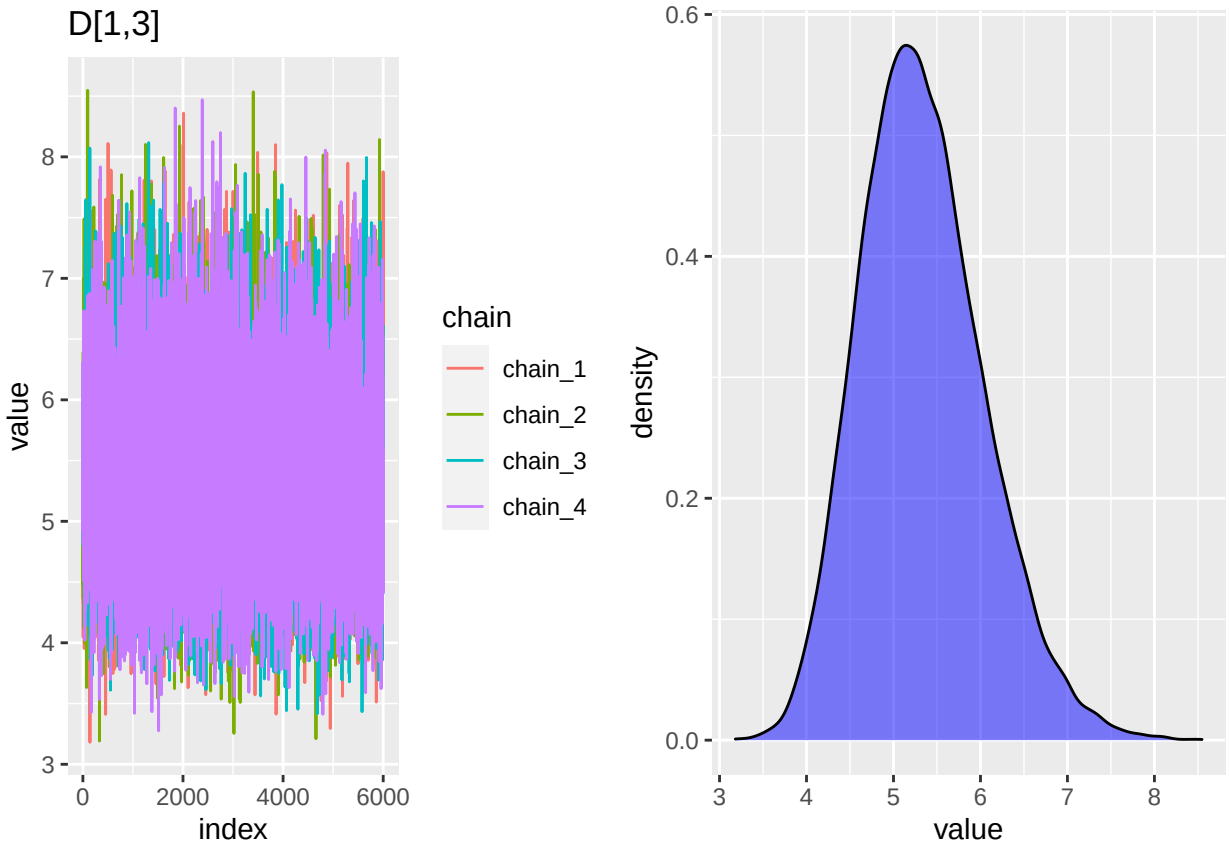
```
}
```

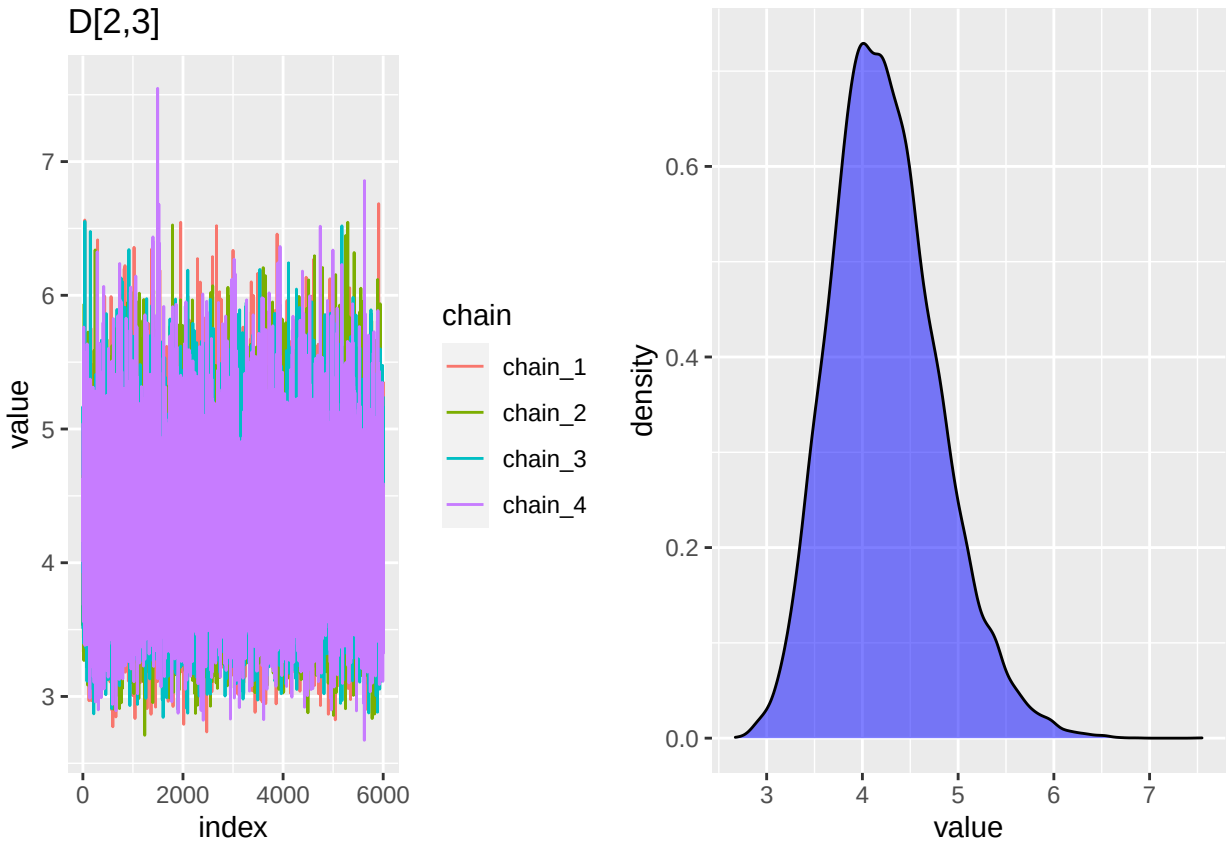

D[1,1]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

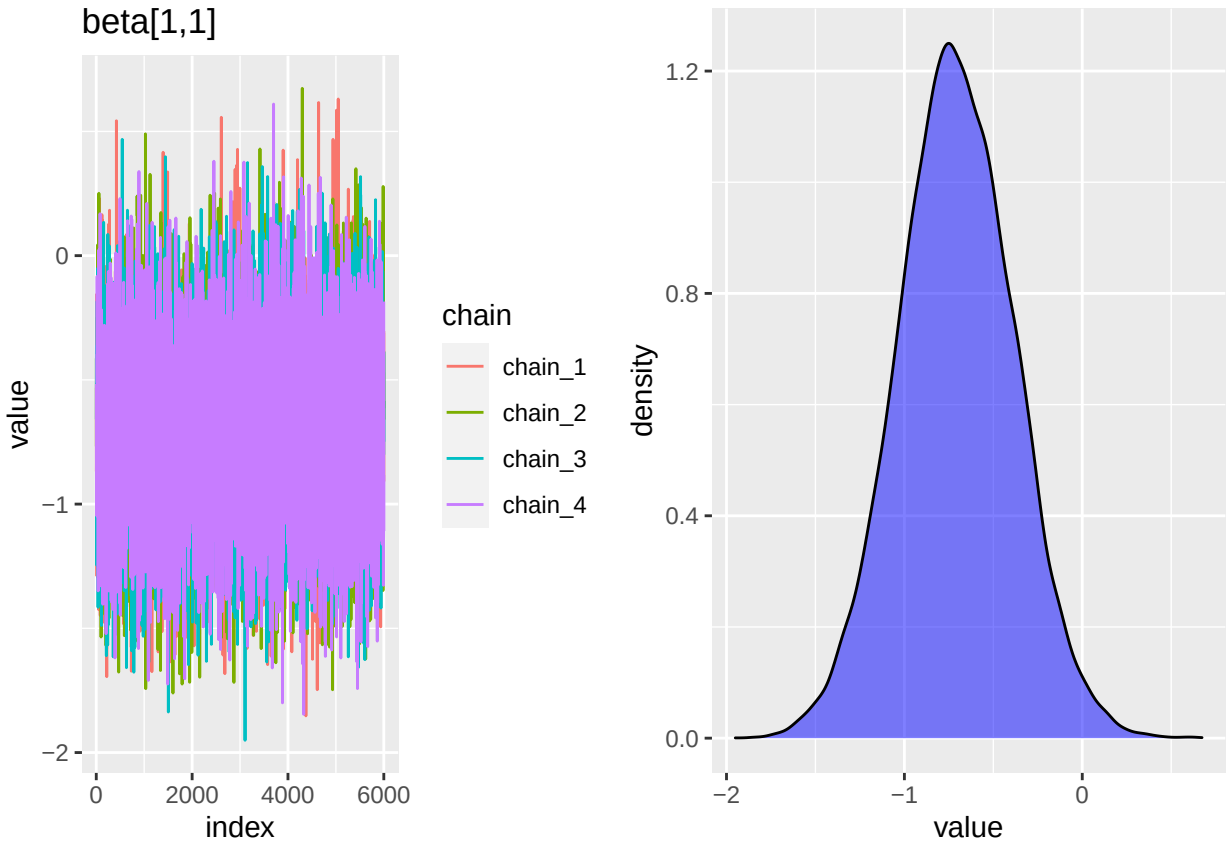

D[2,1]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

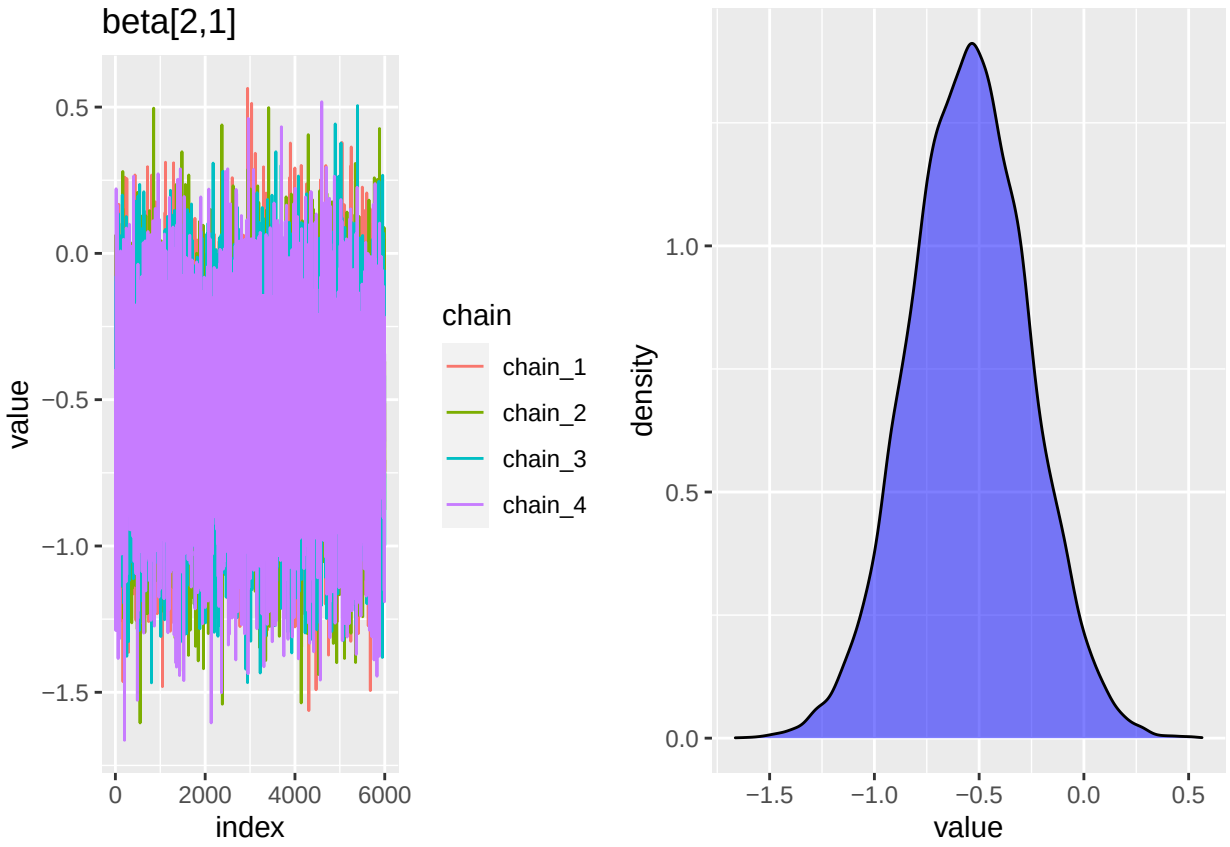

D[1,2]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

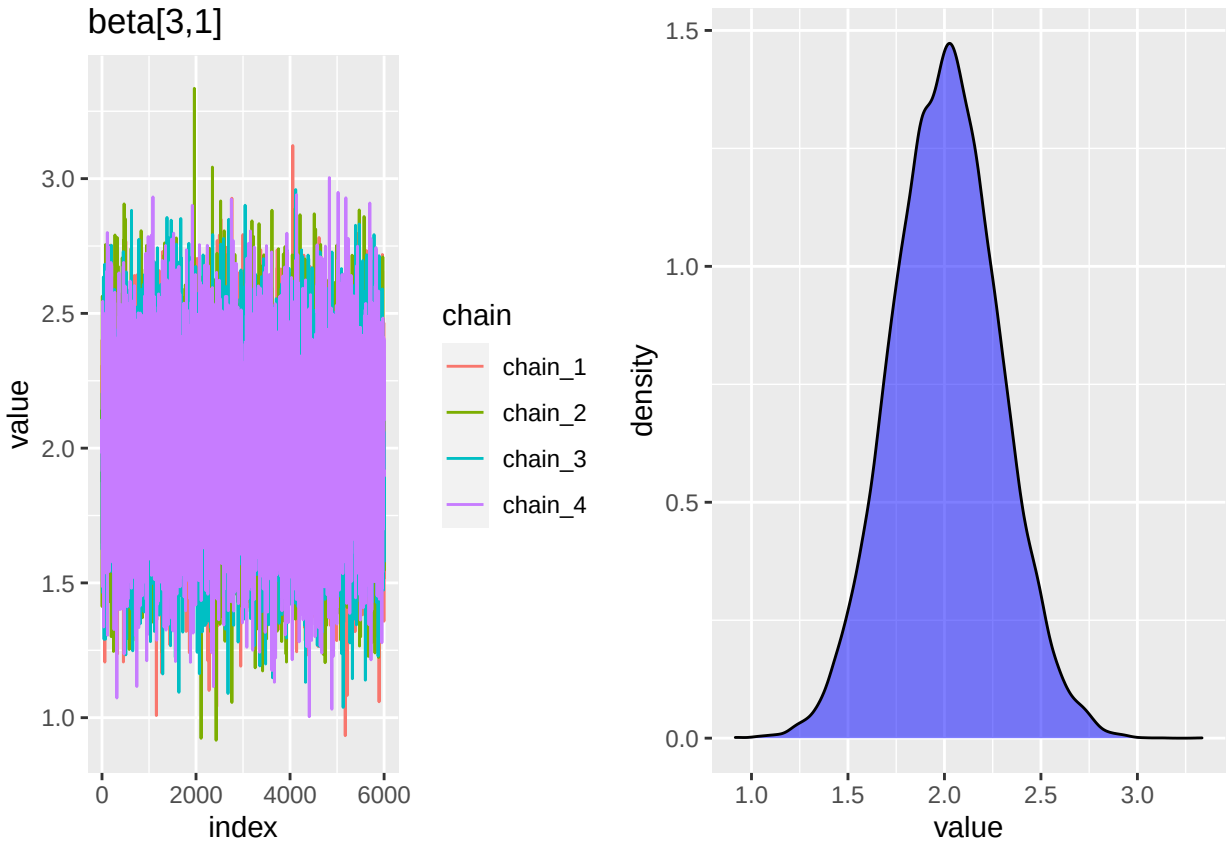

D[2,2]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

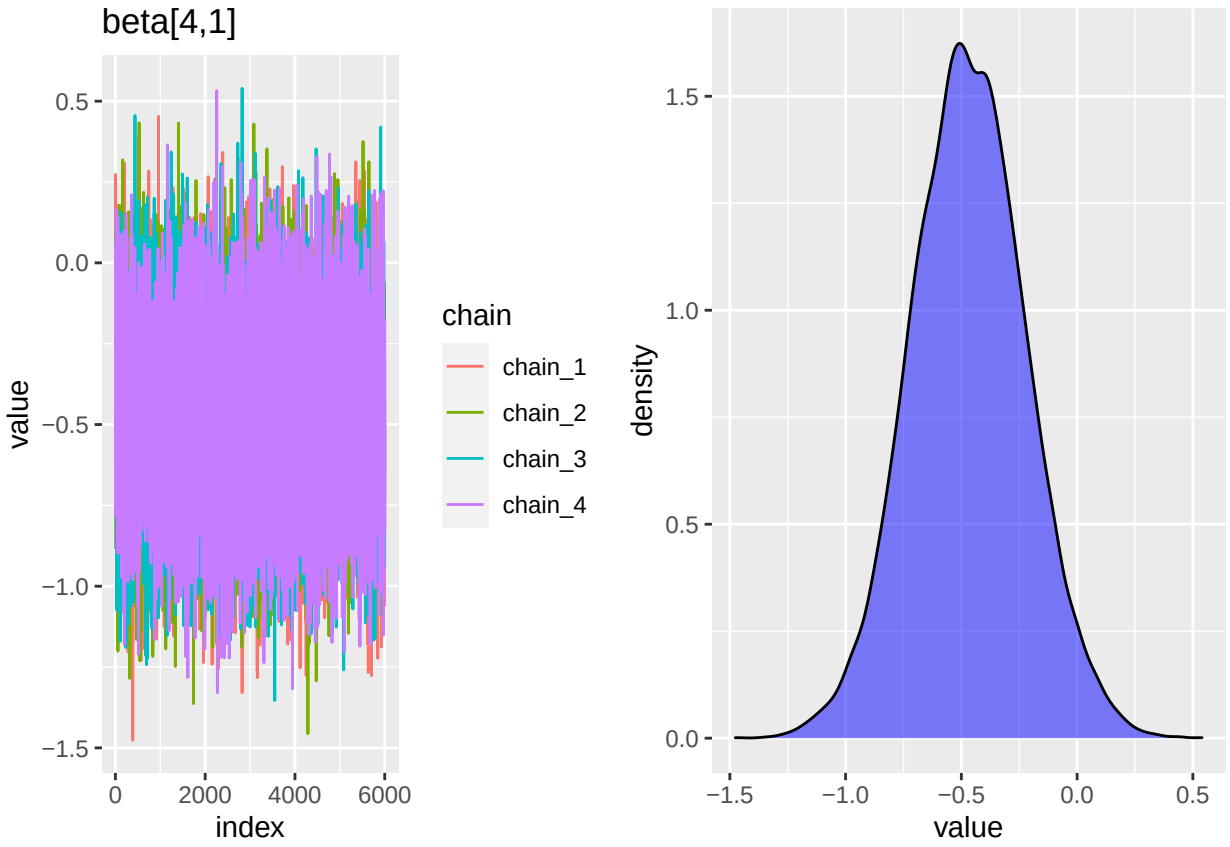

D[1,3]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

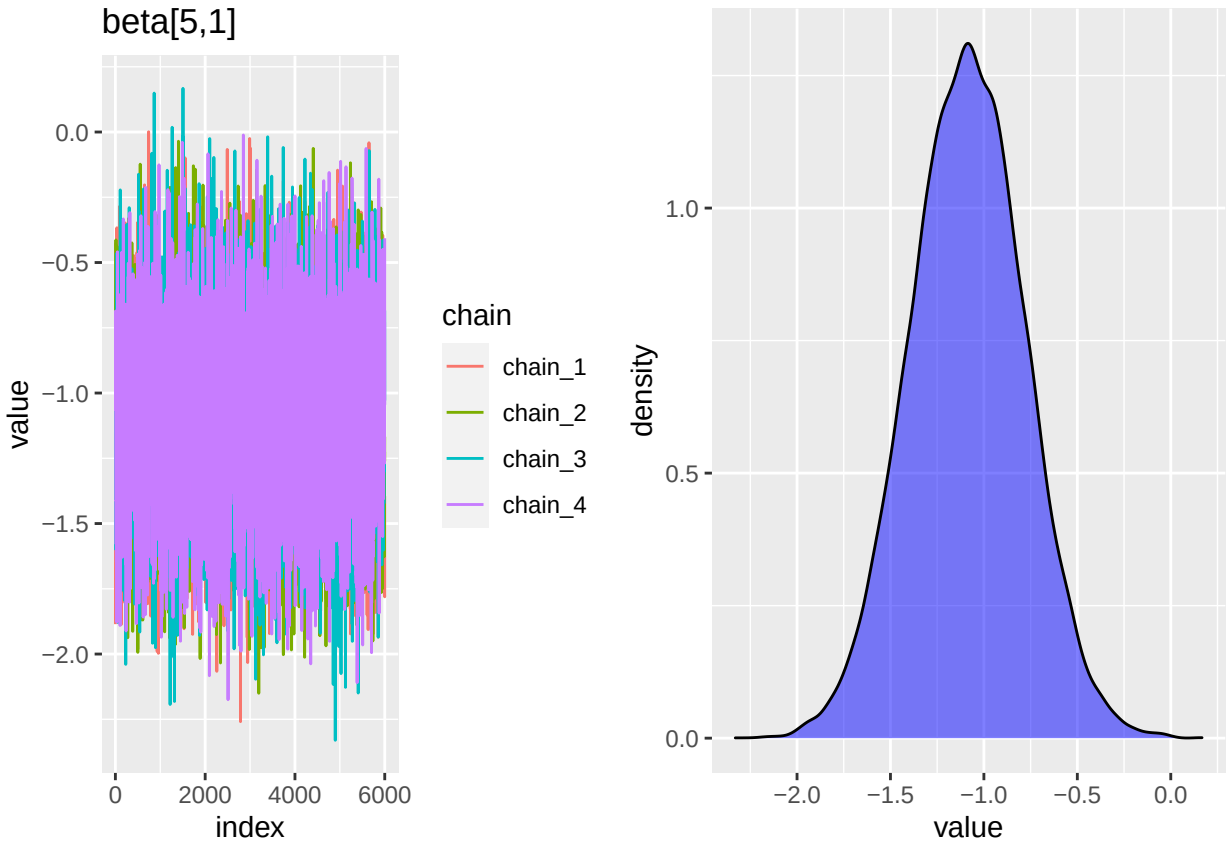

D[2,3]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

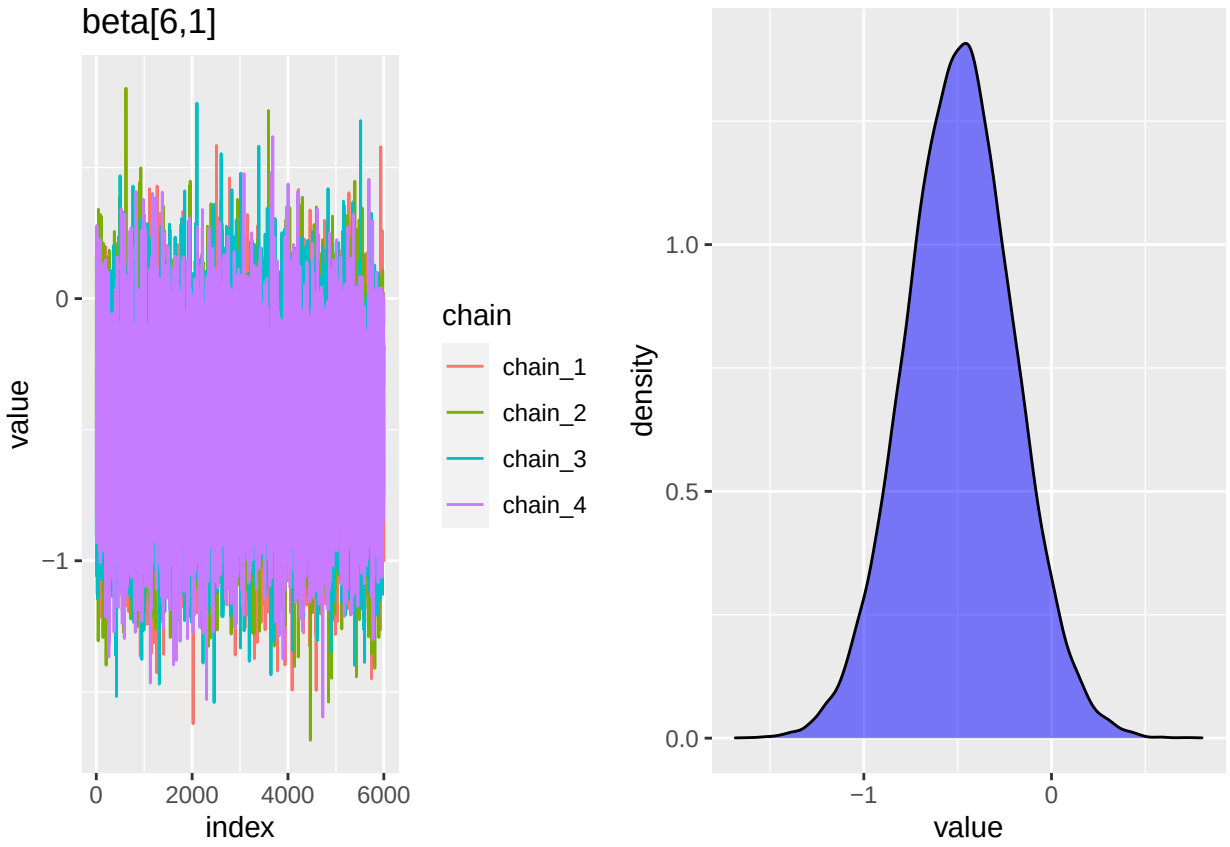

beta[1,1]
value
index
0
2000
4000
6000
0
−1
−2
chain
chain_1
chain_2
chain_3
chain_4
density
value
1.2
0.8
0.4
0.0
−2
−1
0

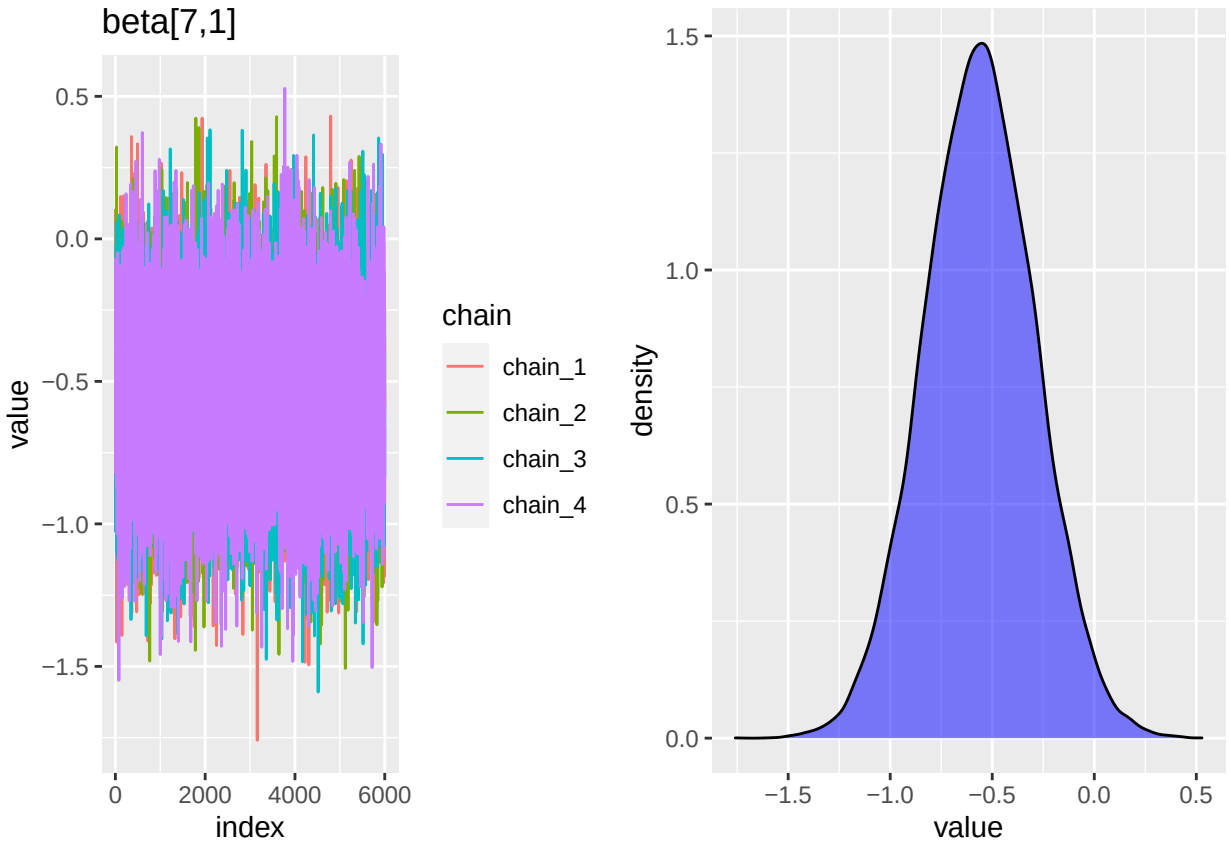

beta[2,1]
value
index
0.5
0.0
−0.5
−1.0
−1.5
0
2000
4000
6000
chain
chain_1
chain_2
chain_3
chain_4
density
value
1.0
0.5
0.0
−1.5
−1.0
−0.5
0.0
0.5

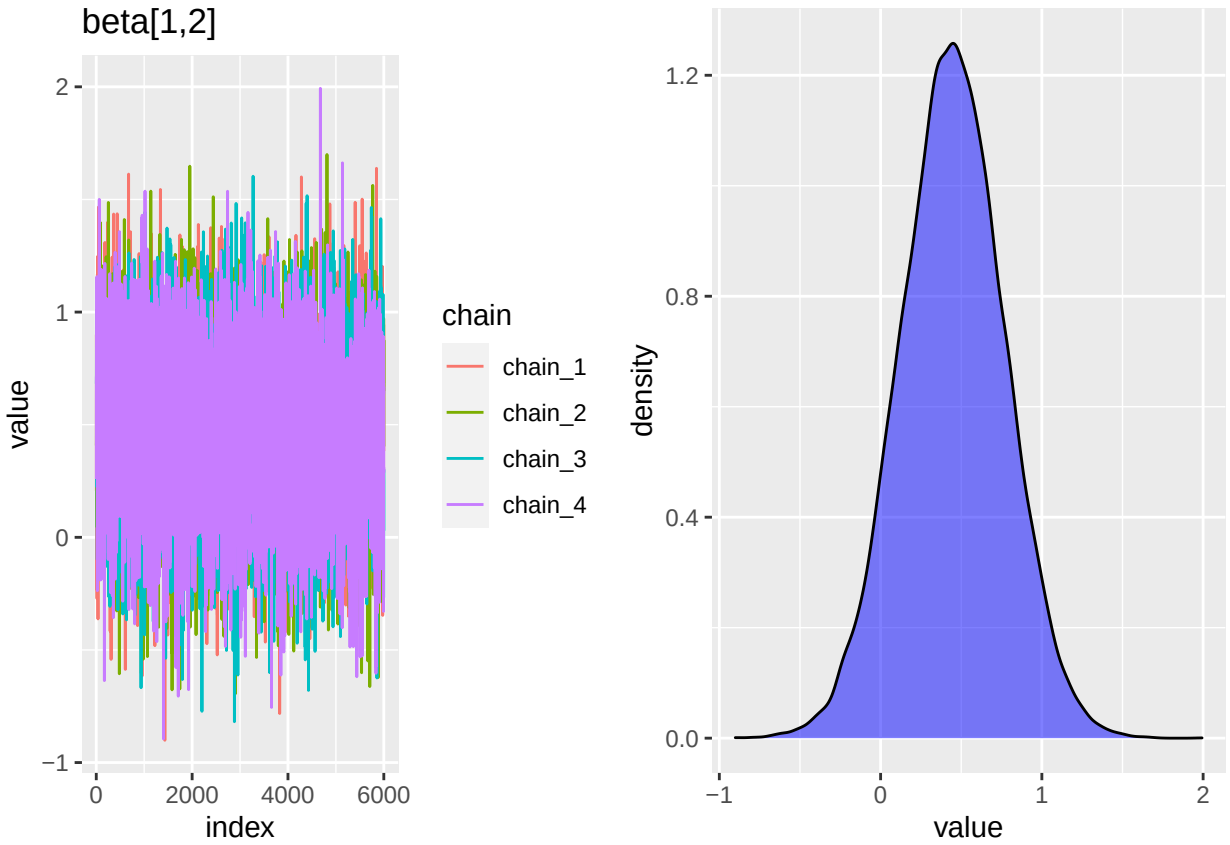

beta[3,1]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

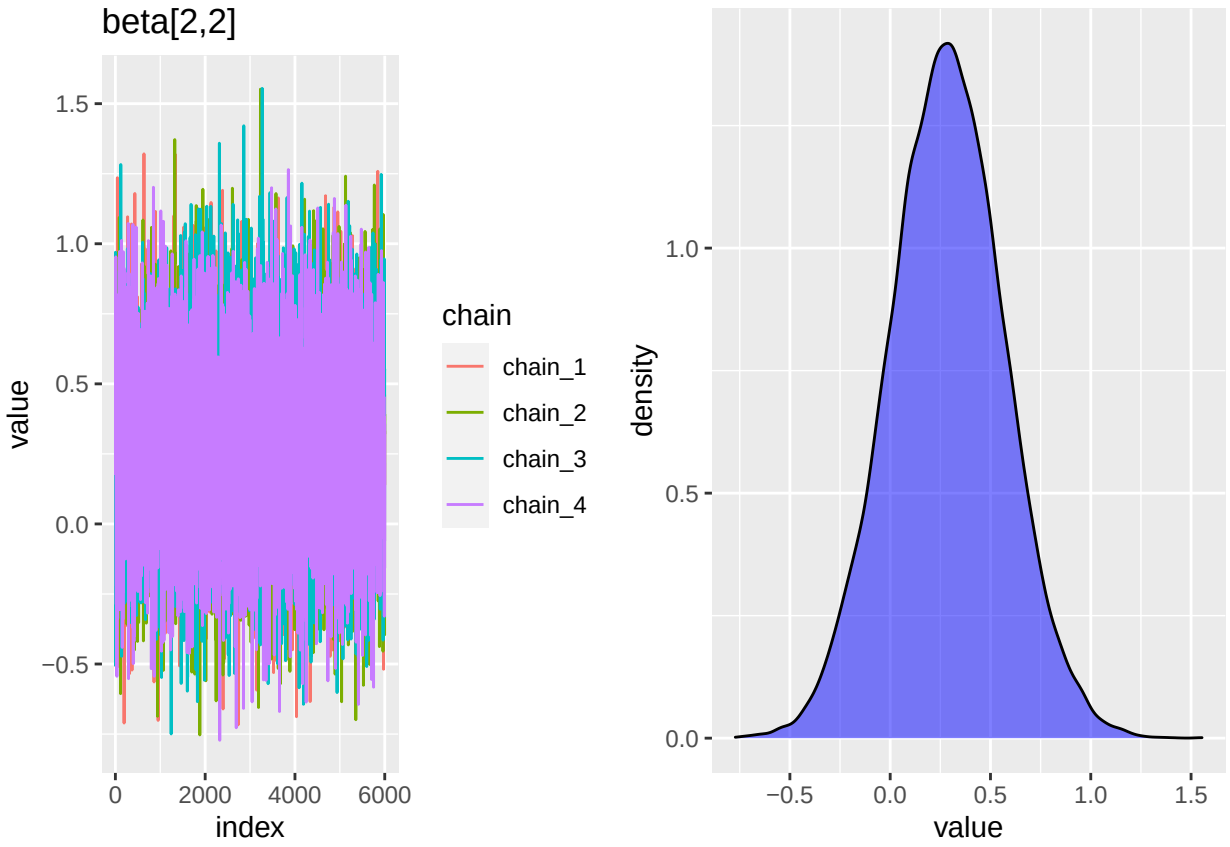

beta[4,1]
value
index
0.5
0.0
−0.5
−1.0
−1.5
0
2000
4000
6000
chain
chain_1
chain_2
chain_3
chain_4
density
value
1.5
1.0
0.5
0.0
−1.5
−1.0
−0.5
0.0
0.5

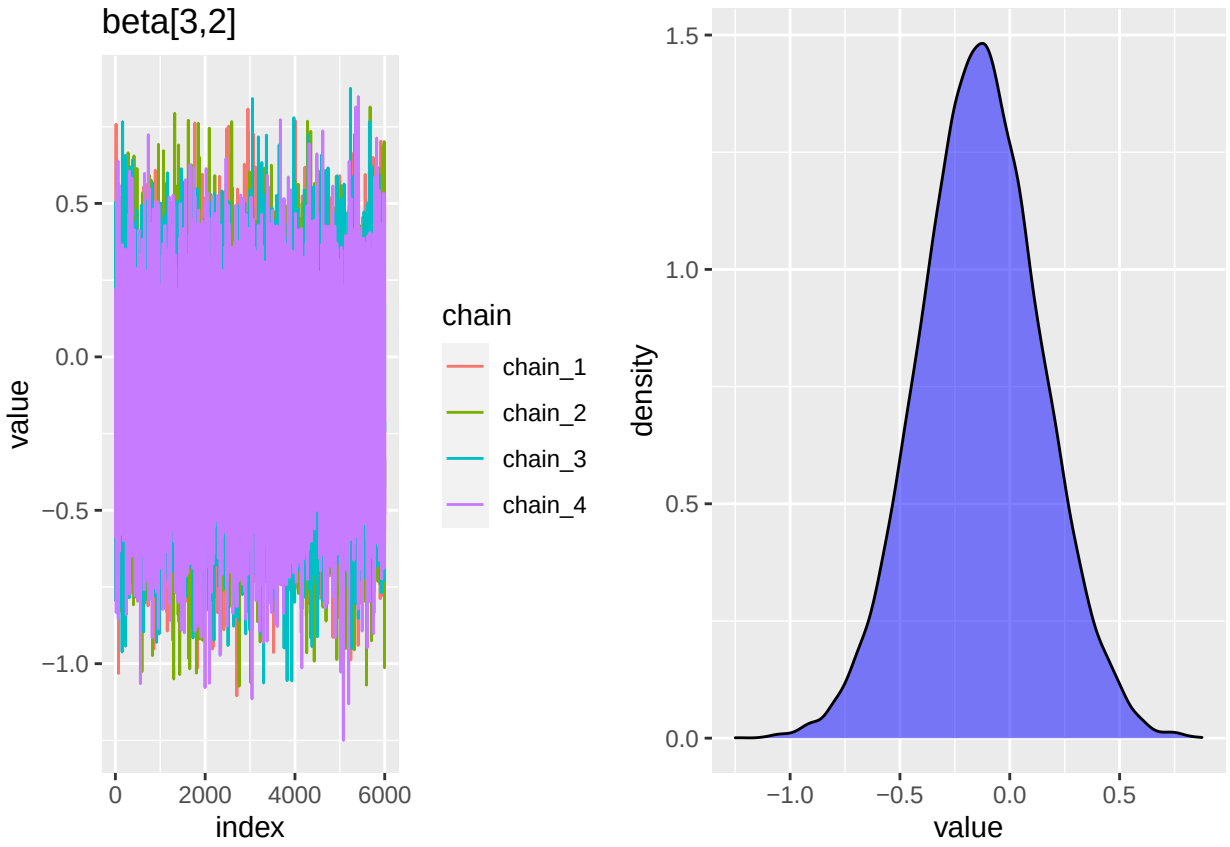

beta[5,1]
value
index
0.0
−0.5
−1.0
−1.5
−2.0
0
2000
4000
6000
chain
chain_1
chain_2
chain_3
chain_4
density
value
1.0
0.5
0.0
−2.0
−1.5
−1.0
−0.5
0.0

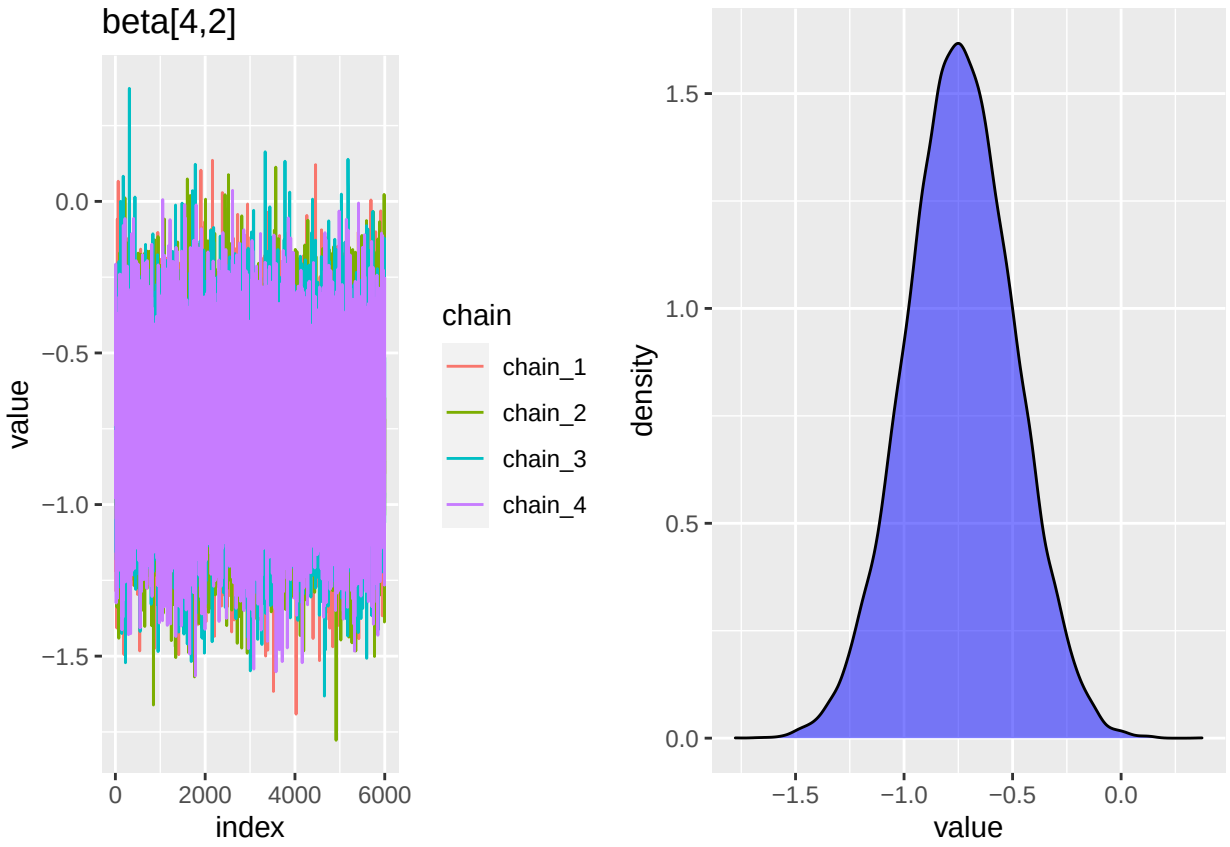

beta[6,1]
value
0
−1
0
2000
4000
6000
index
chain
chain_1
chain_2
chain_3
chain_4
density
1.0
0.5
0.0
−1
0
value

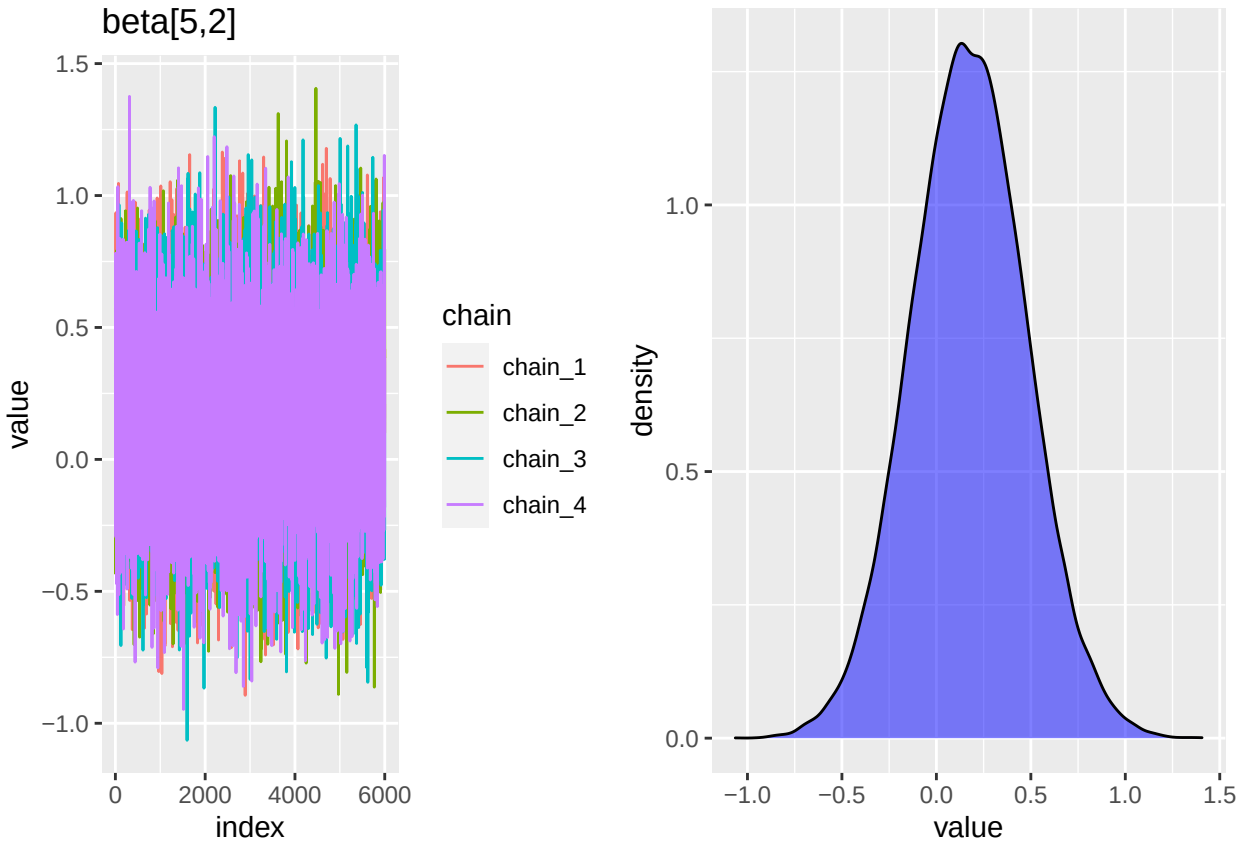

beta[7,1]
value
index
0.5
0.0
−0.5
−1.0
−1.5
0
2000
4000
6000
chain
chain_1
chain_2
chain_3
chain_4
density
value
1.5
1.0
0.5
0.0
−1.5
−1.0
−0.5
0.0
0.5

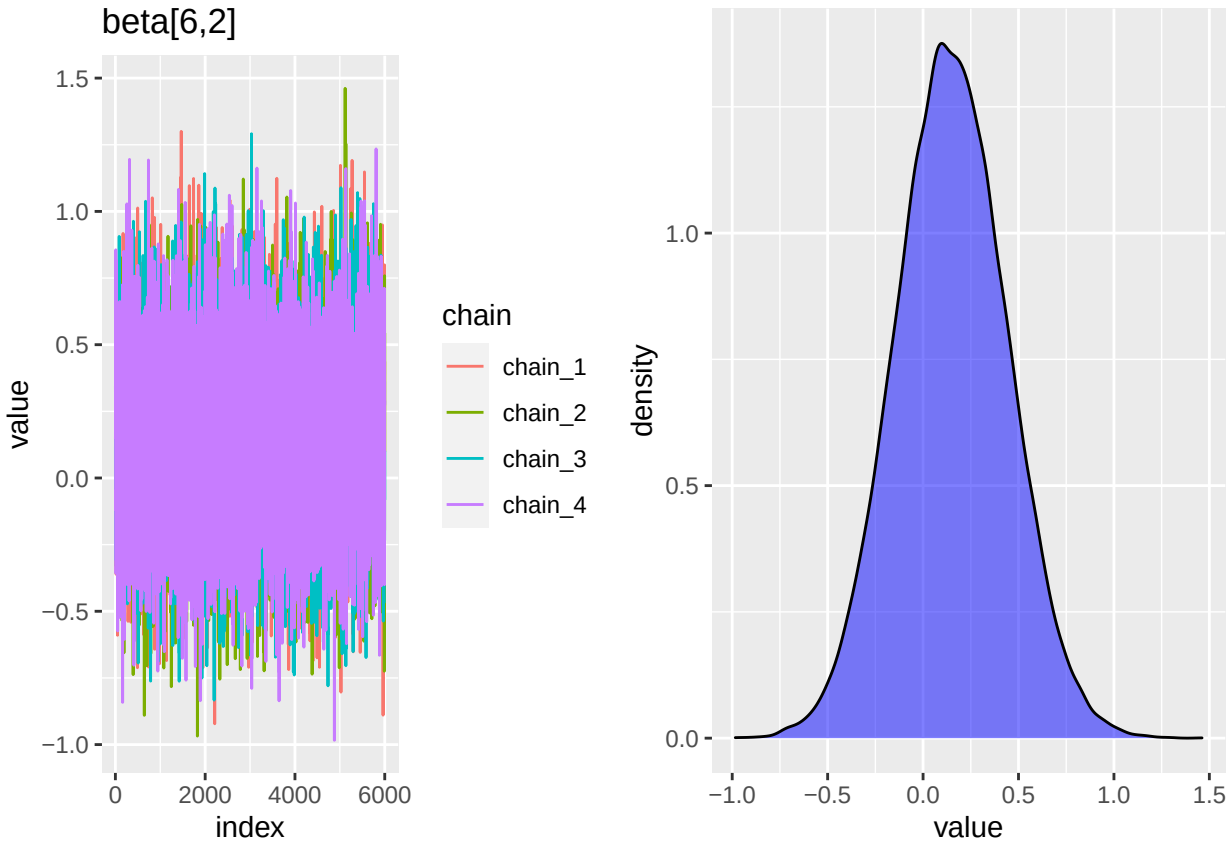

beta[1,2]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

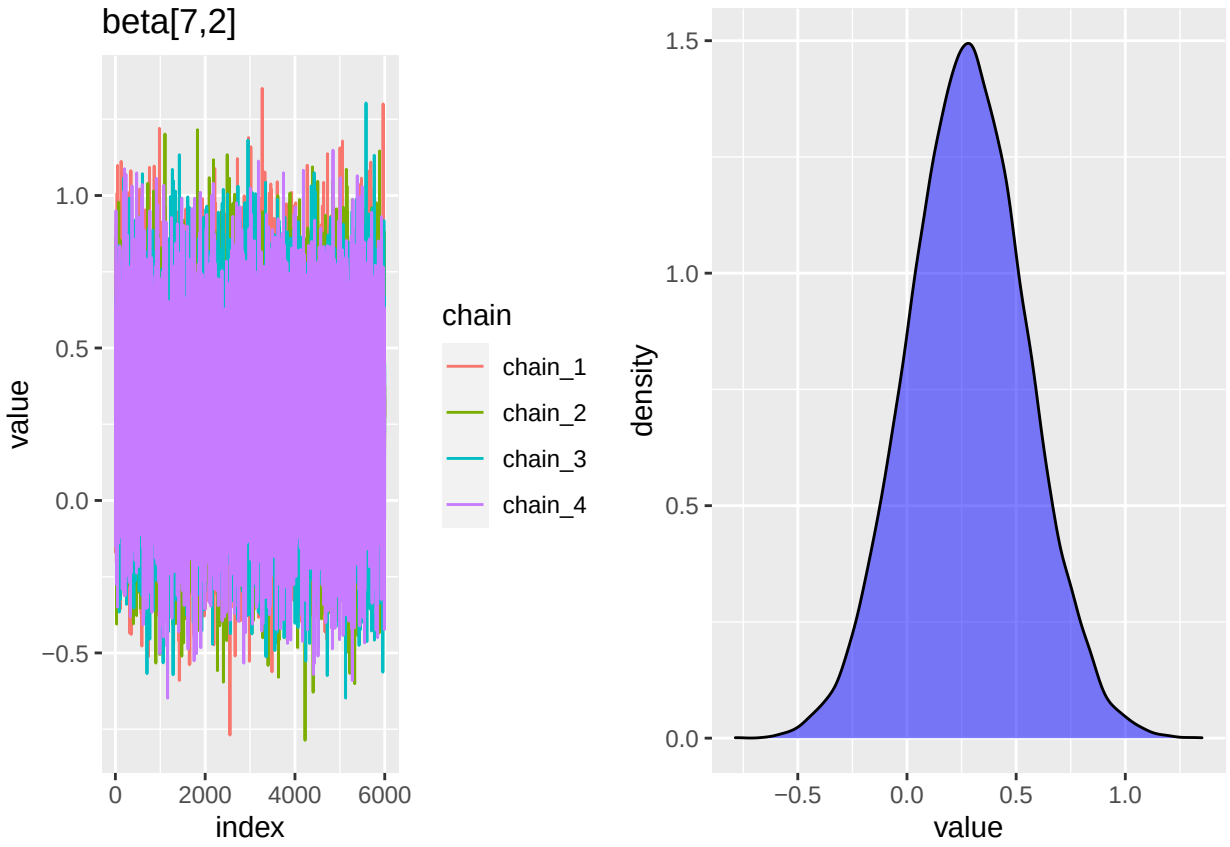

beta[2,2]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

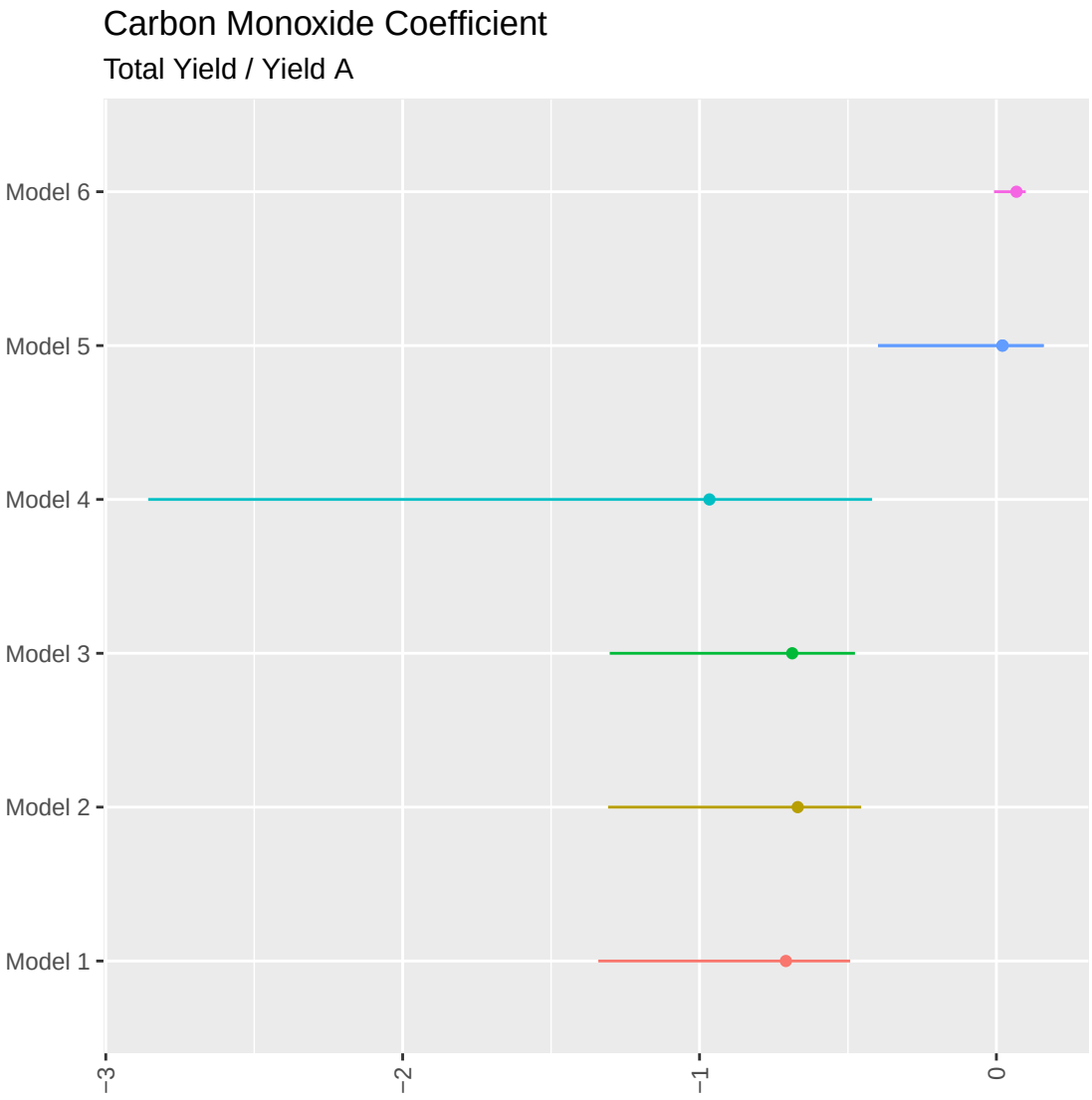

beta[3,2]
value
0.5
0.0
−0.5
−1.0
0
2000
4000
6000
index
chain
chain_1
chain_2
chain_3
chain_4
density
1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
value

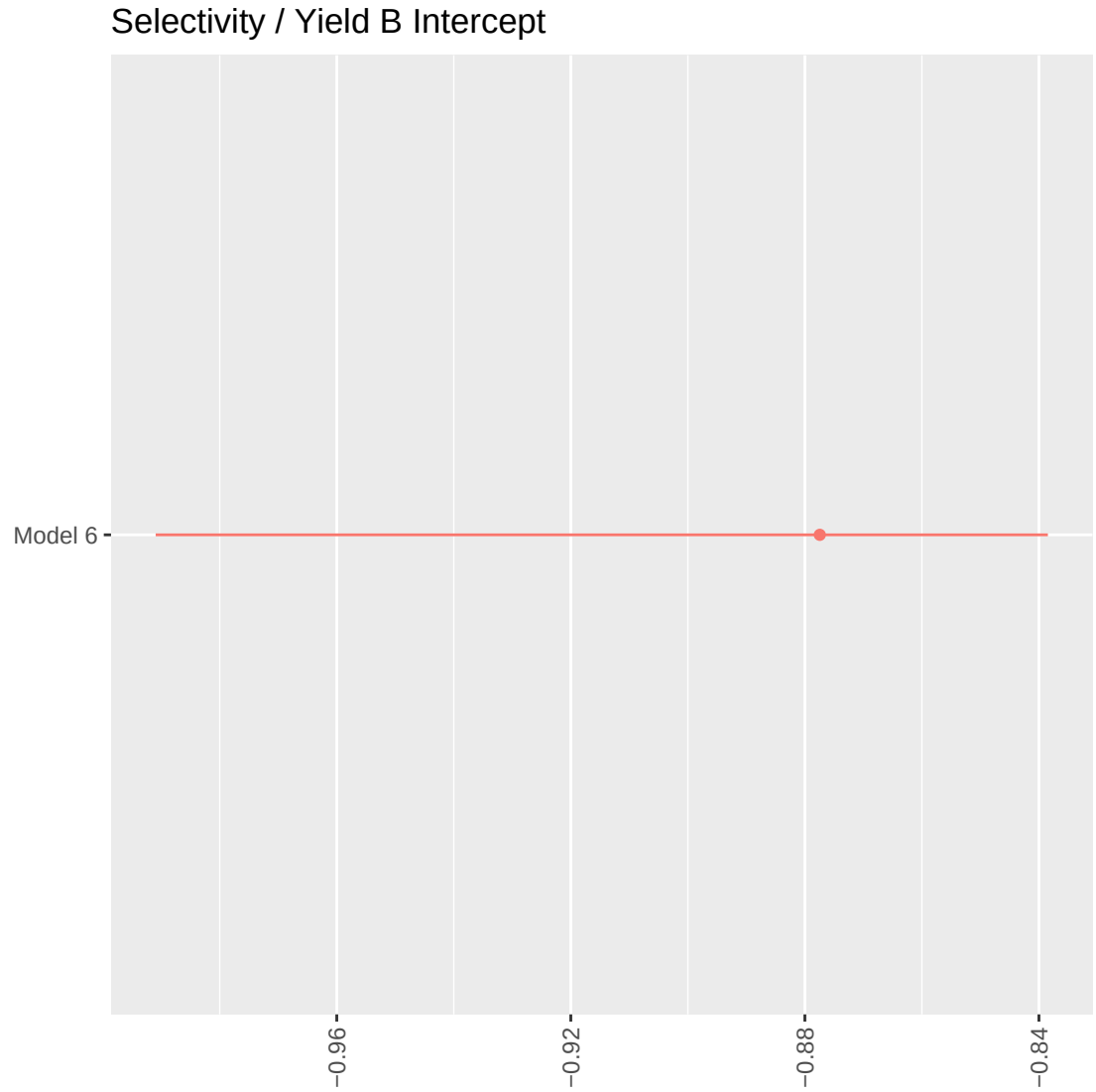

beta[4,2]
value
index
0.0
−0.5
−1.0
−1.5
0
2000
4000
6000
chain
chain_1
chain_2
chain_3
chain_4
density
value
1.5
1.0
0.5
0.0
−1.5
−1.0
−0.5
0.0

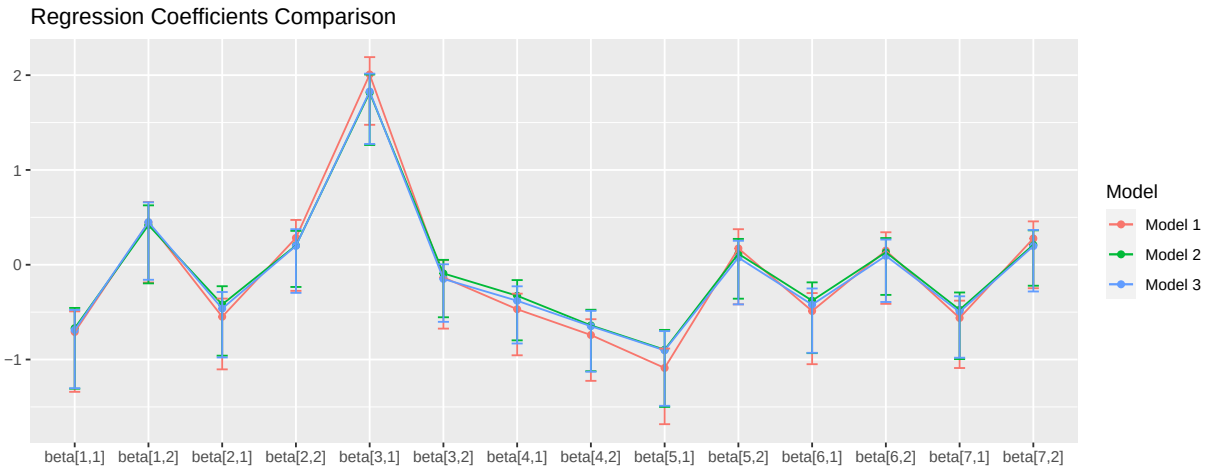

beta[5,2]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value

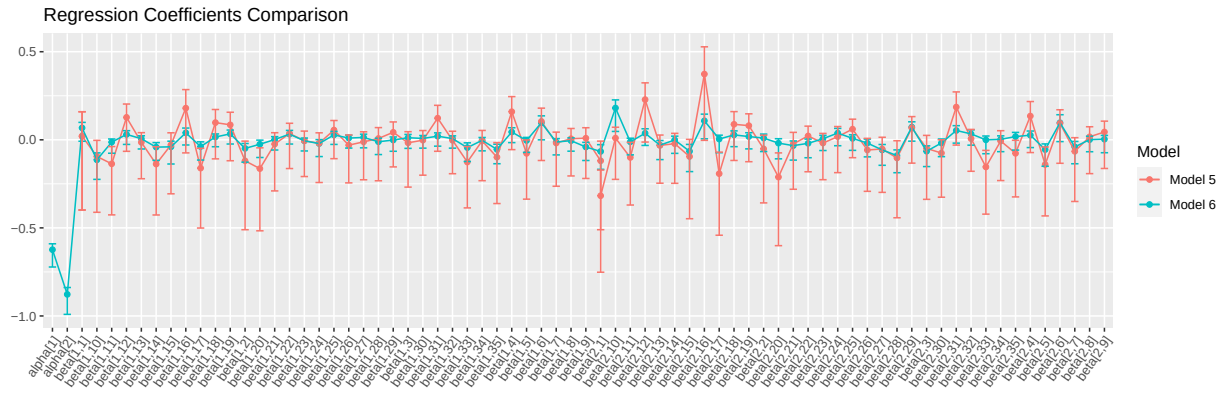

beta[6,2]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value


beta[7,2]
value
index
chain
chain_1
chain_2
chain_3
chain_4
density
value


Carbon Monoxide Coefficient
Total Yield / Yield A
Model 6
Model 5
Model 4
Model 3
Model 2
Model 1
−3
−2
−1
0


Selectivity / Yield B Intercept
Model 6
−0.96
−0.92
−0.88
−0.84


Regression Coefficients Comparison
2
1
0
−1
beta[1,1]
beta[1,2]
beta[2,1]
beta[2,2]
beta[3,1]
beta[3,2]
beta[4,1]
beta[4,2]
beta[5,1]
beta[5,2]
beta[6,1]
beta[6,2]
beta[7,1]
beta[7,2]
Model
Model 1
Model 2
Model 3




```
################################################
########## CONVERGENCE DIAGNOSTICS ##########
################################################


#----------------------#
##### CONVERGENCE #####
#----------------------#


#############################
# SELECTING INITIAL VALUES #
#############################
## Looks as though we are running more iterations and/or more chains
## to compensate for the lack of initial values.



#############
# TRACEPLOT #
#############
#<Insert Traceplot>



#############
# AUTOCORR #
#############
### Low autocorrelation [ x > abs(0.50) ] indicates convergence
ACM1 <- autocorr(samples1[[1]], lag=1)
ACM2 <- autocorr(samples2[[1]], lag=1)
ACM3 <- autocorr(samples3[[1]], lag=1)
ACM4 <- autocorr(samples4[[1]], lag=1)
ACM5 <- autocorr(samples5[[1]], lag=1)
ACM6 <- autocorr(samples6[[1]], lag=1)

##Checks for correlations of |0.5| or greater, by parameter, and counts occurrences.
as.tibble(c(
sum(apply(X=ACM1, 2, function(c)sum(c>abs(0.5)))),
sum(apply(X=ACM2, 2, function(c)sum(c>abs(0.5)))),
```

```
sum(apply(X=ACM3, 2, function(c)sum(c>abs(0.5)))),
sum(apply(X=ACM4, 2, function(c)sum(c>abs(0.5)))),
sum(apply(X=ACM5, 2, function(c)sum(c>abs(0.5)))),
sum(apply(X=ACM6, 2, function(c)sum(c>abs(0.5))))))
```

```
## Warning: ‘as.tibble()‘ was deprecated in tibble 2.0.0.
## i Please use ‘as_tibble()‘ instead.
## i The signature and semantics have changed, see ‘?as_tibble‘.
```

```
## # A tibble: 6 x 1
##   value
##   <int>
## 1     6
## 2     5
## 3     7
## 4    31
## 5    43
## 6    16
```

```
##########
# GEWEKE #
##########
### |z| less than 2 indicates convergence
GEM1 <- geweke.diag(samples1[])
df_GEM1 <- data.frame(c(GEM1[[1]],GEM1[[2]]))
GEM2 <- geweke.diag(samples2[])
df_GEM2 <- data.frame(c(GEM2[[1]],GEM2[[2]]))
GEM3 <- geweke.diag(samples3[])
df_GEM3 <- data.frame(c(GEM3[[1]],GEM3[[2]]))
GEM4 <- geweke.diag(samples4[])
df_GEM4 <- data.frame(c(GEM4[[1]],GEM4[[2]]))
GEM5 <- geweke.diag(samples5[])
df_GEM5 <- data.frame(c(GEM5[[1]],GEM5[[2]]))
#GEM6 <- geweke.diag(samples6[])
#df_GEM6 <- data.frame(c(GEM6[[1]],GEM6[[2]]))


#return the sum of counts of values that are greater than |2|.
as.tibble(c(
sum(apply(X=matrix(df_GEM1$z), 1, function(x) sum(abs(x)>2)))
,sum(apply(X=matrix(df_GEM1$z.1), 1, function(x) sum(abs(x)>2)))
,sum(apply(X=matrix(df_GEM2$z), 1, function(x) sum(abs(x)>2)))
,sum(apply(X=matrix(df_GEM2$z.1), 1, function(x) sum(abs(x)>2)))
,sum(apply(X=matrix(df_GEM3$z), 1, function(x) sum(abs(x)>2)))
,sum(apply(X=matrix(df_GEM3$z.1), 1, function(x) sum(abs(x)>2)))
,sum(apply(X=matrix(df_GEM4$z), 1, function(x) sum(abs(x)>2)))
,sum(apply(X=matrix(df_GEM4$z.1), 1, function(x) sum(abs(x)>2)))
,sum(apply(X=matrix(df_GEM5$z), 1, function(x) sum(abs(x)>2)))
,sum(apply(X=matrix(df_GEM5$z.1), 1, function(x) sum(abs(x)>2)))
#,sum(apply(X=matrix(df_GEM6$z), 1, function(x) sum(abs(x)>2)))
#,sum(apply(X=matrix(df_GEM6$z.1), 1, function(x) sum(abs(x)>2)))
))
```

```
## # A tibble: 10 x 1
##    value
##    <int>
##  1     0
##  2     1
##  3     0
##  4     4
##  5     2
##  6     0
##  7     0
##  8     0
##  9     4
## 10     6
```

```
#################
# GELMAN-RUBIN #
#################
### R less than 1.1 indicates convergence
GRM1 <- gelman.diag(samples1[], multivariate = FALSE);
GRM2 <- gelman.diag(samples2[], multivariate = FALSE);
GRM3 <- gelman.diag(samples3[], multivariate = FALSE);
GRM4 <- gelman.diag(samples4[], multivariate = FALSE);
GRM5 <- gelman.diag(samples5[], multivariate = FALSE);
GRM6 <- gelman.diag(samples6[], multivariate = FALSE);

##Code below looks at each GelmanRubin and pulls in the maximum value for each point estimation. The id
as.tibble(c(
max(tibble(GRM1$psrf[,1])),
max(tibble(GRM2$psrf[,1])),
max(tibble(GRM3$psrf[,1])),
max(tibble(GRM4$psrf[,1])),
max(tibble(GRM5$psrf[,1])),
max(tibble(GRM6$psrf[,1]))))
```

```
## # A tibble: 6 x 1
##   value
##   <dbl>
## 1  1.00
## 2  1.00
## 3  1.00
## 4  1.01
## 5  1.00
## 6  1.18
```

```
#--------------
##### SAMPLES SIZES REQUIRED #####

#########################
# EFFECTIVE SAMPLE SIZE #
#########################
### ESS over 1000 indicates convergence
ESS_M1 <- effectiveSize(samples1[])
ESS_M2 <- effectiveSize(samples2[])
```

```
ESS_M3 <- effectiveSize(samples3[])
ESS_M4 <- effectiveSize(samples4[])
ESS_M5 <- effectiveSize(samples5[])
ESS_M6 <- effectiveSize(samples6[])

##PULLS THE MINIMUM VALUE IN EACH EFFECTIVE SAMPLE SIZE IN LIEU OF DISPLAYING ALL ESS OUTPUTS FOR THE S
as.tibble(c(min(ESS_M1),min(ESS_M2),min(ESS_M3),min(ESS_M4),min(ESS_M5),min(ESS_M6)))
```

```
## # A tibble: 6 x 1
##   value
##   <dbl>
## 1 6539.
## 2 6193.
## 3 6841.
## 4  704.
## 5 1996.
## 6 3256.
```

```
##############################################
############## MODEL SELECTION ###############
##############################################

### BAYES FACTORS

### STOCHASTIC SEARCH VARIABLE SELECTION

### CROSS VALIDATION

### BAYESIAN MODEL AVERAGING
```