Bayesian Modelling

Brian R. Bolivar A. - Cristina Farruku

2022-07-03

```
data <- read.csv(file="diamonds.csv", header = T)
colnames(data)[10] <- "y.m"
knitr::kable(head(data),align = "ccccccccccc", "pipe")
```

X	carat	cut	color	clarity	depth	table	price	x	y.m	z
1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

Variables

Numerical Variables

- **carat**: weight of the diamond;
- **x**: length in mm;
- **y**: width in mm;
- **z**: dept in mm;
- **depth**: total depth percentage, measured as: $z\text{mean}(x,y) = 2\,zx+y$;
- **table**: a diamond's table refers to the flat facet of the diamond seen when the stone is face up. The main purpose of a diamond table is to refract entering light rays and allow reflected light rays from within the diamond to meet the observer's eye;
- **price**: the price of the diamond (range 326 - 18.823 in $);

Categorical Variables

- **cut**: quality of the cut (Fair, Good, Very Good, Premium, Ideal);
- **color**: color of the diamond (D being the best and J the worst);
- **clarity**: a measurement of how the diamonds is (I1 as the worst and IF as the best);

Initial Data Analysis | Full Data-set

Missing Values:

```
summary(data)
```

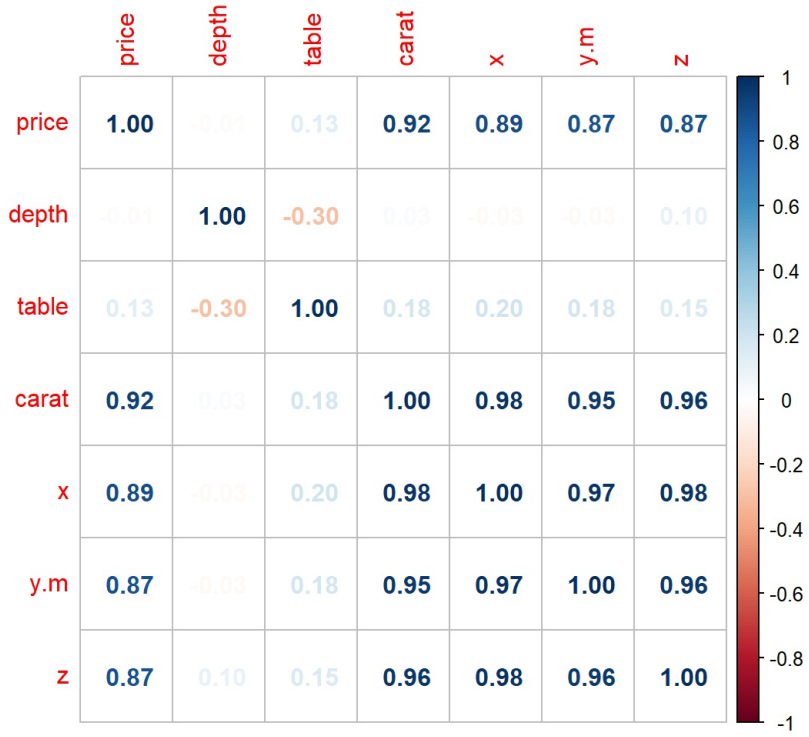
```
##        X             carat            cut               color          
##  Min.   :    1   Min.   :0.2000   Length:53940       Length:53940      
##  1st Qu.:13486   1st Qu.:0.4000   Class :character   Class :character  
##  Median :26971   Median :0.7000   Mode  :character   Mode  :character  
##  Mean   :26971   Mean   :0.7979                                        
##  3rd Qu.:40455   3rd Qu.:1.0400                                        
##  Max.   :53940   Max.   :5.0100                                        
##    clarity             depth           table           price       
##  Length:53940       Min.   :43.00   Min.   :43.00   Min.   :  326  
##  Class :character   1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950  
##  Mode  :character   Median :61.80   Median :57.00   Median : 2401  
##                     Mean   :61.75   Mean   :57.46   Mean   : 3933  
##                     3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324  
##                     Max.   :79.00   Max.   :95.00   Max.   :18823  
##        x              y.m              z         
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910  
##  Median : 5.700   Median : 5.710   Median : 3.530  
##  Mean   : 5.731   Mean   : 5.735   Mean   : 3.539  
##  3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040  
##  Max.   :10.740   Max.   :58.900   Max.   :31.800
```

```
library(dplyr)
data <- filter(data, data$z > 0)    # Remove rows with x, y, z = 0
```

From the summary of the data we can see that variables “x”, “y.m” and “z”, which represent the dimensions of the diamonds, have values equal to 0. Considered the numbers of observation we prefer to remove the rows with this values.

Correlation Plot

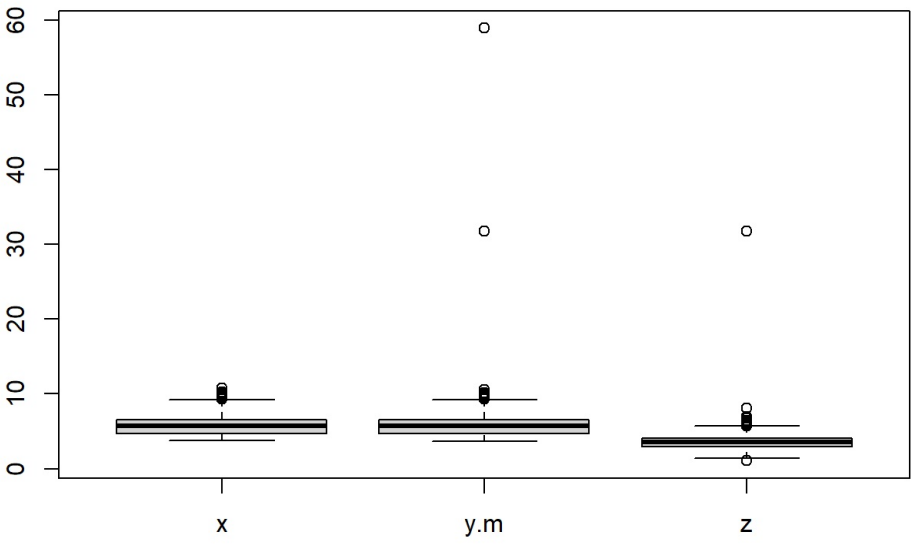
```
numerical.data <- data[c(8, 6, 7, 2, 9, 10, 11 )]
M = cor(numerical.data)
corrplot(M, method = 'number')
```



From the Correlation-Plot:

- Price Variable: this variables have high correlation with “carat” and the 3-dimensions of the diamond;
- There is high correlation between “carat” and the 3-dimensions of the diamond, an explanation can be due to the proportion of the dimension of the diamonds and consequently the carat (weight) of the same.

```
boxplot(numerical.data[c(5, 6, 7)])
```



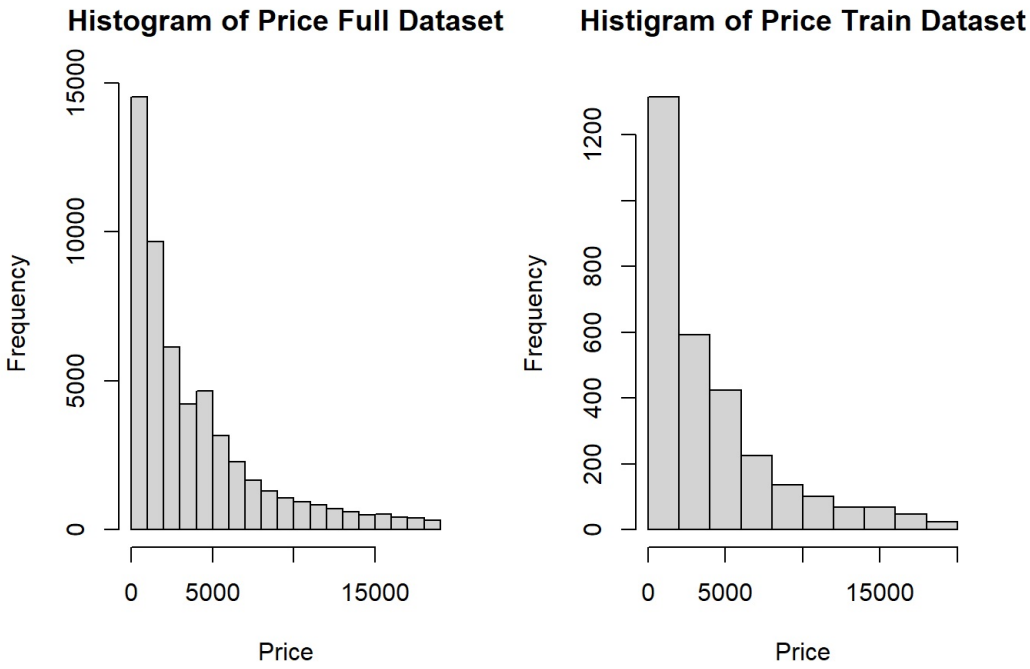
Considering the number of the observation we prefer to keep the outliers.

Initial Analysis | Train-Set

```
set.seed(1)
subsample <- sample(1:nrow(data), size = 4000, replace = FALSE)

train <- data[subsample[1:3000],]
test <- data[subsample[3001:4000],]
```

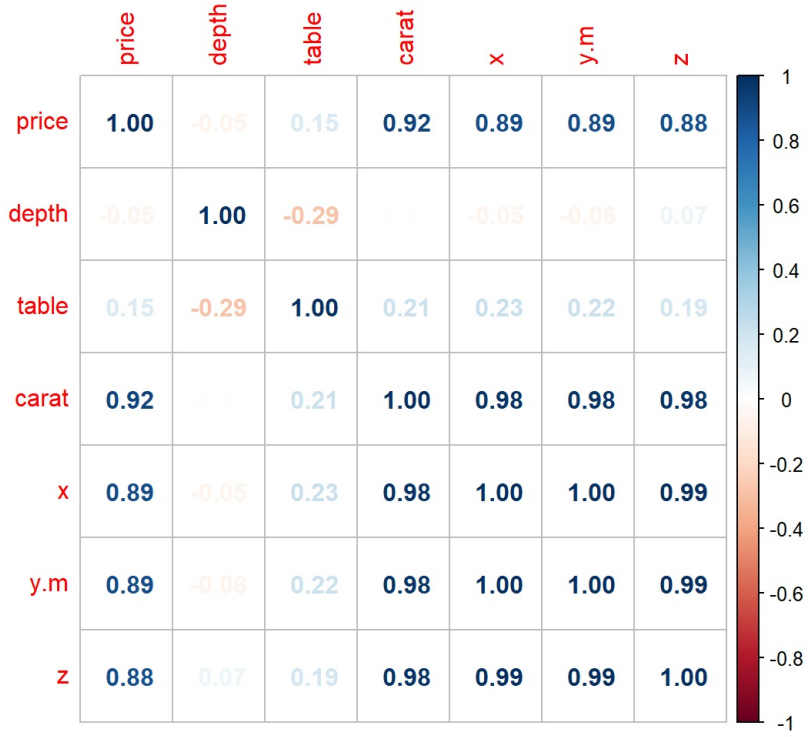
We're considering a sub-set of 4000 observations, 3000 for the train and 1000 for the test.

```
par(mfrow=c(1,2))
hist(data$price, main=c("Histogram of Price Full Dataset"), xlab = c("Price"))
hist(train$price, main=c("Histigram of Price Train Dataset"), xlab = c("Price"))
```

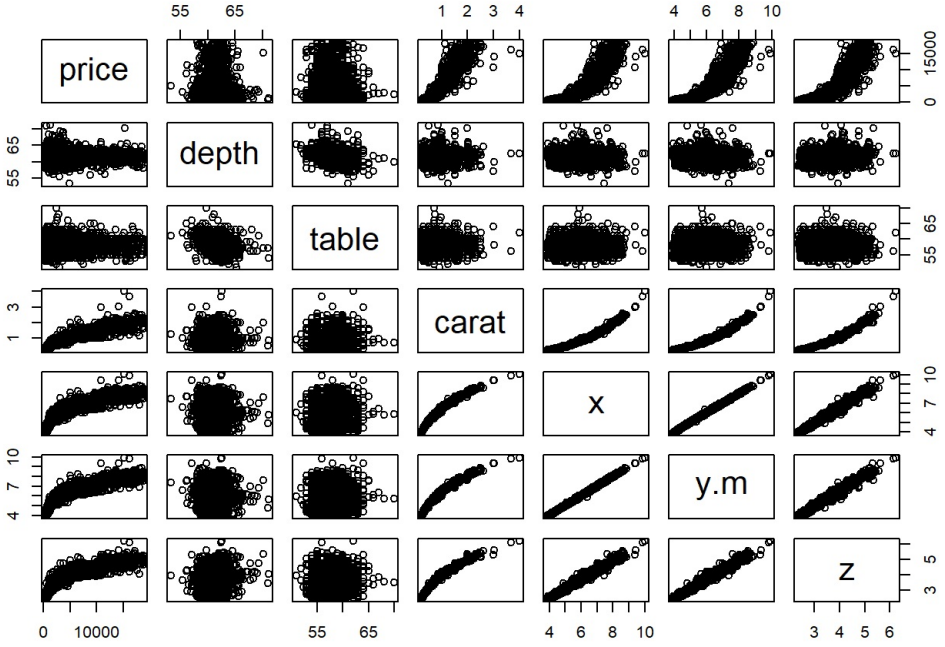


The Price of the Train-Set has the same distribution of the Price of the Full-Set.

```
numerical.data <- train[c(8, 6, 7, 2, 9, 10, 11 )]
M = cor(numerical.data)
corrplot(M, method = 'number')
```




```
plot(numerical.data)
```

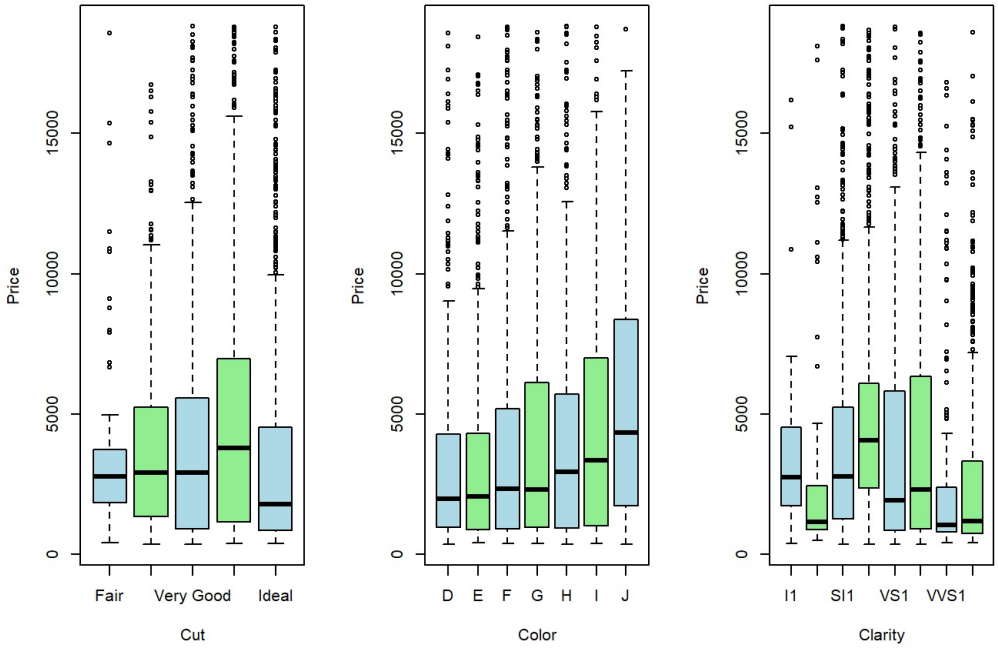


Same conclusion for the Tran-Set, covariates have the same behavior of the covariates of the Full-Set.

Categorical Variables Analysis

Categorical and Price

```
par(mfrow=c(1,3))

mycolor=c("lightblue", "lightgreen")

train$cut <- factor(train$cut , levels=c("Fair", "Good", "Very Good", "Premium", "Ideal"))
boxplot(train$price ~ train$cut, ylab="Price", xlab="Cut", col=mycolor)
boxplot(train$price ~ train$color, ylab="Price", xlab="Color", col=mycolor)
boxplot(train$price ~ train$clarity, ylab="Price", xlab="Clarity", col=mycolor)
```

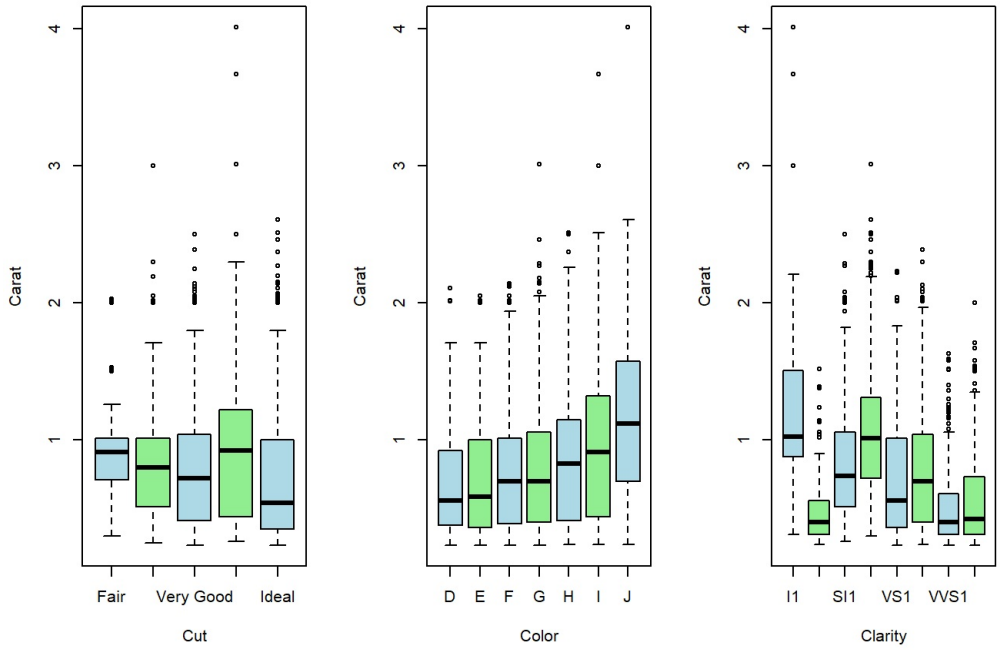


From above we can see:

1. There is a slight positive correlation between the price and the cut of the diamonds;
2. There is a positive correlation between the price and the color of the diamonds, it seems reasonable. Better is the color higher is the price of the diamond;
3. From the third box-plot seems there is not a correlation with the price.

Categorical and Carat

```
par(mfrow=c(1,3))
boxplot(train$carat ~ train$cut, ylab="Carat", xlab="Cut", col=mycolor)
boxplot(train$carat ~ train$color, ylab="Carat", xlab="Color", col=mycolor)
boxplot(train$carat ~ train$clarity, ylab="Carat", xlab="Clarity", col=mycolor)
```

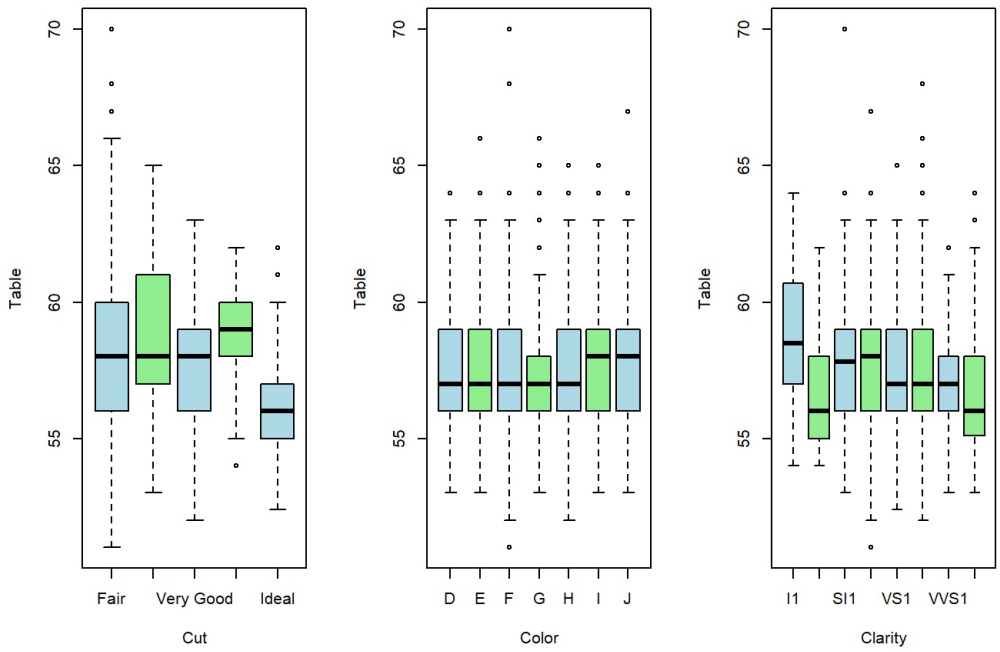


Remember that the carat is the weight of the diamond, and we can see:

1. No correlation with the cut;
2. Positive correlation with the color;
3. No correlation with the clarity.

Categorical and Table

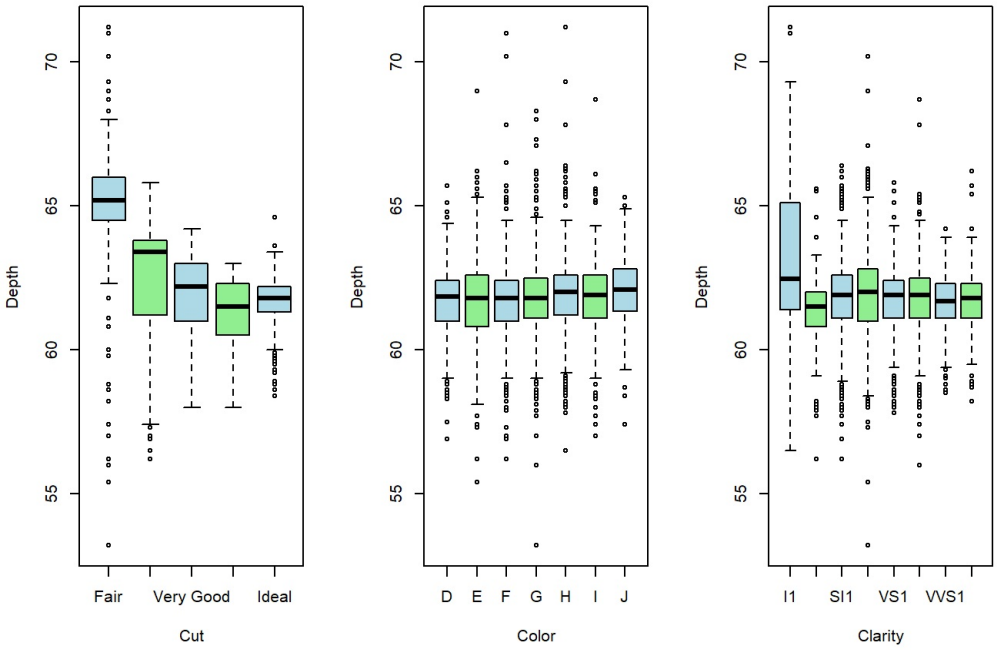
```
par(mfrow=c(1,3))
boxplot(train$table ~ train$cut, ylab="Table", xlab="Cut",col=mycolor)
boxplot(train$table ~ train$color, ylab="Table", xlab="Color", col=mycolor)
boxplot(train$table ~ train$clarity, ylab="Table", xlab="Clarity", col=mycolor)
```



It seems there is not correlation with Table variable.

Categorical and Depth

```
par(mfrow=c(1,3))
boxplot(train$depth ~ train$cut, ylab="Depth", xlab="Cut", col=mycolor)
boxplot(train$depth ~ train$color, ylab="Depth", xlab="Color", col=mycolor)
boxplot(train$depth ~ train$clarity, ylab="Depth", xlab="Clarity", col=mycolor)
```

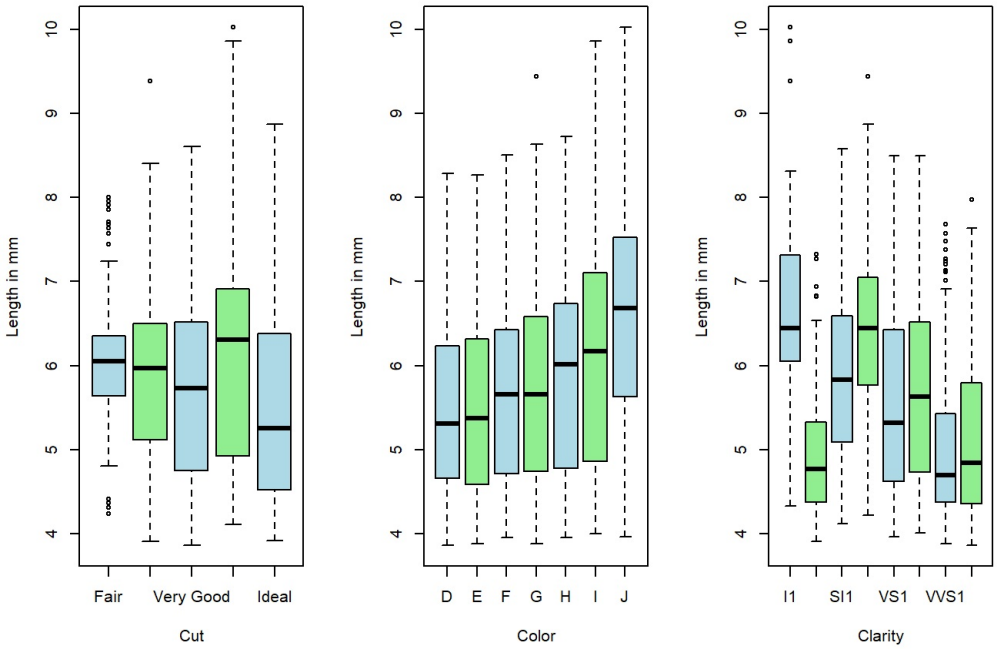


We can see only a negative correlation with the cut of the diamonds.

Categorical and Length in mm

For the last box-plot we're considering only the values of x.

```
par(mfrow=c(1,3))
boxplot(train$x ~ train$cut, ylab="Length in mm", xlab="Cut", col=mycolor)
boxplot(train$x ~ train$color, ylab="Length in mm", xlab="Color", col=mycolor)
boxplot(train$x ~ train$clarity, ylab="Length in mm", xlab="Clarity", col=mycolor)
```



It seems we have:

1. Positive correlation with color;
2. A sort of negative correlation with clarity.

Fitting Models

Now we consider a series of models to compare and see how they perform for prediction purpose.

```
y <- train$price   # response
X_train <- train[, c(2, 3, 4, 5, 6, 7, 9, 10, 11)]  # In the covariates we have a variables which rapresent the i
ndex of the observation, we don't need it.

X_train <- model.matrix(y~. , data=X_train)  # Trasform the categorical variables in matrix with 0-1

n <- nrow(X_train)
p <- ncol(X_train)
```

Bayesian Normal Linear Regression

The first model we consider is the Bayesian Normal Regression, we don't expect a great performance of this one, but we're going to use it to compare with others models with better performance.

The model is defined as:

$Y = \beta_0 + \beta_1 + x_i + \ldots + \beta_p x_p$

where p is the number of predictors.

And Y_i is defined as:

$Y_i \,|\, \beta, \tau \sim N(\beta^T x_i, \tau)$

Where $\tau = {}^1\sigma^2$ (the precision)

Priors:$\beta_j \sim N_p(0, 0.01)$

$\tau \sim 1U(0, 10^6)$

The choice of the **hyper-parameters** was made to have:

- Weak-Informative prior for β;
- Non-Informative prior for τ.

Algorithm

```
model1_code <- function() {
  ## Likelihood
  for (i in 1:n) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- X[i,] %*% b[]
  }
  ## Priors
  for (j in 1:p) {
    b[j] ~ dnorm(0, 0.01)
  }
  tau <- pow(sigma, -2)  # reciprocal of the variance
  sigma ~ dunif(0,10^6)  # range of values (non informative), we always be
                         # in this range also in the posteriori
                         # Important limit in the definition of the possible range of the standard deviation
}

model1_data <- list(y = y, X = X_train, n = n, p = p)
model1_params <- c("b", "mu", "tau")

model1_run <- jags(
  data = model1_data,
  parameters.to.save = model1_params,
  model.file = model1_code,
  n.chains = 1,
  n.iter = 5000,
  n.burnin = 2000,
  n.thin = 1,
  jags.seed = 101
)
```

```
## module glm loaded
```

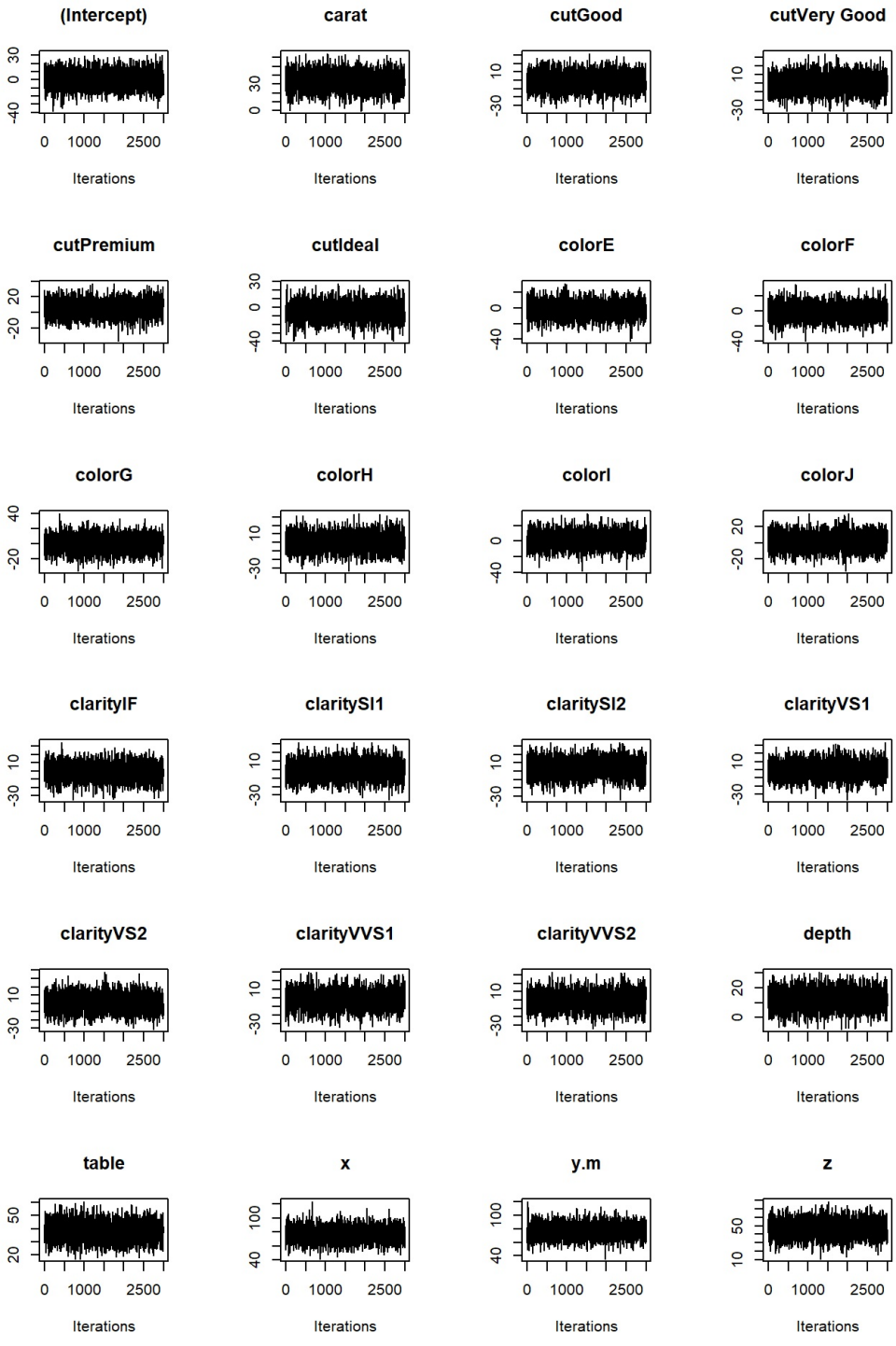
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 3000
##    Unobserved stochastic nodes: 25
##    Total graph size: 81036
##
## Initializing model
```

Diagnostic

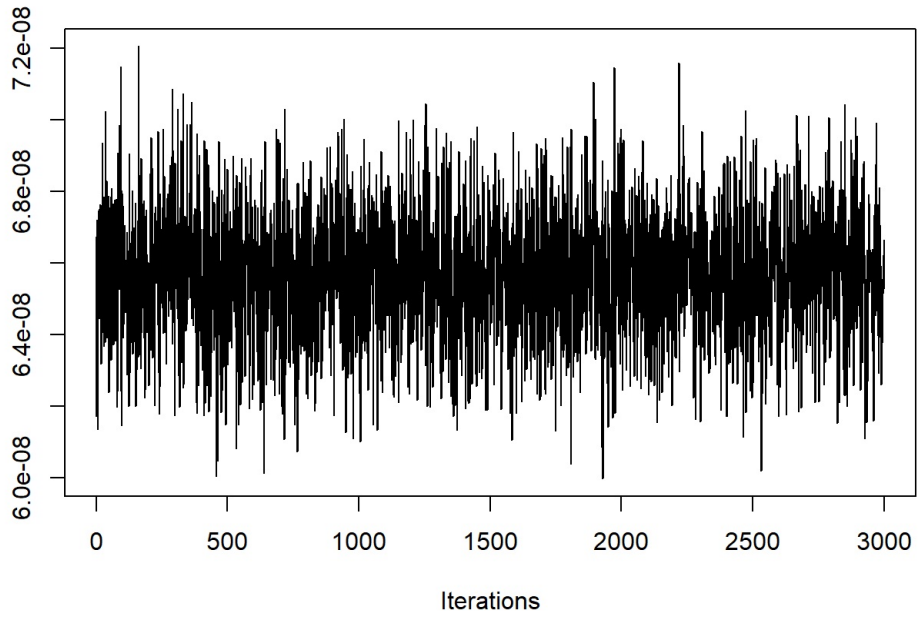
Before using the MCMC output for posterior inference we need to check that the resulting chain provide a appropriate approximation of the true posterior distribution.

MCMC Traceplot

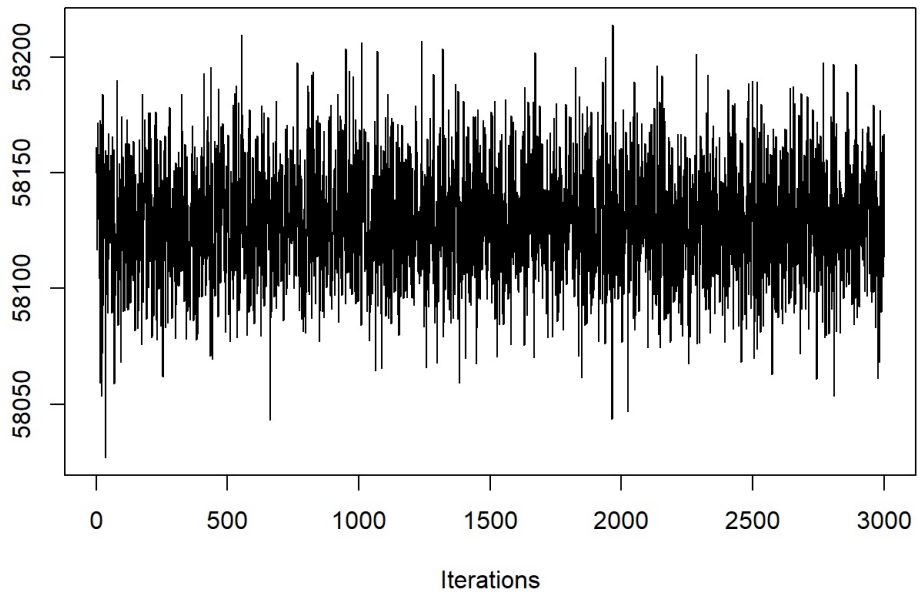
```
par(mfrow=c(3, 4))
for(i in 1:p){
coda::traceplot(mcmc(model1_run$BUGSoutput$sims.list$b[,i]), main=colnames(X_train)[i])
}
```



```
traceplot(mcmc(model1_run$BUGSoutput$sims.list$tau))
```




```
traceplot(mcmc(model1_run$BUGSoutput$sims.list$deviance))
```



No pattern and autocorrelation is detected by the trace-plots.

Auto-Correlation Plot

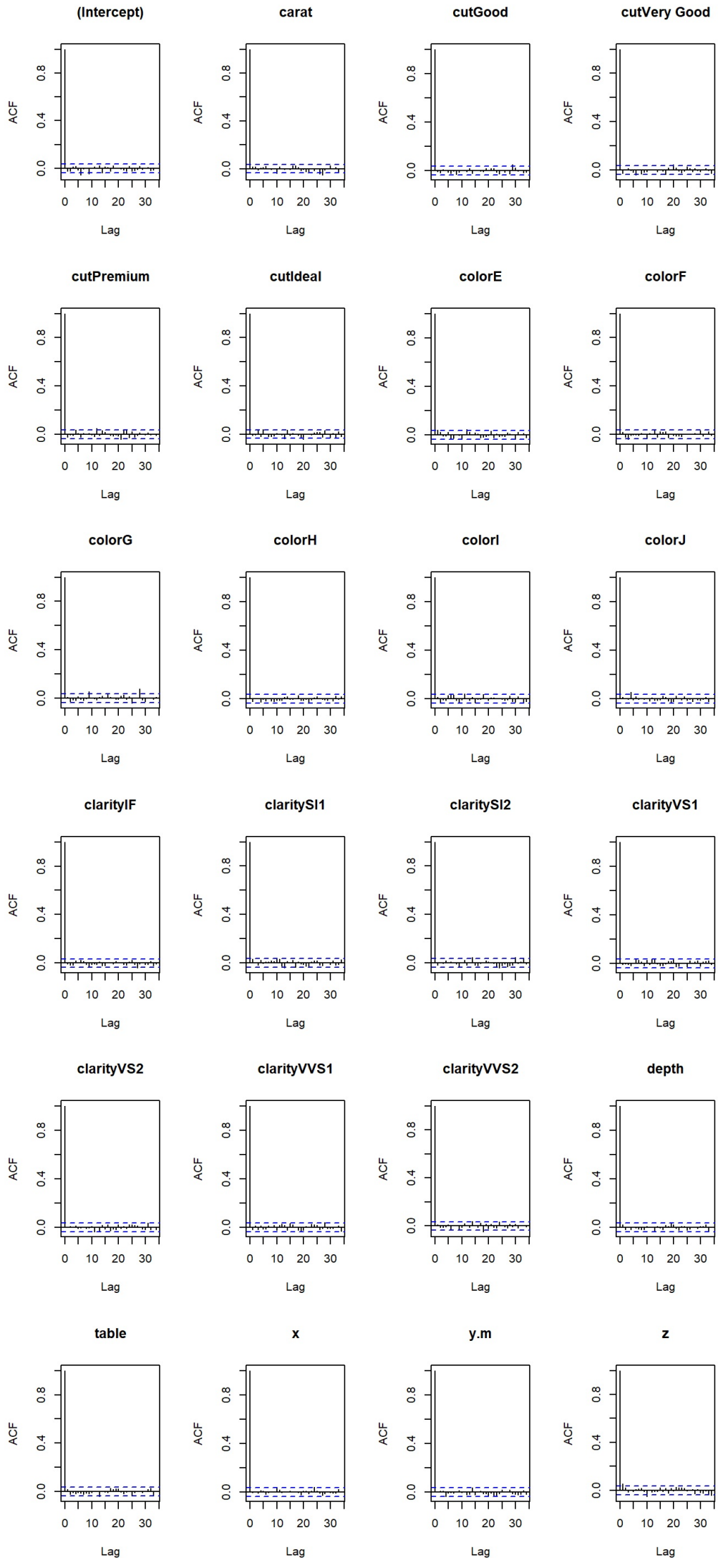
We don't want dependencies in the chain, these in a Markov chain can be measured using autocorrelation.

Autocorrelation of order (lag):

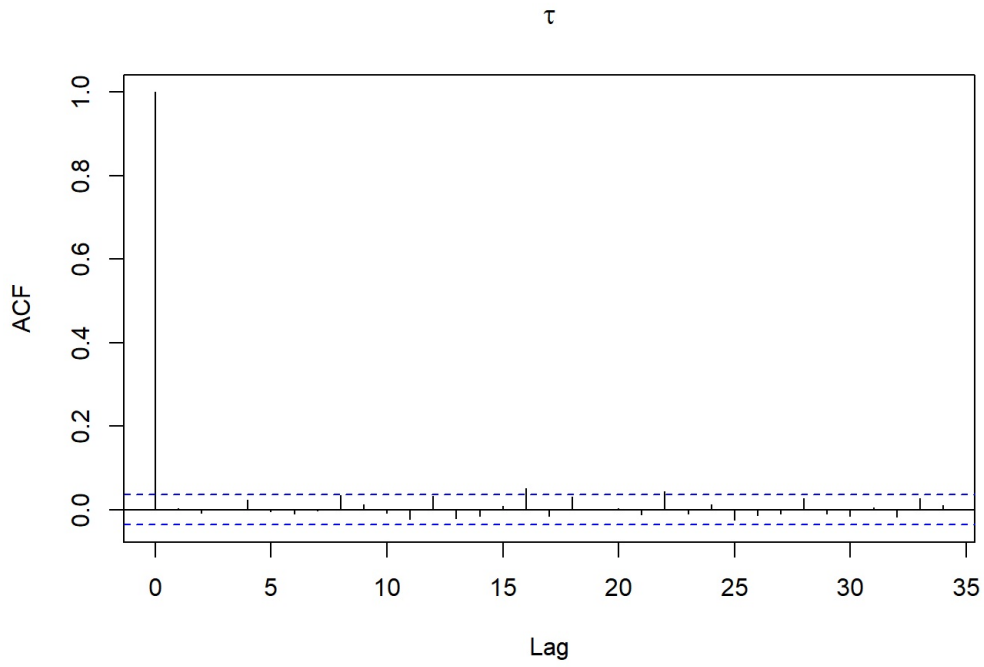
$$acf_k(\theta) = {}^{1}S-k\sum_{S-ks=1}(\theta^{(s)} - \bar{\theta})(\theta^{(s+k)} - \bar{\theta}){}^{1}S-k\sum_{S-ks=1}(\theta^{(s)} - \bar{\theta})^2$$

$$\bar{\theta} = {}^{1}S - {}^{k}S-k\sum s-1$$

```
par(mfrow=c(2,4))
for(i in 1:p){
acf(model1_run$BUGSoutput$sims.list$b[,i], main=colnames(X_train)[i])
}
```


(Intercept)
carat
cutGood
cutVery Good
cutPremium
cutIdeal
colorE
colorF
colorG
colorH
colorI
colorJ
clarityIF
claritySI1
claritySI2
clarityVS1
clarityVS2
clarityVVS1
clarityVVS2
depth
table
x
y.m
z
ACF
Lag
0.0
0.4
0.8
0
10
20
30

```
acf(model1_run$BUGSoutput$sims.list$tau, main=expression(tau))
```



There's not detected high values of acf.

Effective Sample Size

ESS:

$ESS = G1 + 2\sum_{Gg=1}^{acf_g}$

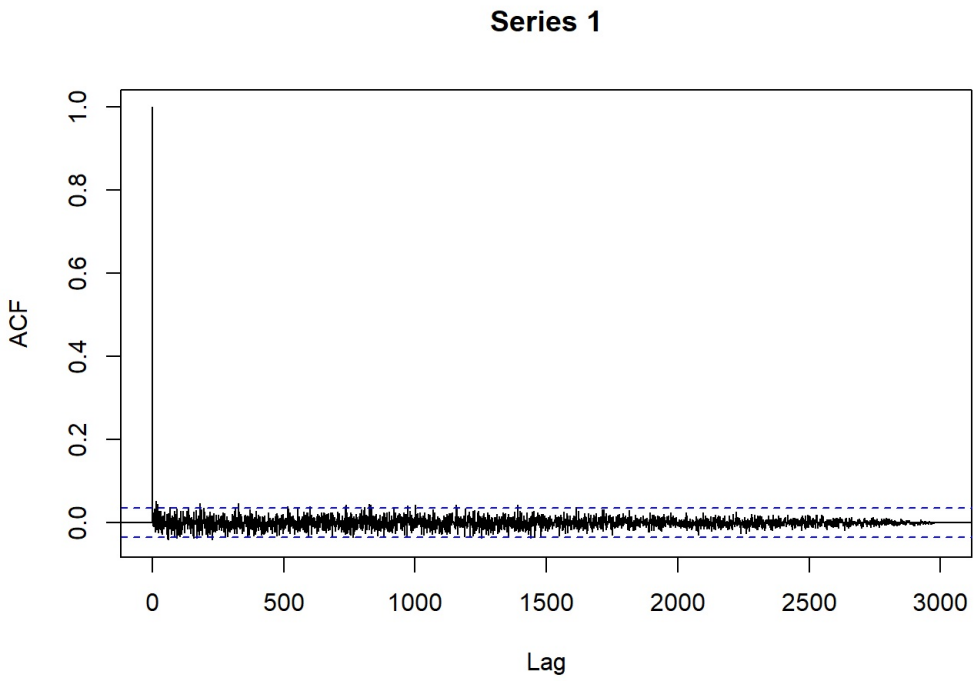
with G: the number of the post burn-in MCMC samples.

```
my_ESS <- function(x, S){
  out = acf(x, lag.max=S)$acf
  return(S/(1 + 2*sum(out)))
}
```

```
for(i in 1:p){
  cat(colnames(X_train)[i],": ",coda::effectiveSize(mcmc(model1_run$BUGSoutput$sims.list$b[,i])), sep="", end="\n
")
}
```

```
## (Intercept): 3517.5
## carat: 3000
## cutGood: 3000
## cutVery Good: 3179.374
## cutPremium: 3000
## cutIdeal: 3000
## colorE: 2770.668
## colorF: 3296.986
## colorG: 3000
## colorH: 3000
## colorI: 2659.214
## colorJ: 2584.626
## clarityIF: 3000
## claritySI1: 2825.429
## claritySI2: 3000
## clarityVS1: 3000
## clarityVS2: 3000
## clarityVVS1: 3000
## clarityVVS2: 3000
## depth: 3000
## table: 3000
## x: 3000
## y.m: 3000
## z: 2679.918
```

```
out <- my_ESS(model1_run$BUGSoutput$sims.list$tau, S=3000)
```




```
cat("tau:", out, end="\n")
```

```
## tau: 1500
```

Geweke Test

Consider two “windows” of the chain:

- The initial %x: θ_I
- The Last y%:θ_L

If the chain is stationaty, the teo corresponding sample means $\bar{\theta}_I$ and $\bar{\theta}_L$ should be equal.

The Geweke statistic is :

$$Z_n = \bar{\theta}_I - \bar{\theta}_L \sqrt{\hat{s}_{2I} + \hat{s}_{2L}} \rightarrow N(0, 1)$$

If $|Z_n| > 1.96$ the hypothesis of the stationary is rejected.

```
for(i in 1:p){
cat('Geweke test for ',colnames(X_train)[i],':',coda::geweke.diag(
model1_run$BUGSoutput$sims.list$b[,i])$z,end="\n")
}
```

```
## Geweke test for  (Intercept) : -0.3699547
## Geweke test for  carat : -0.945198
## Geweke test for  cutGood : -0.3661099
## Geweke test for  cutVery Good : -1.052134
## Geweke test for  cutPremium : -0.6634566
## Geweke test for  cutIdeal : -0.1738825
## Geweke test for  colorE : -0.1210406
## Geweke test for  colorF : 0.2047067
## Geweke test for  colorG : -0.8215438
## Geweke test for  colorH : 0.3977761
## Geweke test for  colorI : -0.6267364
## Geweke test for  colorJ : -0.6116152
## Geweke test for  clarityIF : 0.9907561
## Geweke test for  claritySI1 : -1.854017
## Geweke test for  claritySI2 : 0.434991
## Geweke test for  clarityVS1 : -1.046291
## Geweke test for  clarityVS2 : -1.380999
## Geweke test for  clarityVVS1 : 0.4338231
## Geweke test for  clarityVVS2 : 0.9429859
## Geweke test for  depth : 0.07992018
## Geweke test for  table : -0.2333437
## Geweke test for  x : 0.1984663
## Geweke test for  y.m : -0.6523163
## Geweke test for  z : 1.520926
```

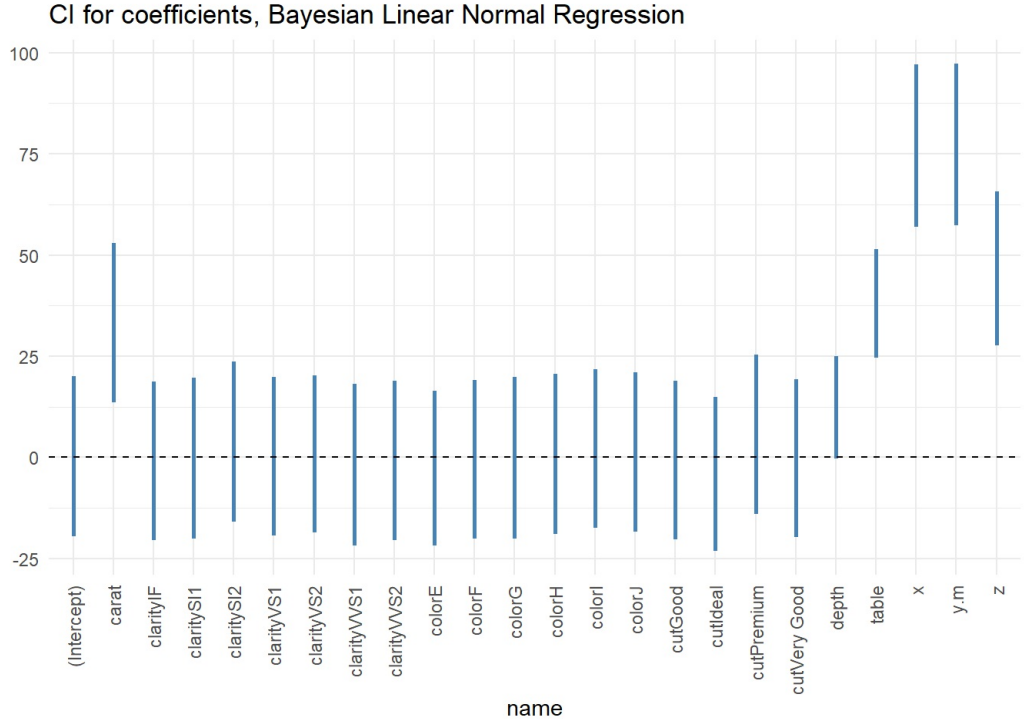
```
cat('Geweke test for Tau:',coda::geweke.diag(
model1_run$BUGSoutput$sims.list$tau^-1)$z)
```

```
## Geweke test for Tau: -0.5857924
```

Parameters Analysis

Significant Variables and Credible Intervals

```
b1_df <- as.data.frame(model1_run$BUGSoutput$sims.list$b)
names(b1_df) <- colnames(X_train)
b1_long <- pivot_longer(b1_df, cols = 1:24) %>%
  group_by(name) %>%
  summarise(q025 = quantile(value, 0.025), q975 = quantile(value, 0.975)) %>%
  ungroup()

ggplot(b1_long, aes(x = name, ymin = q025, ymax = q975)) +
  geom_linerange(size = 1, color = "steelblue") +
  geom_hline(yintercept = 0, linetype = 2) + theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  ggtitle("CI for coefficients, Bayesian Linear Normal Regression")
```



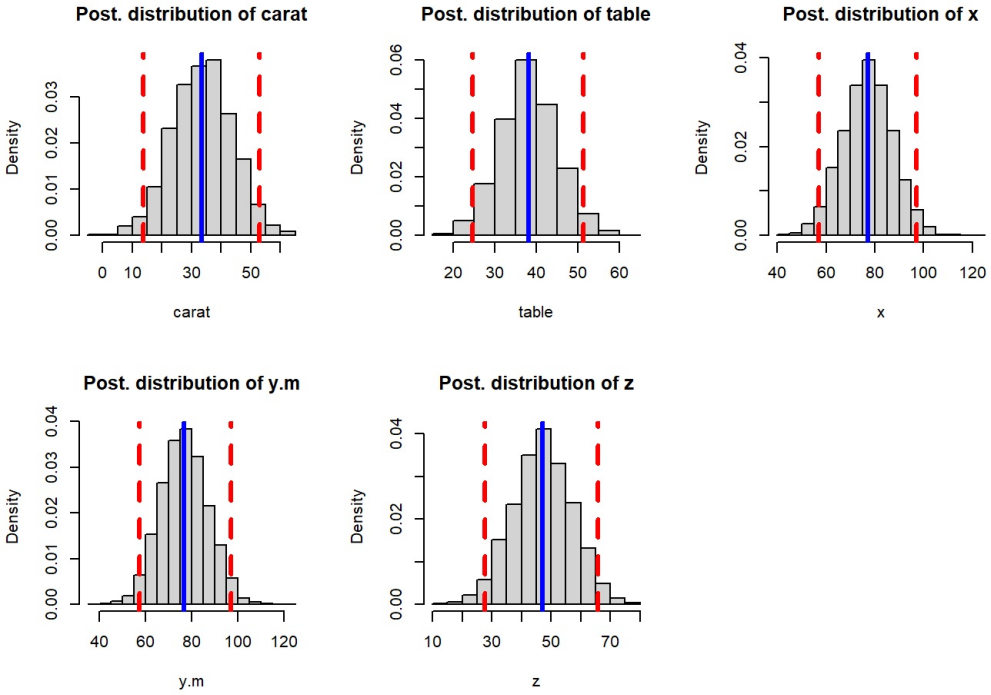
```
resu = as.matrix(colMeans(model1_run$BUGSoutput$sims.list$b),nrow=25)
resu = rbind(resu,colMeans(model1_run$BUGSoutput$sims.list$tau))
rownames(resu) = c(colnames(X_train),"Tau")
colnames(resu) = "Posterior Expectation"

res = t(apply(model1_run$BUGSoutput$sims.list$b,2,quantile,
prob=c(.025,.975)))
res = rbind(res,t(quantile(model1_run$BUGSoutput$sims.list$tau,
prob=c(.025,.975))))
rownames(res) = c(colnames(X_train),"Tau")
colnames(res) = c("2.5 % Quantile","97.5 % Quantile")
knitr::kable(cbind(res,resu), align = "ccc")
```

	2.5 % Quantile	97.5 % Quantile	Posterior Expectation
(Intercept)	-19.5372177	20.0579251	0.1877406
carat	13.6481562	52.8805761	33.5317136
cutGood	-20.3651204	18.8942576	-0.3772706
cutVery Good	-19.8041908	19.2436953	-0.0388490
cutPremium	-14.0502967	25.2888780	5.0470704
cutIdeal	-23.1314444	14.8289088	-4.0359920
colorE	-21.8463369	16.4260917	-2.0123574
colorF	-20.0639737	19.0128990	-0.2863930
colorG	-20.1325307	19.9064304	0.3436131
colorH	-18.9026143	20.5126101	0.4525369
colorI	-17.5204109	21.6950022	1.3588888
colorJ	-18.4930219	21.0368458	1.4649037
clarityIF	-20.4979946	18.6764798	-0.8463640
claritySI1	-20.1115084	19.6415402	-0.1764453
claritySI2	-16.0245963	23.6809647	4.1006121
clarityVS1	-19.2997219	19.8378689	-0.0201296
clarityVS2	-18.5248083	20.1381654	0.4134135
clarityVVS1	-21.7987138	18.0620846	-1.6999570
clarityVVS2	-20.4563016	18.9736724	-1.3405591
depth	-0.2441809	24.8991482	12.1682685
table	24.5791295	51.4089495	38.1472073
x	56.9788281	97.0566373	77.0809743
y.m	57.2858510	97.2223889	76.8179433
z	27.6489323	65.7880054	47.0559618
Tau	0.0000001	0.0000001	0.0000001

From the above result we notice that the majority of the credible intervals do include the zero, meaning that the parameter may not have significant impact on the response variable.

Now we just plot the distribution of significant variables with their CI and mean.

```
par(mfrow=c(2,3))
for(i in c(2, 21, 22, 23, 24)){
hist(model1_run$BUGSoutput$sims.list$b[,i],
main=paste("Post. distribution of", rownames(res)[i]),
xlab = rownames(res)[i],cex=0.8, prob = TRUE)
abline(v=res[i,1], col="red",lty=2, lwd=3)
abline(v=res[i,2], col="red",lty=2, lwd=3)
abline(v=resu[i,1], col="blue", lwd=3)
}
```

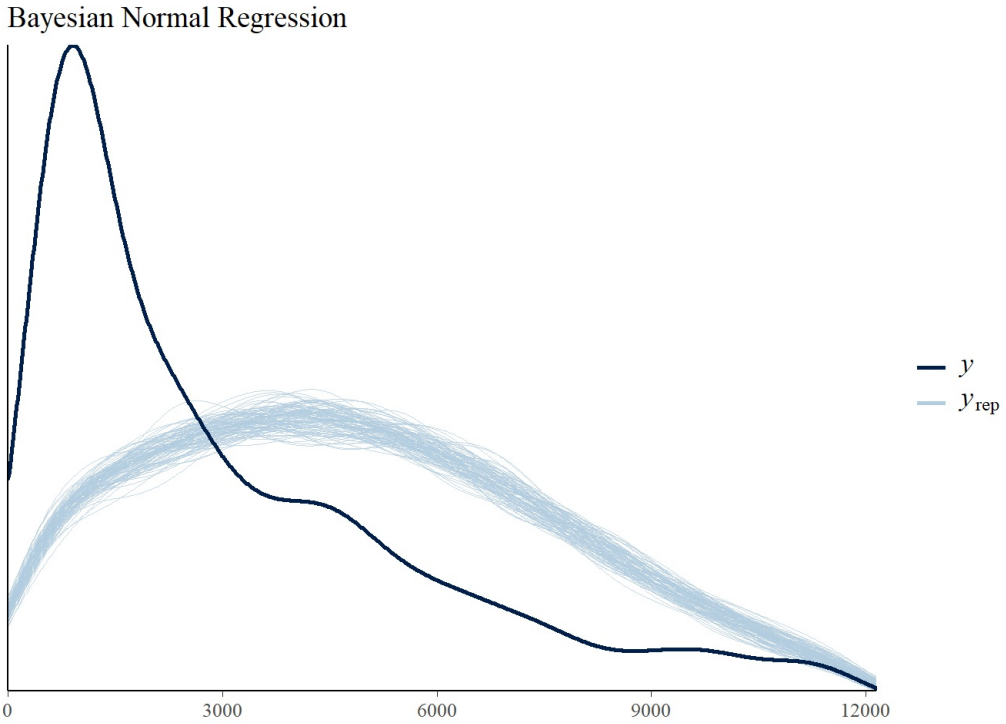


Posterior Predictive Model Checking

```
mu1 <- model1_run$BUGSoutput$sims.list$mu
tau1 <- model1_run$BUGSoutput$sims.list$tau
yrep1 <- t(sapply(1:100, function(i) rnorm(n, mu1[i,], sqrt(1/tau1[i]))))

ppc_dens_overlay(y, yrep1) + ggtitle("Bayesian Normal Regression") + xlim(0, 10.5^4)
```

```
## Warning: Removed 51087 rows containing non-finite values (stat_density).
```

```
## Warning: Removed 203 rows containing non-finite values (stat_density).
```

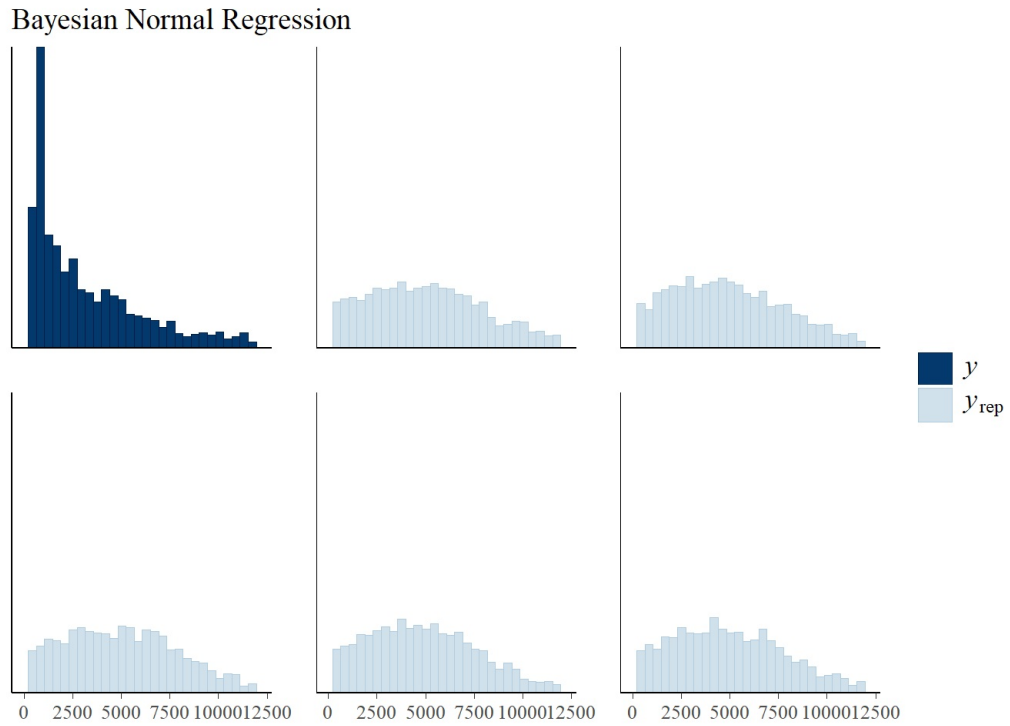



```
ppc_hist(y, yrep1[1:5,]) + ggtitle("Bayesian Normal Regression") + xlim(0, 10.5^4)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 2851 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 12 rows containing missing values (geom_bar).
```



The last result confirm what we think from the beginning, the Bayesian Normal Regression perform very bad for this analysis.

Bayesian t-Student Regression

The next model we consider is the Bayesian t-Student Regression, the response is defined as:

$Y_i \mid \mu_i, \sigma^2, k \sim t_k(\mu_i, \sigma^2)$

and the model:

$\mu_i = \beta_0 + \beta_1 x_i + \ldots + \beta_p x_p$

and priors:

$\sigma \sim U(0, 10^6)$

$\beta_j \sim N(0, 10^6)$

t_k is a non-central and scale t-distribution:

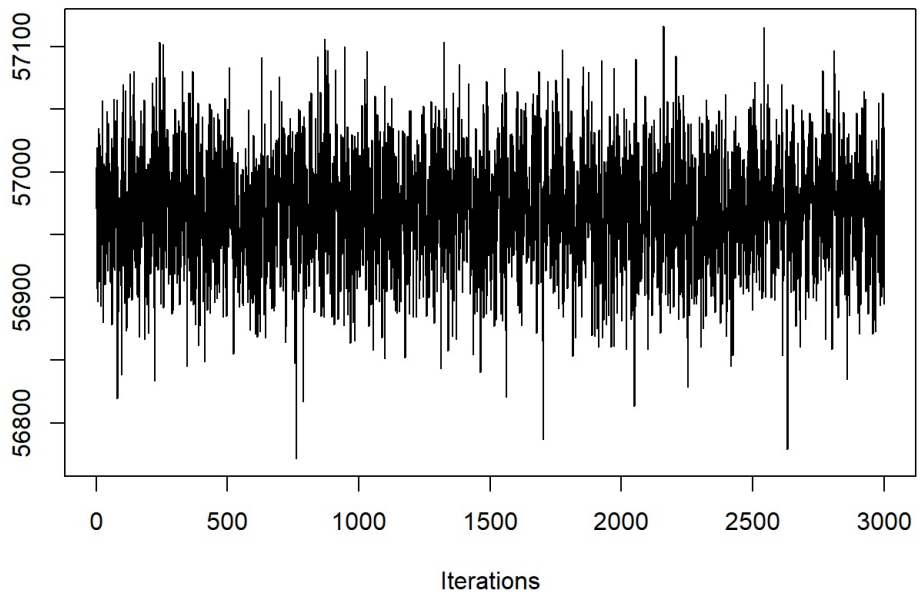
$Y_i = \mu_i + \delta t_k$

Metropolis-Hastings Algorithm

```
model2_code <- function() {
  ## Likelihood
  for (i in 1:n) {
    y[i] ~ dt(mu[i], tau, k)
    mu[i] <- X[i,] %*% b[]
  }
  ## Priors
  for (j in 1:p) {
    b[j] ~ dnorm(0, 0.01)
  }
  tau <- pow(sigma, -2)
  sigma ~ dunif(0,10^6)
  k ~ dgamma(0.04, 0.01)
}

model2_data <- list(y = y, X = X_train, n = n, p = p)
model2_params <- c("b", "mu", "tau", "k")

model2_run <- jags(
  data = model2_data,
  parameters.to.save = model2_params,
  model.file = model2_code,
  n.chains = 1,
  n.iter = 5000,
  n.burnin = 2000,
  n.thin = 1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 3000
##    Unobserved stochastic nodes: 26
##    Total graph size: 81038
##
## Initializing model
```

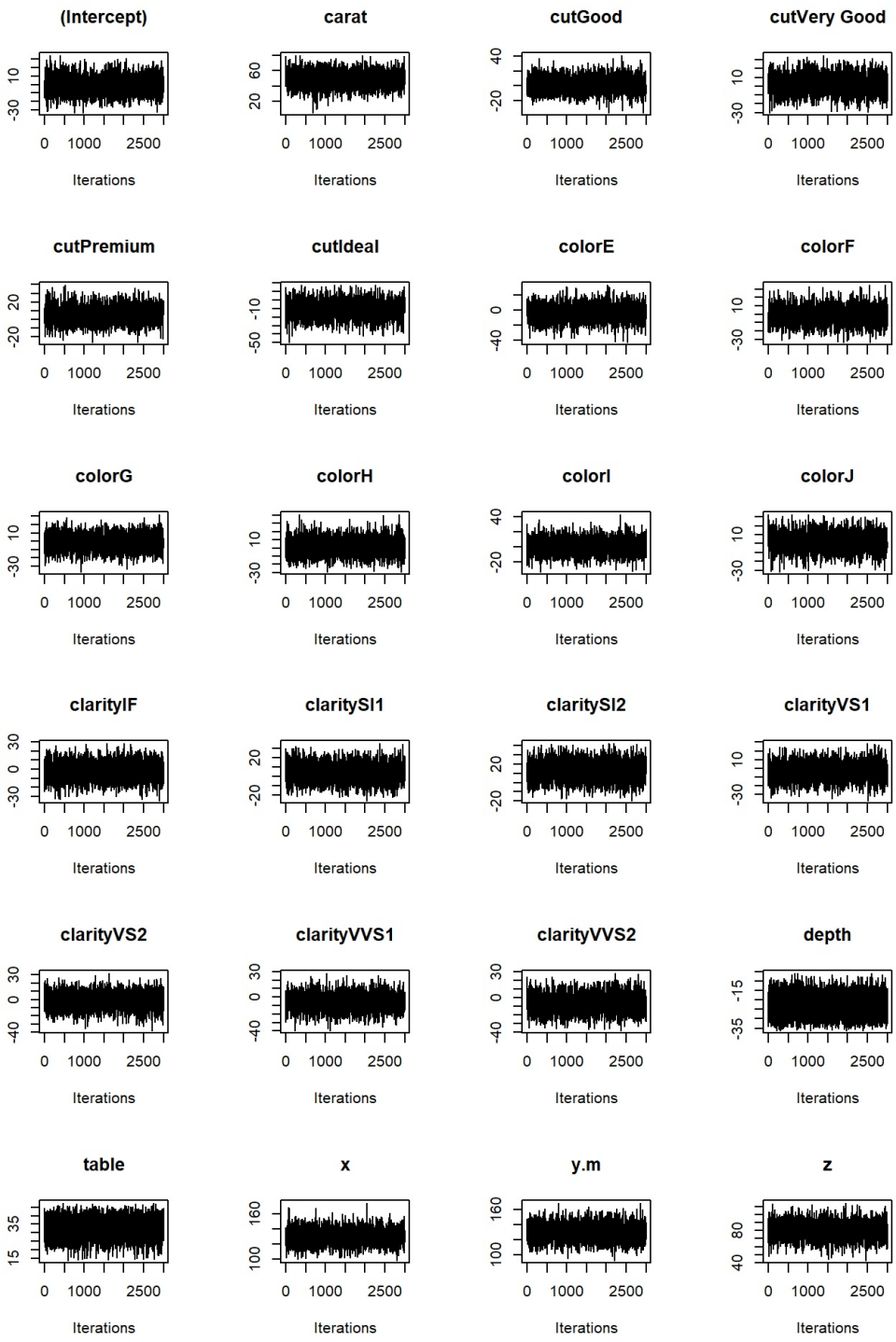
```
traceplot(mcmc(model2_run$BUGSoutput$sims.list$deviance))
```



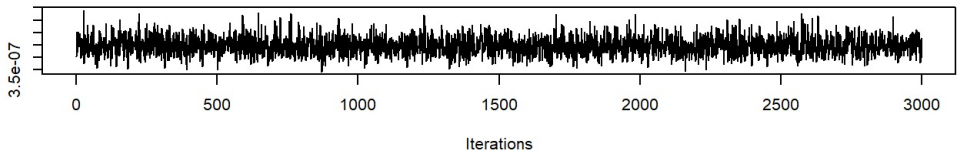
Diagnostic

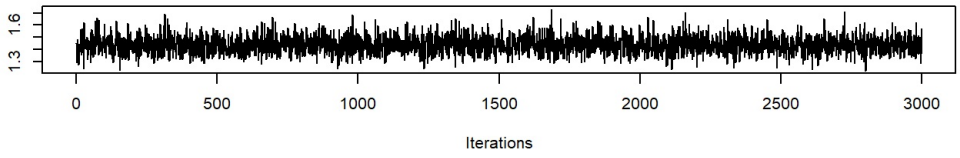
MCMC Traceplot

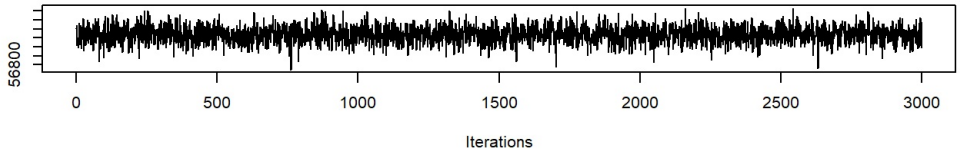
```
par(mfrow=c(3, 4))
for(i in 1:p){
coda::traceplot(mcmc(model2_run$BUGSoutput$sims.list$b[,i]), main=colnames(X_train)[i])
}
```




```
par(mfrow=c(3,1))
traceplot(mcmc(model2_run$BUGSoutput$sims.list$tau))
traceplot(mcmc(model2_run$BUGSoutput$sims.list$k))
traceplot(mcmc(model2_run$BUGSoutput$sims.list$deviance))
```

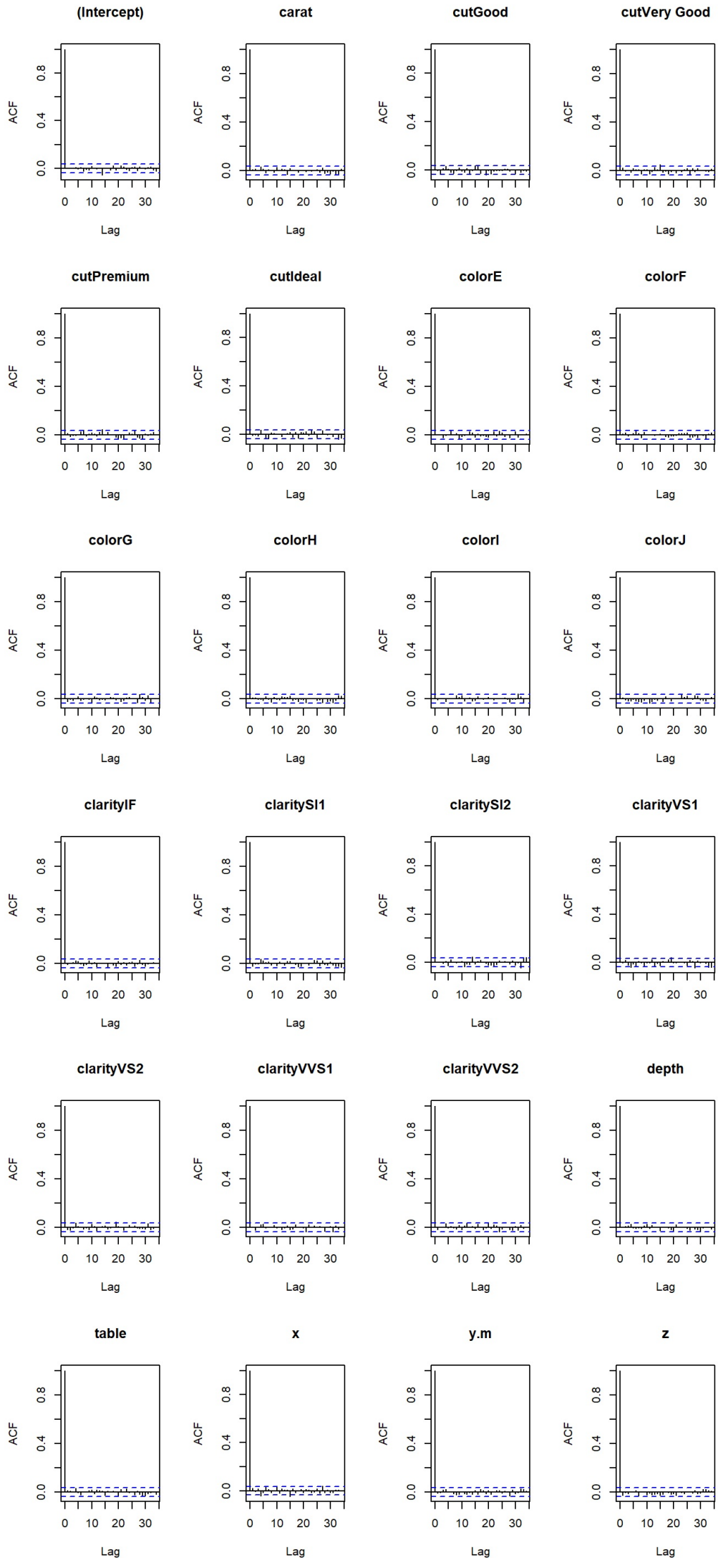




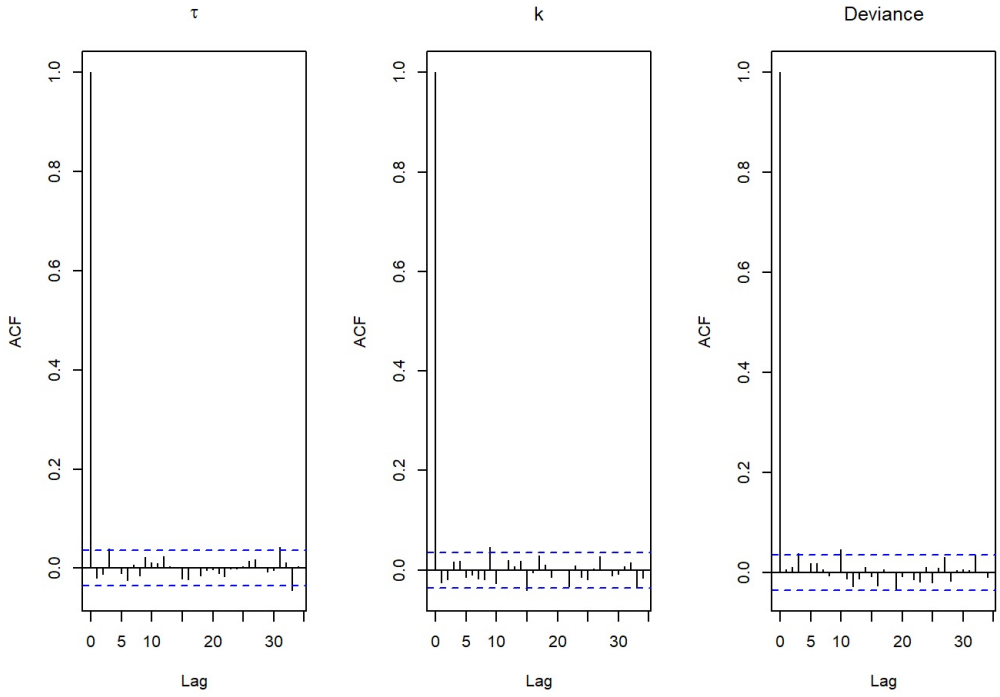


Auto-Correlation Plot

```
par(mfrow=c(2,4))
for(i in 1:p){
acf(model2_run$BUGSoutput$sims.list$b[,i], main=colnames(X_train)[i])
}
```


(Intercept)
carat
cutGood
cutVery Good
cutPremium
cutIdeal
colorE
colorF
colorG
colorH
colorI
colorJ
clarityIF
claritySI1
claritySI2
clarityVS1
clarityVS2
clarityVVS1
clarityVVS2
depth
table
x
y.m
z
ACF
Lag
0.0
0.4
0.8
0
10
20
30

```
par(mfrow=c(1,3))
acf(model2_run$BUGSoutput$sims.list$tau, main=expression(tau))
acf(model2_run$BUGSoutput$sims.list$k, main=expression(k))
acf(model2_run$BUGSoutput$sims.list$deviance, main=expression(Deviance))
```



Effective Sample Size

```
for(i in 1:p){
  cat(colnames(X_train)[i],": ",coda::effectiveSize(mcmc(model2_run$BUGSoutput$sims.list$b[,i])), sep="", end="\n
")
}
```

```
## (Intercept): 3000
## carat: 3000
## cutGood: 3000
## cutVery Good: 3000
## cutPremium: 3000
## cutIdeal: 3000
## colorE: 3000
## colorF: 3000
## colorG: 3000
## colorH: 3000
## colorI: 3000
## colorJ: 3000
## clarityIF: 3000
## claritySI1: 3000
## claritySI2: 3000
## clarityVS1: 3000
## clarityVS2: 3000
## clarityVVS1: 3000
## clarityVVS2: 3000
## depth: 3000
## table: 3000
## x: 3076.154
## y.m: 3000
## z: 3160.268
```

```
cat(colnames(X_train)[i],": ",coda::effectiveSize(mcmc(model2_run$BUGSoutput$sims.list$tau)))
```

```
## z :  2960.182
```

```
cat(colnames(X_train)[i],": ",coda::effectiveSize(mcmc(model2_run$BUGSoutput$sims.list$k)))
```

```
## z :  3000
```

Geweke Test

```
for(i in 1:p){
cat('Geweke test for ',colnames(X_train)[i],':',coda::geweke.diag(
model2_run$BUGSoutput$sims.list$b[,i])$z,end="\n")
}
```

```
## Geweke test for  (Intercept) : 2.671456
## Geweke test for  carat : 1.385375
## Geweke test for  cutGood : 1.000918
## Geweke test for  cutVery Good : 0.1397227
## Geweke test for  cutPremium : -0.9939059
## Geweke test for  cutIdeal : 0.20345
## Geweke test for  colorE : -1.088541
## Geweke test for  colorF : 0.9242661
## Geweke test for  colorG : -1.730148
## Geweke test for  colorH : -0.05912712
## Geweke test for  colorI : -1.309157
## Geweke test for  colorJ : 0.5172347
## Geweke test for  clarityIF : -0.5117547
## Geweke test for  claritySI1 : 1.352052
## Geweke test for  claritySI2 : -1.687405
## Geweke test for  clarityVS1 : -1.275791
## Geweke test for  clarityVS2 : 0.2543577
## Geweke test for  clarityVVS1 : -0.2442372
## Geweke test for  clarityVVS2 : -0.4417202
## Geweke test for  depth : -0.081409
## Geweke test for  table : -0.1023763
## Geweke test for  x : 0.4564869
## Geweke test for  y.m : -1.257036
## Geweke test for  z : -0.6957297
```

```
cat('Geweke test for n0:',coda::geweke.diag(
model2_run$BUGSoutput$sims.list$tau)$z)
```

```
## Geweke test for n0: 2.039605
```

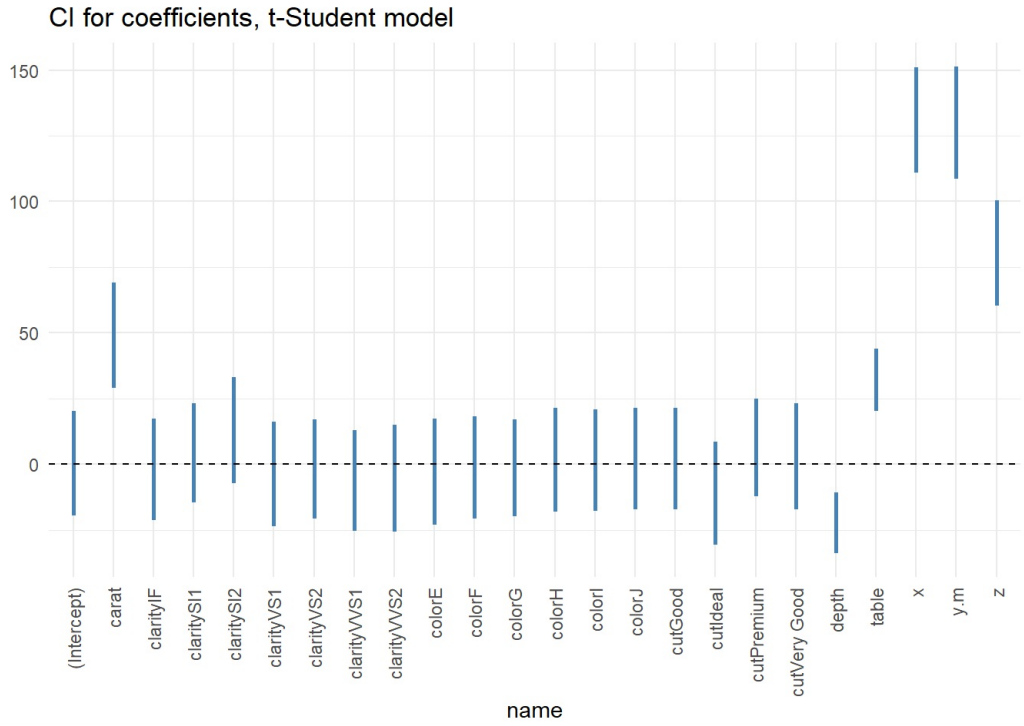
```
cat('Geweke test for n0:',coda::geweke.diag(
model2_run$BUGSoutput$sims.list$k)$z)
```

```
## Geweke test for n0: -1.205112
```

Parameters Analysis

Significant Variables and Credible Intervals

```
b2_df <- as.data.frame(model2_run$BUGSoutput$sims.list$b)
names(b2_df) <- colnames(X_train)
b2_long <- pivot_longer(b2_df, cols = 1:24) %>%
  group_by(name) %>%
  summarise(q025 = quantile(value, 0.025), q975 = quantile(value, 0.975)) %>%
  ungroup()

ggplot(b2_long, aes(x = name, ymin = q025, ymax = q975)) +
  geom_linerange(size = 1, color = "steelblue") +
  geom_hline(yintercept = 0, linetype = 2) + theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  ggtitle("CI for coefficients, t-Student model")
```



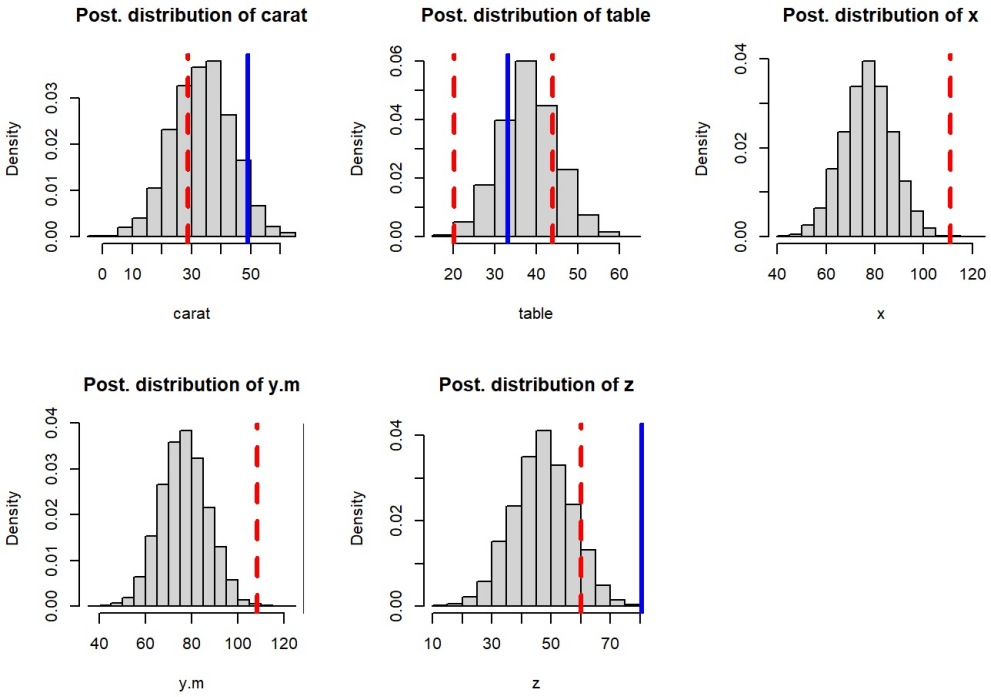

```
resu = as.matrix(colMeans(model2_run$BUGSoutput$sims.list$b),nrow=25)
resu = rbind(resu,colMeans(model2_run$BUGSoutput$sims.list$tau))
rownames(resu) = c(colnames(X_train),"Tau")
colnames(resu) = "Posterior Expectation"

res = t(apply(model2_run$BUGSoutput$sims.list$b,2,quantile,
prob=c(.025,.975)))
res = rbind(res,t(quantile(model2_run$BUGSoutput$sims.list$tau,
prob=c(.025,.975))))
rownames(res) = c(colnames(X_train),"Tau")
colnames(res) = c("2.5 % Quantile","97.5 % Quantile")
knitr::kable(cbind(res,resu), align = "ccc")
```

	2.5 % Quantile	97.5 % Quantile	Posterior Expectation
(Intercept)	-19.5636495	20.1211934	-0.0623434
carat	28.8589306	69.0366534	49.1309675
cutGood	-17.3128249	21.4653128	1.8771222
cutVery Good	-17.1691390	23.0935374	2.3870017
cutPremium	-12.2125999	24.8782552	5.5185152
cutIdeal	-30.6121096	8.5942067	-11.2492167
colorE	-23.0989576	17.2265158	-2.6945445
colorF	-20.6138892	18.1314649	-0.8188580
colorG	-19.7541213	17.1239922	-1.2086137
colorH	-18.0861521	21.4912786	1.8576410
colorI	-17.9447717	20.7991917	1.1723126
colorJ	-17.3284741	21.5383286	2.0975906
clarityIF	-21.4231599	17.2187806	-2.0036454
claritySI1	-14.5807372	23.2434010	4.4004950
claritySI2	-7.3158440	33.2206968	12.8086672
clarityVS1	-23.6343526	16.0450898	-3.1443542
clarityVS2	-20.7959904	17.1495199	-1.6903368
clarityVVS1	-25.2918272	12.9837953	-5.8479932
clarityVVS2	-25.6071090	14.9231228	-5.3633179
depth	-33.8547787	-10.6802616	-23.0018794
table	20.2191825	43.8641186	33.0806911

x	111.0613253	150.9068658	130.4510397
y.m	108.5038981	151.2211881	129.4695357
z	60.1985612	100.3481301	80.6333117
Tau	0.0000004	0.0000005	0.0000004

Now we just plot the distribution of significant variables with their CI and mean.

```
par(mfrow=c(2,3))
for(i in c(2, 21, 22, 23, 24)){
hist(model1_run$BUGSoutput$sims.list$b[,i],
main=paste("Post. distribution of", rownames(res)[i]),
xlab = rownames(res)[i],cex=0.8, prob = TRUE)
abline(v=res[i,1], col="red",lty=2, lwd=3)
abline(v=res[i,2], col="red",lty=2, lwd=3)
abline(v=resu[i,1], col="blue", lwd=3)
}
```

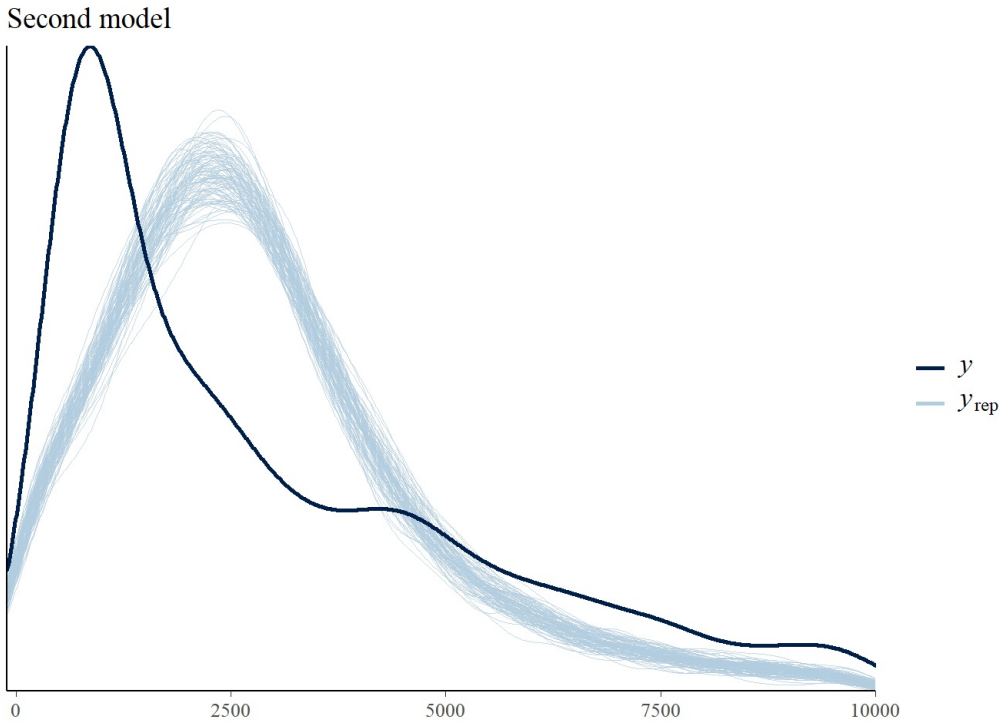


Posterior Predictive Model Checking

```
mu2 <- model2_run$BUGSoutput$sims.list$mu
tau2 <- model2_run$BUGSoutput$sims.list$tau
k2 <- model2_run$BUGSoutput$sims.list$k
yrep2 <- t(sapply(1:100, function(i) mu2[i,] + sqrt(1/tau2[i])*rt(n,k2[i])))

ppc_dens_overlay(y, yrep2) + ggtitle("Second model") + xlim(-10^2, 10^4)
```

```
## Warning: Removed 55864 rows containing non-finite values (stat_density).
```

```
## Warning: Removed 309 rows containing non-finite values (stat_density).
```

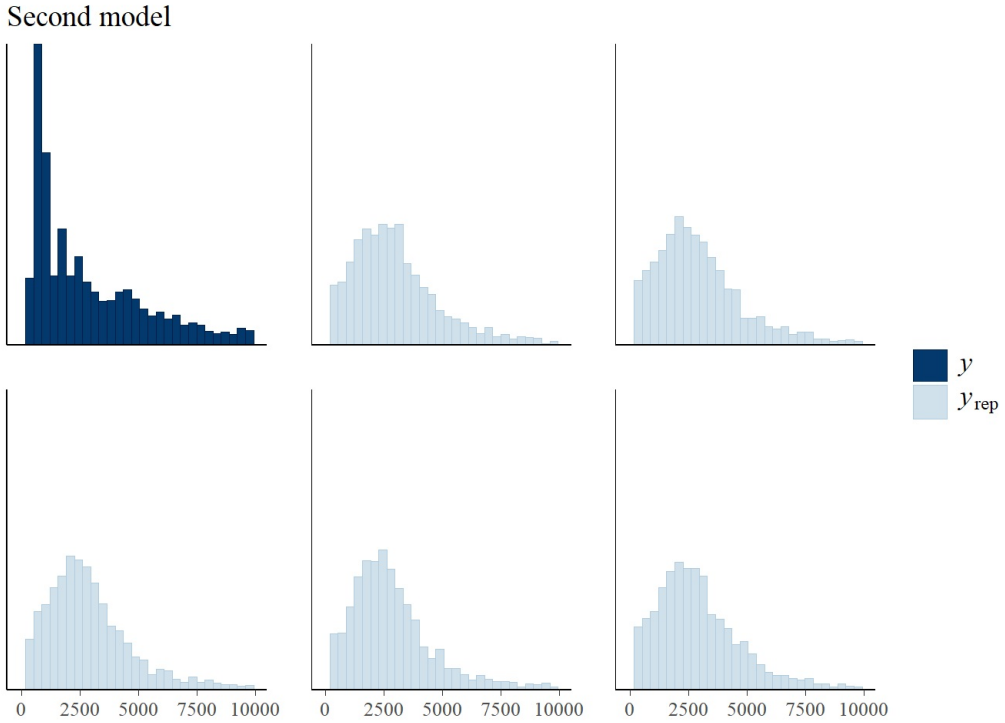



```
ppc_hist(y, yrep2[1:5,]) + ggtitle("Second model") + xlim(-10^2, 10^4)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 3163 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 12 rows containing missing values (geom_bar).
```



The last result confirm what we think from the beginning, the Bayesian t-Student Regression perform very bad for this analysis.

Bayesian Gamma Regression

Considering we have a positive response, we can implement a Bayesian Gamma Regression, which we think can perform better than the others two models.

If:

$Y_i \,|\, \mu_i, n_0 \sim \text{Gamma}(\mu_i\, n_0, n_0)$ with $Y \in (0^+, \infty)$

The model is:

$h(\mu_i) = \beta_0 + \beta_1 x + \ldots + \beta_p x_{ip}$

Where p is the number of predictors.

The regression coefficient may be given negative values, so we have to use a link function, in this case the link function of a gamma distribution is $h(\mu_i) = \log(\mu_i)$.

Furthermore we know that:

$E[Y_i \mid \mu_i, n_0] = \mu_i$

Which is the mean of this distribution.

and:

$Var[Y_i \mid \mu_i, n_0] = [\mu_i n_0 n_0]^2 = \mu_i n_0$

Where n_0 is a value which will decrease the variance.

```
X3_train <- train %>%
  mutate(across(c(6:11), log, .names = "log_{.col}"), .keep = "unused") %>%
  mutate(across(c(1:2), log, .names = "log_{.col}"), .keep = "unused")
X3_train <- X3_train[c(1:9,11)]
X3_train <- model.matrix(log_price ~ ., data = X3_train)
```

Metropolis-Hastings Algorithm

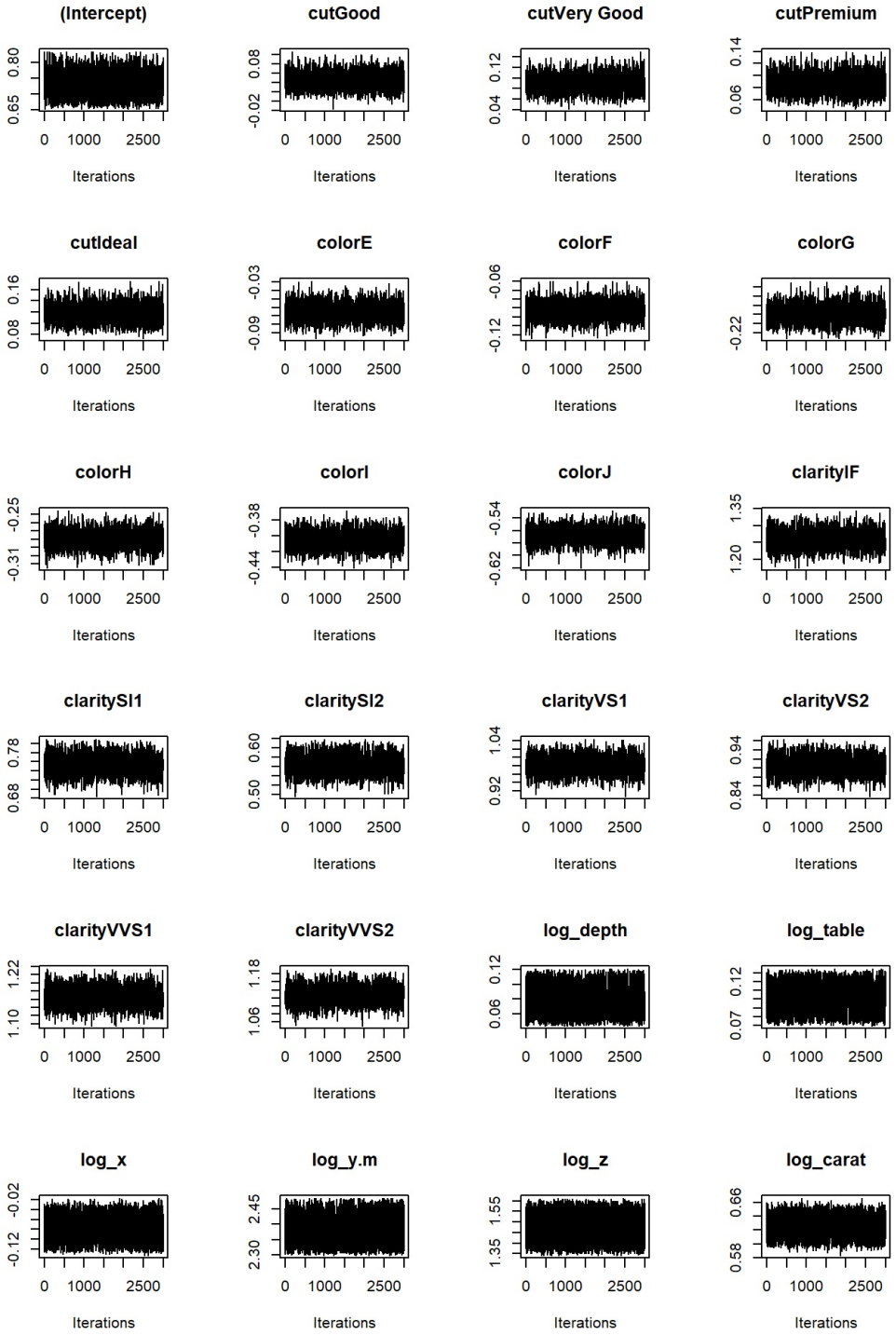
```
p = ncol(X3_train)
n= nrow(X3_train)

model3_code <- function() {
  ## Likelihood
  for (i in 1:n) {
    y[i] ~ dgamma(mu[i]*n0, n0)
    log(mu[i]) <- X[i,] %*% b[]
  }
  ## Priors
  for (j in 1:p) {
    b[j] ~ dnorm(0, 0.01)
  }
  n0 ~ dgamma(0.1, 0.1)
}

model3_data <- list(y = y, X = X3_train, n = n, p = p)
model3_params <- c("b", "mu", "n0")

set.seed(123)

model3_run <- jags(
  data = model3_data,
  parameters.to.save = model3_params,
  model.file = model3_code,
  n.chains = 1,
  n.iter = 5000,
  n.burnin = 2000,
  n.thin = 1,
  jags.seed = 101
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 3000
##    Unobserved stochastic nodes: 25
##    Total graph size: 87031
##
## Initializing model
```

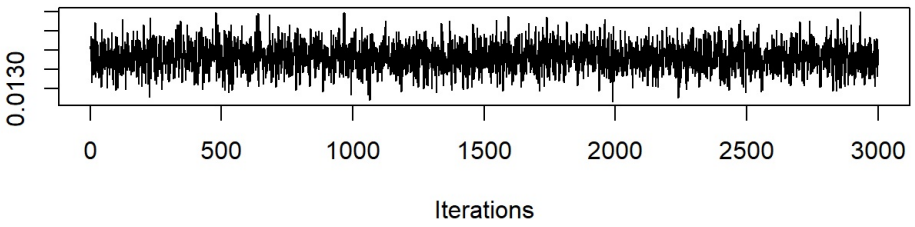
Diagnostic

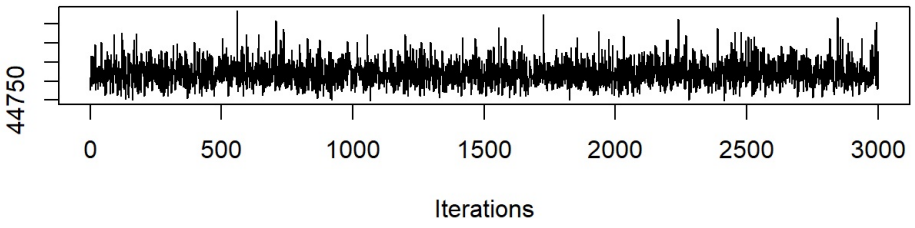
MCMC Traceplot

```
par(mfrow=c(3, 4))
for(i in 1:p){
coda::traceplot(mcmc(model3_run$BUGSoutput$sims.list$b[,i]), main=colnames(X3_train)[i])
}
```



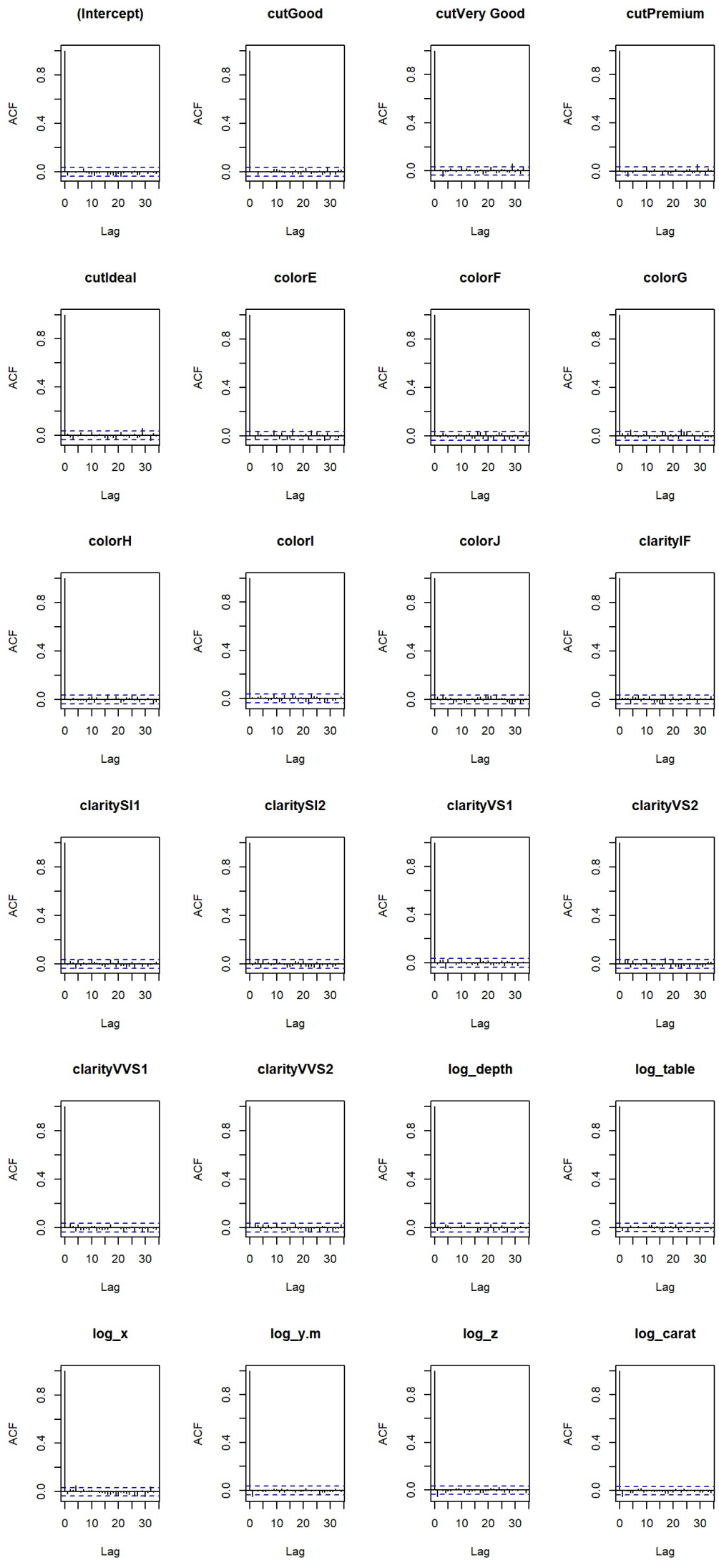

```
par(mfrow=c(2,1))
traceplot(mcmc(model3_run$BUGSoutput$sims.list$n0))
traceplot(mcmc(model3_run$BUGSoutput$sims.list$deviance))
```



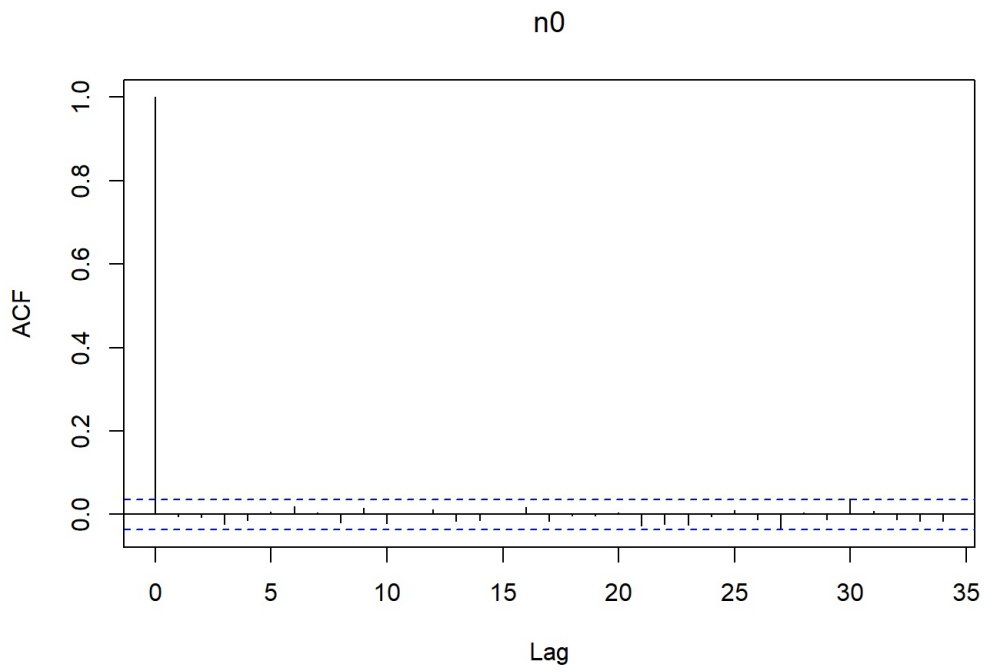


Auto-Correlation Plot

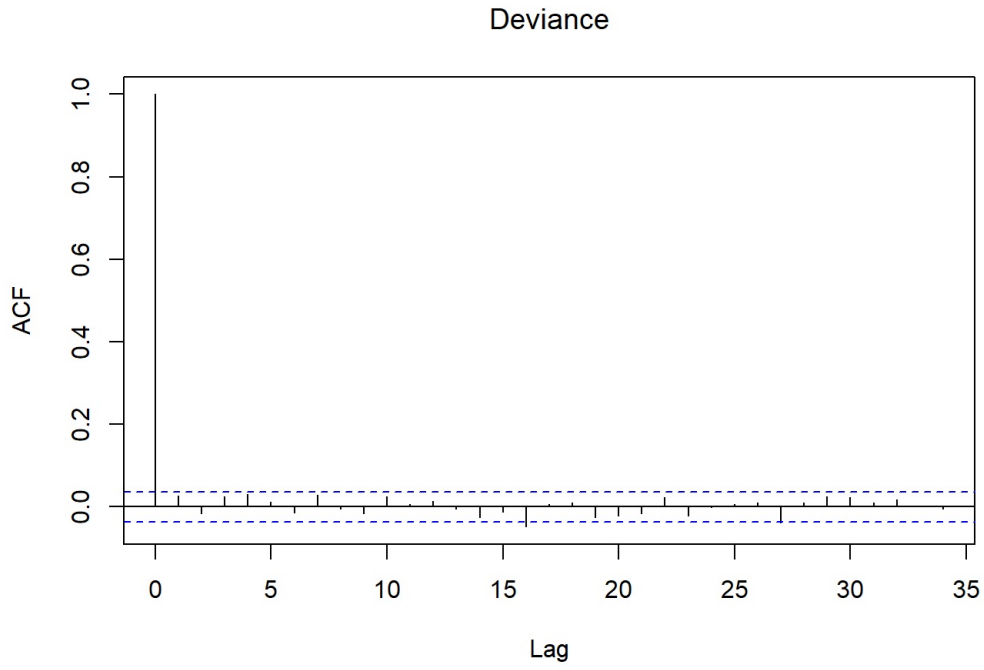
```
par(mfrow=c(2,4))
for(i in 1:p){
acf(model3_run$BUGSoutput$sims.list$b[,i], main=colnames(X3_train)[i])
}
```


(Intercept)
cutGood
cutVery Good
cutPremium
cutIdeal
colorE
colorF
colorG
colorH
colorI
colorJ
clarityIF
claritySI1
claritySI2
clarityVS1
clarityVS2
clarityVVS1
clarityVVS2
log_depth
log_table
log_x
log_y.m
log_z
log_carat
ACF
Lag
0.0
0.4
0.8
0
10
20
30

```
acf(model3_run$BUGSoutput$sims.list$n0, main=expression(n0))
```




```
acf(model3_run$BUGSoutput$sims.list$deviance, main=expression(Deviance))
```



Effective Sample Size

```
for(i in 1:p){
  cat(colnames(X3_train)[i],": ",coda::effectiveSize(mcmc(model3_run$BUGSoutput$sims.list$b[,i])), sep="", end="\
n")
}
```

```
## (Intercept): 3163.14
## cutGood: 3000
## cutVery Good: 3214.13
## cutPremium: 3000
## cutIdeal: 3000
## colorE: 3000
## colorF: 3000
## colorG: 2540.606
## colorH: 3000
## colorI: 3000
## colorJ: 3000
## clarityIF: 3000
## claritySI1: 2841.389
## claritySI2: 2656.586
## clarityVS1: 2855.949
## clarityVS2: 2609.59
## clarityVVS1: 2774.974
## clarityVVS2: 3000
## log_depth: 3000
## log_table: 3158.613
## log_x: 2593.545
## log_y.m: 3327.723
## log_z: 3342.368
## log_carat: 3306.218
```

```
cat(colnames(X_train)[i],": ",coda::effectiveSize(mcmc(model3_run$BUGSoutput$sims.list$n0)))
```

```
## z :  3000
```

Geweke Test

```
for(i in 1:p){
cat('Geweke test for ',colnames(X_train)[i],':',coda::geweke.diag(
model3_run$BUGSoutput$sims.list$b[,i])$z,end="\n")
}
```

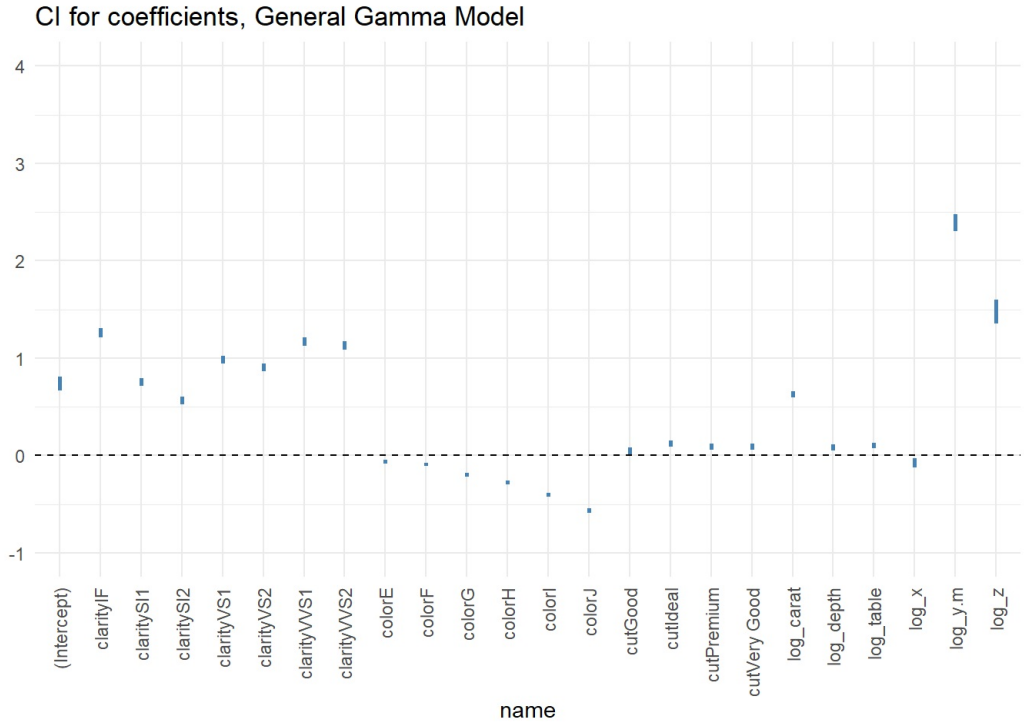
```
## Geweke test for  (Intercept) : 0.6056155
## Geweke test for  carat : 0.2559008
## Geweke test for  cutGood : -0.08414923
## Geweke test for  cutVery Good : -0.485834
## Geweke test for  cutPremium : -0.02361099
## Geweke test for  cutIdeal : -1.017921
## Geweke test for  colorE : -1.098964
## Geweke test for  colorF : -1.474457
## Geweke test for  colorG : -1.417461
## Geweke test for  colorH : -1.536666
## Geweke test for  colorI : -1.269694
## Geweke test for  colorJ : 0.01349285
## Geweke test for  clarityIF : -0.04543351
## Geweke test for  claritySI1 : 0.0259663
## Geweke test for  claritySI2 : 0.100419
## Geweke test for  clarityVS1 : 0.04291253
## Geweke test for  clarityVS2 : 1.088134
## Geweke test for  clarityVVS1 : 0.1084472
## Geweke test for  clarityVVS2 : 2.111318
## Geweke test for  depth : -2.213097
## Geweke test for  table : -0.8331172
## Geweke test for  x : 1.688804
## Geweke test for  y.m : -1.555615
## Geweke test for  z : 1.909089
```

```
cat('Geweke test for n0:',coda::geweke.diag(
model3_run$BUGSoutput$sims.list$n0)$z)
```

```
## Geweke test for n0: -0.4282726
```

Parameters Analysis

```
b3_df <- as.data.frame(model3_run$BUGSoutput$sims.list$b)
names(b3_df) <- colnames(X3_train)
b3_long <- pivot_longer(b3_df, cols = 1:24) %>%
  group_by(name) %>%
  summarise(q025 = quantile(value, 0.025), q975 = quantile(value, 0.975)) %>%
  ungroup()

ggplot(b3_long, aes(x = name, ymin = q025, ymax = q975)) +
  geom_linerange(size = 1, color = "steelblue") + ylim(-1, 4) +
  geom_hline(yintercept = 0, linetype = 2) + theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  ggtitle("CI for coefficients, General Gamma Model")
```



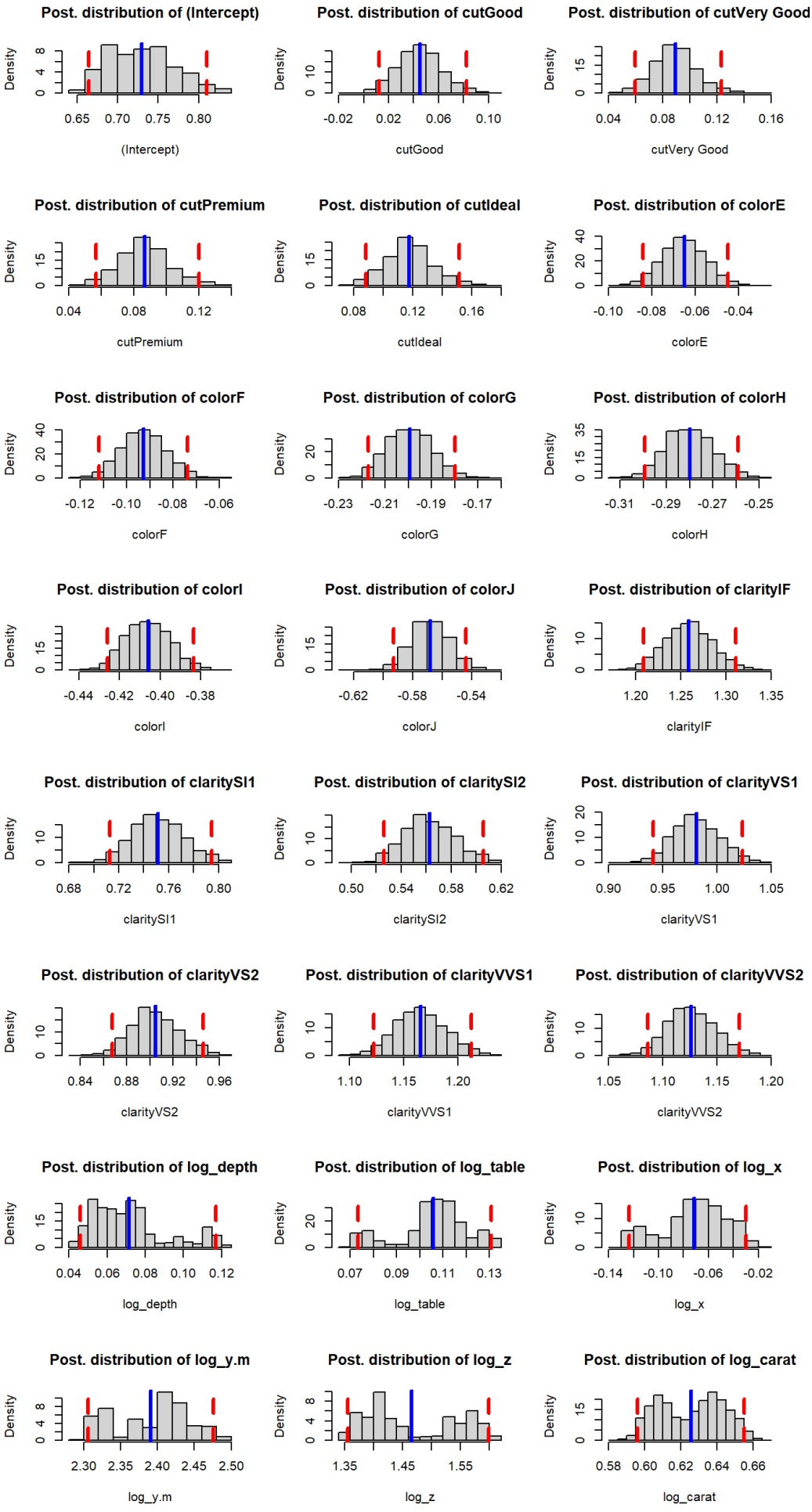
Significant Variables and Credible Intervals

```
resu = as.matrix(colMeans(model3_run$BUGSoutput$sims.list$b),nrow=25)
resu = rbind(resu,colMeans(model3_run$BUGSoutput$sims.list$n0))
rownames(resu) = c(colnames(X3_train),"n0")
colnames(resu) = "Posterior Expectation"
res = t(apply(model3_run$BUGSoutput$sims.list$b,2,quantile,
prob=c(.025,.975)))
res = rbind(res,t(quantile(model3_run$BUGSoutput$sims.list$n0,
prob=c(.025,.975))))
rownames(res) = c(colnames(X3_train),"n0")
colnames(res) = c("2.5 % Quantile","97.5 % Quantile")
knitr::kable(cbind(res,resu), align = "lll")
```

	2.5 % Quantile	97.5 % Quantile	Posterior Expectation
(Intercept)	0.6642219	0.8098201	0.7298232
cutGood	0.0120286	0.0820369	0.0448807
cutVery Good	0.0596691	0.1233259	0.0891581
cutPremium	0.0566946	0.1200438	0.0865740
cutIdeal	0.0881223	0.1515398	0.1177684
colorE	-0.0842602	-0.0448993	-0.0650904
colorF	-0.1120880	-0.0739089	-0.0928307
colorG	-0.2172593	-0.1797595	-0.1994749
colorH	-0.2994491	-0.2590662	-0.2800461
colorI	-0.4261073	-0.3835657	-0.4059762
colorJ	-0.5930453	-0.5439710	-0.5680548
clarityIF	1.2089003	1.3108427	1.2586436

claritySI1	0.7128036	0.7940727	0.7515802
claritySI2	0.5261384	0.6057952	0.5628522
clarityVS1	0.9409111	1.0236809	0.9810276
clarityVS2	0.8673453	0.9458321	0.9046974
clarityVVS1	1.1221820	1.2121676	1.1655264
clarityVVS2	1.0858931	1.1706673	1.1262680
log_depth	0.0459892	0.1169972	0.0715422
log_table	0.0732416	0.1307598	0.1056733
log_x	-0.1235065	-0.0299334	-0.0714083
log_y.m	2.3058861	2.4756407	2.3911957
log_z	1.3558353	1.5981882	1.4656791
log_carat	0.5961292	0.6552156	0.6256332
n0	0.0131058	0.0145172	0.0138246

```
par(mfrow=c(3,3))
for(i in 1:24){
hist(model3_run$BUGSoutput$sims.list$b[,i],
main=paste("Post. distribution of", rownames(res)[i]),
xlab = rownames(res)[i],cex=0.8, prob = TRUE)
abline(v=res[i,1], col="red",lty=2, lwd=3)
abline(v=res[i,2], col="red",lty=2, lwd=3)
abline(v=resu[i,1], col="blue", lwd=3)
}
```

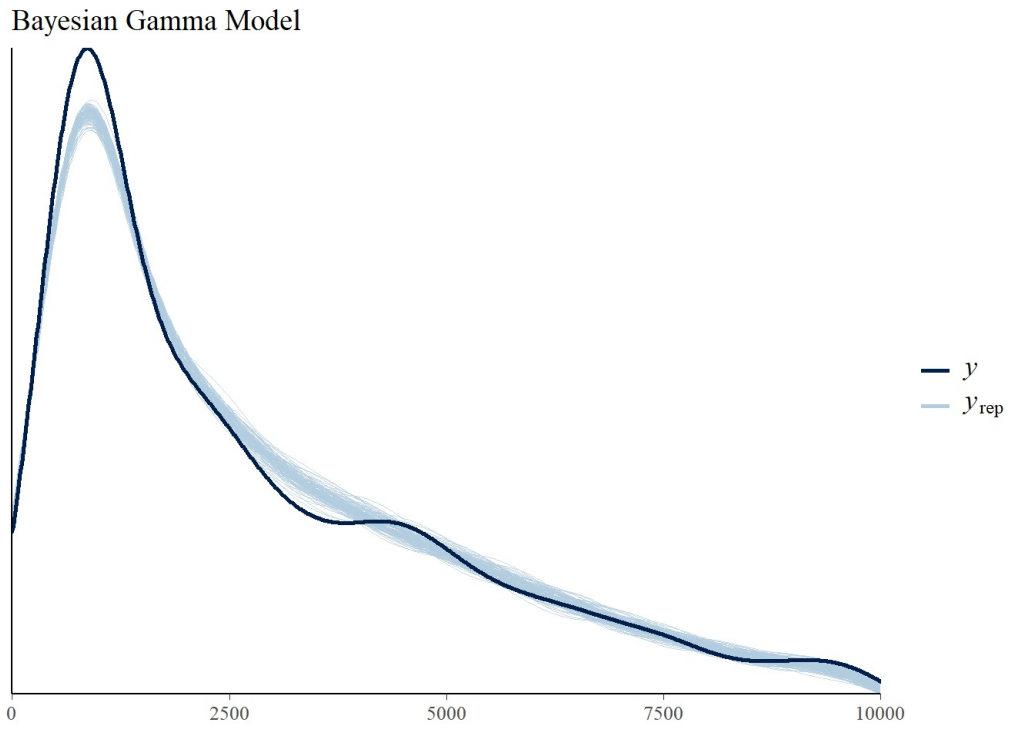

Post. distribution of (Intercept)
Density
(Intercept)
Post. distribution of cutGood
cutGood
Post. distribution of cutVery Good
cutVery Good
Post. distribution of cutPremium
cutPremium
Post. distribution of cutIdeal
cutIdeal
Post. distribution of colorE
colorE
Post. distribution of colorF
colorF
Post. distribution of colorG
colorG
Post. distribution of colorH
colorH
Post. distribution of colorI
colorI
Post. distribution of colorJ
colorJ
Post. distribution of clarityIF
clarityIF
Post. distribution of claritySI1
claritySI1
Post. distribution of claritySI2
claritySI2
Post. distribution of clarityVS1
clarityVS1
Post. distribution of clarityVS2
clarityVS2
Post. distribution of clarityVVS1
clarityVVS1
Post. distribution of clarityVVS2
clarityVVS2
Post. distribution of log_depth
log_depth
Post. distribution of log_table
log_table
Post. distribution of log_x
log_x
Post. distribution of log_y.m
log_y.m
Post. distribution of log_z
log_z
Post. distribution of log_carat
log_carat

Posterior Predictive Model Checking

```
mu3 <- model3_run$BUGSoutput$sims.list$mu
n03 <- model3_run$BUGSoutput$sims.list$n0
yrep3 <- t(sapply(1:100, function(i) rgamma(n, mu3[i,]*n03[i], n03[i])))

ppc_dens_overlay(y, yrep3) + ggtitle("Bayesian Gamma Model ") + xlim(0, 10^4)
```

```
## Warning: Removed 30652 rows containing non-finite values (stat_density).
```

```
## Warning: Removed 309 rows containing non-finite values (stat_density).
```

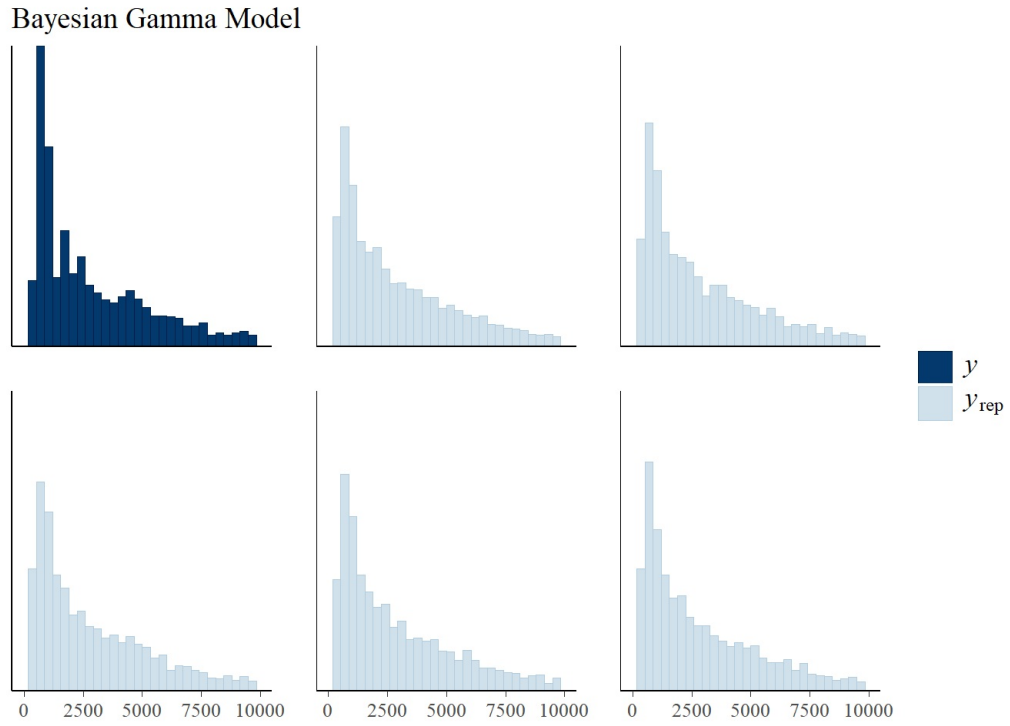



```
ppc_hist(y, yrep3[1:5,]) + ggtitle("Bayesian Gamma Model ") + xlim(0, 10^4)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

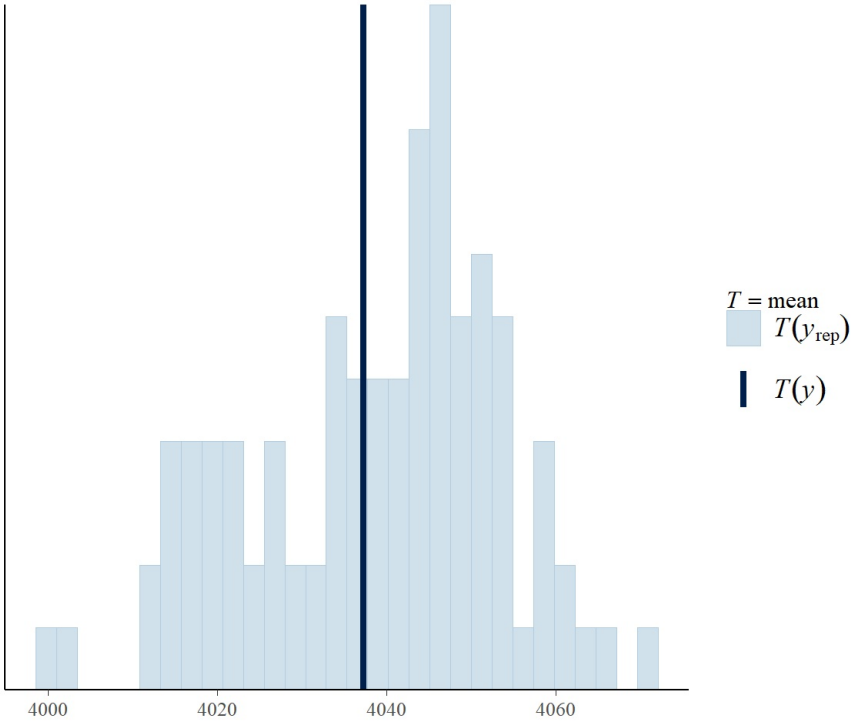
```
## Warning: Removed 1849 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 12 rows containing missing values (geom_bar).
```



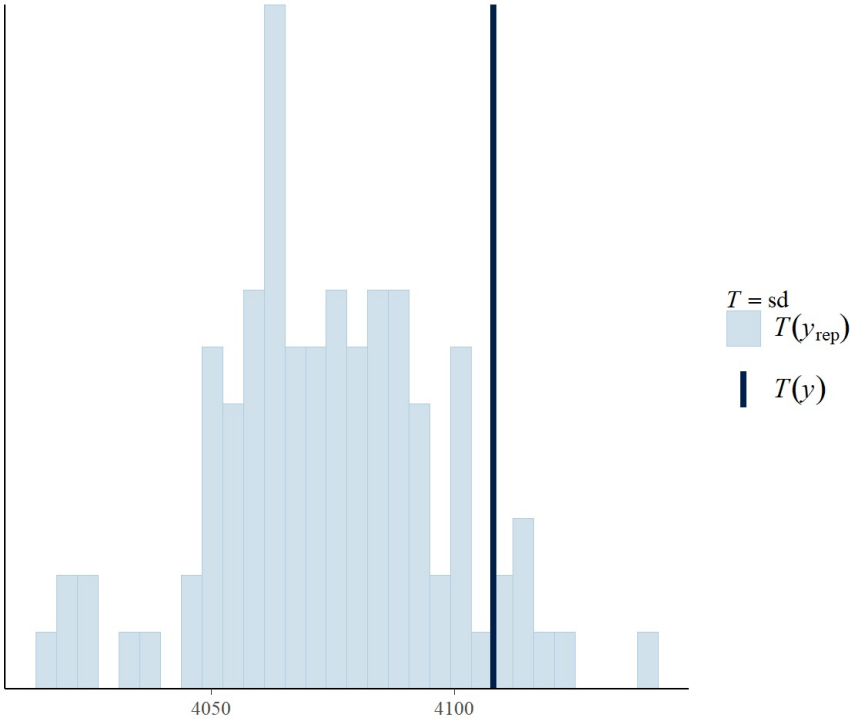

```
ppc_stat(y, yrep3, stat = mean) + theme(aspect.ratio = 1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ppc_stat(y, yrep3, stat = sd) + theme(aspect.ratio = 1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Variable Selection

We implement the Spike and Slap method assigning the following distributions:

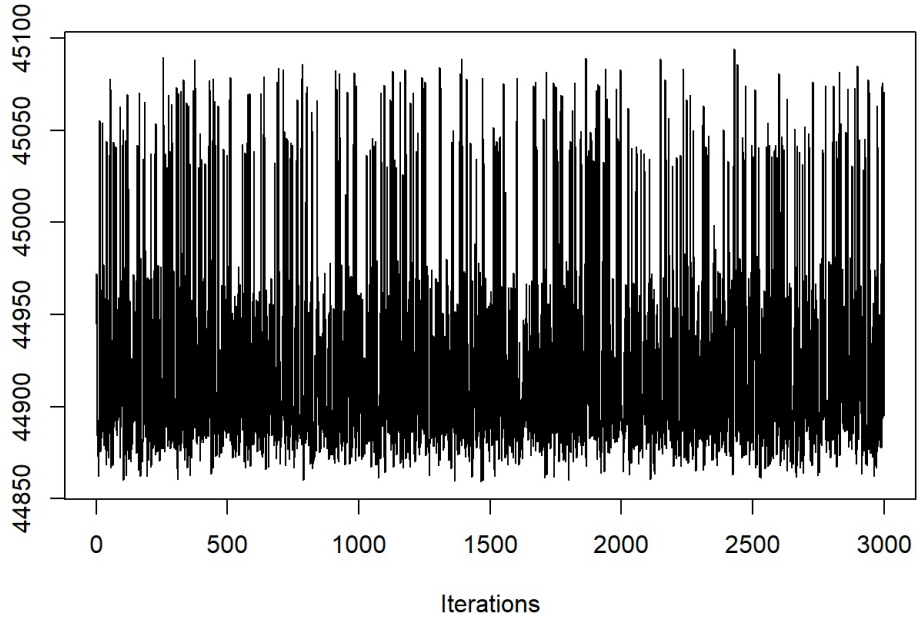
- $\gamma_j \sim \text{Ber}(w)$
- The joint Spike and Slab: $p(\beta_j, \gamma_j) = (1 - w)\delta_0 + w\, dN(\beta_j \mid 0, 0.01)$
- Prior: $w \sim \text{Beta}(1, 1)$

```
model4_code <- function() {
  ## Likelihood
  for (i in 1:n) {
    y[i] ~ dgamma(mu[i]*n0, n0)
    log(mu[i]) <- (phi*b)%*%X[i,]
  }
  #Model
  for(j in 1:p){
  phi[j] ~ dbern(w)
  }
  ## Priors
  for (j in 1:p) {
    b[j] ~ dnorm(0, 0.01)
  }
  n0 ~ dgamma(0.1, 0.1)
  w ~ dbeta(1, 1)
}

model4_data <- list(y = y, X = X3_train, n = n, p = p)
model4_params <- c("b", "mu", "n0", "w", "phi")

model4_run <- jags(
  data = model4_data,
  parameters.to.save = model4_params,
  model.file = model4_code,
  n.chains = 1,
  n.iter = 5000,
  n.burnin = 2000,
  n.thin = 1,
  jags.seed = 101
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 3000
##    Unobserved stochastic nodes: 50
##    Total graph size: 87059
##
## Initializing model
```

MCMC Traceplot

```
traceplot(mcmc(model4_run$BUGSoutput$sims.list$deviance))
```



Now we can use the frequency of $y_{(s)j} = 1$ to compute the probability of inclusion of the j^{th} predictor: $\hat{p}_j = {}^{1S}\sum_{Ss=1} = \phi_{(s)j}$

This represents the percentage of times across the S models that a predictor is included in the model.

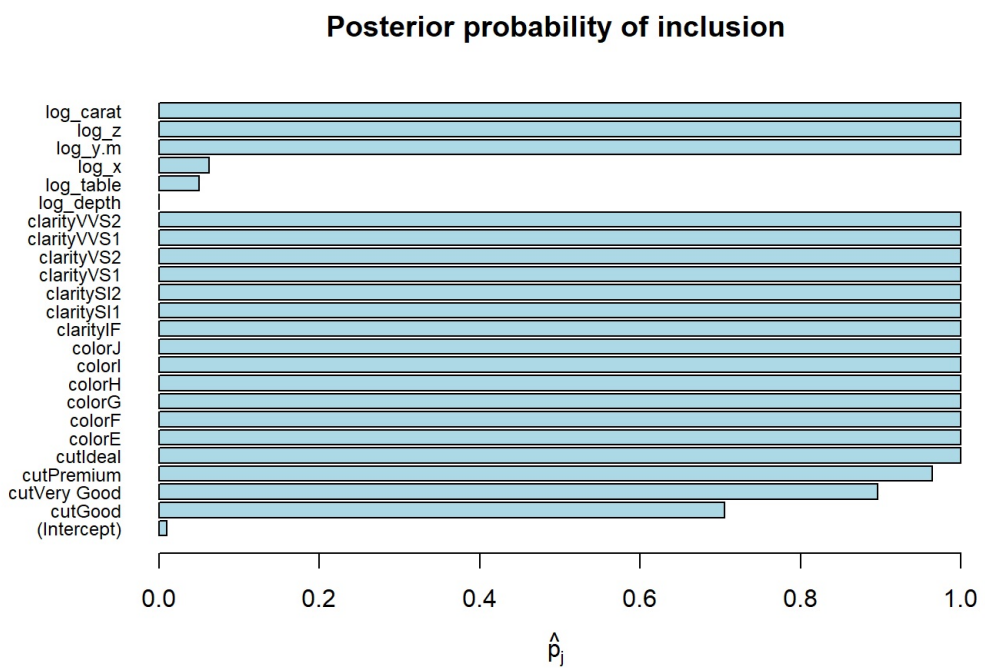
```
out <- model4_run$BUGSoutput
prob.inclusion <- colMeans(out$sims.list$phi)
names(prob.inclusion) <- rownames(res)[1:24]
prob.inclusion <- (as.matrix(prob.inclusion))
colnames(prob.inclusion) <- "Probability of inclusion"
knitr::kable(prob.inclusion)
```

	Probability of inclusion
(Intercept)	0.0093333
cutGood	0.7053333
cutVery Good	0.8960000
cutPremium	0.9643333
cutIdeal	1.0000000
colorE	1.0000000
colorF	1.0000000
colorG	1.0000000
colorH	1.0000000
colorI	1.0000000
colorJ	1.0000000
clarityIF	1.0000000
claritySI1	1.0000000

claritySI2	1.0000000
clarityVS1	1.0000000
clarityVS2	1.0000000
clarityVVS1	1.0000000
clarityVVS2	1.0000000
log_depth	0.0000000
log_table	0.0500000
log_x	0.0620000
log_y.m	1.0000000
log_z	1.0000000
log_carat	1.0000000

Bayesian Gamma Regression with Variable Selection

```
par(mfrow = c(1,1), mar=c(5.1, 5.5, 4.1, 2.1))
prob.inclusion <- colMeans(out$sims.list$phi)
names(prob.inclusion) <- rownames(res)[1:24]
barplot(prob.inclusion, col = "lightblue",
xlab = expression(hat(p)[j]),
cex.names=0.75,las=1, horiz=T, main="Posterior probability of inclusion")
```




```
model_phi<- model4_run$BUGSoutput$sims.list$phi
colnames(model_phi) <- c(colnames(X3_train))

model_MPM <- as.numeric(colMeans(model_phi) >= 0.5)
names(model_MPM) <- c(colnames(X3_train))
knitr::kable(model_MPM)
```

	x
(Intercept)	0
cutGood	1
cutVery Good	1
cutPremium	1
cutIdeal	1
colorE	1
colorF	1

colorG	1
colorH	1
colorI	1
colorJ	1
clarityIF	1
claritySI1	1
claritySI2	1
clarityVS1	1
clarityVS2	1
clarityVVS1	1
clarityVVS2	1
log_depth	0
log_table	0
log_x	0
log_y.m	1
log_z	1
log_carat	1

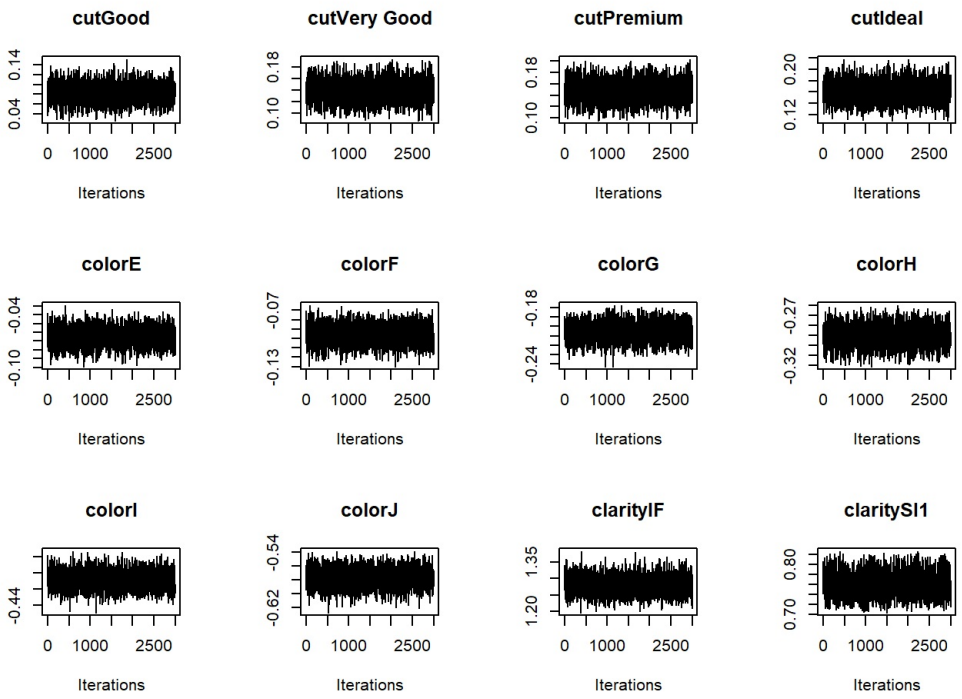
```
X_MPM <- X3_train[,as.logical(model_MPM)]


MPM_data <- list(y = y, X = X_MPM, n = n, p = ncol(X_MPM))

MPM_run <- jags(
  data = MPM_data,
  parameters.to.save = model3_params,
  model.file = model3_code,
  n.chains = 1,
  n.iter = 5000,
  n.burnin = 2000,
  n.thin = 1,
  jags.seed = 101
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 3000
##    Unobserved stochastic nodes: 21
##    Total graph size: 75012
##
## Initializing model
```

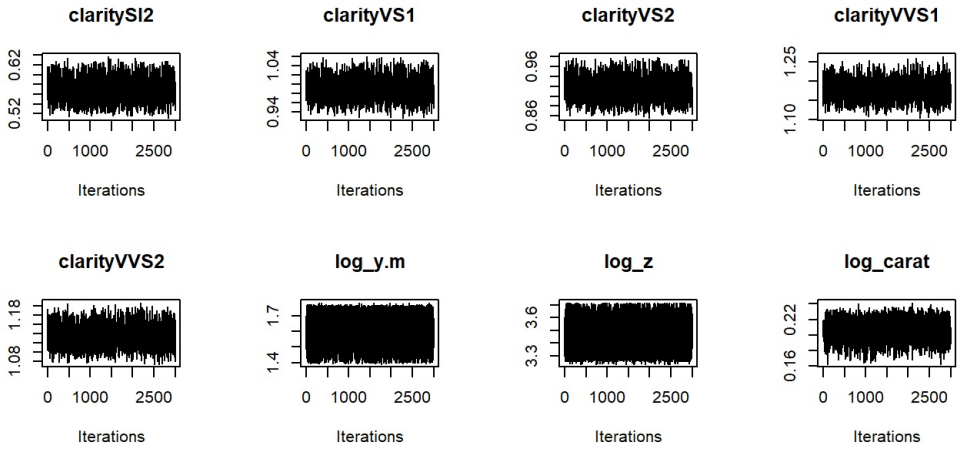
Diagnostic

MCMC Trace-plot

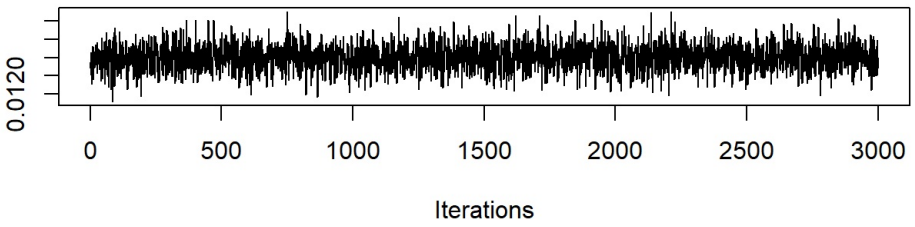
```
p2 = ncol(X_MPM)

par(mfrow=c(3, 4))
for(i in 1:p2){
coda::traceplot(mcmc(MPM_run$BUGSoutput$sims.list$b[,i]), main=colnames(X_MPM)[i])
}
```

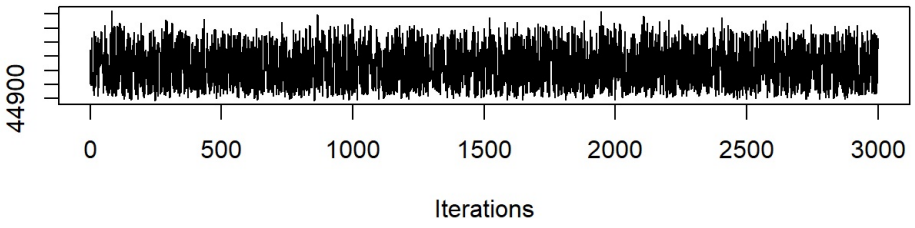



```
par(mfrow=c(2,1))
```



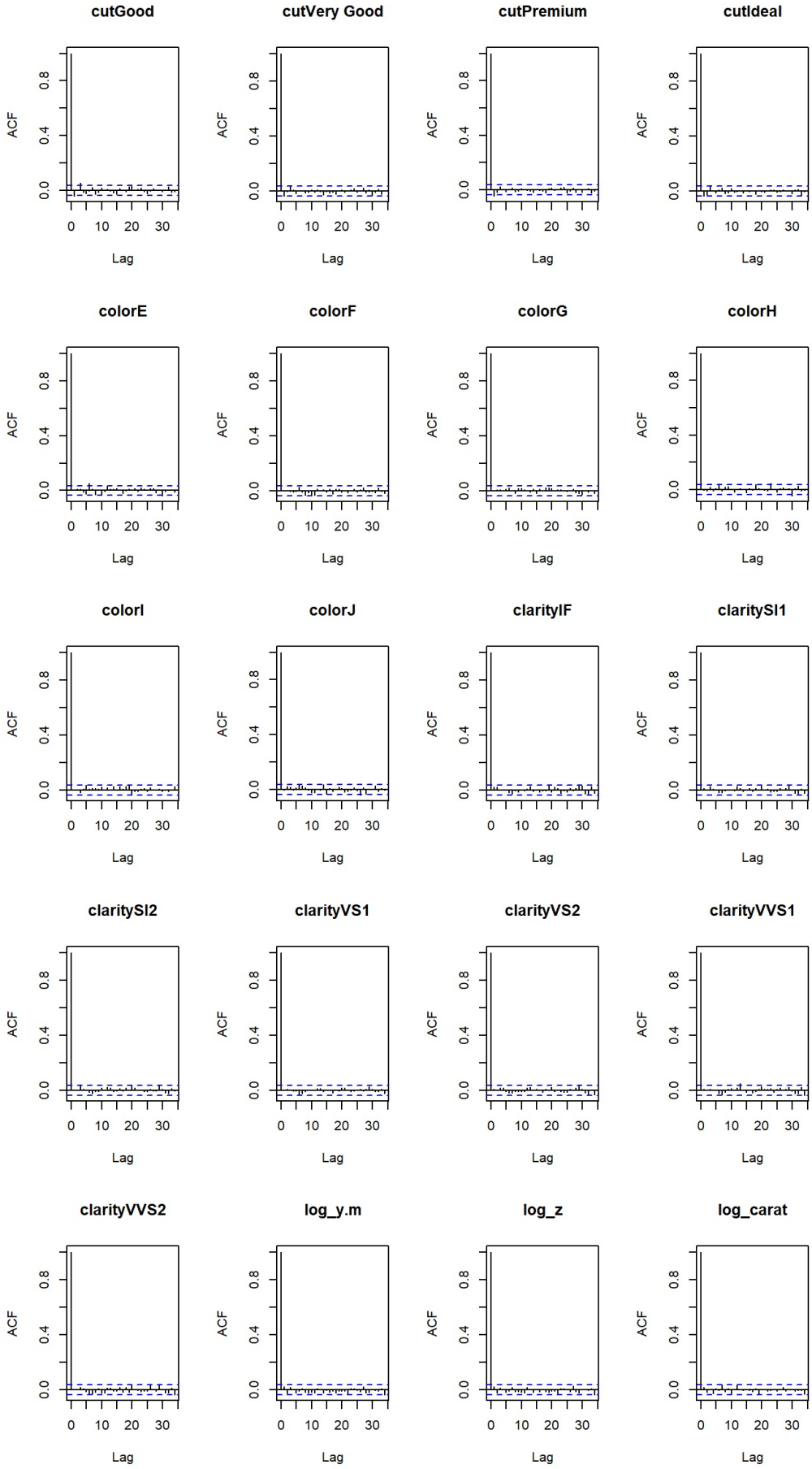

```
traceplot(mcmc(MPM_run$BUGSoutput$sims.list$n0))
traceplot(mcmc(MPM_run$BUGSoutput$sims.list$deviance))
```



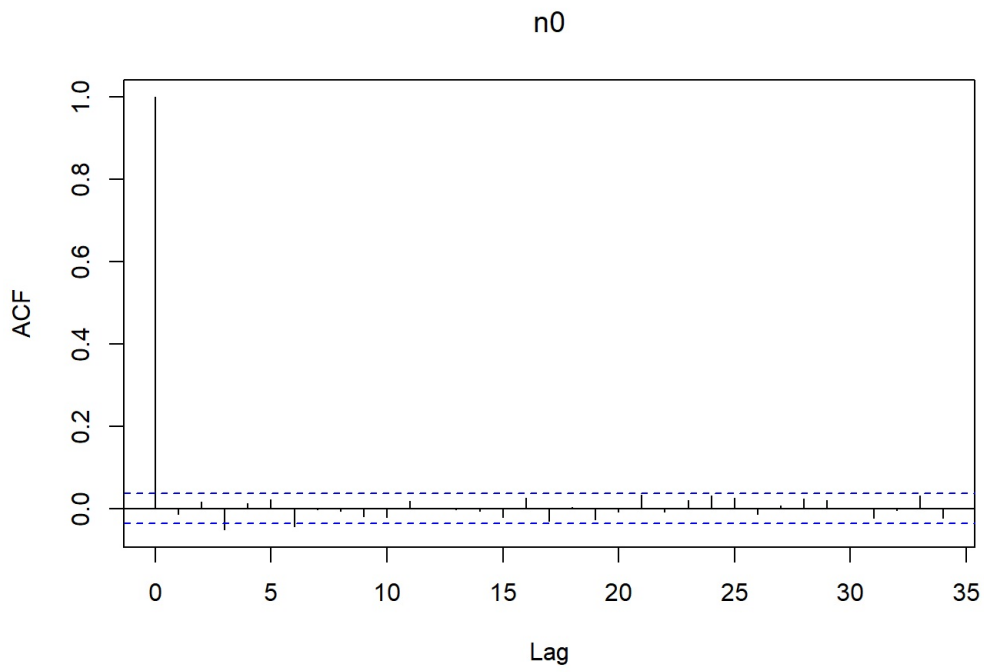


Auto-Correlation Plot

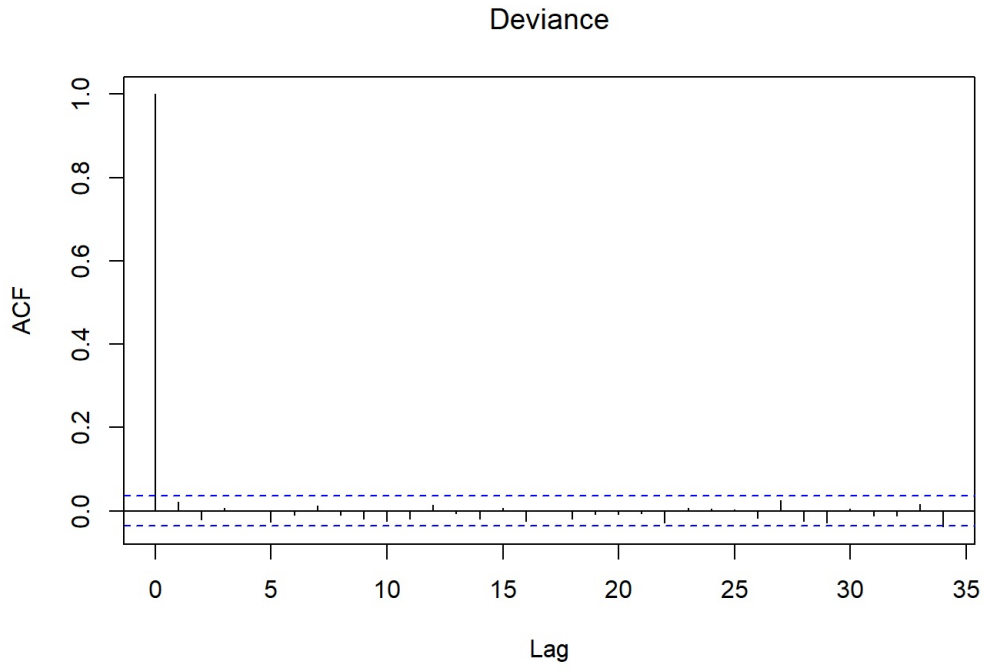
```
par(mfrow=c(2,4))
for(i in 1:p2){
acf(MPM_run$BUGSoutput$sims.list$b[,i], main=colnames(X_MPM)[i])
}
```


cutGood
cutVery Good
cutPremium
cutIdeal
colorE
colorF
colorG
colorH
colorI
colorJ
clarityIF
claritySI1
claritySI2
clarityVS1
clarityVS2
clarityVVS1
clarityVVS2
log_y.m
log_z
log_carat
ACF
Lag
0.0
0.4
0.8
0
10
20
30

```
acf(MPM_run$BUGSoutput$sims.list$n0, main=expression(n0))
```




```
acf(MPM_run$BUGSoutput$sims.list$deviance, main=expression(Deviance))
```



Effective Sample Size

```
for(i in 1:17){
  cat(colnames(X_MPM)[i],": ",coda::effectiveSize(mcmc(MPM_run$BUGSoutput$sims.list$b[,i])), sep="", end="\n")
}
```

```
## cutGood: 2927.151
## cutVery Good: 3219.752
## cutPremium: 3284.814
## cutIdeal: 3140.882
## colorE: 3000
## colorF: 3000
## colorG: 3000
## colorH: 3000
## colorI: 3000
## colorJ: 3000
## clarityIF: 3000
## claritySI1: 3000
## claritySI2: 3000
## clarityVS1: 3000
## clarityVS2: 3000
## clarityVVS1: 3000
## clarityVVS2: 3000
```

```
cat(colnames(X_train)[i],": ",coda::effectiveSize(mcmc(MPM_run$BUGSoutput$sims.list$n0)))
```

```
## clarityVS2 :  3387.598
```

Geweke Test

```
for(i in 1:17){
cat('Geweke test for ',colnames(X_MPM)[i],':',coda::geweke.diag(
MPM_run$BUGSoutput$sims.list$b[,i])$z,end="\n")
}
```

```
## Geweke test for  cutGood : 1.018075
## Geweke test for  cutVery Good : 0.6474782
## Geweke test for  cutPremium : 0.538669
## Geweke test for  cutIdeal : 0.7565693
## Geweke test for  colorE : 0.5867955
## Geweke test for  colorF : -0.2376289
## Geweke test for  colorG : -0.4500884
## Geweke test for  colorH : -0.06303141
## Geweke test for  colorI : 0.3140998
## Geweke test for  colorJ : 1.523239
## Geweke test for  clarityIF : 0.4730794
## Geweke test for  claritySI1 : 1.403256
## Geweke test for  claritySI2 : 1.633343
## Geweke test for  clarityVS1 : 1.283608
## Geweke test for  clarityVS2 : 1.28209
## Geweke test for  clarityVVS1 : 1.591537
## Geweke test for  clarityVVS2 : 0.9617801
```

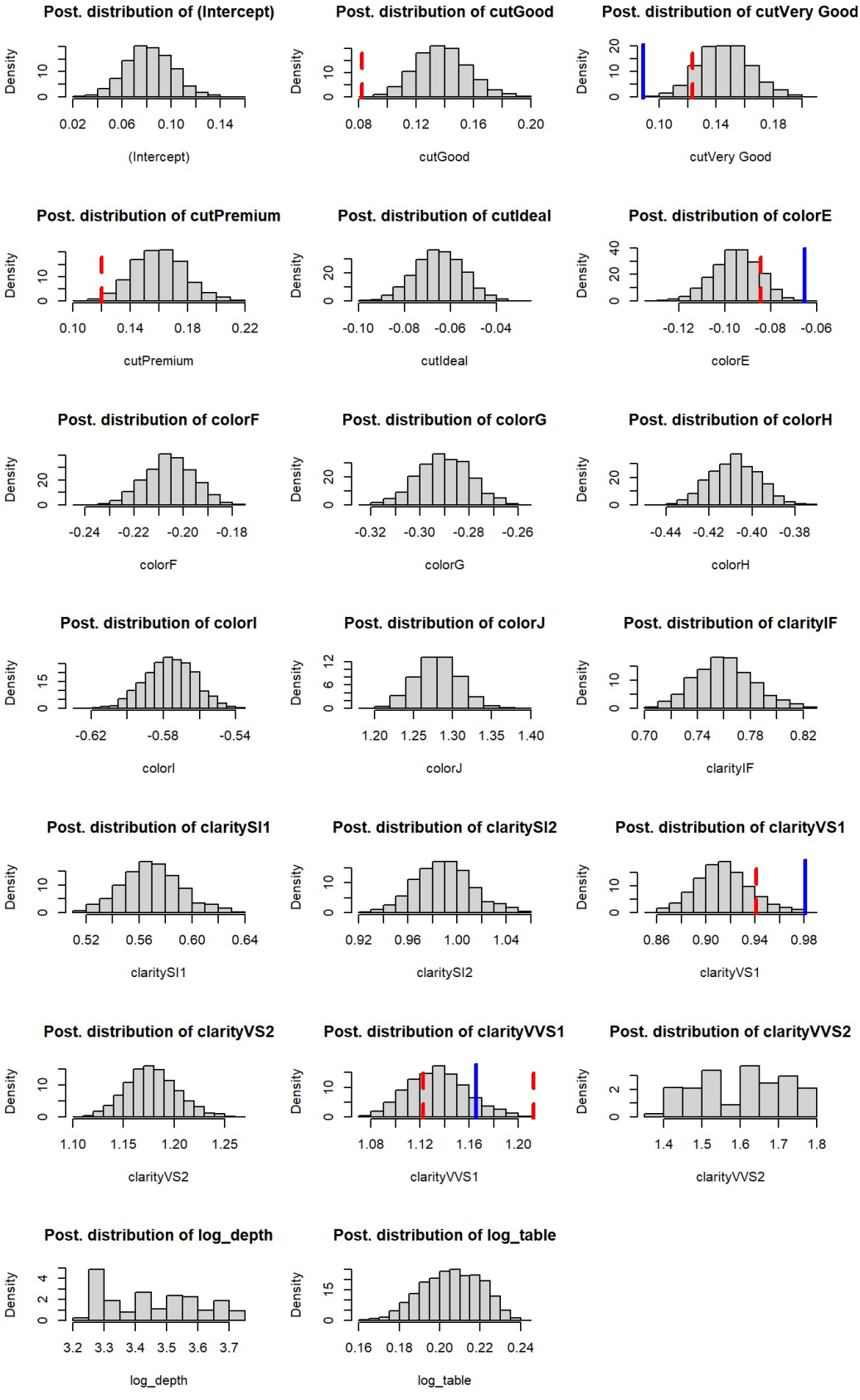
```
cat('Geweke test for n0:',coda::geweke.diag(
MPM_run$BUGSoutput$sims.list$n0)$z)
```

```
## Geweke test for n0: -1.86424
```

Parameters Analysis

Significant Variables and Credible Intervals

```
par(mfrow=c(3,3))
for(i in 1:p2){
hist(MPM_run$BUGSoutput$sims.list$b[,i],
main=paste("Post. distribution of", rownames(res)[i]),
xlab = rownames(res)[i],cex=0.8, prob = TRUE)
abline(v=res[i,1], col="red",lty=2, lwd=3)
abline(v=res[i,2], col="red",lty=2, lwd=3)
abline(v=resu[i,1], col="blue", lwd=3)
}
```

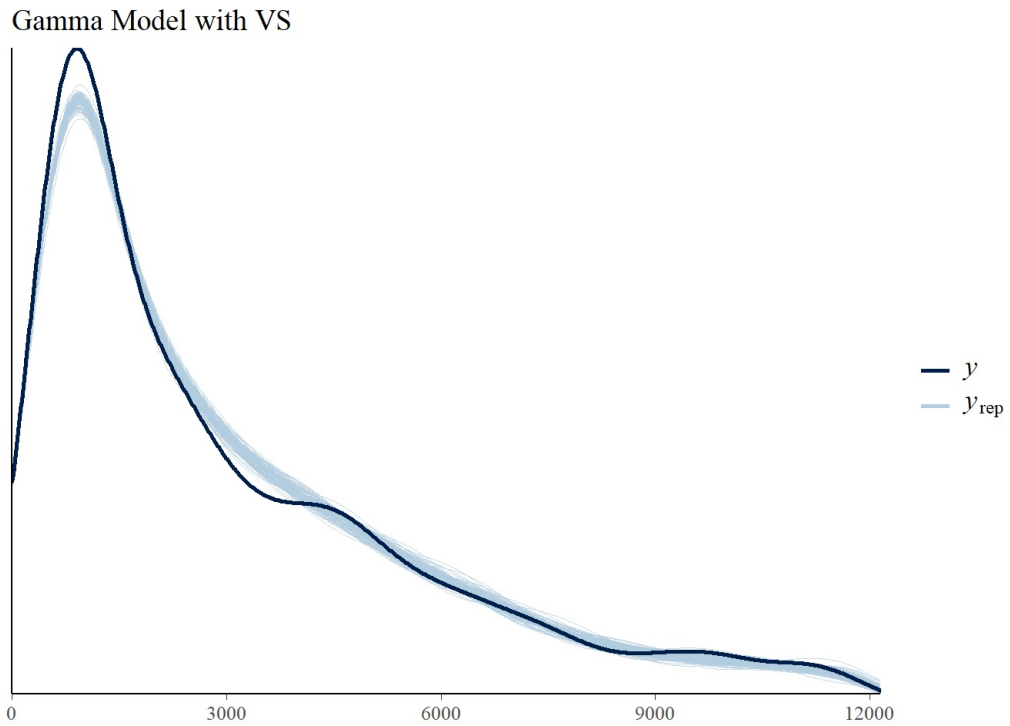

Post. distribution of (Intercept)
Density
(Intercept)
Post. distribution of cutGood
cutGood
Post. distribution of cutVery Good
cutVery Good
Post. distribution of cutPremium
cutPremium
Post. distribution of cutIdeal
cutIdeal
Post. distribution of colorE
colorE
Post. distribution of colorF
colorF
Post. distribution of colorG
colorG
Post. distribution of colorH
colorH
Post. distribution of colorI
colorI
Post. distribution of colorJ
colorJ
Post. distribution of clarityIF
clarityIF
Post. distribution of claritySI1
claritySI1
Post. distribution of claritySI2
claritySI2
Post. distribution of clarityVS1
clarityVS1
Post. distribution of clarityVS2
clarityVS2
Post. distribution of clarityVVS1
clarityVVS1
Post. distribution of clarityVVS2
clarityVVS2
Post. distribution of log_depth
log_depth
Post. distribution of log_table
log_table

Posteriori Predictive Model Checking

```
mu4 <- MPM_run$BUGSoutput$sims.list$mu
n04 <- MPM_run$BUGSoutput$sims.list$n0
yrep4 <- t(sapply(1:100, function(i) rgamma(n, mu4[i,]*n04[i], n04[i])))

ppc_dens_overlay(y, yrep4) + ggtitle("Gamma Model with VS ") + xlim(0, 10.5^4)
```

```
## Warning: Removed 19981 rows containing non-finite values (stat_density).
```

```
## Warning: Removed 203 rows containing non-finite values (stat_density).
```

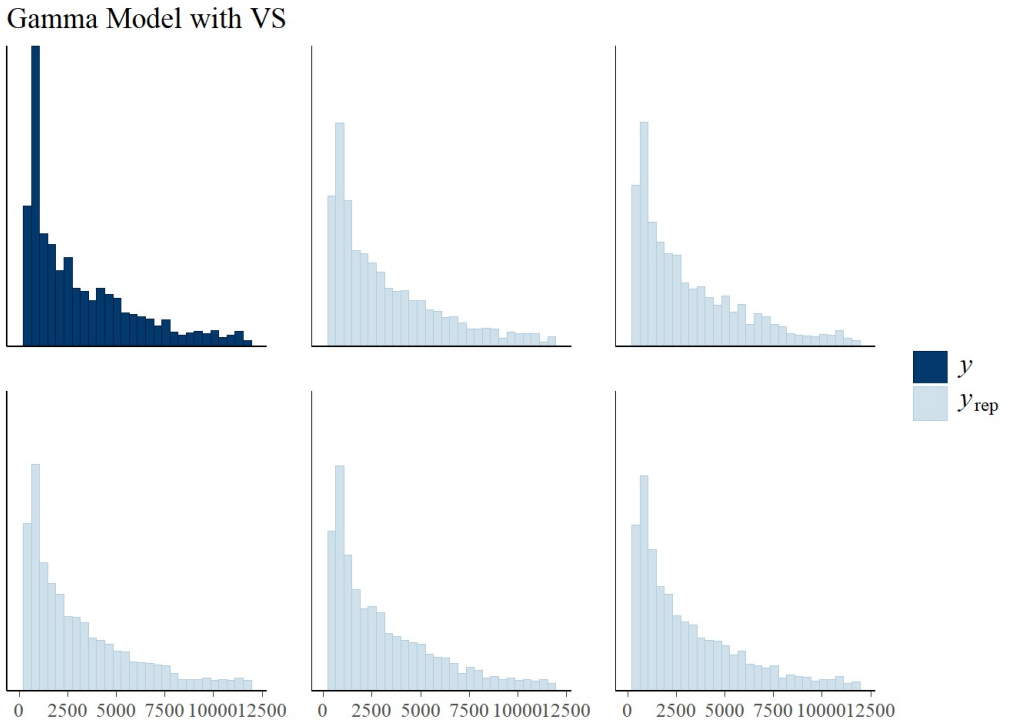



```
ppc_hist(y, yrep4[1:5,]) + ggtitle("Gamma Model with VS") + xlim(0, 10.5^4)
```

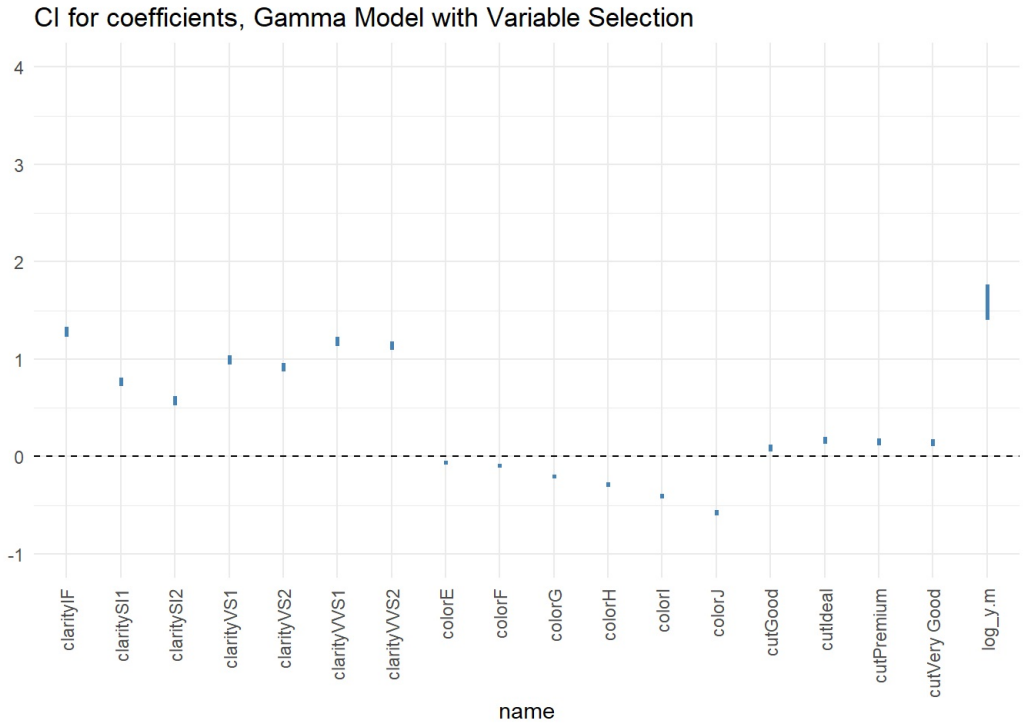
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1194 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 12 rows containing missing values (geom_bar).
```




```
b4_df <- as.data.frame(MPM_run$BUGSoutput$sims.list$b)
names(b4_df) <- colnames(X_MPM)
b4_long <- pivot_longer(b4_df, cols = 1:18) %>%
  group_by(name) %>%
  summarise(q025 = quantile(value, 0.025), q975 = quantile(value, 0.975)) %>%
  ungroup()

ggplot(b4_long, aes(x = name, ymin = q025, ymax = q975)) +
  geom_linerange(size = 1, color = "steelblue") + ylim(-1, 4) +
  geom_hline(yintercept = 0, linetype = 2) + theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  ggtitle("CI for coefficients, Gamma Model with Variable Selection ")
```



Compare models using DIC and MSE

Deviance Information Criterion (DIC)

Deviance can be defined:

$D(\theta) = -2\log p(Y \mid \theta) + C$

where:

$p(Y \mid \theta)$is the likelihood of the data;

C is a constant.

This is basically the likelihood, so model with high likelihood will have a better fit, if we take the log we have a measure we should minimize and moreover it is very naturally to compute. This at each iteration of our model.

$\bar{D}(\theta) = \frac{1}{G} \sum_{g=1} D(\theta_g)$

This can be a good measure of fitting.

An information criterion, such the AIC and BIC, as a measure of model complexity:

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

where $E(\hat{\theta} \mid Y_{obs})$

The p_D can be written as:

$\frac{1}{2} \hat{Var}(D(\theta)) = \frac{1}{2(G-1)} \sum_{g=1} (D(\theta_g) - \bar{D}(\theta))^2$

This is a measure of how the deviance change at each iteration with the respect of the mean.

This is affected by the number of parameters.

At the end the DIC is:

$DIC = \bar{D} + p_D$

```
knitr::kable(cbind(model1_run$BUGSoutput$DIC,model2_run$BUGSoutput$DIC, model3_run$BUGSoutput$DIC, MPM_run$BUGSou
tput$DIC), col.names = c("Normal Model", "t-student model","Gamma Model", "Gamma Model with VS"), align = "ccc")
```

Normal Model	t-student model	Gamma Model	Gamma Model with VS
58446.36	58082.73	44784.31	45390.57

Mean Square Error (MSE)

Defined as:

$E[(Y - \hat{Y})^2]$

This can be defined as:

$\frac{1}{n} \sum_{i=1} (Y_i - \hat{Y})^2$

Y_i are from the test-set.

We want to minimize this loss-function.

$\hat{Y} = E[Y^* \mid Y_{1:n}]$

```
# For Normal Model and t-student
y_test <- test$price

X_test <- test[, c(2, 3, 4, 5, 6, 7, 9, 10, 11)]

X_test <- model.matrix(y_test~. , data=X_test)


# For Gamma Model
X3_test <- test %>%
  mutate(across(c(6:11), log, .names = "log_{.col}"), .keep = "unused") %>%
  mutate(across(c(1:2), log, .names = "log_{.col}"), .keep = "unused")
X3_test <- X3_test[c(1:9,11)]
X3_test <- model.matrix(log_price ~ ., data = X3_test)

## For Gamma Model with Variable Selection

X4_test <- X3_test[,as.logical(model_MPM)]

mu1_pred <- as.matrix(b1_df)%*%t(X_test)
sd1_pred <- sqrt(1/tau1)
ypred1 <- sapply(1:3000, function(i) rnorm(n, mu1_pred[i,], sd1_pred[i]))
ypred1_means <- apply(ypred1, 1, mean)

MSE1 <- mean((y_test - ypred1_means)^2)
RMSE1 <- sqrt(MSE1)

mu2_pred <- as.matrix(b2_df)%*%t(X_test)
sd2_pred <- sqrt(1/tau2)
k2_pred <- k2
ypred2 <- sapply(1:3000, function(i) mu2_pred[i,] + sd2_pred[i]*rt(n, df = k2[i]))
ypred2_means <- apply(ypred2, 1, mean)

MSE2 <- mean((y_test - ypred2_means)^2)
RMSE2 <- sqrt(MSE2)


mu3_pred <- exp(as.matrix(b3_df)%*%t(X3_test))
n03_pred <- n03
ypred3 <- sapply(1:3000, function(i) rgamma(n, mu3_pred[i,]*n03[i], n03[i]))
ypred3_means <- apply(ypred3, 1, mean)

MSE3 <- mean((y_test - ypred3_means)^2)
RMSE3 <- sqrt(MSE3)

mu4_pred <- exp(as.matrix(b4_df)%*%t(X4_test))
n04_pred <- n04
ypred4 <- sapply(1:3000, function(i) rgamma(n, mu4_pred[i,]*n04[i], n04[i]))
ypred4_means <- apply(ypred4, 1, mean)

MSE4 <- mean((y_test - ypred4_means)^2)
RMSE4 <- sqrt(MSE4)

knitr::kable(t(c(RMSE1, RMSE2, RMSE3, RMSE4)),align = "cccc", "pipe", col.names = c("B. Linear Model","B. t-stude
nt model","B. Gamma model","B. Gamma model with VS"))
```

B. Linear Model	B. t-student model	B. Gamma model	B. Gamma model with VS
3766.742	4108.183	655.2346	704.4954

Prior Sensitivity Analysis

In this last paragraph we quickly go through the results we obtained by modifying the priors of our parameters in order to assess the sensitivity of our analysis, which means that we tested how our outputs and our estimates changed with the change in the parameters.

Due to the number of observations that we consider for the train set, a change of the hyper-parameters of the priors for each model would not change the credible intervals of the coefficients.

To see changes in the posterior, it would be necessary to implement a very strong prior. One possible prior of this type could be a t-Student with k degrees of freedom that go to infinity.

Conclusion

Finally, after all the anlysis performed we can state that the Bayesian Gamma Model is the most precise among the others models,, in predicting, which was the goal of this analysis.