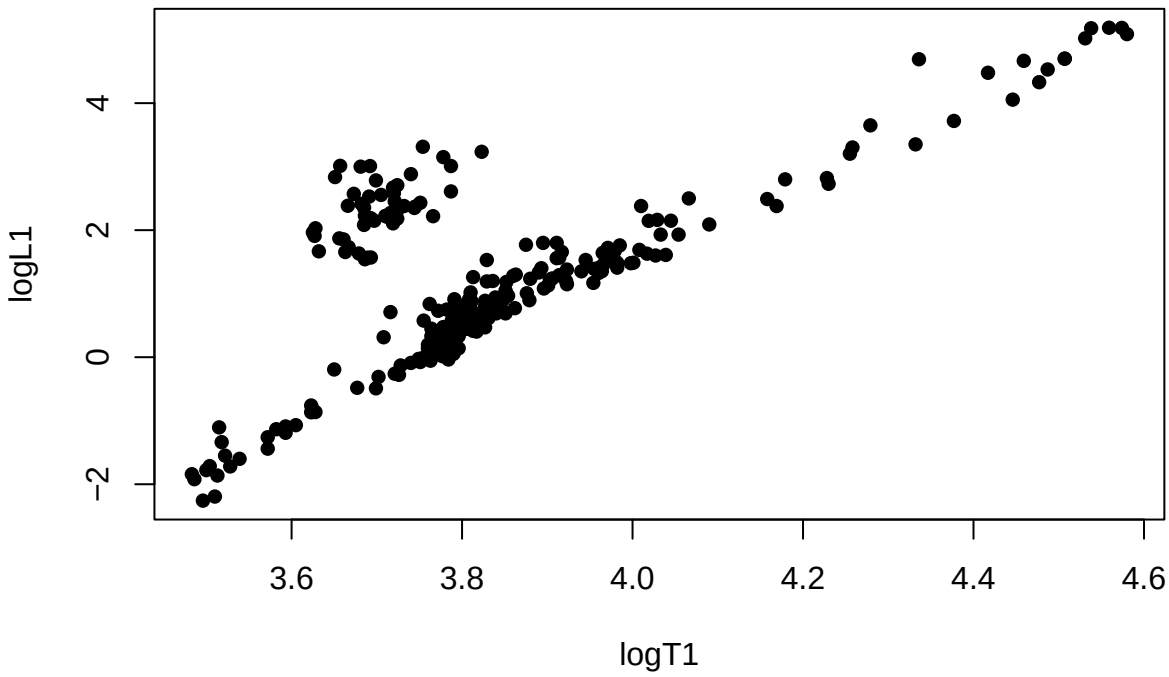
Analysis (with the Red Giants)

Aritra Banerjee

2022-11-17

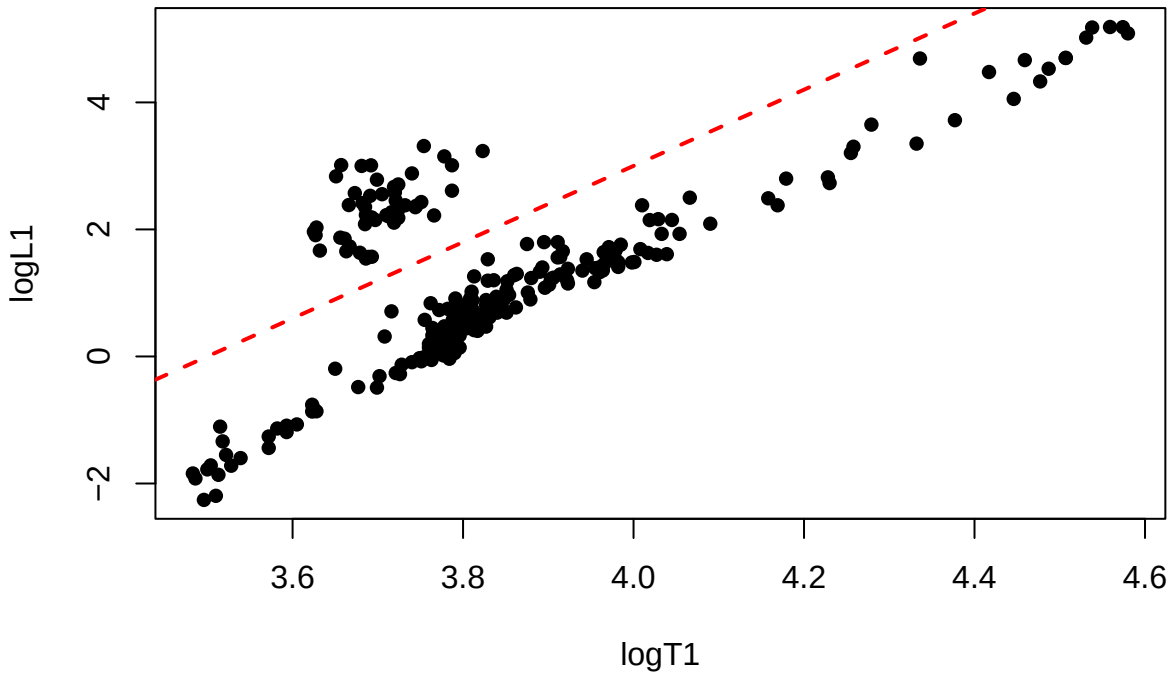
In this project our objective is to build a linear model that explains the association of Luminosity and Temperature of a star. Stefano-Boltzman law says that for perfect radiators we have $L \propto T^4$. Since, radiating surface of a star is a good approximation of black body, we can expect a linear association between $logL$ and $logT$.

```
data <- read.csv('SB_LAW.csv', header = T)

plot(logL1 ~ logT1, data = data, pch = 16)
```



In the plot we can see a cluster of high luminosity points at a low temperature which correspond to the Red Giants in our dataset and these high luminosity data points lie above the $y = 6x - 21$ in the plot

```
plot(logL1 ~ logT1, data = data, pch = 16)
abline(b = 6, a = -21, col = 'red', lty = 2, lwd = 2)
```



To take care of this kind of data it makes sense to consider 2 separate lines for the red giants and the main sequence stars. We can do that by creating a dummy variable for the red-giants in the following manner

$$logL_i = \beta_0 + \beta_1 logT_i + \beta_2 1_{RG} + \epsilon_i$$

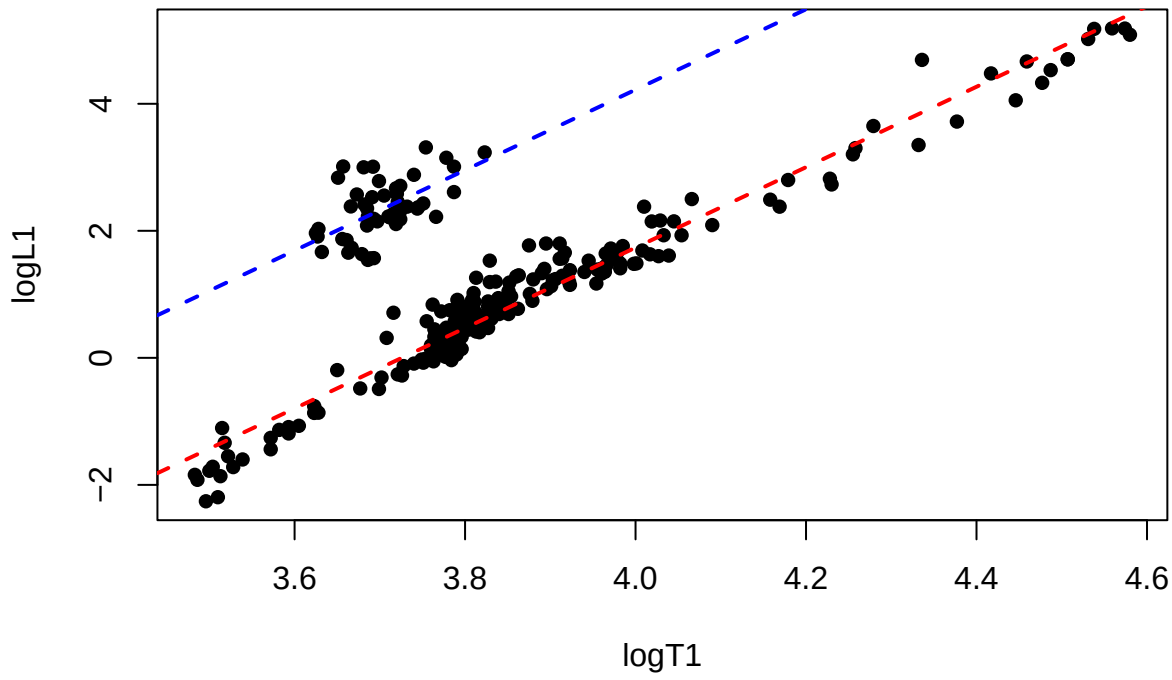
```
data$dummy <- ifelse(data$logL1<6*data$logT1 - 21, 'MC', 'RG')
data$dummy <- as.factor(data$dummy)
```

Fitting a linear model with both the *logT* and *dummy* as predictors we get a model that fits two parallel lines for 2 groups

```
lmod <- lm(logL1 ~ logT1 + dummy, data = data)
b_mod <- coef(lmod)

plot(logL1 ~ logT1, data = data, pch = 16)

abline(a = b_mod[1], b = b_mod[2],
       col = 'red', lwd = 2,
       lty = 2)

abline(a = b_mod[1] + b_mod[3], b = b_mod[2],
```

	Model 1
(Intercept)	−23.586
	(0.375)
logT1	6.330
	(0.097)
dummyRG	2.486
	(0.052)
Num.Obs.	262
R2	0.952
R2 Adj.	0.952
AIC	137.1
BIC	151.4
Log.Lik.	−64.573
RMSE	0.31

```
       col = 'blue',
       lty = 2, lwd = 2)
```



The summary of the model is as below

```
modelsummary(lmod)
```

For the Bayesian analysis on the model, we perform the Linchpin variable sampling to get sample from the

posterior of the regression coefficients. We assume the reference prior for our analysis, i.e.

$$\nu(\beta, \sigma^2) = \frac{1}{\sigma^2}; \ \ \sigma > 0, \beta \in R^3$$

Now, since we are interested in 4 variables in total, we need to know what should be our effective smaple size

```
minESS(p = 4)
```

```
## minESS
##   8431
```

Reference prior:

So, we should make enough number of iterations to get at least an effective sample size of 8000. The linchpin variable sampling is performed using the following R-code (with 10000)

```
sse<-sum(lmod$residuals^2)

lam_shape<-lmod$df/2

mod_mat<-model.matrix(lmod)

XTXinv<-solve(t(mod_mat) %*% mod_mat)

msim<-1e4

ref_post_sample<-matrix(NA_real_, ncol=4, nrow=msim)

colnames(ref_post_sample) <- c(bquote(beta[0]),bquote(beta[1]),
                               bquote(beta[2]), bquote(lambda))

for (iter in 1:msim){
  lam<-rgamma(1, shape=lam_shape, scale = 2/sse)
  cmat<-(1/lam)*XTXinv
  ref_post_sample[iter,]<-c(rmvnorm(1,
                                    mean = lmod$coefficients,
                                    sigma = cmat),lam)
}
```

The effective sample sizes are above are requirement so we are good to go

```
ref_post_sample <- as.mcmc(ref_post_sample)
apply(ref_post_sample, 2, ess)
```

```
## beta[0] beta[1] beta[2]  lambda
##   10000   10000   10000   10000
```

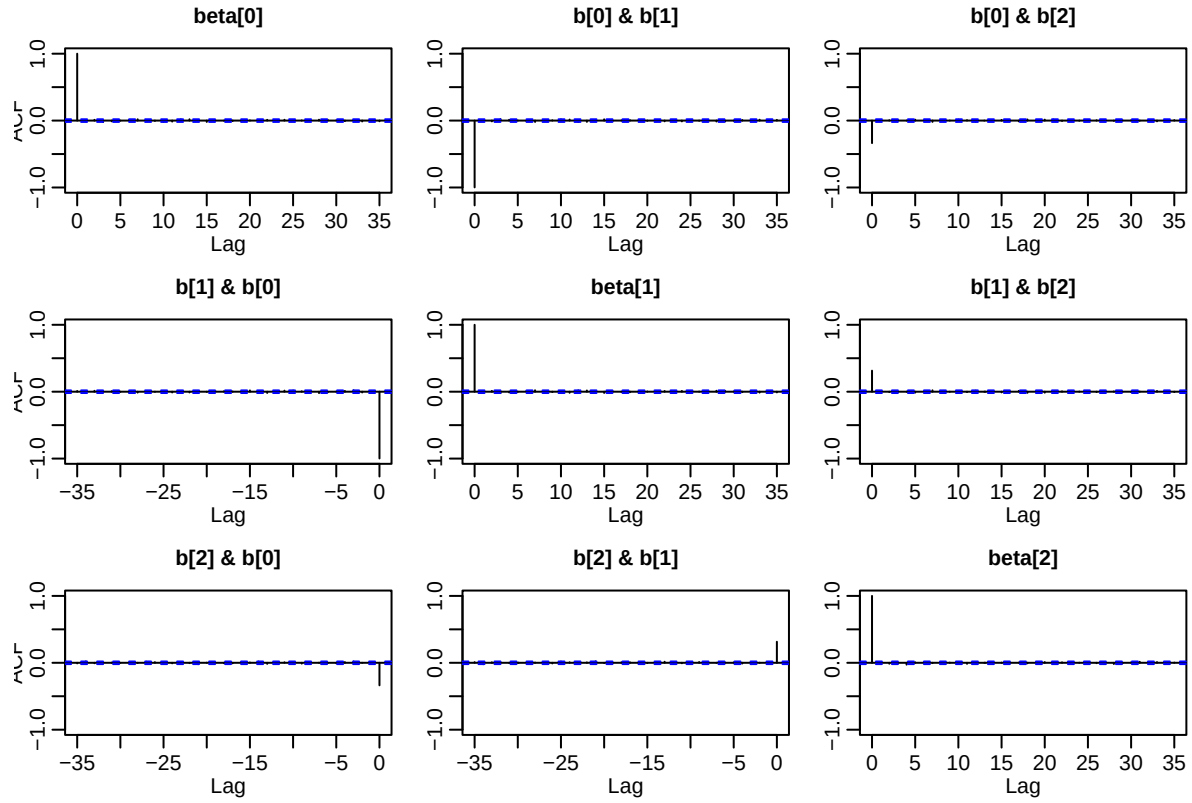
following is brief summary of the samples

```
apply(ref_post_sample, 2, summary)
```

```
##           beta[0]  beta[1]  beta[2]    lambda
## Min.    -24.98649 5.924890 2.292056  7.194565
## 1st Qu. -23.83699 6.265068 2.451604  9.694155
## Median  -23.58096 6.329078 2.486454 10.289185
## Mean    -23.58391 6.329786 2.486144 10.318118
## 3rd Qu. -23.33222 6.395226 2.520827 10.901229
## Max.    -22.01964 6.684160 2.679584 13.720257
```

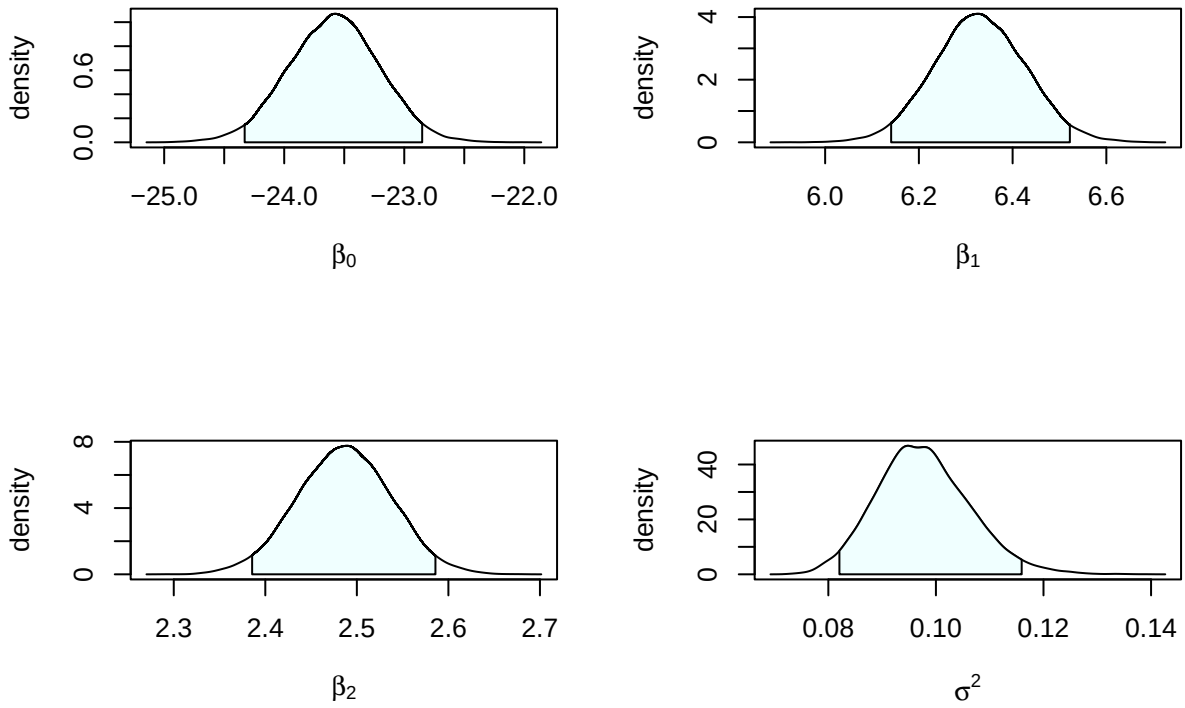
and the ACF's also look good.

```
acf(ref_post_sample[,1:3])
```



The following are the marginal posteriors of the regression parameters

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

hist_ci(ref_post_sample[,1], name = bquote(beta[0]))
hist_ci(ref_post_sample[,2], name = bquote(beta[1]))
hist_ci(ref_post_sample[,3], name = bquote(beta[2]))
hist_ci(1/ref_post_sample[,4], name = bquote(sigma^2))
```



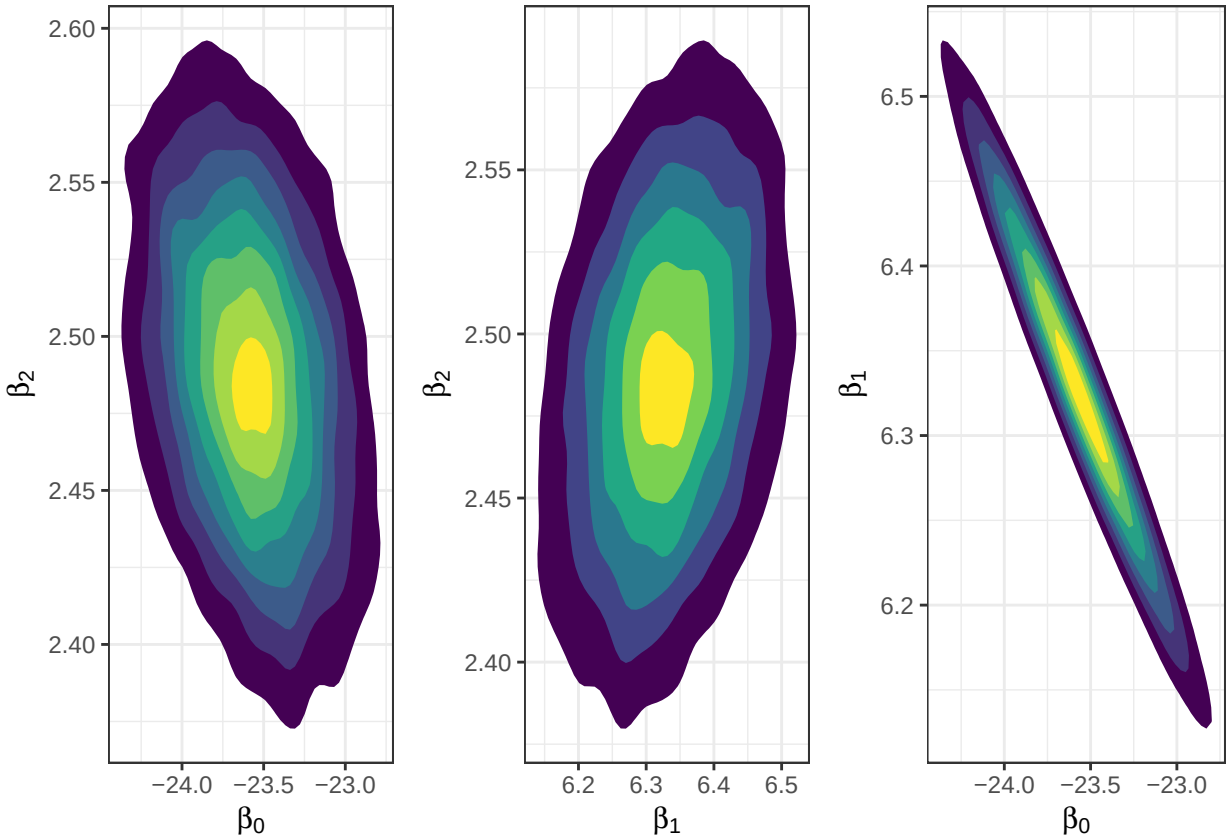
We can also look into the joint distributions of the coefficients of β and the joint distributions of β_0, β_2 and β_1, β_2 look as expected but that of β_0 and β_1 looks very highly correlated

```
b0_post <- ref_post_sample[,1]
b1_post <- ref_post_sample[,2]
b2_post <- ref_post_sample[,3]

df <- data.frame(b0 = as.vector(b0_post),
                 b1 = as.vector(b1_post),
                 b2 = as.vector(b2_post))
```

```
f02 <- ggplot(data = df, aes(x = b0, y=b2) ) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon") +
  scale_fill_continuous(type = "viridis") +
  theme_bw() +
  xlab(bquote(beta[0])) +
  ylab(bquote(beta[2])) +
  theme(
    legend.position='none'
  )

f12 <- ggplot(data = df, aes(x = b1, y=b2) ) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon") +
  scale_fill_continuous(type = "viridis") +
  theme_bw() +
  xlab(bquote(beta[1])) +
```

```
  ylab(bquote(beta[2])) +
  theme(
    legend.position='none'
  )

f01 <- ggplot(data = df, aes(x = b0, y=b1) ) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon") +
  scale_fill_continuous(type = "viridis") +
  theme_bw() +
  xlab(bquote(beta[0])) +
  ylab(bquote(beta[1]))+
  theme(
    legend.position='none'
  )

ggpubr::ggarrange(f02, f12, f01, nrow = 1, ncol = 3)
```

```
## Warning: The dot-dot notation (‘..level..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(level)‘ instead.
```



which we suspect is because the determinant of the corresponding dispersion matrix is very close to 0.

```
XTXinv[1:2, 1:2] |> det()
```

```
## [1] 0.0004553431
```

Independence Prior:

Now we shall be considering the independence prior, i.e.

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

$$\frac{1}{\sigma^2} \sim Gamma(c_0/2, d_0/2)$$

for B_0 we assume it's a diagonal matrix, i.e. we assume an independent prior on the β_j's. We assume $B_0^{-1} = diag(1/b_1, 1/b_1, 1/b_1)$. Here, we can use the `MCMCregress` function to sample from the posterior. For a brief sensitivity analysis let us vary the hyperparameters a little and see what impact it has on the posterior distributions

$b_0 = \hat{\beta}_{MLE}, b_1 = 1, c_0 = 1, d_0 = 1$

```
ref_post_sample <- MCMCregress(logL1 ~ logT1 + dummy, data=data,
                               burnin=0, mcmc=1e4, b0= coef(lmod),
                               B0=1, c0=1, d0=1)

apply(ref_post_sample, 2, ess)
```

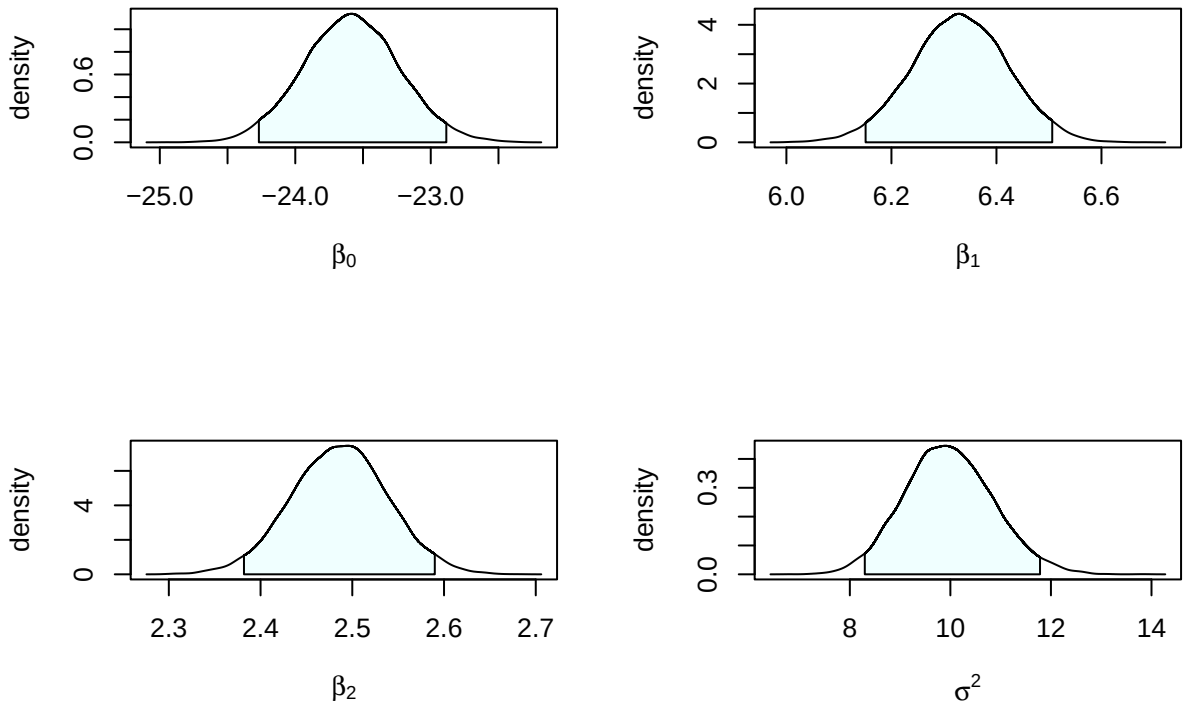
```
## (Intercept)       logT1     dummyRG      sigma2
##       10000       10000       10000       10000
```

```
apply(ref_post_sample, 2, mean)
```

```
## (Intercept)       logT1     dummyRG      sigma2
## -23.5844949   6.3299501   2.4862438   0.1010822
```

Again, we good in effective sample size front. The distributions look like

```
par(mfrow = c(2,2))
hist_ci(ref_post_sample[,1], name = bquote(beta[0]))
hist_ci(ref_post_sample[,2], name = bquote(beta[1]))
hist_ci(ref_post_sample[,3], name = bquote(beta[2]))
hist_ci(1/ref_post_sample[,4], name = bquote(sigma^2))
```



$b_0 = \hat{\beta}_{MLE}, b_1 = 1e-3, c_0 = 1, d_0 = 1$

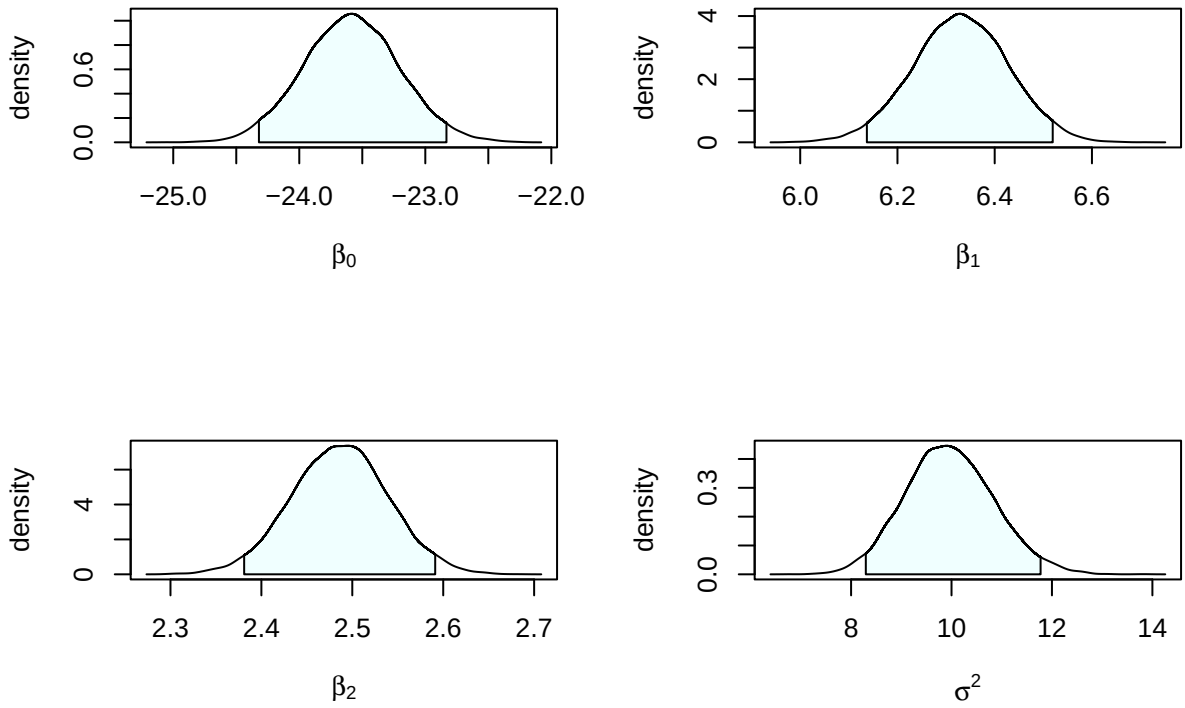
```
ref_post_sample <- MCMCregress(logL1 ~ logT1 + dummy, data=data,
                               burnin=0, mcmc=1e4, b0= coef(lmod),
                               B0=1e-3, c0=1, d0=1)

apply(ref_post_sample, 2, ess)
```

```
## (Intercept)       logT1     dummyRG      sigma2
##       10000       10000       10000       10000
```

```
apply(ref_post_sample, 2, mean)
```

```
## (Intercept)       logT1     dummyRG      sigma2
## -23.5843708   6.3299182   2.4862381   0.1011356
```

```
par(mfrow = c(2,2))
hist_ci(ref_post_sample[,1], name = bquote(beta[0]))
hist_ci(ref_post_sample[,2], name = bquote(beta[1]))
hist_ci(ref_post_sample[,3], name = bquote(beta[2]))
hist_ci(1/ref_post_sample[,4], name = bquote(sigma^2))
```



$b_0 = 0, b_1 = 1e-3, c_0 = 10, d_0 = 10$

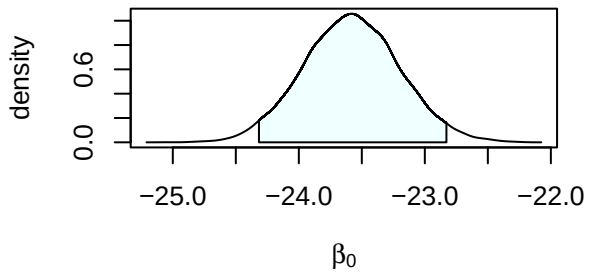
```
ref_post_sample <- MCMCregress(logL1 ~ logT1 + dummy, data=data,
                               burnin=0, mcmc=1e4, b0= c(0,0,0),
                               B0=1e-3, c0=1, d0=1)

apply(ref_post_sample, 2, ess)
```

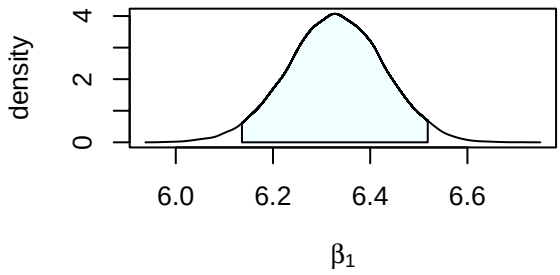
```
## (Intercept)       logT1     dummyRG      sigma2
##       10000       10000       10000       10000
```

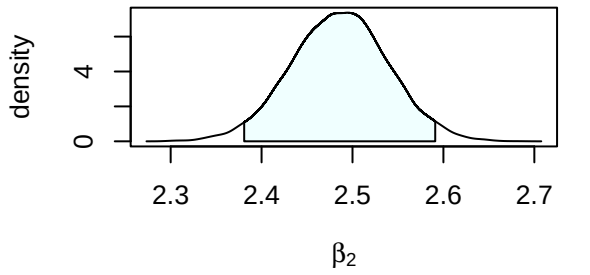
```
apply(ref_post_sample, 2, mean)
```

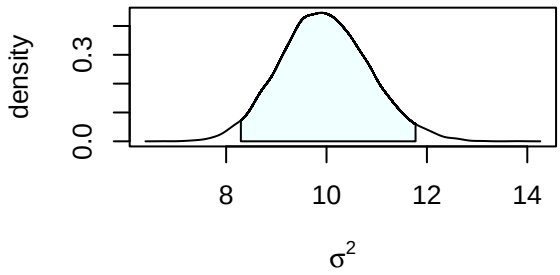
```
## (Intercept)       logT1     dummyRG      sigma2
## -23.5806528   6.3289610   2.4860596   0.1011356
```

```
par(mfrow = c(2,2))
hist_ci(ref_post_sample[,1], name = bquote(beta[0]))
hist_ci(ref_post_sample[,2], name = bquote(beta[1]))
hist_ci(ref_post_sample[,3], name = bquote(beta[2]))
hist_ci(1/ref_post_sample[,4], name = bquote(sigma^2))
```









This pretty much shows that the posterior distribution remains more or less unaffected by the choice of hyper parameters and the both the reference prior and independence prior yield similar outcome. Finally proceeding with the $b_0 = \hat{\beta}_{MLE}, b_1 = 1e-3, c_0 = 1, d_0 = 1$ case if we look into the joint distributions of β coefficients we again see similar plots as before

```
ref_post_sample <- MCMCregress(logL1 ~ logT1 + dummy, data=data,
                               burnin=0, mcmc=2.5e4, b0= coef(lmod),
                               B0=1, c0=1, d0=1)
b0_post <- ref_post_sample[,1]
b1_post <- ref_post_sample[,2]
b2_post <- ref_post_sample[,3]

df <- data.frame(b0 = as.vector(b0_post),
                 b1 = as.vector(b1_post),
                 b2 = as.vector(b2_post))
```

```
f02 <- ggplot(data = df, aes(x = b0, y=b2) ) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon") +
  scale_fill_continuous(type = "viridis") +
  theme_bw() +
  xlab(bquote(beta[0])) +
  ylab(bquote(beta[2])) +
  theme(
    legend.position='none'
  )
```

```
f12 <- ggplot(data = df, aes(x = b1, y=b2) ) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon") +
  scale_fill_continuous(type = "viridis") +
  theme_bw() +
  xlab(bquote(beta[1])) +
  ylab(bquote(beta[2])) +
  theme(
    legend.position='none'
  )

f01 <- ggplot(data = df, aes(x = b0, y=b1) ) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon") +
  scale_fill_continuous(type = "viridis") +
  theme_bw() +
  xlab(bquote(beta[0])) +
  ylab(bquote(beta[1]))+
  theme(
    legend.position='none'
  )

ggpubr::ggarrange(f02, f12, f01, nrow = 1, ncol = 3)
```