Stat4DS2+DS - Final Project

Michele Cernigliaro, n°matricola: 1869097

18/08/2019

Death from liver cirrhosis and specific alcoholic beverage consumption in 1950

In this project we explore the dataset from the article *“Death from liver cirrhosis and specific alcoholic beverage consumption: an ecological study”* written by *Schmidt W., Bronetto J.* and published in 1950. We will perform linear regression models using bayesian approaches, thus we will compare the results with “standard” frequentistic approaches.

---

1. The dataset

The dataset contains liver cirrhosis mortality and specific alcohol consumption rates for the 46 states of the United States where in 1950 all the classes of beverage (wine, beer, spirits) were legally sold. The information are related to the year 1950. We have then a dataframe with 46 observations representing each state and for each one of them the features are the following:

- **State**: The state name
- **Spirits**: Spirits consumption (gallons pro capita)
- **Wine**: Wine consumption (gallons pro capita)
- **Beer**: Beer consumption (gallons pro capita)
- **Total**: Total alcohol consumption - sum of the other three variables Wine, Beer, and Spirits (gallons pro capita)
- **DeathRate**: liver chirrosis death rates (expected deaths per standard million population). The values have been standardized in order to smooth differences in ages and sex composition of the state population.
- **Urbanism**: Index of Urbanism (percentage of population living in urban area)

```
# load dataframe
df <- read.csv('~/Desktop/SDS2_project/drinking_1950.csv', stringsAsFactors = F)
# get a look
head(df, 10)
```

```
##             State Spirits Wine Beer Total DeathRate Urbanism
## 1         Alabama    0.30 0.05 0.17  0.52     41.23     23.6
## 2           Idaho    0.41 0.04 0.58  1.03     31.69     10.3
## 3            Iowa    0.38 0.03 0.59  1.00     39.40     28.4
## 4           Maine    0.48 0.07 0.58  1.13     57.54     16.4
## 5        Michigan    0.53 0.11 0.99  1.63     74.78     48.9
## 6         Montana    0.65 0.09 0.77  1.51     59.85     17.6
## 7   New Hampshire    0.73 0.06 0.79  1.58     54.35     27.3
## 8  North Carolina    0.32 0.03 0.18  0.53     47.86     16.3
## 9            Ohio    0.56 0.12 0.85  1.53     77.23     49.3
## 10         Oregon   57.00 0.07 0.75  1.39     56.61     29.8
```

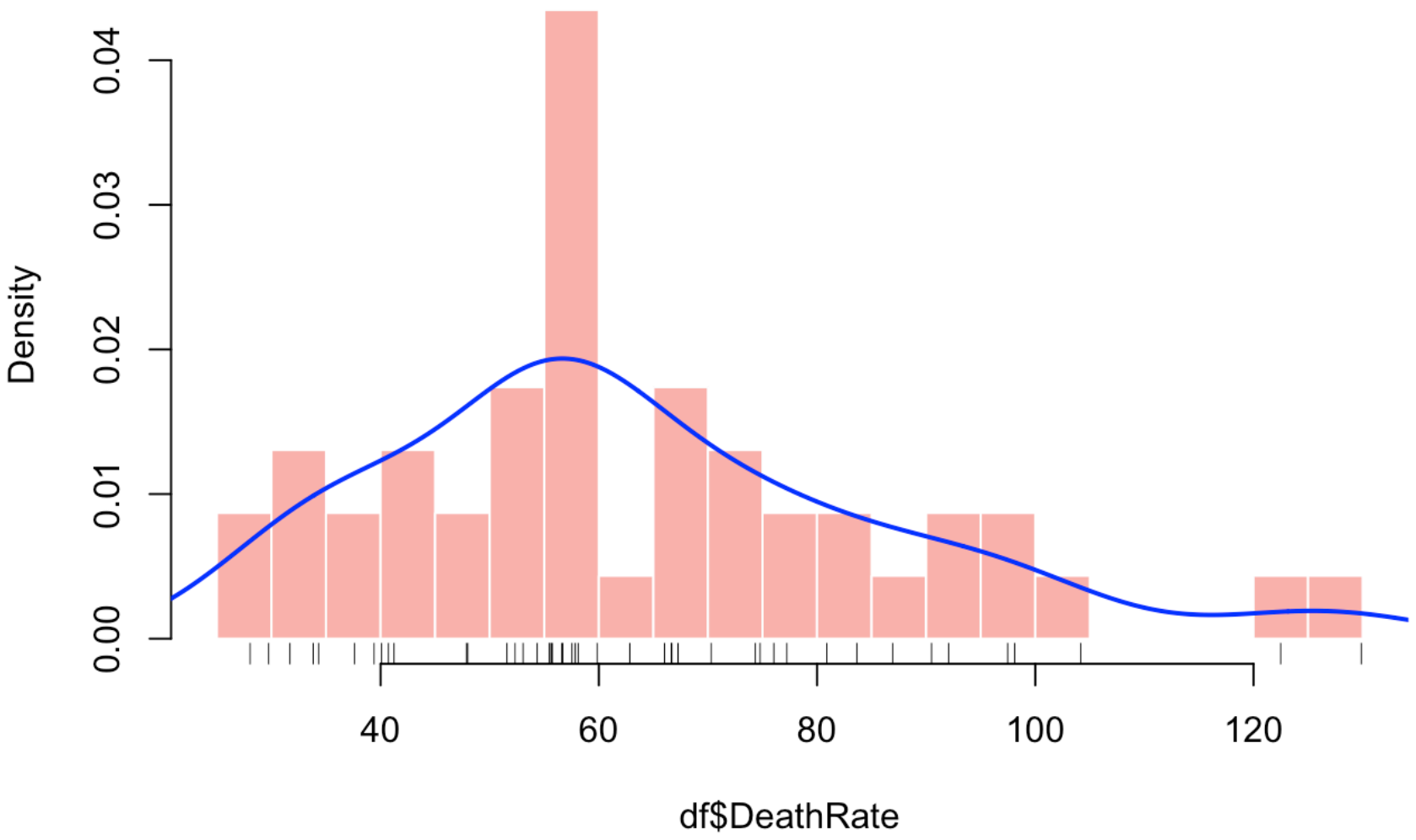
```
set.seed(1869097) # for reproducibility
```

The scope of our analysis is to show the linear relationship between liver cirrhosis mortality (*dependent variable*) and the different classes of alcohol consumption (*independent or explanatory variables*).

Let's get a look now into the features we're interested in with some summary statistics:

DeathRate

```
# let's see how DeathRate is distributed
cols <- RColorBrewer::brewer.pal(6, "Pastel1")
hist(df$DeathRate, probability = T, breaks = 20, col = cols[1], border = "white",
     main = "Histogram (and density curve) of DeathRate")
rug(df$DeathRate)

lines(density(df$DeathRate), col = "blue", lwd = 2)
```




```
summary(df$DeathRate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   28.04   48.90   57.69   63.50   75.72  129.88
```

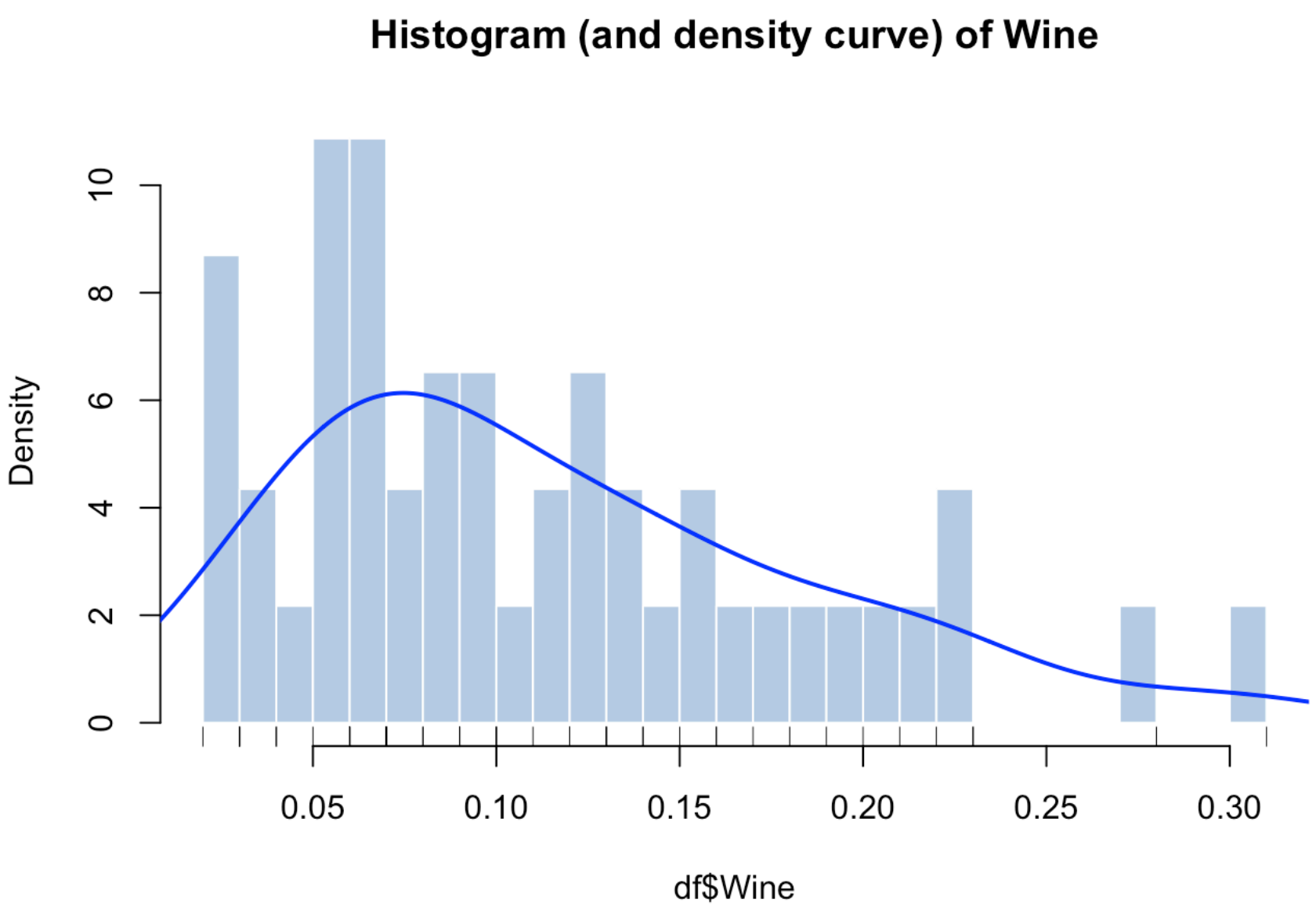
```
c(skewness = moments::skewness(df$DeathRate), kurtosis = moments::skewness(df$Deat
hRate))
```

```
##  skewness  kurtosis
## 0.8259019 0.8259019
```

DeathRate values seem to be slightly positive skewed (right tailed) and *platykurtic* (thinner tails). We have median of 57.69 and the range is between 28.04 and 129.88. The 3rd Quantile is 75.72 which means that 75% of DeathRate observations have value less than 75.72 expected deaths per standard million population.

Wine

```
# let's see how Wine per capita consumption is distributed
hist(df$Wine, probability = T, breaks = 35, col = cols[2], border = "white",
     main = "Histogram (and density curve) of Wine")
rug(df$Wine)
lines(density(df$Wine), col = "blue", lwd = 2)
```




```
summary(df$Wine)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0200  0.0625  0.1000  0.1159  0.1575  0.3100
```

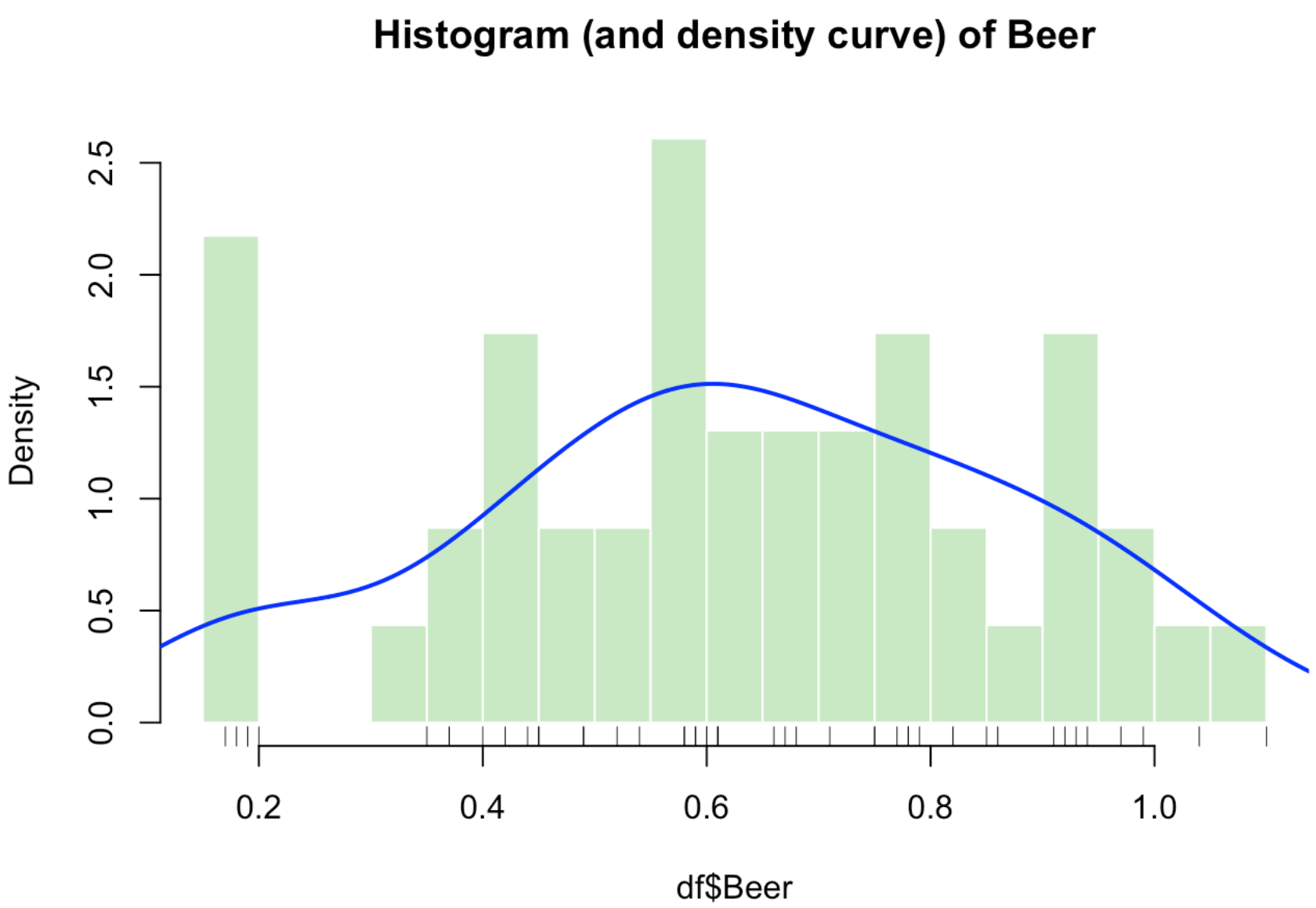
```
c(skewness = moments::skewness(df$Wine), kurtosis = moments::kurtosis(df$Wine))
```

```
##  skewness  kurtosis
## 0.8742638 3.1744782
```

Wine values seem to be positive skewed (right tailed) and *mesokurtic* (normal tailed). We have median of 0.1 gallons (pro capita) and the range is between 0.02 and 0.31 gallons. The 3rd Quantile is 0.1575 gallons which means that 75% of gallons consumption in each state have value less than 0.1575.

Beer

```
# let's see how Beer per capita consumption is distributed
hist(df$Beer, probability = T, breaks = 30, col = cols[3], border = "white",
     main = "Histogram (and density curve) of Beer")
rug(df$Beer)
lines(density(df$Beer), col = "blue", lwd = 2)
```




```
summary(df$Beer)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1700  0.4600  0.6100  0.6280  0.7875  1.1000
```

```
c(skewness = moments::skewness(df$Beer), kurtosis = moments::skewness(df$Beer))
```

```
##   skewness   kurtosis
## -0.1586508 -0.1586508
```

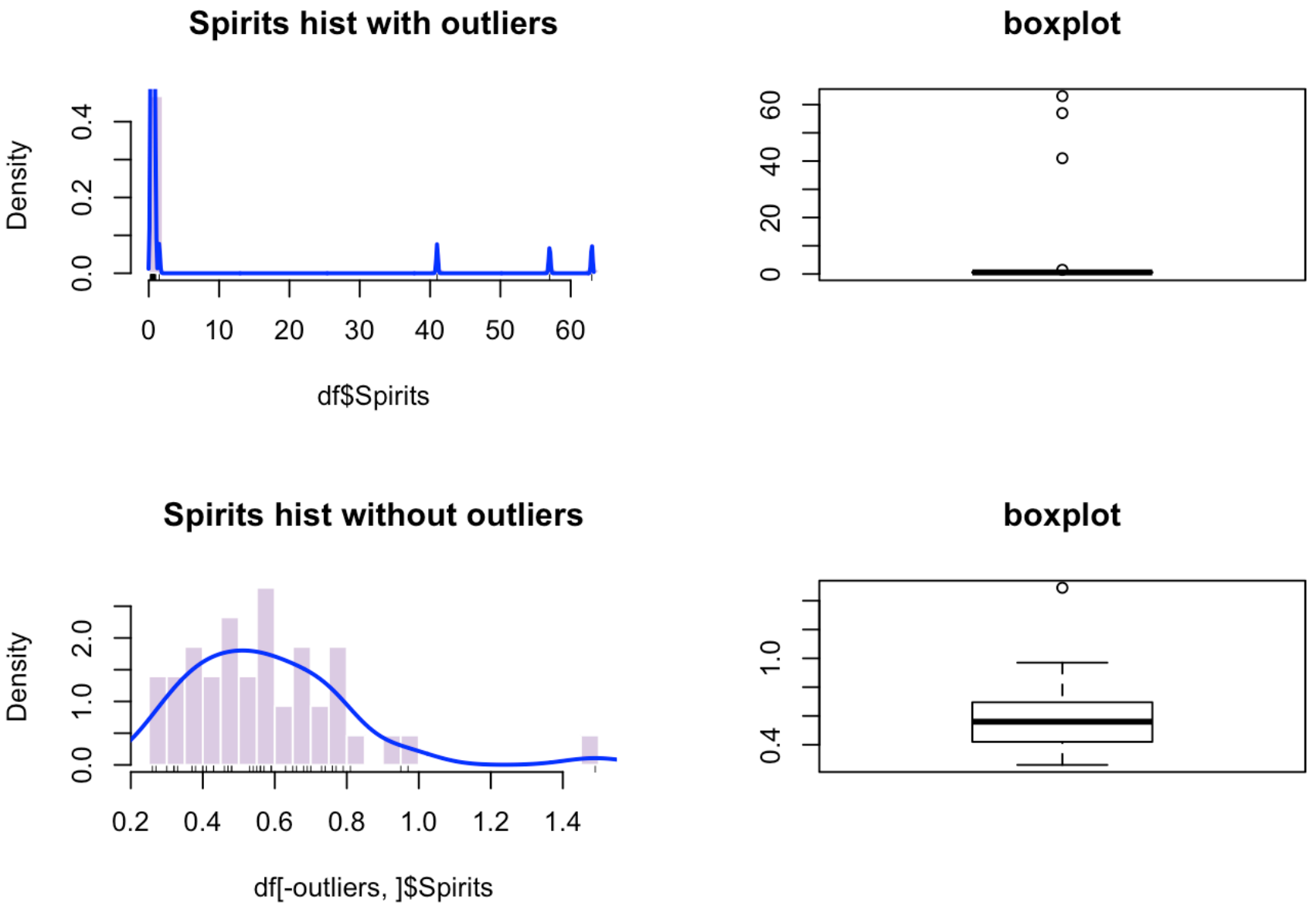
```
# State with minimum consumption of Beer
df[which.min(df$Beer),]
```

```
##     State Spirits Wine Beer Total DeathRate Urbanism
## 1 Alabama     0.3 0.05 0.17  0.52     41.23     23.6
```

Beer values are quite greater than Wine ones. We have for istance that the minimum value of gallons consumed in Beer (0.17 gallons in Alabama) is greater than the value of gallons consumed in Wine in 75% of the other states (3rd quantile is 0.1575 as we already saw). We have similar median and mean (around 0.61) and 50% of the Beer consumption is between 0.46 and 0.79 gallons.

Spirits

```
par(mfrow = c(2,2))
# let's see how Spirit per capita consumption is distributed
hist(df$Spirits, probability = T, breaks = 40, col = cols[4], border = "white",
     main = "Spirits hist with outliers")
rug(df$Spirits)
lines(density(df$Spirits), col = "blue", lwd = 2)
boxplot(df$Spirits, main = "boxplot")

# let's remove the 3 outliers
outliers <- which(df$Spirits > 10)
hist(df[-outliers,]$Spirits, probability = T, breaks = 40, col = cols[4], border =
"white",
     main = "Spirits hist without outliers")
rug(df[-outliers,]$Spirits)
lines(density(df[-outliers,]$Spirits), col = "blue", lwd = 2)
boxplot(df[-outliers,]$Spirits, main = "boxplot")
```



```
# summaries
summary(df$Spirits)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2600  0.4300  0.5650  4.0400  0.7375 63.0000
```

```
c(skewness = moments::skewness(df$Spirits), kurtosis = moments::skewness(df$Spirit
s))
```

```
## skewness kurtosis
## 3.697353 3.697353
```

Spirits has 3 outliers which have really high values w.r.t. the others. Indeed, we can see it from the fact that the mean differs much more from the median (4.04 vs 0.56 respectively). Probably it's a data collection error since a consumption pro capita of more than 40 gallons (around 151 liters) seems to be really unlikely. We noticed also how these outliers affect the correlation with DeathRate:

```
# outliers
idx <- which(df$Spirits > 10)
df[idx,]
```

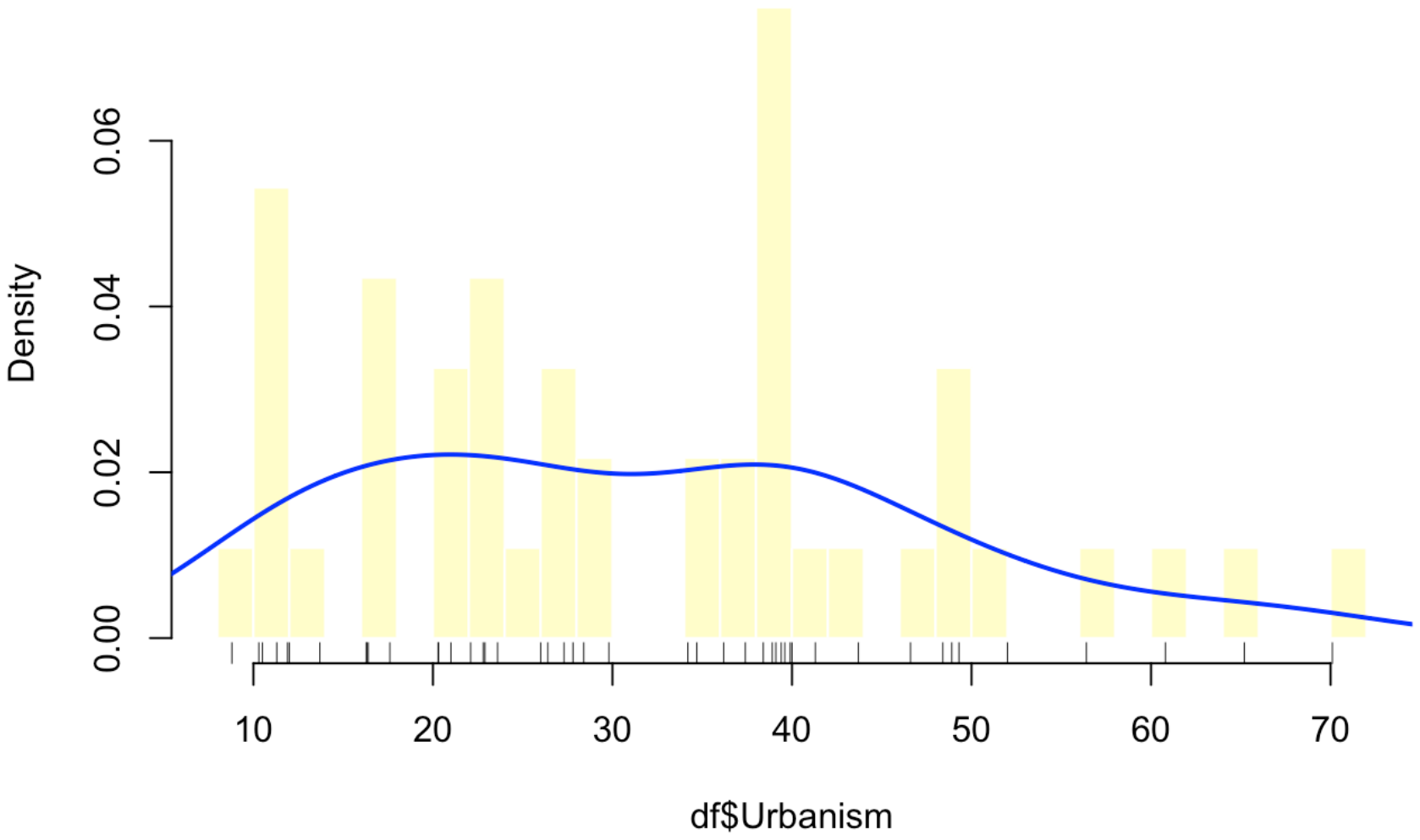
```
##              State Spirits Wine Beer Total DeathRate Urbanism
## 10          Oregon      57 0.07 0.75  1.39     56.61     29.8
## 41    Rhode Island      63 0.16 0.97  1.76     90.49     65.2
## 42 South Carolina      41 0.02 0.19  0.62     29.74     11.9
```

```
# with outliers
Spirits.correlation <- c(with_outliers = cor(df$DeathRate, df$Spirits), without_ou
tliers = cor(df[-idx,"DeathRate"], df[-idx, "Spirits"]))
```

Urbanism

The index of urbanism shows the percentage of population living in urban area.

```
# let's see how the index of Urbanism varies within the States
hist(df$Urbanism, probability = T, breaks = 30, col = cols[6], border = "white",
     main = "Histogram (and density curve) of Urbanism")
rug(df$Urbanism)
lines(density(df$Urbanism), col = "blue", lwd = 2)
```




```
summary(df$Urbanism)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.80   20.30   29.10   31.83   39.98   70.10
```

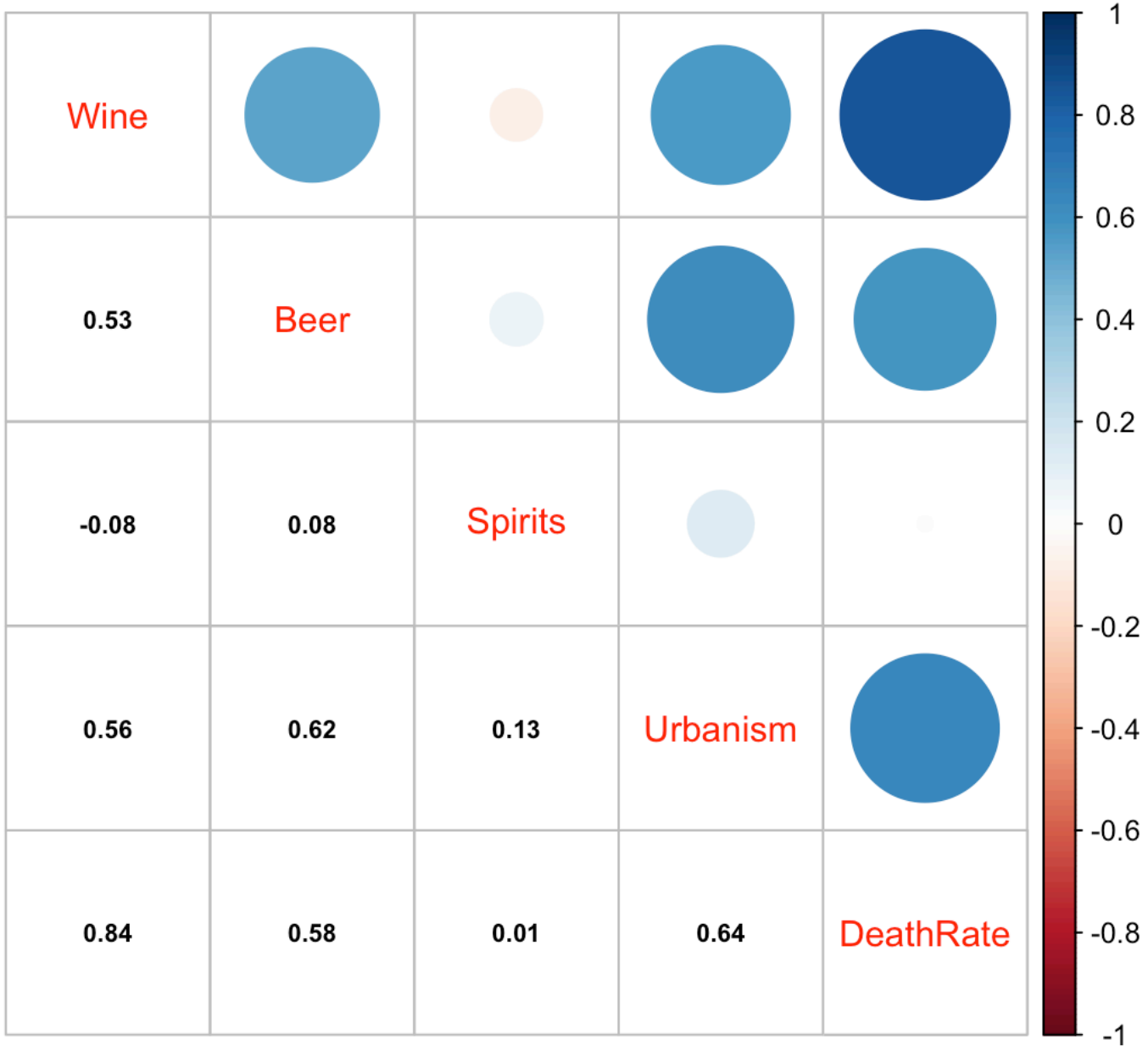
```
c(skewness = moments::skewness(df$Urbanism), kurtosis = moments::skewness(df$Urban
ism))
```

```
##  skewness  kurtosis
## 0.4755574 0.4755574
```

We have kind of flatted distribution of the values, with peaks around 10%, 22% and 40%. We notice that most of the values are below 40%.

Correlation matrix

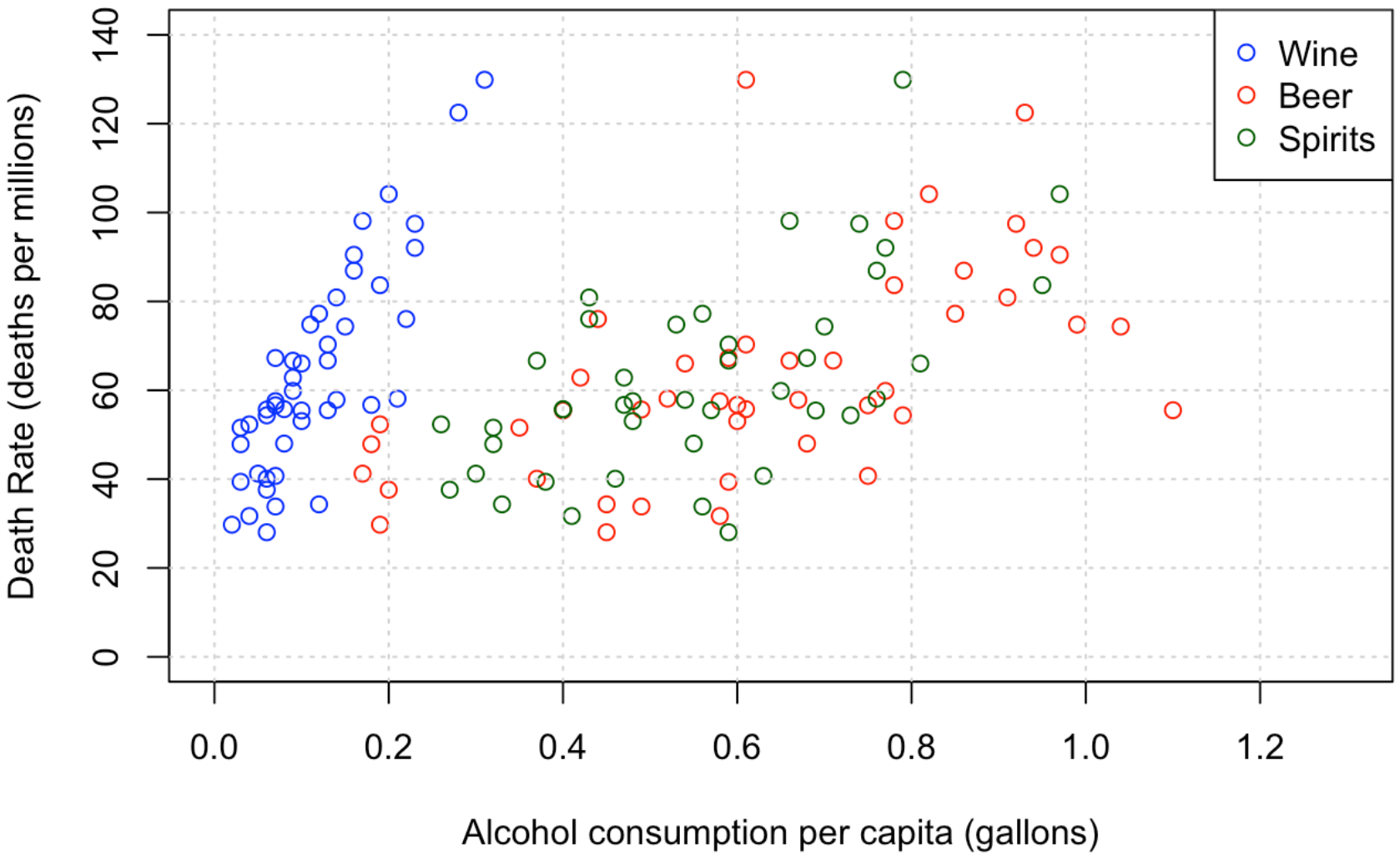
```
selected <- c("Wine" , "Beer", "Spirits" , "Urbanism", "DeathRate")
corrplot::corrplot.mixed(cor(df[selected]), lower.col = "black", number.cex = .7)
```



Alcohol consumption vs DeathRate scatterplot

We can illustrate also visually the relation between DeathRates in 1950 and alcohol consumption, using a scatterplot.

```
# making scatter-plot
plot(df$Wine, df$DeathRate, col = "blue", xlim = c(0, 1.3), ylim = c(0,140),
     xlab = "Alcohol consumption per capita (gallons)", ylab = "Death Rate (deaths
per millions)",
     main = "Liver Chirrosis deaths and alcohol consumption in 1950 (US)")
points(df$Beer, df$DeathRate, col = "red")
points(df$Spirits, df$DeathRate, col = "darkgreen")
grid()
legend("topright", legend = c("Wine", "Beer", "Spirits"),
       pch = 21, col = c("blue", "red", "darkgreen"))
```



3. Regression models

Regression modeling describes how the sampling distribution of one random variable y (in our case DeathRate variable) varies with another (set) of variable(s) $\boldsymbol{x} = (x_1, \ldots, x_p)$. Indeed, a regression model describes a form for the conditional distribution of y given $\boldsymbol{x}$, which is $f(y|\boldsymbol{x})$.

Normal regression model - a generic definition

Using a *normal regression model*, the response variable (the stochastic component of the model - r.v. of which outcome is uncertain before it's observed) has the following univariate form:

$$y|x_1, \ldots, x_p \sim N(\mu(\beta, x_1, \ldots x_p), \sigma^2)$$

where the mean is expressed as a linear combination of the explanatory (independent) variables:

$$\mu(\beta, x_1, \ldots x_p) = E[y|\boldsymbol{x}] = \int y \cdot f(y|\boldsymbol{x})dy = \beta_0 + \sum_{j=1}^{p} \beta_j x_j = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}$$

σ^2 and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)^T$ is the set of regression parameters to be estimated.

Another representation of the regression model is the following:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon\,; \qquad \varepsilon \sim N(0, \sigma^2)$$

The likelihood

It's important to enlight our assumption over the model:

- We assumed $E[y|\boldsymbol{x}]$ being linear
- We assume the sampling variability around the mean as being i.i.d. from a normal distribution ($\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$)

In this way, we're able to provide a specification of **the Likelihood** - which is - the joint probability density of the observed data, let's say $y_1, \dots, y_n$ conditionally on $\boldsymbol{x_1}, \dots, \boldsymbol{x_n}, \boldsymbol{\beta}, \sigma^2$:

$$
\begin{aligned}
(1) \quad \mathcal{L}_{y_1,\dots,y_n}(\boldsymbol{x_1}, \dots, \boldsymbol{x_n}, \boldsymbol{\beta}, \sigma^2) &= f(y_1, \dots, y_n | \boldsymbol{x_1}, \dots, \boldsymbol{x_n}, \boldsymbol{\beta}, \sigma^2) \\
&= \prod_{i=1}^{n} f(y_i | \boldsymbol{x_i}, \boldsymbol{\beta}, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left\{ -\frac{(y_i - \mu_i)^2}{2\sigma^2} \right\} \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^n \cdot \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \boldsymbol{\beta^T x_i}) \right)^2 \right\}
\end{aligned}
$$

Notice that the sum in the exponent is the so called *Sum of Squared Residuals*, $SSR(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - (\beta_0 + \boldsymbol{\beta^T x_i}) \right)^2$.

3. Models elicitation

We will use 3 different linear models:

1. A linear model with only one explanatory variable (Wine)

2. A linear model with 2 explanatory variables (Wine and Beer)

3. A linear model with all the explanatory variables (Wine, Beer, Spirits, Urbanism)

For each model we'll perform:

- Frequentistic estimation of β's parameters (first two models)
- Bayesian estimation of the parameters through MCMC simulation
    - Estimation using different priors for the parameters β and σ
    - MCMC diagnostic (distplots, traceplots, autocorrelation, …)
    - Comparison between the different bayesian models (using **DIC** - *Deviance Information Criterion*)
- Final comparison between Bayesian and Frequentistic approaches
    - One possible way is compare models using $RMSE$ (*Root Mean Squared Error*), which is defined as the squared root of the MSE. Given m predictions $\left\{ \hat{y_i}, \dots, \hat{y_m} \right\}$ of a generic model (also called predictor) the RMSE is defined as:

$$RMSE = \sqrt{MSE} = \sqrt{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$$

A note on MCMC diagnostic

We use diagnostic methods to confirm wether or not our chains converge to the stationary distribution. Which means that, if this is true, our generated samples with MCMC simulation come from their correct target distribution (which is, for a generic parametric vector $\boldsymbol{\theta}$, the posterior $\pi_{\boldsymbol{\theta}}(\boldsymbol{y})$). To do so, there are different methods:

- **Traceplots**: The plots of the iterations versus the generated values. We could assume convergence if we don't see strong periodicities and tendencies assumed by the values.
- **Autocorrelation**: Monitoring autocorrelations is also very useful since low or high values indicate fast or slow convergence, respectively. The *lag-k* autocorrelation, ρ_k , is defined as the correlation between every draw and its k-th lag:

$$\rho_k = \frac{\sum_{i=1}^{n-k}(x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

We expect to see the lag-k being smaller as k increases

- **Graphical behaviour of the empirical averages**: for each parameter we can see if its empircal average stabilize after some iterations. This is another indication of convergence of the algorithm (we say that our process *may* have *ergodic properties*).

A note on *DIC - Deviance Information Criterion*.

The Deviance Information Criterior (DIC) is a measure of model comparison and adequacy. Smaller DIC values indicate a better fitting model. Thus, the smaller DIC a model has, the better it is. It is defined as:

$$\begin{aligned} DIC(m) &= 2\overline{D(\boldsymbol{\theta}_m, m)} - D(\bar{\boldsymbol{\theta}}_m, m) \\ &= D(\bar{\boldsymbol{\theta}}_m, m) + 2p_m \\ &= \overline{D(\boldsymbol{\theta}_m, m)} + p_m \end{aligned}$$

where $D(\boldsymbol{\theta}_m, m)$ is the deviance measure defined as minus twice the log-likelihood

$$D(\boldsymbol{\theta}_m, m) = -2\log f(\boldsymbol{y} \mid \boldsymbol{\theta}_m, m)$$

and $\overline{D(\boldsymbol{\theta}_m, m)}$ is its posterior mean, defined as:

$$\begin{aligned} \overline{D(\boldsymbol{\theta}_m, m)} &= E_{\theta \mid \boldsymbol{y}}\left[D(\theta_m, m) \mid \boldsymbol{y}\right] \\ &= \int D(\theta_m, m)\pi(\theta \mid \boldsymbol{y})\, d\theta \\ &= \int -2\log f(\boldsymbol{y} \mid \theta_m, m)\pi(\theta \mid \boldsymbol{y})\, d\theta \end{aligned}$$

and

$$\widehat{D(\theta_m, m)} \approx \frac{1}{M} \cdot \sum_j -2 \log f(\mathbf{y} \mid \theta^j)$$

p_m can be seen as the "number of unconstrained (effective) parameters" for model m, and it's given by:

$$p_m = \overline{D(\boldsymbol{\theta}_m, m)} - D(\bar{\boldsymbol{\theta}}_m, m)$$

$\bar{\boldsymbol{\theta}}_m$ is the posterior mean of the parameters involved in model m.

For non-hierarchical models, p_m should be approximately the true number of parameters.

Notice also that DICs are comparable only over models with exactly the same observed data. This is the reason why we decided to drop the 3 outliers in section 1: the last model will improve more than the others.

Removal of the outliers

Since in our analysis at some point we'll use Spirits variable, which has 3 outliers with really odd values, in order to make proper comparisons between different bayesian models, we decided to drop the 3 states with that odd values (see section 3 for more details regarding model comparisons).

```
df <- df[-idx,]
```

```
# let's see again DeathRate and Spirits correlation improvement
# before and after outliers removal
Spirits.correlation
```

```
##    with_outliers without_outliers
##      0.005850855      0.683230047
```

---

1. Linear model with only one explanatory variable (Wine)

Denoting the **Wine** variable with x_1 , we are dealing with the following linear model:

$$y = \beta_0 + \beta_1 x_1$$

Now, denoting $\boldsymbol{y} = (y_1, \dots, y_n)$ and $\boldsymbol{x_1} = (x_{1,1}, \dots, x_{n,1})$, from equation (1), with only one explanatory variable x_1 , we have that:

$$
\begin{aligned}
\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{x_1}, \beta_0, \beta_1, \sigma^2) &= f(\boldsymbol{y}|\boldsymbol{x_1}, \beta_0, \beta_1, \sigma^2) \\
&= \prod_{i=1}^{n} f(y_i|x_{i,1}, \beta_0, \beta_1, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left\{ -\frac{(y_i - \mu_i)^2}{2\sigma^2} \right\} \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^n \cdot \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 \cdot x_{i,1}) \right)^2 \right\} \\
&\propto \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 \cdot x_{i,1}) \right)^2 \right\}
\end{aligned}
$$

hence

$$
SSR(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 \cdot x_{i,1}) \right)^2
$$

Frequentistic approach

Ordinary Least Squares estimation

We can use this approximation method with $lm()$ function in R. It approximates β_0 and β_1 by maximizing the likelihood (hence by minimizing the $SSR(\boldsymbol{\beta})$). In R we have:

```
# OLS - Ordinary Least Squares estimation --------------------------------

x = df$Wine         # independent variable - Wine consumption
y = df$DeathRate    # dependent variable - Death Rate
n = length(y)       # sample size

# Fitting the linear model
ols.model.1 = lm(y ~ x)

# Get a look to the coefficients beta0 and beta1
ols.model.1$coefficients
```

```
## (Intercept)           x
##    30.56095   281.52739
```

```
beta.vec.ols = ols.model.1$coefficients;
names(beta.vec.ols) = c('beta0', 'beta1')
beta.vec.ols
```

```
##     beta0     beta1
##  30.56095 281.52739
```

```
# compute RMSE on train set (we don't have test set!)
y.predict.ols = beta.vec.ols[[1]] + beta.vec.ols[[2]]*x
rmse.ols.mod1 = c(rmse.ols.m1 = Metrics::rmse(y, y.predict.ols)); rmse.ols.mod1
```

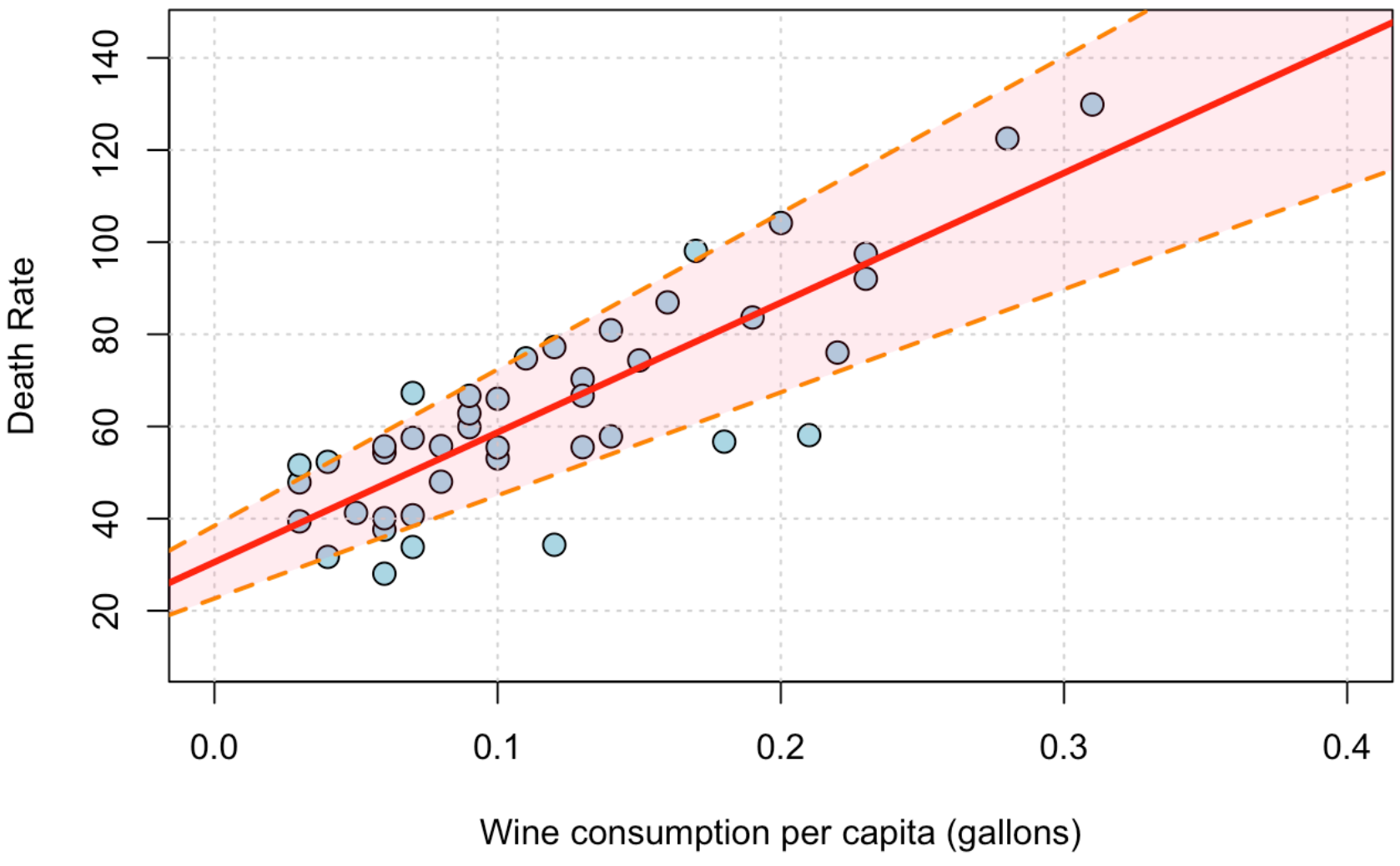
```
## rmse.ols.m1
##    12.56579
```

Obtained the estimates of the β's, we now plot the points and the fitted regression line with its 95% confidence bands.

```
# Plot the linear regression line
x_seq = seq(min(x)-0.05,max(x)+0.5, 0.01)
y_seq.ols = beta.vec.ols[1] + beta.vec.ols[2]*x_seq
plot(x, y, xlim = c(0, 0.4), ylim = c(10,145), cex = 1.4, pch = 21,bg = 'lightblue
',
     xlab = "Wine consumption per capita (gallons)", ylab = "Death Rate",
     main = "Linear regression with OLS")
grid()
lines(x_seq, y_seq.ols, col="red", lwd=3)

# 95% confidence level
confint(ols.model.1)
```

```
##                 2.5 %    97.5 %
## (Intercept)  22.66396  38.45794
## x           223.71057 339.34422
```

```
# beta's bounds
beta0.ci <- confint(ols.model.1)[1,]
beta1.ci <- confint(ols.model.1)[2,]

# low and upper bounds (for the polygon function)
lower.bounds  <- beta0.ci[[1]] + beta1.ci[[1]] * x_seq
upper.bounds <- beta0.ci[[2]] + beta1.ci[[2]] * x_seq

# plotting the bounds
abline(a = beta0.ci[2], b = beta1.ci[2], lwd = 2, lty = 2, col = "darkorange")
abline(a = beta0.ci[1], b = beta1.ci[1], lwd = 2, lty = 2, col = "darkorange")
polygon(c(x_seq,rev(x_seq)), c(lower.bounds,rev(upper.bounds)),
        col = rgb(1,.2,.4,.1), border = NA)
```



First Bayesian Analysis: variance with InverseGamma distribution

Our model is $y_i|x_{i,1}, \boldsymbol{\beta}, \sigma^2 \overset{iid}{\sim} N(\mu_i, \sigma^2)$ where μ_i is defined as:

$$\mu_i = \beta_0 + \beta_1 x_{i,1}$$

We perform a non-conjugate bayesian analysis, assignign to the parameters the following priors:

$$\begin{aligned}\beta_0 &\sim N(0, 10^{-4}) \\ \beta_1 &\sim N(0, 10^{-4}) \\ \tau^2 &\sim Gamma(10^{-3}, 10^{-3})\end{aligned}$$

Where $\tau = 1/\sigma^2$ is the precision.

We already know the likelihood as:

$$\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{x_1}, \beta_0, \beta_1, \tau^2) \propto \exp\left\{ -\frac{\tau^2}{2} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 \cdot x_{i,1})\right)^2 \right\}$$

The full conditionals for the parameters are:

$$\pi(\beta_0|\beta_1, \tau^2, x_1, y) = \frac{\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{x_1}, \beta_0, \beta_1, \tau^2) \cdot \pi(\beta_0)}{f(y)} \propto \mathcal{L}_{\boldsymbol{y}}(\boldsymbol{x_1}, \beta_0, \beta_1, \tau^2) \cdot \pi(\beta_0)$$

$$\pi(\beta_1|\beta_0, \tau^2, x_1, y) = \frac{\mathcal{L}_{\boldsymbol{y}}(\boldsymbol{x_1}, \beta_0, \beta_1, \tau^2) \cdot \pi(\beta_1)}{f(y)} \propto \mathcal{L}_{\boldsymbol{y}}(\boldsymbol{x_1}, \beta_0, \beta_1, \tau^2) \cdot \pi(\beta_1)$$

$$\pi(\tau^2|\beta_0,\beta_1,,x_1,y) = \frac{\mathcal{L}_y(\boldsymbol{x_1},\beta_0,\beta_1,\tau^2)\cdot\pi(\tau^2)}{f(y)} \propto \mathcal{L}_y(\boldsymbol{x_1},\beta_0,\beta_1,\tau^2)\cdot\pi(\tau^2)$$

We're ready to use RJAGS package in order to estimate the parameters

```
##### first model
# linear mu, sigma with inverse gamma prior (tau ~ gamma prior)

library(R2jags, quietly = T)
```

```
data = list( Y = y,
             x = x,
             N = n)

first.model = function() {
  for(i in 1:N){
    # likelihood
    Y[i] ~ dnorm(mu[i], tau)
    mu[i] = beta[1] + beta[2]*x[i]
  }

  # std normal priors for beta's
  beta[1] ~ dnorm(0.0, 1.0E-4)
  beta[2] ~ dnorm(0.0, 1.0E-4)

  # prior for sigma (using precision tau = 1/sigma)
  tau ~ dgamma(0.001,0.001)
  sigma = sqrt(1.0 / tau) # precision
}

# jags parameters
params = c("beta", "sigma")

## inits for the model
inits.mod1 = function(){
  list("beta" = rnorm(2, mean = 0.0, sd = 10.0),
       "tau" = rgamma(1, shape = 1.0, rate = 1.0))
}


# run the model:
# 3 chains
# 15000 iterations (burn in of 2000)
model.1 = jags(data=data, inits.mod1, params, n.chains=3,
               n.iter=17000, n.burnin=2000, n.thin = 1,
               model.file=first.model, DIC=TRUE)
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 43
##    Unobserved stochastic nodes: 3
##    Total graph size: 142
##
## Initializing model
```

```
model.1
```

```
## Inference for Bugs model at "/var/folders/s2/9gqx63yx2jsb9hg77fr834mw0000gn/T//
Rtmp3zcQs5/model136a29b2694e.txt", fit using jags,
##  3 chains, each with 17000 iterations (first 2000 discarded)
##  n.sims = 45000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
## beta[1]   33.171   3.938  25.614  30.527  33.117  35.760  41.021 1.001
## beta[2]  259.384  28.702 201.544 240.488 259.925 278.784 314.287 1.001
## sigma     13.188   1.513  10.640  12.119  13.047  14.112  16.527 1.001
## deviance 343.325   2.919 339.953 341.186 342.597 344.667 350.861 1.001
##          n.eff
## beta[1]  12000
## beta[2]  13000
## sigma    27000
## deviance  5700
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 4.3 and DIC = 347.6
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# DIC value
model.1$BUGSoutput$DIC
```

```
## [1] 347.5854
```

```
# picking up beta's and sigma summaries
beta = rep(NA, 2)
names(beta) = c('beta_0', 'beta_1')
beta[1] = model.1$BUGSoutput$summary[,"mean"]["beta[1]"]
beta[2] = model.1$BUGSoutput$summary[,"mean"]["beta[2]"]
beta
```

```
##    beta_0    beta_1
##  33.17113 259.38450
```

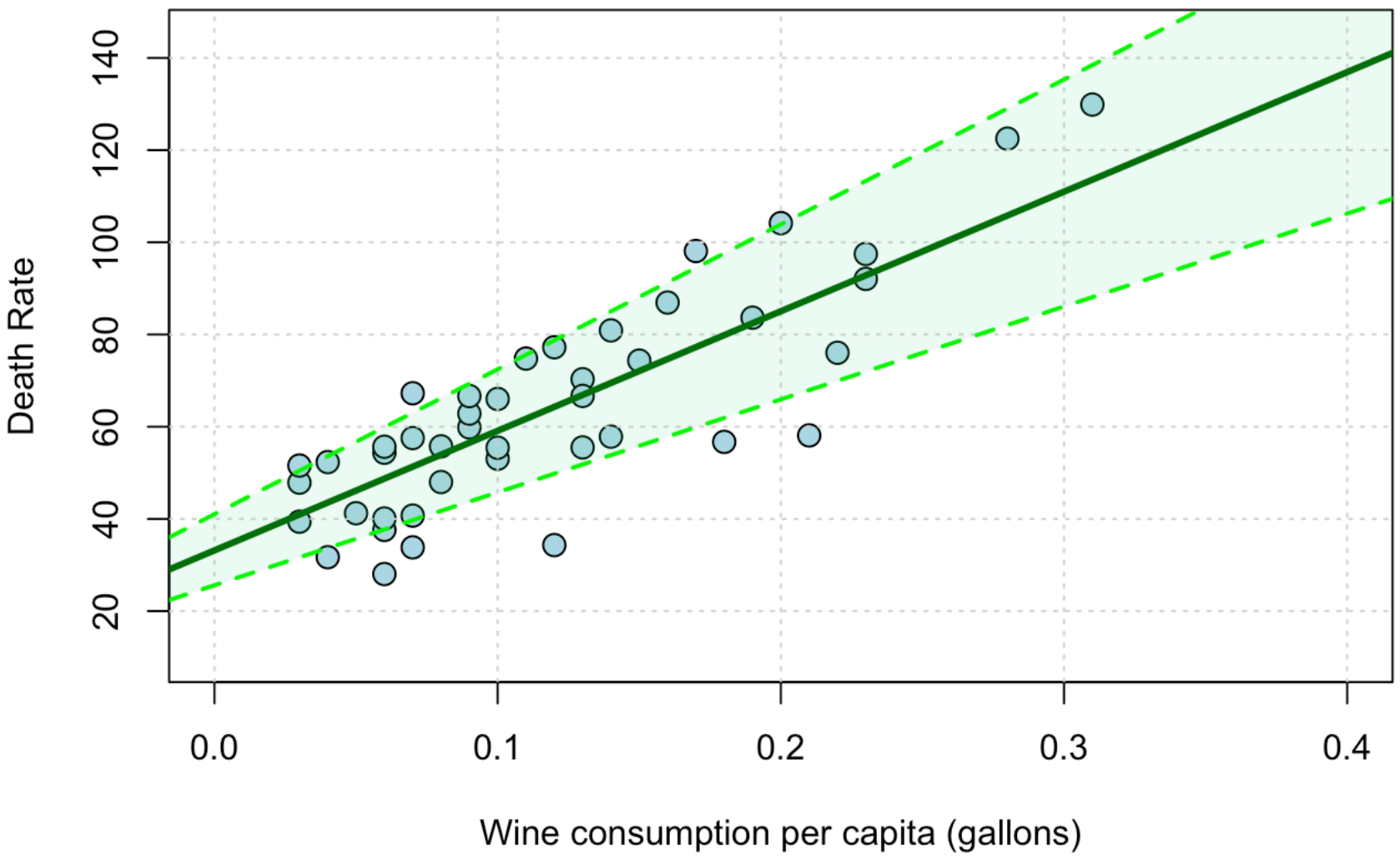
```
sigma = model.1$BUGSoutput$summary[,"mean"]["sigma"]
sigma
```

```
##    sigma
## 13.18802
```

```
# RMSE on train set using mean for beta's
y.predict.model.1 = beta[1] + beta[2]*x
rmse.model.1 = Metrics::rmse(y, y.predict.model.1); rmse.model.1
```

```
## [1] 12.65714
```

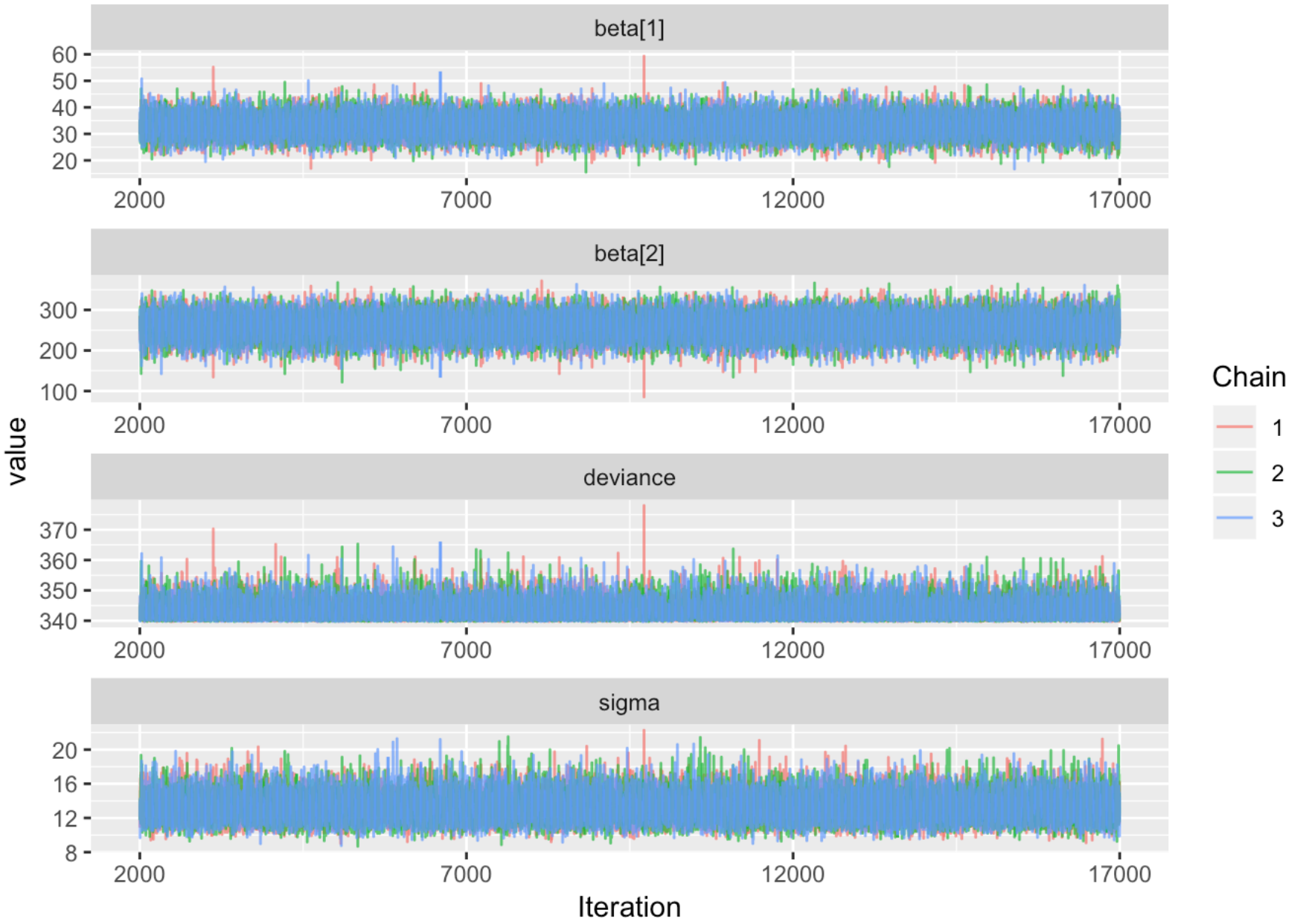
```
# Plot the linear regression line
x_seq = seq(min(x)-0.05,max(x)+0.5, 0.01)
y_seq.model.1 = beta[1] + beta[2]*x_seq

plot(x, y, xlim = c(0, 0.4), ylim = c(10,145), cex = 1.4, pch = 21,bg = 'lightblue
',
     xlab = "Wine consumption per capita (gallons)", ylab = "Death Rate",
     main = "Linear regression with Bayesian approach - 1st model")
grid()

lines(x_seq, y_seq.model.1,col="darkgreen", lwd=3)

beta0.ci = model.1$BUGSoutput$summary["beta[1]", c("2.5%", "97.5%")]
beta1.ci = model.1$BUGSoutput$summary["beta[2]", c("2.5%", "97.5%")]

lower.bounds  <- beta0.ci[[1]] + beta1.ci[[1]] * x_seq
upper.bounds <- beta0.ci[[2]] + beta1.ci[[2]] * x_seq

# plotting the bounds
abline(a = beta0.ci[2], b = beta1.ci[2], lwd = 2, lty = 2, col = "green")
abline(a = beta0.ci[1], b = beta1.ci[1], lwd = 2, lty = 2, col = "green")
polygon(c(x_seq,rev(x_seq)), c(lower.bounds,rev(upper.bounds)),
        col = rgb(.1,.9,.5,.1), border = NA)
```



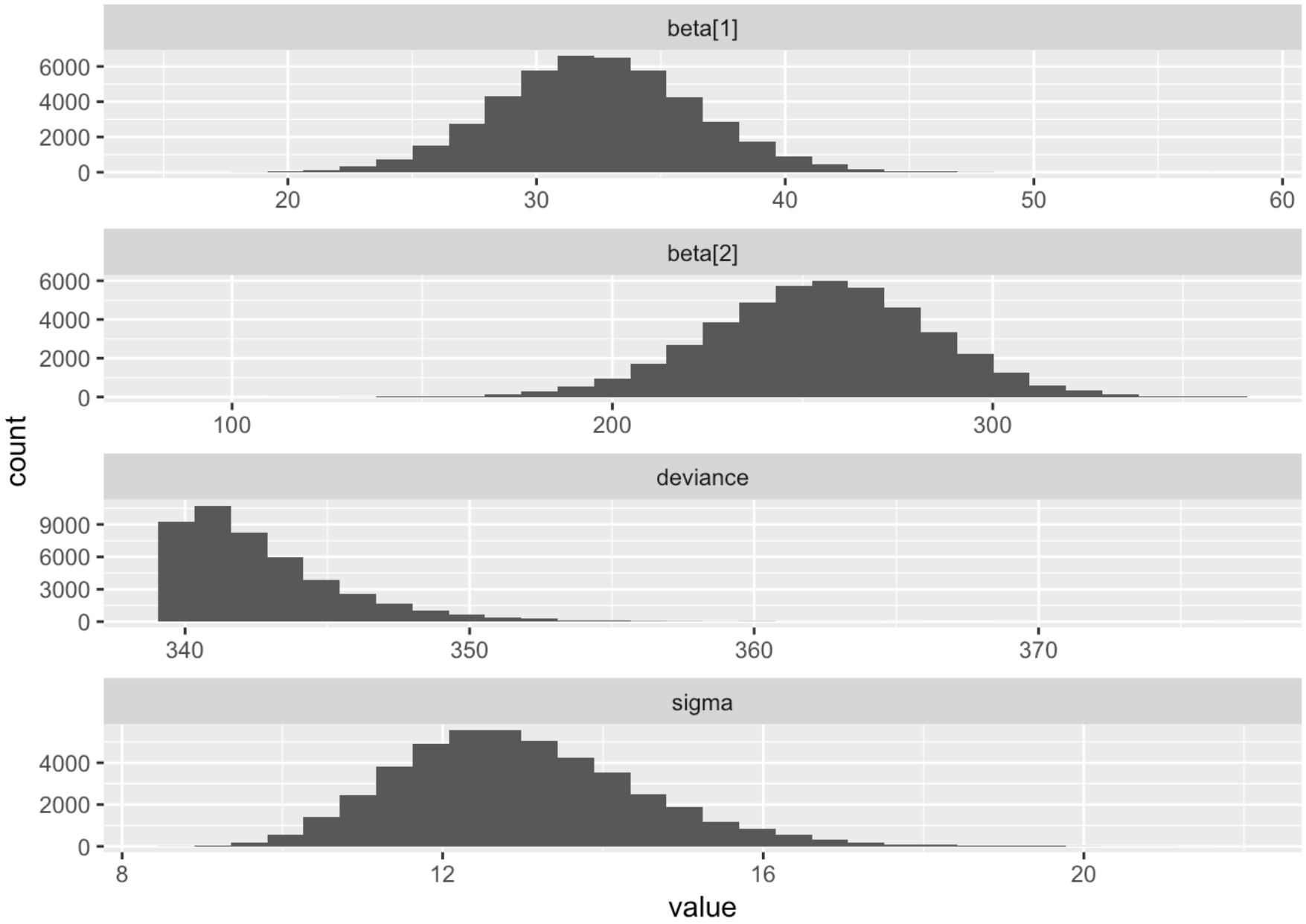
MCMC diagnostic - first model

```
# MCMC diagnostic ------------------------------------------------------

library(ggmcmc, quietly = T)
```

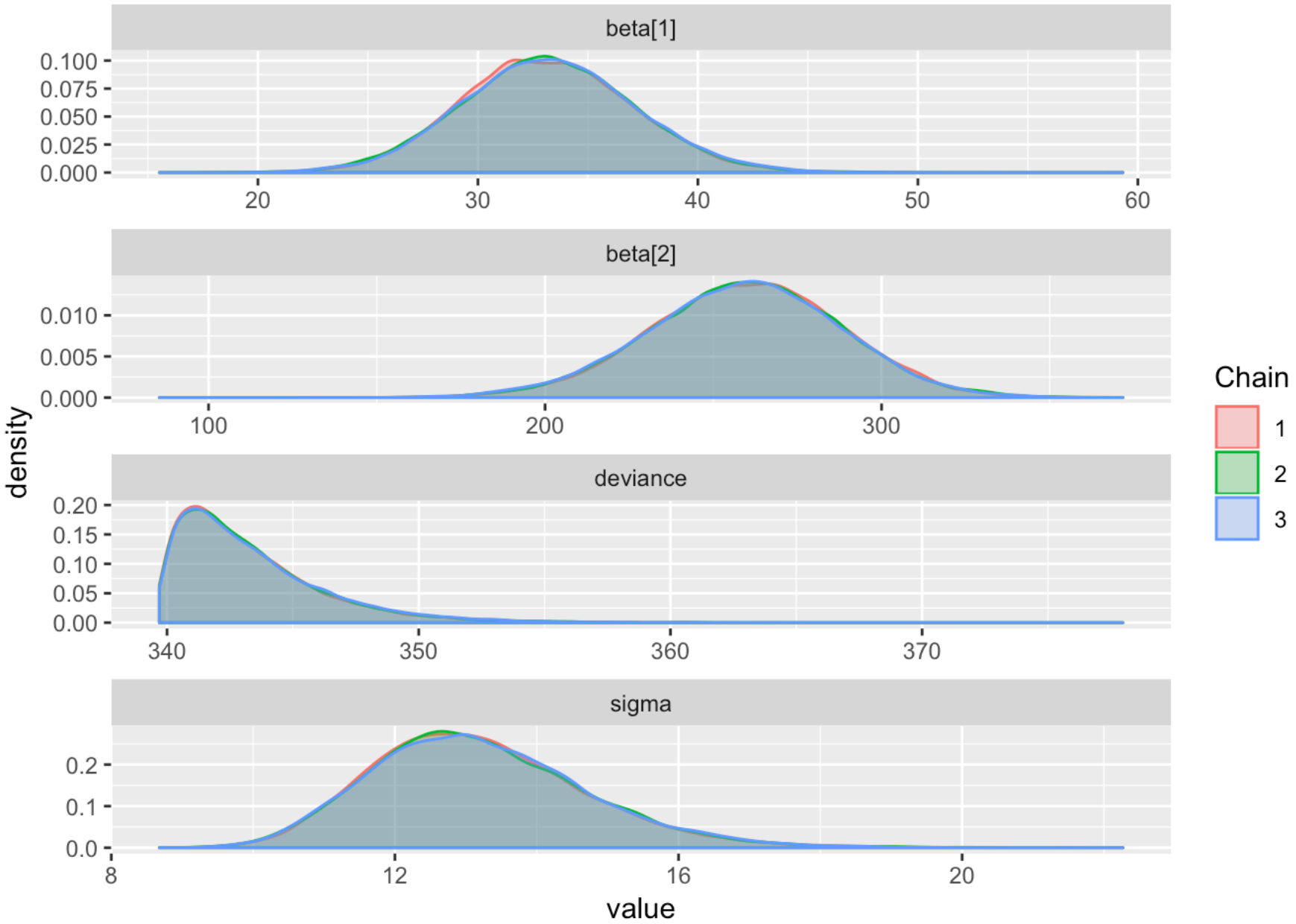
```
# Create ggs object using ggmcmc library
ggs.mod1 = ggs(as.mcmc(model.1))

ggs_traceplot(ggs.mod1)
```



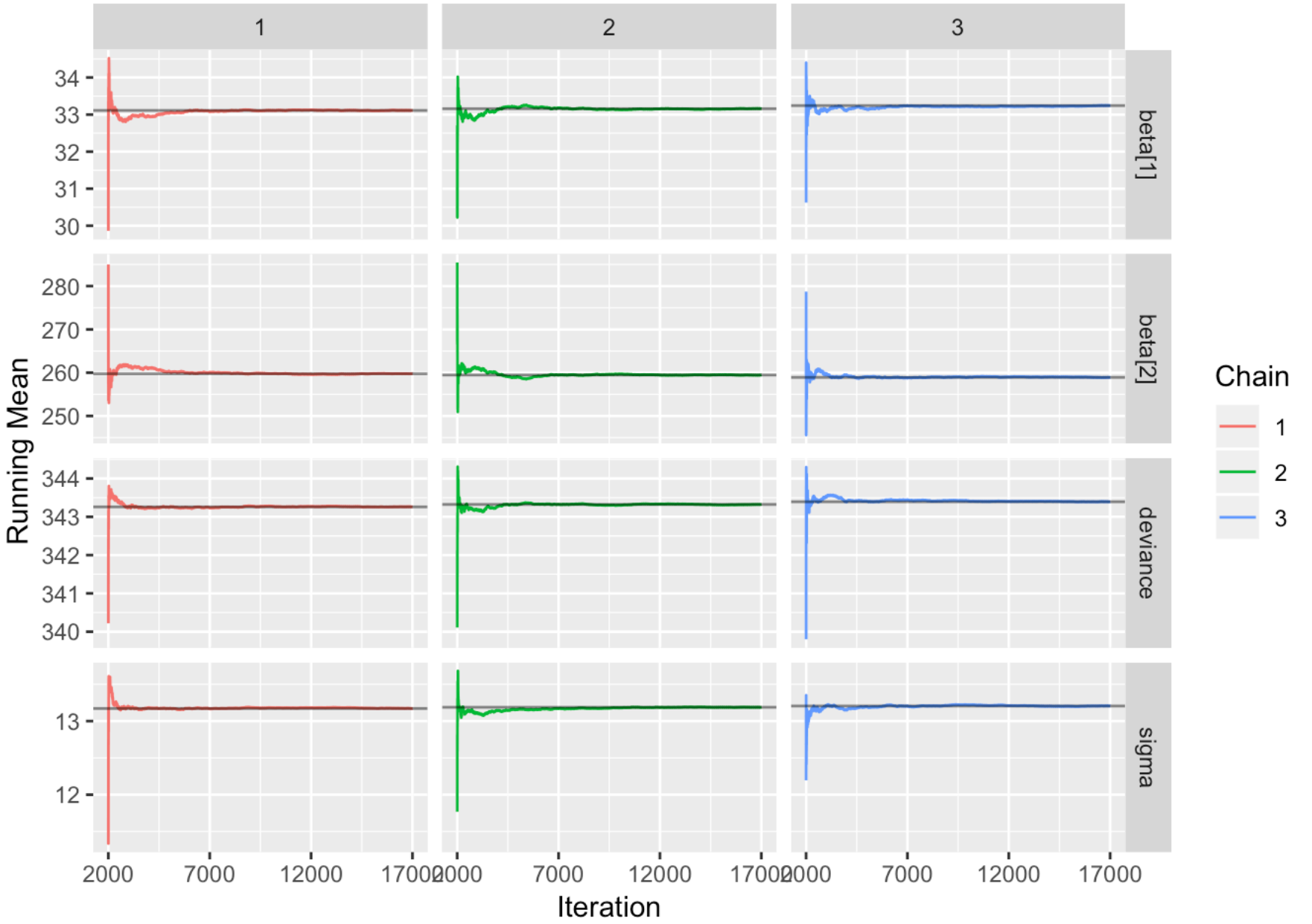

```
# let's see how the values are sampled
ggs_histogram(ggs.mod1)
```



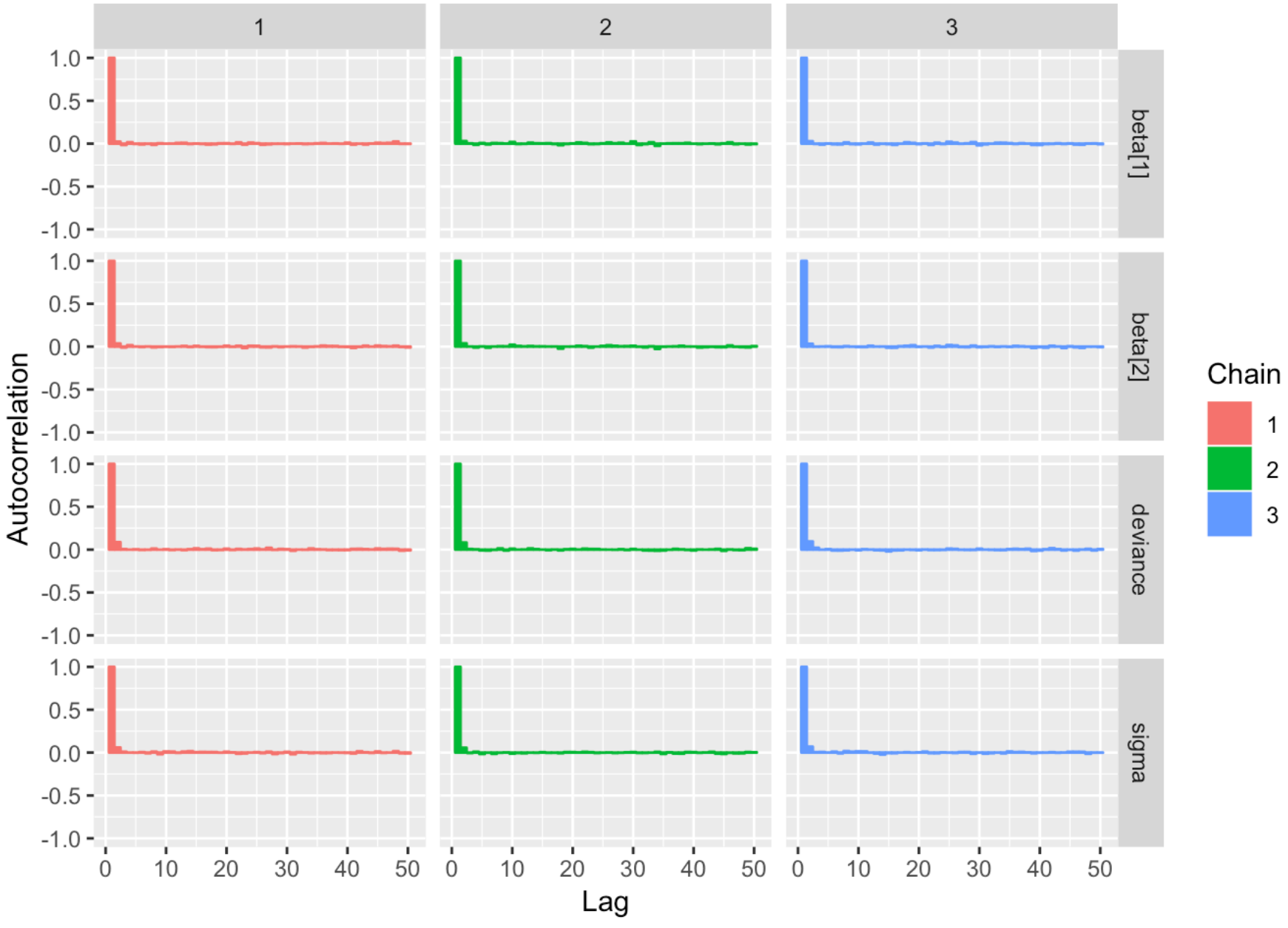

```
# we can see it as densities
ggs_density(ggs.mod1)
```




```
# let's see graphically the behaviour of the empirical averages
# of the parameters with the increase of t = 1,...., 15000
ggs_running(ggs.mod1)
```



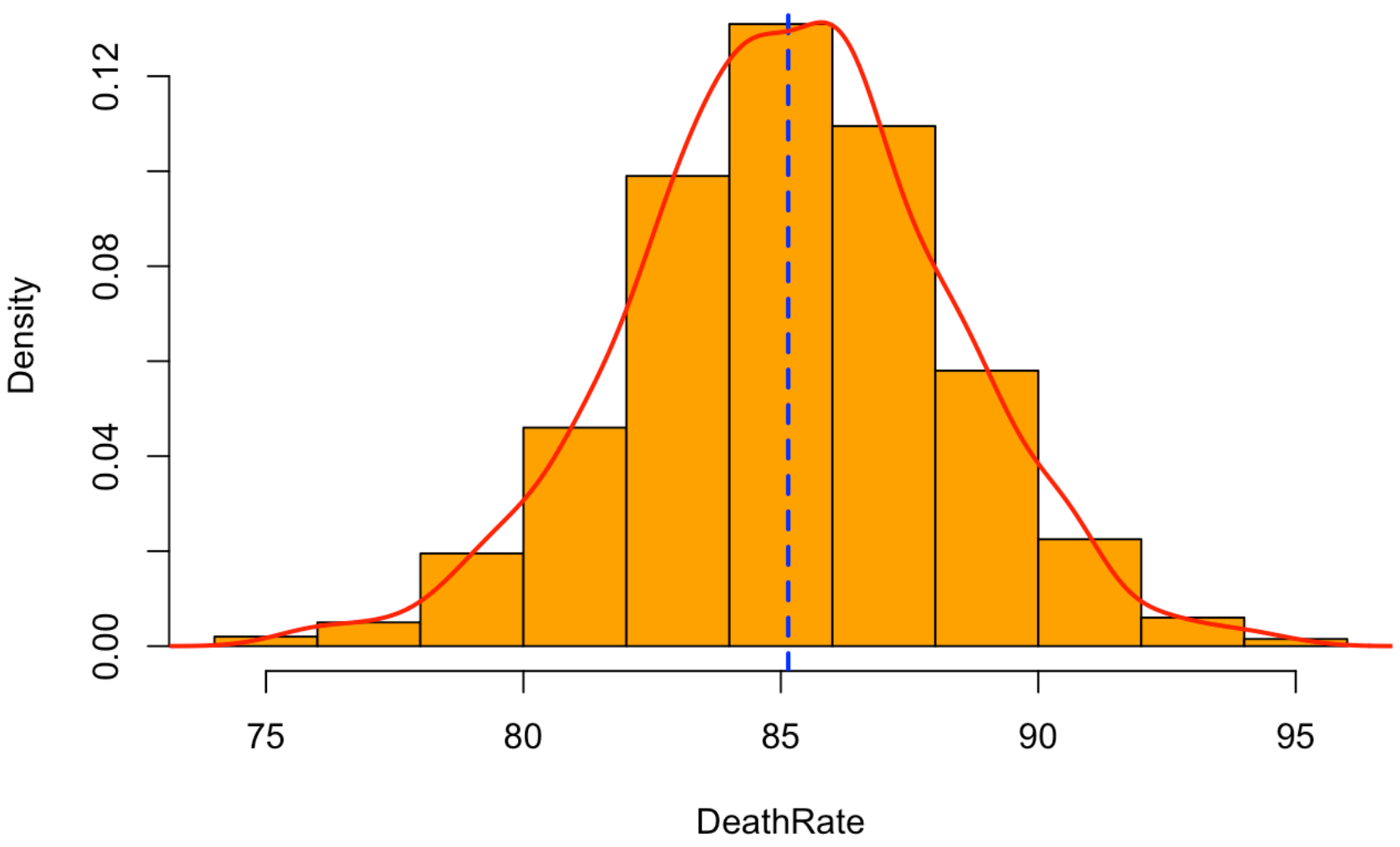

```
# good! our process may have suitable ergodic properties

# let's see the autocorrelation of the chain
ggs_autocorrelation(ggs.mod1)
```



Plus: Approximation of the posterior predictive distribution of death rate per million for wine consumption of 0.2 gallons.

We can use the Markov chain to approximate the posterior predictive distribution of the DeathRate of a State which is consuming on average 0.2 gallons of Wine.

```
simsize <- 1000
wine_0.2 <- rep(NA, simsize)
XX <- model.1$BUGSoutput$sims.list

for(t in (1:simsize)){
  mu.temp <- XX$beta[t,1]+ XX$beta[t,2]*0.2
  sd.temp <- sqrt(1/XX$sigma[t])
  wine_0.2[t] <- rnorm(1, mu.temp, sd.temp)
}


hist(wine_0.2, probability = T, col = "orange", xlab = "DeathRate", main = "")
title("Approximation of the posterior predictive distribution \n 0.2 gallons wine
consumption (pro capita)")
abline(v = mean(wine_0.2), col = "blue", lwd = 2, lty = 2)
lines(density(wine_0.2), col = "red", lwd= 2)
```




```
c(DeathRate_prediction = mean(wine_0.2))
```

```
## DeathRate_prediction
##              85.1438
```

Second Bayesian Analysis: flat prior distribution

Instead of using a gamma prior for τ^2 , let's use a uniform distribution:

$$\tau^2 \sim Unif(0, 10)$$

We can derive Likelihood and full conditionals in the same way as for the model one. In RJAGS we have:

```
##### second model
# linear mu, tau with flat prior (tau ~ uniform prior)

# data already loaded

# second model
second.model = function() {
  for(i in 1:N){
    # likelihood
    Y[i] ~ dnorm(mu[i], tau)
    mu[i] = beta[1] + beta[2]*x[i]
  }

  # std normal priors for beta's
  beta[1] ~ dnorm(0.0, 1.0E-4)
  beta[2] ~ dnorm(0.0, 1.0E-4)

  # prior for sigma (using precision tau = 1/sigma)
  tau ~ dunif(0.0, 10.0)
  sigma = sqrt(1.0 / tau) # precision
}

# jags parameters
params = c("beta", "sigma")

## inits for the model
inits.mod2 = function(){
  list("beta" = rnorm(2, mean = 0.0, sd = 10.0),
       "tau" = runif(1, 0.0, 10.0))
}


# run the model:
# 3 chains
# 15000 iterations (burn in of 2000)
model.2 = jags(data=data, inits.mod2, params, n.chains=3,
               n.iter=17000, n.burnin=2000, n.thin = 1,
               model.file=second.model, DIC=TRUE)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 43
##    Unobserved stochastic nodes: 3
##    Total graph size: 142
##
## Initializing model
```

```
model.2
```

```
## Inference for Bugs model at "/var/folders/s2/9gqx63yx2jsb9hg77fr834mw0000gn/T//
Rtmp3zcQs5/model136a2c0535a1.txt", fit using jags,
##  3 chains, each with 17000 iterations (first 2000 discarded)
##  n.sims = 45000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
## beta[1]   33.057   3.838  25.739  30.462  33.004  35.563  40.771 1.001
## beta[2]  260.237  27.849 204.135 242.078 260.764 279.083 313.628 1.001
## sigma     12.848   1.433  10.411  11.834  12.716  13.717  16.011 1.001
## deviance 343.179   2.822 339.946 341.112 342.448 344.486 350.388 1.001
##          n.eff
## beta[1]  32000
## beta[2]  40000
## sigma    45000
## deviance 45000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 4.0 and DIC = 347.2
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# DIC value
# model.2$BUGSoutput$DIC
# model.1$BUGSoutput$DIC

# picking up beta's and sigma summaries
beta = rep(NA, 2)
names(beta) = c('beta_0', 'beta_1')
beta[1] = model.2$BUGSoutput$summary[,"mean"]["beta[1]"]
beta[2] = model.2$BUGSoutput$summary[,"mean"]["beta[2]"]
beta
```

```
##   beta_0   beta_1
##  33.0571 260.2366
```

```
sigma = model.2$BUGSoutput$summary[,"mean"]["sigma"]
sigma
```

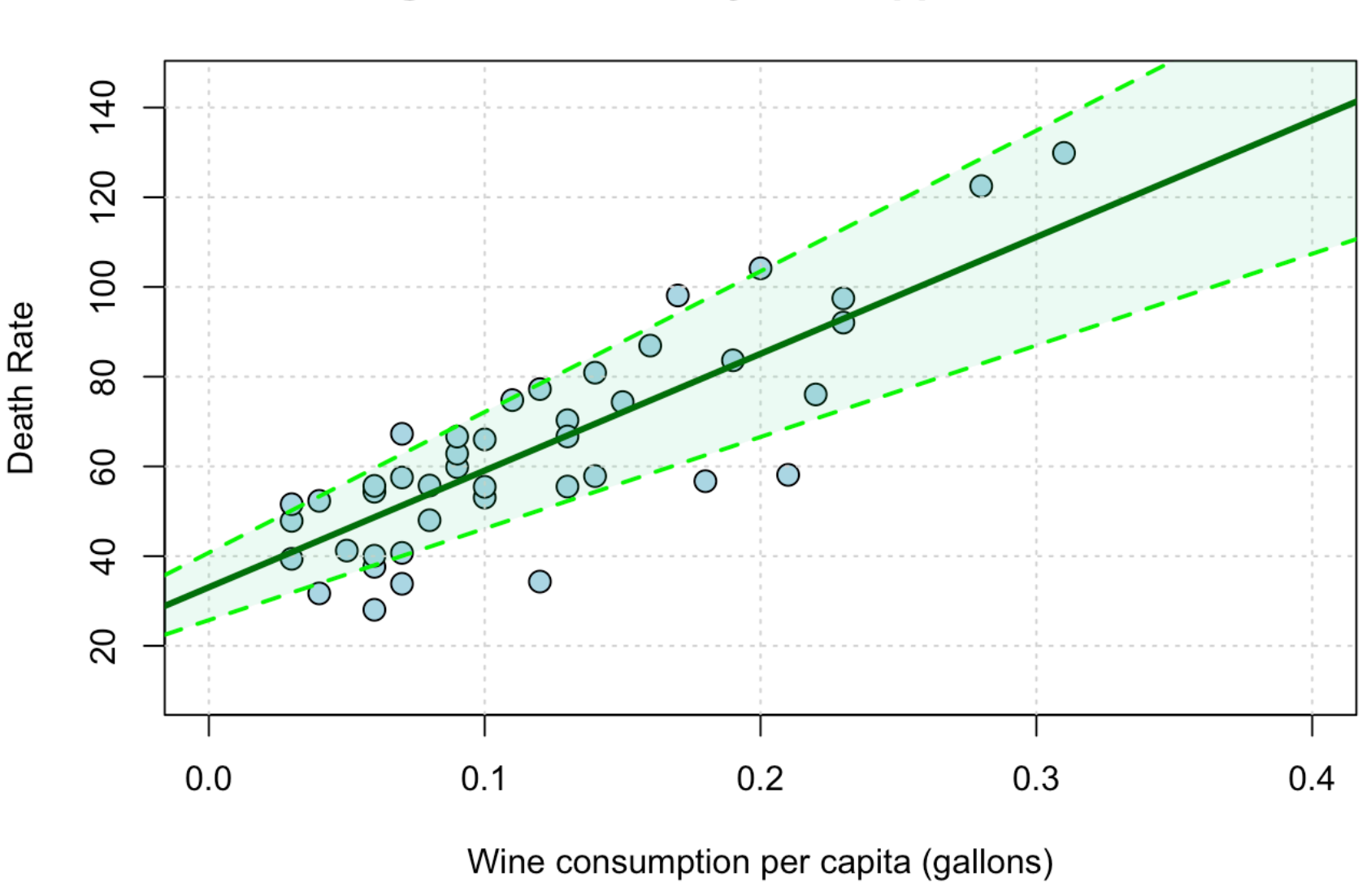
```
##    sigma
## 12.84768
```

```
# RMSE on train set using mean for beta's
y.predict.model.2 = beta[1] + beta[2]*x
rmse.model.2 = Metrics::rmse(y, y.predict.model.2); rmse.model.2
```

```
## [1] 12.65028
```

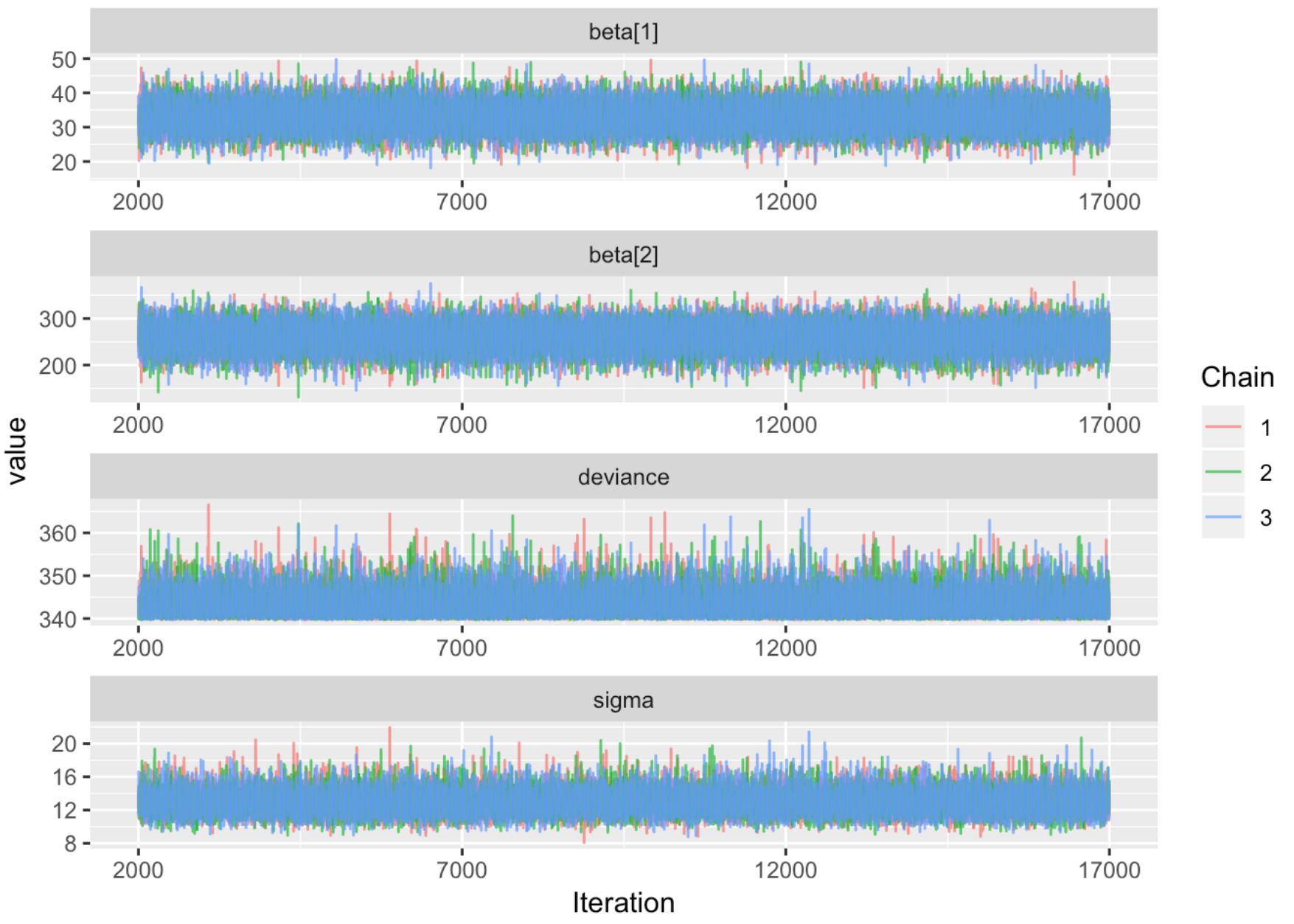
We can plot the regression line estimated with the second model:

```
# Plot the linear regression line
x_seq = seq(min(x)-0.05,max(x)+0.5, 0.01)
y_seq.model.2 = beta[1] + beta[2]*x_seq

plot(x, y, xlim = c(0, 0.4), ylim = c(10,145), cex = 1.4, pch = 21,bg = 'lightblue
',
     xlab = "Wine consumption per capita (gallons)", ylab = "Death Rate",
     main = "Linear regression with Bayesian approach - 2nd model")
grid()

lines(x_seq, y_seq.model.2,col="darkgreen", lwd=3)

beta0.ci = model.2$BUGSoutput$summary["beta[1]", c("2.5%", "97.5%")]
beta1.ci = model.2$BUGSoutput$summary["beta[2]", c("2.5%", "97.5%")]

lower.bounds  <- beta0.ci[[1]] + beta1.ci[[1]] * x_seq
upper.bounds <- beta0.ci[[2]] + beta1.ci[[2]] * x_seq

# plotting the bounds
abline(a = beta0.ci[2], b = beta1.ci[2], lwd = 2, lty = 2, col = "green")
abline(a = beta0.ci[1], b = beta1.ci[1], lwd = 2, lty = 2, col = "green")
polygon(c(x_seq,rev(x_seq)), c(lower.bounds,rev(upper.bounds)),
        col = rgb(.1,.9,.5,.1), border = NA)
```

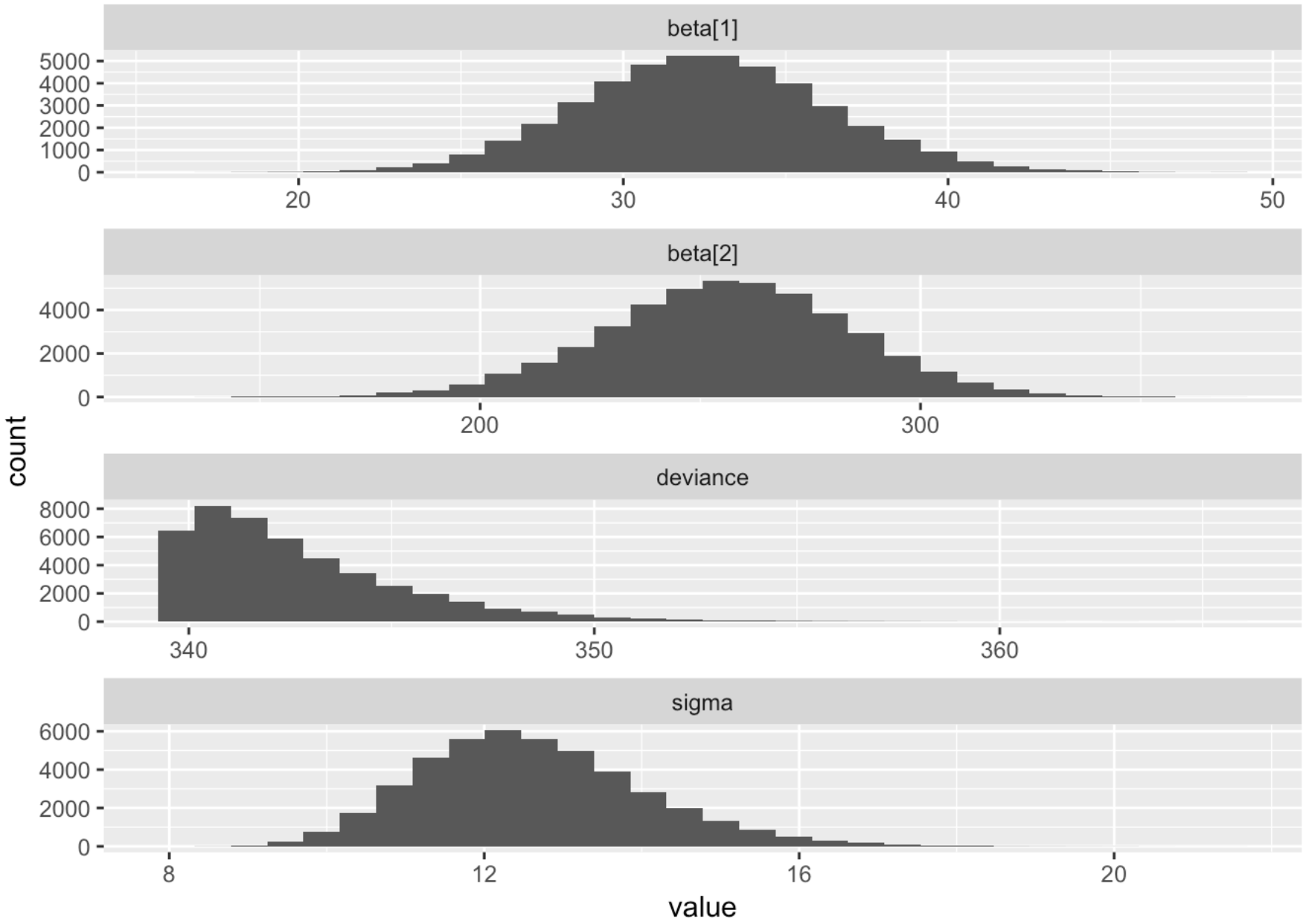


MCMC diagnostic - second model

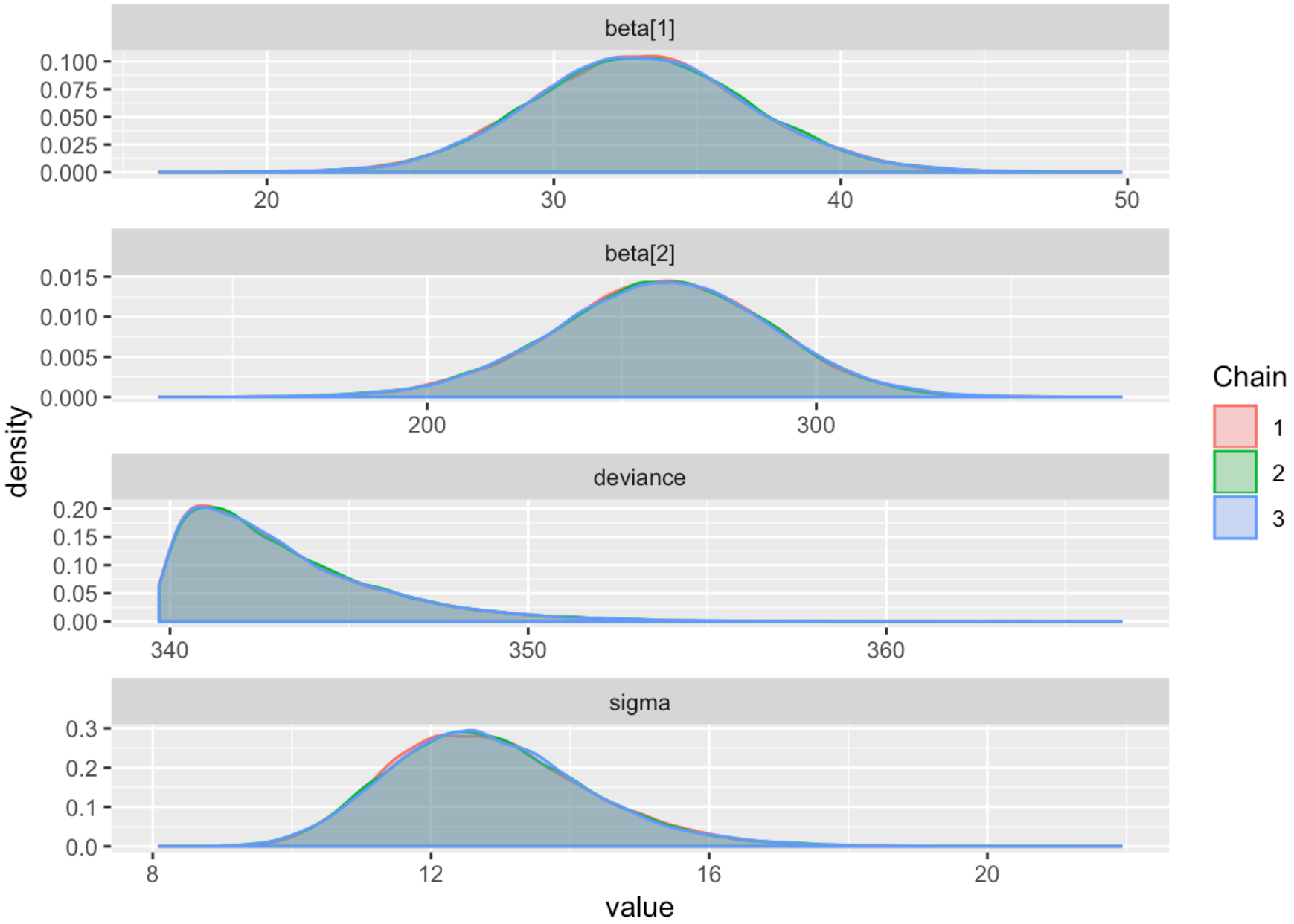
```
# MCMC diagnostic ------------------------------------------------------

library(ggmcmc, quietly = T)


# Create ggs object using ggmcmc library
ggs.mod2 = ggs(as.mcmc(model.2))

ggs_traceplot(ggs.mod2)
```



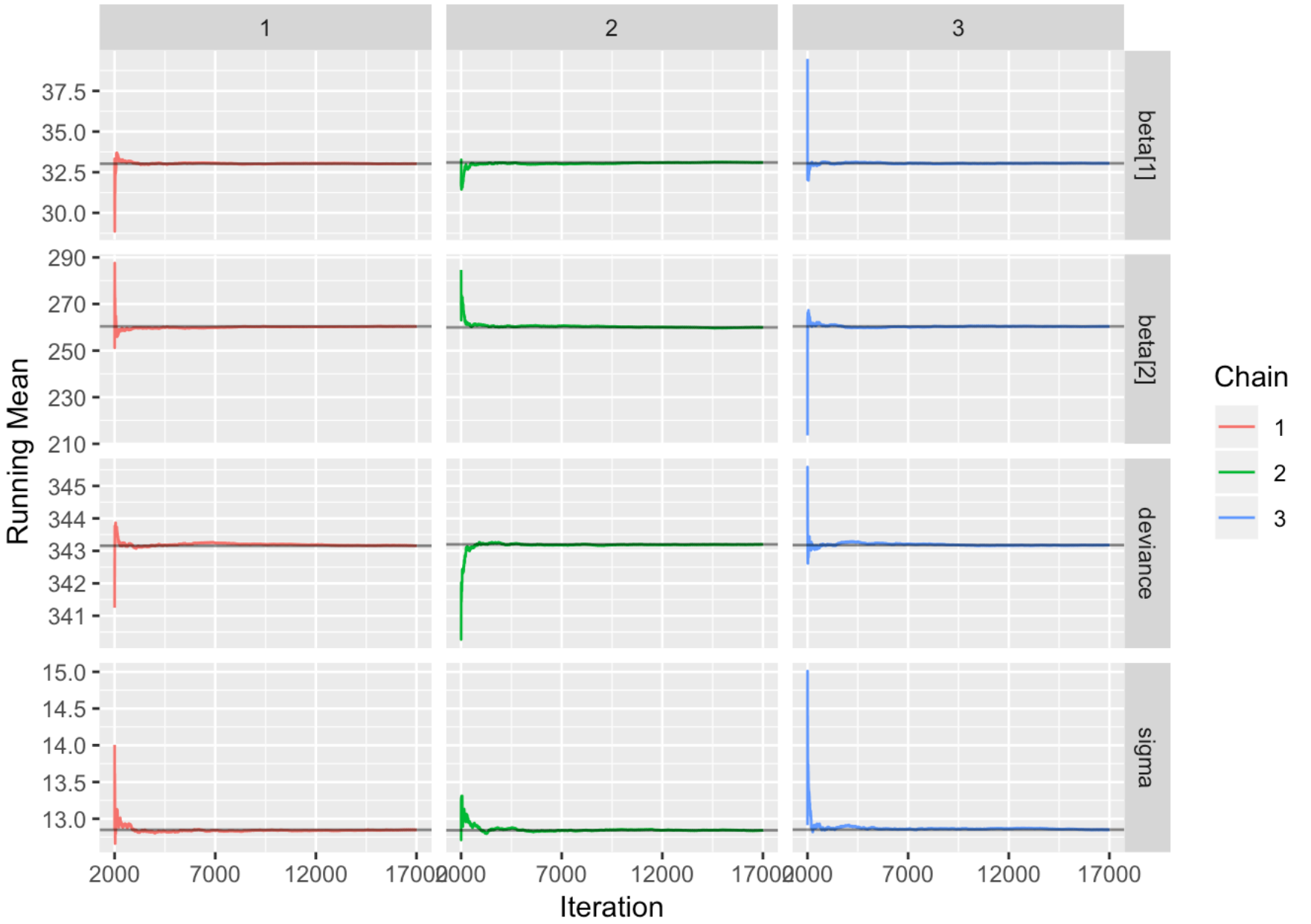

```
# let's see how the values are sampled
ggs_histogram(ggs.mod2)
```



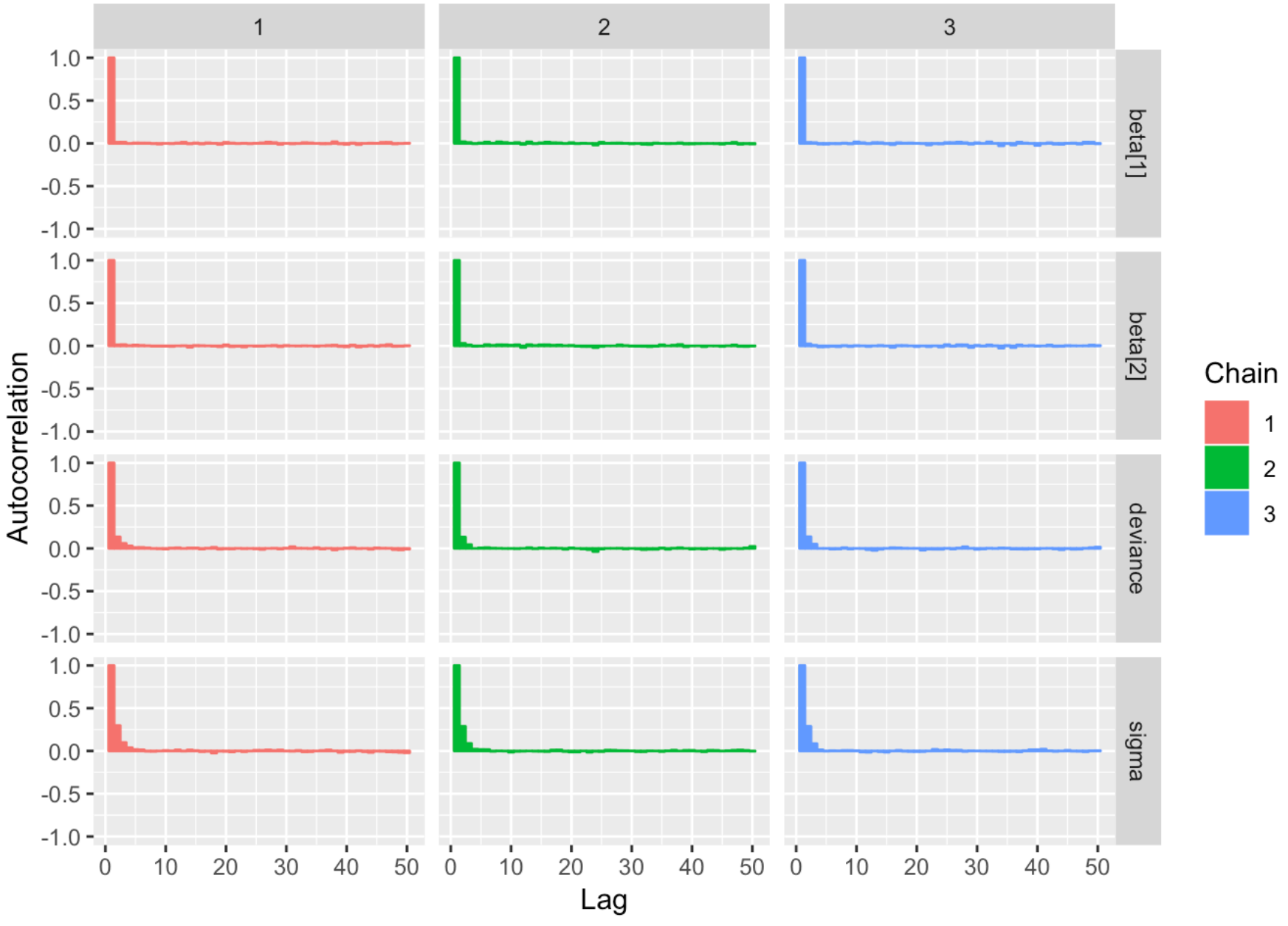

```
# we can see it as densities
ggs_density(ggs.mod2)
```




```
# let's see graphically the behaviour of the empirical averages
# of the parameters with the increase of t = 1,...., 15000
ggs_running(ggs.mod2)
```



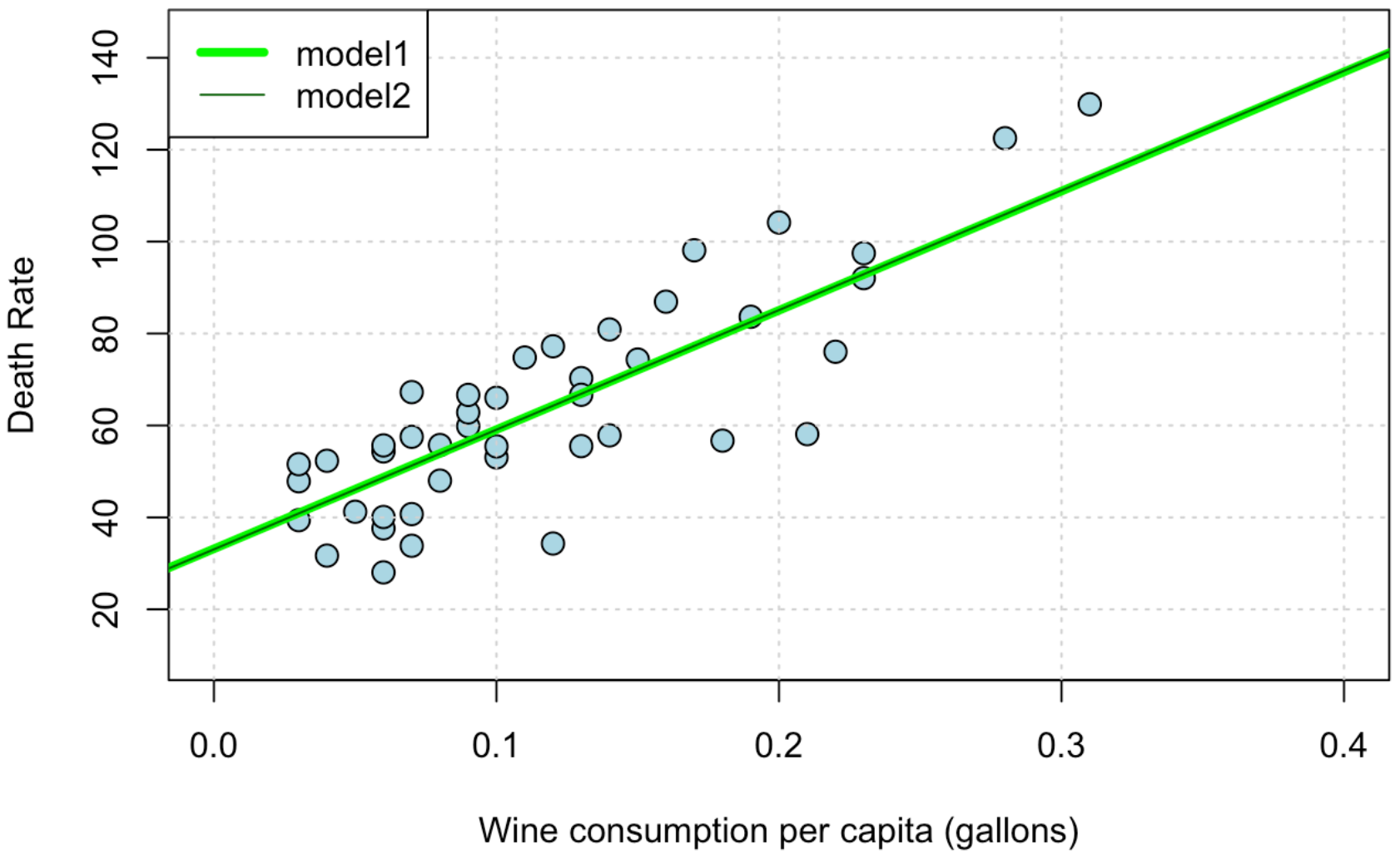

```
# good! our process may have suitable ergodic properties

# let's see the autocorrelation of the chain
ggs_autocorrelation(ggs.mod2)
```



Comparison between the two bayesian models

The models gives basically the same regression line. We can see it visually plotting both the lines:

```
plot(x, y, xlim = c(0, 0.4), ylim = c(10,145), cex = 1.4, pch = 21,bg = 'lightblue
',
     xlab = "Wine consumption per capita (gallons)", ylab = "Death Rate",
     main = "Linear regression with Bayesian approaches")
grid()

lines(x_seq, y_seq.model.1,col="green", lwd=4)
lines(x_seq, y_seq.model.2,col="darkgreen", lwd=1)
legend("topleft", legend = c("model1", "model2"), lwd = c(4,1),
       col = c("green", "darkgreen"))
```



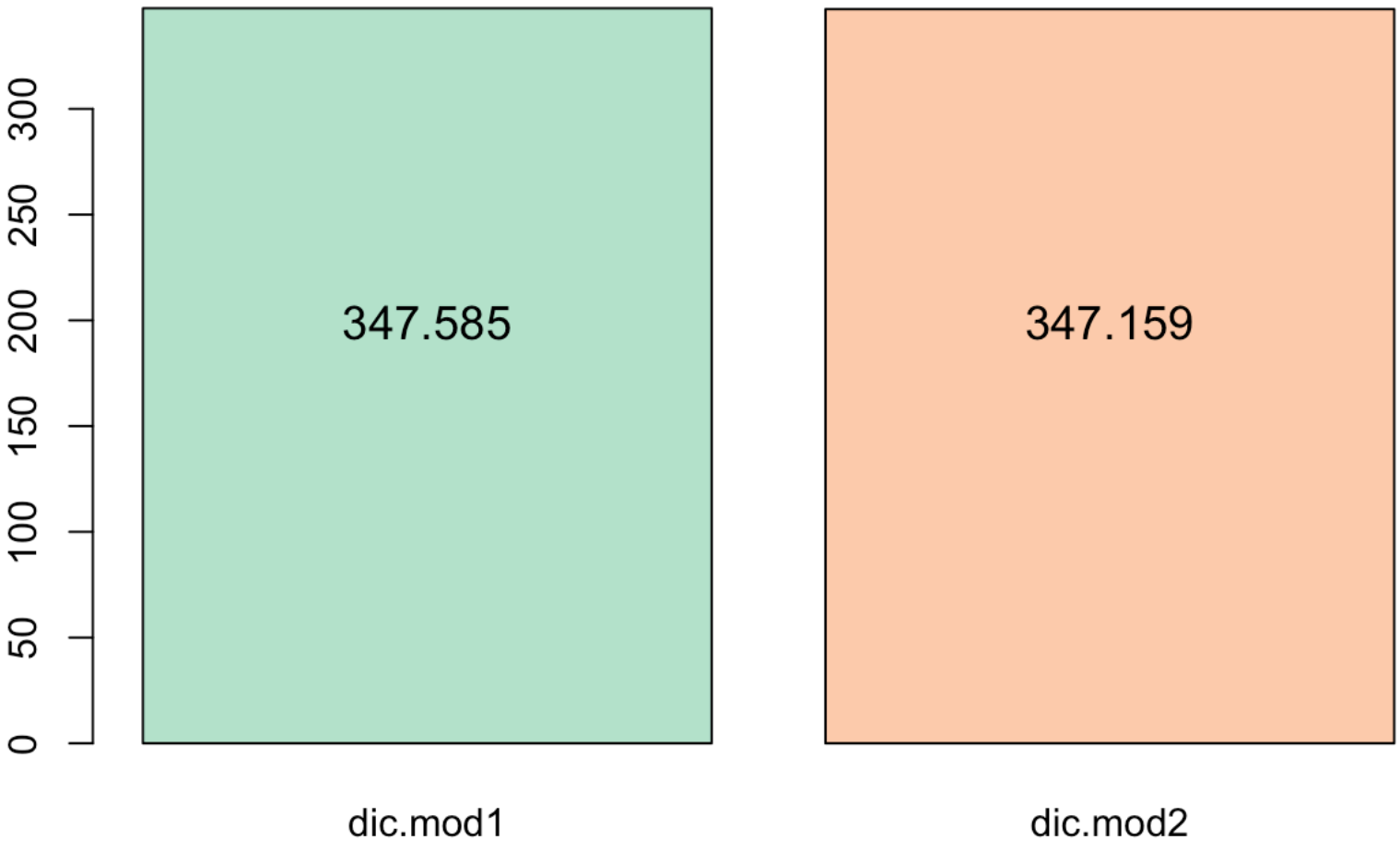
We expected this: in both models we used flat priors (non-informative InvGamma and uniform) for the variance.

Below we show also a comparison between the two bayesian models through **DIC**.

```
# DIC value
res.dic <- c(dic.mod1 = model.1$BUGSoutput$DIC, dic.mod2 = model.2$BUGSoutput$DIC)
res.dic
```

```
## dic.mod1 dic.mod2
## 347.5854 347.1595
```

```
bars <- barplot(res.dic, col = RColorBrewer::brewer.pal(3, "Pastel2"))
text(x = bars, y = 180, label = round(res.dic,3), pos = 3, cex = 1.2, col = "black
")
```

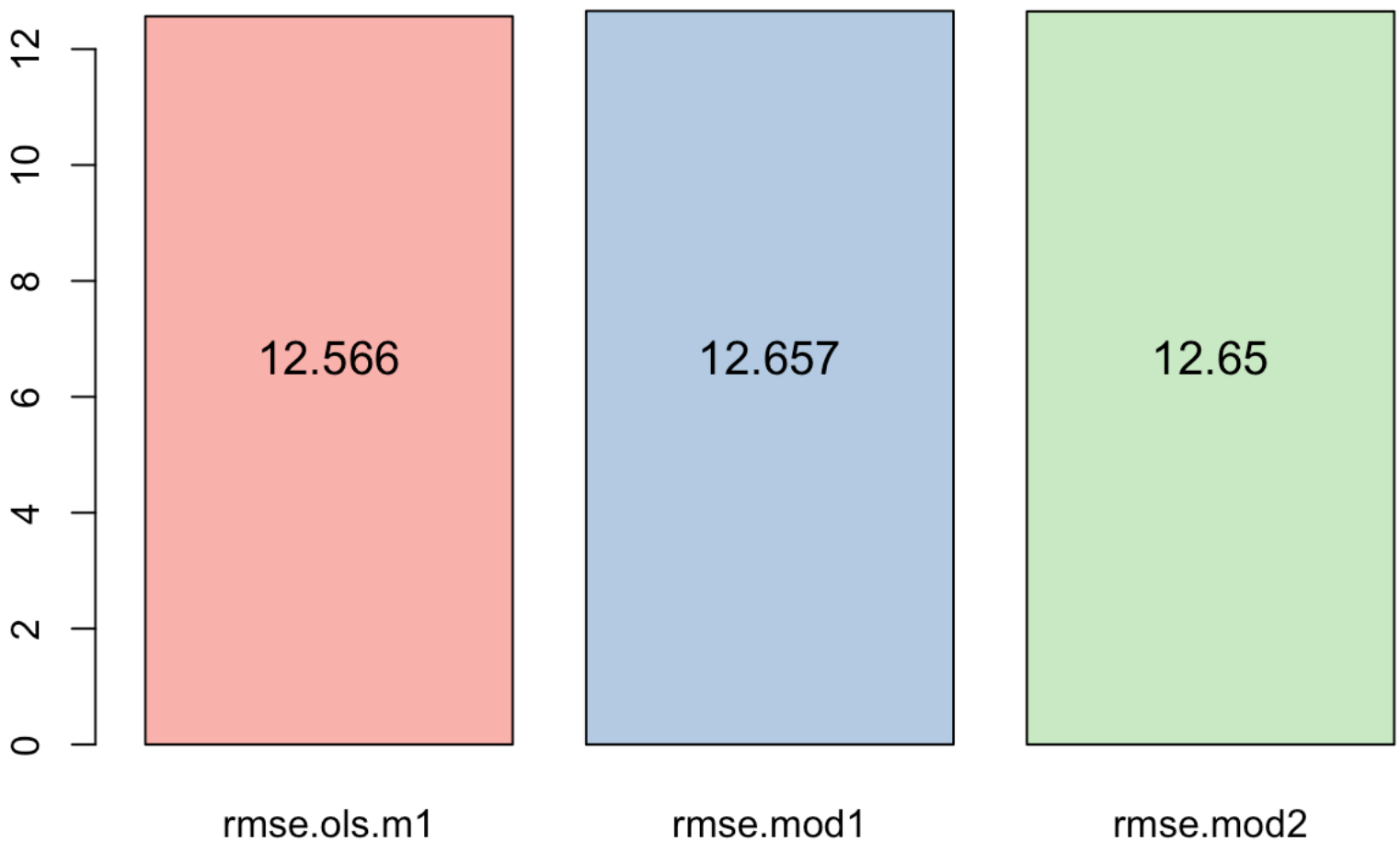


As we can see, the 2 bayesian models give similar DIC values.

Comparison between frequentistic and Bayesian approaches

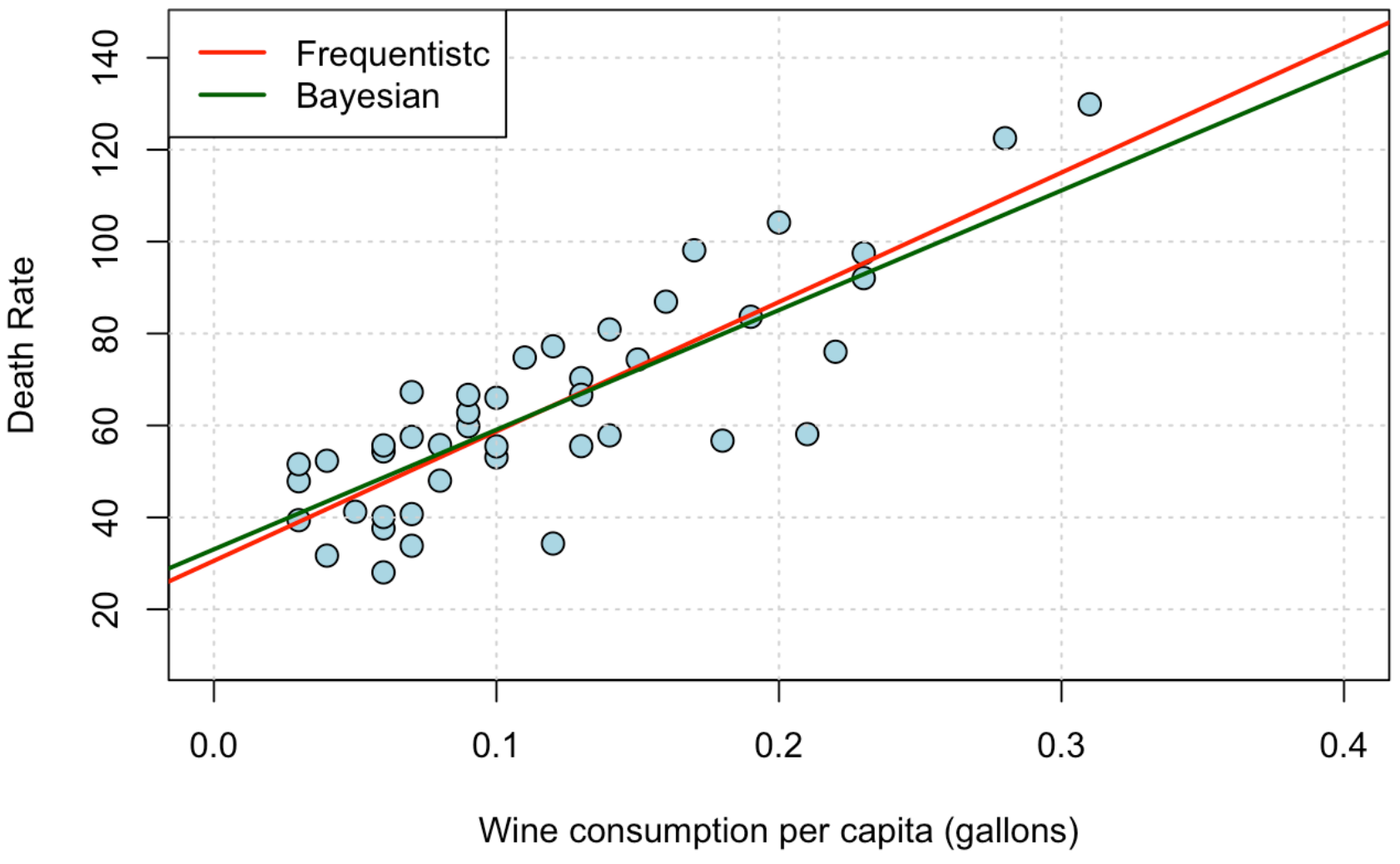
We make the comparison through $RMSE$:

```
res.rmse <- c(rmse.ols.mod1, rmse.mod1 = rmse.model.1, rmse.mod2 = rmse.model.2)

bars <- barplot(res.rmse, col = RColorBrewer::brewer.pal(3, "Pastel1"))
text(x = bars, y = 6, label = round(res.rmse,3), pos = 3, cex = 1.2, col = "black"
)
```



It seems that the models, in both frequentistic and bayesian approaches, give us similar results in terms of $RMSE$. The frequentistic model have a slighly lower $RMSE$. We can also see here the two regression lines by using the scatterplot:

```
plot(x, y, xlim = c(0, 0.4), ylim = c(10,145), cex = 1.4, pch = 21,bg = 'lightblue
',
     xlab = "Wine consumption per capita (gallons)", ylab = "Death Rate",
     main = "Linear regression with different approaches")
grid()

lines(x_seq, y_seq.ols,col="red", lwd=2)
lines(x_seq, y_seq.model.2,col="darkgreen", lwd=2)
legend("topleft", legend = c("Frequentistc", "Bayesian"), lwd = c(2,2),
       col = c("red", "darkgreen"))
```



2. Linear model with 2 explanatory variables (Wine and Beer)

In this case we're handle with the following model:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Where x_1 and x_2 are referred to Wine and Beer respectively.

As already done for the first model, denoting $\boldsymbol{y} = (y_1, \dots, y_n)$, $\boldsymbol{x_1} = (x_{1,1}, \dots, x_{n,1})$, $\boldsymbol{x_2} = (x_{1,2}, \dots, x_{n,2})$, we can derive the likelihood being proportional to:

$$\begin{aligned} \mathcal{L}_y(\boldsymbol{x_1}, \boldsymbol{x_2}, \beta_0, \beta_1, \sigma^2) &= f(\boldsymbol{y}|\boldsymbol{x_1}, \boldsymbol{x_2}, \beta_0, \beta_1, \sigma^2) \\ &\propto \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 \cdot x_{i,1} + \beta_2 \cdot x_{i,2})\right)^2 \right\} \end{aligned}$$

Frequentistic approach - OLS

We perform linear regression through *OLS* in the same way as before, using *lm()* function. Note that in this case we're not fitting a line but a plane (since we're working in 3 dimensions).

```
library(plotly) # for 3d plot
```

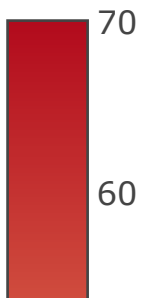
```r
# estimating the coefficients with ols
ols.model.2 <- lm(DeathRate ~ Wine + Beer , data = df)

betas <- ols.model.2$coefficients
names(betas) <- c('beta0', 'beta1', 'beta2')
betas
```

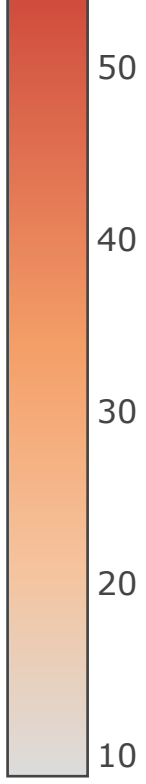
```
##     beta0     beta1     beta2
##  23.43328 252.78108  16.77246
```

```r
# now we define a function to make 3d plots
make.3d.plot <- function(betas){
  # creating sequence of x1 and x2 for the plane
  x1.seq <- seq(min(df$Wine), max(df$Wine), length.out = 30)
  x2.seq <- seq(min(df$Beer), max(df$Beer), length.out = 30)

  # expand them as a grid and make a dataframe
  grid <- expand.grid(x1.seq, x2.seq)
  d <- setNames(data.frame(grid), c("Wine", "Beer"))

  # make the regression plane with the estimated coefficients
  DeathRate <- betas[[1]] + betas[[2]] * d$Wine + betas[3] * d$Beer
  z.seq <- matrix(DeathRate, nrow = length(d$Wine), ncol = length(d$Beer))


  # Markers and plane
  p <- plot_ly(data = df, z = ~DeathRate, x = ~Wine, y = ~Beer, opacity = 0.8,
               marker = list(color = ~Urbanism,
                             colorscale = c('#FFE1A1', '#683531'), showscale = TRU
E)) %>%
    add_markers()

  p <- p %>%
    add_surface(z = z.seq, x = x1.seq, y = x2.seq, showscale = FALSE) %>%
    layout(showlegend = F)

  return(p)
}
```

We can now make a 3d plot, showing the data points and the plane that we've fitted.

```r
# call the function we created
make.3d.plot(betas)
```






```
# compute rmse for the model2
y.test = df$DeathRate
y.predict.ols = betas[[1]] + betas[[2]]* df$Wine + betas[[3]]*df$Beer
rmse.ols.mod2 = c(rmse.ols.m2 = Metrics::rmse(y.test, y.predict.ols))
rmse.ols.mod2
```

```
## rmse.ols.m2
##    12.10602
```

Bayesian Analysis: variance with InverseGamma distribution

In this new model we simply add a new variable (with its own coefficient) to the linear combination which describes the mean of y_i :

$$\mu_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2}$$

performing again a non-conjugate bayesian analysis, we define the priors as following:

$$\begin{aligned}
\beta_0 &\sim N(0, 10^{-4}) \\
\beta_1 &\sim N(0, 10^{-4}) \\
\beta_2 &\sim N(0, 10^{-4}) \\
\tau^2 &\sim Gamma(10^{-3}, 10^{-3})
\end{aligned}$$

The full conditionals can be computed in the same way as part 1.

Now we can build and run our model in RJAGS:

```
#####  third bayesian model

# linear mu, sigma with inverse gamma prior (tau ~ gamma prior)

y  = df$DeathRate
x1 = df$Wine
x2 = df$Beer
n  = nrow(df)

data = list( Y = y,
             x1 = x1,
             x2 = x2,
             N = n)


third.model = function(){
  for(i in 1:N){
    # likelihood
    Y[i] ~ dnorm(mu[i], tau)
    mu[i] = beta[1] + beta[2]*x1[i] + beta[3]*x2[i]
  }

  # std normal priors for beta's
  beta[1] ~ dnorm(0.0, 1.0E-4)
  beta[2] ~ dnorm(0.0, 1.0E-4)
  beta[3] ~ dnorm(0.0, 1.0E-4)

  # prior for sigma (using precision tau = 1/sigma)
  tau ~ dgamma(0.001, 0.001)
  sigma = sqrt(1.0 / tau) # precision
}

# jags parameters
params = c("beta", "sigma")

## inits for the model
inits.mod3 = function(){
  list("beta" = rnorm(3, mean = 0.0, sd = 10.0),
       "tau" = rgamma(1, 1.0, 1.0))
}


# run the model:
# 3 chains
# 15000 iterations (burn in of 2000)
model.3 = jags(data=data, inits.mod3, params, n.chains=3,
               n.iter=15000, n.burnin=2000, n.thin = 1,
               model.file=third.model, DIC=TRUE)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 43
##    Unobserved stochastic nodes: 4
##    Total graph size: 241
##
## Initializing model
```

```
model.3
```

```
## Inference for Bugs model at "/var/folders/s2/9gqx63yx2jsb9hg77fr834mw0000gn/T//
Rtmp3zcQs5/model136a6a9f4bfc.txt", fit using jags,
##  3 chains, each with 15000 iterations (first 2000 discarded)
##  n.sims = 39000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
## beta[1]   24.058   5.682  12.809  20.310  24.062  27.859  35.246 1.001
## beta[2]  228.024  32.126 162.951 207.050 228.638 249.581 289.728 1.001
## beta[3]   20.407   9.649   1.634  13.987  20.241  26.803  39.746 1.001
## sigma     12.870   1.496  10.357  11.815  12.726  13.765  16.201 1.001
## deviance 341.170   3.286 337.067 338.746 340.427 342.791 349.372 1.001
##          n.eff
## beta[1]  39000
## beta[2]  39000
## beta[3]  39000
## sigma    39000
## deviance 39000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 5.4 and DIC = 346.6
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# DIC value
model.3$BUGSoutput$DIC
```

```
## [1] 346.5691
```

```
# picking up beta's and sigma summaries
beta = rep(NA, 3)
names(beta) = c('beta_0', 'beta_1', 'beta_2')
beta[1] = model.3$BUGSoutput$summary[,"mean"]["beta[1]"]
beta[2] = model.3$BUGSoutput$summary[,"mean"]["beta[2]"]
beta[3] = model.3$BUGSoutput$summary[,"mean"]["beta[3]"]
beta
```

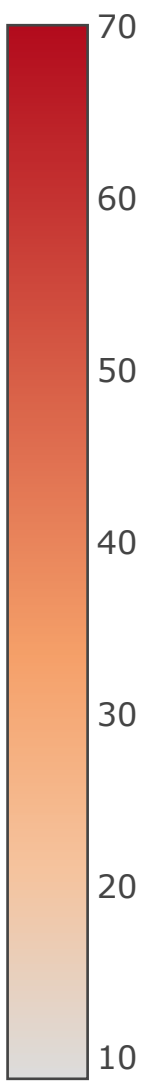
```
##    beta_0    beta_1    beta_2
##  24.05785 228.02366  20.40689
```

```
sigma = model.3$BUGSoutput$summary[,"mean"]["sigma"]
sigma
```

```
##    sigma
## 12.87029
```

Making a plot also here using our function:

```
make.3d.plot(beta)
```



Updating RMSE and DIC results:

```
y.test = df$DeathRate

y.predict.model.3 = beta[[1]] + beta[[2]]* df$Wine + beta[[3]]*df$Beer
rmse.model.3 = c(rmse.mod3 = Metrics::rmse(y.test, y.predict.model.3))
rmse.model.3
```

```
## rmse.mod3
##  12.19433
```

```
# adding RMSE results for model 2
res.rmse <- c(res.rmse, rmse.ols.mod2, rmse.model.3)

# DIC value
res.dic <- c(res.dic,  dic.mod3 = model.3$BUGSoutput$DIC)
res.dic
```

```
## dic.mod1 dic.mod2 dic.mod3
## 347.5854 347.1595 346.5691
```

3. Linear model with all the explanatory variables (Wine, Beer, Spirits, Urbanism)

With this final model we're using all the variables that we have in the dataset. The model is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$$

Where x_i 's are referred respectively to: Wine, Beer, Spirits, Urbanism variables.

The likelihood is proportional to:

$$\mathcal{L}_{y_1,\dots,y_n}(\boldsymbol{x_1},\dots,\boldsymbol{x_n},\boldsymbol{\beta},\sigma^2) = f(y_1,\dots,y_n\,|\boldsymbol{x_1},\dots,\boldsymbol{x_n},\boldsymbol{\beta},\sigma^2)$$
$$\propto \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - (\beta_0 + \boldsymbol{\beta^T x_i})\right)^2\right\}$$

where $\boldsymbol{x_i} = (x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}, x_{i,5})$.

In order not to dwell too much with the project (and since we are confident about similar results between frequentistic and bayesian approaches), in this case we'll perform only the bayesian analysis.

```
# removing outliers
#idx <- which(df$Spirits > 10)
#df[idx, ]
#new.df <- df[-idx,]

# fourth model
# linear mu, sigma with inverse gamma prior (tau ~ gamma prior)
y  = df$DeathRate
x1 = df$Wine
x2 = df$Beer
x3 = df$Spirits
x4 = df$Urbanism
n  = nrow(df)

data = list( Y = y,
             x1 = x1,
             x2 = x2,
```

```
              x3 = x3,
              x4 = x4,
              N = n)

# third model
fourth.model = function() {
  for(i in 1:N){
    # likelihood
    Y[i] ~ dnorm(mu[i], tau)
    mu[i] = beta[1] + beta[2]*x1[i] + beta[3]*x2[i] + beta[4]*x3[i] +beta[5]*x4[i]
  }

  for(i in 1:5){
    # std normal priors for beta's
    beta[i] ~ dnorm(0.0, 1.0E-4)
  }

  # prior for sigma (using precision tau = 1/sigma)
  tau ~ dgamma(0.001, 0.001)
  sigma = sqrt(1.0 / tau) # precision
}

# jags parameters
params = c("beta", "sigma")

## inits for the model
inits.mod4 = function(){
  list("beta" = rnorm(5, mean = 0.0, sd = 10.0),
       "tau" = rgamma(1, 1.0, 1.0))
}


# run the model:
# 3 chains
# 10000 iterations (burn in of 2000)
model.4 = jags(data=data, inits.mod4, params, n.chains=3,
               n.iter=15000, n.burnin=2000, n.thin = 1,
               model.file=fourth.model, DIC=TRUE)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 43
##    Unobserved stochastic nodes: 6
##    Total graph size: 404
##
## Initializing model
```

```
model.4
```

```
## Inference for Bugs model at "/var/folders/s2/9gqx63yx2jsb9hg77fr834mw0000gn/T//
Rtmp3zcQs5/model136a4c871066.txt", fit using jags,
##  3 chains, each with 15000 iterations (first 2000 discarded)
##  n.sims = 39000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
## beta[1]   14.939   6.090   2.945  10.909  14.973  19.022  26.861 1.001
## beta[2]  151.956  38.892  73.549 126.488 152.402 178.065 227.119 1.001
## beta[3]   -1.908  11.326 -24.349  -9.332  -1.894   5.573  20.513 1.001
## beta[4]   31.139  12.161   7.581  23.072  31.069  39.188  55.458 1.001
## beta[5]    0.448   0.172   0.110   0.334   0.449   0.562   0.787 1.001
## sigma     12.014   1.418   9.589  11.019  11.890  12.875  15.129 1.001
## deviance 335.257   3.839 329.957 332.433 334.548 337.306 344.588 1.001
##          n.eff
## beta[1]  39000
## beta[2]  19000
## beta[3]  39000
## beta[4]  39000
## beta[5]  39000
## sigma    29000
## deviance  9700
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 7.4 and DIC = 342.6
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# DIC value
model.4$BUGSoutput$DIC
```

```
## [1] 342.6265
```

```
# picking up beta's and sigma summaries
beta = rep(NA, 5)
names(beta) = c('beta_0', 'beta_1', 'beta_2', 'beta_3', 'beta_4')
beta[1] = model.4$BUGSoutput$summary[,"mean"]["beta[1]"]
beta[2] = model.4$BUGSoutput$summary[,"mean"]["beta[2]"]
beta[3] = model.4$BUGSoutput$summary[,"mean"]["beta[3]"]
beta[4] = model.4$BUGSoutput$summary[,"mean"]["beta[4]"]
beta[5] = model.4$BUGSoutput$summary[,"mean"]["beta[5]"]
beta
```

```
##      beta_0      beta_1      beta_2      beta_3      beta_4
##  14.9392465 151.9557620  -1.9084000  31.1389826   0.4482452
```

```
sigma = model.4$BUGSoutput$summary[,"mean"]["sigma"]
sigma
```

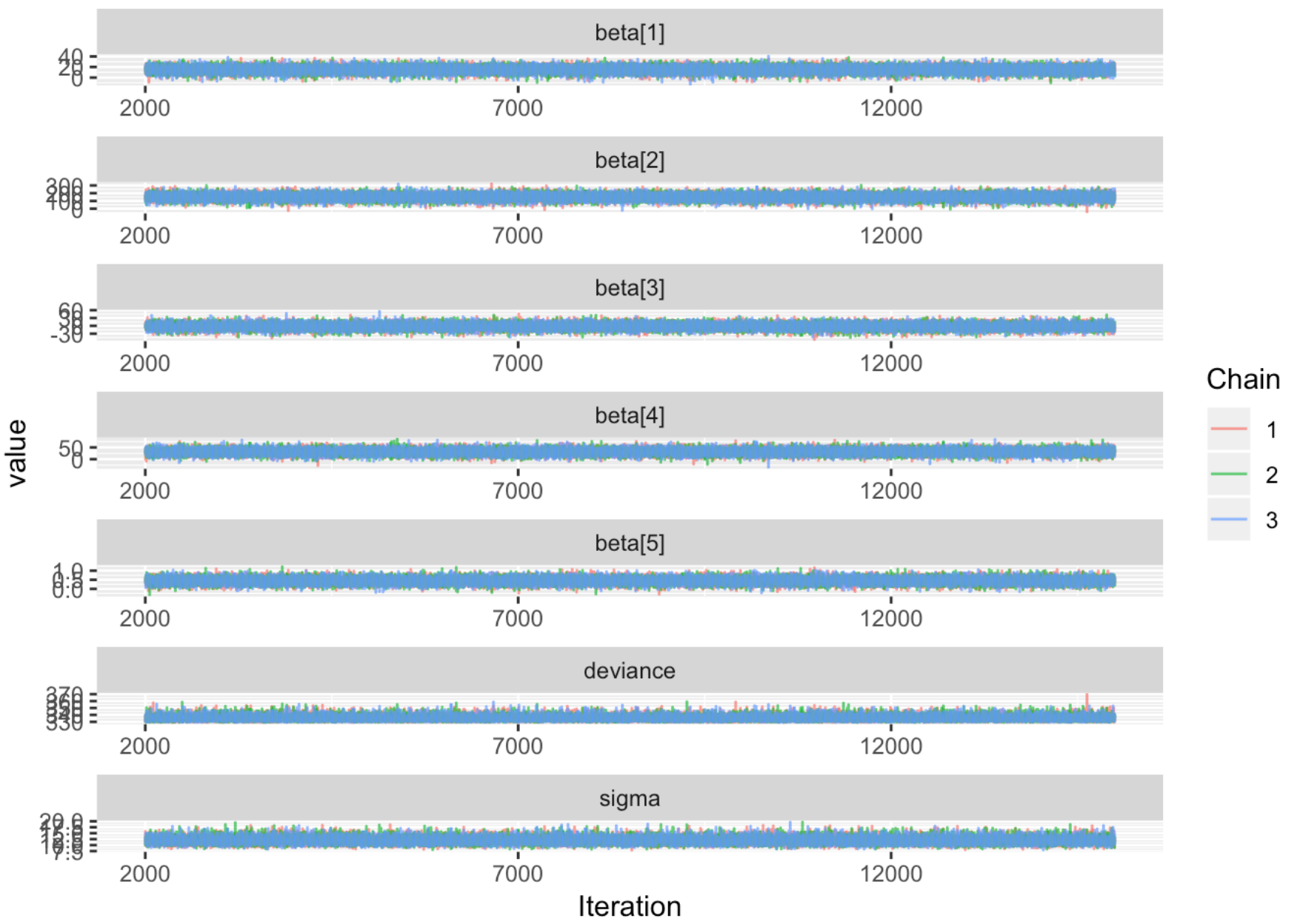
```
##    sigma
## 12.01423
```

```
# RMSE on new.df set using mean for beta's
y.test = df$DeathRate
y.predict.model.4 = beta[1] + beta[2]*df$Wine + beta[3]*df$Beer + beta[4]*df$Spiri
ts +beta[5]*df$Urbanism
rmse.model.4 = Metrics::rmse(y.test, y.predict.model.4); rmse.model.4
```

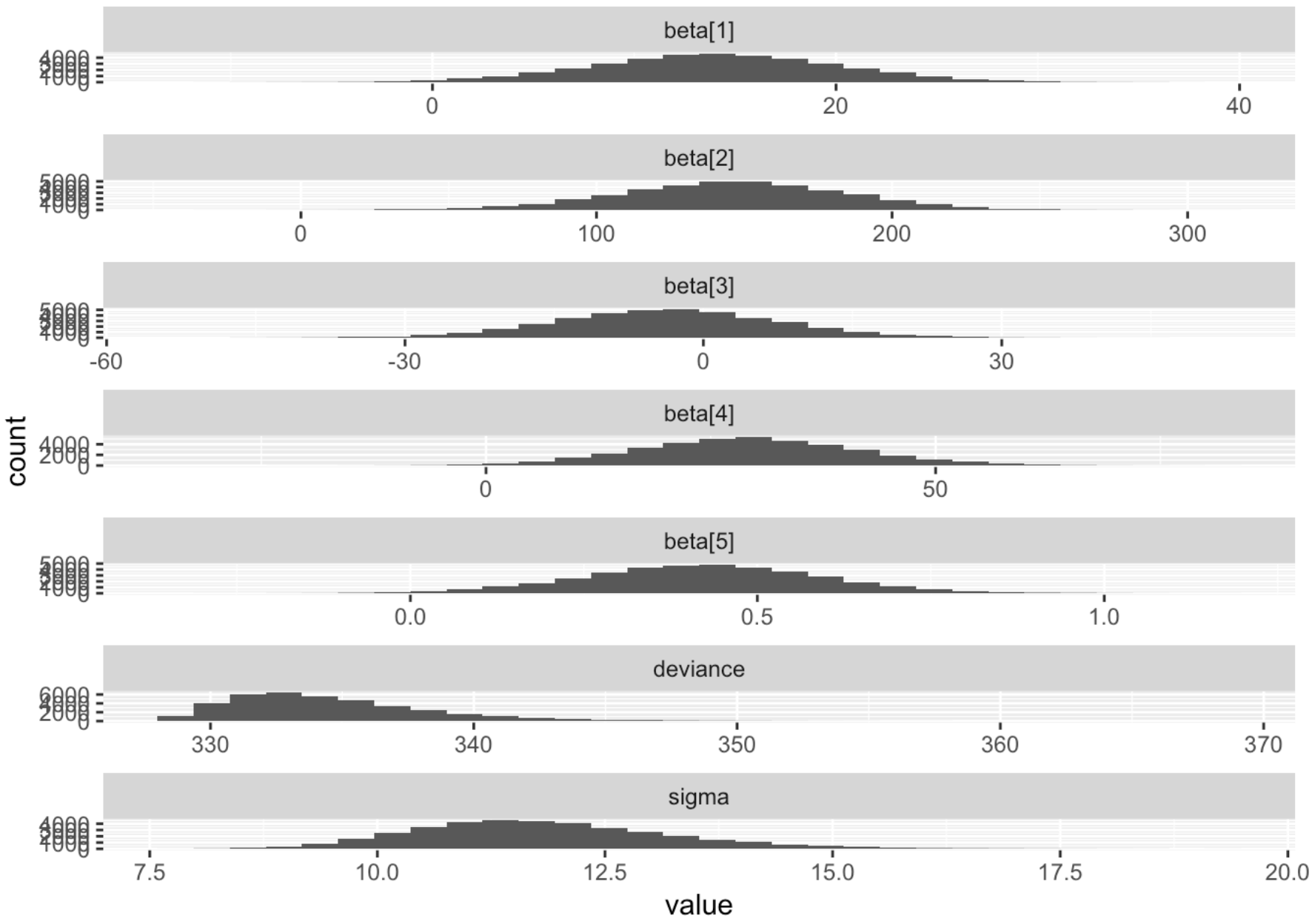
```
## [1] 11.09715
```

```
# updating rmse's vector
res.rmse <- c(res.rmse, rmse.mod4 = rmse.model.4)

# updating DIC's vector
res.dic <- c(res.dic, dic.mod4 = model.4$BUGSoutput$DIC)
```

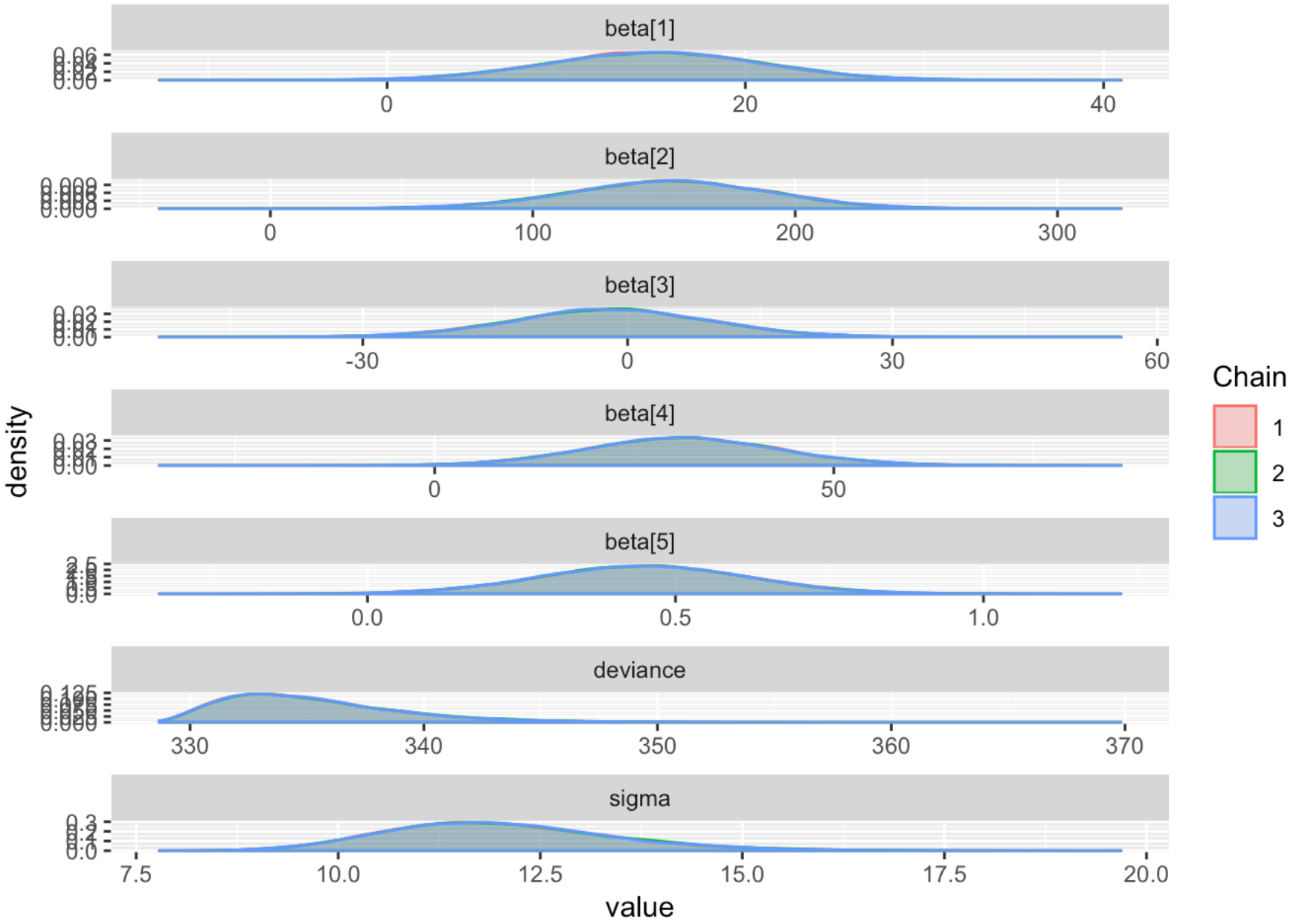
```
# Create ggs object using ggmcmc library
ggs.mod4 = ggs(as.mcmc(model.4))

ggs_traceplot(ggs.mod4)
```



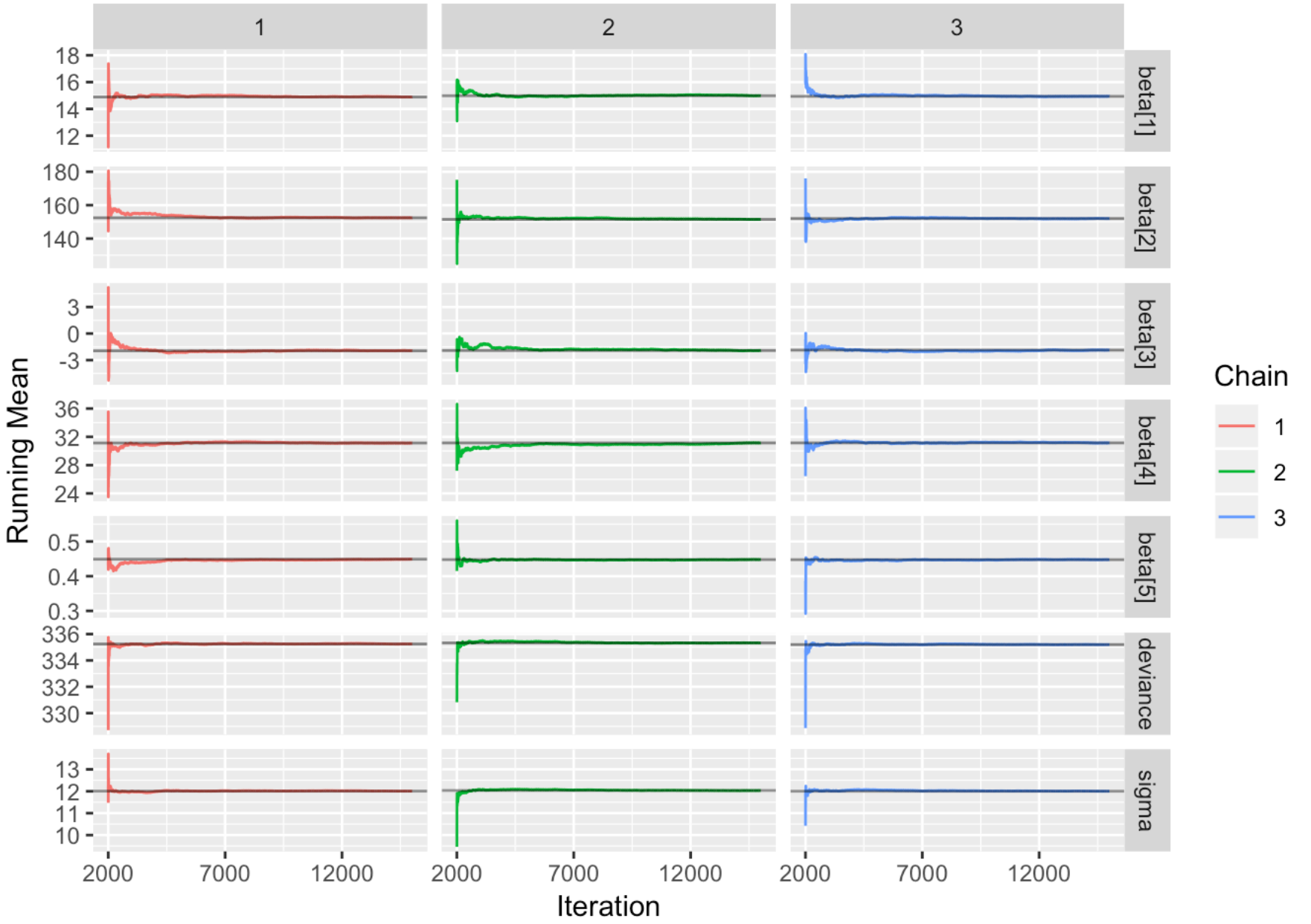
```
# let's see how the values are sampled
ggs_histogram(ggs.mod4)
```



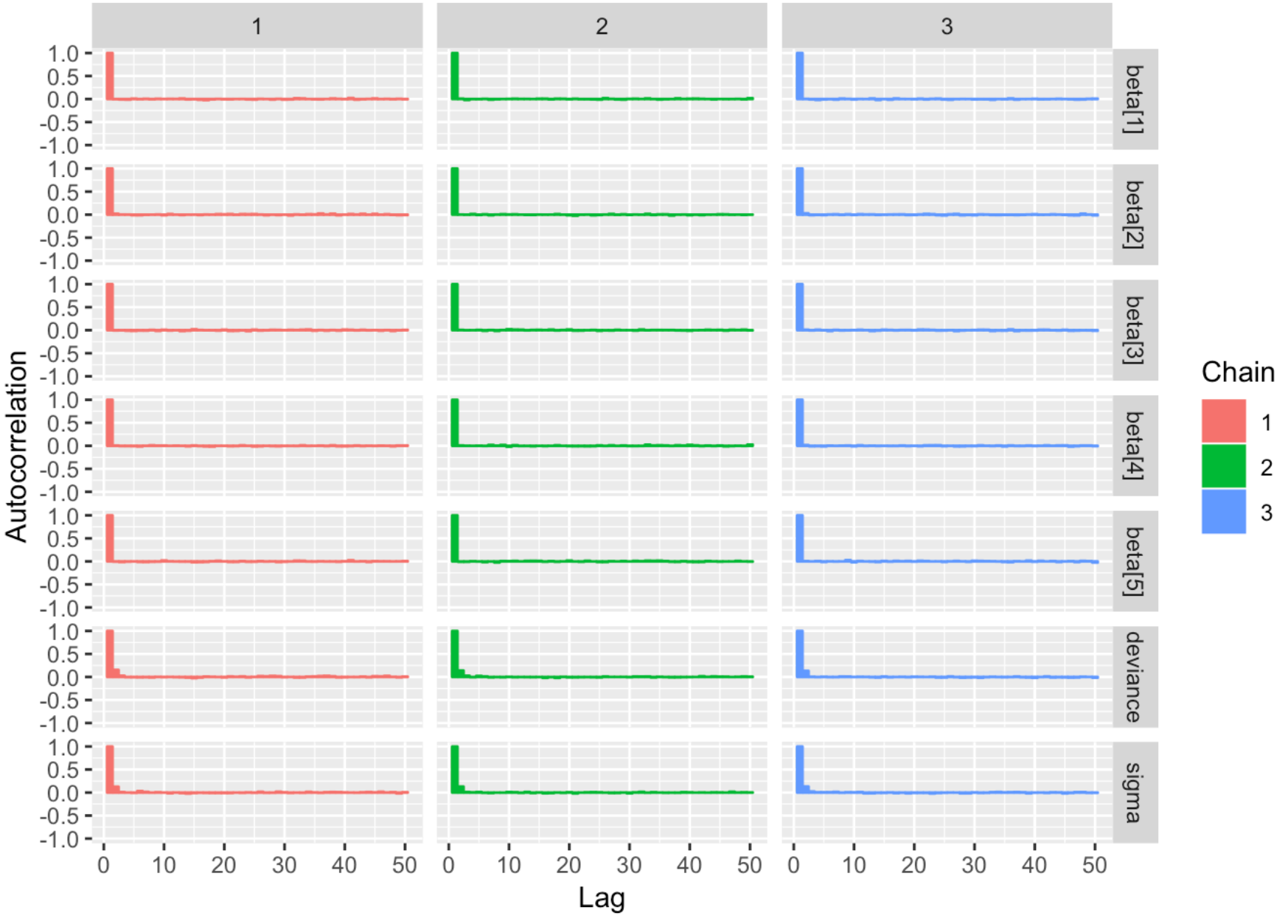

```
# we can see it as densities
ggs_density(ggs.mod4)
```




```
# let's see graphically the behaviour of the empirical averages
# of the parameters with the increase of t = 1,...., 15000
ggs_running(ggs.mod4)
```



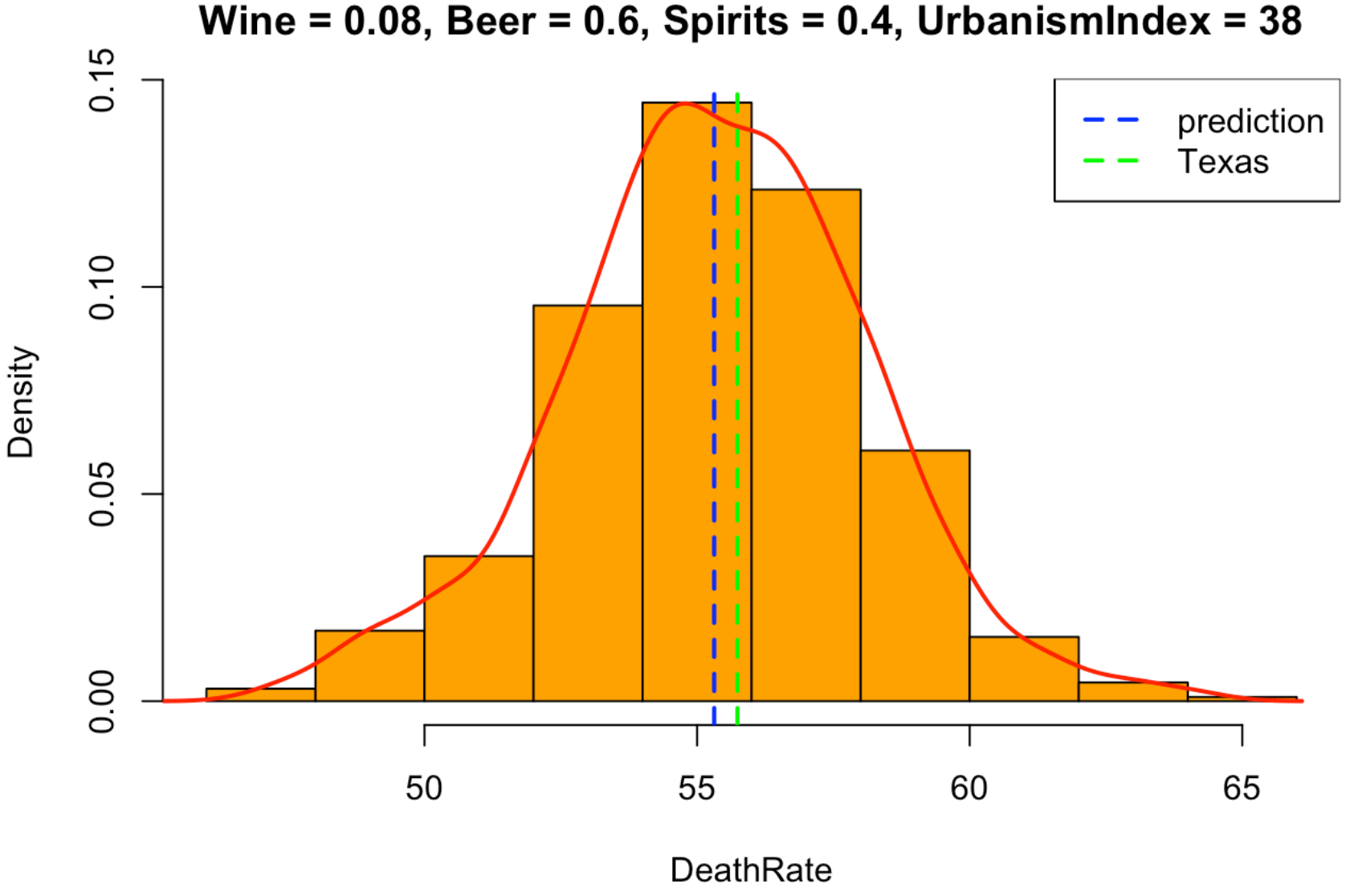

```
# good! our process may have suitable ergodic properties

# let's see the autocorrelation of the chain
ggs_autocorrelation(ggs.mod4)
```



Plus: Approximation of the posterior predictive distribution of death rate per million for alcoholic consumptions (pro capita) of:

- 0.08 gallons of Wine
- 0.6 gallons of Beer
- 0.4 gallons of Spirits
- Index of Urbanism: 38

```
set.seed(1869097)
simsize <- 1000
alcohol_cons <- rep(NA, simsize)
XX <- model.4$BUGSoutput$sims.list

for(t in (1:simsize)){
  mu.temp <- XX$beta[t,1]+ XX$beta[t,2]*0.08 +
    XX$beta[t,3]*0.6 + XX$beta[t,4]*0.4 + XX$beta[t,5]*38
  sd.temp <- sqrt(1/XX$sigma[t])
  alcohol_cons[t] <- rnorm(1, mu.temp, sd.temp)
}


hist(alcohol_cons, probability = T, col = "orange", xlab = "DeathRate", main ="")
title(main="Approximation of the posterior predictive distribution \n Wine = 0.08,
Beer = 0.6, Spirits = 0.4, UrbanismIndex = 38 ")
abline(v = mean(alcohol_cons), col = "blue", lwd = 2, lty = 2)
abline(v = df[which(df$State == "Texas"), "DeathRate"], col = "green", lwd = 2, lt
y = 2)
lines(density(alcohol_cons), col = "red", lwd= 2)
legend("topright", legend = c("prediction", "Texas"),
       col = c("blue", "green"), lwd = 2, lty = 2)
```




```
df[which(df$State == "Texas"),]
```

```
##    State Spirits Wine Beer Total DeathRate Urbanism
## 45 Texas     0.4 0.08 0.61  1.09     55.74     37.4
```

In this case, Texas has similar features we've choosen for our approximation (Wine, Beer, Spirits, Urbanism) and its DeathRate falls into our approximation of the posterior predictive distribution.

The prediction of DeathRate into a state with alcohol consumption of 0.08 (Wine), 0.6 (Beer), 0.4 (Spirit) gallons and with index of urbanism of 38 is:

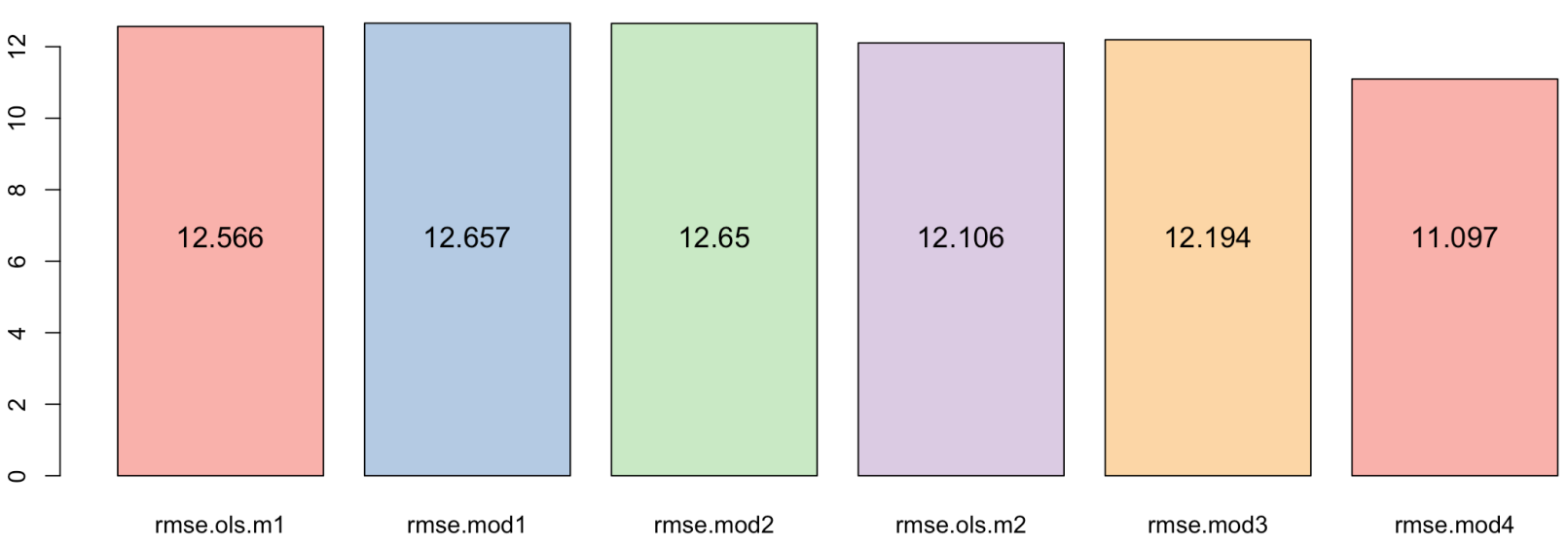
```
mean(alcohol_cons)
```

```
## [1] 55.31385
```

Comparisons between all the models

RMSE

```
bars <- barplot(res.rmse, col = RColorBrewer::brewer.pal(5, "Pastel1"))
text(x = bars, y = 6, label = round(res.rmse,3), pos = 3, cex = 1.2, col = "black"
)
```



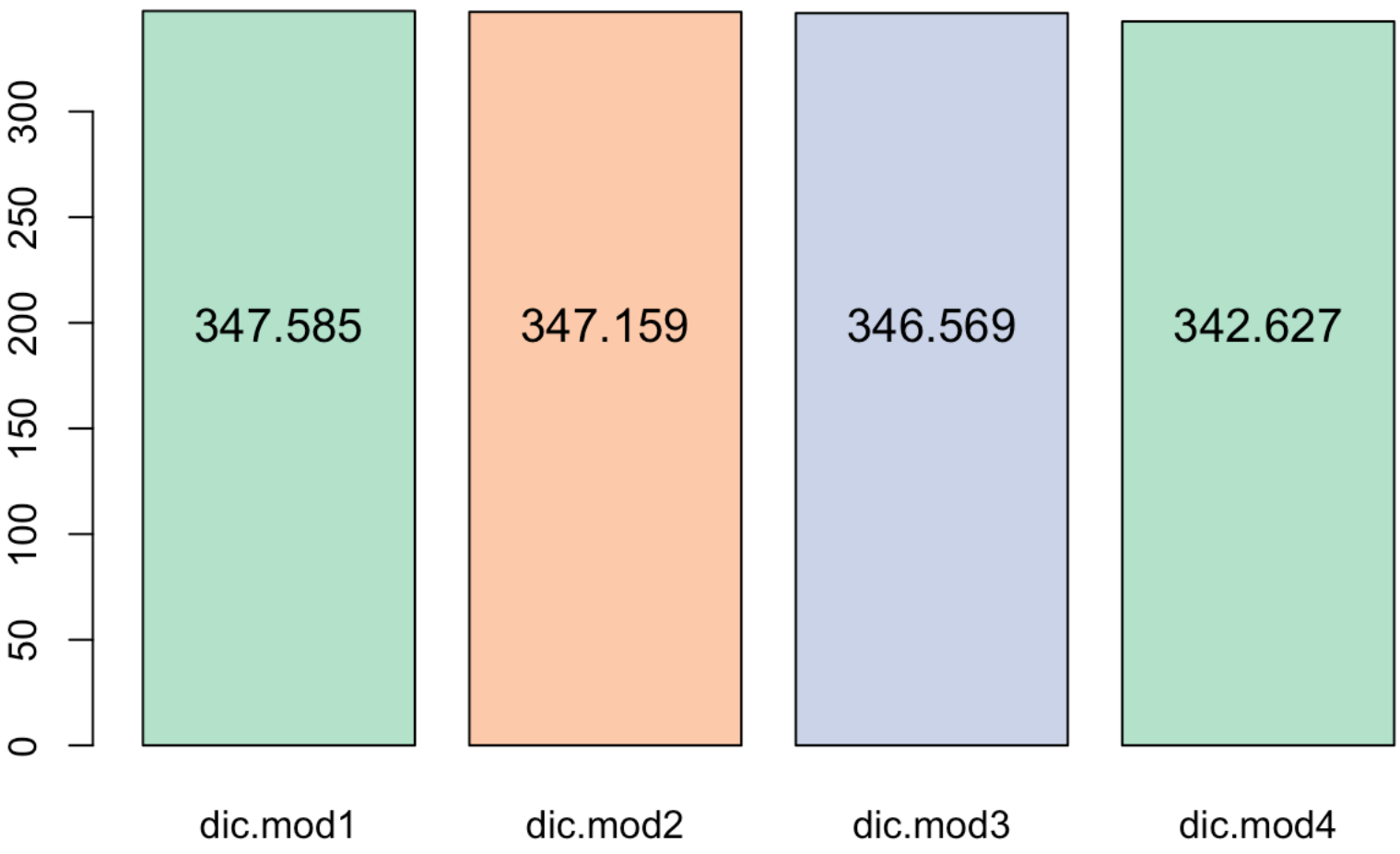
As we can see, using the Normal model for y with two explanatory variables (*Wine* and *Beer*), we improve a little bit our RMSE.

The best result is of the final bayesian model, where we use all the features in the model.

Comparison between the bayesian models

DIC

```
bars <- barplot(res.dic, col = RColorBrewer::brewer.pal(3, "Pastel2"))
text(x = bars, y = 180, label = round(res.dic,3), pos = 3, cex = 1.2, col = "black
")
```



The first three models have similar DIC. The last one model seems to be slightly better than the previous three.

Conclusion

In this project we applied linear regression using from 1 up to 5 independent variables in order describe relationships between esimated liver chyrrosis deaths and alcohol consumption in 1950 in 46 USA states. We obtained good and similar results in both frequentistic and Bayesian approaches and it seems there's a remarkable linear relationship between Death Rate and the alcohol consumption.

Since our focus is on bayesian analysis, we can conclude that all the 4 bayesian models we applied on the data fit in similar way, with slightly better results for the last model. Moreover, the diagnostic results show us that the models chosen are reliable and show ergodic properties (chains converge to stationary distributions).