Final Project - SDS II

Fabio Montello

05/08/2019

This work is a full Bayesian analysis, over data regarding the death of beetles after exposure of different concentration of carbondisulphide. The aim of this work is to figure out which of the three proposed models is the best for the given data.

Beetles data

The binary dose-response data used in this work is the one published by Bliss (1935) and analyzed by Dobson (1983) and represent the number of beetles killed after 5 hours exposure to carbon disulphide at $N = 8$ different concentrations.

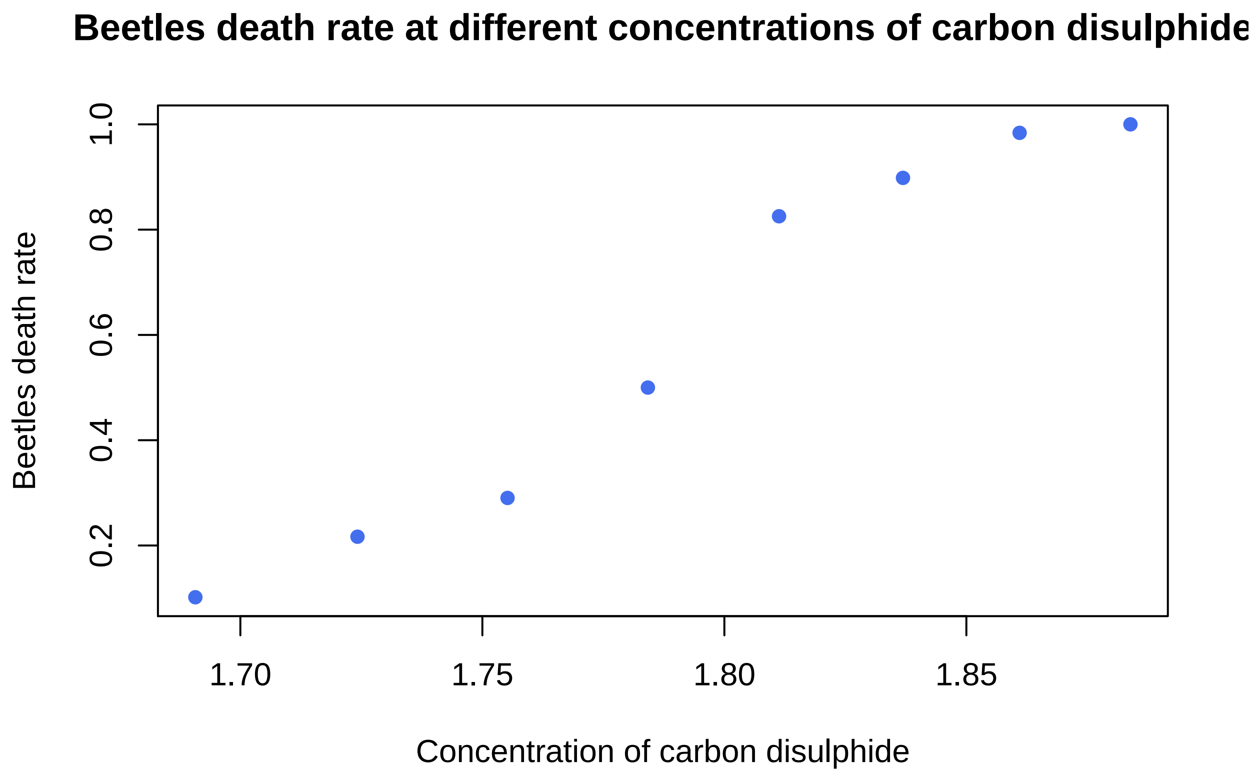
Let's have a look at the database:

```
data = list( x = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839),
             n = c(59, 60, 62, 56, 63, 59, 62, 60),
             r = c(6, 13, 18, 28, 52, 53, 61, 60),
             N = 8)

df = as.data.frame(list("Concentration" = data$x,
                        "Number_of_beetles" = data$n,
                        "Number_killed" = data$r))

df$Rate = df$Number_killed/df$Number_of_beetles
toprint = df
names(toprint) = list("Concentration", "Number of beetles", "Number of dead beetles", "Death rate")
kable(toprint, "latex", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Concentration	Number of beetles	Number of dead beetles	Death rate
1.6907	59	6	0.1016949
1.7242	60	13	0.2166667
1.7552	62	18	0.2903226
1.7842	56	28	0.5000000
1.8113	63	52	0.8253968
1.8369	59	53	0.8983051
1.8610	62	61	0.9838710
1.8839	60	60	1.0000000

Assuming that the observed number of deaths r_i at each concentration x_i is binomial with sample size n_i and true rate p_i, we can plot the death rate of our sample:

```
plot(x = data$x, y = (data$r/data$n),
     main = "Beetles death rate at different concentrations of carbon disulphide",
     xlab = "Concentration of carbon disulphide",
     ylab = "Beetles death rate",
     pch = 16, col = "royalblue2")
```



We can have a little recap for the feautures of our data:

- r_i = number of dead beetles $\in \mathbb{N}^+$
- x_i = concentration of carboon disulphide $\in \mathbb{R}^+$
- n_i = sample size $\in \mathbb{N}^+$
- p_i = true death rate $\in [0, 1]$

Models and parameters

As said previously, we assume that the observed number of deaths r_i at each concentration x_i is binomial, with a sample size equal to the number of beetles n_i and a true rate of deaths p_i:

$$Y_i \sim Binomial(n_i, p_i)$$

with the parameters $n_i \in \mathbb{N}$ and $p_i \in [0, 1]$.

For this purpose, Dobson in 1983 proposed to model the binary dose-response of deaths with three different types plausible link functions for p_i. These three models are the logistic, the probit and the extreme values (complementary log-log):

$$\begin{aligned}
p_i &= \frac{\exp(\alpha + \beta x_i)}{1 + \exp(\alpha + \beta x_i)} \\
p_i &= \Phi(\alpha + \beta x_i) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha + \beta x_i} e^{-\frac{1}{2}} dt \\
p_i &= 1 - \exp(-\exp(\alpha + \beta x_i))
\end{aligned}$$

Since we have three different models that seems suitable for our data, we want to do a MCMC simulation with each of these models, and then use these simulation to compare the models, so to observe which one fits the data the best.

The models parameters are α and β. In this case we assume that these parameters are normally distribuited:

$$\begin{aligned}\alpha &\sim Norm(0, \sigma_\alpha^2)\\ \beta &\sim Norm(0, \sigma_\beta^2)\end{aligned}$$

We are looking for a non-informative prior choice for these parameters, so we are going to set for both a σ^2 value close to ∞ or better a low $\tau = \frac{1}{\sigma^2}$:

```
tau_alpha = tau_beta = 0.00001
```

Likelihood and posterior

As we know, according to the Bayes theorem, the posterior distribution can be written as:

$$f(\alpha, \beta | n, r, X) = \frac{f(r|\alpha, \beta, n, X) f(p)}{f(r)} \propto f(r|\alpha, \beta, n, X) f(\alpha, \beta) \propto \mathcal{L}(r, \alpha, \beta, n, X)\pi(\alpha)\pi(\beta)$$

The posterior distribution embodies both prior and observed data information, which is expressed by the prior distribution $f(p)$ and the likelihood:

$$\begin{aligned}\mathcal{L}(r, \alpha, \beta, n, X) &= \prod_{i=1}^{m} \binom{n_i}{r_i} link_f(\alpha, \beta, x_i)^{r_i} \, (1 - link_f(\alpha, \beta, x_i))^{n_i - r_i}\\ &\propto \prod_{i=1}^{m} link_f(\alpha, \beta, x_i)^{r_i} \, (1 - link_f(\alpha, \beta, x_i))^{n_i - r_i}\end{aligned}$$

where $link_f(\alpha, \beta, x_i)$ is either the logistic, the probit or the extreme values function, according to which link function we want to use in our model. So we have that our posterior probability will be:

$$\begin{aligned}f(\alpha, \beta | n, r, X) &\propto \mathcal{L}(r|n, X, \alpha, \beta) \, \pi(\alpha) \, \pi(\beta)\\ &\propto \prod_{i=1}^{m} link_f(\alpha, \beta, x_i)^{r_i} \, (1 - link_f(\alpha, \beta, x_i))^{n_i - r_i} \, \frac{1}{\sqrt{2\pi\sigma_\alpha^2}} e^{-\frac{\alpha^2}{2\sigma_\alpha^2}} \, \frac{1}{\sqrt{2\pi\sigma_\beta^2}} e^{-\frac{\beta^2}{2\sigma_\beta^2}}\end{aligned}$$

Bayesian Analysis with MCMC

Before starting any type of analysis, is a good practice to create a list that contains all the data needed to be processed later on:

```
data.list= list(r = data$r,
                n = data$n,
                x = data$x,
                tau_alpha = tau_alpha,
                tau_beta = tau_beta,
                N = 8)
```

It is important also to point out that we are going to standardize each dose of x_i about the mean: this gives approximately uncorrelated regression coefficients, and greatly improves convergence. Of course this imply that then α is going to be a latent variable, which can be derived from α^* in the model or after the model has run, simply by inverting the standardization:

$$\alpha = \alpha^* - \beta\bar{x}$$

Concerning the constant parameter α^*, its interpretation corresponds to the expected value of the response variable Y when the observed values of all covariates are equal to zero. Frequently such combination lies outside the range of the observed covariate values. In such cases, the interpretation of α^* is not reliable. Frequently,direct interpretation of α^* does not lead to realistic and sensible interpretation. An alternative is to center around zero all explanatory variables X, by subtracting their sample mean.

Logistic model

As first model for our Bayesian analysis we want to start with the logistic model:

$$Y_i \sim Binomial(n_i, p_i)$$

$$p_i = \frac{\exp(\alpha + \beta x_i)}{1 + \exp(\alpha + \beta x_i)}$$

All the models presented will be executed through JAGS.
As premised previously, the prior distributions for the two parameters to estimate have been chosen to be uninformative priors, due to our ignorance over the topic. This means we picked Normal distributions with mean equal to zero and precision equal to 0.00001 (infinite variance).

Then we can proceed by translating into code the model we formulized previously, taking into account the link function chosen:

```
model_1 = "
      model {
       for( i in 1 : N ) {
          r[i] ~ dbin(p[i],n[i])
          logit(p[i]) <- alpha.star + beta * (x[i] - mean(x[]))
          rhat[i] <- n[i] * p[i]
       }
       alpha = alpha.star - beta * mean(x[])
       beta ~ dnorm(0.0,tau_beta)
       alpha.star ~ dnorm(0.0,tau_alpha)
    }";
```

The chains will have different initial values, because this is a good check for the stationarity of the chain: in fact, a stationary chain is not influenced by the initial state, hence any different simulation of the chain should return the same results. The initial values are then chosen different from each others, but with no particular meaning:

```
init.list.nod =list(list(alpha.star = 0, beta =0),
                    list(alpha.star = 1, beta =2),
                    list(alpha.star = 3, beta =0))
```

In order to get more accurate results, the model will run 3 different chains, where the number of replications for each one it's equal to 11000. From each one the first 1000 observation have been burnt, just to remove the initial noise of the first states of the chains, and to take just the part where we wish the chains become stationary.

```
mcmc_res = jags( model.file = textConnection(model_1),
                 data = data.list,
                 n.chains = 3,
                 n.iter = 11000,
                 n.burnin = 1000,
                 inits = init.list.nod,
                 parameters.to.save = c("alpha","alpha.star","beta", "rhat"))
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 74
##
## Initializing model
```

After the model has been fitted and compiled properly, it's time to have a look at a first summary of the results:

```r
logit_res <- mcmc_res
print(mcmc_res)
```

```
## Inference for Bugs model at "5", fit using jags,
##  3 chains, each with 11000 iterations (first 1000 discarded), n.thin = 10
##  n.sims = 3000 iterations saved
##            mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
## alpha      -61.199   5.644 -72.062 -64.874 -60.990 -57.538 -51.312 1.002
## alpha.star   0.751   0.146   0.471   0.660   0.749   0.842   1.025 1.004
## beta        34.543   3.167  28.989  32.484  34.423  36.604  40.600 1.005
## rhat[1]      3.537   1.494   1.912   2.819   3.409   4.075   5.663 1.003
## rhat[2]      9.878   1.998   6.689   8.672   9.759  10.953  13.410 1.003
## rhat[3]     22.449   2.261  18.306  21.010  22.426  23.869  26.756 1.002
## rhat[4]     33.933   1.823  30.184  32.779  33.944  35.113  37.303 1.002
## rhat[5]     50.135   1.726  46.590  49.057  50.206  51.289  53.297 1.003
## rhat[6]     53.284   1.204  50.815  52.575  53.366  54.097  55.289 1.011
## rhat[7]     59.188   0.883  57.470  58.747  59.276  59.735  60.452 1.042
## rhat[8]     58.704   0.619  57.705  58.468  58.772  59.027  59.392 1.103
## deviance    39.878  13.189  37.482  38.010  38.822  40.345  45.510 1.028
##            n.eff
## alpha       2100
## alpha.star  1700
## beta        2200
## rhat[1]     1300
## rhat[2]     1200
## rhat[3]     1100
## rhat[4]     1600
## rhat[5]     2600
## rhat[6]     2700
## rhat[7]     2500
## rhat[8]     2700
## deviance    1900
##
## 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 = 87.0 and DIC = 126.9
## DIC is an estimate of expected predictive error (lower deviance is better).
```

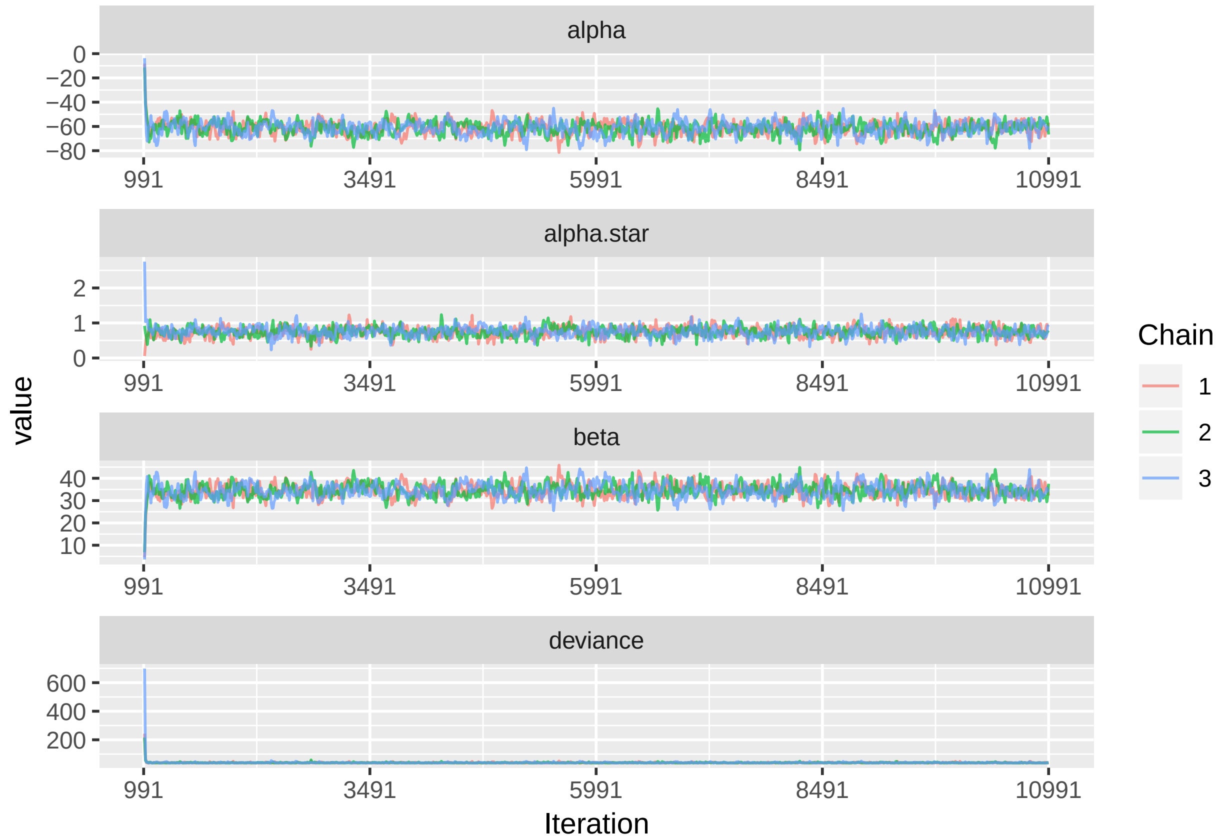
Before discussing any kind of conclusion on the results is good practice to check if the model is proper, which mean that the distribution converged.
A first indicator of convergence can be checked in the summary printed above: the $\hat{R}$ is a test that checks

the convergence of a serie, and it is equal to 1 when the series converges. As we can see, all the parameters displayed seem to have a value very close to 1, hence we can start assuming the series are converging.

To go further with the checks, it is possible to have a look also at the traceplots and the mean plots.

The traceplot helps us check whether a Markov Chain has a random sampling behavior, and assess mixing across chains and convergence:

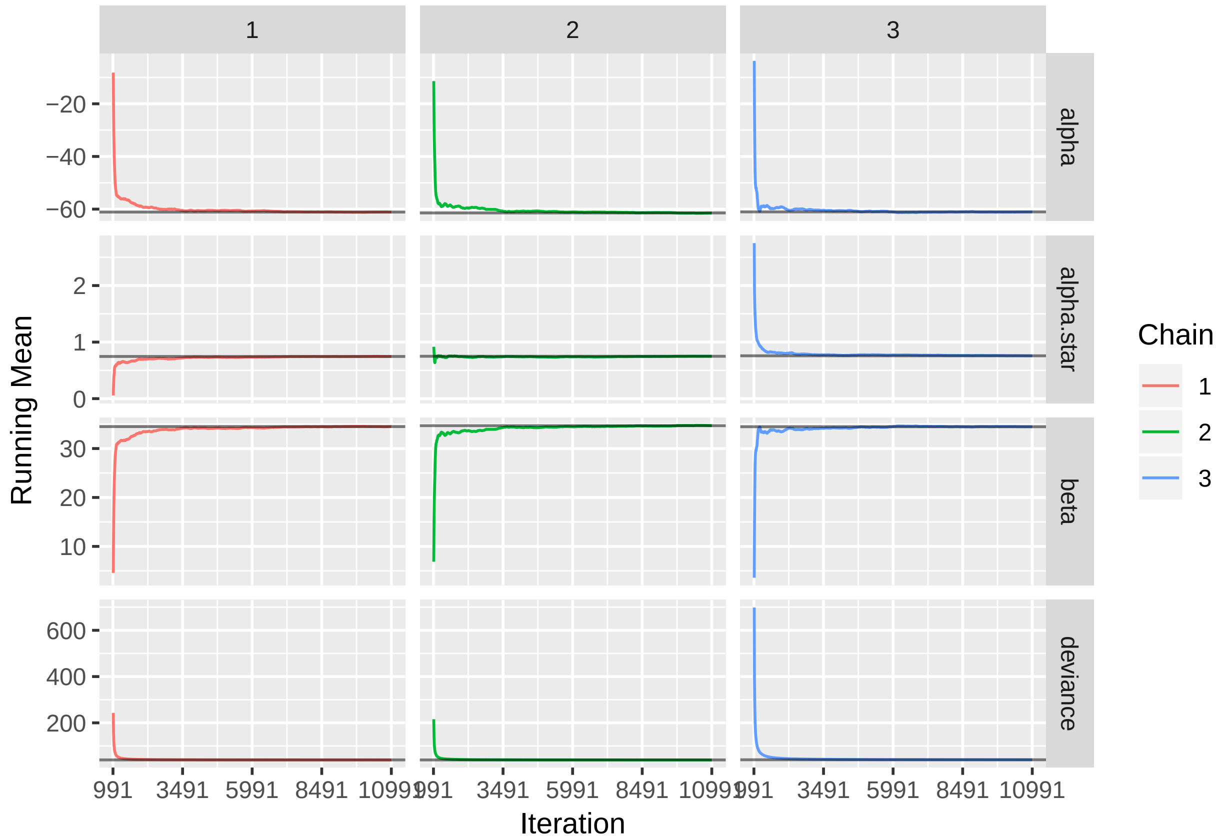
```
chain_print = ggs(as.mcmc(mcmc_res))
ggs_traceplot(chain_print)
```



The chains for all the parameters does not seem to follow some patterns, depending on the states or on the initial points. Having a random behaviour over the same interval of values is a good indicator of the convergence of the chain.

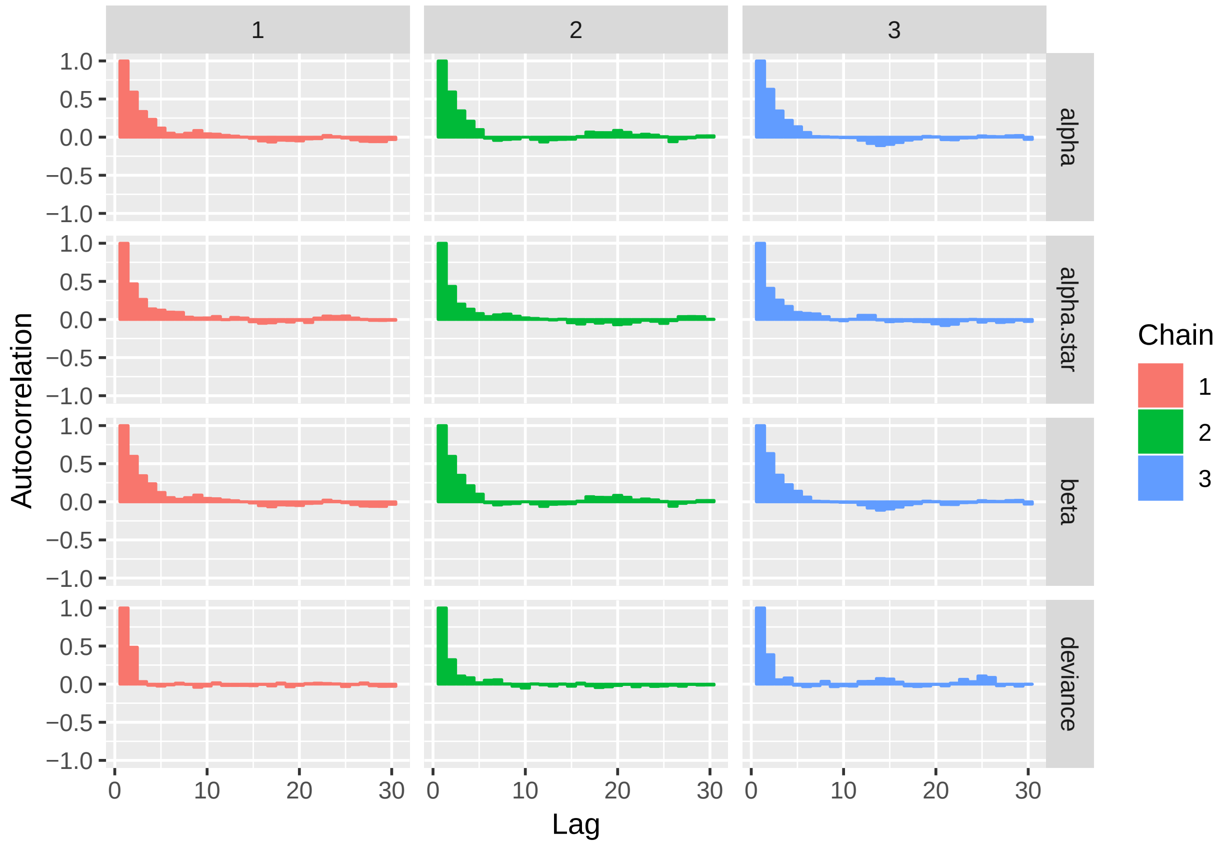
This can also be confirmed by the running mean plot, which shows how the mean of the estimator converges to the final average value after a while:

```
ggs_running(chain_print)
```



Another important check for the presence of pattern in the chain is the autocorrelation plot, which let us find out if in any part of the chain there are any some parts that are correlated each other, meaning the data seems to follow some patterns. What we are expecting is an autocorrelation plot that has values as close as possible to 0, avoiding any loop occurance.

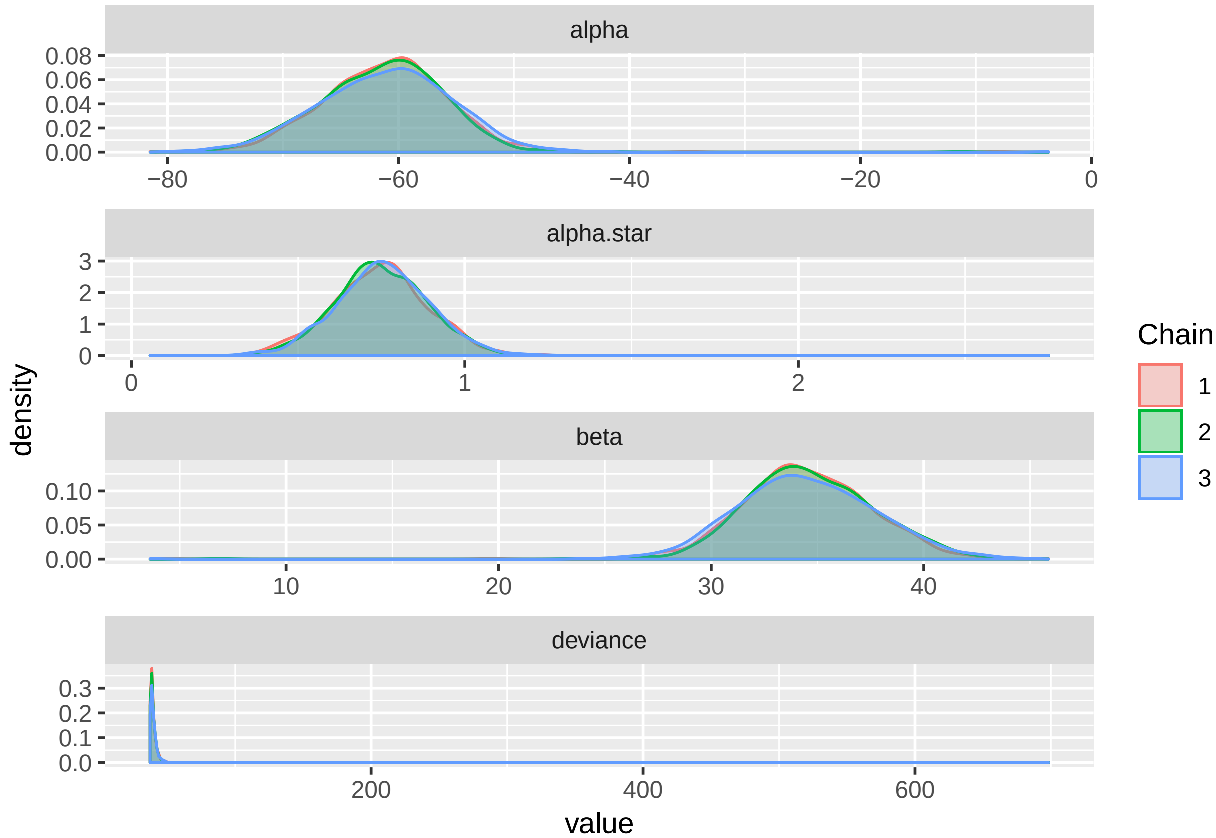
```
ggs_autocorrelation(chain_print, nLags = 30)
```



As we can see, after an initial state of autocorrelation, this seems to disappear almost totally by getting very close to zero. These three results together give us a solid proof of the convergence of our chains.

A final step, and good practice, to assess the convergence is to estimate the density of the parameters:

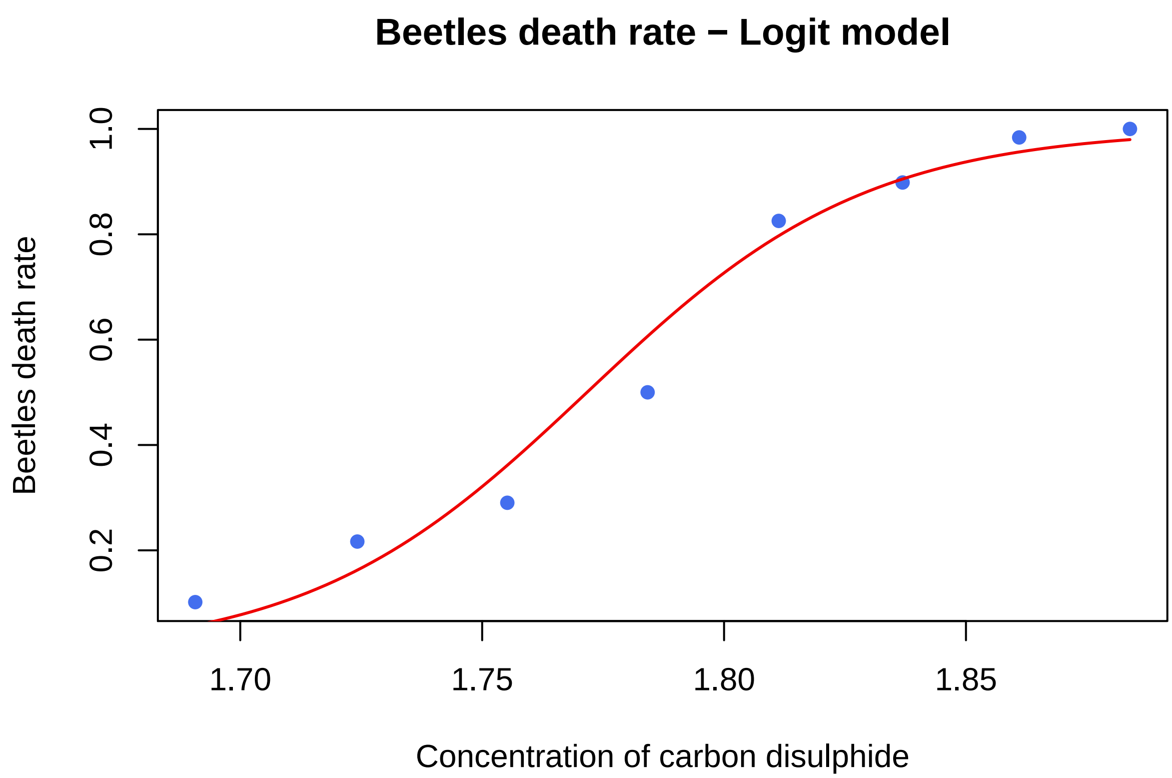
```
ggs_density(chain_print)
```



The presence of a smooth unimodal distribution is desirable, since it means that the chains have found the stationarity in the interval around the mode of the distribution. Here we can see that for the 4 chains, results are more or less identical, hence it is also another indicator of the goodness of this model.

Once we made sure the parameters we estimated have been correctly simulated, we can proceed by visualizing our logistic model with the data we had:

```
alpha <-as.numeric(mcmc_res$BUGSoutput$mean$alpha)
beta <- as.numeric(mcmc_res$BUGSoutput$mean$beta)

logit <- function(x, a, b) return(exp(a + b * x)/(1+exp(a + b * x)))

plot(x = data$x, y = (data$r/data$n),
     main = "Beetles death rate - Logit model",
     xlab = "Concentration of carbon disulphide",
     ylab = "Beetles death rate",
     pch = 16, col = "royalblue2")

curve(logit(x, alpha, beta), add = T, col = 'red2', lwd = 1.5)
```

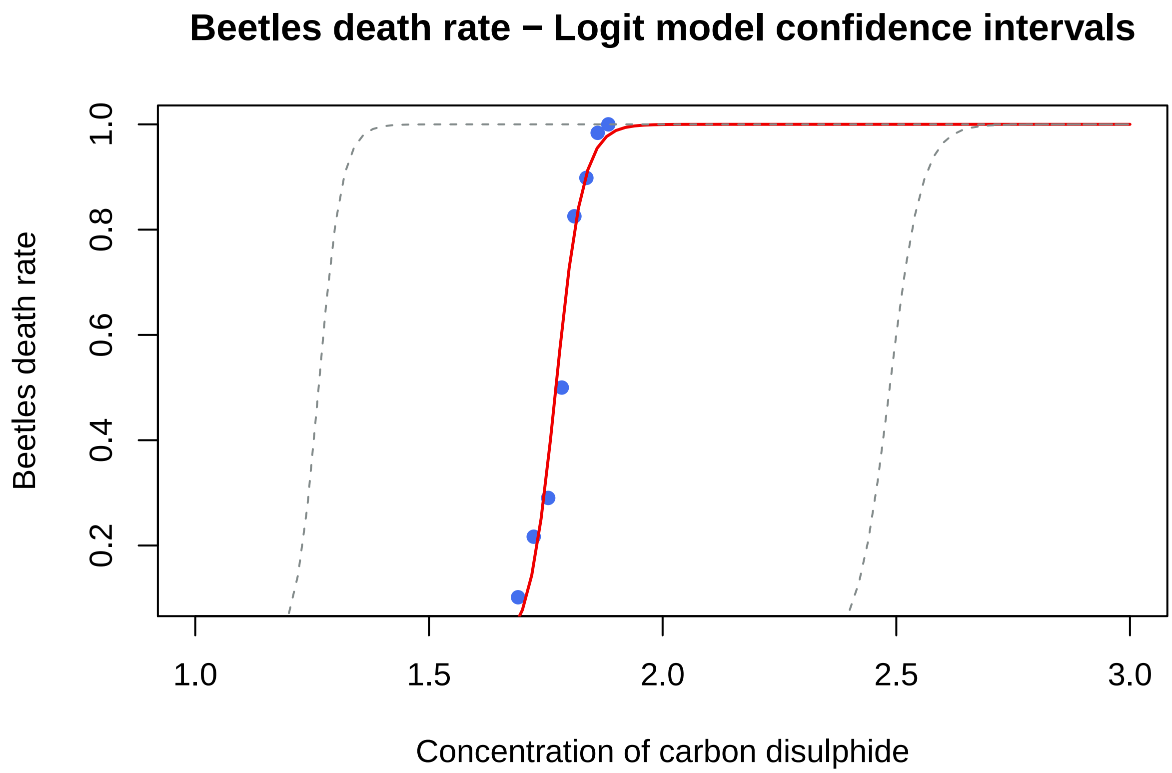


The model seems correct and graphically fitting the data. Let's have a look at the confidence interval values:

```
print(mcmc_res$BUGSoutput$summary[1:4, c(1:3,7)])
```

```
##                    mean         sd        2.5%      97.5%
## alpha       -61.1986841  5.6438027 -72.0618660 -51.312251
## alpha.star    0.7511321  0.1456266   0.4711782   1.024682
## beta         34.5427416  3.1673761  28.9889496  40.599850
## deviance     39.8778407 13.1890541  37.4820501  45.509700
```

```
alpha25 <-as.numeric(mcmc_res$BUGSoutput$summary[1,3])
beta25 <- as.numeric(mcmc_res$BUGSoutput$summary[3,3])
alpha975 <-as.numeric(mcmc_res$BUGSoutput$summary[1,7])
beta975 <- as.numeric(mcmc_res$BUGSoutput$summary[3,7])

plot(x = data$x, y = (data$r/data$n),
     main = "Beetles death rate - Logit model confidence intervals",
     xlab = "Concentration of carbon disulphide",
     ylab = "Beetles death rate",
     pch = 16, col = "royalblue2", xlim = c(1,3))

curve(logit(x, alpha, beta), add = T, col = 'red2', lwd = 1.5)
curve(logit(x, alpha25, beta25), add = T, lwd = 1, lty = 2, col = 'azure4')
curve(logit(x, alpha975, beta975), add = T, lwd = 1, lty = 2, col = 'azure4')
```



Probit model

After the logistic model, the second model we want to work with is the probit model:

$$Y_i \sim Binomial(n_i, p_i)$$

$$p_i = \Phi(\alpha + \beta x_i) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha+\beta x_i} e^{-\frac{1}{2}} dt$$

The only variation in respect with the previous model is the link function, that this time is a probit function. All the settings of the Markov Chain, the prior distribution, the number of chains, the initial values, number of iterations, precision value, size of burn-in remained exactly the same.

```
model_2 = "
      model {
       for( i in 1 : N ) {
          r[i] ~ dbin(p[i],n[i])
          probit(p[i]) <- alpha.star + beta * (x[i] - mean(x[]))
          rhat[i] <- n[i] * p[i]
       }
       alpha = alpha.star - beta * mean(x[])
       beta ~ dnorm(0.0,tau_beta)
       alpha.star ~ dnorm(0.0,tau_alpha)
    }"

init.list.nod =list(list(alpha.star = 0, beta =0),
                    list(alpha.star = 1, beta =2),
```

```
                    list(alpha.star = 3, beta =0))

mcmc_res = jags( model.file = textConnection(model_2),
                  data = data.list,
                  n.chains = 3,
                  n.iter = 11000,
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta", "rhat"))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 74
##
## Initializing model
```

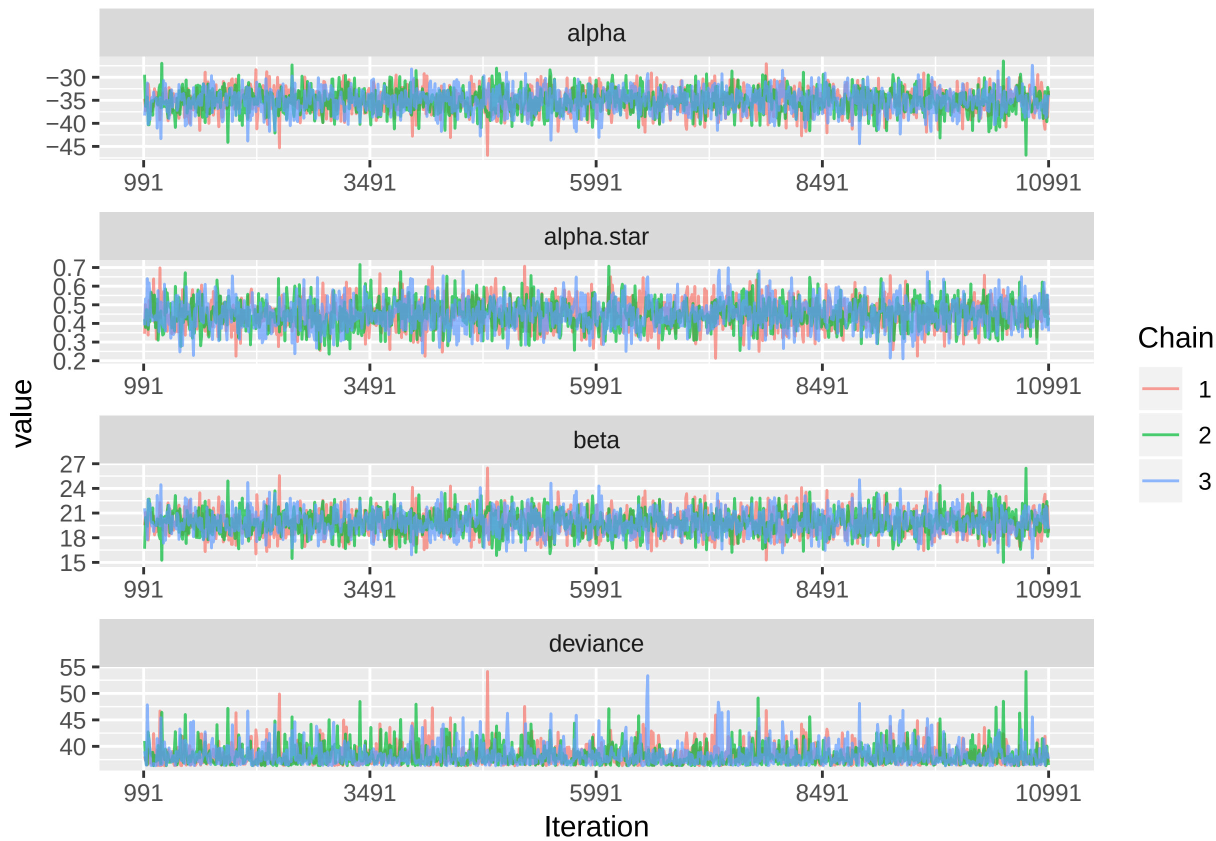
```
probit_res <- mcmc_res
print(mcmc_res)
```

```
## Inference for Bugs model at "6", fit using jags,
##  3 chains, each with 11000 iterations (first 1000 discarded), n.thin = 10
##  n.sims = 3000 iterations saved
##            mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
## alpha      -35.170   2.625 -40.410 -36.897 -35.099 -33.408 -30.130 1.001
## alpha.star   0.449   0.078   0.303   0.396   0.448   0.500   0.606 1.001
## beta        19.861   1.476  17.032  18.867  19.823  20.831  22.818 1.001
## rhat[1]      3.408   1.007   1.751   2.701   3.278   4.032   5.618 1.001
## rhat[2]     10.722   1.698   7.566   9.575  10.593  11.841  14.200 1.001
## rhat[3]     23.476   1.923  19.842  22.167  23.409  24.741  27.307 1.001
## rhat[4]     33.855   1.612  30.723  32.796  33.815  34.922  37.125 1.001
## rhat[5]     49.670   1.617  46.412  48.557  49.718  50.773  52.755 1.001
## rhat[6]     53.330   1.141  50.918  52.601  53.388  54.130  55.401 1.001
## rhat[7]     59.638   0.729  57.998  59.216  59.712  60.157  60.833 1.001
## rhat[8]     59.194   0.356  58.339  59.006  59.250  59.450  59.708 1.001
## deviance    38.318   2.051  36.372  36.890  37.693  39.067  43.929 1.001
##            n.eff
## alpha       3000
## alpha.star  3000
## beta        3000
## rhat[1]     3000
## rhat[2]     3000
## rhat[3]     3000
## rhat[4]     3000
## rhat[5]     3000
## rhat[6]     3000
## rhat[7]     3000
## rhat[8]     3000
## deviance    3000
##
## 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 = 2.1 and DIC = 40.4
## DIC is an estimate of expected predictive error (lower deviance is better).
```

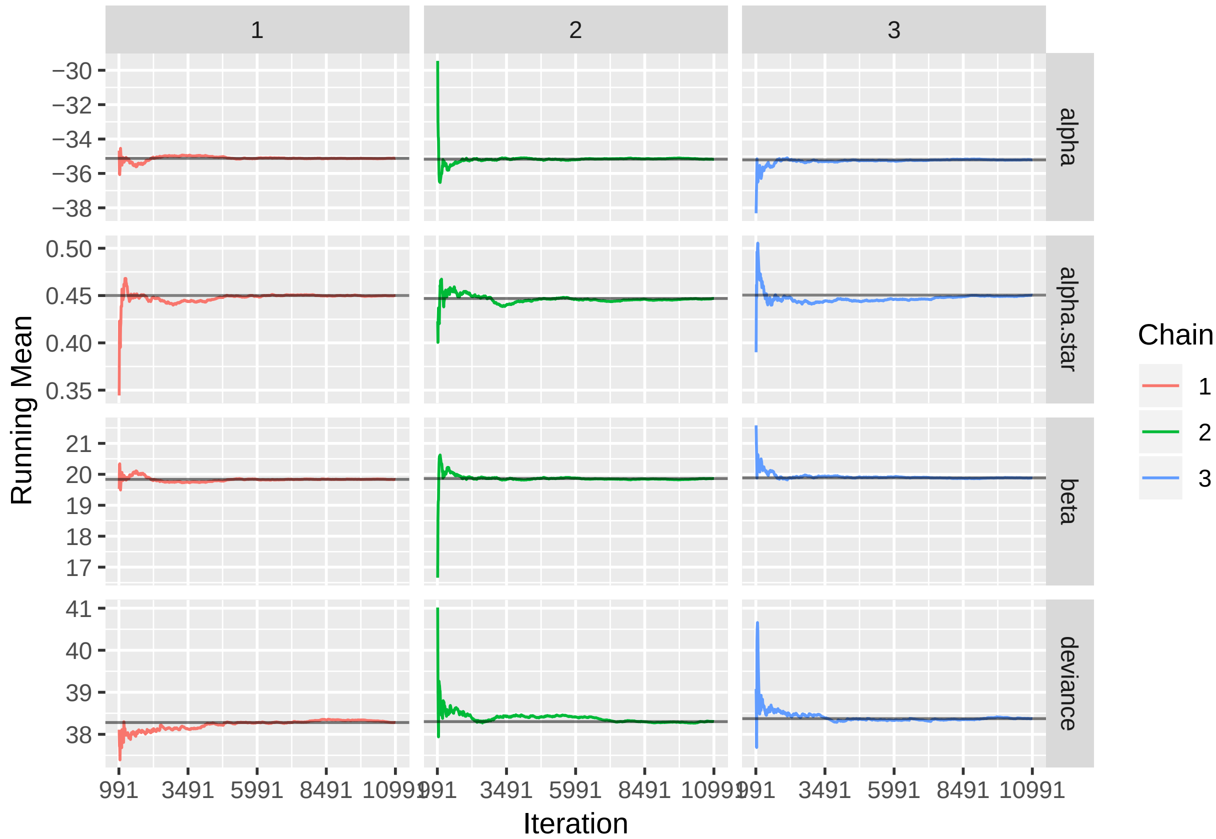
Before anything else, let's check again that the chains converges: the $\hat{R}$ information is very close to 1, hence it seems we are dealing with a stationary chain. Let's have further checks with the traceplots:

```
chain_print = ggs(as.mcmc(mcmc_res))
ggs_traceplot(chain_print)
```



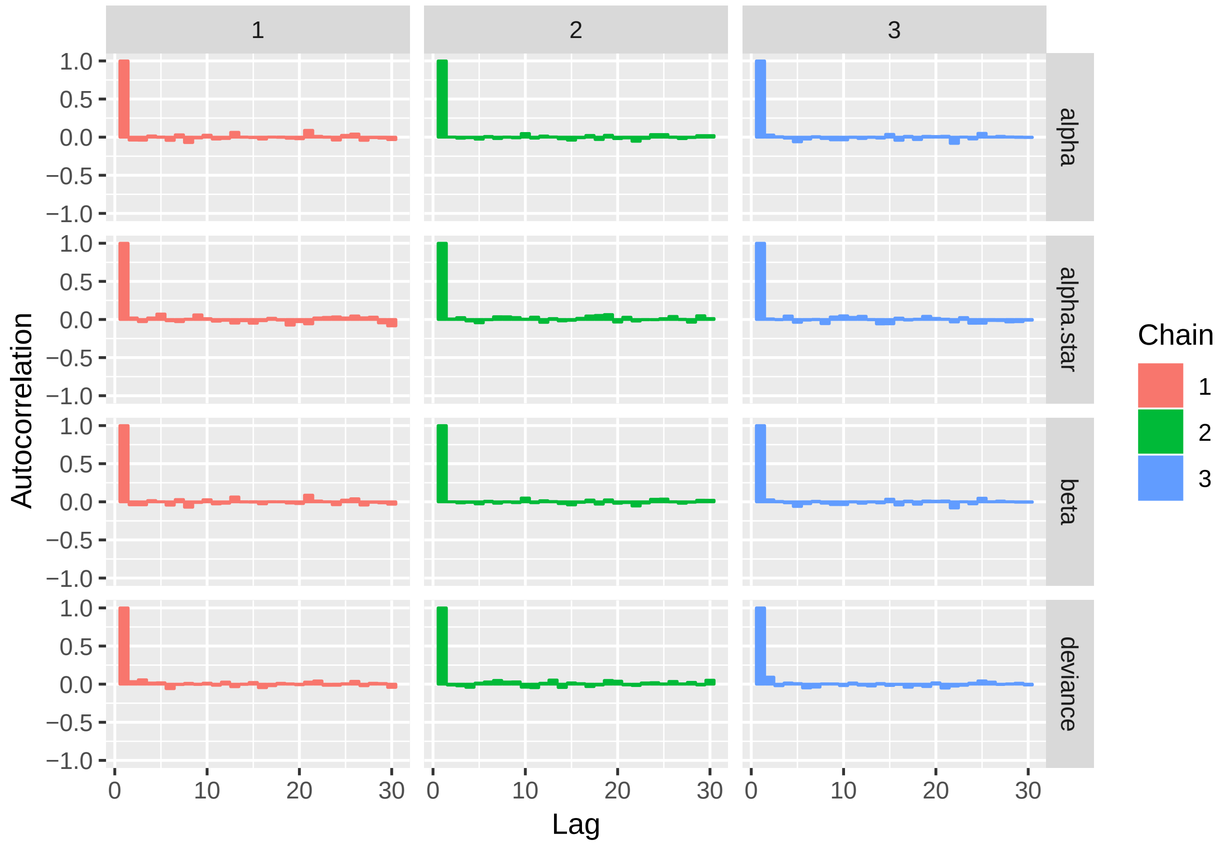
The random mixing pattern seems solid, another confermation of the validity of the chains. We can also have a look at the mean plot:

```
ggs_running(chain_print)
```



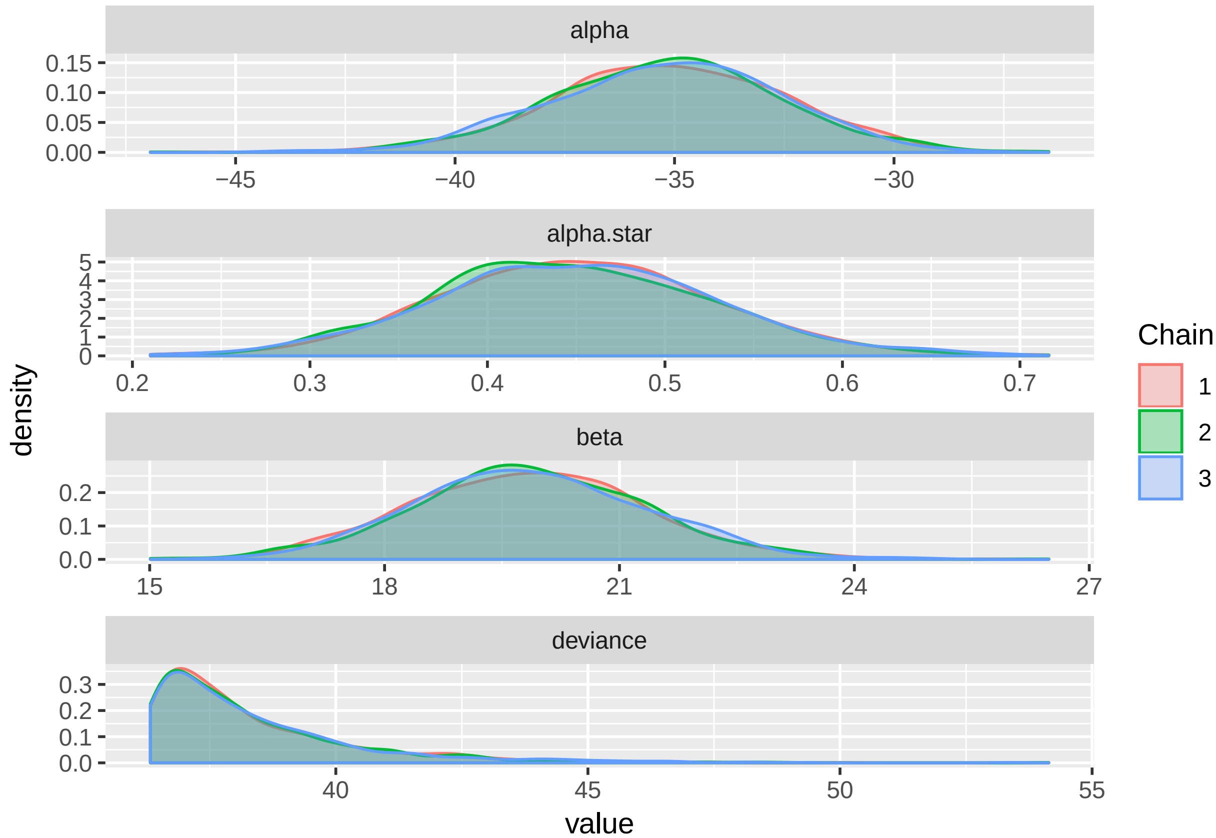
which seems to converge properly after few states. Let's check also the presence of any weird pattern of our chain with the autocorrelation plot:

```
ggs_autocorrelation(chain_print, nLags = 30)
```



where we can observe that, after the initial state, the value is very close to 0, indicating no correlation at all. Finally, we can also check the estimated density of the estimators:

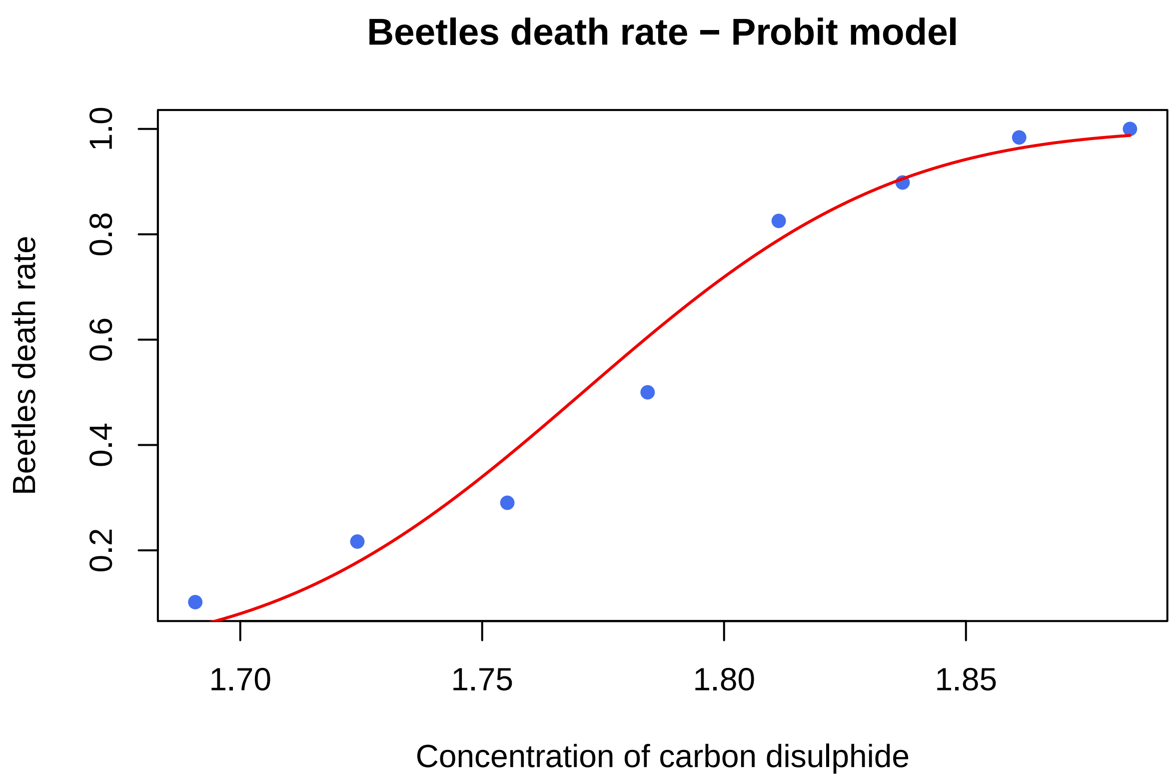
```
ggs_density(chain_print)
```



Again the distributions seems to be smooth, unimodal and almost identical between the chains. This is the ultimate assessment of the stationarity, and thus validity, of the chains in our model.

We can proceed by looking at the goodness of fit of our estimates:

```
alpha <-as.numeric(mcmc_res$BUGSoutput$mean$alpha)
beta <- as.numeric(mcmc_res$BUGSoutput$mean$beta)

plot(x = data$x, y = (data$r/data$n),
     main = "Beetles death rate - Probit model",
     xlab = "Concentration of carbon disulphide",
     ylab = "Beetles death rate",
     pch = 16, col = "royalblue2")

curve(pnorm(alpha + (beta * x), mean = 0, sd = 1, lower.tail = TRUE),
      add = T, col = 'red2', lwd = 1.5)
```

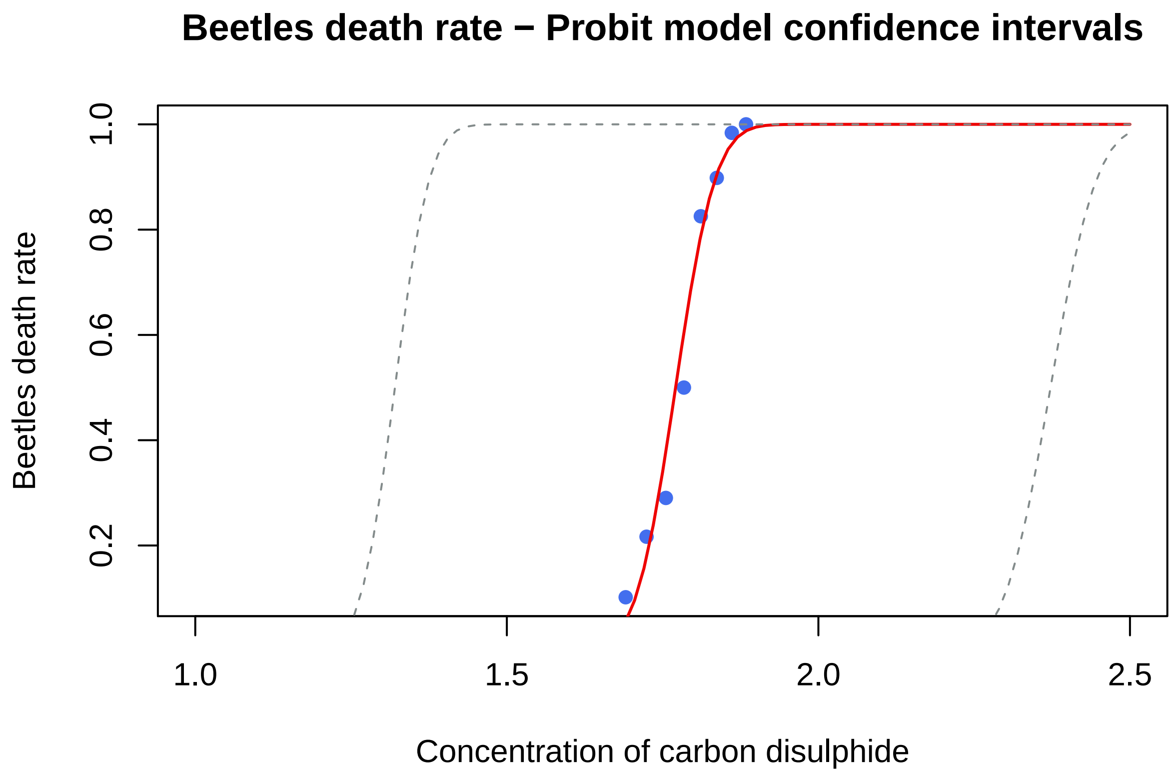


Seems to be correct. We can check also the confidence intervals:

```
print(mcmc_res$BUGSoutput$summary[1:4, c(1:3,7)])
```

```
##                   mean         sd        2.5%       97.5%
## alpha      -35.1702775 2.62529294 -40.4096872 -30.1298130
## alpha.star   0.4491214 0.07790201   0.3027681   0.6062352
## beta        19.8611032 1.47596264  17.0322184  22.8184020
## deviance    38.3183097 2.05135690  36.3720636  43.9291015
```

```
alpha25 <-as.numeric(mcmc_res$BUGSoutput$summary[1,3])
beta25 <- as.numeric(mcmc_res$BUGSoutput$summary[3,3])
alpha975 <-as.numeric(mcmc_res$BUGSoutput$summary[1,7])
beta975 <- as.numeric(mcmc_res$BUGSoutput$summary[3,7])

plot(x = data$x, y = (data$r/data$n),
     main = "Beetles death rate - Probit model confidence intervals",
     xlab = "Concentration of carbon disulphide",
     ylab = "Beetles death rate",
     pch = 16, col = "royalblue2", xlim = c(1,2.5))

curve(pnorm(alpha + (beta * x), mean = 0, sd = 1, lower.tail = TRUE),
      add = T, col = 'red2', lwd = 1.5)
curve(pnorm(alpha25 + (beta25 * x), mean = 0, sd = 1, lower.tail = TRUE),
      add = T, lwd = 1, lty = 2, col = 'azure4')
curve(pnorm(alpha975 + (beta975 * x), mean = 0, sd = 1, lower.tail = TRUE),
      add = T, lwd = 1, lty = 2, col = 'azure4')
```



Extreme values model (or complementary log-log model)

Finally, we want to do the same thing for the the extreme values model. Once again, let's have a look at the model first:

$$Y_i \sim Binomial(n_i, p_i)$$

$$p_i = 1 - \exp(-\exp(\alpha + \beta x_i))$$

For the building of the model, as previously, we changed only the link function, this time a extreme values (or complementary log-log) function. Let's have a reminder on the settings of the Markov Chain: the prior distributions are distribuited as a Normal, centered on 0 and with a variance equal to 100000, 3 chains, 11000 iterations for each chain and a burn-in of 1000 and random initial values. Now we can proceed with the model:

```
model_3 = "
      model {
       for( i in 1 : N ) {
          r[i] ~ dbin(p[i],n[i])
          cloglog(p[i]) <- alpha.star + beta * (x[i] - mean(x[]))
          rhat[i] <- n[i] * p[i]
       }
       alpha = alpha.star - beta * mean(x[])
       beta ~ dnorm(0.0,tau_beta)
       alpha.star ~ dnorm(0.0,tau_alpha)
    }"

init.list.nod =list(list(alpha.star = 0, beta =0),
```

```r
                    list(alpha.star = 1, beta =2),
                    list(alpha.star = 3, beta =0))

mcmc_res = jags( model.file = textConnection(model_3),
                  data = data.list,
                  n.chains = 3,
                  n.iter = 11000,
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta", "rhat"))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 74
##
## Initializing model
```

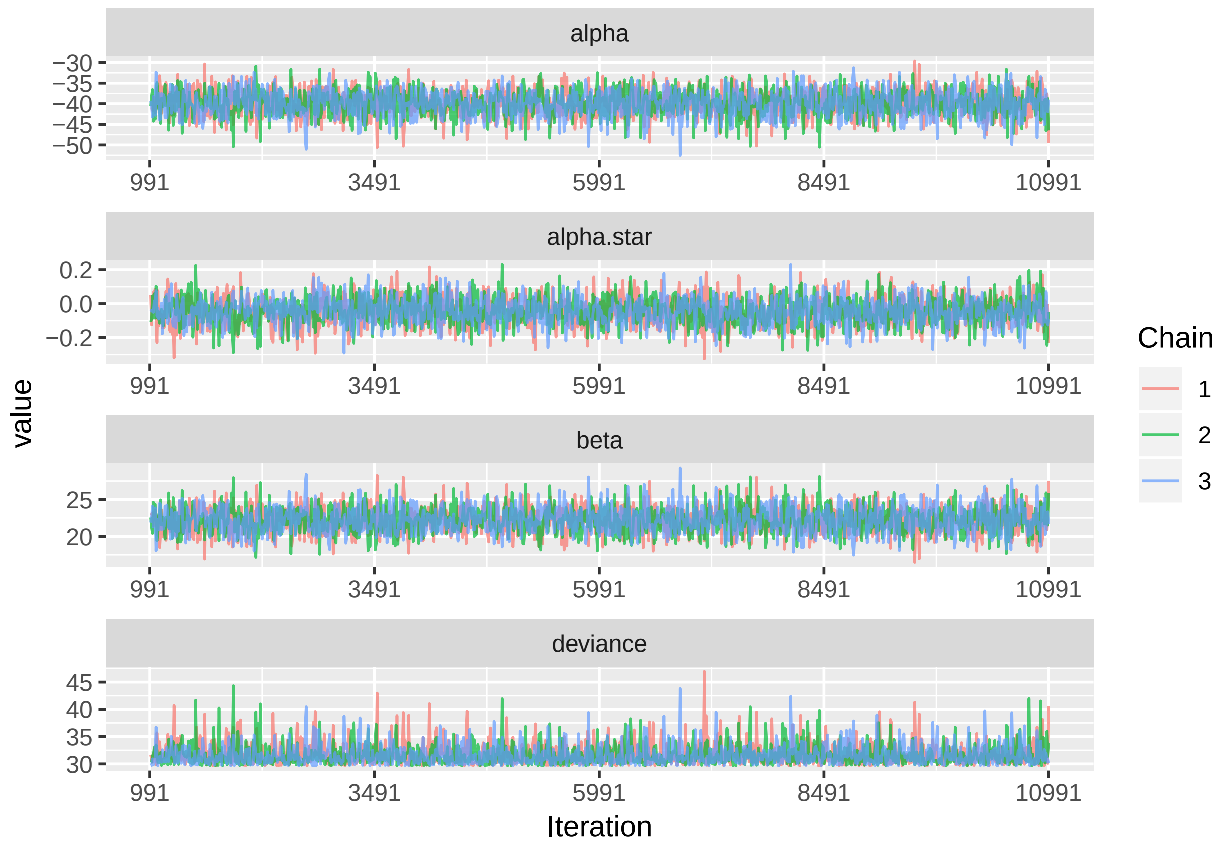
```r
cloglog_res <- mcmc_res
print(mcmc_res)
```

```
## Inference for Bugs model at "7", fit using jags,
##  3 chains, each with 11000 iterations (first 1000 discarded), n.thin = 10
##  n.sims = 3000 iterations saved
##             mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
## alpha       -39.880   3.254 -46.345 -42.121 -39.871 -37.579 -33.939 1.001
## alpha.star   -0.044   0.082  -0.203  -0.097  -0.045   0.013   0.114 1.001
## beta         22.212   1.808  18.912  20.918  22.208  23.454  25.806 1.001
## rhat[1]       5.601   1.126   3.664   4.793   5.511   6.345   8.013 1.001
## rhat[2]      11.256   1.597   8.392  10.142  11.178  12.350  14.625 1.001
## rhat[3]      20.899   1.921  17.231  19.575  20.830  22.217  24.838 1.001
## rhat[4]      30.343   1.710  27.021  29.190  30.326  31.482  33.732 1.001
## rhat[5]      47.799   1.795  44.276  46.593  47.835  49.018  51.231 1.002
## rhat[6]      54.127   1.258  51.491  53.297  54.223  55.013  56.371 1.002
## rhat[7]      61.043   0.534  59.786  60.739  61.144  61.446  61.780 1.002
## rhat[8]      59.919   0.094  59.668  59.893  59.951  59.981  59.997 1.008
## deviance     31.691   2.003  29.705  30.279  31.083  32.498  37.120 1.005
##             n.eff
## alpha        3000
## alpha.star   3000
## beta         3000
## rhat[1]      3000
## rhat[2]      3000
## rhat[3]      3000
## rhat[4]      3000
## rhat[5]      3000
## rhat[6]      3000
## rhat[7]      3000
## rhat[8]      1800
## deviance      740
##
```

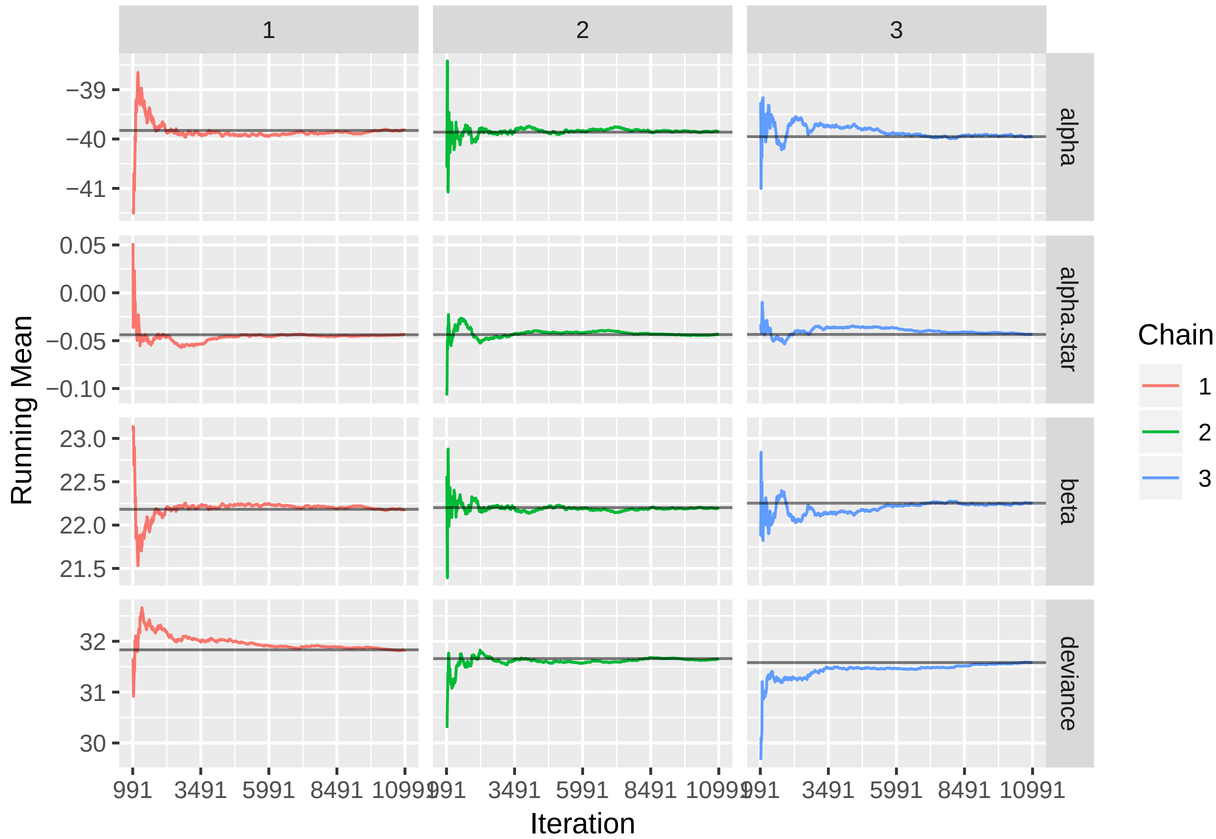
```
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 2.0 and DIC = 33.7
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Before having a look at the model with the data, let's check again the chains convergence: $\hat{R}$ information is very close to 1, which indicates a convergence. Now let's look at the plots of the chains, the parameters mean and the autocorrelation, as we did before:

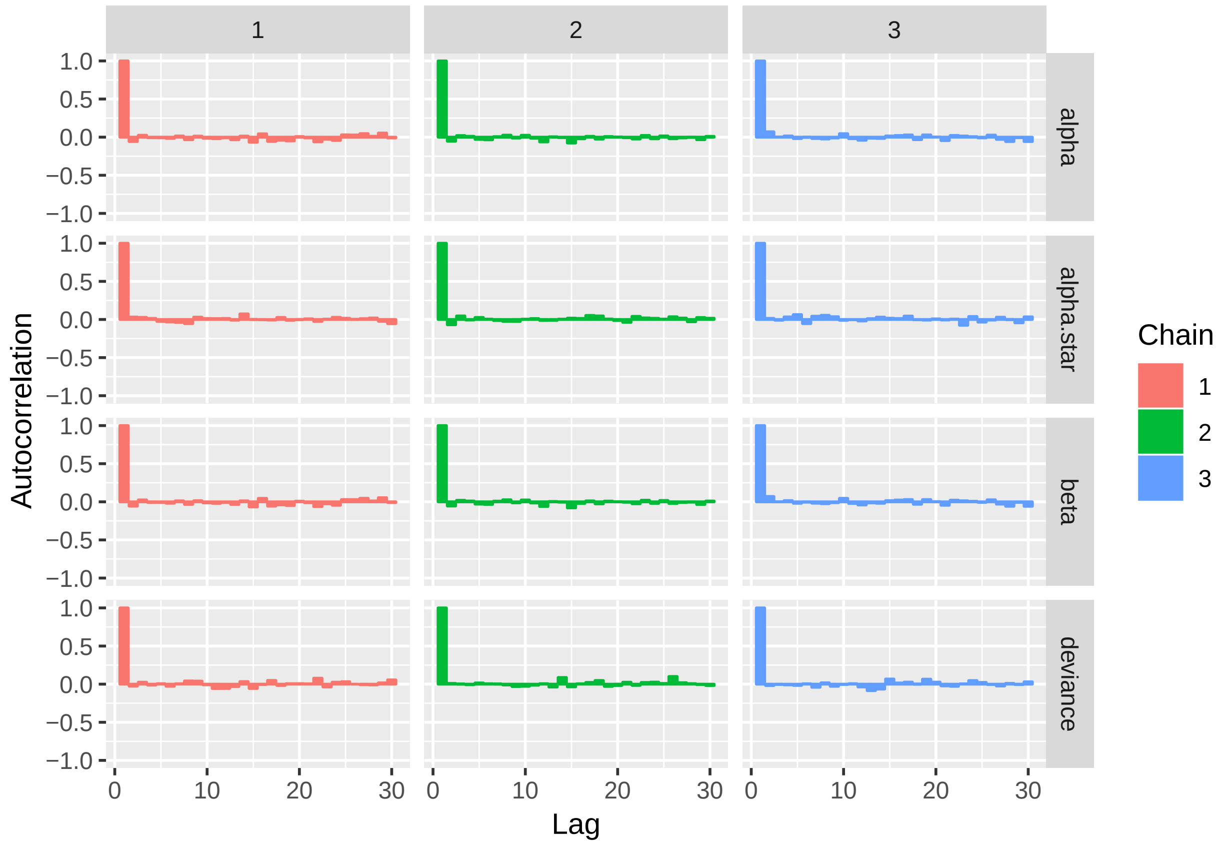
```
chain_print = ggs(as.mcmc(mcmc_res))
ggs_traceplot(chain_print)
```




```
ggs_running(chain_print)
```

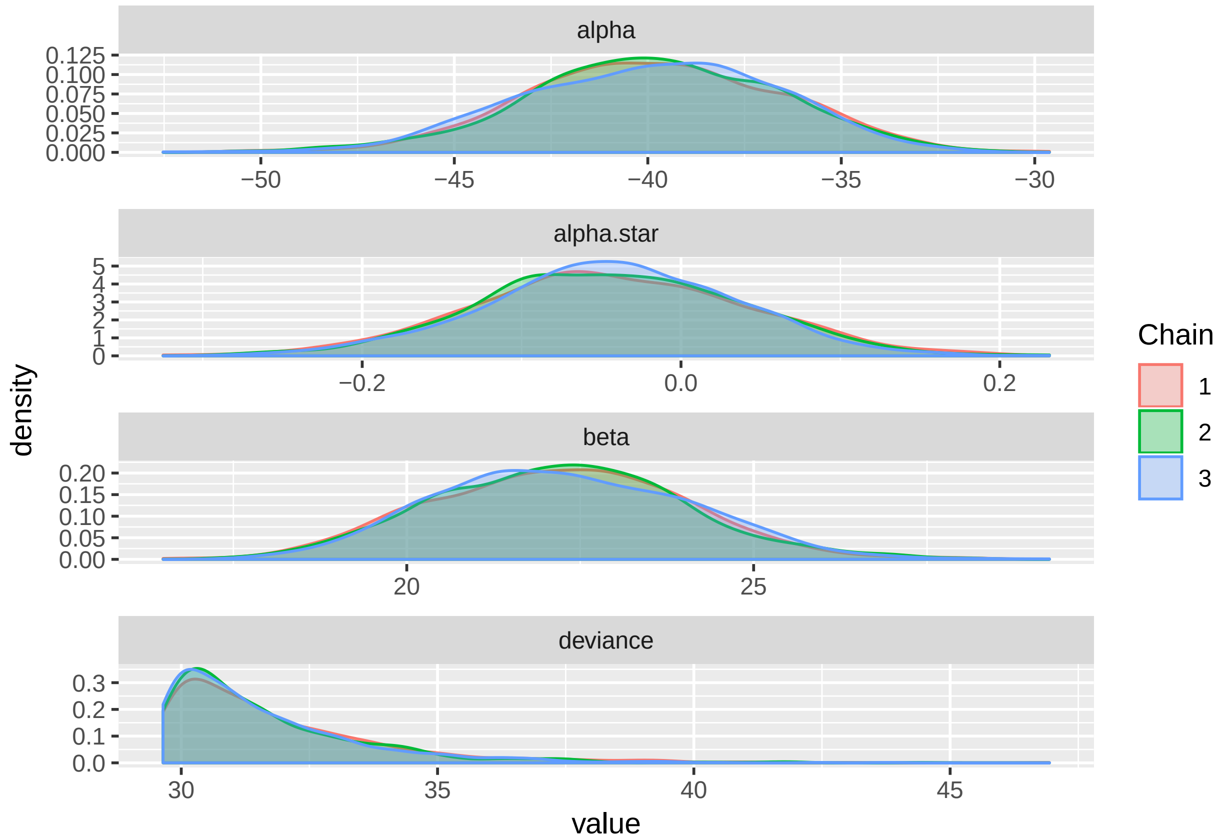



```r
ggs_autocorrelation(chain_print, nLags = 30)
```



The plots of the chains don't seem to follow any pattern, the parameters mean seems to converge and the autocorrelation has values very close to zero after the first few iterations: all this indicates that the chains converges properly. Finally, let's have a look at the estimated density of the estimators:

```
ggs_density(chain_print)
```



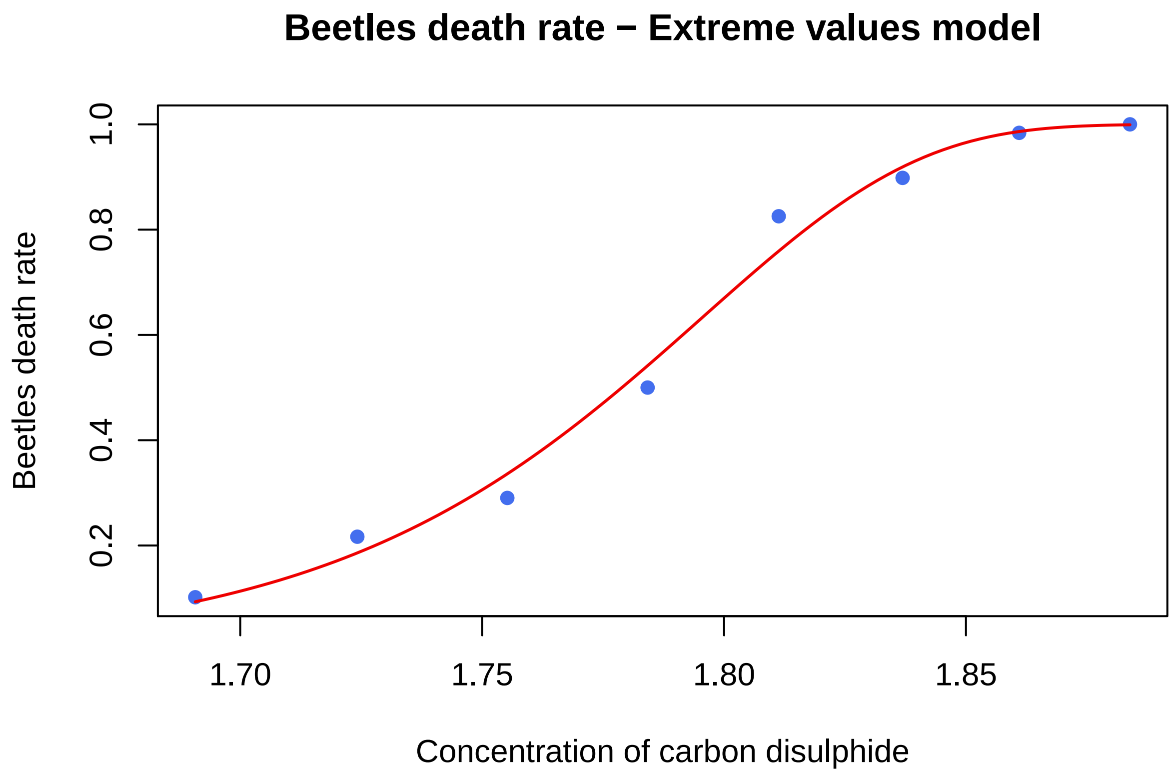
Smooth, unimodal and almost identical between the chains, we can be confident in saying that all the chains converges.

Let's have now a look at the goodness of fit of our estimate:

```
print(mcmc_res$BUGSoutput$summary[1:4,c(1:3,7)])
```

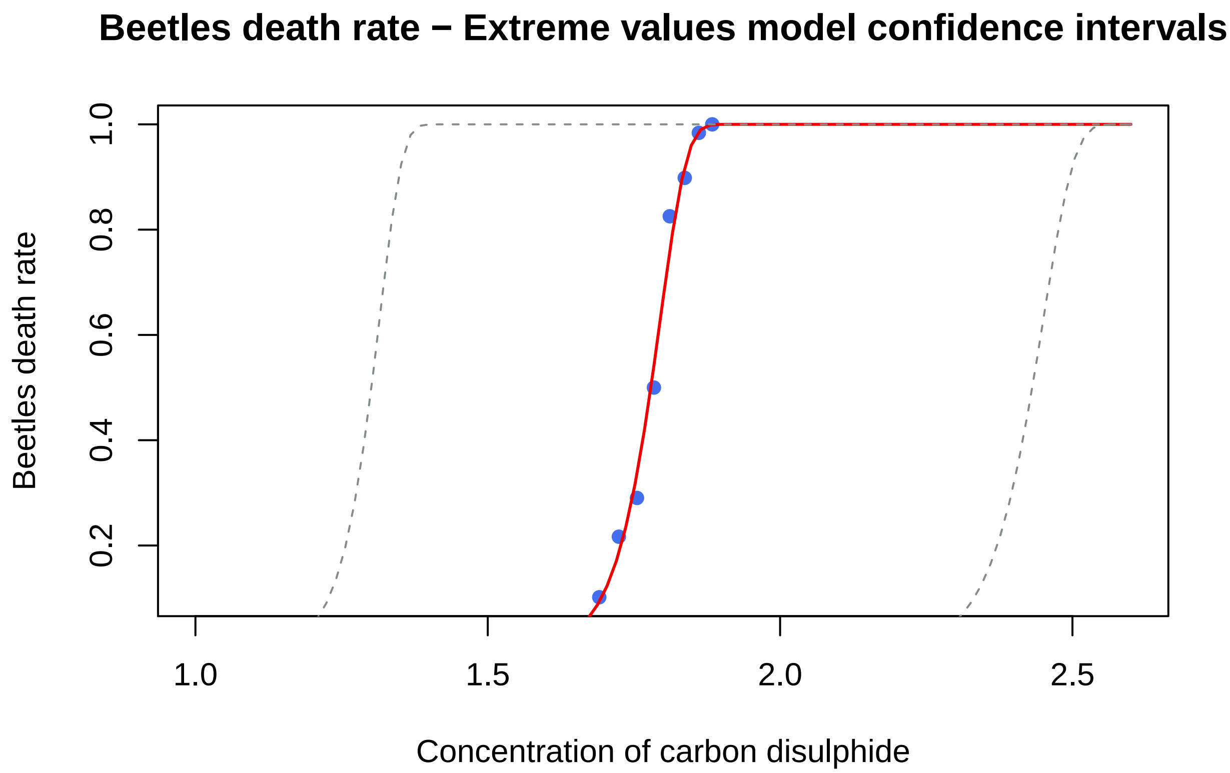
```
##                   mean         sd       2.5%       97.5%
## alpha      -39.87974197 3.25428612 -46.3453158 -33.9393451
## alpha.star  -0.04365802 0.08170396  -0.2031907   0.1136823
## beta        22.21229433 1.80758010  18.9117051  25.8057077
## deviance    31.69079795 2.00252570  29.7050936  37.1199464
```

```
alpha <-as.numeric(mcmc_res$BUGSoutput$mean$alpha)
beta <- as.numeric(mcmc_res$BUGSoutput$mean$beta)

plot(x = data$x, y = (data$r/data$n),
     main = "Beetles death rate - Extreme values model",
     xlab = "Concentration of carbon disulphide",
     ylab = "Beetles death rate",
     pch = 16, col = "royalblue2")

cloglog <- function(x, a, b) return(1-exp(-exp(a+b * x)))

curve(cloglog(x, alpha, beta), add = T, col = 'red2', lwd = 1.5)
```



As we did before, we want to check also the confidence intervals:

```
alpha25 <-as.numeric(mcmc_res$BUGSoutput$summary[1,3])
beta25 <- as.numeric(mcmc_res$BUGSoutput$summary[3,3])
alpha975 <-as.numeric(mcmc_res$BUGSoutput$summary[1,7])
beta975 <- as.numeric(mcmc_res$BUGSoutput$summary[3,7])

plot(x = data$x, y = (data$r/data$n),
     main = "Beetles death rate - Extreme values model confidence intervals",
     xlab = "Concentration of carbon disulphide",
     ylab = "Beetles death rate",
     pch = 16, col = "royalblue2", xlim = c(1,2.6))

curve(cloglog(x, alpha, beta), add = T, col = 'red2', lwd = 1.5)
curve(cloglog(x, alpha25, beta25), add = T, lwd = 1, lty = 2, col = 'azure4')
curve(cloglog(x, alpha975, beta975), add = T, lwd = 1, lty = 2, col = 'azure4')
```



Comparison with frequentist inference

When we want to compare the different results between frequentist approach and Bayesian approach, we have to keep in mind first the differences between the two philosophies of the approaches. In the frequentist approach we use the likelihood function to get $P(D|\theta, M)$, meanwhile in the Bayesian approach we are looking for the probability of the model parameters $P(\theta|D, M)$. So we can say that the frequentists operate on the probability of the data, while the Bayesian operate on a probability of the model parameters. So what is done in the frequentist approach, is to compute the likelihood directly from the model, and by optimizinge this likelihood expression directly, it's possible to arrive a t an estimated best model fit. Bayesian, on the other hand, have a slightly more difficult task that involves the priors beliefs. In situations where the model doesn't belong to twell known parametric families, it's much difficult to evaluate the simple formula of the postirior, so we have to use more advanced techniques, such as Markov Chain Monte Carlo.

To arrange the comparison, the first step to do is to find the optimal parameters of the likelihood (Maximum Likelihood Estimation), as required in the frequentist approach. We will use the GLM (General Linear Methods) library built-in in R:

```
logit_freq <- glm(Rate ~ Concentration, family = binomial(link = "logit"), data = df)
probit_freq <- glm(Rate ~ Concentration, family = binomial(link = "probit"), data = df)
cloglog_freq <- glm(Rate ~ Concentration, family = binomial(link = "cloglog"), data = df)
```

After we prepared the models, it's time to have a look at the results. First we want to observe how much the frequentist results differs from the bayesian ones:

```
toprint = as.data.frame(list("Concentration" = data$x,
                        "Number_of_beetles" = data$n,
                        "Number_killed" = data$r))
```

```r
comp_res <- as.data.frame(matrix(as.numeric(list(
                              logit_res$BUGSoutput$mean$alpha,
                              logit_res$BUGSoutput$mean$beta,
                              logit_freq$coefficients[1],
                              logit_freq$coefficients[2],
                              probit_res$BUGSoutput$mean$alpha,
                              probit_res$BUGSoutput$mean$beta,
                              probit_freq$coefficients[1],
                              probit_freq$coefficients[2],
                              cloglog_res$BUGSoutput$mean$alpha,
                              cloglog_res$BUGSoutput$mean$beta,
                              cloglog_freq$coefficients[1],
                              cloglog_freq$coefficients[2])), nrow = 2, ncol = 6))
names(comp_res) <- rep(c('Post.', 'MLE'), 3)
row.names(comp_res) <- c('alpha', 'beta')

kable(comp_res, "latex", booktabs = T) %>%
  kable_styling(latex_options =c("striped", "hold_position")) %>%
  add_header_above(c(" ", "Logistic" = 2, "Probit" = 2, "Extreme values" = 2))
```

	Logistic		Probit		Extreme values	
	Post.	MLE	Post.	MLE	Post.	MLE
alpha	-61.19868	-60.45910	-35.17028	-34.80375	-39.87974	-39.53294
beta	34.54274	34.12134	19.86110	19.65214	22.21229	22.01742

What we can see is that, for all the three models, the results between bayesian and frequentist approaches do not differ from each other. In fact it's not possible to see any major difference between the results, which does not give us space to have a toughtful considerations on the differences in between these two approaches, in this specific case. What we can do for the frequentist approach (and we will do the same also for the bayesian in the next section) is to compare the goodness of fit of the three models. To do so, we will use the Akaike information criterion (AIC), already computed by default from the *glm* function in R.

The AIC is an estimator of the relative quality of statistical models for a given set of data. The formula to compute it is the following:

$$AIC = 2k - 2ln(\hat{L})$$

where k is the number of estimated parameters of the model, and $\hat{L}$ is the maximum value of the log-likelihood. Hence, a good model must have a good trade-off between these two values. Obviously, the lowest the AIC, the better the model. It is important to remember that, unlike adjusted R-squared, the number itself is not meaningful. So it is useful for comparing models, but it is not interpretable on its own. What we are looking for is the model with the lowest AIC, which is indicating the best fitting model:

```r
toprint <- as.data.frame(matrix(
                          as.numeric(
                            list(
                              logit_freq$aic,
                              probit_freq$aic,
                              cloglog_freq$aic)),
                          nrow = 3, ncol =1))
names(toprint) <- "AIC models comparison"
row.names(toprint) <- c('Logistic model: ', 'Probit model: ', "Extreme values model: ")
kable(toprint, "latex", booktabs = T) %>%
  kable_styling(latex_options =c("striped", "hold_position"))
```

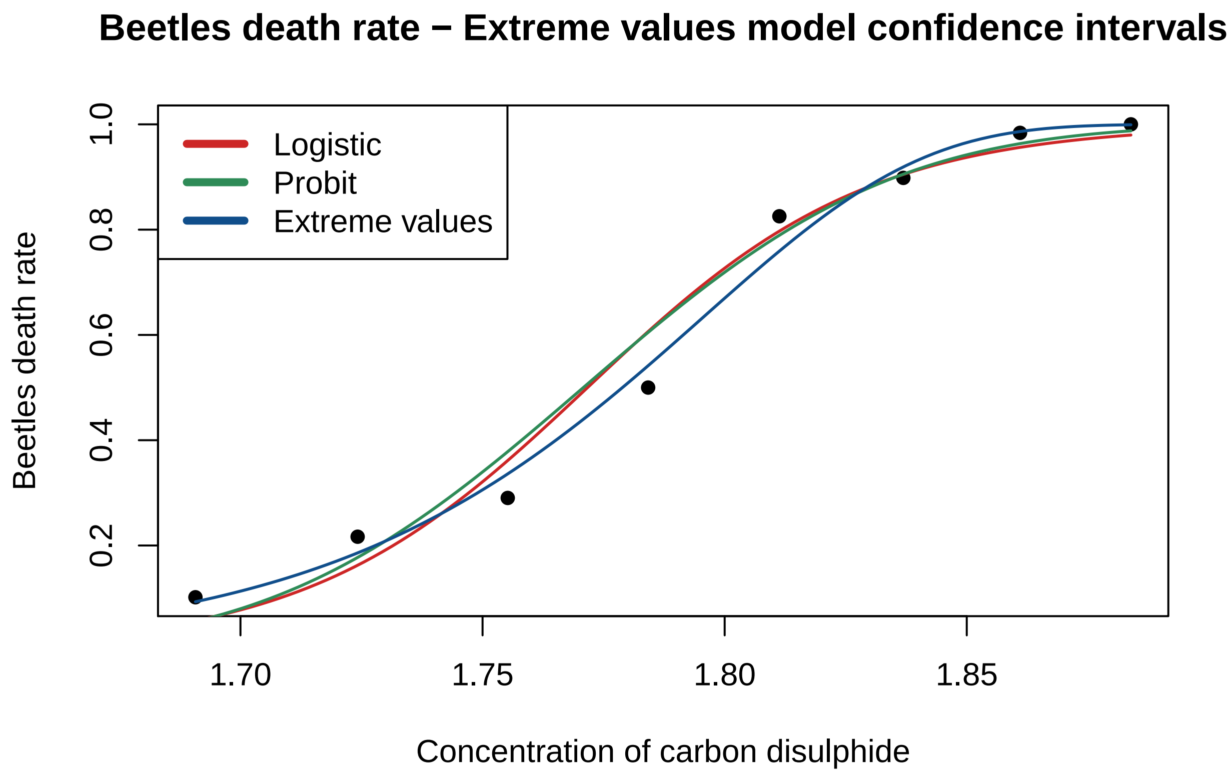
	AIC models comparison
Logistic model:	8.033914
Probit model:	8.099567
Extreme values model:	7.757714

As is possible to observe, al the three AIC values seems to be fairly close to each other, having the Extreme Values model with the lowest value, which indicates that for this data, it seems the best fitting model.

Bayesian model comparison

Once we did the comparison between frequentist and bayesian worlds (and inside the frequentist itself), it is time to compare the models with the parameters we obtained through bayesian inference. We can start by plotting all the models together, to have a visual comprehension of them:

```
comp_res <- as.data.frame(matrix(as.numeric(list(
                                logit_res$BUGSoutput$mean$alpha,
                                logit_res$BUGSoutput$mean$beta,
                                probit_res$BUGSoutput$mean$alpha,
                                probit_res$BUGSoutput$mean$beta,
                                cloglog_res$BUGSoutput$mean$alpha,
                                cloglog_res$BUGSoutput$mean$beta)), nrow = 2, ncol = 3))
names(comp_res) <- c('logit', 'probit', "cloglog")
row.names(comp_res) <- c('alpha', 'beta')


plot(x = data$x, y = (data$r/data$n),
     main = "Beetles death rate - Extreme values model confidence intervals",
     xlab = "Concentration of carbon disulphide",
     ylab = "Beetles death rate", col = 'black',
     pch = 16)

curve(logit(x, comp_res['alpha','logit'], comp_res['beta','logit']), add = T,
      col = 'firebrick3', lwd = 1.5)
curve(pnorm(comp_res['alpha','probit'] + (comp_res['beta','probit'] * x), mean = 0,
            sd = 1, lower.tail = TRUE), add = T, col = 'seagreen4', lwd = 1.5)
curve(cloglog(x, comp_res['alpha','cloglog'], comp_res['beta','cloglog']), add = T,
      col = 'dodgerblue4', lwd = 1.5)
legend(1, 95, legend=c("Logistic", "Probit", "Extreme values"),
       col=c("firebrick3", "seagreen4", "dodgerblue4"), lty=1:2, cex=0.8)
legend("topleft", legend=c("Logistic", "Probit", "Extreme values"),
       col=c("firebrick3", "seagreen4", "dodgerblue4"),  lty=1, lwd = 4)
```



As we can observe, from the plot itself it is very difficult to identify which model fits the best our data. In this case, as we did before in the frequentist case, we will need to use an Information Criteria to evaluate how well the model fit the data, in this case the DIC.
The deviance information criterion (DIC) was introduced by *Spiegelhalter et al. (2002)* as a measure of model comparison and adequacy. It is given by the expression

$$DIC(m) = 2\overline{D(\theta_m, m)} - D(\overline{\theta}_m, m) = D(\overline{\theta}_m, m) + 2p_m$$

where $D(\overline{\theta}_m, m)$ is the usual deviance measure, which is equal to minus twice the log-likelihood:

$$D(\theta_m, m) = -2log\, f(y|\theta_m, m)$$

and $\overline{D(\theta_m, m)}$ is its posterior mean, p_m, can be interpeted as the number of "effective" parameters form model m given by

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

and $\overline{\theta}_m$ is the posterior mean of the parameters involved in the model m.
So we will have that smalle DIC values indicate a better-fittinf model. DIC must be used with caution since it assumes that the posterior mean can be used as a "good" summary of central location for description of the posterior distribution.
Generally DIC is considered as a generalization of AIC. In simple one-stage models, AIC and DIC are identical. But, like in our case, differences occur in hierarchical and latent variable models where DIC uses the number of "effective" parameters instead of the actual number of parameters used by AIC. This is the reason why, in this kind of analysis was not possible for us to compare the AIC results of the frequentist analysis with the DIC results we will compare now, from the bayesian analysis. So again, as we saw previously with the AIC, the idea is that the model with a smaller DIC is preferable to a model having a larger DIC. Also here, the concept is to have a good trade-off between number of parametrs to estimate, and the deviance of the model (instead the likelihood, as in the AIC).
Let's have a look at the DIC values we obtained from the run of the chains:

```r
toprint <- as.data.frame(matrix(
                          as.numeric(
                            list(
                              logit_res$BUGSoutput$DIC,
                              probit_res$BUGSoutput$DIC,
                              cloglog_res$BUGSoutput$DIC)),
                          nrow = 3, ncol =1))
names(toprint) <- "DIC models comparison"
row.names(toprint) <- c('Logistic model: ', 'Probit model: ', "Extreme values model: ")
kable(toprint, "latex", booktabs = T) %>%
  kable_styling(latex_options =c("striped", "hold_position"))
```

	DIC models comparison
Logistic model:	126.86158
Probit model:	40.42298
Extreme values model:	33.69169

As is possible to see, the model that seems to have the best goodness of fit is again the extreme values model. This match the results of the frequentist analysis, confirming that the extreme values model is the one that is most suitable for our data, probably due to a smaller discrepancy between observed and predicted values.

Since we want to go a little bit deeper with the comparison, it is useful to have a comparison done by generating new data from a new distribution proposed by us. The models will then be tested again, in light of the new data, to see if they can recover the structure of the resampled data. We do this with the aim to understand if the performances of the proposed models are due to randomness or the model can generalize well any combination of parameters.

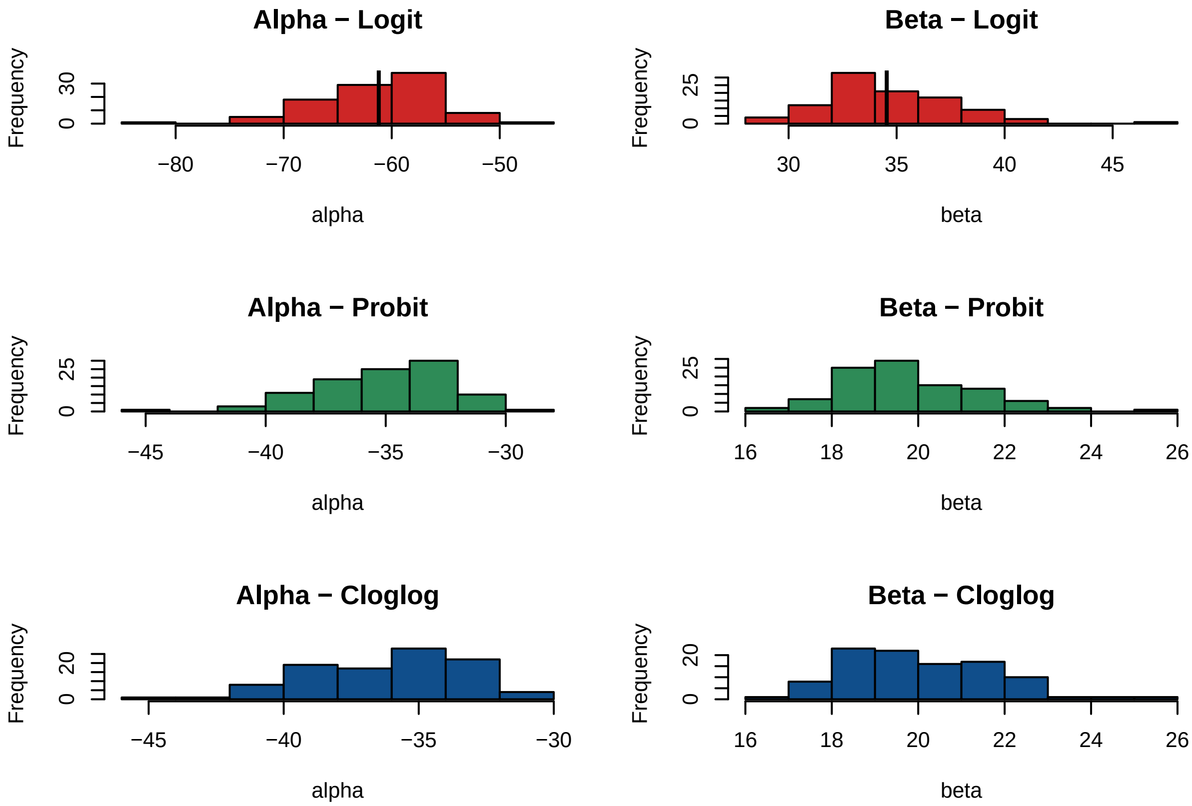
The first step taken was the generation of new dataset, from existing models: we generated 300 different datasets, and for each, all the three model proposed where fitted. The first 100 datasets are generated from a logistic model, another batch of 100 datasets using the probit model and the last 100 datasets using the extreme values model.

Let's start with the logit model:

```r
n_data = 100
res <- matrix(rep(NA,n_data*3*3), nrow = n_data, ncol = 9)
colnames(res) <- c('logit_alpha','logit_beta','logit_DIC',
                   'probit_alpha','probit_beta','probit_DIC',
                   'cloglog_alpha','cloglog_beta','cloglog_DIC')
a <- logit_res$BUGSoutput$mean$alpha
b <- logit_res$BUGSoutput$mean$beta

for(i in 1:n_data){
  y_new = rbinom(length(data$n), data$n, logit(data$x, a, b))

  data.list= list(r = y_new,
                n = data$n,
                x = data$x,
                tau_alpha = tau_alpha,
                tau_beta = tau_beta,
                N = 8)

  mcmc_res = jags( model.file = textConnection(model_1),
                   data = data.list,
```

```
                  n.chains = 3,
                  n.iter = 11000,
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta"))

  res[i, 'logit_alpha'] = mcmc_res$BUGSoutput$mean$alpha
  res[i, 'logit_beta'] = mcmc_res$BUGSoutput$mean$beta
  res[i, 'logit_DIC'] = mcmc_res$BUGSoutput$DIC

  mcmc_res = jags( model.file = textConnection(model_2),
                  data = data.list,
                  n.chains = 3,
                  n.iter = 11000,
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta"))

  res[i, 'probit_alpha'] = mcmc_res$BUGSoutput$mean$alpha
  res[i, 'probit_beta'] = mcmc_res$BUGSoutput$mean$beta
  res[i, 'probit_DIC'] = mcmc_res$BUGSoutput$DIC

  mcmc_res = jags( model.file = textConnection(model_3),
                  data = data.list,
                  n.chains = 3,
                  n.iter = 11000,
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta"))

  res[i, 'cloglog_alpha'] = mcmc_res$BUGSoutput$mean$alpha
  res[i, 'cloglog_beta'] = mcmc_res$BUGSoutput$mean$beta
  res[i, 'cloglog_DIC'] = mcmc_res$BUGSoutput$DIC
}
```

```
par(mfrow = c(3,2))
hist(res[, 'logit_alpha'], main = "Alpha - Logit", col = "firebrick3", xlab = 'alpha')
abline(v = a, lwd = 2)
hist(res[, 'logit_beta'], main = "Beta - Logit", col = "firebrick3", xlab = 'beta')
abline(v = b, lwd = 2)

hist(res[, 'probit_alpha'], main = "Alpha - Probit", col = "seagreen4", xlab = 'alpha')
abline(v = a, lwd = 2)
hist(res[, 'probit_beta'], main = "Beta - Probit", col = "seagreen4", xlab = 'beta')
abline(v = b, lwd = 2)

hist(res[, 'cloglog_alpha'], main = "Alpha - Cloglog", col = "dodgerblue4", xlab = 'alpha')
abline(v = a, lwd = 2)
hist(res[, 'cloglog_beta'], main = "Beta - Cloglog", col = "dodgerblue4", xlab = 'beta')
abline(v = b, lwd = 2)
```




```r
kable(summary(res[, c('logit_DIC','probit_DIC', 'cloglog_DIC')]), 'latex', booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

logit_DIC	probit_DIC	cloglog_DIC
Min. : 97.02	Min. :31.91	Min. :30.12
1st Qu.:116.31	1st Qu.:35.05	1st Qu.:39.45
Median :124.18	Median :36.96	Median :43.84
Mean :125.06	Mean :38.18	Mean :45.62
3rd Qu.:134.47	3rd Qu.:41.13	3rd Qu.:50.73
Max. :159.03	Max. :53.22	Max. :69.83

```r
true_alpha = rep(a, 100)
true_beta = rep(b, 100)
rmse_logit = c(RMSE(res[,'logit_alpha'],true_alpha),RMSE(res[,'logit_beta'],true_beta))
rmse_probit = c(RMSE(res[,'probit_alpha'],true_alpha),RMSE(res[,'probit_beta'],true_beta))
rmse_cloglog = c(RMSE(res[,'cloglog_alpha'],true_alpha),RMSE(res[,'cloglog_beta'],true_beta))
toprint = as.data.frame(cbind(rmse_logit, rmse_probit, rmse_cloglog))
names(toprint) <- c('Logistic', 'Probit', 'Extreme values')
row.names(toprint) <- c('Alpha', 'Beta')

kable(toprint, "latex", booktabs = T) %>%
  kable_styling(latex_options =c("striped", "hold_position"))
```

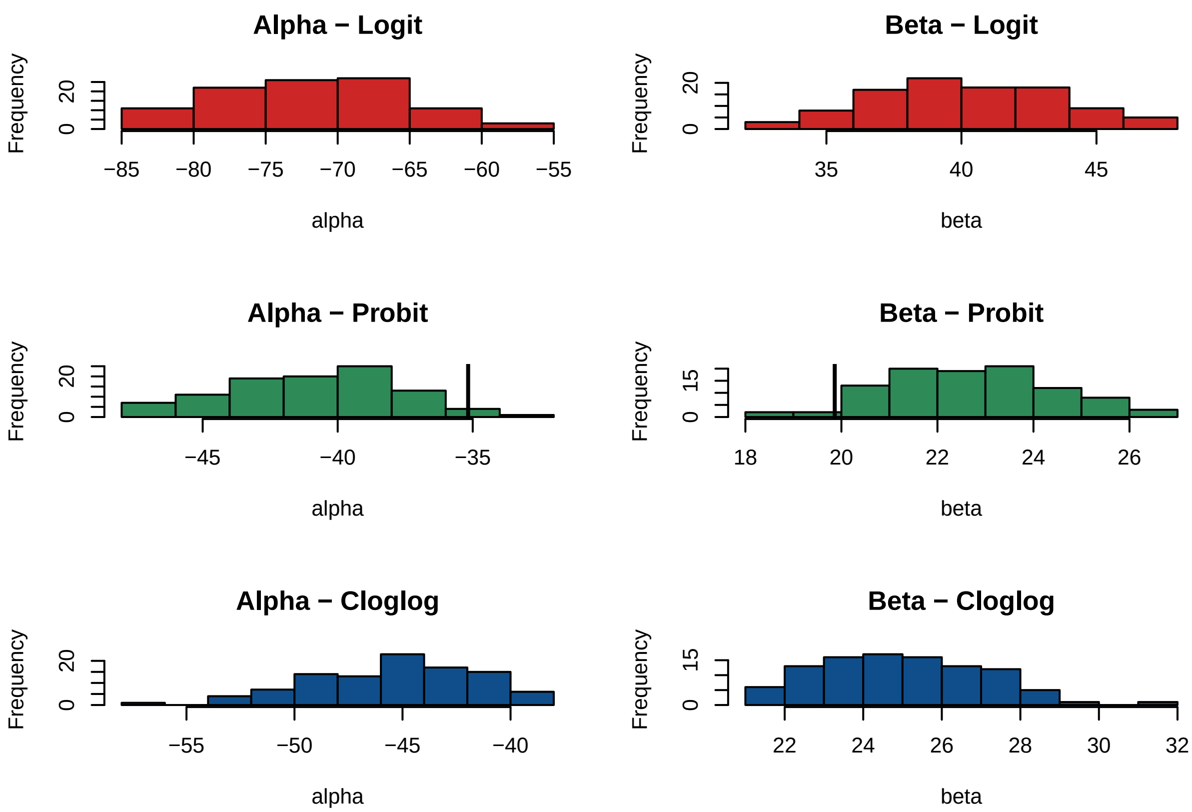
Let's have a look also at the probit and extreme values models, before proceeding with the commentary. The next model then is the probit:

	Logistic	Probit	Extreme values
Alpha	5.410247	26.34490	25.34154
Beta	3.036789	14.88004	14.59721

```
n_data = 100
res <- matrix(rep(NA,n_data*3*3), nrow = n_data, ncol = 9)
colnames(res) <- c('logit_alpha','logit_beta','logit_DIC',
                   'probit_alpha','probit_beta','probit_DIC',
                   'cloglog_alpha','cloglog_beta','cloglog_DIC')
a <- probit_res$BUGSoutput$mean$alpha
b <- probit_res$BUGSoutput$mean$beta

for(i in 1:n_data){

  y_new = rbinom(length(data$n), data$n, pnorm(alpha + (beta * data$x), mean = 0, sd = 1, lower.tail = '


  data.list= list(r = y_new,
                n = data$n,
                x = data$x,
                tau_alpha = tau_alpha,
                tau_beta = tau_beta,
                N = 8)

  mcmc_res = jags( model.file = textConnection(model_1),
                  data = data.list,
                  n.chains = 3,
                  n.iter = 11000,
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta"))

  res[i, 'logit_alpha'] = mcmc_res$BUGSoutput$mean$alpha
  res[i, 'logit_beta'] = mcmc_res$BUGSoutput$mean$beta
  res[i, 'logit_DIC'] = mcmc_res$BUGSoutput$DIC

  mcmc_res = jags( model.file = textConnection(model_2),
                  data = data.list,
                  n.chains = 3,
                  n.iter = 11000,
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta"))

  res[i, 'probit_alpha'] = mcmc_res$BUGSoutput$mean$alpha
  res[i, 'probit_beta'] = mcmc_res$BUGSoutput$mean$beta
  res[i, 'probit_DIC'] = mcmc_res$BUGSoutput$DIC

    mcmc_res = jags( model.file = textConnection(model_3),
                  data = data.list,
                  n.chains = 3,
                  n.iter = 11000,
```

```
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta"))

  res[i, 'cloglog_alpha'] = mcmc_res$BUGSoutput$mean$alpha
  res[i, 'cloglog_beta'] = mcmc_res$BUGSoutput$mean$beta
  res[i, 'cloglog_DIC'] = mcmc_res$BUGSoutput$DIC
}
```

```
par(mfrow = c(3,2))
hist(res[, 'logit_alpha'], main = "Alpha - Logit", col = "firebrick3", xlab = 'alpha')
abline(v = a, lwd = 2)
hist(res[, 'logit_beta'], main = "Beta - Logit", col = "firebrick3", xlab = 'beta')
abline(v = b, lwd = 2)

hist(res[, 'probit_alpha'], main = "Alpha - Probit", col = "seagreen4", xlab = 'alpha')
abline(v = a, lwd = 2)
hist(res[, 'probit_beta'], main = "Beta - Probit", col = "seagreen4", xlab = 'beta')
abline(v = b, lwd = 2)

hist(res[, 'cloglog_alpha'], main = "Alpha - Cloglog", col = "dodgerblue4", xlab = 'alpha')
abline(v = a, lwd = 2)
hist(res[, 'cloglog_beta'], main = "Beta - Cloglog", col = "dodgerblue4", xlab = 'beta')
abline(v = b, lwd = 2)
```



```r
kable(summary(res[, c('logit_DIC','probit_DIC', 'cloglog_DIC')]), 'latex', booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

logit_DIC	probit_DIC	cloglog_DIC
Min. :157.6	Min. :30.19	Min. :31.91
1st Qu.:187.3	1st Qu.:34.55	1st Qu.:39.38
Median :200.3	Median :36.19	Median :43.62
Mean :200.4	Mean :36.57	Mean :43.47
3rd Qu.:214.2	3rd Qu.:38.62	3rd Qu.:46.83
Max. :241.8	Max. :46.39	Max. :61.80

```r
true_alpha = rep(a, 100)
true_beta = rep(b, 100)
rmse_logit = c(RMSE(res[,'logit_alpha'],true_alpha),RMSE(res[,'logit_beta'],true_beta))
rmse_probit = c(RMSE(res[,'probit_alpha'],true_alpha),RMSE(res[,'probit_beta'],true_beta))
rmse_cloglog = c(RMSE(res[,'cloglog_alpha'],true_alpha),RMSE(res[,'cloglog_beta'],true_beta))
toprint = as.data.frame(cbind(rmse_logit, rmse_probit, rmse_cloglog))
names(toprint) <- c('Logistic', 'Probit', 'Extreme values')
row.names(toprint) <- c('Alpha', 'Beta')

kable(toprint, "latex", booktabs = T) %>%
  kable_styling(latex_options =c("striped", "hold_position"))
```

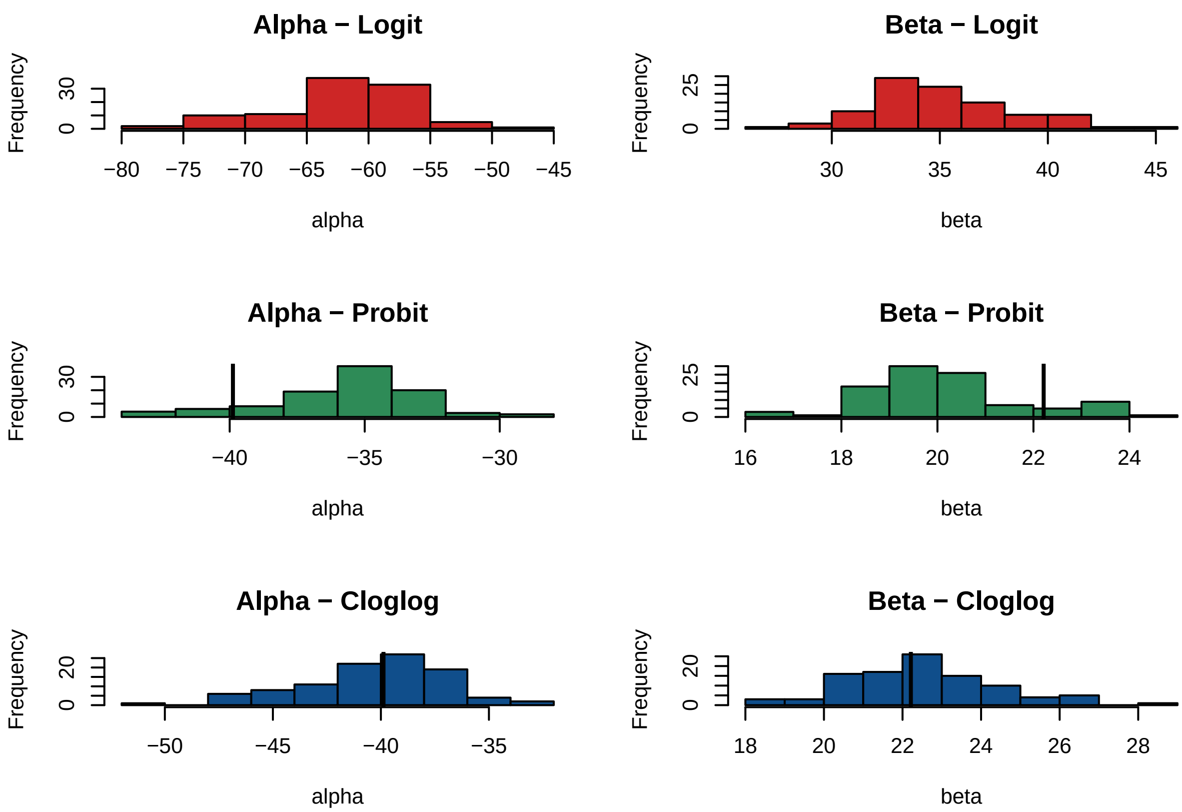
	Logistic	Probit	Extreme values
Alpha	37.37204	6.472876	10.910925
Beta	20.53763	3.365092	5.521215

And finally we have a look also at the extreme values results:

```r
n_data = 100
res <- matrix(rep(NA,n_data*3*3), nrow = n_data, ncol = 9)
colnames(res) <- c('logit_alpha','logit_beta','logit_DIC',
                   'probit_alpha','probit_beta','probit_DIC',
                   'cloglog_alpha','cloglog_beta','cloglog_DIC')
a <- cloglog_res$BUGSoutput$mean$alpha
b <- cloglog_res$BUGSoutput$mean$beta

for(i in 1:n_data){

  y_new = rbinom(length(data$n), data$n, cloglog(data$x, a, b))

  data.list= list(r = y_new,
              n = data$n,
              x = data$x,
              tau_alpha = tau_alpha,
              tau_beta = tau_beta,
              N = 8)

  mcmc_res = jags( model.file = textConnection(model_1),
               data = data.list,
               n.chains = 3,
               n.iter = 11000,
```

```
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta"))

  res[i, 'logit_alpha'] = mcmc_res$BUGSoutput$mean$alpha
  res[i, 'logit_beta'] = mcmc_res$BUGSoutput$mean$beta
  res[i, 'logit_DIC'] = mcmc_res$BUGSoutput$DIC

  mcmc_res = jags( model.file = textConnection(model_2),
                  data = data.list,
                  n.chains = 3,
                  n.iter = 11000,
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta"))

  res[i, 'probit_alpha'] = mcmc_res$BUGSoutput$mean$alpha
  res[i, 'probit_beta'] = mcmc_res$BUGSoutput$mean$beta
  res[i, 'probit_DIC'] = mcmc_res$BUGSoutput$DIC

    mcmc_res = jags( model.file = textConnection(model_3),
                  data = data.list,
                  n.chains = 3,
                  n.iter = 11000,
                  n.burnin = 1000,
                  inits = init.list.nod,
                  parameters.to.save = c("alpha","alpha.star","beta"))

  res[i, 'cloglog_alpha'] = mcmc_res$BUGSoutput$mean$alpha
  res[i, 'cloglog_beta'] = mcmc_res$BUGSoutput$mean$beta
  res[i, 'cloglog_DIC'] = mcmc_res$BUGSoutput$DIC
}
```

```
par(mfrow = c(3,2))
hist(res[, 'logit_alpha'], main = "Alpha - Logit", col = "firebrick3", xlab = 'alpha')
abline(v = a, lwd = 2)
hist(res[, 'logit_beta'], main = "Beta - Logit", col = "firebrick3", xlab = 'beta')
abline(v = b, lwd = 2)

hist(res[, 'probit_alpha'], main = "Alpha - Probit", col = "seagreen4", xlab = 'alpha')
abline(v = a, lwd = 2)
hist(res[, 'probit_beta'], main = "Beta - Probit", col = "seagreen4", xlab = 'beta')
abline(v = b, lwd = 2)

hist(res[, 'cloglog_alpha'], main = "Alpha - Cloglog", col = "dodgerblue4", xlab = 'alpha')
abline(v = a, lwd = 2)
hist(res[, 'cloglog_beta'], main = "Beta - Cloglog", col = "dodgerblue4", xlab = 'beta')
abline(v = b, lwd = 2)
```




```r
kable(summary(res[, c('logit_DIC','probit_DIC', 'cloglog_DIC')]), 'latex', booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

logit_DIC	probit_DIC	cloglog_DIC
Min. :105.3	Min. :32.20	Min. :29.55
1st Qu.:120.7	1st Qu.:37.32	1st Qu.:32.51
Median :128.2	Median :39.81	Median :34.86
Mean :128.7	Mean :40.37	Mean :35.18
3rd Qu.:135.7	3rd Qu.:42.72	3rd Qu.:37.43
Max. :174.1	Max. :54.36	Max. :47.09

```r
true_alpha = rep(a, 100)
true_beta = rep(b, 100)
rmse_logit = c(RMSE(res[,'logit_alpha'],true_alpha),RMSE(res[,'logit_beta'],true_beta))
rmse_probit = c(RMSE(res[,'probit_alpha'],true_alpha),RMSE(res[,'probit_beta'],true_beta))
rmse_cloglog = c(RMSE(res[,'cloglog_alpha'],true_alpha),RMSE(res[,'cloglog_beta'],true_beta))
toprint = as.data.frame(cbind(rmse_logit, rmse_probit, rmse_cloglog))
names(toprint) <- c('Logistic', 'Probit', 'Extreme values')
row.names(toprint) <- c('Alpha', 'Beta')

kable(toprint, "latex", booktabs = T) %>%
  kable_styling(latex_options =c("striped", "hold_position"))
```

From the tests run above, we can observe few different things: first of all, all the parameters seems to be normally distributed around a value, according to the new data generated, indicating that they are not biased

	Logistic	Probit	Extreme values
Alpha	22.96247	5.057023	3.340278
Beta	13.23530	2.599868	1.850479

by any external factor, but they tend to follow the normality curve given from the generated x. The second consideration is that each model seems the best to recover the data it generated: this is an expected but good news, since it means that the models can properly reconstruct the random data generated from them, and it is always that model and that model only that gets the best Root Mean Square Error. This is also confirmed graphically by noticing that for each model, the average parameter alpha and beta is always around the parameters we obtained before running the test, which is the same parameters we used to generate the data. This once again confirms that each model is the best to predict himself, which seems logical if the model is not biased by any external factor. Finally, there's the Deviance Information Criteria: as said previously, it measure the general good fitness of the data. In this case, we can observe that the DIC is always against the logistic model, and changes in favor of the probit or extreme values model according to the model the data has been generated from. This result is probably given by the fact that the probit and cloglog model seems to recover better the very low values, being more flexible than the logistic model.

Finally, before going to the conclusion, we can check the ability from the cloglog to recover better the data with respect from the other two models, simply by comparing the actual and predicted $\hat{R}$:

```
toprint = as.data.frame(cbind(data$r,
                              logit_res$BUGSoutput$summary[5:12,'mean'],
                              probit_res$BUGSoutput$summary[5:12,'mean'],
                              cloglog_res$BUGSoutput$summary[5:12,'mean']))
names(toprint) <- c('Actual', 'Logistic', 'Probit', 'Extreme values')
kable(toprint, 'latex', booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

	Actual	Logistic	Probit	Extreme values
rhat[1]	6	3.537229	3.407764	5.600726
rhat[2]	13	9.877754	10.722038	11.255768
rhat[3]	18	22.449003	23.475872	20.899300
rhat[4]	28	33.932623	33.854934	30.343186
rhat[5]	52	50.134532	49.670453	47.799016
rhat[6]	53	53.284498	53.329939	54.126660
rhat[7]	61	59.187887	59.638069	61.043368
rhat[8]	60	58.704275	59.194187	59.919286

So as it is possible to observe, where all the models are quite good in predicting high values, with little difference between extreme values model and probit model, and slightly worst performance by the logistic model, where the extreme values model manage to make the difference between the three is in the lowest values. This means the extreme values model manages to recover better the extreme values of the underlying function. This information can be summed up also by just having a look at the deviance values of the three models:

```
toprint = as.data.frame(cbind(logit_res$BUGSoutput$summary['deviance', 'mean'],
                              probit_res$BUGSoutput$summary['deviance','mean'],
                              cloglog_res$BUGSoutput$summary['deviance','mean']))
names(toprint) <- c('Logistic', 'Probit', 'Extreme values')
kable(toprint, 'latex', booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Logistic	Probit	Extreme values
39.87784	38.31831	31.6908

Conclusions

All the comparison we did indicates that the extreme value model fits the data considerably better than do the logistic or probit models. This appears to be due to a smaller discrepancy between observed (r_i) and fitted ($\hat{r}_i$) values at the lower concentration for extreme value model. Furthermore we observed that there is not a huge difference between the results in the frequentist approach compared to the bayesian approach, with the final result being the same: the extreme value model is the best fitting model.