The analysis of ELISA serial dilution and optical density measurements

By *Huijun Park*

This code is written using JupyterLab with R kernel

Memory Clearance

Make sure the memory is clear at the beginning

In [32]:
```
rm(list=ls()) # All the preloaded variables are removed so that they do not in
terfere with the code to be followed by
```

The version of R used for this code

In [35]:
```
version
```

```
               _
platform       x86_64-w64-mingw32
arch           x86_64
os             mingw32
system         x86_64, mingw32
status
major          3
minor          5.1
year           2018
month          07
day            02
svn rev        74947
language       R
version.string R version 3.5.1 (2018-07-02)
nickname       Feather Spray
```

A brief summary of theory

A dataset "DNase" was elected to be used. The dataset was chosen not only because the underlying mathematical model itself is fairly simple and unequivocally defined but the dilution process used for data acquisition is widely used over many scientific fields including biology and chemistry, which widens the applicability of the model. This is a dataset that is included with the basic R, so it won't be necessary to acquire and curate the data.

ELISA (Enzyme-Linked Immunosorbent Assay) is mainly used for qualititive detection of antigens in sample. Respective antibodies are applied to the sample containing the antigens and they act as ligaments to attach marker chemicals that are easier for the observer to detect through various means such as color or electrical conductivity.

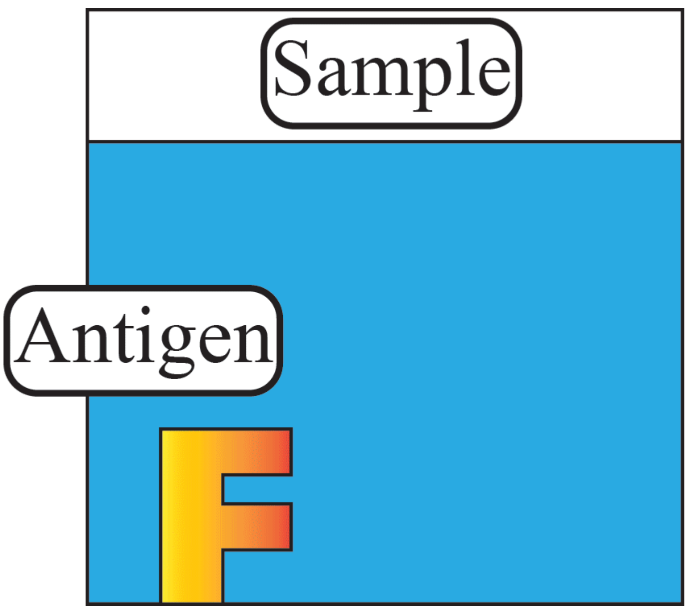
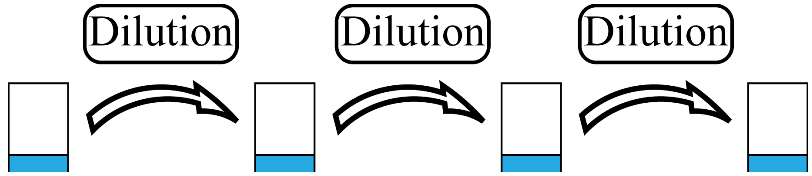


Figure 1. *A marker is applied to make the antigen visible*

In this case the indicator property used is optical density which is usually abbreviated as O.D.. This property measures the opacity of sample by measuring how much of light traveling through the sample reaches the detector. Additionally, when a need for quantitative analysis is engendered, a serial dilution is performed on the sample.



In [3]:
```
data("DNase")
#?DNase # This document includes an RAS syndrome. Can you spot it?
```

Data Exploration

In [2]:
```
head(DNase) # The first n samples
tail(DNase) # The last n samples
```

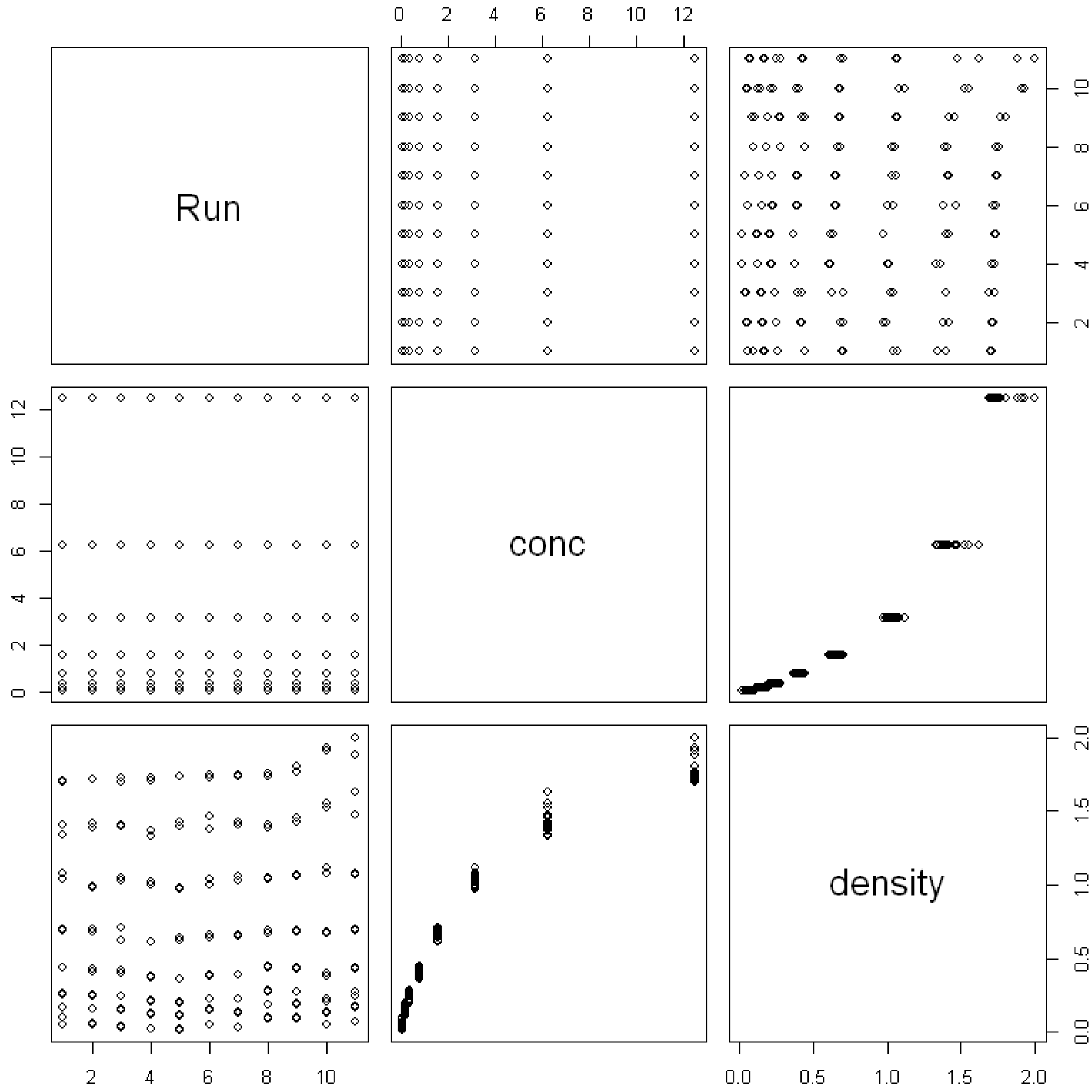
A nfnGroupedData: 6 × 3

Run	conc	density
<ord>	<dbl>	<dbl>
1	0.04882812	0.017
1	0.04882812	0.018
1	0.19531250	0.121
1	0.19531250	0.124
1	0.39062500	0.206
1	0.39062500	0.215

A nfnGroupedData: 6 × 3

	Run	conc	density
	<ord>	<dbl>	<dbl>
171	11	3.125	0.994
172	11	3.125	0.980
173	11	6.250	1.421
174	11	6.250	1.385
175	11	12.500	1.715
176	11	12.500	1.721

In [28]:
```
pairs(DNase)  # Correlation plot
```



It looks like there are 11 "run"s of serial dilution, ordered from 1 to 11. Let us check the nature of the variables "conc" and "density".

In [11]:
```
DNase[1:16,]
```

A nfnGroupedData: 16 × 3

Run	conc	density
<ord>	<dbl>	<dbl>
1	0.04882812	0.017
1	0.04882812	0.018
1	0.19531250	0.121
1	0.19531250	0.124
1	0.39062500	0.206
1	0.39062500	0.215
1	0.78125000	0.377
1	0.78125000	0.374
1	1.56250000	0.614
1	1.56250000	0.609
1	3.12500000	1.019
1	3.12500000	1.001
1	6.25000000	1.334
1	6.25000000	1.364
1	12.50000000	1.730
1	12.50000000	1.710

There are 16 data subsets per one run. In reverse order, the concentration starts from 12.5 and exactly halves every two datasets. From this observation of how they have the exact numbers, it can be safely assumed that the variable "conc" is not an observation of the real concentration but rather an assumed parameter based on calculation. On the other hand, the density variable should be the value that is acquired through experiments to predict the actual concentration. It is likely that the observer measured the O.D. twice per a dilution of sample.

Optical density

Optical density is defined as written below

$A = -\log_{10} T$

Where A denotes the optical density and T denotes the transmittance to be observed

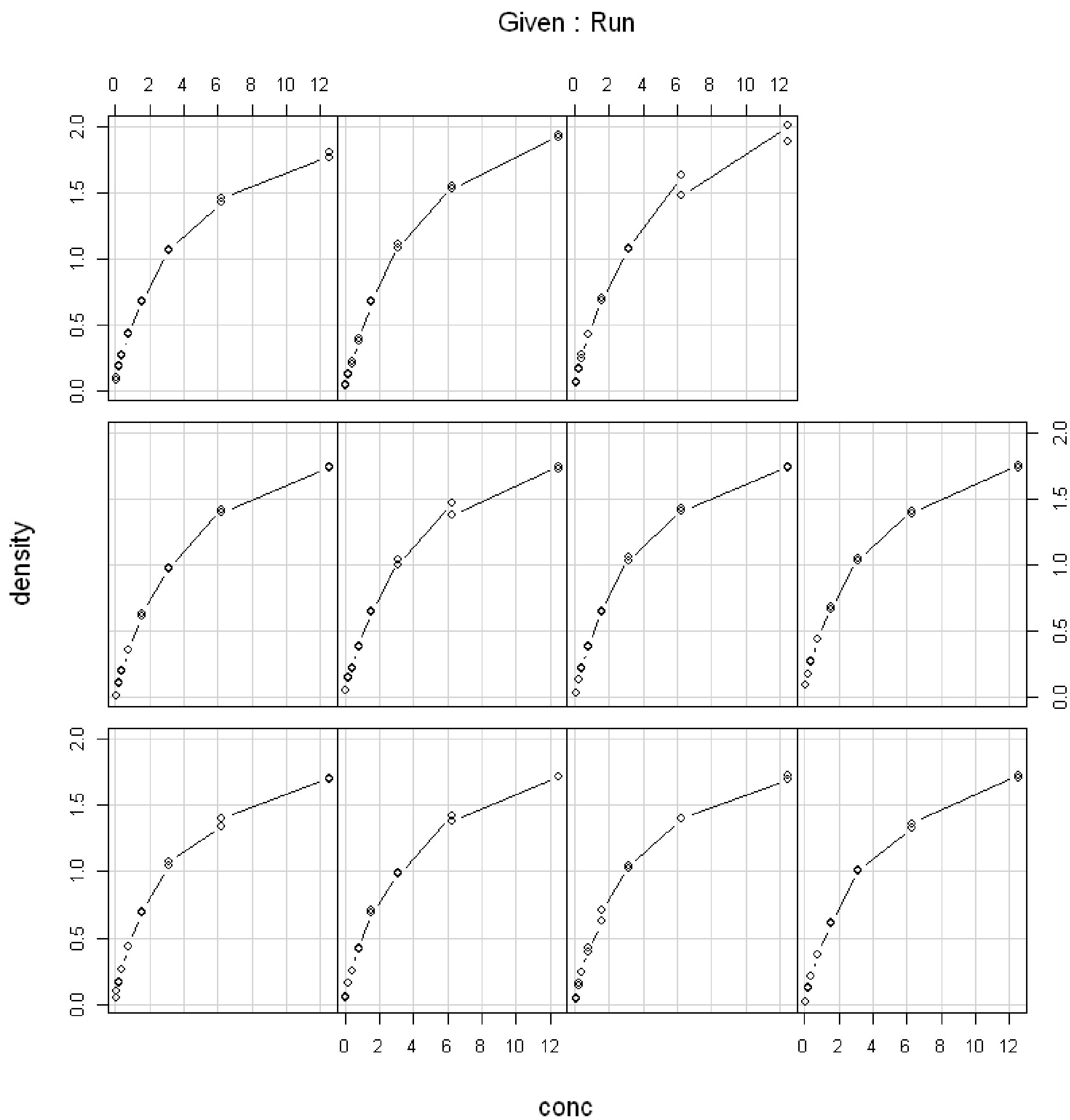
Beer-Lambert law

Beer-Lambert law states that,

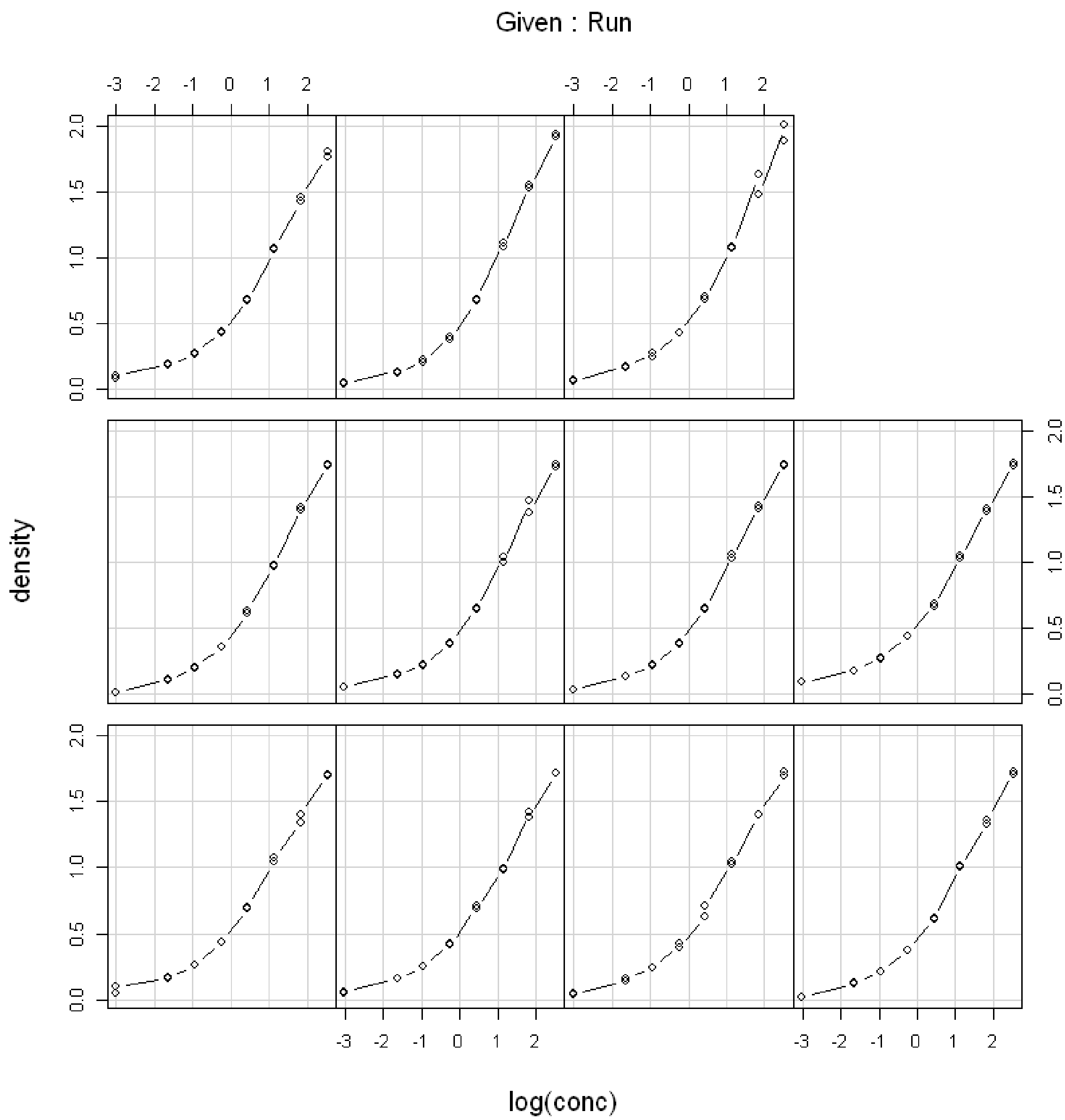
$A = \varepsilon \int c \, \mathrm{d}l$

ε ***is attenuation coefficient unique to the material and*** c ***is molar concentration which is integrated over the optical path. If the sample is homogeneous,*** c ***is constant over the optical path and the optical density*** A ***is proportional to the conecentration*** c

In [25]:
```
# Density ~ conc plot for each runs of serial dilution
coplot(density ~ conc | Run, data = DNase,
       show.given = FALSE, type = "b")
coplot(density ~ log(conc) | Run, data = DNase,
       show.given = FALSE, type = "b")
```



Given : Run
density
conc



Given : Run
density
log(conc)

Optical density seems to lose its linearity at higher concentration. This is possibly not measured with laser based instruments and the interaction from other wavelength is spilling over at high concentration samples. Also, the sample becomes diffuse and multiple scattering affects the photodiode. Regardless of the reason, it is generally recommended to measure the O.D. only between 0.2 and 0.8 because of this nonlinearity problem. However, we can still try to fit a curve to the observed data. The O.D. seems to follow the logarithmic curve at higher concentration. We will dissect this curve into two parts.

Linear curve

The O.D. for the lower concentration regime which follows Beer-Lambert law should be linear.

The O.D. for the higher concentration will be fit to a sigmoid curve since the O.D. should always be positive.

$$A = \begin{cases} a_1 c, & \text{for small } c \\ \frac{a_4}{1+e^{-(a_2 c + a_3)}} + a_5, & \text{for large } c \end{cases}$$

Give weights to each models to combine them into one equation. We will use a reversed sigmoid weight here.

$$w = \frac{1}{1+e^{a_6 c + a_7}}$$

$$A = w a_1 c + (1-w)\left(\frac{a_4}{1+e^{-(a_2 c + a_3)}} + a_5\right)$$

Serial dilution

An each step of dilution can be thought of as a combination of Bernoulli trials of $p = \frac{1}{2}$ ***for all of the antigen particles in the sample. With the number of starting antigen*** n, ***it follows the binomial distribution with mean and variance of*** $np = \frac{n}{2}$, $np(1-p) = \frac{n}{4}$ ***respectively. The standard deviation is*** $\frac{\sqrt{n}}{2}$ ***in this case. Let's not forget there is human error from dilution process so*** p ***itself has deviation. Since*** n ***is a large number of particles the standard deviation compared to the mean is relatively small (***$O(\frac{1}{\sqrt{n}})$***). So the human dilution error should be the dominant***

factor here and we will simplify the dilution process into the normal distribution $N(\frac{\sqrt{n}}{2}, n\sigma^2)$. The real concentration of a sample is correlated to the concentration of prior dilutions

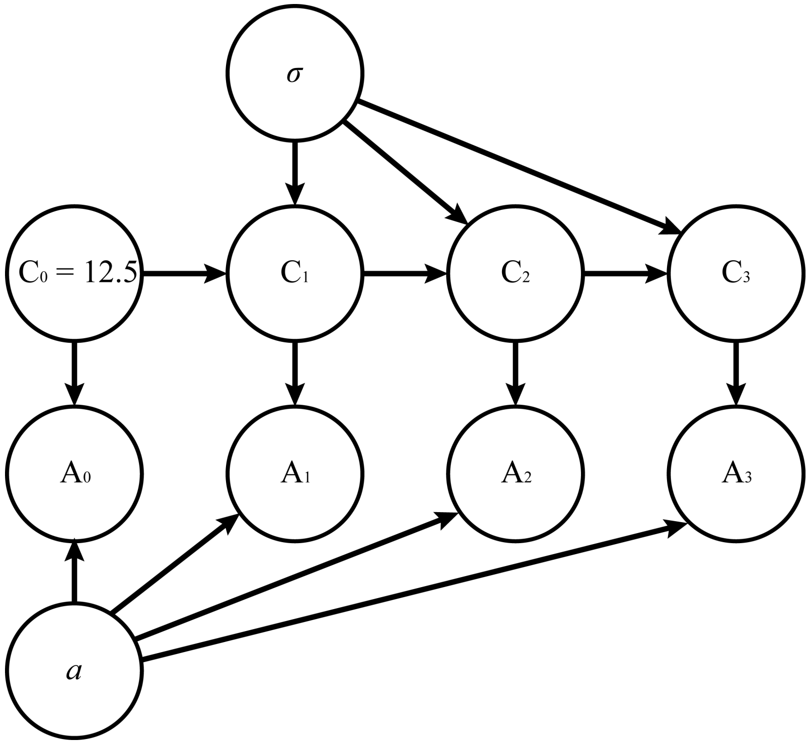


Figure 3. *Graphical model of a serial dilution process. C here denotes the actual concentration, not the assumed concentration*

This hierarchical model should be established for each run.

Side note: Notice how this model would fit nicely for an RNN(Recurrent Neural Network) for the deterministic analysis. As we are concerned with the stochastic nature of the problem we are dealing with, we opt to do the analysis that is based on Bayesian statistics.

Sensor noise

The O.D. measurement itself is also bound to include noise. First, photodiodes have their inherent dark current noises and digitization noise. Also the process of creation and observation of photons also follows Bernoulli process which culminates into binomial distribution which in a long measurement span and low chance approximates to Poisson distribution which creates shot noise. We will disregard these factors here lest we should make the analysis too involved.

R-JAGS

We are going to use R-JAGS(Just Another Gibbs Sampler) to create a model for the Gibbs sampler and create a Markov chain to conduct an MCMC(Markov Chan Monte Carlo) simulation.

In [73]:
```
DNase$Run=as.numeric(as.character(DNase$Run)) # Make sure run is numeric
```

In [74]:
```
max(DNase$Run) # Check that there are 11 runs
```

11

In [18]:
```
any(is.na(DNase)) # Check if there is a missing data
```

FALSE

In [21]:
```
library("rjags") # Load rjags library
```

Given that there are many hidden variables and not so many observation data points, there bound to be some predicament with the curse of dimensionality. With that in mind, the priors were given in a quite heavy handed way. Refer to the supplementary material which shows how the priors were determined.

In [403]:

```
# set the model as a string
mod_string = " model{
    for (i in 1:11){
        realconc[1,i]~dnorm(25,prec/2*25.0)
        for (j in 2:7){
            realconc[j,i] ~ dnorm(realconc[j-1,i]/2,prec/(2*realconc[j-1,i]))
        }
        for (j in c(8)){
            realconc[j,i] ~ dnorm(realconc[j-1,i]/4,prec/(1.0*realconc[j-1,
i]))
        }
        density[i*16] ~ dnorm(realconc[1,i]*a[1]/(1.0+exp(realconc[1,i]*a[6]+a
[7]))+(1-1/(1.0+exp(realconc[1,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc[1,
i]-a[3]))+a[5]),prec_obs)
        density[i*16-1] ~ dnorm(realconc[1,i]*a[1]/(1.0+exp(realconc[1,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[1,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[1,i]-a[3]))+a[5]),prec_obs)
        density[i*16-2] ~ dnorm(realconc[2,i]*a[1]/(1.0+exp(realconc[2,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[2,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[2,i]-a[3]))+a[5]),prec_obs)
        density[i*16-3] ~ dnorm(realconc[2,i]*a[1]/(1.0+exp(realconc[2,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[2,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[2,i]-a[3]))+a[5]),prec_obs)
        density[i*16-4] ~ dnorm(realconc[3,i]*a[1]/(1.0+exp(realconc[3,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[3,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[3,i]-a[3]))+a[5]),prec_obs)
        density[i*16-5] ~ dnorm(realconc[3,i]*a[1]/(1.0+exp(realconc[3,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[3,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[3,i]-a[3]))+a[5]),prec_obs)
        density[i*16-6] ~ dnorm(realconc[4,i]*a[1]/(1.0+exp(realconc[4,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[4,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[4,i]-a[3]))+a[5]),prec_obs)
        density[i*16-7] ~ dnorm(realconc[4,i]*a[1]/(1.0+exp(realconc[4,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[4,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[4,i]-a[3]))+a[5]),prec_obs)
        density[i*16-8] ~ dnorm(realconc[5,i]*a[1]/(1.0+exp(realconc[5,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[5,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[5,i]-a[3]))+a[5]),prec_obs)
        density[i*16-9] ~ dnorm(realconc[5,i]*a[1]/(1.0+exp(realconc[5,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[5,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[5,i]-a[3]))+a[5]),prec_obs)
        density[i*16-10] ~ dnorm(realconc[6,i]*a[1]/(1.0+exp(realconc[6,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[6,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[6,i]-a[3]))+a[5]),prec_obs)
        density[i*16-11] ~ dnorm(realconc[6,i]*a[1]/(1.0+exp(realconc[6,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[6,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[6,i]-a[3]))+a[5]),prec_obs)
        density[i*16-12] ~ dnorm(realconc[7,i]*a[1]/(1.0+exp(realconc[7,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[7,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[7,i]-a[3]))+a[5]),prec_obs)
        density[i*16-13] ~ dnorm(realconc[7,i]*a[1]/(1.0+exp(realconc[7,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[7,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[7,i]-a[3]))+a[5]),prec_obs)
        density[i*16-14] ~ dnorm(realconc[8,i]*a[1]/(1.0+exp(realconc[8,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[8,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[8,i]-a[3]))+a[5]),prec_obs)
```

```
        density[i*16-15] ~ dnorm(realconc[8,i]*a[1]/(1.0+exp(realconc[8,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[8,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[8,i]-a[3]))+a[5]),prec_obs)
    }
    prec ~ dgamma(1.0e-4, 1.0e-4/1.0e2)
    prec_obs ~ dgamma(1.0e-4, 1.0e-4/1.0e2)
    sig = 1/prec
    sig_obs = 1/prec_obs
    a[1] ~dnorm(0.4, 1.0/1e-1)
    a[2] ~dnorm(0.3, 1.0/1e-1)
    a[3] ~dnorm(-0.6, 1.0/1e-1)
    a[4] ~dnorm(2, 1.0/1e-1)
    a[5] ~dnorm(-0.2, 1.0/1e-1)
    a[6] ~dnorm(1, 1.0/1e-1)
    a[7] ~dnorm(-2, 1.0/1e-1)
}
"
```

In [404]:
```
set.seed(101) # Set the seed
```

In [405]:
```
data_jags = as.list(DNase) # Set the data
```

In [406]:
```
mod = jags.model(textConnection(mod_string), data=data_jags, n.chains=3) # Ini
tialize the model
```

```
Warning message in jags.model(textConnection(mod_string), data = data_jags,
n.chains = 3):
"Unused variable "Run" in data"Warning message in jags.model(textConnection(m
od_string), data = data_jags, n.chains = 3):
"Unused variable "conc" in data"

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 176
   Unobserved stochastic nodes: 97
   Total graph size: 1933

Initializing model
```

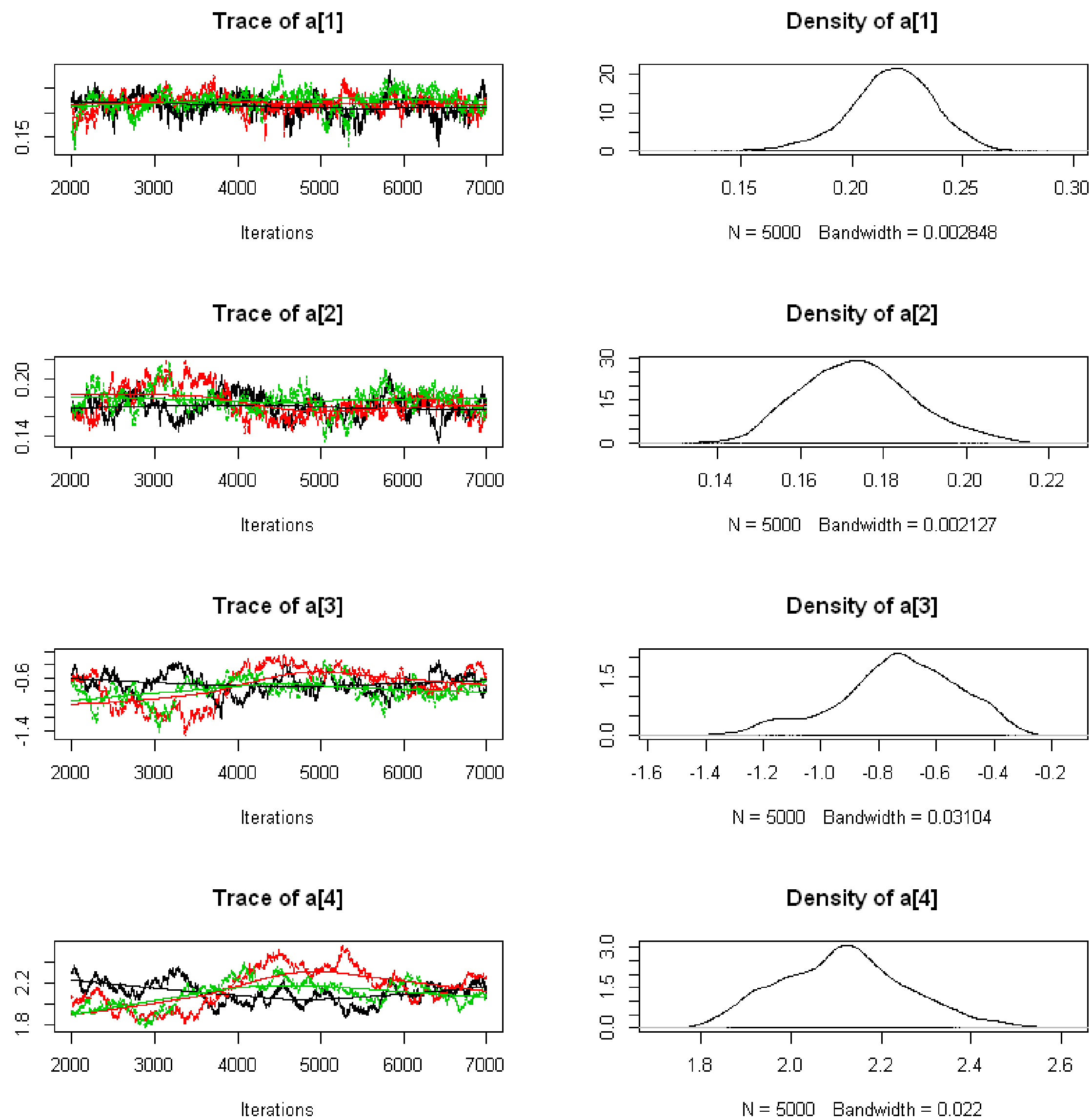
In [407]:
```
update(mod, 1e3) # Run the burn in period for 1000 iterations
```

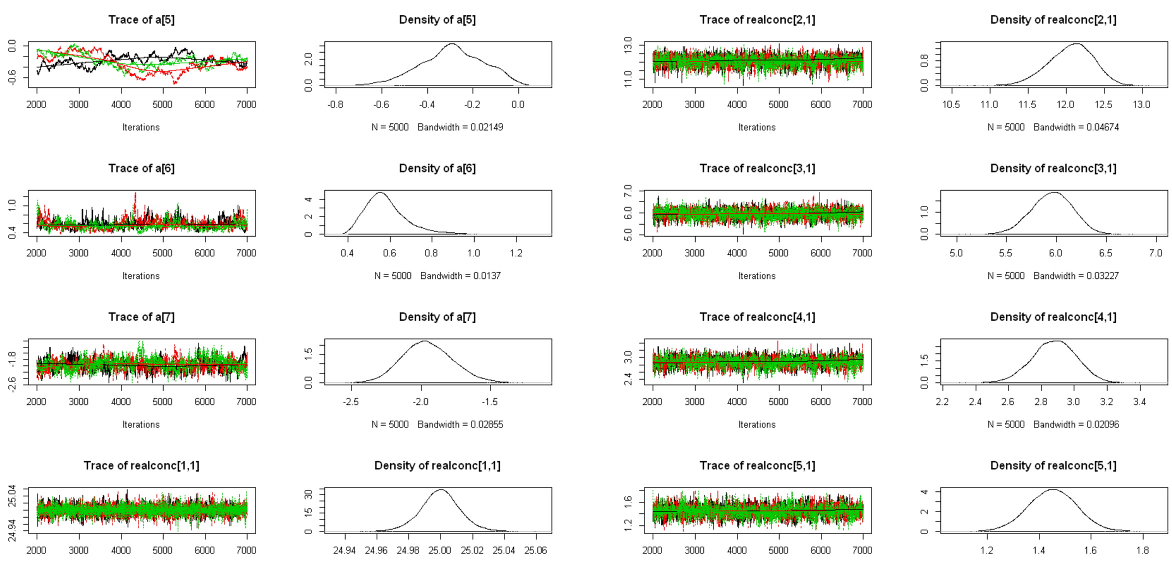
In [408]:
```
params = c("realconc", "sig","sig_obs","a") # Set the parameters to analyse
```

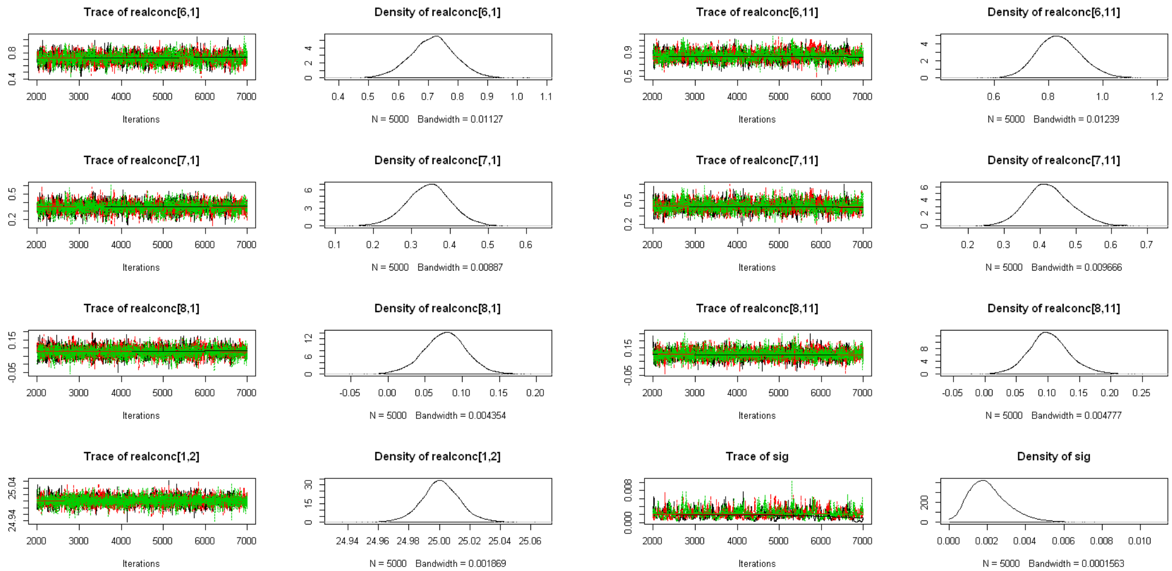
In [409]:
```
# Run and save the Markov chain for 5000 iterations
mod_sim = coda.samples(model=mod,
                        variable.names=params,
                        n.iter=5e3)
mod_csim = as.mcmc(do.call(rbind, mod_sim))
```

In [410]:

```
plot(mod_sim) # Plot the Markov chain
```


Trace of a[1]
Iterations
Density of a[1]
N = 5000 Bandwidth = 0.002848
Trace of a[2]
Iterations
Density of a[2]
N = 5000 Bandwidth = 0.002127
Trace of a[3]
Iterations
Density of a[3]
N = 5000 Bandwidth = 0.03104
Trace of a[4]
Iterations
Density of a[4]
N = 5000 Bandwidth = 0.022


Trace of a[5]
Iterations
Density of a[5]
N = 5000 Bandwidth = 0.02149
Trace of realconc[2,1]
Iterations
Density of realconc[2,1]
N = 5000 Bandwidth = 0.04674
Trace of a[6]
Iterations
Density of a[6]
N = 5000 Bandwidth = 0.0137
Trace of realconc[3,1]
Iterations
Density of realconc[3,1]
N = 5000 Bandwidth = 0.03227
Trace of a[7]
Iterations
Density of a[7]
N = 5000 Bandwidth = 0.02855
Trace of realconc[4,1]
Iterations
Density of realconc[4,1]
N = 5000 Bandwidth = 0.02096
Trace of realconc[1,1]
Iterations
Density of realconc[1,1]
N = 5000 Bandwidth = 0.001808
Trace of realconc[5,1]
Iterations
Density of realconc[5,1]
N = 5000 Bandwidth = 0.01469



The estimated posterior of real concentration seems to have quite big deviation. Since there seems to be an insufficiency of the observations to affect the posterior deeply enough, we will try to change the sigma value and see how it goes.

In [411]:
```
# reset the model as a string
mod_string = " model{
    for (i in 1:11){
        realconc[1,i]~dnorm(25,prec/2*25.0)
        for (j in 2:7){
            realconc[j,i] ~ dnorm(realconc[j-1,i]/2,prec/(2*realconc[j-1,i]))
        }
        for (j in c(8)){
            realconc[j,i] ~ dnorm(realconc[j-1,i]/4,prec/(1.0*realconc[j-1,
i]))
        }
        density[i*16] ~ dnorm(realconc[1,i]*a[1]/(1.0+exp(realconc[1,i]*a[6]+a
[7]))+(1-1/(1.0+exp(realconc[1,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc[1,
i]-a[3]))+a[5]),prec_obs)
        density[i*16-1] ~ dnorm(realconc[1,i]*a[1]/(1.0+exp(realconc[1,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[1,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[1,i]-a[3]))+a[5]),prec_obs)
        density[i*16-2] ~ dnorm(realconc[2,i]*a[1]/(1.0+exp(realconc[2,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[2,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[2,i]-a[3]))+a[5]),prec_obs)
        density[i*16-3] ~ dnorm(realconc[2,i]*a[1]/(1.0+exp(realconc[2,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[2,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[2,i]-a[3]))+a[5]),prec_obs)
        density[i*16-4] ~ dnorm(realconc[3,i]*a[1]/(1.0+exp(realconc[3,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[3,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[3,i]-a[3]))+a[5]),prec_obs)
        density[i*16-5] ~ dnorm(realconc[3,i]*a[1]/(1.0+exp(realconc[3,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[3,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[3,i]-a[3]))+a[5]),prec_obs)
        density[i*16-6] ~ dnorm(realconc[4,i]*a[1]/(1.0+exp(realconc[4,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[4,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[4,i]-a[3]))+a[5]),prec_obs)
        density[i*16-7] ~ dnorm(realconc[4,i]*a[1]/(1.0+exp(realconc[4,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[4,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[4,i]-a[3]))+a[5]),prec_obs)
        density[i*16-8] ~ dnorm(realconc[5,i]*a[1]/(1.0+exp(realconc[5,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[5,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[5,i]-a[3]))+a[5]),prec_obs)
        density[i*16-9] ~ dnorm(realconc[5,i]*a[1]/(1.0+exp(realconc[5,i]*a[6]
+a[7]))+(1-1/(1.0+exp(realconc[5,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realconc
[5,i]-a[3]))+a[5]),prec_obs)
        density[i*16-10] ~ dnorm(realconc[6,i]*a[1]/(1.0+exp(realconc[6,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[6,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[6,i]-a[3]))+a[5]),prec_obs)
        density[i*16-11] ~ dnorm(realconc[6,i]*a[1]/(1.0+exp(realconc[6,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[6,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[6,i]-a[3]))+a[5]),prec_obs)
        density[i*16-12] ~ dnorm(realconc[7,i]*a[1]/(1.0+exp(realconc[7,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[7,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[7,i]-a[3]))+a[5]),prec_obs)
        density[i*16-13] ~ dnorm(realconc[7,i]*a[1]/(1.0+exp(realconc[7,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[7,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[7,i]-a[3]))+a[5]),prec_obs)
        density[i*16-14] ~ dnorm(realconc[8,i]*a[1]/(1.0+exp(realconc[8,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[8,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[8,i]-a[3]))+a[5]),prec_obs)
```

```
        density[i*16-15] ~ dnorm(realconc[8,i]*a[1]/(1.0+exp(realconc[8,i]*a
[6]+a[7]))+(1-1/(1.0+exp(realconc[8,i]*a[6]+a[7])))*(a[4]/(1+exp(-a[2]*realcon
c[8,i]-a[3]))+a[5]),prec_obs)
    }
    prec ~ dgamma(1.0e-4, 1.0e-4/1.0e4)
    prec_obs ~ dgamma(1.0e-4, 1.0e-4/1.0e4)
    sig = 1/prec
    sig_obs = 1/prec_obs
    a[1] ~dnorm(0.4, 1.0/1e-1)
    a[2] ~dnorm(0.3, 1.0/1e-1)
    a[3] ~dnorm(-0.6, 1.0/1e-1)
    a[4] ~dnorm(2, 1.0/1e-1)
    a[5] ~dnorm(-0.2, 1.0/1e-1)
    a[6] ~dnorm(1, 1.0/1e-1)
    a[7] ~dnorm(-2, 1.0/1e-1)
}
"
```

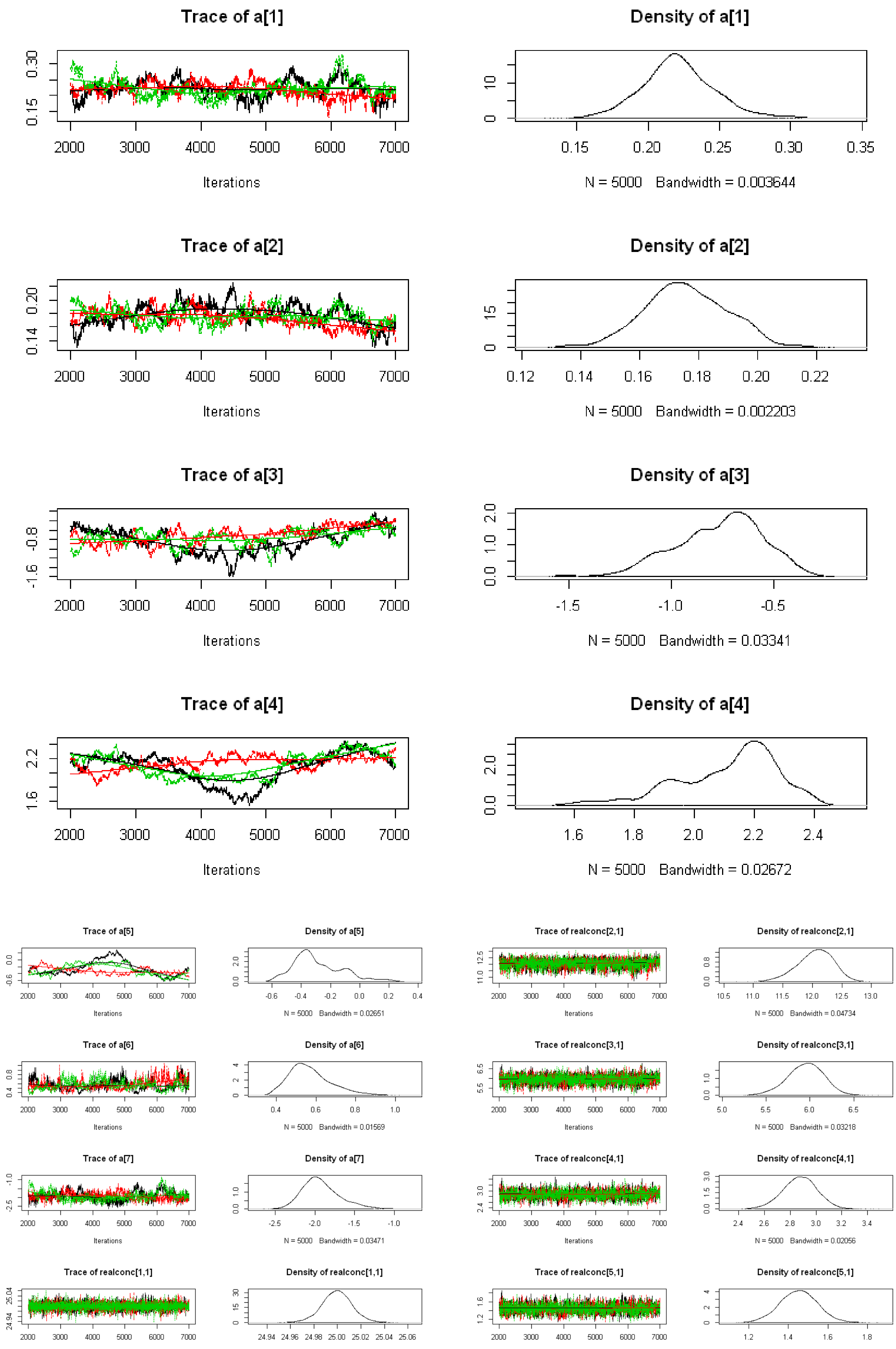
In [412]:
```
mod = jags.model(textConnection(mod_string), data=data_jags, n.chains=3) # Rei
nitialize the model
```

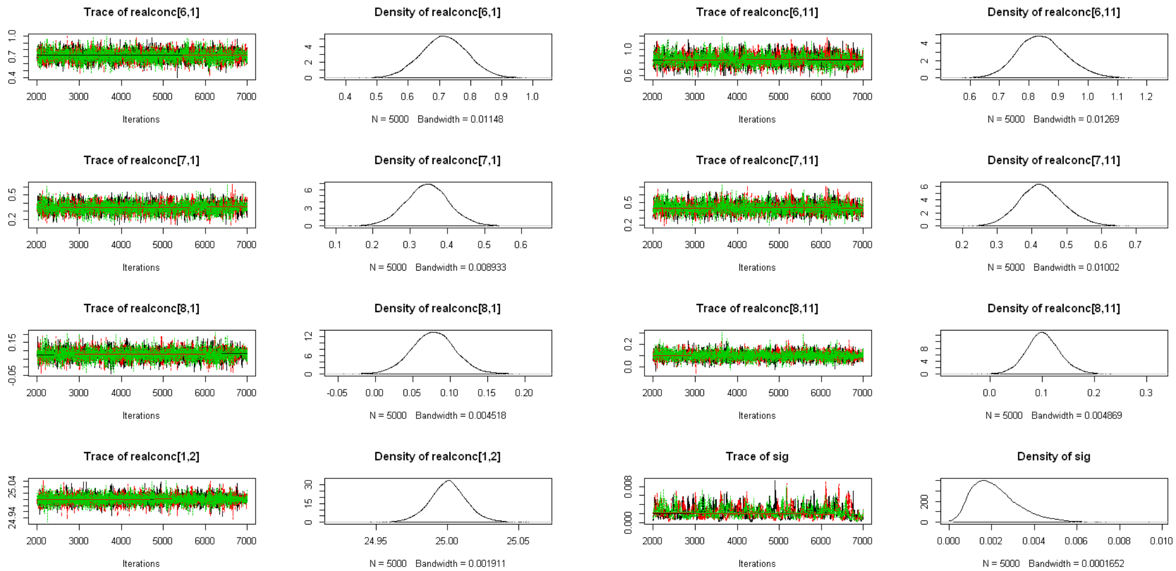
```
Warning message in jags.model(textConnection(mod_string), data = data_jags,
n.chains = 3):
"Unused variable "Run" in data"Warning message in jags.model(textConnection(m
od_string), data = data_jags, n.chains = 3):
"Unused variable "conc" in data"

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 176
   Unobserved stochastic nodes: 97
   Total graph size: 1933

Initializing model
```

In [413]:
```
update(mod, 1e3) # Run the burn in period for 1000 iterations
```

In [414]:
```
# Run and save the Markov chain for 5000 iterations
mod_sim = coda.samples(model=mod,
                        variable.names=params,
                        n.iter=5e3)
mod_csim = as.mcmc(do.call(rbind, mod_sim))
```

In [415]:
```
plot(mod_sim) # Plot the Markov chain
```


Trace of a[1]
Iterations
Density of a[1]
N = 5000 Bandwidth = 0.003644
Trace of a[2]
Iterations
Density of a[2]
N = 5000 Bandwidth = 0.002203
Trace of a[3]
Iterations
Density of a[3]
N = 5000 Bandwidth = 0.03341
Trace of a[4]
Iterations
Density of a[4]
N = 5000 Bandwidth = 0.02672
Trace of a[5]
Iterations
Density of a[5]
N = 5000 Bandwidth = 0.02651
Trace of realconc[2,1]
Iterations
Density of realconc[2,1]
N = 5000 Bandwidth = 0.04734
Trace of a[6]
Iterations
Density of a[6]
N = 5000 Bandwidth = 0.01569
Trace of realconc[3,1]
Iterations
Density of realconc[3,1]
N = 5000 Bandwidth = 0.03218
Trace of a[7]
Iterations
Density of a[7]
N = 5000 Bandwidth = 0.03471
Trace of realconc[4,1]
Iterations
Density of realconc[4,1]
N = 5000 Bandwidth = 0.02056
Trace of realconc[1,1]
Iterations
Density of realconc[1,1]
N = 5000 Bandwidth = 0.001937
Trace of realconc[5,1]
Iterations
Density of realconc[5,1]
N = 5000 Bandwidth = 0.01467



It seems like it didn't have much of a serious effect. However the parameter 'a' is moving a bit.

Gelman diagnosis

Many of parameters a are not really close to 1

In [416]:

```
gelman.diag(mod_sim)
```

Potential scale reduction factors:

	Point est.	Upper C.I.
a[1]	1.02	1.05
a[2]	1.14	1.42
a[3]	1.31	1.95
a[4]	1.20	1.64
a[5]	1.19	1.62
a[6]	1.04	1.11
a[7]	1.02	1.03
realconc[1,1]	1.00	1.00
realconc[2,1]	1.00	1.00
realconc[3,1]	1.00	1.00
realconc[4,1]	1.00	1.01
realconc[5,1]	1.00	1.01
realconc[6,1]	1.00	1.01
realconc[7,1]	1.00	1.01
realconc[8,1]	1.00	1.01
realconc[1,2]	1.00	1.00
realconc[2,2]	1.00	1.00
realconc[3,2]	1.00	1.00
realconc[4,2]	1.00	1.00
realconc[5,2]	1.00	1.00
realconc[6,2]	1.00	1.00
realconc[7,2]	1.00	1.00
realconc[8,2]	1.00	1.00
realconc[1,3]	1.00	1.00
realconc[2,3]	1.00	1.00
realconc[3,3]	1.00	1.00
realconc[4,3]	1.00	1.00
realconc[5,3]	1.00	1.00
realconc[6,3]	1.00	1.00
realconc[7,3]	1.00	1.00
realconc[8,3]	1.00	1.00
realconc[1,4]	1.00	1.00
realconc[2,4]	1.00	1.00
realconc[3,4]	1.00	1.00
realconc[4,4]	1.00	1.00
realconc[5,4]	1.00	1.00
realconc[6,4]	1.00	1.00
realconc[7,4]	1.00	1.00
realconc[8,4]	1.00	1.00
realconc[1,5]	1.00	1.00
realconc[2,5]	1.00	1.00
realconc[3,5]	1.00	1.00
realconc[4,5]	1.00	1.01
realconc[5,5]	1.00	1.01
realconc[6,5]	1.00	1.01
realconc[7,5]	1.00	1.01
realconc[8,5]	1.00	1.00
realconc[1,6]	1.00	1.00
realconc[2,6]	1.00	1.00
realconc[3,6]	1.00	1.00
realconc[4,6]	1.00	1.00
realconc[5,6]	1.00	1.00
realconc[6,6]	1.00	1.00
realconc[7,6]	1.00	1.00

```
realconc[8,6]       1.00       1.00
realconc[1,7]       1.00       1.00
realconc[2,7]       1.00       1.00
realconc[3,7]       1.00       1.00
realconc[4,7]       1.00       1.00
realconc[5,7]       1.00       1.00
realconc[6,7]       1.00       1.00
realconc[7,7]       1.00       1.00
realconc[8,7]       1.00       1.00
realconc[1,8]       1.00       1.00
realconc[2,8]       1.00       1.00
realconc[3,8]       1.00       1.00
realconc[4,8]       1.00       1.00
realconc[5,8]       1.00       1.00
realconc[6,8]       1.00       1.00
realconc[7,8]       1.00       1.00
realconc[8,8]       1.00       1.00
realconc[1,9]       1.00       1.00
realconc[2,9]       1.00       1.00
realconc[3,9]       1.00       1.00
realconc[4,9]       1.00       1.00
realconc[5,9]       1.00       1.00
realconc[6,9]       1.00       1.00
realconc[7,9]       1.00       1.00
realconc[8,9]       1.00       1.00
realconc[1,10]      1.00       1.00
realconc[2,10]      1.00       1.00
realconc[3,10]      1.00       1.00
realconc[4,10]      1.00       1.00
realconc[5,10]      1.00       1.00
realconc[6,10]      1.00       1.01
realconc[7,10]      1.00       1.01
realconc[8,10]      1.00       1.00
realconc[1,11]      1.00       1.00
realconc[2,11]      1.00       1.00
realconc[3,11]      1.00       1.01
realconc[4,11]      1.00       1.01
realconc[5,11]      1.00       1.02
realconc[6,11]      1.00       1.01
realconc[7,11]      1.00       1.01
realconc[8,11]      1.00       1.01
sig                 1.00       1.00
sig_obs             1.00       1.00

Multivariate psrf

1.21
```

Autocorrelation diagnosis

The chain for the parameter 'a' is quite correlated

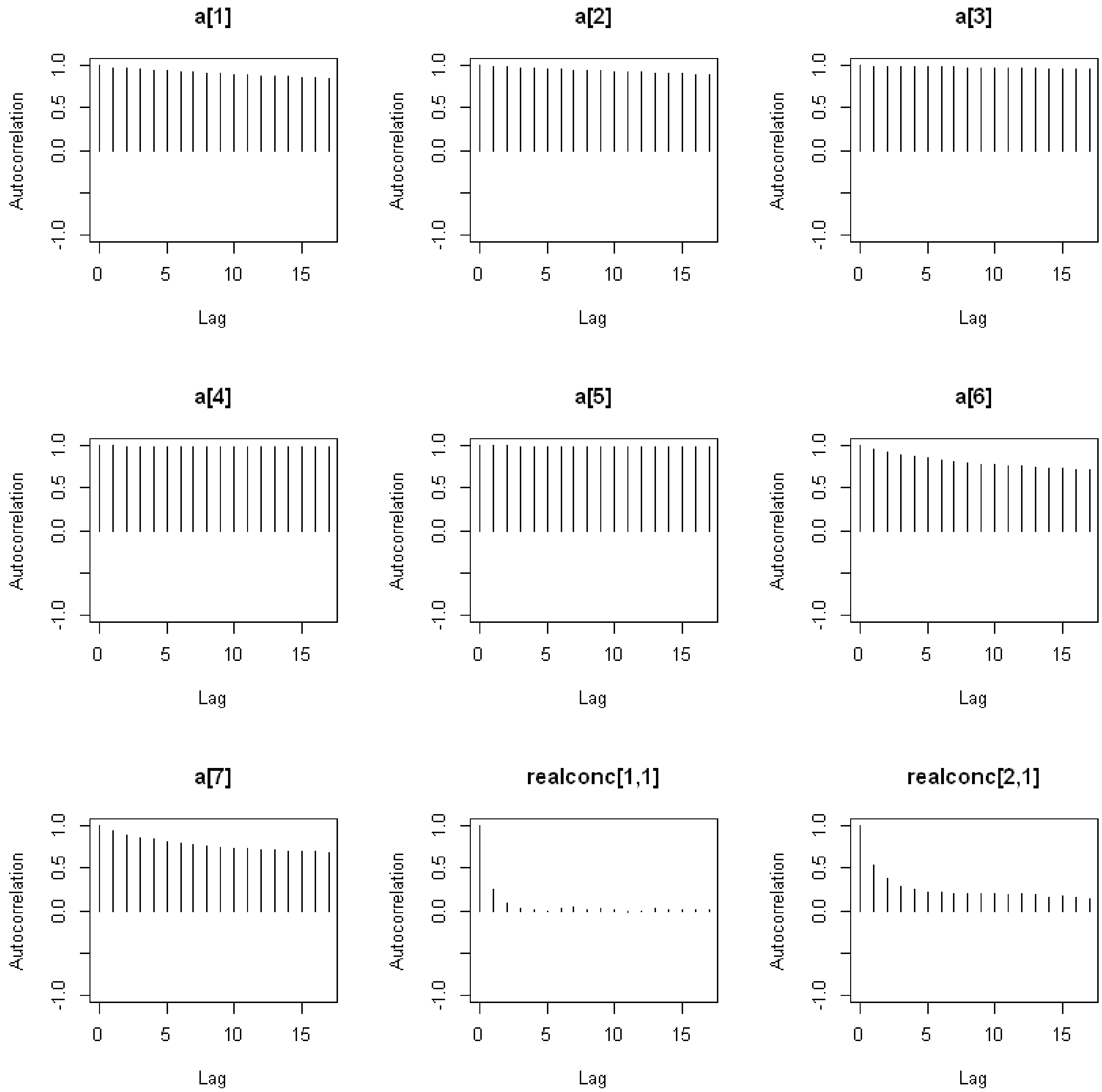
In [417]:
```
autocorr.diag(mod_sim)
```

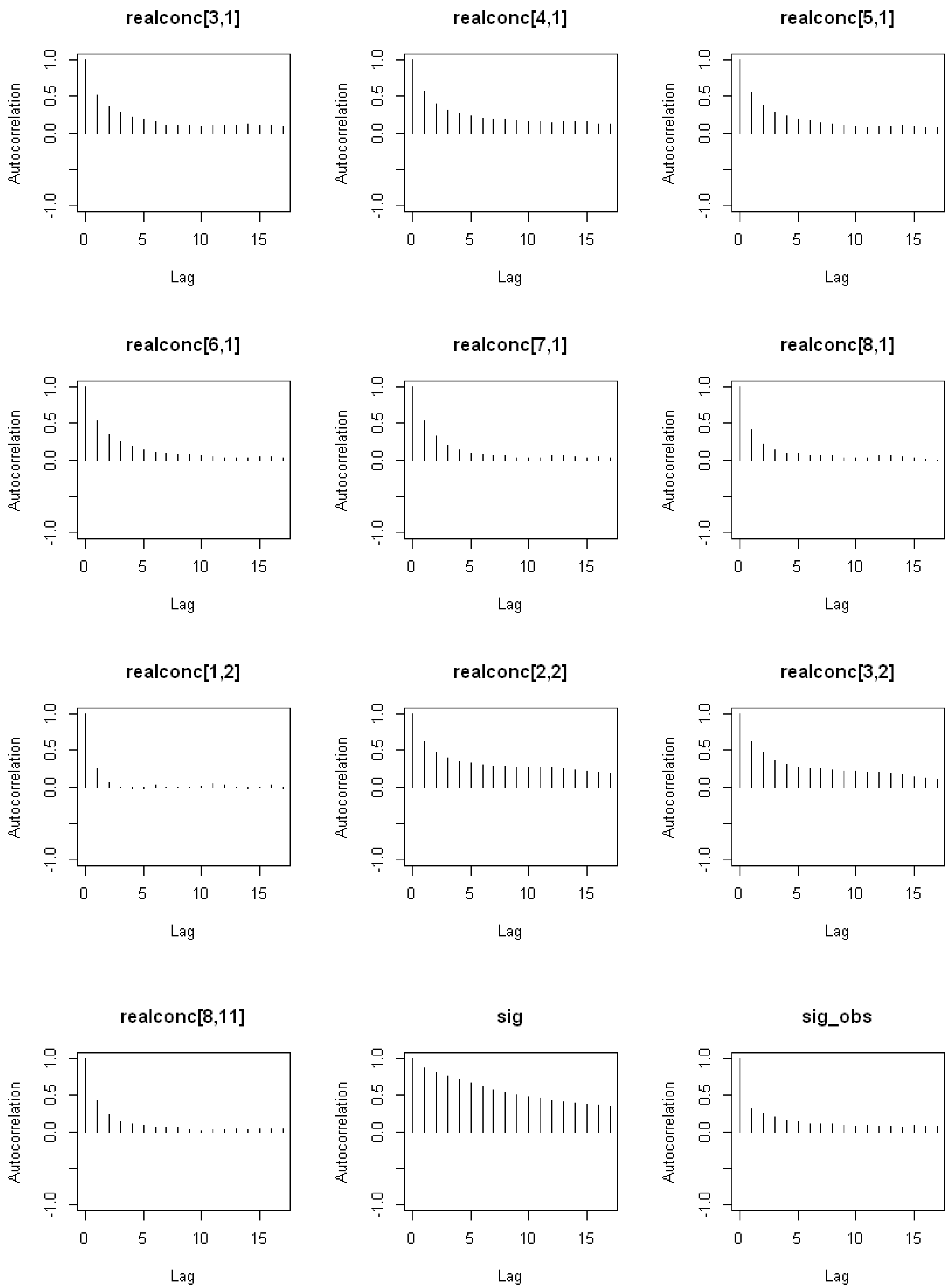
A matrix: 5 × 97 of type dbl

	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	realconc[1,1]
Lag 0	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000	1.00000000
Lag 1	0.9772894	0.9865691	0.9953475	0.9974737	0.9983483	0.9514079	0.9201140	0.27357579
Lag 5	0.9267941	0.9430034	0.9785066	0.9887297	0.9909979	0.8300116	0.7678532	-0.00191754
Lag 10	0.8808648	0.8984300	0.9588843	0.9784303	0.9815824	0.7410313	0.6827357	0.00131215
Lag 50	0.6878538	0.6865401	0.8148605	0.9102857	0.9083166	0.5157057	0.5299662	-0.01436245

In [418]:

```
autocorr.plot(mod_sim)
```


a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
realconc[1,1]
realconc[2,1]
Autocorrelation
Lag
-1.0
0.0
0.5
1.0
0
5
10
15


realconc[3,1]
realconc[4,1]
realconc[5,1]
realconc[6,1]
realconc[7,1]
realconc[8,1]
realconc[1,2]
realconc[2,2]
realconc[3,2]
realconc[8,11]
sig
sig_obs
Autocorrelation
Lag
-1.0
0.0
0.5
1.0
0
5
10
15

In [419]:

```
effectiveSize(mod_sim)
```

a[1]	*76.7594573860687*
a[2]	*67.3381989389826*
a[3]	*33.1155901708168*
a[4]	*17.0125073158708*
a[5]	*13.7184107633014*
a[6]	*116.96156735949*
a[7]	*97.4658252005726*
realconc[1,1]	*8127.14584874573*
realconc[2,1]	*1379.8655446816*
realconc[3,1]	*2488.35561239834*
realconc[4,1]	*1651.71235438296*
realconc[5,1]	*2409.09287261978*
realconc[6,1]	*3218.8301201745*
realconc[7,1]	*3442.91098069515*
realconc[8,1]	*3964.6222969993*
realconc[1,2]	*8691.81539090158*
realconc[2,2]	*935.736053470472*
realconc[3,2]	*1472.71902263711*
realconc[4,2]	*2574.14700879153*
realconc[5,2]	*2773.05295803478*
realconc[6,2]	*2944.33754063708*
realconc[7,2]	*3645.8467697376*
realconc[8,2]	*5323.19200984287*
realconc[1,3]	*8497.41743582672*
realconc[2,3]	*810.102994422017*
realconc[3,3]	*1410.47018224218*
realconc[4,3]	*2140.63253989758*
realconc[5,3]	*1665.29318001923*
realconc[6,3]	*2033.13626535738*
realconc[7,3]	*2376.23845234261*
realconc[8,3]	*4687.63258819698*
realconc[1,4]	*8428.81792930508*
realconc[2,4]	*4015.64193725135*
realconc[3,4]	*1829.32802122781*
realconc[4,4]	*2181.43285333998*
realconc[5,4]	*2194.31870843041*
realconc[6,4]	*2963.35969888377*
realconc[7,4]	*3382.86297849954*
realconc[8,4]	*4282.46158513219*
realconc[1,5]	*8888.41318425093*
realconc[2,5]	*4962.3976989269*
realconc[3,5]	*3112.19521872161*
realconc[4,5]	*2360.59323005501*
realconc[5,5]	*2606.1564333981*
realconc[6,5]	*2998.68847862102*
realconc[7,5]	*3430.84256361824*
realconc[8,5]	*4924.49176512944*
realconc[1,6]	*8157.55416537856*

realconc[2,6]	*5192.25979629715*
realconc[3,6]	*2828.80545428456*
realconc[4,6]	*2529.0478228825*
realconc[5,6]	*1662.97379834833*
realconc[6,6]	*1387.09677200753*
realconc[7,6]	*1297.85912836551*
realconc[8,6]	*2709.98477698254*
realconc[1,7]	*8588.83942923375*
realconc[2,7]	*4649.45418342513*
realconc[3,7]	*3428.03525422743*
realconc[4,7]	*2316.78389332958*
realconc[5,7]	*1421.35784994986*
realconc[6,7]	*1317.95612171474*
realconc[7,7]	*1439.73169185004*
realconc[8,7]	*3026.4564943185*
realconc[1,8]	*7915.48305539054*
realconc[2,8]	*4890.62196898903*
realconc[3,8]	*3815.03020514216*
realconc[4,8]	*2251.00100570778*
realconc[5,8]	*2685.57711299993*
realconc[6,8]	*3028.77126573925*
realconc[7,8]	*3097.60294289972*
realconc[8,8]	*4712.99643410748*
realconc[1,9]	*7867.77302357417*
realconc[2,9]	*4025.58168088058*
realconc[3,9]	*3811.11414658001*
realconc[4,9]	*2622.46388680596*
realconc[5,9]	*2460.15918667865*
realconc[6,9]	*2976.05405368908*
realconc[7,9]	*3412.42770822673*
realconc[8,9]	*5462.17675854183*
realconc[1,10]	*8567.13396168339*
realconc[2,10]	*3065.2479256207*
realconc[3,10]	*3475.26996794351*
realconc[4,10]	*2030.56656333844*
realconc[5,10]	*1458.96298176036*
realconc[6,10]	*1703.69704770518*
realconc[7,10]	*2464.68266970292*
realconc[8,10]	*5087.23734312177*
realconc[1,11]	*8626.31139880812*
realconc[2,11]	*3070.16362644335*
realconc[3,11]	*2641.05027411566*
realconc[4,11]	*1982.2192348554*
realconc[5,11]	*1638.53911624476*
realconc[6,11]	*1733.51518123179*
realconc[7,11]	*2591.55348202401*
realconc[8,11]	*5237.4376241712*
sig	*486.732240442622*
sig_obs	*1838.40933893738*

DIC calculation

It's giving quite egregious numbers.

In [420]:
```
dic = dic.samples(mod, n.iter=1e3)
```

In [421]:
```
dic
```

```
Mean deviance:  -610.1
penalty 26.38
Penalized deviance: -583.7
```

From the information gathered from the diagnosis, I would conclude that the chain needs to be run much longer than 5000 iterations.

Inference

Let's infer some informations from the chain

In [423]:

```
head(mod_csim)
```

```
Markov Chain Monte Carlo (MCMC) output:
Start = 1
End = 7
Thinning interval = 1
          a[1]      a[2]       a[3]     a[4]       a[5]      a[6]      a[7]
[1,] 0.2140201 0.1652910 -0.6020445 2.137660 -0.3281899 0.5013373 -1.893663
[2,] 0.2056735 0.1659661 -0.6172437 2.153058 -0.3252558 0.5054294 -1.752631
[3,] 0.2064943 0.1644933 -0.6243514 2.150235 -0.3187493 0.5336042 -1.977749
[4,] 0.2214139 0.1626985 -0.6417867 2.163053 -0.3293995 0.5350070 -1.952238
[5,] 0.2212293 0.1620244 -0.6330523 2.164811 -0.3318511 0.5359185 -1.956807
[6,] 0.2137852 0.1645869 -0.6113273 2.170306 -0.3477834 0.5518911 -2.025706
[7,] 0.2099539 0.1665148 -0.6270470 2.177306 -0.3502241 0.5432106 -1.798690
     realconc[1,1] realconc[2,1] realconc[3,1] realconc[4,1] realconc[5,1]
[1,]      24.99609      12.36192      6.198060      2.973204      1.391764
[2,]      24.99789      12.43210      5.968395      3.098828      1.591618
[3,]      24.99282      12.47675      6.254145      2.932386      1.563230
[4,]      24.99723      12.51081      6.232276      2.927163      1.528176
[5,]      25.00336      12.70143      6.217015      3.010497      1.503728
[6,]      25.00101      12.36223      6.329704      3.020241      1.502601
[7,]      24.99686      12.10973      6.057780      2.904653      1.421651
     realconc[6,1] realconc[7,1] realconc[8,1] realconc[1,2] realconc[2,2]
[1,]     0.6701167     0.3034967    0.06667049      25.00170      12.68503
[2,]     0.7458847     0.3535031    0.07152063      24.99346      12.46637
[3,]     0.7349558     0.3475463    0.07638409      24.99910      12.67587
[4,]     0.7472958     0.3481707    0.07339691      25.00009      12.71166
[5,]     0.7207129     0.4034380    0.10225073      25.00656      13.16831
[6,]     0.7546648     0.3339855    0.09473432      24.98548      12.72181
[7,]     0.7111665     0.3249338    0.06631248      25.00401      12.74385
     realconc[3,2] realconc[4,2] realconc[5,2] realconc[6,2] realconc[7,2]
[1,]      6.476272      3.172228      1.562717     0.7482765     0.3853436
[2,]      6.334506      3.171242      1.579324     0.7877564     0.3959459
[3,]      6.164591      3.181796      1.584757     0.8055965     0.3726918
[4,]      6.538900      3.203167      1.559834     0.7918203     0.3939118
[5,]      6.486651      3.266500      1.616842     0.7920896     0.4415865
[6,]      6.601970      3.304043      1.604023     0.7471214     0.3720061
[7,]      6.610563      3.271468      1.608351     0.8155870     0.3411984
     realconc[8,2] realconc[1,3] realconc[2,3] realconc[3,3] realconc[4,3]
[1,]    0.10766770      25.00859      12.88151      6.612313      3.401026
[2,]    0.10461759      24.99741      12.84024      6.624715      3.425867
[3,]    0.09660813      25.00022      12.94902      6.620603      3.398788
[4,]    0.08790456      24.99814      13.16782      6.555398      3.265379
[5,]    0.09361998      25.00108      13.19954      6.697549      3.251949
[6,]    0.10023965      24.99402      12.74507      6.762743      3.191483
[7,]    0.10052618      24.99717      12.95088      6.729620      3.165741
     realconc[5,3] realconc[6,3] realconc[7,3] realconc[8,3] realconc[1,4]
[1,]      1.725072     0.8614574     0.4297936    0.12781869      25.00389
[2,]      1.669142     0.8722192     0.4468685    0.10029279      25.00422
[3,]      1.731051     0.8276265     0.4596408    0.10281251      24.99769
[4,]      1.624503     0.7958283     0.3772834    0.11161257      25.00175
[5,]      1.643785     0.8708216     0.4498808    0.07370208      25.00382
[6,]      1.606388     0.8481864     0.4143627    0.09283016      25.01255
[7,]      1.586738     0.8479920     0.4236157    0.11665010      25.01461
     realconc[2,4] realconc[3,4] realconc[4,4] realconc[5,4] realconc[6,4]
[1,]      12.47508      6.131232      3.007615      1.503911     0.7244713
[2,]      12.47151      6.140136      3.008938      1.562005     0.6995800
[3,]      12.52216      6.117992      3.024282      1.459615     0.7292249
[4,]      12.41605      6.104169      3.023439      1.480660     0.7794858
```

```
[5,]     12.43875      6.186246      3.085909      1.508060     0.7312274
[6,]     12.23802      6.003633      2.992497      1.533609     0.7321804
[7,]     12.28838      6.165394      3.072763      1.540653     0.7289238
     realconc[7,4] realconc[8,4] realconc[1,5] realconc[2,5] realconc[3,5]
[1,]     0.3807498    0.10263162      24.99754      12.58348      6.408047
[2,]     0.3155001    0.09657118      24.99790      12.50949      6.392912
[3,]     0.3191127    0.06569891      24.99267      12.31129      6.040670
[4,]     0.3614223    0.10102141      24.99554      12.66166      6.322238
[5,]     0.3982385    0.11101201      24.99838      12.44845      6.161953
[6,]     0.3743486    0.08008959      24.99714      12.20862      6.060404
[7,]     0.3442779    0.06773781      24.99740      12.52953      6.075763
     realconc[4,5] realconc[5,5] realconc[6,5] realconc[7,5] realconc[8,5]
[1,]      3.275572      1.732426     0.8392540     0.4531109    0.08172620
[2,]      3.334657      1.661787     0.9098046     0.4508870    0.14375851
[3,]      3.096758      1.632130     0.8594473     0.5114301    0.15565241
[4,]      3.158407      1.594539     0.8289313     0.4558175    0.12156904
[5,]      3.054072      1.614186     0.8401381     0.4124324    0.10162933
[6,]      2.972090      1.484147     0.7407069     0.3830424    0.08587352
[7,]      2.983141      1.407181     0.7116042     0.3444297    0.06324619
     realconc[1,6] realconc[2,6] realconc[3,6] realconc[4,6] realconc[5,6]
[1,]      24.99585      12.84614      6.499970      3.236464      1.646173
[2,]      24.99325      12.46937      6.362791      3.266651      1.626477
[3,]      24.99702      12.42291      6.129399      3.106582      1.612979
[4,]      25.00000      12.35527      6.067805      2.992729      1.609934
[5,]      25.00123      12.56217      6.007426      3.118291      1.647354
[6,]      25.00159      12.57553      6.069266      3.013972      1.599275
[7,]      24.98967      12.67853      6.550424      3.136637      1.598731
     realconc[6,6] realconc[7,6] realconc[8,6] realconc[1,7] realconc[2,7]
[1,]     0.8717919     0.4790589    0.09975191      25.00054      12.36982
[2,]     0.7864368     0.3868752    0.10806094      25.00182      12.43917
[3,]     0.8892139     0.4324040    0.11570029      24.99728      12.35930
[4,]     0.7777995     0.4440739    0.10795538      25.00724      12.33768
[5,]     0.9168302     0.4276030    0.11171639      25.00018      12.14332
[6,]     0.9326167     0.5124999    0.12453443      24.99920      12.47737
[7,]     0.9281282     0.5454679    0.13307309      24.98844      12.27755
     realconc[3,7] realconc[4,7] realconc[5,7] realconc[6,7] realconc[7,7]
[1,]      6.220065      3.064983      1.621305     0.7855115     0.3996063
[2,]      6.183609      3.012101      1.509748     0.7715059     0.3793761
[3,]      6.178240      3.135946      1.551122     0.7695666     0.4058725
[4,]      6.186222      3.029839      1.595664     0.7724253     0.4227098
[5,]      6.046875      3.023612      1.591259     0.8381293     0.4298849
[6,]      6.029708      2.986932      1.566676     0.8371634     0.4725275
[7,]      6.295275      3.152203      1.510263     0.8389681     0.4523582
     realconc[8,7] realconc[1,8] realconc[2,8] realconc[3,8] realconc[4,8]
[1,]    0.09966979      24.99616      12.16324      5.981331      2.991815
[2,]    0.09256120      25.00205      12.48807      5.955422      2.927198
[3,]    0.09686553      25.00105      12.24113      6.000402      2.917018
[4,]    0.12356081      24.99992      12.35829      6.085736      2.880798
[5,]    0.12080054      24.99344      12.33937      6.051318      2.954573
[6,]    0.12532634      25.00577      12.35319      6.105451      2.922578
[7,]    0.10026359      24.98592      12.52848      6.111498      3.108369
     realconc[5,8] realconc[6,8] realconc[7,8] realconc[8,8] realconc[1,9]
[1,]      1.492428     0.7239885     0.3600980    0.09787258      25.00622
[2,]      1.502689     0.7561067     0.3894440    0.11543857      24.99796
[3,]      1.525874     0.7575243     0.3625440    0.07884016      25.00250
[4,]      1.536637     0.7033631     0.3558105    0.07279947      25.00802
[5,]      1.553534     0.6784868     0.3208688    0.07224245      24.99759
```

```
[6,]      1.524630     0.8100482     0.3181463    0.07956824      24.99690
[7,]      1.539654     0.8057241     0.3558556    0.09073383      24.99765
     realconc[2,9] realconc[3,9] realconc[4,9] realconc[5,9] realconc[6,9]
[1,]      12.81626      6.122012      2.997057      1.481925     0.6461500
[2,]      12.81943      6.266539      3.031984      1.539502     0.7805214
[3,]      12.61328      6.279075      3.207550      1.540911     0.7867018
[4,]      12.62198      6.216702      3.079888      1.545233     0.8263937
[5,]      12.64852      6.204843      3.213560      1.627481     0.8529247
[6,]      12.31691      6.212127      3.179438      1.613984     0.8277585
[7,]      12.77751      6.358692      3.122720      1.655953     0.8379504
     realconc[7,9] realconc[8,9] realconc[1,10] realconc[2,10] realconc[3,10]
[1,]     0.3211733    0.09483529       24.99646       12.63207       6.201799
[2,]     0.3267370    0.08019463       24.98767       12.22039       6.130365
[3,]     0.4289810    0.08236390       24.98638       12.33962       6.075736
[4,]     0.4655728    0.12952791       24.99945       12.33818       6.102904
[5,]     0.4612841    0.10206492       25.00769       12.99482       6.526611
[6,]     0.4476925    0.11195896       24.99562       12.95357       6.543027
[7,]     0.4142676    0.13043209       25.00183       12.86495       6.607115
     realconc[4,10] realconc[5,10] realconc[6,10] realconc[7,10] realconc[8,1
0]
[1,]       3.252249       1.698065      0.8590402      0.3855394     0.077233
39
[2,]       3.138364       1.682116      0.8740965      0.4019876     0.116658
15
[3,]       3.000961       1.574098      0.8269490      0.4191490     0.123418
19
[4,]       3.107792       1.557283      0.8104041      0.3714869     0.085811
26
[5,]       3.169290       1.597216      0.8305835      0.3996617     0.086559
87
[6,]       3.227667       1.649305      0.8928012      0.4043374     0.124324
72
[7,]       3.259734       1.749089      0.8913674      0.4976715     0.121607
49
     realconc[1,11] realconc[2,11] realconc[3,11] realconc[4,11] realconc[5,1
1]
[1,]       25.01198       12.44739       6.268902       3.140123        1.5684
08
[2,]       25.00369       12.53174       6.132856       3.144285        1.5814
67
[3,]       25.00042       12.48614       6.066229       3.084920        1.5972
23
[4,]       24.99457       12.42962       6.014334       3.003877        1.5867
26
[5,]       25.00045       12.43943       6.121681       3.010986        1.5788
15
[6,]       25.00697       12.44415       6.258492       3.155971        1.6265
94
[7,]       25.00330       12.15150       6.116408       3.102013        1.6586
85
     realconc[6,11] realconc[7,11] realconc[8,11]          sig     sig_obs
[1,]      0.7847440      0.2916020     0.06147569 0.0006773110 0.002067647
[2,]      0.7389864      0.2922403     0.05924740 0.0004695907 0.001992387
[3,]      0.7678628      0.3298344     0.08219975 0.0005380854 0.002340183
[4,]      0.7293871      0.3226925     0.06951289 0.0006602556 0.002001121
[5,]      0.8007129      0.3351983     0.08733945 0.0008096556 0.002060964
```

```
[6,]      0.8958602      0.4122882      0.07210468 0.0010218471 0.002076713
[7,]      0.8936669      0.4572477      0.11035542 0.0008923223 0.001876207
```

Let's calculate the probability of the solution of desired concentration 3.125 from the 3rd serial dilution run having actual concentration bigger than 3.2

In [433]:
```
mean(mod_csim[,"realconc[4,3]"]>3.2)
```

0.751466666666667

75% sounds reasonable considering the O.D. measurement from the 3rd run was a bit higher than the other runs. How about the last dilution being higher than the desired concentration from this run?

In [434]:
```
mean(mod_csim[,"realconc[8,3]"]>0.048828125)
```

0.973333333333333

The answer is 97%. Let's analyze the quantiles of the second dilution of the 5th run.

In [436]:
```
quantile(mod_csim[,"realconc[3,5]"],probs = seq(0, 1, 0.05))
```

0%	5.54734647585879
5%	5.90542359874354
10%	5.97685025786341
15%	6.02632743418486
20%	6.0661408629103
25%	6.10223585471334
30%	6.13236660977141
35%	6.16122422277587
40%	6.18648452720703
45%	6.21227719272305
50%	6.23716554813331
55%	6.2619778819074
60%	6.28804691434652
65%	6.31472286322206
70%	6.34412600624314
75%	6.37497375385601
80%	6.40997710610922
85%	6.45170420545397
90%	6.50828413673612
95%	6.59765308749071
100%	7.18631702976666

There is a 90% probablity that the real concentration falls between 5.905 and 6.598, given the prior

Conclusion

We have analyized the dataset "DNase", and we inferred the posterior distributions of the real concentration from the observations of O.D. so that we can estimate the distribution of the concentration of the solution that are made from each run of serial dilution.