Bayesian statistics project wage, education, experience dataset made in R

1. Understand the problem

The goal is to successfully model the wages, education and experience relationship.

2. Plan and properly collect relevant data

The dataset i used was available from the links in the course, so i didn't have to collect my data.

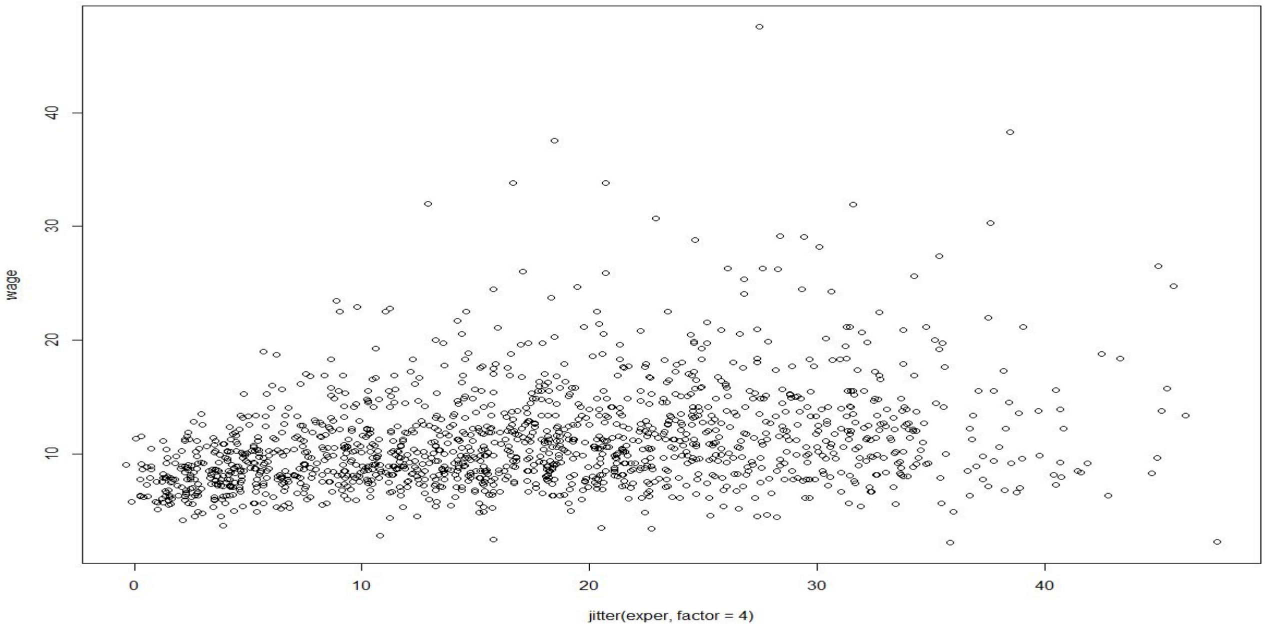
3. Explore data

The dataset

wage	educ	exper	sex
7,7802076852	1	23	NA
4,81850475584	1	15	NA
10,563645423	1	31	NA
7,0424294557	1	32	NA
7,88752079207	1	9	NA

The education column is ranging from 1 to 5. The sex column is not filled in anywhere.

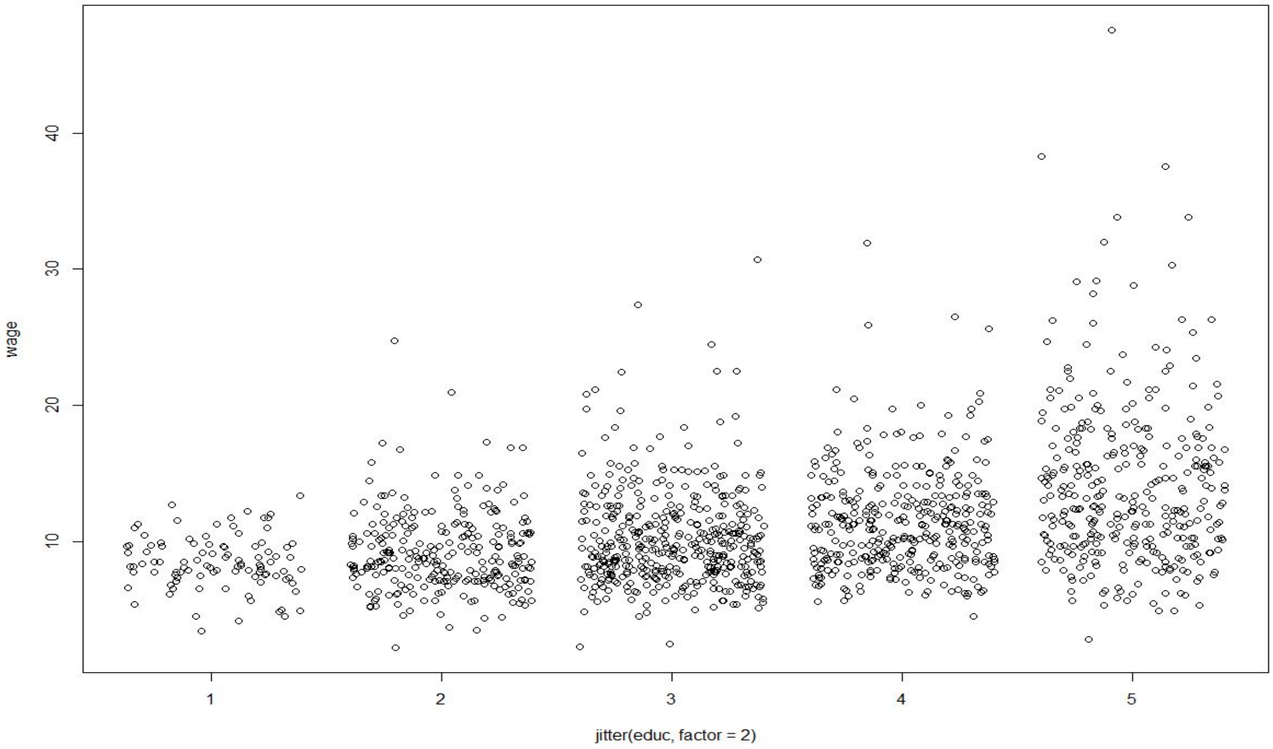
Getting some insight from the data



Wage plotted against experience

`plot(wage ~ jitter(exper, factor = 4), data = data)`

From graph it is visible that the variance of the wage is increasing with the experience, altho the mean is only changing a little.



Wage plotted against the education level

```
plot(wage ~ jitter(educ, factor = 2), data = data)
```

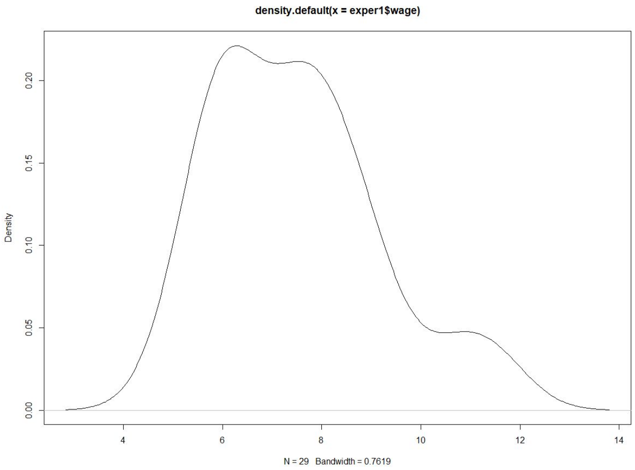
We can make similar conclusions here, with the exception that the mean is depending a little bit more this case.

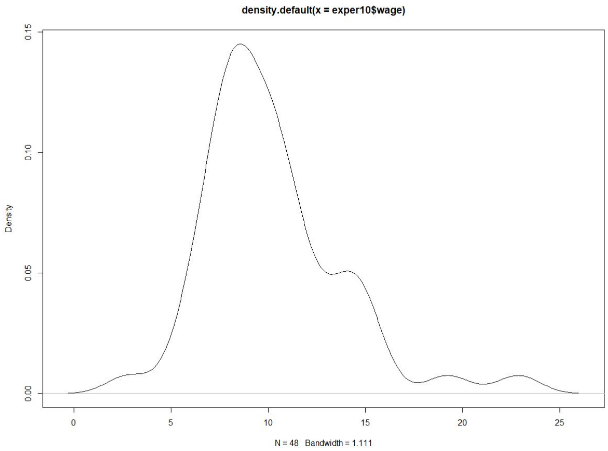
We can test further analyze these observations by separating the dataset on different experience levels.

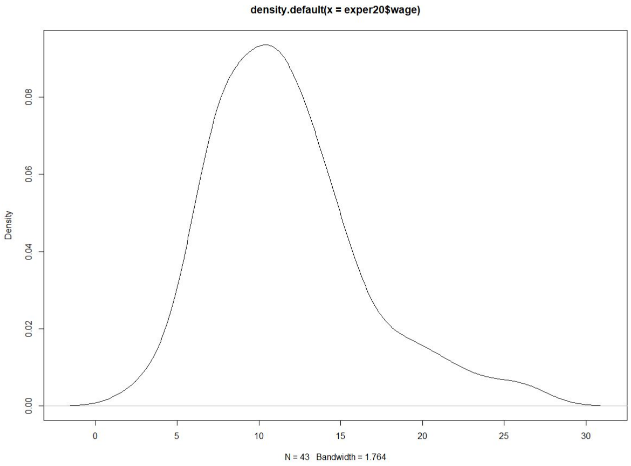
```
# Get the data where the experience is 1, 10, 20, and 30
exper1 = data[(data[, "exper"] == 1),]
exper10 = data[(data[, "exper"] == 10),]
exper20 = data[(data[, "exper"] == 20),]
exper30 = data[(data[, "exper"] == 30),]
```

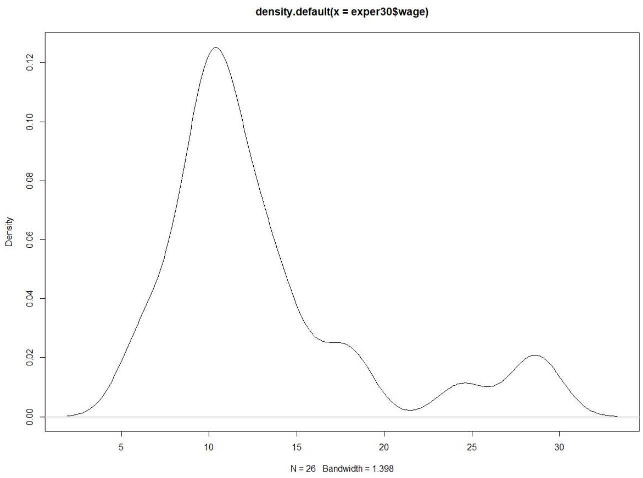
And plotting their wage density functions

```
# Plot the wage density of these experience level datas
plot( density(exper1$wage) )
plot( density(exper10$wage) )
plot( density(exper20$wage) )
plot( density(exper30$wage) )
```







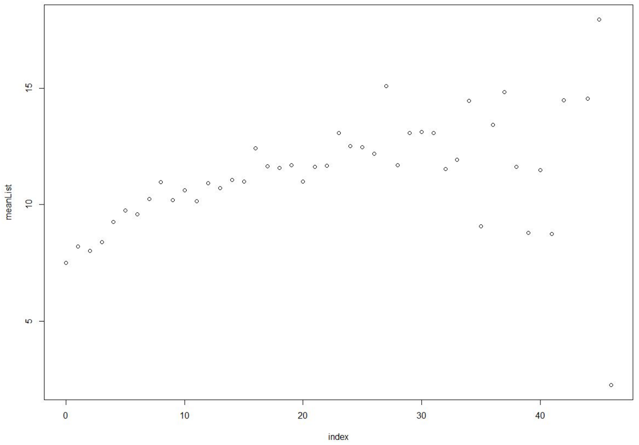


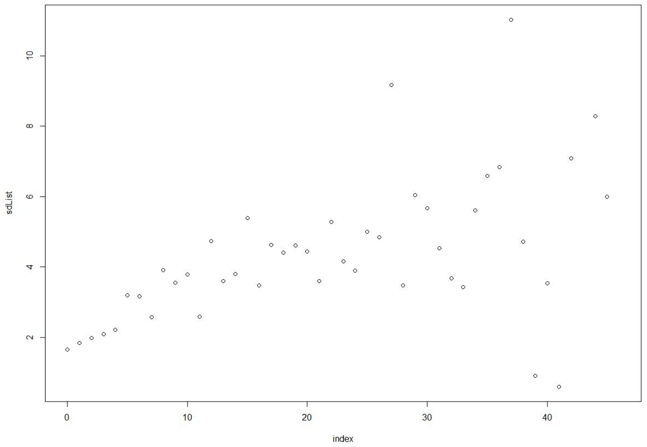
A little movement in the mean, and a wider range in the variance is visible from these diagrams.

Record the means and standard deviances for each experience group.

```
meanList <- list()
sdList <- list()
```

```
for (i in 0:max(data$exper)) {
  meanList[i] <- mean( (data[(data[, "exper"] == i),])$wage )
  sdList[i] <- sd( (data[(data[, "exper"] == i),])$wage )
}
index <- seq(0, 46, 1)
plot(index, meanList)
plot(index, sdList)
```





A small increase in both is also visible here. Although it’s not completely distinct. There are points which are opposing the simple rule that the experience and the education level will surely increase the wage.

4. Postulate a model

The model i would like to set up is considering the education levels as independent categories, and also the experience levels as a parameter to determine the wage.

```
mod_string =
  " model {
    for (i in 1:length(wage)) {
      wage[i] ~ dnorm( mu[i], prec[i] )
      mu[i] = a_mu[educ[i]] + b_mu_exp * exper[i]
      prec[i] = a_prec[educ[i]] + b_prec_exp * exper[i]
    }

    for (j in 1:max(educ)){
      a_mu[j] ~ dnorm(0.0, 1.0/1.0e6)
      a_prec[j] ~ dgamma(1.0/2.0, 1.0*1.0/2.0)
    }

    b_mu_exp ~ dnorm(0.0, 1.0/1.0e6)
    b_prec_exp ~ dgamma(1.0/2.0, 1.0*1.0/2.0)

  } "

data_jags = list(wage=data$wage, educ=data$educ, exper=data$exper)
params = c("a_mu","b_mu_exp","a_prec", "b_prec_exp")
```

The wage has a normal distribution depending on the mu and precision parameters.
The mu parameter is considering the education level in the variable a_mu, and the experience level multiplied by the variable b_mu_exp. The precision parameter is constructed by the same way as the mu, considering both the experience and the education. The educations levels are grouped in 5 different groups.

The a_mu has a normal distribution set to cover a wide range of numbers. The a_prec has a gamma distribution set to the same wide range of numbers. The same can be said about the experience parameters too.

We will monitor the a_mu, b_mu_exp, a_prec, and the b_prec_exp parameters.

5. Fit the model

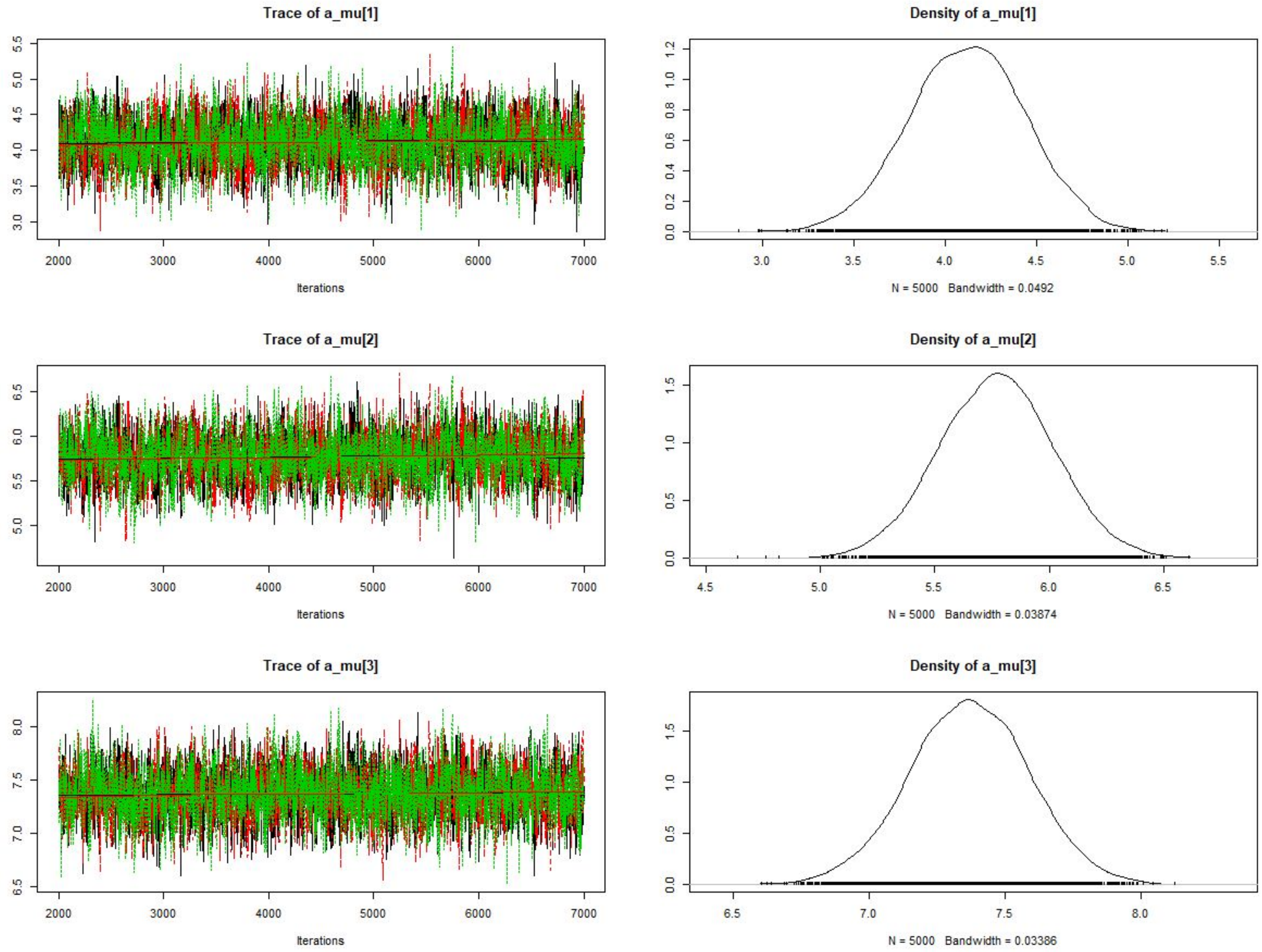
The model was fitted the conventional way, using 3 chains with 5000 iterations each. It was fairly fast, less than one minute.

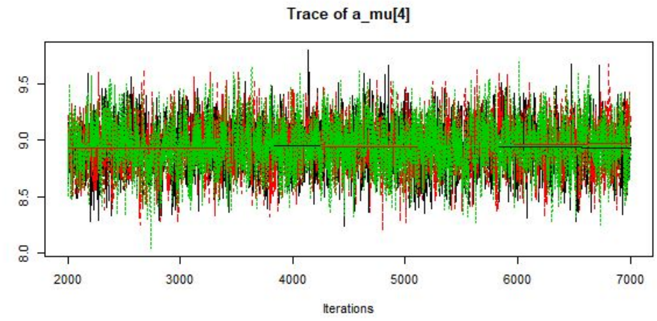
```
mod = jags.model(textConnection(mod_string), data=data_jags, n.chains = 3)
update(mod, 1e3)
mod_sim = coda.samples(model = mod, variable.names = params, n.iter = 5e3)
mod_csim = as.mcmc(do.call(rbind, mod_sim))
```

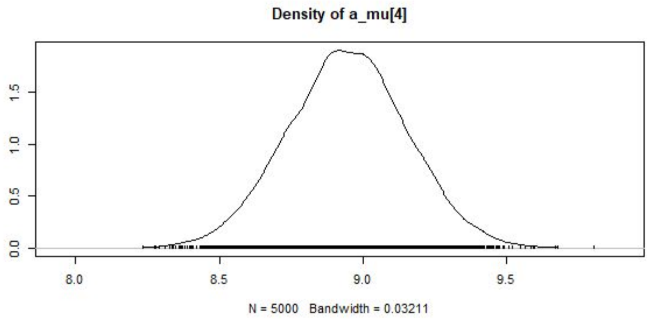
6. Check the model

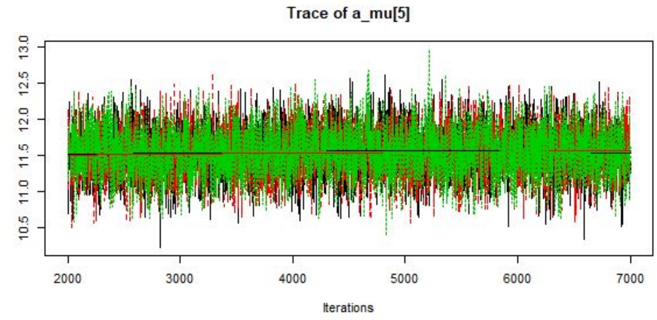
Checking if the model has converged.

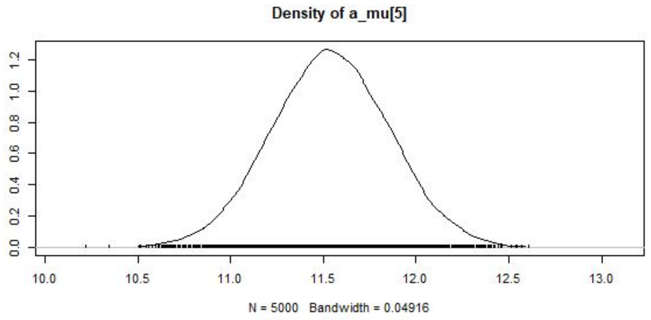
```
# convergence diagnostics
plot(mod_sim, ask=TRUE)
summary(mod_sim)

gelman.diag(mod_sim)
autocorr.diag(mod_sim)
autocorr.plot(mod_sim)
effectiveSize(mod_sim)
```

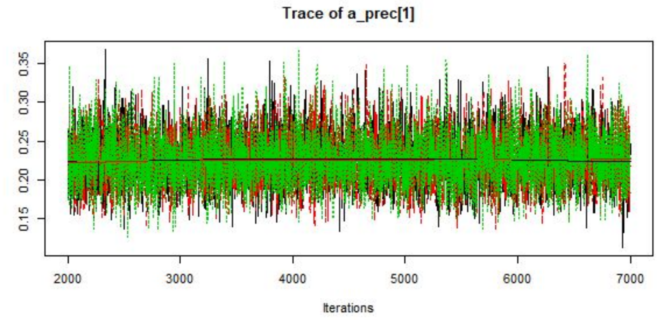


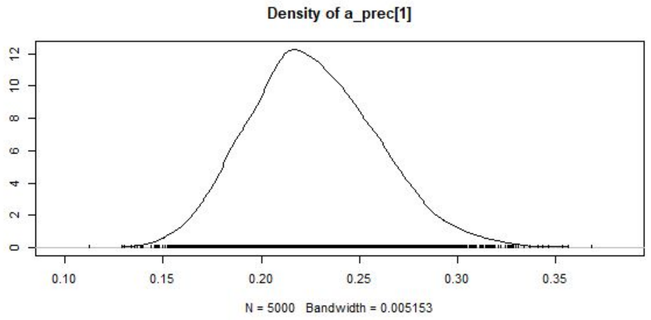

Trace of a_mu[4]
9.5
9.0
8.5
8.0
2000
3000
4000
5000
6000
7000
Iterations

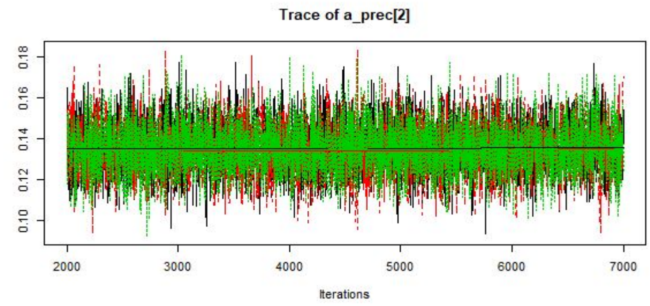

Density of a_mu[4]
1.5
1.0
0.5
0.0
8.0
8.5
9.0
9.5
N = 5000 Bandwidth = 0.03211

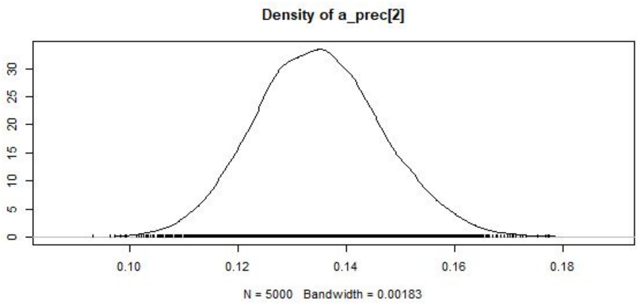

Trace of a_mu[5]
13.0
12.5
12.0
11.5
11.0
10.5
2000
3000
4000
5000
6000
7000
Iterations

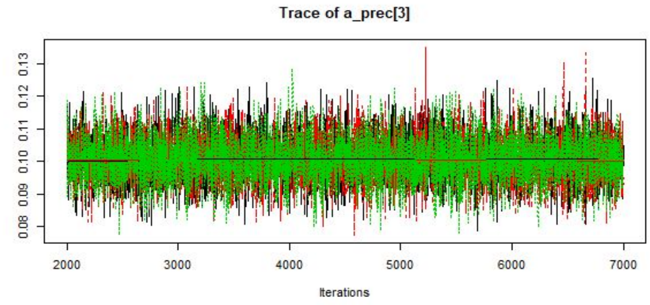

Density of a_mu[5]
1.2
1.0
0.8
0.6
0.4
0.2
0.0
10.0
10.5
11.0
11.5
12.0
12.5
13.0
N = 5000 Bandwidth = 0.04916

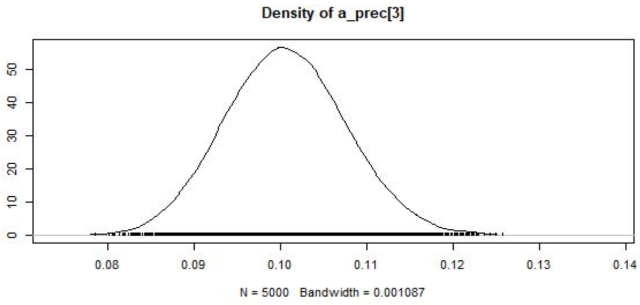

Trace of a_prec[1]
0.35
0.30
0.25
0.20
0.15
2000
3000
4000
5000
6000
7000
Iterations

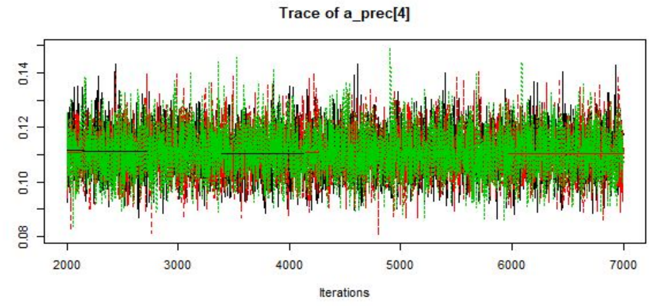

Density of a_prec[1]
12
10
8
6
4
2
0
0.10
0.15
0.20
0.25
0.30
0.35
N = 5000 Bandwidth = 0.005153

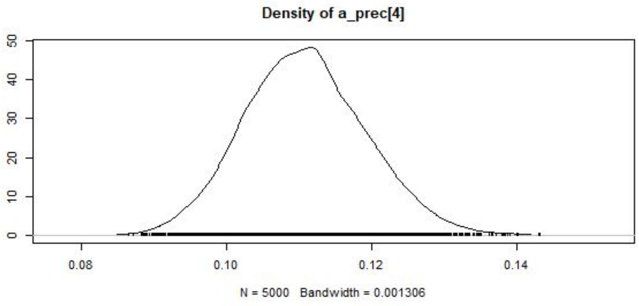

Trace of a_prec[2]
0.18
0.16
0.14
0.12
0.10
2000
3000
4000
5000
6000
7000
Iterations

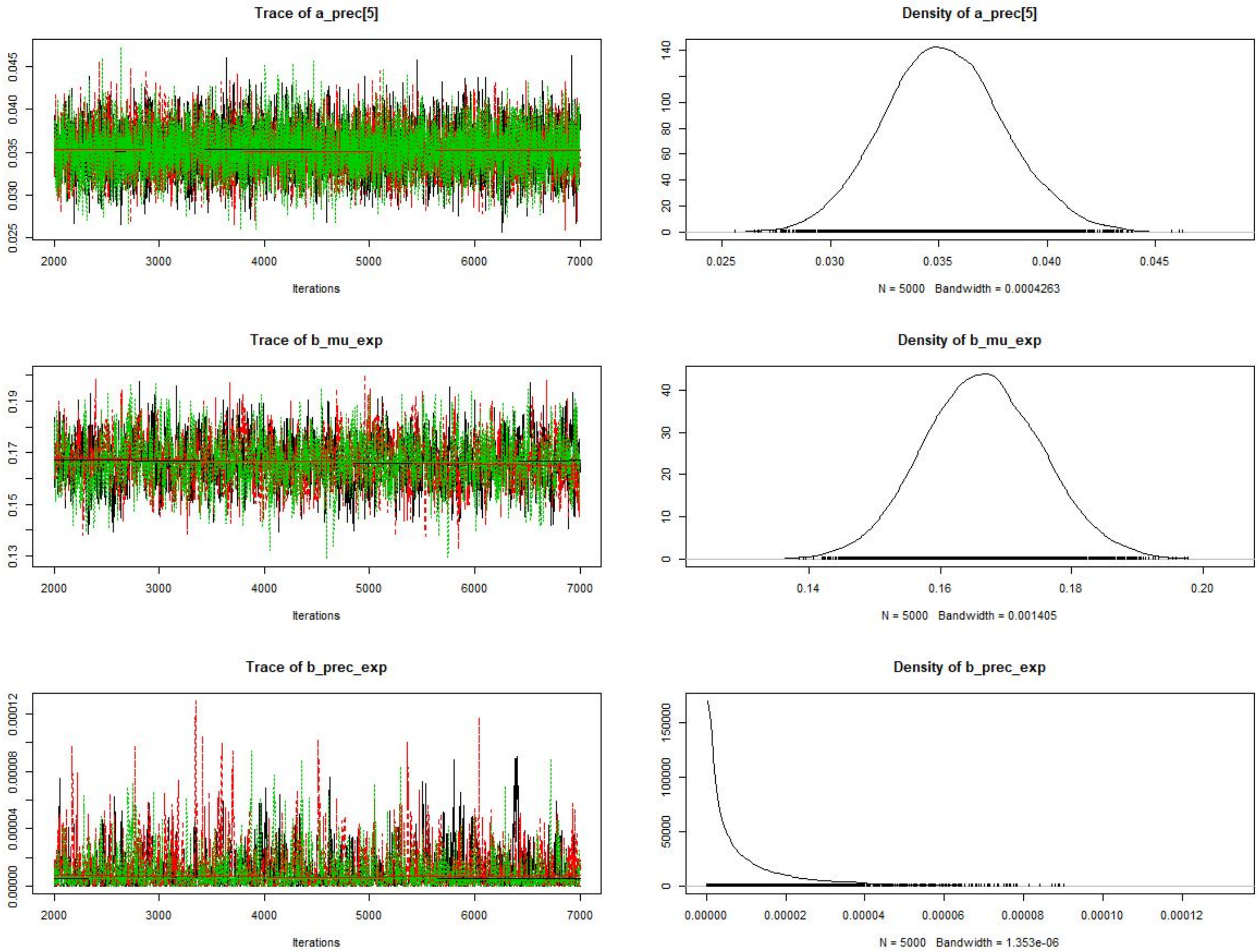

Density of a_prec[2]
30
25
20
15
10
5
0
0.10
0.12
0.14
0.16
0.18
N = 5000 Bandwidth = 0.00183


Trace of a_prec[3]
0.13
0.12
0.11
0.10
0.09
0.08
2000
3000
4000
5000
6000
7000
Iterations


Density of a_prec[3]
50
40
30
20
10
0
0.08
0.09
0.10
0.11
0.12
0.13
0.14
N = 5000 Bandwidth = 0.001087


Trace of a_prec[4]
0.14
0.12
0.10
0.08
2000
3000
4000
5000
6000
7000
Iterations


Density of a_prec[4]
50
40
30
20
10
0
0.08
0.10
0.12
0.14
N = 5000 Bandwidth = 0.001306



Summary

```
             2.5%       25%       50%       75%      97.5%
a_mu[1]    3.493e+00 3.907e+00 4.126e+00 4.337e+00 4.734e+00
a_mu[2]    5.276e+00 5.599e+00 5.771e+00 5.937e+00 6.258e+00
a_mu[3]    6.932e+00 7.221e+00 7.370e+00 7.518e+00 7.794e+00
a_mu[4]    8.534e+00 8.807e+00 8.946e+00 9.085e+00 9.359e+00
a_mu[5]    1.092e+01 1.133e+01 1.154e+01 1.175e+01 1.217e+01
a_prec[1]  1.672e-01 2.038e-01 2.246e-01 2.484e-01 2.968e-01
a_prec[2]  1.125e-01 1.268e-01 1.347e-01 1.426e-01 1.591e-01
a_prec[3]  8.730e-02 9.584e-02 1.005e-01 1.053e-01 1.147e-01
a_prec[4]  9.504e-02 1.050e-01 1.106e-01 1.164e-01 1.282e-01
a_prec[5]  2.994e-02 3.336e-02 3.519e-02 3.705e-02 4.082e-02
b_mu_exp   1.489e-01 1.603e-01 1.664e-01 1.726e-01 1.846e-01
b_prec_exp 1.578e-08 1.144e-06 4.699e-06 1.285e-05 4.865e-05
```

Effective sample sizes

a_mu[1] :	2137.108	a_mu[2]:	2226.767
a_mu[3]:	2444.076	a_mu[4]:	2853.023
a_mu[5]:	5989.371	a_prec[1]:	6877.322
a_prec[2]:	8792.075	a_prec[3]:	9664.895
a_prec[4]:	7784.534	a_prec[5]:	8719.555
b_mu_exp:	1524.889	b_prec_exp:	1448.090

around 1500 sample should be enough the conclude basic properties, like mean and standard deviance.

7. Iterate if necessary

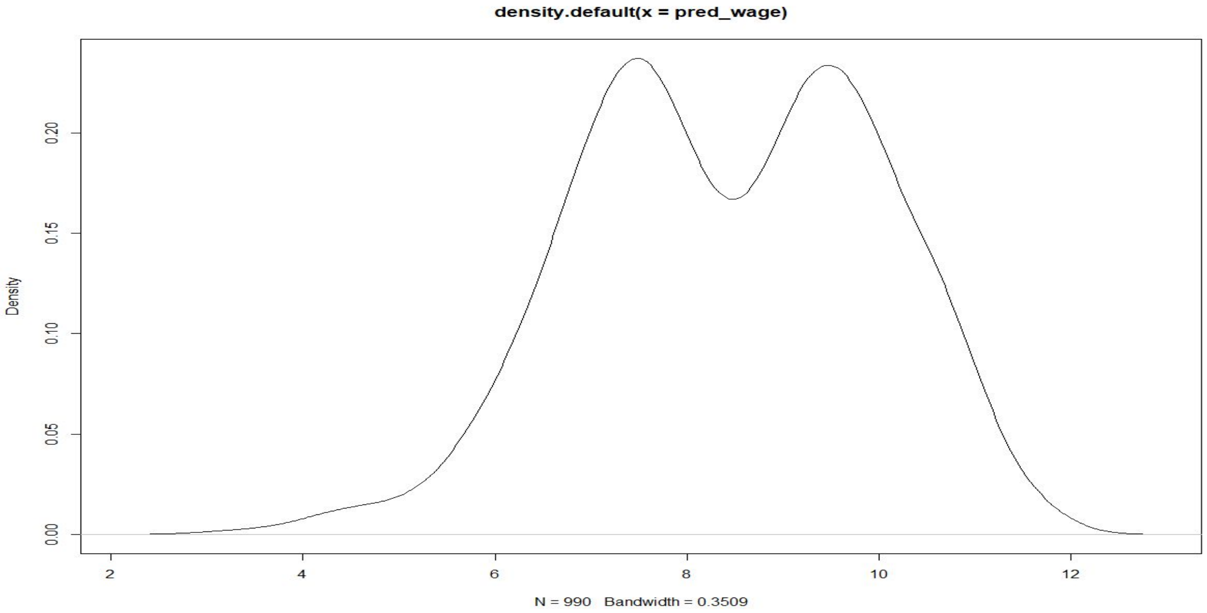
At first i thought the model has some problems with the b_prec_exp parameter, because it is quite small. I converted it to sigma, which was a huge value, and it was a little bit disturbing why does it have such high variance. Then i realized it cannot be taken as the sigma so simply, because it is only a parameter to calculate the sigma from. So the model was correct from this viewpoint.

An improvement to the model would be if i replace the Normal distribution at the wage prediction to a gamma distribution, because the wages has a lot of similar properties as the gamma distribution. For example it cannot be negative, and skewed to the higher values, as it can be seen from the data exploration chapter.

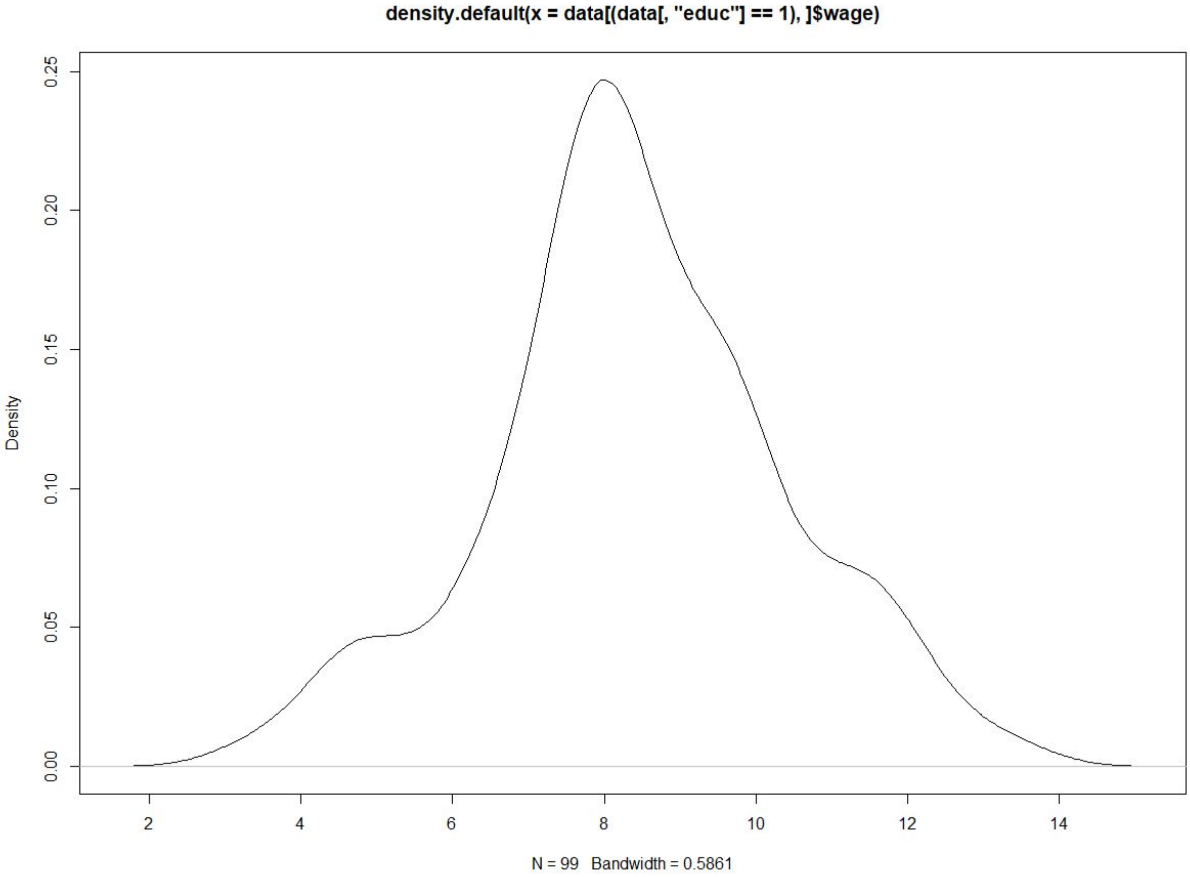
8. Use the model

We can generate predicted values using the values from the data set, this way we can check the residuals too. Using the dataset values where education parameter is equals to 1 to predict and compare. 99 is the datasets which has the education parameter value equals to 1.

```
# Generate predicted mu values
pred_mu = mod_csim[1:(99*10), "a_mu[1]"] + mod_csim[1:(99*10), "b_mu_exp"]
* data[(data[, "educ"] == 1),]$exper

# Generate predicted sig values
pred_sig = mod_csim[1:(99*10), "a_prec[1]"] + mod_csim[1:(99*10),
"b_prec_exp"] * data[(data[, "educ"] == 1),]$exper

# Generate predicted wage value
pred_wage = rnorm(length(pred_sig), mean = pred_mu, sd = pred_sig)

# Check predicted wage values
plot(density(pred_wage))
mean(pred_wage)            ( = 8.429)
sd(pred_wage)              ( = 1.549)
```

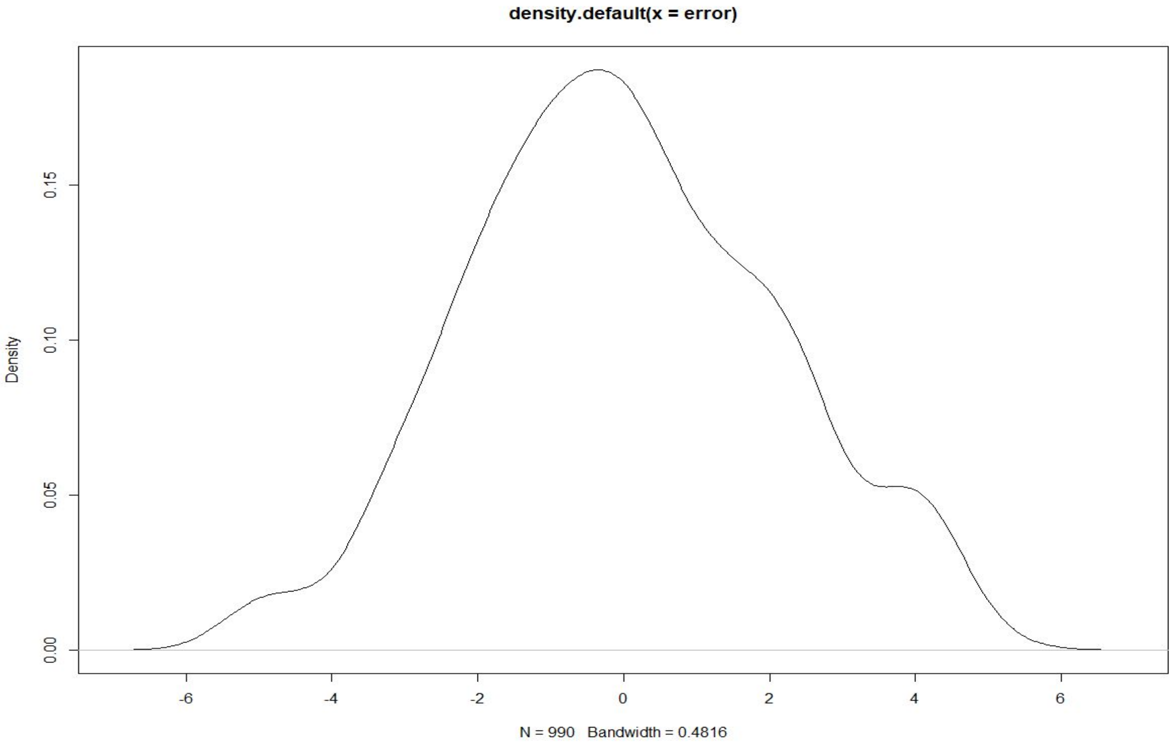


```
# Check original wage values
plot( density( data[(data[, "educ"] == 1),]$wage ) )
mean( data[(data[, "educ"] == 1),]$wage )                ( = 8.429)
sd( data[(data[, "educ"] == 1),]$wage )                  ( = 1.951)
```



Finally we can check the errors

```
# Compare the predicted and original wage values
error = pred_wage - data[(data[, "educ"] == 1),]$wage
# Check the difference between the original and predicted values
plot(error)
plot(density(error))
mean(error)                         ( = -0.00914)
sd(error)                           ( =  2.12583)
```



From this i think we can conclude that the module is good for finding the mean value, but not really to estimate the standard deviation, which is probably coming from the fact that the normal distribution is not the best idea to model the wages.