LSTAT 2120

Introduction to Bayesian Statistics: Project

May 2020
Academic year 2019-2020

Project: Bayesian analysis

Stéphanie Willems
14 38 1500
Clara Debelle
38 10 1700
Alexandre Tytgat
22 06 1500
(Group B)

Philippe Lambert
Oswaldo Gressani

1 Exercise 1

The purpose of this section is to prove this statement validity:

$$if \;\; W_i \sim \chi^2_\nu, \;\; then \;\; \lambda_i = \frac{W_i}{\nu} \sim \mathcal{G}(\frac{\nu}{2}, \frac{\nu}{2})$$

where $\mathcal{G}(a, b)$ denotes the Gamma distribution with mean $\frac{a}{b}$ and variance $\frac{a}{b^2}$

We based our reflexion on the cumulative distribution functions:

Chi-square distribution : $F(X, k) = \frac{\gamma(\frac{k}{2}, \frac{x}{2})}{\Gamma(\frac{k}{2})}$

Gamma distribution : $F(x; a, b) = \frac{\gamma(a, bx)}{\Gamma(a)}$

The cumulative distribution function for the W_i variables is the following :

$$F(W_i) = P(W_i < w) = \frac{\gamma(\frac{\nu}{2}, \frac{w}{2})}{\Gamma(\frac{\nu}{2})}$$

And then,

$$F\left(\frac{W_i}{\nu}\right) = P\left(\frac{W_i}{\nu} < w\right) = P(W_i < w\nu)$$
$$= \frac{\gamma(\frac{\nu}{2}, \frac{w\nu}{2})}{\Gamma(\frac{\nu}{2})}$$

which is the expression of a cumulative distribution of a Gamma distribution $\mathcal{G}(\frac{\nu}{2}, \frac{\nu}{2})$.
As we know that $\lambda_i = \frac{W_i}{\nu}$ then we can affirm that

$$\lambda_i \sim \mathcal{G}(\frac{\nu}{2}, \frac{\nu}{2})$$

2 Exercise 2

(a)

We want to find the distribution of $(Y_i|\beta_0, \tau, \nu, \lambda_i)$ with $Y_i = \beta_0 + \frac{Z_i}{\sqrt{\lambda_i \tau}}$ and $Z_i \sim N(0, 1)$.

We know that if $Z \sim N(\mu, \sigma^2)$ then $aZ + b \sim N(a\mu + b, a^2\sigma^2)$.

Since in our case $Z_i \sim N(0, 1)$ and $a = \frac{1}{\sqrt{\lambda_i \tau}}$, $b = \beta_0$ we get that $Y_i \sim N(\beta_0, \frac{1}{\lambda_i \tau})$.

So the distribution of $(Y_i|\beta_0, \tau, \nu, \lambda_i)$ is a Normal with mean β_0 and variance $\frac{1}{\lambda_i \tau}$.

(b)

We want to find the joint posterior density $p(\beta_0, \tau, \lambda_1, ..., \lambda_n|D)$ and we know that the joint prior of β_0 and τ is equal to : $p(\beta_0, \tau) \propto \tau^{-1}$ and that $\lambda_i \sim \mathcal{G}(\frac{\nu}{2}, \frac{\nu}{2})$.

From that, we can calculate the prior for λ_i and we get:

$$p(\lambda_i) \propto \lambda_i^{\frac{\nu}{2}-1} exp(\frac{-\nu}{2}\lambda_i)$$

To find the joint posterior density, we also need to find the likelihood of Y_i.

Since $Y_i \sim N(\beta_0, \frac{1}{\lambda_i \tau})$ we get the following likelihood:

$$\prod_{i=1}^{n} \sqrt{\lambda_i \tau} \exp\{\frac{-\lambda_i \tau}{2}(Y_i - \beta_0)^2\}$$

And thus,

$$p(\beta_0, \tau, \lambda_1, ..., \lambda_n | D) \propto \text{Likelihood x Priors}$$

$$p(\beta_0, \tau, \lambda_1, ..., \lambda_n | D) \propto p(D|\beta_0, \tau, \lambda_1, ..., \lambda_n) p(\beta_0, \tau) p(\lambda_i)$$

$$p(\beta_0, \tau, \lambda_1, ..., \lambda_n | D) \propto \prod_{i=1}^{n} [\sqrt{\lambda_i \tau} \exp\{\frac{-\lambda_i \tau}{2}(Y_i - \beta_0)^2\}] \tau^{-1} \prod_{i=1}^{n} [\lambda_i^{\frac{\nu}{2}-1} exp(\frac{-\nu}{2}\lambda_i)]$$

$$p(\beta_0, \tau, \lambda_1, ..., \lambda_n | D) \propto \tau^{\frac{n}{2}-1} \prod_{i=1}^{n} \sqrt{\lambda_i} \exp\{\sum_{i=1}^{n} \frac{-\lambda_i \tau}{2}(Y_i - \beta_0)^2\} \prod_{i=1}^{n} \lambda_i^{\frac{\nu}{2}-1} exp(\frac{-\nu}{2} \sum_{i=1}^{n} \lambda_i)$$

$$p(\beta_0, \tau, \lambda_1, ..., \lambda_n | D) \propto \tau^{\frac{n}{2}-1} \exp\{\sum_{i=1}^{n} \frac{-\lambda_i \tau}{2}(Y_i - \beta_0)^2\} \prod_{i=1}^{n} \lambda_i^{\frac{\nu-1}{2}} exp(\frac{-\nu}{2} \sum_{i=1}^{n} \lambda_i)$$

(c)

The objective now is to obtain the conditional posterior distributions for each parameters from the joint posterior density found in the previous point. This is done by considering the other parameters as given fixed quantities.

- The conditional posterior for β_0 is given by

$$p(\beta_0 | \tau, \lambda_1, ..., \lambda_n, D) \propto \exp\{\sum_{i=1}^{n} \frac{-\lambda_i \tau}{2}(Y_i - \beta_0)^2\}$$

  So we know that

$$(\beta_0 | \tau, \lambda_1, ..., \lambda_n, D) \sim \mathcal{N}(\mu_{post}, (\tau_{post})^{-1})$$

  where

$$\mu_{post} = \text{argmax}_{\mu_{post}} \log(\exp\{\sum_{i=1}^{n} \frac{-\lambda_i \tau}{2}(Y_i - \mu_{post})^2\})$$

$$\mu_{post} = \text{argmax}_{\mu_{post}} \sum_{i=1}^{n} \frac{-\lambda_i \tau}{2}(Y_i - \mu_{post})^2$$

$$\Rightarrow \sum_{i=1}^{n} \lambda_i \tau (Y_i - \mu_{post}) = 0$$

$$\Rightarrow \mu_{post} = \frac{\sum_{i=1}^{n} \lambda_i Y_i}{\sum_{i=1}^{n} \lambda_i}$$

And

$$\tau_{post} = \frac{-\partial^2 \log(\exp\{\sum_{i=1}^{n} \frac{-\lambda_i \tau}{2} (Y_i - \beta_0)^2\})}{\partial \beta_0^2}$$

$$\tau_{post} = \frac{-\partial \sum_{i=1}^{n} \lambda_i \tau (Y_i - \beta_0)}{\partial \beta_0}$$

$$\Rightarrow \tau_{post} = \sum_{i=1}^{n} \lambda_i \tau$$

Hence, we conclude that

$$(\beta_0 | \tau, \lambda_1, ..., \lambda_n, D) \sim \mathcal{N}(\frac{\sum_{i=1}^{n} \lambda_i Y_i}{\sum_{i=1}^{n} \lambda_i}, (\sum_{i=1}^{n} \lambda_i \tau)^{-1})$$

- The conditional posterior for τ is given by

$$p(\tau | \beta_0, \lambda_1, ..., \lambda_n, D) \propto \tau^{\frac{n}{2}-1} \exp\{\sum_{i=1}^{n} \frac{-\lambda_i \tau}{2} (Y_i - \beta_0)^2\}$$

Hence, we conclude that

$$(\tau | \beta_0, \lambda_1, ..., \lambda_n, D) \sim \mathcal{G}(\frac{n}{2}, \sum_{i=1}^{n} \frac{\lambda_i}{2} (Y_i - \beta_0)^2\})$$

- The conditional posterior for λ_i is given by

$$p(\lambda_i | \beta_0, \tau, D) \propto \lambda_i^{\frac{\nu-1}{2}} \exp\{\frac{-\lambda_i \tau}{2} (Y_i - \beta_0)^2 - \frac{\nu}{2} \lambda_i\}$$

Hence, we conclude that

$$(\lambda_i | \beta_0, \tau, D) \sim \mathcal{G}(\frac{\nu + 1}{2}, \frac{\tau}{2} (Y_i - \beta_0)^2 + \frac{\nu}{2})$$

(d)

The purpose of Gibbs sampler algorithm is to construct a random sample from the joint posterior. In our case, this algorithm produces the samples $(\beta_0^t, \tau^t, \lambda_1^t, ..., \lambda_n^t)$ at iteration t from the conditional posterior distribution of each parameter. Each parameter is updated conditionally on the last available updates for the other components.
After a large number of iterations, lets say T, the so-constructed set

$$\{(\beta_0^{T+1}, \tau^{T+1}, \lambda_1^{T+1}, ..., \lambda_n^{T+1}), ..., (\beta_0^{t+M}, \tau^{t+M}, \lambda_1^{t+M}, ..., \lambda_n^{t+M})\}$$

will be a random sample from the joint posterior $p(\beta_0, \tau, \lambda_1, ..., \lambda_n|D)$.

To implement this Gibbs sampler, we designed the following iterative algorithm:

Gibbs algorithm

Input: initial values for each parameters
Output: a random sample for each parameter

For t=2 to M
{for i=1 to n
Sample λ_i^t from $\mathcal{G}(\frac{\nu+1}{2}, \frac{\tau^{t-1}}{2}(Y_i - \beta_0^{t-1})^2 + \frac{\nu}{2})$
}
Sample β_0^t from $\mathcal{N}(\frac{\sum_{i=1}^n \lambda_i^t Y_i}{\sum_{i=1}^n \lambda_i^t}, (\sum_{i=1}^n \lambda_i^t \tau^{t-1})^{-1})$

Sample τ^t from $\mathcal{G}(\frac{n}{2}, \sum_{i=1}^n \frac{\lambda_i^t}{2}(Y_i - \beta_0^t)^2)$

End for

(e)

The purpose of this subsection, is to prove for β_0 & τ the convergence of the chains with two different sets of initial conditions for $\beta_0, \tau, \lambda_1, ..., \lambda_n$:

- (140, 3, 0.5, ..., 0.5)
- (200, 7, 2, ..., 2)

We implemented the previous Gibbs algorithm defined in the subsection *2.d* into R with some additional information about the different variables (see code in annex):

- nu =3
- n =500
- y = 170 + 5*rt(n,df= nu)

The Figures 1 and 2 show the convergence of the two chains for both of the variables of interest. Below each figure, there is a table displaying the 95% credible interval for each variable.

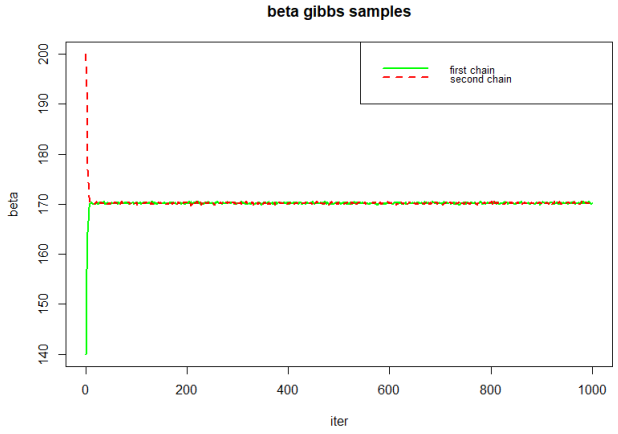

Figure 1: Convergence of the two chains for β_0

Considering this graphic, the mean over the two chains for β_0 is 170.2317, which is included in the credible interval of Table 1.

	0.025	0.975
Values	169.9428	170.4951

Table 1: Credible interval for β_0 defined by the 2,5% and 97,5% quantiles

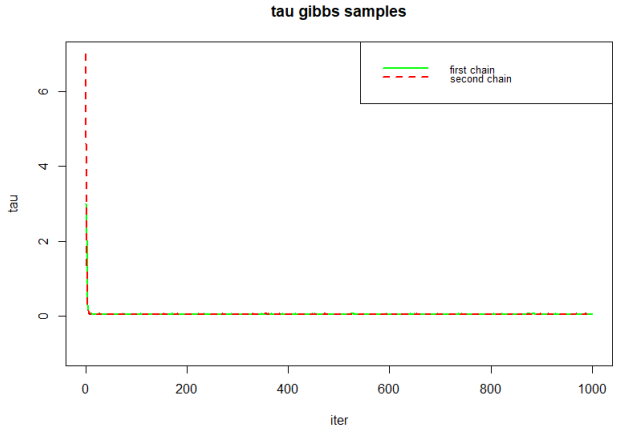

Figure 2: Convergence of the two chains for τ

Considering this graphic, the estimate value for τ is 0.0547, again obtained by the mean over the two chains, which is included in the credible interval of Table 2.

In order to support the convergence, we use the Gelman-Rubin diagnostic tool which monitors the convergence through a "scale reduction factors" ($\hat{R}$). This factor is measured by the difference between the variance within the two chains and the variance between the two chains.

	0.025	0.975
Values	0.0389	0.0561

Table 2: Credible interval for τ defined by the 2,5% and 97,5% quantiles

This diagnostic gives a scale reduction factor for each variable of interest which, in our case, means β_0 and τ.

A scale reduction factor of 1 means that the between variance and the within chain variance are equivalent. If the factor gets a larger value that means there is a notable difference between the two chains. Generally, factors below 1.1 stands for a convergence for the variable of interest between the two chains. The scale reduction factor for β_0 and τ are given below. Both factors are below 1.1. The convergence is thus supported.

	β_0	τ
$\hat{R}$	1.0062	0.9991

Table 3: Scale reduction factors for β_0 and τ

The Gelman plots in Figure 3 exhibit the development of the scale reduction factor over time which is useful to detect whether the factor is stable. The Gelman plots of the two parameters demonstrate a factor stable over time as the variation of the shrink factor is mainly between 1 and 1.1 and is further reduced at iteration 800 and forward.

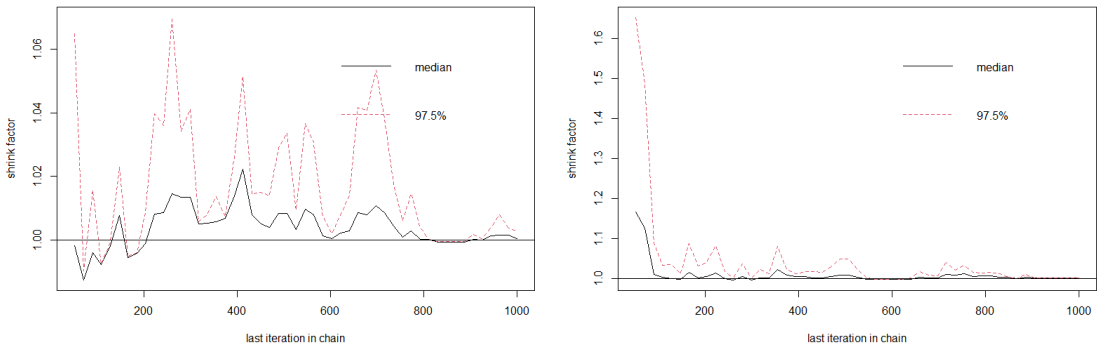

Figure 3: Gelman-Rubin diagnostic plot for β_0 and τ respectively

(f)

The purpose of this subsection is similar to the previous one but instead of using pure R code, it is required to work with JAGS code. At the end a comparison between the results obtained with our implementation of the Gibbs algorithm and the one from JAGS will be made.
The following Figures 4 and 5 display the iteration of the two chains and grant the convergence of them for both of the variables of interest. Note that 10 000 iterations are used instead of 1000 like previously. Below each figure, the Tables 4 and 5 display the 95% credible interval for each variable given by the 2.5% and 97,5% quantiles.

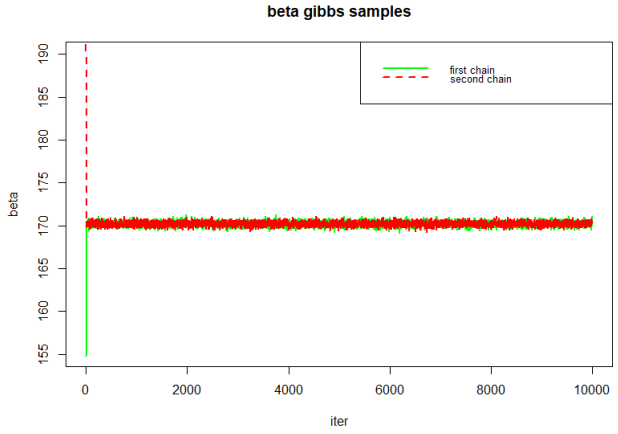

Figure 4: Convergence of the two chains for β_0 obtained with JAGS

Considering this graphic, the estimate value for β_0 is 170.2188 which is included in the credible interval.

	0.025	0.975
Values	169.7237	170.7223

Table 4: Credible interval for β_0

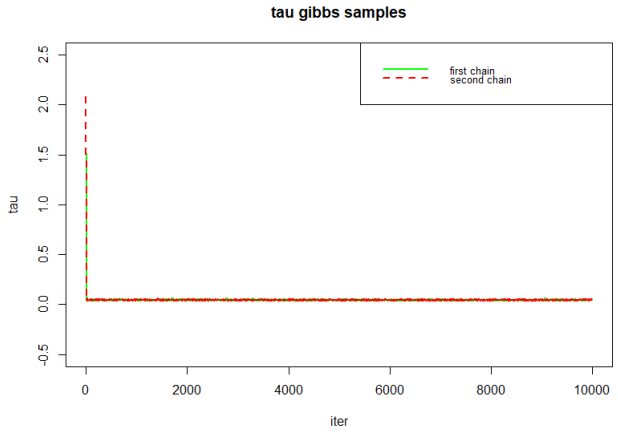

Figure 5: Convergence of the two chains for τ obtained with JAGS

Considering this graphic, the estimate value for τ is 0.0466 which is included in the credible interval.

	0.025	0.975
Values	0.0386	0.0553

Table 5: Credible interval for τ

No matter the implementation used for the Gibbs sampler, the results are similar for both β_0 and τ. However, the results should be slightly better with JAGS. Indeed, the chain is cut after a large number of iterations T in order to have a random sample once the chain has converged. Whereas in our implementation of the Gibbs sampler, the whole chain is used for predictions. However, since the chain converges quickly, the difference is small.

As in the previous subsection, in order to support the convergence we used the Gelman-Rubin diagnostic tool. The scale reduction factor for β_0 and τ are given below. Both factors are below 1.1. The convergence is again supported as it can be seen in Table 6.

	β_0	τ
$\hat{R}$	1	1

Table 6: Scale reduction factors for β_0 and τ

In addition to the Gelman Gubin diagnostic, we also implemented the Geweke diagnostic to corroborate the convergence. The Geweke graphics are given in the appendix (Figures 12- 13).

Comparatively to our previous scale reduction factor obtained with a pure R code, they are close to each other respectively. The Figure 6 shows the Gelman diagnostic plots for β_0 and τ and confirm the convergence. The shrink factor is even lower for τ than it was with one obtained from our Gibbs sampler implementation.

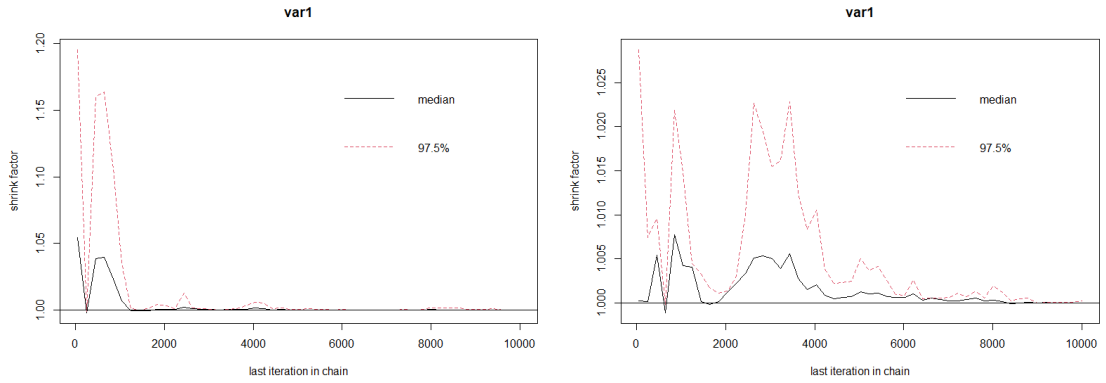

Figure 6: Gelman-Rubin diagnostic plot obtained with JAGS for β_0 and τ respectively

3 Exercise 3

(a)

In this section, we consider a dataset relating the total body mass (in kg) with the mass of the brain (in g) for 28 animals. The goal is to build a linear model from this dataset from a bayesian

approach and then compare it to a classical frequentist linear regression.
The model used is the following :

$$Y_i = \beta_0 + \beta_1(x_i - \bar{x}) + \tau^{-1/2}\epsilon_i$$

with $Y_i = \log(\text{brainweight})$, $x_i = \log(\text{bodyweight})$ and $\epsilon_i \sim t_\nu$ i.i.d. $(i = 1, ..., n)$.

From the results of section 1 and 2, the model above can be expressed as :

$$Y_i = \beta_0 + \beta_1(x_i - \bar{x}) + \frac{Z_i}{\sqrt{\lambda_i \tau}}$$

with $\lambda_i \sim \mathcal{G}(\frac{\nu}{2}, \frac{\nu}{2})$ i.i.d. and $Z_i \sim \mathcal{N}(0, 1)$. From now, it is assumed that $\nu = 3$ to simplify. From the same reasoning as used in the subsection 2.a), it is obvious that

$$(Y_i|\beta_0, \beta_1, \tau, \lambda_i) \sim \mathcal{N}(\beta_0 + \beta_1(x_i - \bar{x}), \frac{1}{\lambda_i \tau})$$

and thus the likelihood is given by :

$$p(D|\beta_0, \beta_1, \tau, \lambda_1, ..., \lambda_n) = \prod_{i=1}^{n} \sqrt{\lambda_i \tau} \exp\{\frac{-\lambda_i \tau}{2}(Y_i - (\beta_0 + \beta_1(x_i - \bar{x})))^2\}$$

The uninformative joint prior density $p(\boldsymbol{\beta}, \tau) \propto \tau^{-1}$ is used in this model for τ and $\boldsymbol{\beta} = (\beta_0, \beta_1)'$ as done in the section 2. As mentionned previously, the prior for λ_i is assumed to be the following form,

$$p(\lambda_i) \propto \lambda_i^{\frac{\nu}{2}-1} exp(\frac{-\nu}{2}\lambda_i)$$

for $i = 1, ..., n$.
Finally, the posterior density can be expressed as :

$$\begin{aligned} p(\beta_0, \beta_1, \tau, \lambda_1, ..., \lambda_n|D) &\propto \text{Likelihood x Priors} \\ &= p(D|\beta_0, \beta_1, \tau, \lambda_1, ..., \lambda_n)p(\boldsymbol{\beta}, \tau)p(\lambda_i) \end{aligned}$$

The JAGS software was used to sample the joint posterior density and the code is in the appendix.

(b)

The Figures 7 and 8 show the trace plots of the chains obtained for β_0, β_1, τ when the following starting values are used $(\beta_0, \beta_1, \tau, \lambda_1, ..., \lambda_n) = (3, 0, 0.4, 0.5, ..., 0.5)$. We observe that the three chains seems to converge as the variation is focused in relatively small intervals for each parameter.

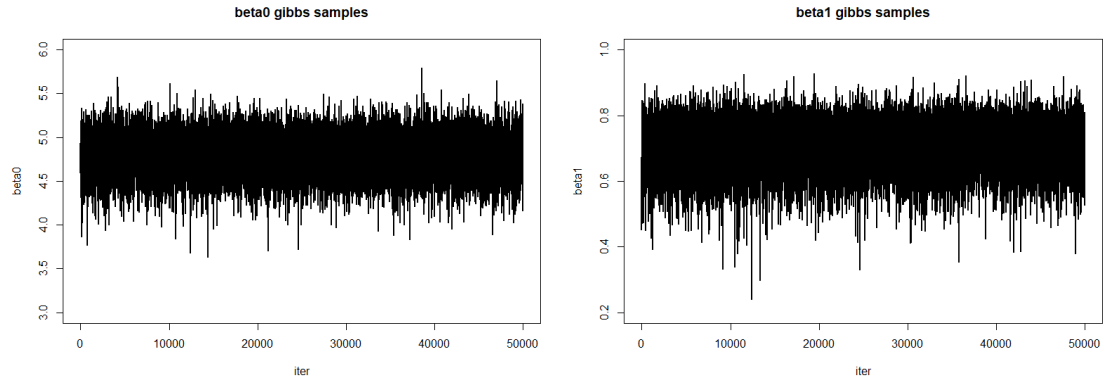

Figure 7: Trace plot of the chain for β_0 and β_1 respectively

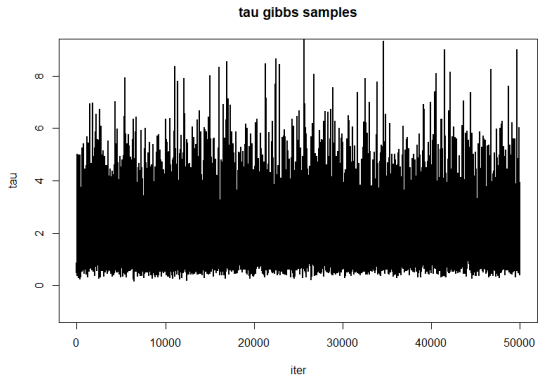

Figure 8: Trace plot of the chain for τ

This confirms that the algorithm converges pretty well and pretty fast.
In order to confirm further the convergence of the chains, the Geweke diagnostic was used. The Geweke is a diagnostic used to asses the convergence of a single chain. The diagnostic is based on a test for equality of means which are the mean of first 10% of the generated chain and the mean of the second half of the chain. To assume the independence of the two blocks, a consequent buffer is got between them. The Figure 9 and 10 show the Geweke diagnostic plot for 200 000 iterations with 20 bins. An 95% credible interval is used, thus it is expected that on average only one test will be in the rejection zone for each chain. As it can be observed, only one test is rejected for the chain of the τ parameter. Thus, we conclude that the convergence is achieved.

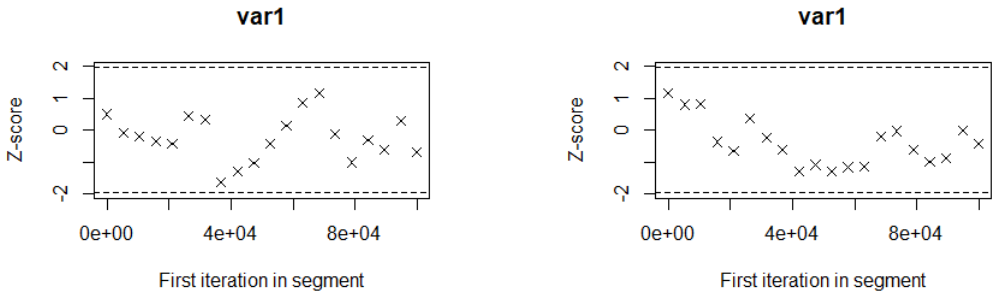

Figure 9: Geweke diagnostic plot of the chain for β_0 and β_1 respectively

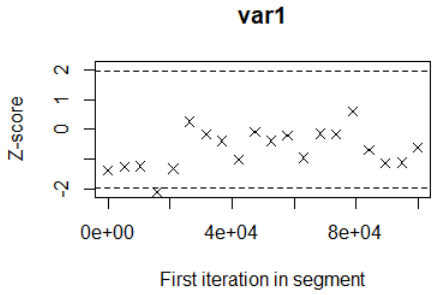

Figure 10: Geweke diagnostic plot of the chain for τ

(c)

Considering the previous samples obtained, it is possible to use them in order to compute point estimates and 95% HPD credible intervals for β_0, β_1 and τ. The Table 7 gathers the results obtained with JAGS using the mean as points estimates for each parameter of interest and the function `HPDinterval` is used to compute the 95% HDP confidence interval.

	mean	lower bound	upper bound
β_0	4.7835	4.4189	5.1375
β_1	0.709	0.5882	0.8189
τ	1.9067	0.4609	3.7436

Table 7: Point estimates and 95% HDP CI for each parameter of interest.

(d)

Finally, it is time to evaluate our bayesian model. For this purpose, for each value x_k in the x interval with $k = 1, ..., K$, the conditional mean $\mu_k = \beta_0 + \beta_1(x_k - \bar{x})$ is sampled along the

other parameters with JAGS. Afterward, the mean of each μ_k is taken over the M iterations and used for the regression. This procedure is repeated for $\nu = 3, 5, 10, 15, 25$ and thus yields five regression lines.

As mentioned in the beginning of this section, we also wish to compare these results with the regression line obtained using the classical frequentist linear regression model. To achieve this, the R function `lm()` is used to produce a linear model fitted to the brain dataset.

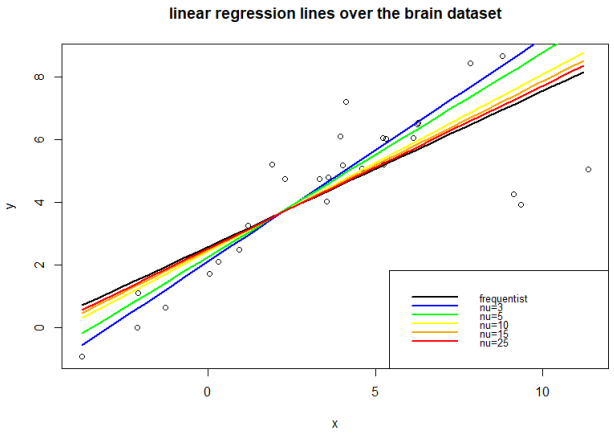

Figure 11: Regression lines obtained from a classical frequentist approach and from a bayesian approach with five different values of ν.

First, Figure 11 shows that as the values of ν increase the intercept β_0 increases as well whereas the slope β_1 decreases. It might be because Bayesian model built with high ν values are more sensible to outliers. Indeed, it can be seen that the higher ν is, the closer the lines are to the three outliers on the right of Figure 11. It is also interesting to note that those data points corresponds to the Diplodocus, Triceratops and Brachiosorus species.
Those species have huge body weight values but very low brain weight values. Furthermore, they do not belong to mammals as do the rest of the species in the dataset. Excluding those data from the model might be interesting if one wishes to get better predictions on mammal species. Secondly, the regression line obtained with the frequentist approach is notably the line with the lowest slope and the highest intercept. As stated above, it might be because this model is more sensitive to outliers.

In the Table 8 the values of the RMSE, MSE, MAE are reprensented for the frequentist approach and for different value of ν. We observe that the RMSE and the MSE are better for higher value of ν . But the frequentist approach gives better result for both MSE and RMSE.
On the contrary, the bayesian regression lines with lower ν perform better with respect to the MAE metric. The reason is probably due to the fact that the RMSE and MSE take the square of the error. Thus, outliers have a higher impact on RMSE and MSE metrics than on MAE. Thus, a regression line that is "closer" to the outliers will have lower RMSE and MSE values. However, it might not be a good thing because they will tend to overfit the data more than regression line that are more resistant to outliers. Thus, we conclude that the Bayesian approach yields better results as it can generalize better.

	frequentist	$\nu = 3$	$\nu = 5$	$\nu = 10$	$\nu = 15$	$\nu = 25$
RMSE	1.4759	1.7125	1.6078	1.5032	1.486	1.479
MSE	2.178	2.9331	2.5851	2.2598	2.2065	2.1873
MAE	1.1697	0.9885	1.01	1.0775	1.1143	1.1372

Table 8: Evaluation of the regressions with different metrics : RMSE, MSE and MAE.

4 Appendix

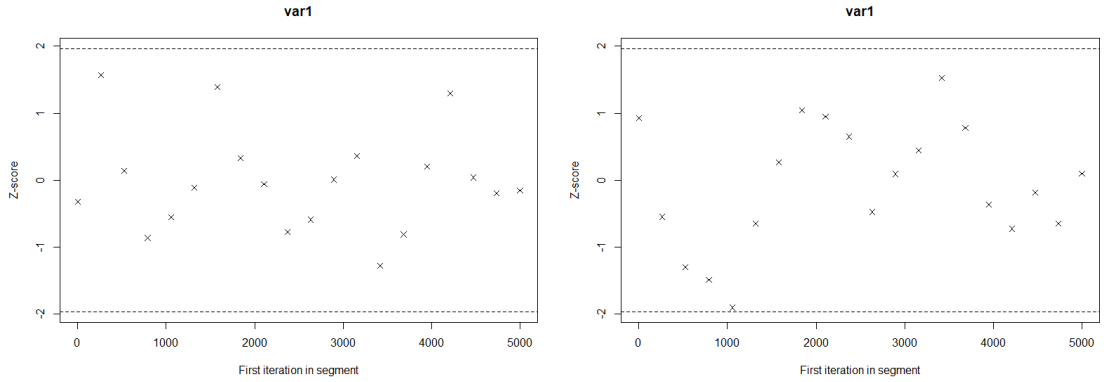

Figure 12: Geweke diagnostic plot of β_0 for the first and the second chain respectively

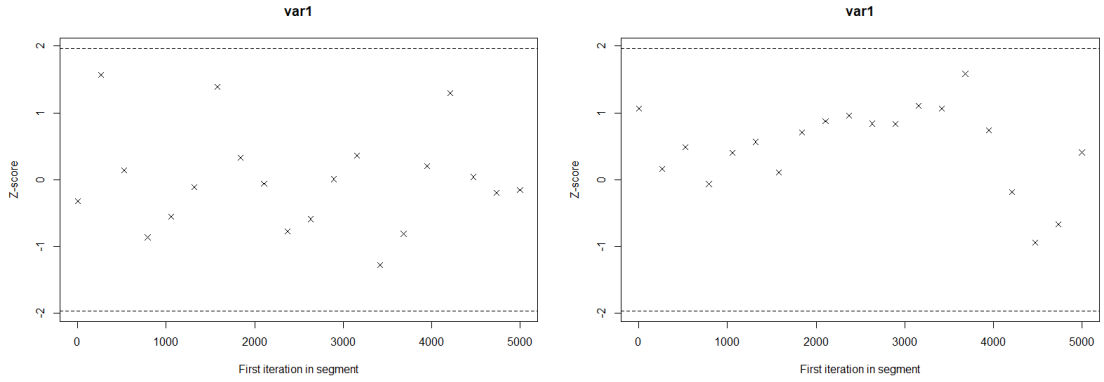

Figure 13: Geweke diagnostic plot of τ for the first and the second chain respectively

4.1 R code

```
#####################################################################
#       Q2 e) : Gibbs algorithm and Gelman-Rubin diagnostic         #
#####################################################################
```

```
rm(list=ls())

########################
#    GIBBS SAMPLING    #
########################

# parameters initialization
set.seed(1234)
nu = 3; n = 500
y = 170 + 5*rt(n, df=nu)
M = 1000    # number of samples


# compute the mean of the rv beta
beta.mean <- function(y.samples, lambda.samples) {
  y.samples %*% lambda.samples / sum(lambda.samples)
}

# compute the variance of the rv beta
beta.var <- function(lambda.samples, tau.sample) {
  ( tau.sample * sum(lambda.samples) )^-1
}

# compute the rate (parameter b of the gamma distribtion) of rv tau
tau.rate <- function(beta.sample, y.samples, lambda.samples) {
  ( y.samples - beta.sample )^2 %*%  lambda.samples/2
}



######################
#    FIRST CHAIN    #
######################

# parameters initialization
beta_1 = tau_1 = rep(0, M)
lbds_1 = matrix(nrow = M, ncol = n)


# initial values
beta_1[1] = 140
tau_1[1] = 3
lbds_1[1,] = rep(0.5, n)


# Gibbs sampling
for (i in 2:M) {

  # new samples lambdas
  for (j in 1:n){
    lbds_1[i,j] = rgamma( 1, (nu+1)/2, tau_1[i-1]/2 * (y[j]-beta_1[i-1])^2 + nu/2 )
  }

  # new sample beta
  beta_1[i] = rnorm( 1, beta.mean(y, lbds_1[i,]), beta.var(lbds_1[i,], tau_1[i-1])  )

  # new sample tau
  tau_1[i] = rgamma( 1, n/2, tau.rate(beta_1[i], y, lbds_1[i,]) )

}



######################
#    SECOND CHAIN    #
######################

# parameters initialization
beta_2 = tau_2 = rep(0, M)
lbds_2 = matrix(nrow = M, ncol = n)


# initial values
```

```
beta_2[1] = 200
tau_2[1] = 7
lbds_2[1,] = rep(2, n)


# Gibbs sampling
for (i in 2:M) {

  # new samples lambdas
  for (j in 1:n){
    lbds_2[i,j] = rgamma( 1, (nu+1)/2, tau_2[i-1]/2 * (y[j]-beta_2[i-1])^2 + nu/2 )
  }

  # new sample beta
  beta_2[i] = rnorm( 1, beta.mean(y, lbds_2[i,]), beta.var(lbds_2[i,], tau_2[i-1])  )

  # new sample tau
  tau_2[i] = rgamma( 1, n/2, tau.rate(beta_2[i], y, lbds_2[i,]) )

}



########################
#      TRACE PLOT      #
########################

# trace plot for beta samples
plot(1:M, beta_1, type="l", lwd=2, col="green", xlab="iter", ylab="beta", ylim=range(140,200)
     ,  main="beta gibbs samples")
lines(1:M, beta_2, type="l", col="red", lwd=2, lty=2)
legend("topright",c("first chain","second chain"), lwd=c(2,2), col=c("green","red"), box.lty
     =1, lty=1:5, cex=0.8, y.intersp=0.2)

# trace plot for tau samples
plot(1:M, tau_1, type="l", lwd=2, col="green", xlab="iter", ylab="tau", ylim=range(-1,7),
     main="tau gibbs samples")
lines(1:M, tau_2, type="l", col="red", lwd=2, lty=2)
legend("topright",c("first chain","second chain"), lwd=c(2,2), col=c("green","red"), box.lty
     =1, lty=1:5, cex=0.8, y.intersp=0.2)



########################################
#       GELMAN-RUBIN DIAGNOSTIC        #
########################################
# start at iteration T < M ?


gelman.diagnostic <- function(beta) {
  # Input : parameter matrix size (I,J)
  # Output : R hat value

  I = dim(beta)[1]
  J = dim(beta)[2]

  # compute the mean of the J chains over the I iterations
  beta.mean.J = rep(0, J)
  for (j in 1:J) {
    beta.mean.J[j] = mean(beta[,j])
  }

  # compute mean over all indices (sequences and iterations)
  beta.mean = mean(beta.mean.J)

  # Between-sequence estimator
  B = I/(J-1) * sum( (beta.mean.J - beta.mean)^2 )

  # compute s_j^2 (variance of chain j)
  s.square.J = rep(0, J)
  for (j in 1:J) {
    s.square.J[j] = 1/(I-1) * sum( (beta[,j] - beta.mean.J[j])^2 )
  }
```

```
  # Within-sequence estimator
  W = 1/J * sum(s.square.J)

  # Unbiased estimate of the overall variance
  V.hat = (I-1)/I * W + 1/I * B

  # R hat
  R.hat = V.hat / W

}


J = 2  # number of chains
I = M  # number of iterations in each chain

beta = matrix(c(beta_1, beta_2), nrow = I) # column j = iterations of chain j
tau = matrix(c(tau_1, tau_2), nrow = I)    # column j = iterations of chain j

# Gelman-Rubin diagnostic of parameters beta and tau
# Necessary condition for convergence : R close to 1
R.beta = gelman.diagnostic(beta)
R.beta  # 1.006316

R.tau = gelman.diagnostic(tau)
R.tau   # 0.9992421


# Gelman diagnostic plots
gelman.plot(  list(mcmc(beta_1), mcmc(beta_2) )  )
gelman.plot(  list(mcmc(tau_1), mcmc(tau_2) )  )



### Point estimation and 95% CI given respectively by the quantiles 0.5, 0.025 and 0.975 over
      the two chains
quantile(c(beta_1, beta_2), p = c(0.025, 0.5, 0.975))
quantile(c(tau_1, tau_2), p = c(0.025, 0.5, 0.975))
mean(c(beta_1, beta_2))
mean(c(tau_1, tau_2))


#############################################################################
#       Q2 f) : Gibbs algorithm and Gelman-Rubin diagnostic using JAGS      #
#############################################################################
require(R2WinBUGS)
require(coda)
require(rjags)

rm(list=ls())

# Model creation
model2f <- function(){
  for (i  in  1:n){
    y[i] ~ dnorm(beta0, lambda[i]*tau )
    lambda[i] ~ dgamma(a,b)
  }
  beta0 ~ dnorm(0,1e-6)   # approx. uniform prior over [0,1]
  tau ~ dgamma(1e-6,1e-6) # approx. uniform prior over [0,tau]
}

model2f.file <- "Model2f.bug"
write.model(model2f, model2f.file)

# Data initialization
set.seed(1234)
nu = 3; n = 500
y = 170 + 5*rt(n, df=nu)

data <- list(y = y, n = n, a = nu/2, b = nu/2)

# Starting values of the two chains
```

```
inits <- list( list(beta0 = 140, tau = 3, lambda = rep(0.5, n)), list(beta0 = 200, tau = 7,
    lambda = rep(2, n)) )

# JAGS model
myjags.model.2f <- jags.model(file = model2f.file,
                              data = data,
                              inits = inits,
                              n.chains = length(inits),
                              n.adapt = 1000)

# Sampling parameters beta0, tau and lambdas
samples.2f <- coda.samples(model = myjags.model.2f,
                           variable.names = c("beta0", "tau"),
                           n.iter = 10000)

# chain 1
chain1 = samples.2f[1]
beta0.chain1 = as.matrix(chain1[,'beta0'])
tau.chain1 = as.matrix(chain1[,'tau'])

# chain 2
chain2 = samples.2f[2]
beta0.chain2 = as.matrix(chain2[,'beta0'])
tau.chain2 = as.matrix(chain2[,'tau'])

# Gelman diagnostic
gelman.diag( list(mcmc(beta0.chain1), mcmc(beta0.chain2) ) )
gelman.diag( list(mcmc(tau.chain1), mcmc(tau.chain2) ) )

gelman.plot(  list(mcmc(beta0.chain1), mcmc(beta0.chain2) )  )
gelman.plot(  list(mcmc(tau.chain1), mcmc(tau.chain2) )  )


# trace plot for beta samples
plot(1:10000, beta0.chain1, type="l", lwd=2, col="green", xlab="iter", ylab="beta", ylim=
    range(155,190),  main="beta gibbs samples")
lines(1:10000, beta0.chain2, type="l", col="red", lwd=2, lty=2)
legend("topright",c("first chain","second chain"), lwd=c(2,2), col=c("green","red"), box.lty
    =1, lty=1:5, cex=0.8, y.intersp=0.2)

# trace plot for tau samples
plot(1:10000, tau.chain1, type="l", lwd=2, col="green", xlab="iter", ylab="tau", ylim=range
    (-0.5,2.5), main="tau gibbs samples")
lines(1:10000, tau.chain2, type="l", col="red", lwd=2, lty=2)
legend("topright",c("first chain","second chain"), lwd=c(2,2), col=c("green","red"), box.lty
    =1, lty=1:5, cex=0.8, y.intersp=0.2)



# geweke diagnostic (not asked)
geweke.diag(mcmc(beta0.chain1))
geweke.diag(mcmc(beta0.chain2))

geweke.diag(mcmc(tau.chain1))
geweke.diag(mcmc(tau.chain2))

# geweke diagnostic plot
geweke.plot(mcmc(beta0.chain1), nbins = 20)
geweke.plot(mcmc(beta0.chain2), nbins = 20)

geweke.plot(mcmc(tau.chain1), nbins = 20)
geweke.plot(mcmc(tau.chain2), nbins = 20)



### Point estimation and 95% CI given respectively by the mean and the 2.5% and 97.5%
    quantiles
summary(samples.2f)
```

```
##############################################################################
#                  Q3 a) + b) : sampling from the posterior                  #
##############################################################################
require(R2WinBUGS)
require(coda)
require(rjags)

rm(list=ls())

# loading data (/!\ change to LOCAL directory)
setwd('C:/Users/alext/OneDrive - UCL/UCL/DATA M1/Q2/LSTAT2130 - Introduction to Bayesian
    statistics/Projet Bayesian')
brain.data = read.table('brain.txt', header = TRUE, dec =".")
x = log(brain.data[,'bodyweight'] )
y = log(brain.data[,'brainweight'] )


# Model creation
model3a <- function(){
  for (i  in  1:n){
    y[i] ~ dnorm(beta0 + beta1 * (x[i] - x.mean), lambda[i]*tau )
    lambda[i] ~ dgamma(a,b)
  }
  beta0 ~ dnorm(0,1e-6)   # approx. uniform prior over [0,1]
  beta1 ~ dnorm(0,1e-6)   # approx. uniform prior over [0,1]
  tau ~ dgamma(1e-6,1e-6) # approx. uniform prior over [0,tau]
}

model3a.file <- "Model3a.bug"
write.model(model3a, model3a.file)

# Data initialization
n = length(y)
nu = 3
data <- list(y = y, n = n, x = x, x.mean = mean(x), a = nu/2, b = nu/2)

# Starting values of the  chain
inits <- list( list(beta0 = 3, beta1 = 0, tau = 0.4, lambda = rep(0.5, n)) )

# JAGS model
myjags.model.3a <- jags.model(file = model3a.file,
                              data = data,
                              inits = inits,
                              n.chains = length(inits),
                              n.adapt = 1000)

# Sampling parameters beta0, tau and lambdas
samples.3a <- coda.samples(model = myjags.model.3a,
                           variable.names = c("beta0", "beta1", "tau"),
                           n.iter = 200000)
summary(samples.3a)

# MCMC chain
mcmc.chain = samples.3a[1]
beta0 = as.matrix(mcmc.chain[,'beta0'])
beta1 = as.matrix(mcmc.chain[,'beta1'])
tau = as.matrix(mcmc.chain[,'tau'])

# trace plot of the chain for each parameters
plot(1:50000, beta0, type="l", lwd=2, col="black", xlab="iter", ylab="beta0", ylim=range(3,6)
    ,  main="beta0 gibbs samples")
plot(1:50000, beta1, type="l", lwd=2, col="black", xlab="iter", ylab="beta1", ylim=range
    (0.2,1),  main="beta1 gibbs samples")
plot(1:50000, tau, type="l", lwd=2, col="black", xlab="iter", ylab="tau", ylim=range(-1,9),
    main="tau gibbs samples")



# geweke diagnostic
geweke.diag(mcmc(beta0))
geweke.diag(mcmc(beta1))
geweke.diag(mcmc(tau))
```

```
# geweke diagnostic plot
geweke.plot(mcmc(beta0), nbins = 20)
geweke.plot(mcmc(beta1), nbins = 20)
geweke.plot(mcmc(tau), nbins = 20)


#############################################################################
#                       Q3 c) : 95% HDP computation                         #
#############################################################################
# Points estimation of beta0, beta1 and tau respectively
mean(beta0) # 4.78236
mean(beta1) # 0.7089617
mean(tau)   # 1.915994

# 95% HPD
obj = mcmc.chain
HPDinterval(obj = obj, prob = 0.95)


#############################################################################
#                    Q3 d) : regression lines estimation                    #
#############################################################################
require(R2WinBUGS)
require(coda)
require(rjags)
require(Metrics)

rm(list=ls())

# loading data (/!\ change to LOCAL directory)
setwd('C:/Users/alext/OneDrive - UCL/UCL/DATA M1/Q2/LSTAT2130 - Introduction to Bayesian
     statistics/Projet Bayesian')
brain.data = read.table('brain.txt', header = TRUE, dec =".")
x = log(brain.data[,'bodyweight'] )
y = log(brain.data[,'brainweight'] )

# Interval x for the conditional mean desired
#int = seq(min(x),max(x),by=0.5) ; K=length(int)
int = x; K=length(int)


postmean.estimate <- function(nu) {
  # input : parameter nu value
  # output : prediction of the conditional mean for all values x in int


  # Model creation
  model3d <- function(){
    for (i  in  1:n){
      y[i] ~ dnorm(beta0 + beta1 * (x[i] - x.mean), lambda[i]*tau )
      lambda[i] ~ dgamma(a,b)
    }

    # Credible region of interest for the conditional mean
    for (k in 1:K){
      mu.k[k] <- beta0 + beta1 * (int[k] - mean(x))
    }

    beta0 ~ dnorm(0,1e-6)   # approx. uniform prior over [0,1]
    beta1 ~ dnorm(0,1e-6)   # approx. uniform prior over [0,1]
    tau ~ dgamma(1e-6,1e-6) # approx. uniform prior over [0,tau]
  }

  model3d.file <- "Model3d.bug"
  write.model(model3d, model3d.file)

  # Data initialization
  n = length(y)
  nu = nu
  data <- list(y = y, n = n, x = x, x.mean = mean(x), a = nu/2, b = nu/2, int = int, K = K)

  # Starting values of the chain
  inits <- list( list(beta0 = 3, beta1 = 0, tau = 0.4, lambda = rep(0.5, n)) )
```

```
  # JAGS model
  myjags.model.3d <- jags.model(file = model3d.file,
                                data = data,
                                inits = inits,
                                n.chains = length(inits),
                                n.adapt = 1000)

  # Sampling parameters beta0, beta1, tau and mu[k]
  samples.3d <- coda.samples(model = myjags.model.3d,
                             variable.names = c("beta0", "beta1", "tau", "mu.k"),
                             n.iter = 50000)

  # point esimations of mu for all x values of interest
  mcmc.chain = samples.3d[1]
  mu = rep(0, K)
  for (k in 1:K) {
    mu[k] = mean(as.matrix(mcmc.chain[,paste("mu.k[", k, "]", sep="")]))
  }

  return(mu)
}

### regression lines estimated with with nu = 3,5,10,15,25 in a bayesian setting
predict1 = postmean.estimate(3)
predict2 = postmean.estimate(5)
predict3 = postmean.estimate(10)
predict4 = postmean.estimate(15)
predict5 = postmean.estimate(25)

### regression line estimated in a classical frequentist setting
brain.df = log(brain.data[,3:4])
x.centered =  brain.df[,'bodyweight'] - mean(brain.df[,'bodyweight'])
brain.df[,'bodyweight'] = x.centered
lm.fit = lm(formula = brainweight~bodyweight, data = brain.df)

# prediction using the fitted model
beta0 = lm.fit$coefficients[1]
beta1 = lm.fit$coefficients[2]
predict.fr = beta0 + beta1 * (int - mean(int))

# plot of the regression lines
plot(x,y, main="linear regression lines over the brain dataset")
legend("bottomright",c("frequentist","nu=3","nu=5","nu=10","nu=15","nu=25"), lwd=c
     (2,2,2,2,2,2), col=c("black","blue","green","yellow","orange","red"), box.lty=1, cex=0.8,
      y.intersp=0.2)

lines(int, predict1, col="blue",lwd=2)
lines(int, predict2, col="green",lwd=2)
lines(int, predict3, col="yellow",lwd=2)
lines(int, predict4, col="orange",lwd=2)
lines(int, predict5, col="red",lwd=2)
lines(int, predict.fr, col="black", lwd=2)

# rmse of the regressions
rmse(y, predict.fr)
rmse(y, predict1)
rmse(y, predict2)
rmse(y, predict3)
rmse(y, predict4)
rmse(y, predict5)


# mse of the regressions
mse(y, predict.fr)
mse(y, predict1)
mse(y, predict2)
mse(y, predict3)
mse(y, predict4)
mse(y, predict5)
```

```
# mae of the regressions
mae(y, predict.fr)
mae(y, predict1)
mae(y, predict2)
mae(y, predict3)
mae(y, predict4)
mae(y, predict5)
```