Bayesian Modelling – Problem Sheet 5
Part B
(Solutions)

Problem 1

1. Let P be a transition matrix on $\mathcal{Y}$, $S = \{\mu \in \mathbb{R}^m_{\geq 0}$ such that $\sum_{i=1}^m \mu_i = 1\}$ and f be the mapping $\mu \mapsto P^T\mu$. Note that S is closed and bounded (and therefore compact) while f is continuous on S. To use Brouwer's fixed point theorem it remains to show that $f(S) = S$. Clearly, for any $\mu \in S$ all the components of $f(\mu)$ are non-negative because $\min_{i,j\in\mathcal{Y}} p_{ij} \geq 0$ and $\min_{i\in\mathcal{Y}} \mu_i \geq 0$. Let $f_i(\mu)$ be the i-th component of $f(\mu)$ and $\mathbf{1}_m = (1, \ldots, 1)$. Then,

$$\sum_{i=1}^{m} f_i(\mu) = \mathbf{1}_m^T(P^T\mu) = (P\mathbf{1}_m)^T\mu = \mathbf{1}_m^T\mu = 1$$

so that $f(S) = S$. Therefore, by Brouwer's fixed point theorem, there exists a $\mu \in S$ such that

$$\mu = f(\mu) = P^T\mu \Leftrightarrow \mu^T = \mu^T P.$$

2. Let $\mu = (\mu_1, \mu_2, 1 - \mu_1 - \mu_2)$. Then,

$$P^T\mu = \mu \Leftrightarrow (P^T - \mathbf{1}_3)\mu = 0$$

with

$$P^T - \mathbf{1}_3 = \begin{pmatrix} 0 & 1/3 & 0 \\ 0 & -2/3 & 0 \\ 0 & 1/3 & 0 \end{pmatrix}.$$

Therefore, $(\mu_1, 0, 1-\mu_1)$ is an invariant distribution of P for any $\mu_1 \in [0, 1]$ so that P has infinity many invariant distributions.

Using the result in part 1 and in Theorem 7.2, if P is irreducible and aperiodic then P must have a unique invariant distribution. Since this is not the case we deduce that P is not an irreducible and aperiodic transition matrix.

3. It is easily checked that the system of equations $P^T\mu = \mu$ has a unique solution at $\mu = (1/3, 1/3, 1/3)$. However, $\mu_1 p_{12} = 1/3$ while $\mu_2 P_{21} = 0$ so that the detailed balance condition is not satisfied.

4. a) We have $P^t = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ if t is even and $P^t = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ if t is odd. Hence P is irreducible but periodic.

   b) It is easily checked that the system of equations $P^T\mu = \mu$ has a unique solution at $\mu = (1/2, 1/2)$.

c) Let $i = j = 1$ for instance. Then, from part 4.a),

$$1 = \limsup_{t\to+\infty} p_{11}^{(t)} > \liminf_{t\to+\infty} p_{11}^{(t)} = 0$$

and therefore $\lim_{t\to+\infty} p_{11}^{(t)}$ does not exist. A similar argument shows that, for any $(i,j) \in \mathcal{Y}$, we have

$$\limsup_{t\to+\infty} p_{ij}^{(t)} \neq \liminf_{t\to+\infty} p_{ij}^{(t)}.$$

Problem 2

1. Let $(\tilde{y}, y) \in \mathcal{Y}^2$. Then,

$$\begin{aligned}
\alpha(y,\tilde{y}) = \min\left\{1, \frac{\mu(\tilde{y})q_i(y|\tilde{y})}{\mu(y)q_i(\tilde{y}|y)}\right\} &= \min\left\{1, \mathbb{1}_{\{y^{(-i)}\}}\big(\tilde{y}^{(-i)}\big)\frac{\mu(\tilde{y})\mu^{(i)}(y^{(i)}|\,\tilde{y}^{(-i)})}{\mu(y)\mu^{(i)}(\tilde{y}^{(i)}|\,y^{(-i)})}\right\} \\
&= \min\left\{1, \mathbb{1}_{\{y^{(-i)}\}}\big(\tilde{y}^{(-i)}\big)\frac{\mu(\tilde{y})\mu^{(i)}(y^{(i)}|\,y^{(-i)})}{\mu(y)\mu^{(i)}(\tilde{y}^{(i)}|\,y^{(-i)})}\right\}
\end{aligned}$$

where (with obvious notation)

$$\mu^{(i)}(y^{(i)}|\,y^{(-i)}) = \frac{\mu(y)}{\mu(y^{(-i)})}, \quad \mu^{(i)}(\tilde{y}^{(i)}|\,y^{(-i)}) = \frac{\mu(\tilde{y}^{(i)},\, y^{(-i)})}{\mu(y^{(-i)})}$$

so that

$$\frac{\mu^{(i)}(y^{(i)}|\,y^{(-i)})}{\mu^{(i)}(\tilde{y}^{(i)}|\,y^{(-i)})} = \frac{\mu(y)}{\mu(\tilde{y}^{(i)},\, y^{(-i)})}.$$

Therefore

$$\begin{aligned}
\alpha(y,\tilde{y}) &= \min\left\{1, \mathbb{1}_{\{y^{(-i)}\}}\big(\tilde{y}^{(-i)}\big)\frac{\mu(\tilde{y})}{\mu(y)}\frac{\mu(y)}{\mu(\tilde{y}^{(i)},\, y^{(-i)})}\right\} \\
&= \min\left\{1, \mathbb{1}_{\{y^{(-i)}\}}\big(\tilde{y}^{(-i)}\big)\frac{\mu(\tilde{y})}{\mu(y)}\frac{\mu(y)}{\mu(\tilde{y})}\right\} \\
&= \min\left\{1, \mathbb{1}_{\{y^{(-i)}\}}\big(\tilde{y}^{(-i)}\big)\right\} \\
&= \mathbb{1}_{\{y^{(-i)}\}}\big(\tilde{y}^{(-i)}\big)
\end{aligned}$$

and thus

$$\begin{aligned}
\mathbb{P}\big(Y_t = \tilde{Y}_t\big) = \mathbb{E}\big[\mathbb{P}\big(Y_t = \tilde{Y}_t|\tilde{Y}_t,\, Y_{t-1}\big)\big] &= \mathbb{E}\big[\alpha(Y_{t-1}, \tilde{Y}_t)\big] \\
&= \mathbb{E}\Big[\mathbb{E}\big[\alpha(Y_{t-1}, \tilde{Y}_t)|\, Y_{t-1}\big]\Big] \\
&= \mathbb{E}\big[\mathbb{P}\big(Y_{t-1}^{(-i)} = \tilde{Y}_t^{(-i)}|Y_{t-1}\big)\big] \\
&= \mathbb{E}[1] \\
&= 1.
\end{aligned}$$

2. Let $\tilde{y} \in \mathcal{Y}$. Then,

$$\begin{aligned}
\int_{\mathcal{Y}} k(\tilde{y}|y)\mu(y)\mathrm{d}y &= \int_{\mathcal{Y}} \Big(\int_{\mathcal{Y}} k_2(\tilde{y}|y')k_1(y'|y)\mathrm{d}y' \Big)\mu(y)\mathrm{d}y \\
&= \int_{\mathcal{Y}} k_2(\tilde{y}|y') \Big(\int_{\mathcal{Y}} k_1(y'|y)\mu(y)\mathrm{d}y \Big)\mathrm{d}y' \\
&= \int_{\mathcal{Y}} k_2(\tilde{y}|y')\mu(y')\mathrm{d}y' \\
&= \mu(\tilde{y})
\end{aligned}$$

where the first equality uses the definition of $k(\tilde{y}|y)$, the second one uses Fubini's theorem, the third one the fact that $k_1(\tilde{y}|y)$ has μ as invariant distribution and the last one the fact that $k_2(\tilde{y}|y)$ has μ as invariant distribution.

3. From part 1, for any $i \in \{1, \dots, d\}$ the transition kernel $q_i(\tilde{y}|y)$ has μ has invariant distribution. Indeed, when $\mathbb{P}(Y_t = \tilde{Y}_t) = 1$ the proposal distribution and the Metropolis-Hastings kernel P^{MH} coincide, and by construction this latter has μ as invariant distribution.

   Next, let $\bar{q}_1(\tilde{y}|y) = q_1(\tilde{y}|y)$ and

$$\bar{q}_i(\tilde{y}|y) = \int_{\mathcal{Y}} q_i(\tilde{y}|y')\bar{q}_{i-1}(y'|y)\mathrm{d}y', \quad i = 2, \dots d.$$

   Then, using the result in part 2, $\bar{q}_d(\tilde{y}|y)$ has μ as invariant distribution.

   To show that $\bar{q}_d(\tilde{y}|y)$ is the Gibbs kernel remark that we can generate a random draw from $\bar{q}_d(\tilde{y}|y)\mathrm{d}\tilde{y}$ using the following algorithm

   Set $Z_0 = y$

   For $i = 1, ..., d$

   $Z_i \sim q_i(z_i|Z_{i-1})\mathrm{d}z_i$

   End for

   return $\tilde{Y} = Z_d$

   or equivalently, since q_i updates only component $Z_i^{(i)}$ of Z_i (and using obvious notation)

   Set $\tilde{Y}_0 = y$

   For $i = 1, ..., d$

   $Z^{(i)} \sim \mu^{(i)}(z^{(i)}|\tilde{Y}_{i-1}^{(-i)})\mathrm{d}z^{(i)}$

   $\tilde{Y}_i = (\tilde{Y}_i^{(1:(i-1))}, z^{(i)}, \tilde{Y}_i^{(i+1:d)})$

   End for

   return $\tilde{Y} = \tilde{Y}_d$

   which is the Gibbs sampler (Algorithm (A3) in the lecture notes).

Problem 3

We fix the seed (so that you and I get the same numbers) and load two useful packages:

```
set.seed(90585)
# Load package to sample from multivariate normal distributions
library(MASS)
# Load package to analyse output of MCMC algorithms
library(coda)
```

We load the data:

```
# Load SP500 data
p<-read.table('DataSheet5/ARCH/SP500.txt')
# Compute log-returns
x<-diff(log(p[,5]))
```

We now specify the prior distribution for ARCH(q) models:

```
dprior<-function(theta, param){
   return(dgamma(theta[1],shape= param[1], rate= param[2], log=TRUE))
}
```

We define the likelihood function for ARCH(q) models:

```
loglik_ARCH<-function(x,theta){
   q<-length(theta)-1
   t<-length(x)
   sigma2_vec<-rep(0,t)
   work<-rep(0,q)
   for (s in 1:t){
       sigma2_vec[s]<-theta[1]+sum(theta[2:(q+1)]*work)
       work<-c(x[s]^2, work)[1:q]
   }
   return(sum(dnorm(x,0,sd=sqrt(sigma2_vec), log=TRUE)))
}
```

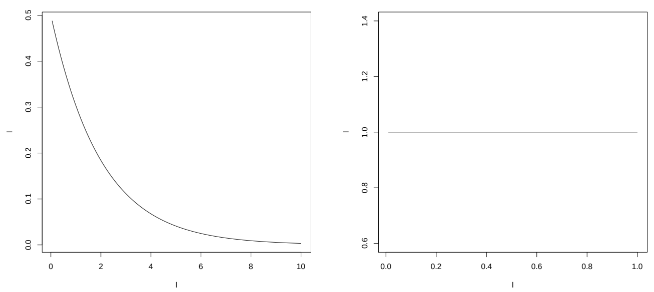
Lastly, we write a generic function that allows to run the M-H algorithm for ARCH(q) models:

```
MH_ARCH<-function(theta0, n_iter, Sigma, x, prior, loglik){
  q<-length(theta0)-1
  theta<-matrix(0,n_iter, length(theta0))
```

```
  acceptance_rate<-0
  theta[1,]<-theta0
  l0<-loglik(x,theta0)+prior$density(theta0, prior$param)
  for(n in 2:n_iter){
      theta_prop<-mvrnorm(1,theta[n-1,], Sigma)
      theta[n,]<-theta[n-1,]
      if(min(theta_prop)>=0 && sum(theta_prop[2:(q+1)])<1){
           l1<-loglik(x,theta_prop)+prior$density(theta_prop, prior$param)
           if(log(runif(1))<=l1-l0){
              theta[n,]<-theta_prop
              l0<-l1
              acceptance_rate<-acceptance_rate+1
           }
     }
 }
 return(list(CHAIN=theta, RT=acceptance_rate/n_iter))
}
```

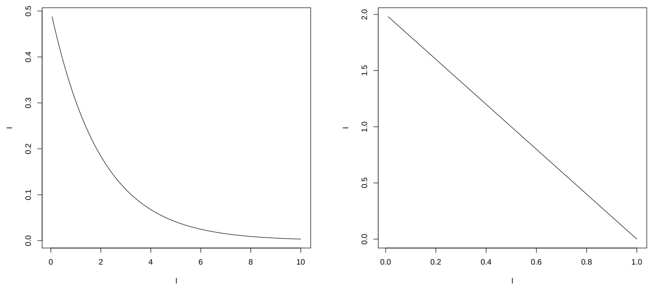
1. For $q = 1$:

```
plot(seq(0.05,10,0.05), dgamma(seq(0.05,10,0.05),1,rate=1/2),
type='l',xlab='l',ylab='l')
plot(seq(0.01,1,0.01), dbeta(seq(0.01,1,0.01),1,1),
type='l',xlab='l',ylab='l')
```



For $q = 2$:

```
plot(seq(0.05,10,0.05), dgamma(seq(0.05,10,0.05),1,rate=1/2),
type='l',xlab='l',ylab='l')
plot(seq(0.01,1,0.01), dbeta(seq(0.01,1,0.01),1,2),
type='l', xlab='l',ylab='l')
```



We therefore observe that the prior distribution for α_1 is non-informative in Laplace's sense for $q = 1$ but not for $q = 2$.

2. a) We Choose the ARCH(1) model:

```
q<-1
```

We specify the prior parameters and starting value:

```
# Prior parameters
prior_ARCH<-list(density=dprior, param=c(1,1/2))
# Starting values
theta0<-c(0.01,rep(0.1,q))
```

We take for instance take for Σ_1 the matrix given in the file Sigma_1.txt:

```
Sigma_1<-as.matrix(read.table('DataSheet5/ARCH/Sigma_1.txt'))
```

We now run the M-H algorithm for 100 000 iterations:

```
n_iter<-100000
run<-MH_ARCH(theta0, n_iter, Sigma_1, x, prior_ARCH, loglik_ARCH)
```

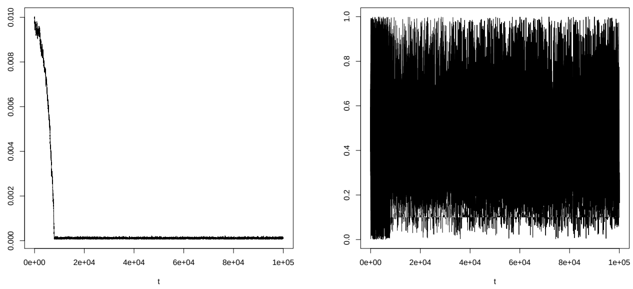
We first compute the acceptance rate

```
run$RT

## [1] 0.27502
```

Then we look at the trace plots to choose the length B of the burn-in period

```
plot(1:n_iter, run$CHAIN[,1], type='l', xlab='t', ylab=' ')
plot(1:n_iter, run$CHAIN[,2], type='l', xlab='t', ylab=' ')
```

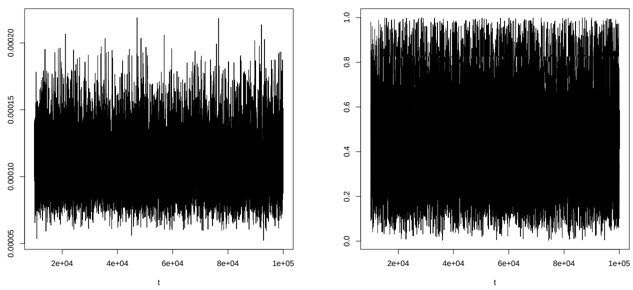


It is clear from the left plot that some time is needed for the algorithm to enter in its stationary regime. Visually it seems that a burn-in period of $B = 10\,000$ iterations should be enough.

```
B<-10000
```

We check again the trace plots to make sure that this is the case:

```
plot(B:n_iter, run$CHAIN[B:n_iter,1], type='l', xlab='t', ylab=' ')
plot(B:n_iter, run$CHAIN[B:n_iter,2], type='l', xlab='t', ylab=' ')
```

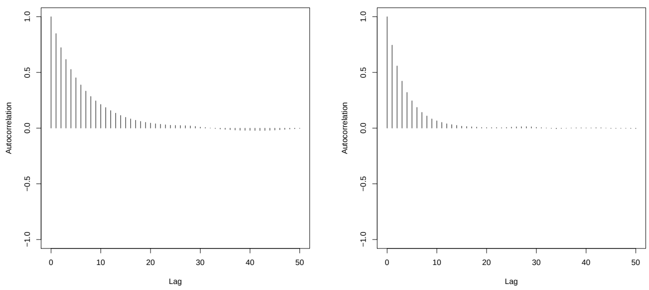


The value of B we choose looks good. To proceed further we convert the simulated chain into an mcmc object:

```
# Convert the results into an mcmc object
sample_1<-as.mcmc(run$CHAIN[B:n_iter,])
```

We can now inspect the ACFs:

```
autocorr.plot(sample_1[,1], lag.max=50)
autocorr.plot(sample_1[,2], lag.max=50)
```



The autocorrelations decrease quickly and therefore there are no reasons to believe that $T := n_{\text{iter}} - B = 90\,000$, the length of for the simulated trajectory (after the burn-in period), is not enough.

As a last check we can run the command:

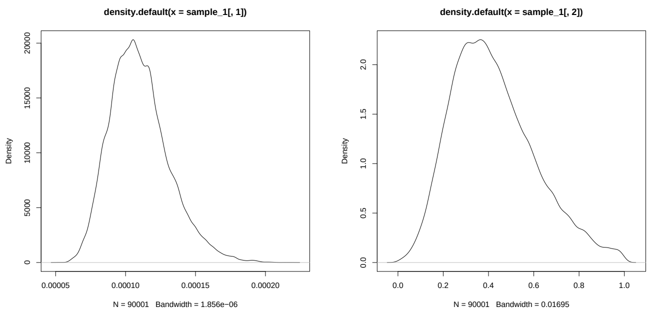
```
summary(sample_1)$statistics[,c(1,4)]
```

```
##                Mean Time-series SE
## [1,] 0.0001100254   2.503007e-07
## [2,] 0.4216700975   1.633615e-03
```

We observe that the estimated variances (second column) for our estimated values of the two the posterior means (first column) are small.

b) We plot the estimated marginal distributions:

```
plot(density(sample_1[,1]))
plot(density(sample_1[,2]))
```



We provide estimated values and 99% confidence intervals for the two posterior means:

```
estimates<-summary(sample_1)$statistics
# Estimated value of the two posterior mean
estimates[,1]
```

```
## [1] 0.0001100254 0.4216700975
```

```
# Lower bounds for the 99% confidence interval for these estimates
```

```
estimates[,1]-qnorm(0.995)*estimates[,4]

## [1] 0.0001093806 0.4174621844

# Upper bounds for the 99% confidence interval for these estimates
estimates[,1]+qnorm(0.995)*estimates[,4]

## [1] 0.0001106701 0.4258780105
```

3. a) We now consider the ARCH(2) model:

```
q<-2
```

We specify the prior parameters and starting value:

```
# Prior parameters
prior_ARCH<-list(density=dprior, param=c(1,1/2))
# Starting values
theta0<-c(0.01,rep(0.1,q))
```

We load the matrix Σ_2 used in the proposal distribution:

```
Sigma_2<-as.matrix(read.table('DataSheet5/ARCH/Sigma_2.txt'))
```

We now run the M-H algorithm for 100 000 iterations:

```
n_iter<-100000
run<-MH_ARCH(theta0, n_iter, Sigma_2, x, prior_ARCH, loglik_ARCH)
```

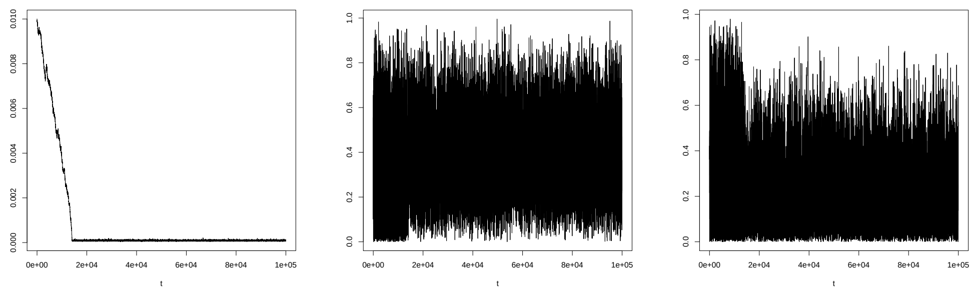
We first compute the acceptance rate:

```
run$RT

## [1] 0.25964
```

Then we look at the trace plots to choose the length B of the burn-in period:

```
plot(1:n_iter, run$CHAIN[,1], type='l', xlab='t', ylab=' ')
plot(1:n_iter, run$CHAIN[,2], type='l', xlab='t', ylab=' ')
plot(1:n_iter, run$CHAIN[,3], type='l', xlab='t', ylab=' ')
```

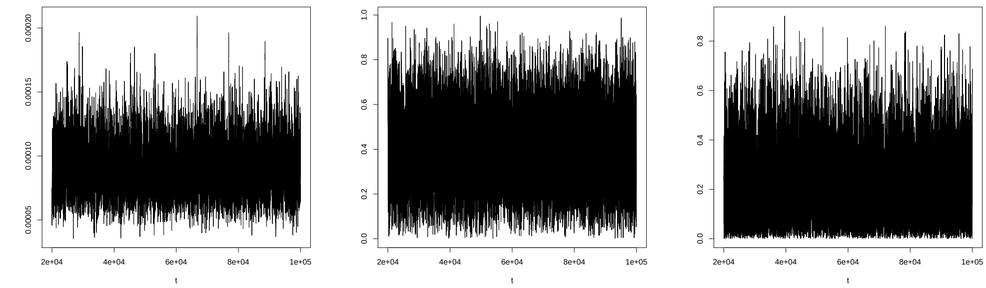


It is clear from the left plot that some time is needed for the algorithm to enter in its stationary regime. Visually it seems that a burn-in period of $B = 20\,000$ iterations should be enough.

```
B<-20000
```

We check again the trace plots to make sure that this is the case:

```
plot(B:n_iter, run$CHAIN[B:n_iter,1], type='l', xlab='t', ylab=' ')
plot(B:n_iter, run$CHAIN[B:n_iter,2], type='l', xlab='t', ylab=' ')
plot(B:n_iter, run$CHAIN[B:n_iter,3], type='l', xlab='t', ylab=' ')
```

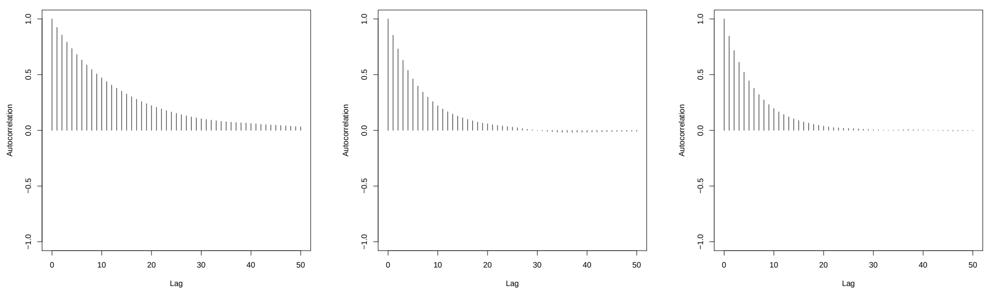


The value of B we choose looks good. To proceed further we convert the simulated chain into an mcmc object:

```
# Convert the results into an mcmc object
sample_2<-as.mcmc(run$CHAIN[B:n_iter,])
```

We can now inspect the ACFs:

```
autocorr.plot(sample_2[,1], lag.max=50)
autocorr.plot(sample_2[,2], lag.max=50)
autocorr.plot(sample_2[,3], lag.max=50)
```



The autocorrelations decrease quickly and therefore there are no reason believe that $T := n_{\text{iter}} - B = 80\,000$, the length of for the simulated trajectory (after the burn-in period), is not enough.

As a last check we can run the command:

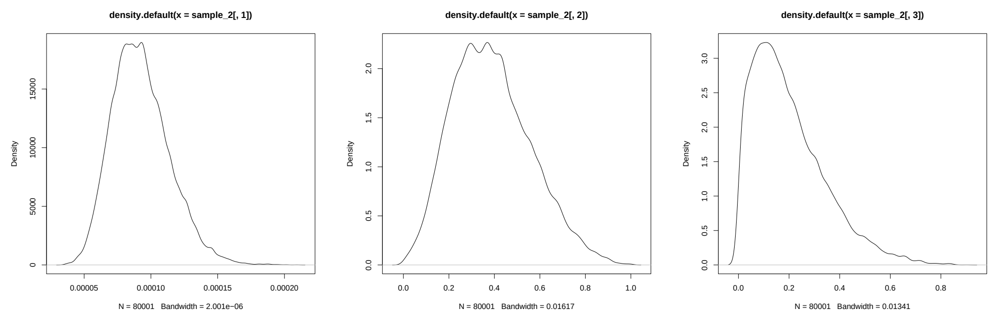
```
summary(sample_2)$statistics[,c(1,4)]

##                Mean Time-series SE
## [1,] 9.253843e-05   3.883424e-07
## [2,] 3.881804e-01   2.206563e-03
## [3,] 1.994341e-01   1.793777e-03
```

We observe that the estimated variances (second column) for our estimated values of the three the posterior means (first column) are small.

b) We plot the three estimated marginal distributions:

```
plot(density(sample_2[,1]))
plot(density(sample_2[,2]))
plot(density(sample_2[,3]))
```



We provide estimated values and 99% confidence intervals for the three posterior means:

```
estimates<-summary(sample_2)$statistics
# Estimated value of the two posterior mean
estimates[,1]

## [1] 9.253843e-05 3.881804e-01 1.994341e-01

# Lower bounds for the 99% confidence interval for these estimates
estimates[,1]-qnorm(0.995)*estimates[,4]

## [1] 9.153813e-05 3.824967e-01 1.948136e-01

# Upper bounds for the 99% confidence interval for these estimates
estimates[,1]+qnorm(0.995)*estimates[,4]

## [1] 9.353874e-05 3.938641e-01 2.040545e-01
```

4. a) The following function returns and estimate a_T of $a := \pi_2(\{\theta\colon \alpha_2 \leq 0.1\}|x)$ as well as an estimate of the variance of a_T:

```
test<-function(sample){
  g_values<-0+(sample[,3]<=0.1)
```

```
    est_val<-mean(g_values)
    est_var<-spectrum0(g_values)$spec/length(g_values)
    return(list(VAL=est_val, VAR=est_var))
}
```

The estimated value a_T of a is:

```
test_res<-test(sample_2)
a_T<-test_res$VAL
print(a_T)

## [1] 0.2848339
```

which is smaller than the acceptance level of $1/2$ (see Chapter 4).

To check that the Monte Carlo error (i.e. the approximation error) does not influence the outcome of the test we compute a 99% confidence interval for a:

```
# 99% confidence interval for aT
c(a_T-qnorm(0.995)*sqrt(test_res$VAR),
  a_T+qnorm(0.995)*sqrt(test_res$VAR))

## [1] 0.2685812 0.3010867
```

The upper bound of the confidence interval is smaller than the acceptance level and thus we reject H_0.

b) By definition, and recalling that $a = \pi_2(\{\theta_2 : \alpha_2 \leq 0.1\}|x)$,

$$B_{01}^{\pi_2}(x) = \frac{a}{1-a}\frac{1-c}{c}, \quad c := \pi_2\big(\{\theta_2 : \alpha_2 \leq 0.1\}\big).$$

An estimate b_T of $B_{01}^{\pi_2}(x)$ is therefore given by

$$b_T = \frac{a_T}{1-a_T}\frac{1-c}{c}$$

where, since the marginal distribution of α_2 under $\pi_2(\theta_2)$ is the Beta$(1,2)$ distribution, the value of c is

```
c<-pbeta(0.1,1,2)
print(c)

## [1] 0.19
```

Hence, our estimated value b_T of the Bayes factor is:

```
b_T<-(a_T/c)*((1-c)/(1-a_T))
print(b_T)

## [1] 1.697916
```

c) The Bayes factor is larger than one, meaning that the information brought by the observations is in favour of H_0. Despite of this, the posterior probability of H_0 is only about 0.28 and therefore the outcome of the test is driven by the prior probability that we assign to H_0 (which is about 0.19). This result means that the evidence in favour of H_0 brought by the observations is not sufficient to compensate our prior beliefs regarding H_0.

5. Since in part 4.a) we have rejected the hypothesis that $\alpha_2 \leq 0.1$ we logically choose the ARCH(2) model.

Problem 4

We fix the seed (so that you and I get the same numbers):

```
set.seed(90585)
```

1. We load the training set:

```
train_set<-read.table('DataSheet5/spam/spambase_train.txt')
x_train<-as.matrix(train_set[,ncol(train_set)])
Z_train<-as.matrix(train_set[,1:(ncol(train_set)-1)])
```

We keep the variables of interest:

```
keep<-c(5,6,7,8,9,16,17,18,19,20,21,23,24,45,57)
Z_train<-Z_train[,keep]
```

We add an intercept:

```
Z_train<-cbind(rep(1,nrow(Z_train)),Z_train)
```

We load JAGS and the module 'glm'

```
library(rjags)
## Linked to JAGS 4.3.0
## Loaded modules:  basemod,bugs
load.module('glm')
## module glm loaded
```

We write the model in a filed named Spam.bug:

```
cat('
 model{
   for(i in 1:n){
       x[i] ~ dbern(p[i])
       probit(p[i]) <- sum(theta*Z[i,])
   }
   theta ~ dmnorm(mu0 ,Omega0)
   Omega0<-inverse(Sigma0)
 }', file='Spam.bug')
```

We specify the deterministic elements of the model:

```
spam_data<-list(n=nrow(x_train), x=c(x_train), Z=Z_train,
 mu0=rep(0,ncol(Z_train)),Sigma0=diag(100,ncol(Z_train)))
```

We construct the JAGS model:

```
spam_mu<-jags.model('Spam.bug', data=spam_data)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 3680
##    Unobserved stochastic nodes: 1
##    Total graph size: 75505
##
## Initializing model
```

We check that JAGS is going to use the right sampler:

```
list.samplers(spam_mu)

## $`glm::Holmes-Held`
## [1] "theta"
```

We run the Markov chain for a burn-in period of $B = 10\,000$:

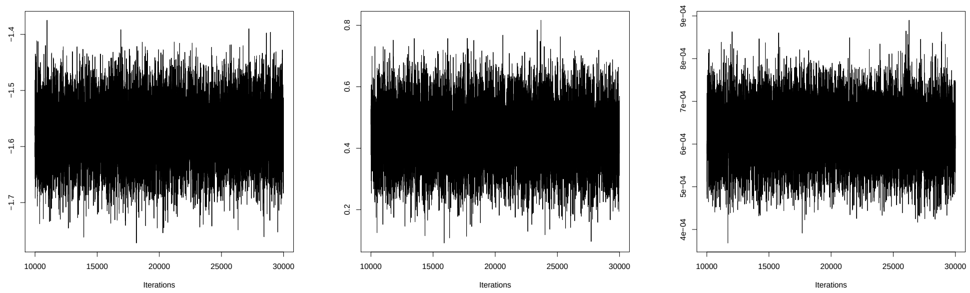
```
update(spam_mu, n.iter=10000)
```

We consider a trajectory of length $T = 20\,000$:

```
sample<-coda.samples(spam_mu,c('theta'), n.iter=20000)
```

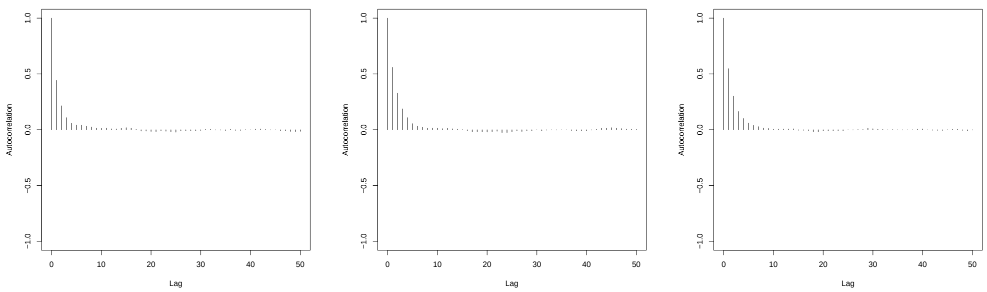
We look at the trace plots to check that the chain has converged. Here is what we get for coordinates 1, 8 and 16:

```
traceplot(sample[,1])
traceplot(sample[,8])
traceplot(sample[,16])
```



We look at the ACFs to check that the chain has converged. Here is what we get for coordinates 1, 8 and 16:

```
autocorr.plot(sample[,1],50)
autocorr.plot(sample[,8],50)
autocorr.plot(sample[,16],50)
```



As a last check we run the command:

```
summary(sample)$statistics[,c(1,4)]

##                    Mean  Time-series SE
## theta[1]   -1.573453446    6.132393e-04
## theta[2]    0.348259744    3.700781e-04
## theta[3]    0.530139966    8.219369e-04
```

```
## theta[4]    1.699924989   2.314675e-03
## theta[5]    0.541631573   8.695556e-04
## theta[6]    0.414808195   9.444242e-04
## theta[7]    0.267246415   2.317680e-04
## theta[8]    0.439257916   1.241664e-03
## theta[9]    0.249725334   5.504901e-04
## theta[10]   0.126252959   1.547249e-04
## theta[11]   0.699605776   4.170893e-03
## theta[12]   0.215080150   1.977338e-04
## theta[13]   2.098781153   4.946344e-03
## theta[14]   0.598444250   1.068156e-03
## theta[15]  -0.288019623   1.039041e-03
## theta[16]   0.000625138   8.323110e-07
```

We observe that the estimated variances (second column) for our estimated values of the the posterior means (first column) are small.

2. a) Let $\pi(x', \theta|x_{\text{train}})$ be the posterior distribution of (X', θ) given $X_{\text{train}} = x_{\text{train}}$. Then, since X' is conditionally independent of X_{train} given θ, we have

$$\pi(x'|x_{\text{train}}) = \int_\Theta \pi(x', \theta|x_{\text{train}})\mathrm{d}\theta = \int_\Theta \tilde{f}_z(x'|\theta)\pi(\theta|x_{\text{train}})\mathrm{d}\theta.$$

   b) Here is our classifier (where Z is a matrix with d columns):

```
classifier<-function(Z,sample){
   proba<-apply(pnorm(as.matrix(sample)%*%t(Z)),2,mean)
   return(0+(proba>= 0.5))
}
```

3. We load the test set:

```
test_set<-read.table('DataSheet5/spam/spambase_test.txt')
x_test<-as.matrix(test_set[,ncol(test_set)])
Z_test<-as.matrix(test_set[,1:(ncol(test_set)-1)])
```

We keep the same variables as above:

```
Z_test<-Z_test[,keep]
```

We add an intercept:

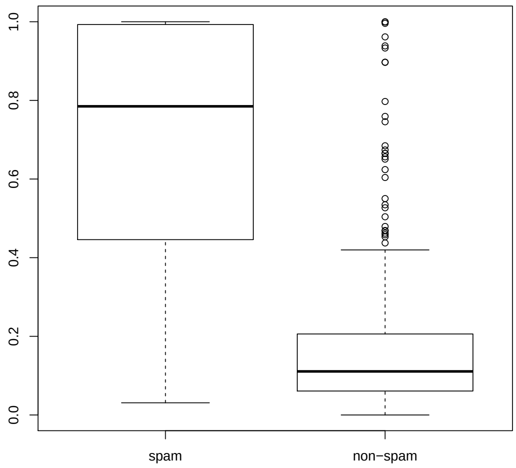
```
Z_test<-cbind(rep(1,nrow(Z_test)),Z_test)
```

a) We compute the estimated value of $\pi(x_i|x_{\text{train}})$ for all $i \in I_{\text{test}}$:

```
proba_test<-apply(pnorm(as.matrix(sample)%*%t(Z_test)),2,mean)
```

We plot $\pi(x_i|x_{\text{train}})$ for $i \in I_{1,\text{test}}$ (left plot) and $\pi(x_i|x_{\text{train}})$ for $i \in I_{0,\text{test}}$ (right plot):

```
boxplot(proba_test[x_test==1], proba_test[x_test==0],
        names = c('spam','non-spam'))
```



We observe that the estimated probability that an e-email is a spam tends to be higher when the email is indeed a spam. Here are some summary statistics that support this point:

```
summary(proba_test[x_test==1])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03095 0.44645 0.78488 0.68633 0.99286 1.00000

summary(proba_test[x_test==0])

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000001 0.0609087 0.1107530 0.1642678 0.2060164 1.0000000
```

From these results we deduce that our model is able (to some extend) to discriminate between spam and non-spam emails.

b) Here are the different classification errors:

```r
# Predictions
pred_test<-classifier(Z_test,sample)
# Total classification error
mean(abs(pred_test-x_test))
```

```
## [1] 0.1433225
```

```r
# Classification error for spam emails
mean(abs(pred_test-x_test)[x_test==1])
```

```
## [1] 0.3027778
```

```r
# Classification error for non-spam emails
mean(abs(pred_test-x_test)[x_test==0])
```

```
## [1] 0.04099822
```

In the training set there are approximatively 40% of spam emails. Hence, a naive approach would be to randomly classify a given email as a spam with probability 0.4. For this naive classifier the expected classification error for spam emails would be 0.6 and the expected classification error for non-spam emails would be 0.4. Therefore, our classifier does much better than this random guess approach.