STA623 - Bayesian Data Analysis - Practical 5 (Solutions)

Marc Henrion

9 September 2022

Practical 5

Notation

- X, Y, Z - random variables
- x, y, z - measured / observed values
- $\bar{X}$, $\bar{Y}$, $\bar{Z}$ - sample mean estimators for X, Y, Z
- $\bar{x}$, $\bar{y}$, $\bar{z}$ - sample mean estimates of X, Y, Z
- $\hat{T}$, $\hat{t}$ - given a statistic T, estimator and estimate of T
- $P(A)$ - probability of an event A occuring
- $f_X(.)$, $f_Y(.)$, $f_Z(.)$ - probability mass / density functions of X, Y, Z; sometimes $p_X(.)$ etc. rather than $f_X(.)$
- p(.) - used as a shorthand notation for pmfs / pdfs if the use of this is unambiguous (i.e. it is clear which is the random variable)
- $X \sim F$ - X distributed according to distribution function F
- $E[X]$, $E[Y]$, $E[Z]$, $E[T]$ - the expectation of X, Y, Z, T respectively

Exercise 1

Fit the model from Practical 3, Exercise 3 using JAGS and the `rjags` package. Use this as the data from the sampling model:

$$y = (1, 3, 2, 3, 0, 2, 6, 4, 4, 1, 1, 3, 2, 3, 1, 1, 3, 0)$$

Inspect the trace plot and plot the posterior distribution.

Compute the posterior mean and the quantile-based 95% Bayesian confidence interval.

Exercise 1 (Solution)

Write the following JAGS model into a file called `jagsP5ex1.jags`:

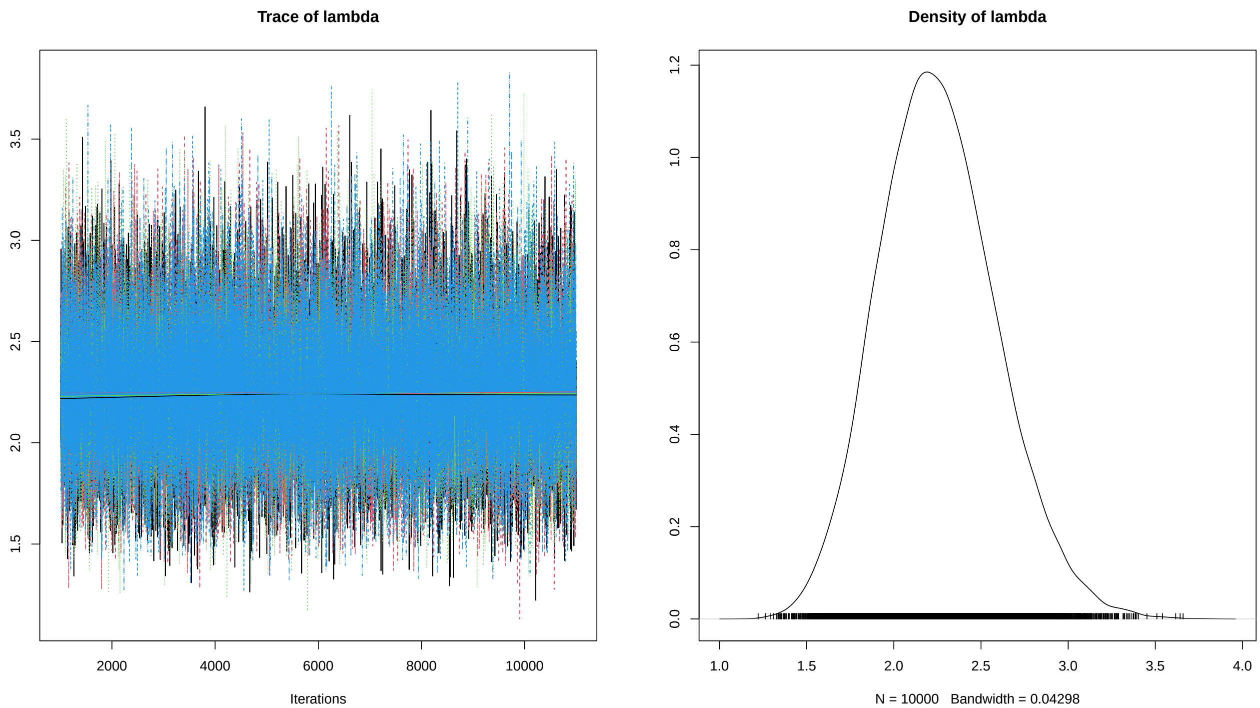
```
model{
  # sampling model
  for(i in 1:N){
    y[i]~dpois(lambda)
```

```
  }

  # prior
  lambda~dgamma(5,2)
}
```

This specifies the model. Now we need to fit this model using MCMC.

For this we use `R` and the `rjags` library.

```
library(rjags)
## Loading required package: coda
## Linked to JAGS 4.3.0
## Loaded modules: basemod,bugs

set.seed(123)

y<-c(1,3,2,3,0,2,6,4,4,1,1,3,2,3,1,1,3,0)
dat<-list(N=length(y),y=y) # 18 observations, y_i sum to 40

# set-up the model
jagsMod<-jags.model("jagsP5ex1.jags",data=dat,n.chains=4,n.adapt=1000)
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 18
##    Unobserved stochastic nodes: 1
##    Total graph size: 22
##
## Initializing model

# run more MCMC iterations
update(jagsMod,1000)

# pull out the chains for the parameters of interest
parsPosterior<-coda.samples(model=jagsMod,variable.names=c("lambda"),n.iter=1e4)

# check trace plot and empirical posterior distribution
plot(parsPosterior)
```




```r
# posterior mean estimate
summary(parsPosterior)$statistics["Mean"]
##     Mean
## 2.255283

# posterior quantile based 95% credible interval
summary(parsPosterior)$quantiles[c("2.5%","97.5%")]
##     2.5%    97.5%
## 1.639327 2.962287
```

Exercise 2

Generate the following data

```
N<-100
x<-rnorm(N)
z<-2-4*x
p<-1/(1+exp(-z))
y<-rbinom(n=N,size=1,prob=p)

# I reformat the data and write it to a file on the hard drive, just so
# I can show you how to read data in and reformat for JAGS
df<-data.frame(
  x=x,
  y=y
)

write.csv(df,row.names=FALSE,quote=FALSE,file="Pract5Ex2Data.csv")
```

Use `R` and JAGS to fit a Bayesian logistic regression model to this data:

$$g(E[Y|X]) = \beta_0 + \beta_1 X$$

where $g(\pi) = \log(\pi/(1-\pi))$

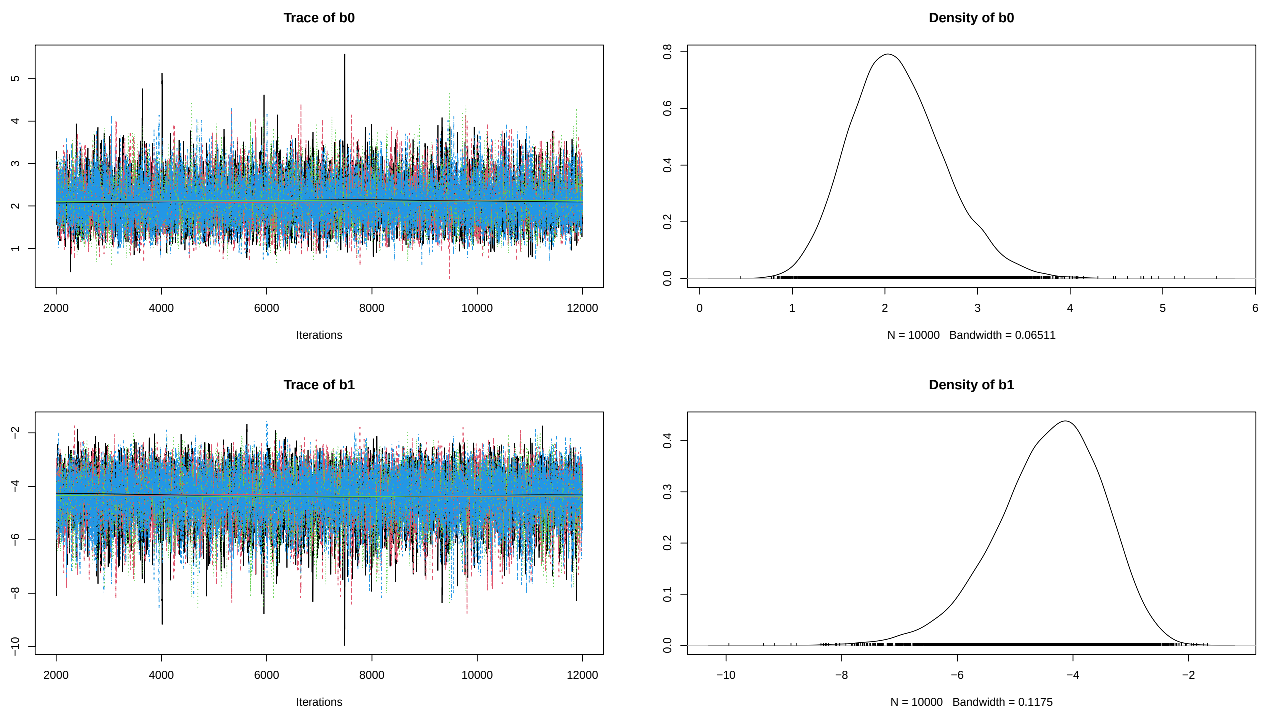
Exercise 2 (solution)

JAGS model file (save this as `jagsP5ex2.jags`):

```
model{
  # logistic regression model
  for(i in 1:N){
    y[i]~dbern(p[i])
    p[i]<-1/(1+exp(-z[i]))
    z[i]<-b0+b1*x[i]
  }

  # priors
  b0~dnorm(0,0.01) # try different priors!
  b1~dnorm(0,0.01) # try different priors!
}
```

```
library(rjags)

set.seed(123)


# read data
df<-read.csv("Pract5Ex2Data.csv")

# reformat data for JAGS
dat<-list(N=nrow(df),x=df$x,y=df$y)

# set-up model, run MCMC, pull-out chains for parameters of interest
jagsMod<-jags.model("jagsP5ex2.jags",data=dat,n.chains=4,n.adapt=1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 100
##    Unobserved stochastic nodes: 2
##    Total graph size: 806
##
## Initializing model
update(jagsMod,1000)
parsPosterior<-coda.samples(model=jagsMod,variable.names=c("b0","b1"),n.iter=1e4)

# check trace plot and empirical posterior distribution
plot(parsPosterior)
```

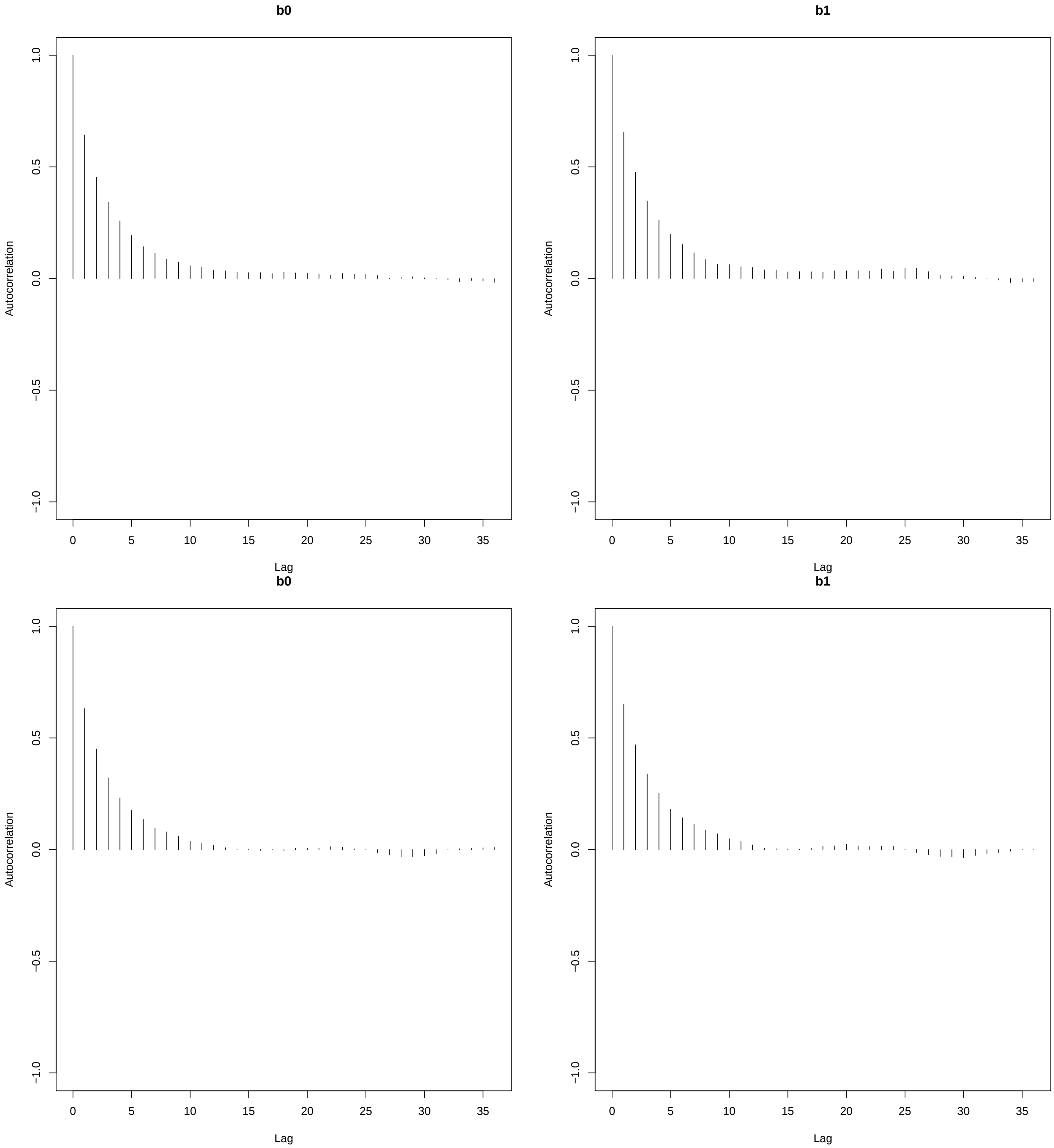


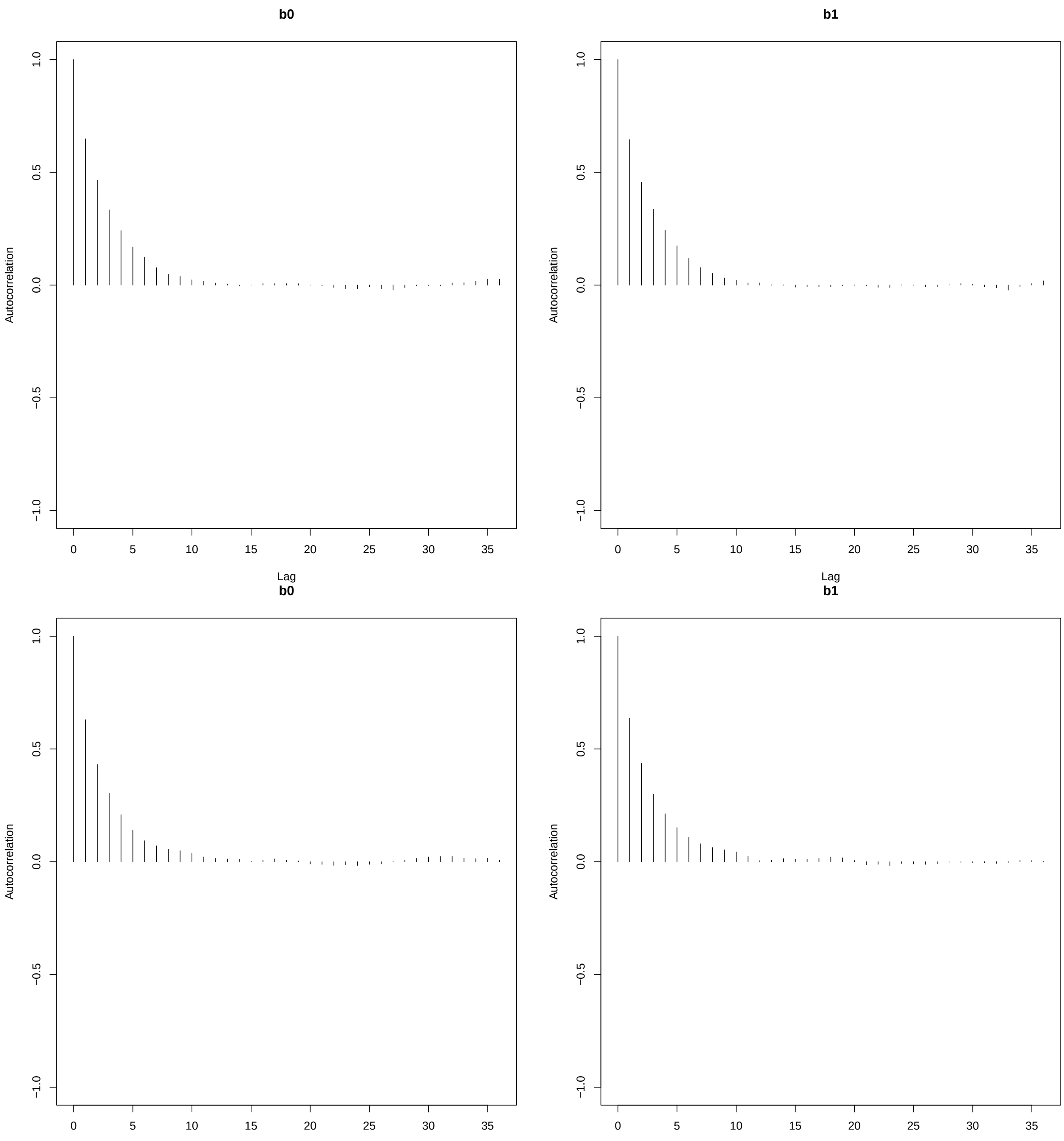

```
# model summary
library(MCMCvis)
MCMCsummary(parsPosterior)
##          mean        sd      2.5%       50%     97.5% Rhat n.eff
## b0   2.137294 0.5165667  1.225238  2.100916  3.244635    1  7399
## b1  -4.413154 0.9332448 -6.464204 -4.332850 -2.820204    1  7214
```

Trace plots, histograms look good and we get sensible potential scale reduction factors and effective sample sizes. So MCMC diagnostics look good.

Regarding effective sample size and autocorrelations, we can actually inspect these directly, though the ESSs are a good summary. The `autocorr.plot()` functions produces an autocorreltion plot for each parameter and each chain - here we have 2 parameters, 4 chains, hence 8 graphs.

```
par(mfrow=c(4,2))
autocorr.plot(parsPosterior)
```


b0
b1
b0
b1
Autocorrelation
Lag
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
25
30
35



We see that the autocorrelations drop rapidly and are negligble from iteration ~15 or so upwards.

In practice we would now also conduct posterior predictive checks to investigate that our model is suitable for the data. This is left as an exercise - but refer to the lecture notes for examples of this.

[end of STA623 BDA Practical 5]