
BEES
School of Biological,
Earth & Environmental
Sciences at University
College Cork, Ireland

BL6024 - Quantitative Skills for Biologists using R

Lecture 10: An Introduction to Bayesian Statistics

Contents

History of Bayesian statistics

- **Bayes**' rule

$$p(x, y) = p(x|y) \times p(y)$$

$$p(x, y) = p(y, x)$$

History of Bayesian statistics

- **Bayes**' rule

$$p(x, y) = p(x|y) \times p(y)$$

$$p(x, y) = p(y, x)$$

$$p(y|x) = \frac{p(x|y) \times p(y)}{p(x)}$$

History of Bayesian statistics

- **Bayes**' rule

$$p(x, y) = p(x|y) \times p(y)$$

$$p(x, y) = p(y, x)$$

$$p(y|x) = \frac{p(x|y) \times p(y)}{p(x)}$$

- Use in statistical inference

Priors and posteriors

$$p(\theta|x) = \frac{p(x|\theta) \times p(\theta)}{p(x)}$$

Priors and posteriors

$$p(\theta|x) = \frac{p(x|\theta) \times p(\theta)}{p(x)}$$

Simplified to: $Posterior\ distribution \sim Likelihood \times Prior\ distribution$

i.e. we update our prior belief in light of the data to get a posterior distribution for the parameters

$p(x)$ → Evidence (normalization)

We think Bayesian!

Bayesian vs. Frequentist

- Both: data are observed realizations of stochastic systems containing random processes

- Classical (frequentist) stats: the quantities used to describe these random processes (parameters) are fixed and unknown constants

- Bayesian stats: the parameters are viewed as unobserved realizations of random processes

Bayesian vs. Frequentist

- Both: data are observed realizations of stochastic systems containing random processes

- Classical (frequentist) stats: the quantities used to describe these random processes (parameters) are fixed and unknown constants

- Bayesian stats: the parameters are viewed as unobserved realizations of random processes

Estimating a single point vs. a distribution

Bayesian vs. Frequentist

Uncertainty:

- **Classical (frequentist) stats**: *frequency of hypothetical replicates*
- **Bayesian stats**: *posterior distribution of a parameter, given the data, the model and the priors*

Bayesian vs. Frequentist

Uncertainty:

- **Classical (frequentist) stats**: *frequency of hypothetical replicates*
- **Bayesian stats**: *posterior distribution of a parameter, given the data, the model and the priors*

Parameters θ are random variables $\rightarrow$ we can make probabilistic statements

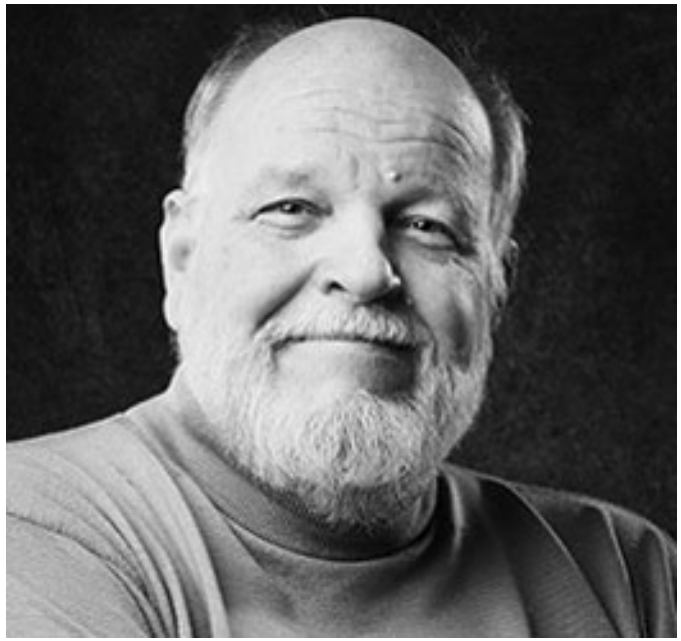
Pros & cons of the Bayesian approach

- Flexibility and tractability
- No asymptotics
- Incorporate existing information
- Error propagation
- Intuitive interpretation

- Prior choice
- Controversies on model selection
- Computing times

> *The argument in the academic community is mostly esoteric tail wagging anyway*.
>
> *In truth most analysts out of the ivory tower don't care that much, if at all, about Bayesian vs. Frequentist*.

(Rob Balon)



Bayesian machinery

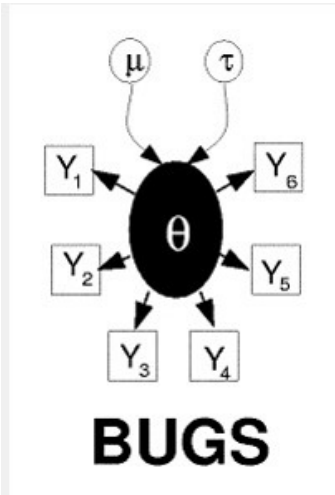
- Problem is the estimation of $p(x)$:

$$p(x) = \int p(x, \theta)\, d\theta$$

- *Solution*: approximate inference

Bayesian machinery

- Markov Chain Monte Carlo (MCMC) algorithms
- Not the only solution (e.g. INLA)
- *Software*: WinBUGS (OpenBUGS), JAGS, Stan, Nimble, MCMCglmm, ...



NIMBLE

Example: Gibbs sampler

- x =data, θ =vector of k unknowns $(\theta_1, \theta_2, \ldots, \theta_k)$

Example: Gibbs sampler

- x =data, θ =vector of *k* unknowns $(\theta_1, \theta_2, \ldots, \theta_k)$
- Choose starting values $\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_k^{(0)}$

Example: Gibbs sampler

- x =data, θ =vector of k unknowns $(\theta_1, \theta_2, \ldots, \theta_k)$
- Choose starting values $\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_k^{(0)}$
- Sample $\theta_1^{(1)}$ from $p(\theta_1 | \theta_2^{(0)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, x)$
  Sample $\theta_2^{(1)}$ from $p(\theta_2 | \theta_1^{(1)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, x)$
  .....
  Sample $\theta_k^{(1)}$ from $p(\theta_k | \theta_1^{(1)}, \theta_2^{(1)}, \ldots, \theta_{k-1}^{(1)}, x)$

Example: Gibbs sampler

- x =data, θ =vector of k unknowns $(\theta_1, \theta_2, \ldots, \theta_k)$
- Choose starting values $\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_k^{(0)}$
- Sample $\theta_1^{(1)}$ from $p(\theta_1 | \theta_2^{(0)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, x)$
  Sample $\theta_2^{(1)}$ from $p(\theta_2 | \theta_1^{(1)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, x)$
  .....
  Sample $\theta_k^{(1)}$ from $p(\theta_k | \theta_1^{(1)}, \theta_2^{(1)}, \ldots, \theta_{k-1}^{(1)}, x)$
- Repeat previous step many times to get a good approximation of $p(\theta | x)$

Example: Gibbs sampler

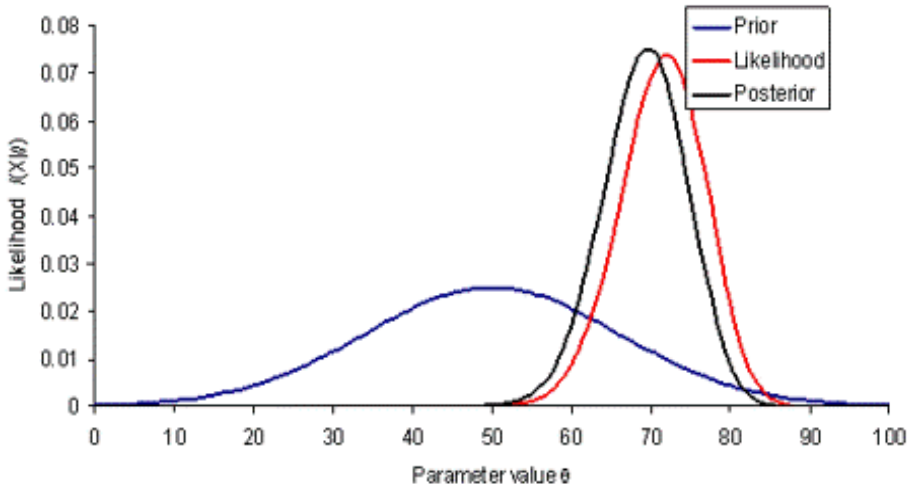
- x =data, θ =vector of *k* unknowns $(\theta_1, \theta_2, \ldots, \theta_k)$
- Choose starting values $\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_k^{(0)}$
- Sample $\theta_1^{(1)}$ from $p(\theta_1 | \theta_2^{(0)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, x)$
  Sample $\theta_2^{(1)}$ from $p(\theta_2 | \theta_1^{(1)}, \theta_3^{(0)}, \ldots, \theta_k^{(0)}, x)$
  .....
  Sample $\theta_k^{(1)}$ from $p(\theta_k | \theta_1^{(1)}, \theta_2^{(1)}, \ldots, \theta_{k-1}^{(1)}, x)$
- Repeat previous step many times to get a good approximation of $p(\theta | x)$
- The sequence of random draws for each parameter *k* forms a Markov chain

BUGS/JAGS in practice

- Define model
- Set priors and constraints
- Compile model based on data
- Initialise chains
- Iterate chains until convergence
- Obtain posterior sample
- Monitor convergence and prior sensitivity
- Model validation and selection

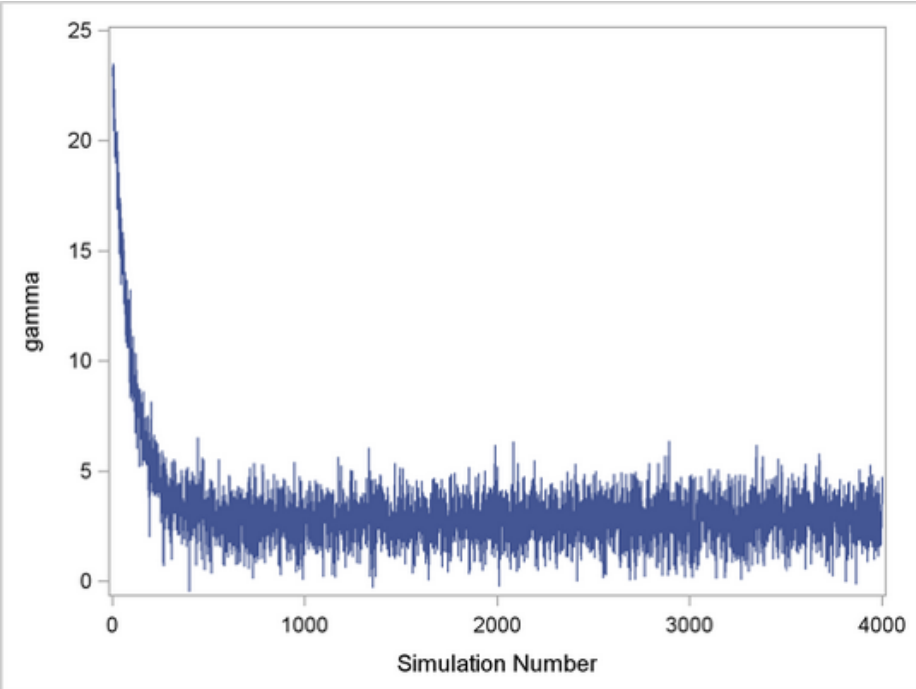
Choosing your priors

- What is a reasonable prior?
- Uninformative vs. informative
- Can have subtle effects in multidimensional cases
- Assess prior sensitivity



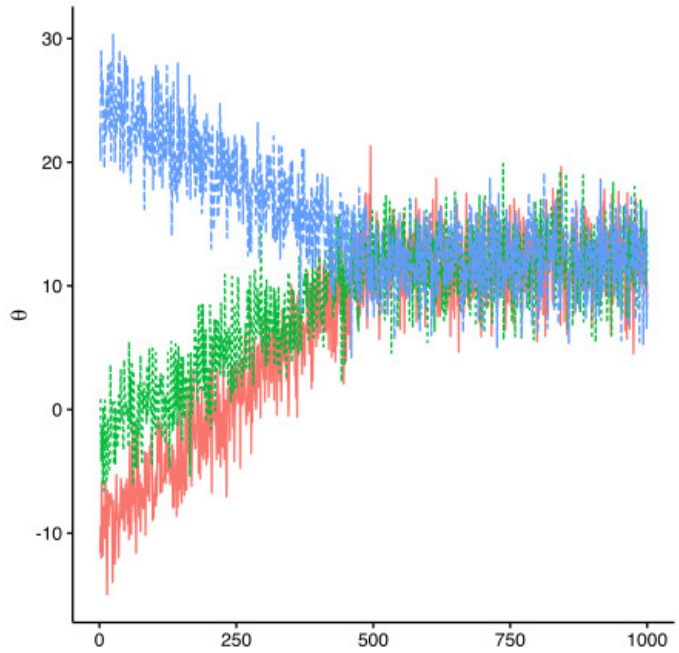
Convergence monitoring

- *Trace plots and burn-in*



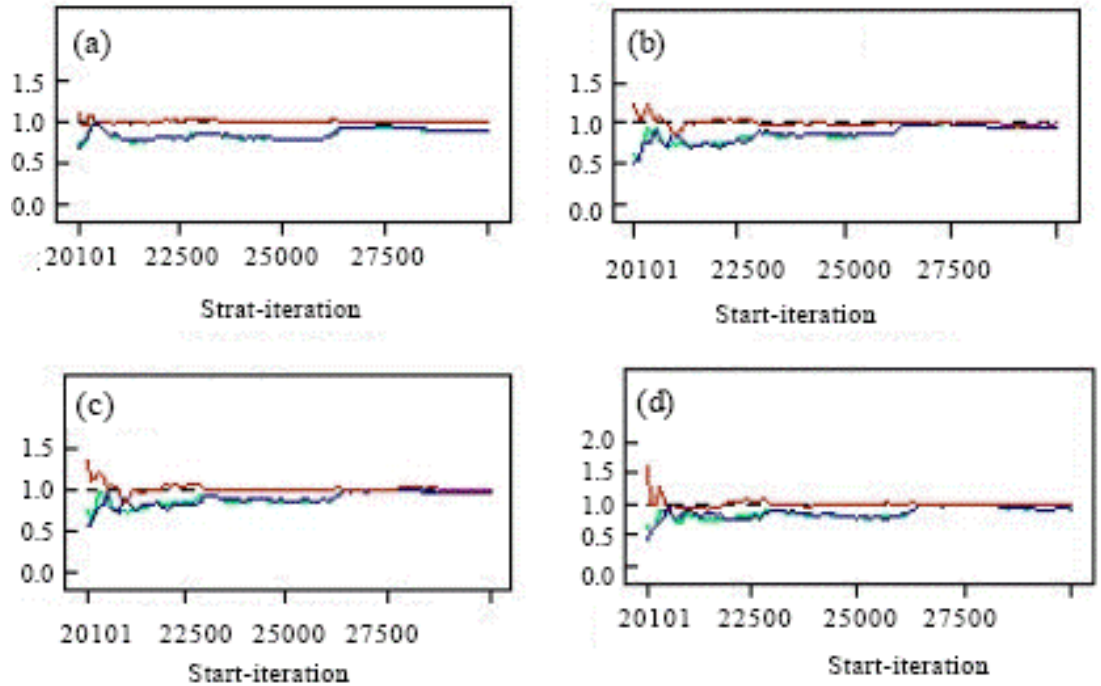
Convergence monitoring

- Trace plots and burn-in
- *Use multiple parallel chains*



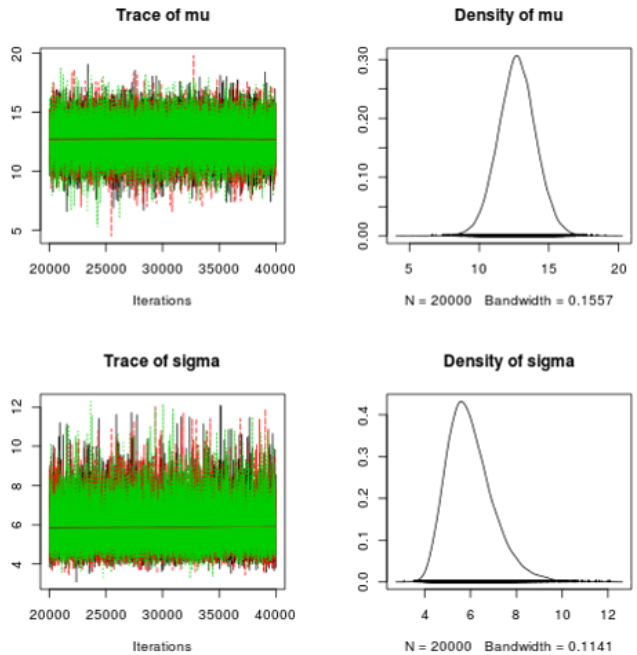
Convergence monitoring

- Trace plots and burn-in
- Use multiple parallel chains
- *Brooks-Gelman-Rubin (BGR) diagnostic*



Convergence monitoring

- Trace plots and burn-in
- Use multiple parallel chains
- Brooks-Gelman-Rubin (BGR) diagnostic
- *Density plots*

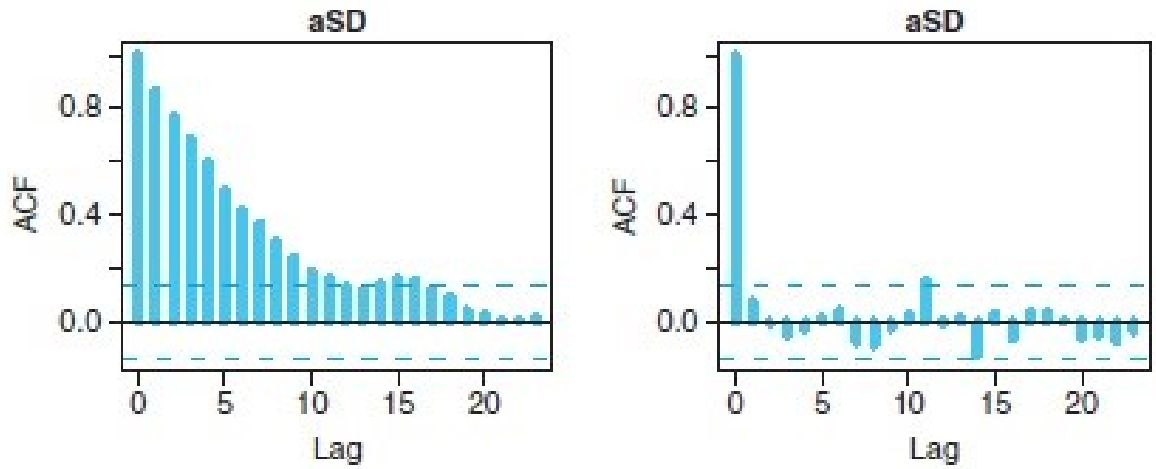


Convergence monitoring

- Trace plots and burn-in
- Use multiple parallel chains
- Brooks-Gelman-Rubin (BGR) diagnostic
- Density plots
- *Monte Carlo error*

Convergence monitoring

- Trace plots and burn-in
- Use multiple parallel chains
- Brooks-Gelman-Rubin (BGR) diagnostic
- Density plots
- Monte Carlo error
- *Autocorrelation and thinning*

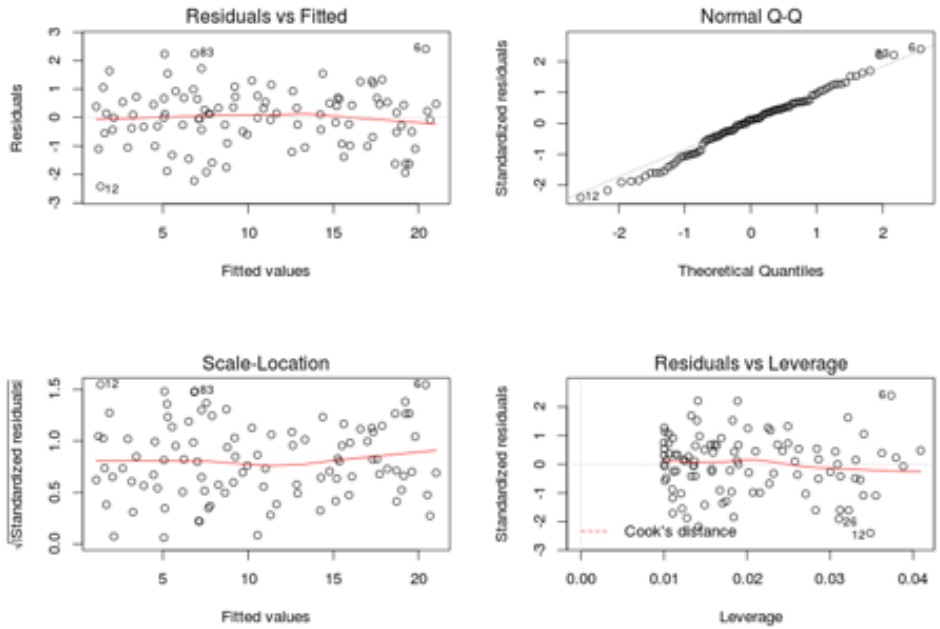


Convergence monitoring

- Trace plots and burn-in
- Use multiple parallel chains
- Brooks-Gelman-Rubin (BGR) diagnostic
- Density plots
- Monte Carlo error
- Autocorrelation and thinning
- *Effective sample size and posterior summary*

Model validation and selection

- Challenging (as in frequentist!)
- Residual diagnostics (for non-hierarchical models)



Model validation and selection

- Challenging (as in frequentist!)
- Residual diagnostics (for non-hierarchical models)
- Hierarchical models: cross-validation, validation with test data, posterior predictive checks.
- 95% credible intervals and overlap with 0
- Deviance Information Criterion (DIC)
- RJ-MCMC

Examples

- I will show you a couple of example in **OpenBUGS**
- A bit clunky, but it clarifies the process
- It can be run from **R** (`R2OpenBUGS` package)
- We'll do the last example in **JAGS** (very similar syntax but fully in `R`)

Example: estimating the mean

- In **OpenBUGS**

- Two datasets: 10 and 1,000 observations

Example: is the coin fair?

- Also in **OpenBUGS**

- The data includes a string of heads (1s) and tails (0s)

- These have different (non-Gaussian) distribution

Example: simple linear regression

- Data:

```
dat<-read.csv("Data_LinRegExample.csv",header=T)
head(dat)
```

```
##   Weight Temperature
## 1   3.37        11.0
## 2   3.06        10.3
## 3   3.76        14.9
## 4   3.26        12.8
## 5   1.79        10.2
## 6   2.94        14.5
```

Example: simple linear regression

- Model:

```
model
{
 #Priors
 beta[1] ~ dunif(-500,2000)
 beta[2] ~ dnorm(0,0.0001)
 std ~ dunif(0,1000)
 precision <- 1/std/std

 #Likelihood
 for (i in 1:nobs){
  lin.pred[i] <- beta[1] + beta[2]*Temperature[i]
  Weight[i] ~ dnorm(lin.pred[i], precision)
  }
}
```

Example: simple linear regression

```
library(runjags)

#data
dim(dat)
```

```
## [1] 1000    2
```

```
dataList<- list(nobs=1000, Weight=dat$Weight, Temperature=dat$Temperature)

#initial values
initsList1<- list(beta=c(0,0), std=1)
initsList2<- list(beta=c(1,1), std=10)
initsList3<- list(beta=c(-1,-1), std=5)

#model file name
model.file <- "JAGS_LinRegExample_model.R"
```

Example: simple linear regression

```
#set parameters to monitor
par.monitor <- c("beta","std")

#run MCMC
psamples<-run.jags(model = model.file, monitor = par.monitor,
                   data = dataList, n.chains = 3, inits = list(initsList1, initsList2, initsList3),
                   burnin = 1000, adapt=200, thin=10, sample = 10000)
```

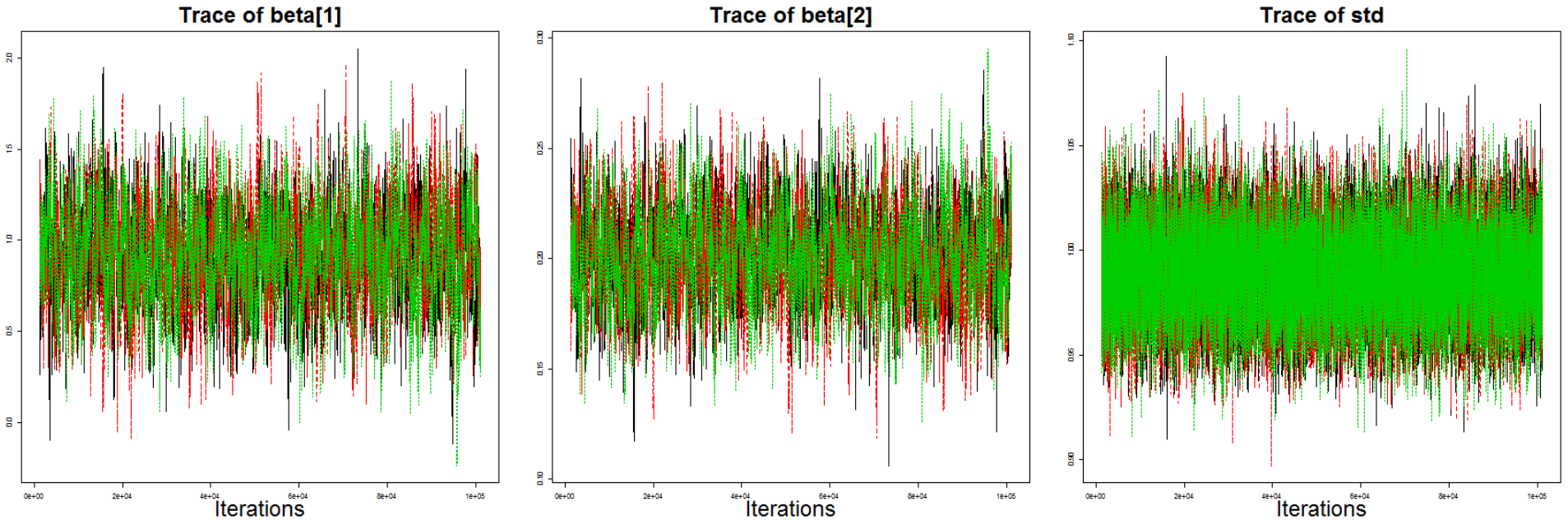
Example: simple linear regression

```
library(coda)

mcmc <-as.mcmc.list(psamples$mcmc)

#effective size
effectiveSize(mcmc)
```

```
##   beta[1]   beta[2]       std
##  1603.786  1606.760 30000.000
```

Example: simple linear regression

```
#trace plots
par(mfrow=c(1,3), cex.main=3, cex.lab=3)
traceplot(mcmc)
```



Example: simple linear regression

```
#BGR diagnostic
gelman.diag(mcmc)
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## beta[1]       1.01       1.02
## beta[2]       1.01       1.02
## std           1.00       1.00
##
## Multivariate psrf
##
## 1
```

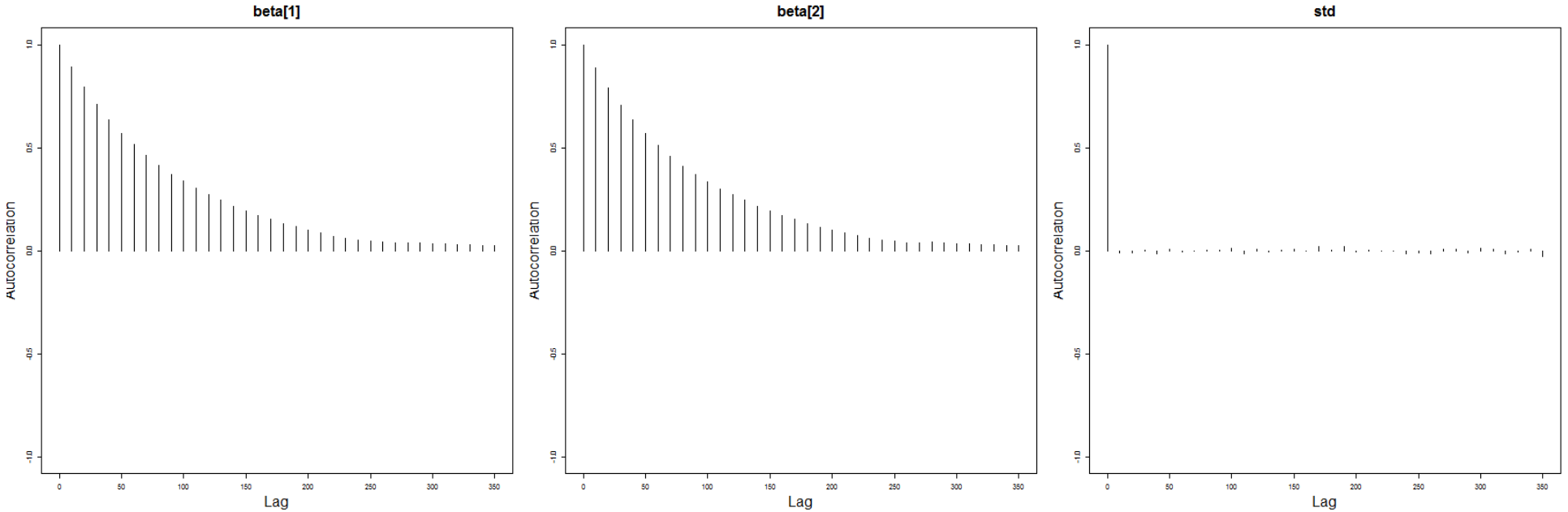
Example: simple linear regression

```
#MC error vs SD (%)
summa<-summary(mcmc)
summa[[1]][,4]/summa[[1]][,2]*100
```

```
##   beta[1]   beta[2]       std
## 2.5029190 2.5081865 0.5773279
```

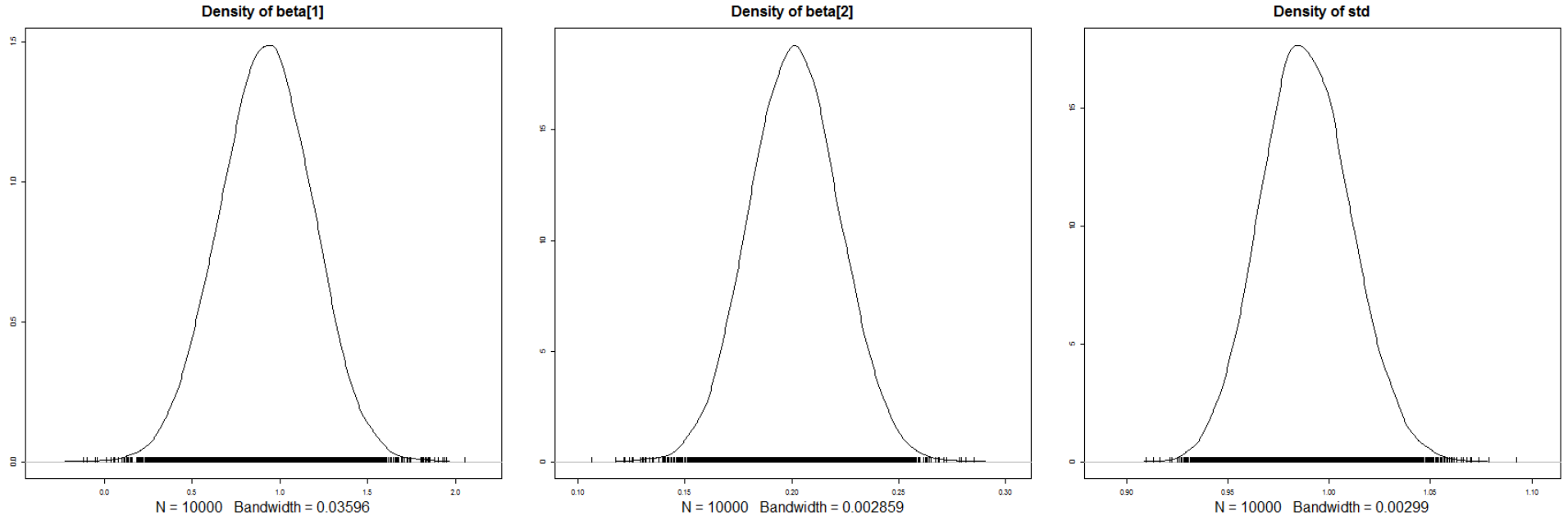
Example: simple linear regression

```
#autocorrelation plots
par(mfrow=c(1,3),cex.main=2,cex.lab=2)
autocorr.plot(mcmc[[1]], auto.layout = F)
```



Example: simple linear regression

```
#density plots
par(mfrow=c(1,3),cex.main=2,cex.lab=2)
densplot(mcmc)
```



Example: simple linear regression

```
#Summary of the posterior distribution
summary(mcmc)
```

```
##              Mean         SD     Naive SE Time-series SE
## beta[1] 0.9209643 0.26667889 0.0015396713   0.0066747567
## beta[2] 0.2014023 0.02120046 0.0001224009   0.0005317472
## std     0.9886692 0.02217291 0.0001280154   0.0001280104
```

```
##              2.5%       25%       50%       75%     97.5%
## beta[1] 0.3958284 0.7431943 0.9229834 1.1019540 1.4364026
## beta[2] 0.1605388 0.1870467 0.2012812 0.2155177 0.2432101
## std     0.9464767 0.9734196 0.9880827 1.0033114 1.0331482
```

Practical - Objectives

- Estimate the mean and standard deviation of a set of numbers, and the parameters of a simple linear regression (using the example from Session 4)
- Familiarise yourself with BUGS language and `R2OpenBUGS` and `coda` packages in `R`
- Compare the results obtain in a frequentist vs. Bayesian setting

Any questions?

Acknowledgements:
Introduction to WinBUGS for Ecologists - M. Kéry
Adam Kane