Introduction

Paolo Eusebi

16/09/2021

Bayes Rule

Bayes' theorem is at the heart of Bayesian statistics:

$$P(\theta|Y) = \frac{P(\theta) \times P(Y|\theta)}{P(Y)}$$

where:

- θ is our parameter value(s);
- Y is the data that we have observed;
- $P(\theta|Y)$ is the posterior probability of the parameter value(s);
- $P(\theta)$ is the prior probability of the parameters;
- $P(Y|\theta)$ is the likelihood of the data given the parameters value(s);
- $P(Y)$ is the probability of the data, integrated over parameter space.

Bayesian statistics

- In practice we usually work with the following:

$$P(\theta|Y) \propto P(\theta) \times P(Y|\theta)$$

- Our Bayesian posterior is therefore always a combination of the likelihood of the data $P(Y|\theta)$, and the parameter priors $P(\theta)$.

MCMC

- A way of obtaining a numerical approximation of the posterior
- Highly flexible
- Not inherently Bayesian but most widely used in this context
- Assessing convergence is essential, otherwise we may not be summarising the true posterior
- Our chains are correlated so we need to consider the effective sample size

Theory and application of MCMC

MCMC in practice

We can write a Metropolis algorithm but this is complex and inefficient

There are a number of general purpose languages that allow us to define the problem and leave the details to the software:

- WinBUGS/OpenBUGS
- Bayesian inference Using Gibbs Sampling
  - JAGS(Just Another Gibbs Sampler)
    - Stan (named in honour of Stanislaw Ulam, pioneer of the Monte Carlo method)

JAGS

JAGS uses the BUGS language

- This is a declarative (non-procedural) language
- The order of statements does not matter
- The compiler converts our model syntax into an MCMC algorithm with appropriately defined likelihood and priors
- You can only define each variable once!!!

Different ways to run JAGS from R: rjags, runjags, R2jags, jagsUI

JAGS

A simple JAGS model might look like this:

```
basicjags <- "model{
  # Likelihood part:
  Positives ~ dbinom(prevalence, TotalTests)

  # Prior part:
  prevalence ~ dbeta(2, 2)

  # Hooks for automatic integration with R:
  #data# Positives, TotalTests
  #monitor# prevalence
  #inits# prevalence
}
"
```

JAGS

There are two model statements:

1. The number of *Positive* test samples is Binomially distributed with probability parameter *prevalence* and total trials *TotalTests*

```
Positives ~ dbinom(prevalence, TotalTests)
```

2. Our prior probability distribution for the parameter *prevalence* is Beta(2,2)

```
prevalence ~ dbeta(2,2)
```

JAGS

The other lines in this model:

```
#data# Positives, TotalTests
#monitor# prevalence
#inits# prevalence
```

are automated hooks that are only used by runjags

This JAGS model is:

- Easy to write and understand
- Efficient (low autocorrelation)
- Fast to run

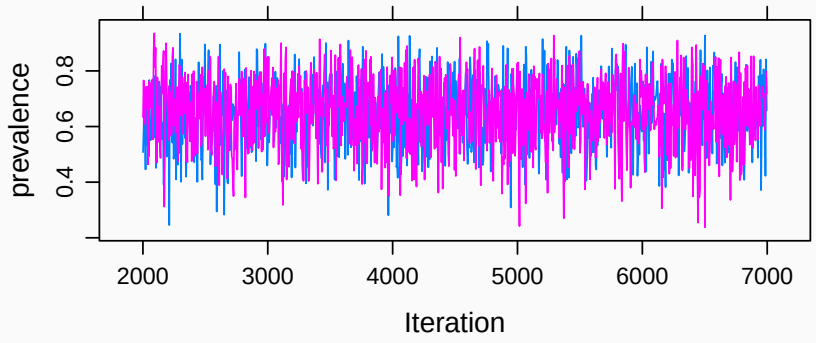
JAGS

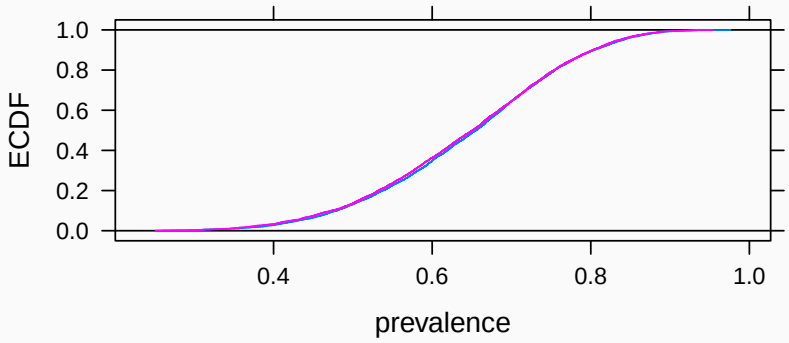
Let's run this model with some data.

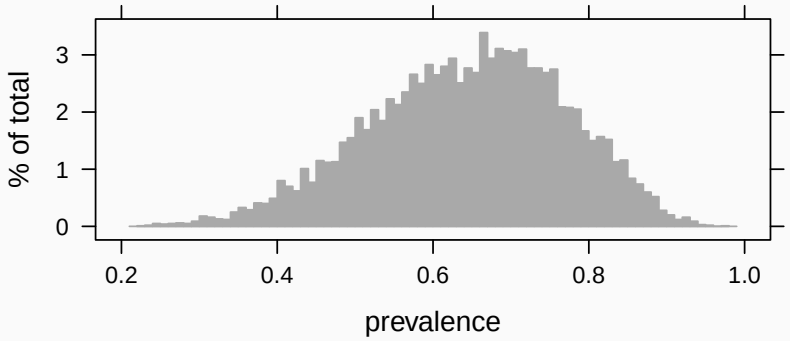
```
# data to be retrieved by runjags:
Positives <- 7
TotalTests <- 10
# initial values to be retrieved by runjags:
prevalence <- list(chain1=0.05, chain2=0.95)

results <- run.jags(model = basicjags,
                    n.chains = 2,
                    burnin = 1000,
                    sample = 5000)
```

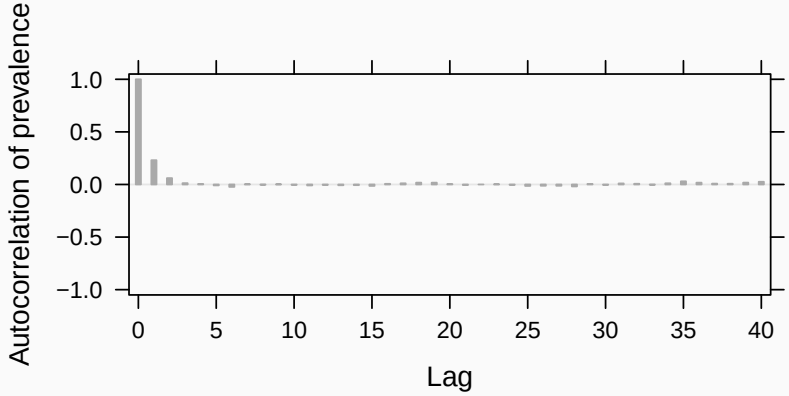
JAGS - Check the plots for convergence

```
## Generating plots...
```







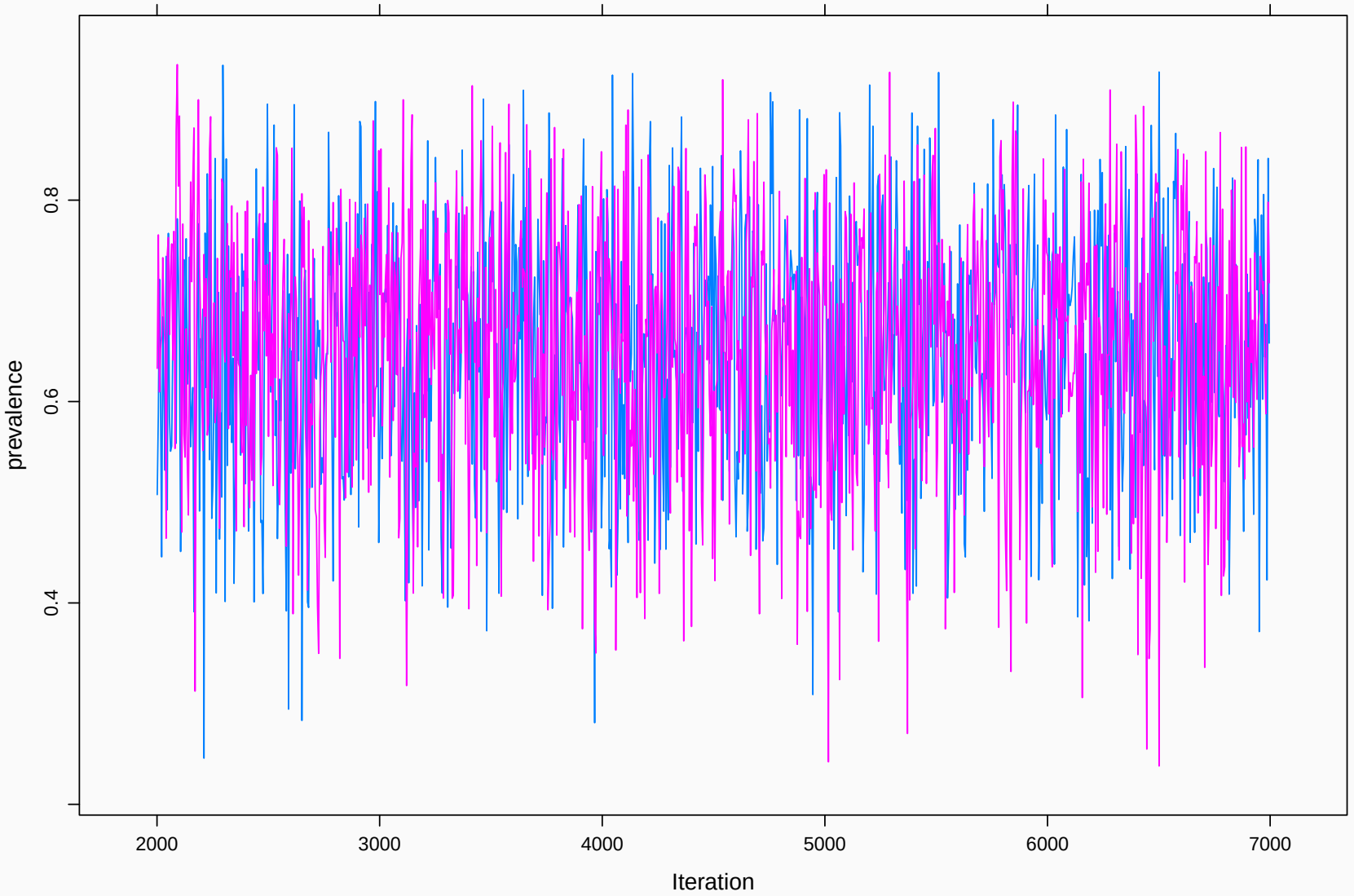



```
pt <- plot(results)
```

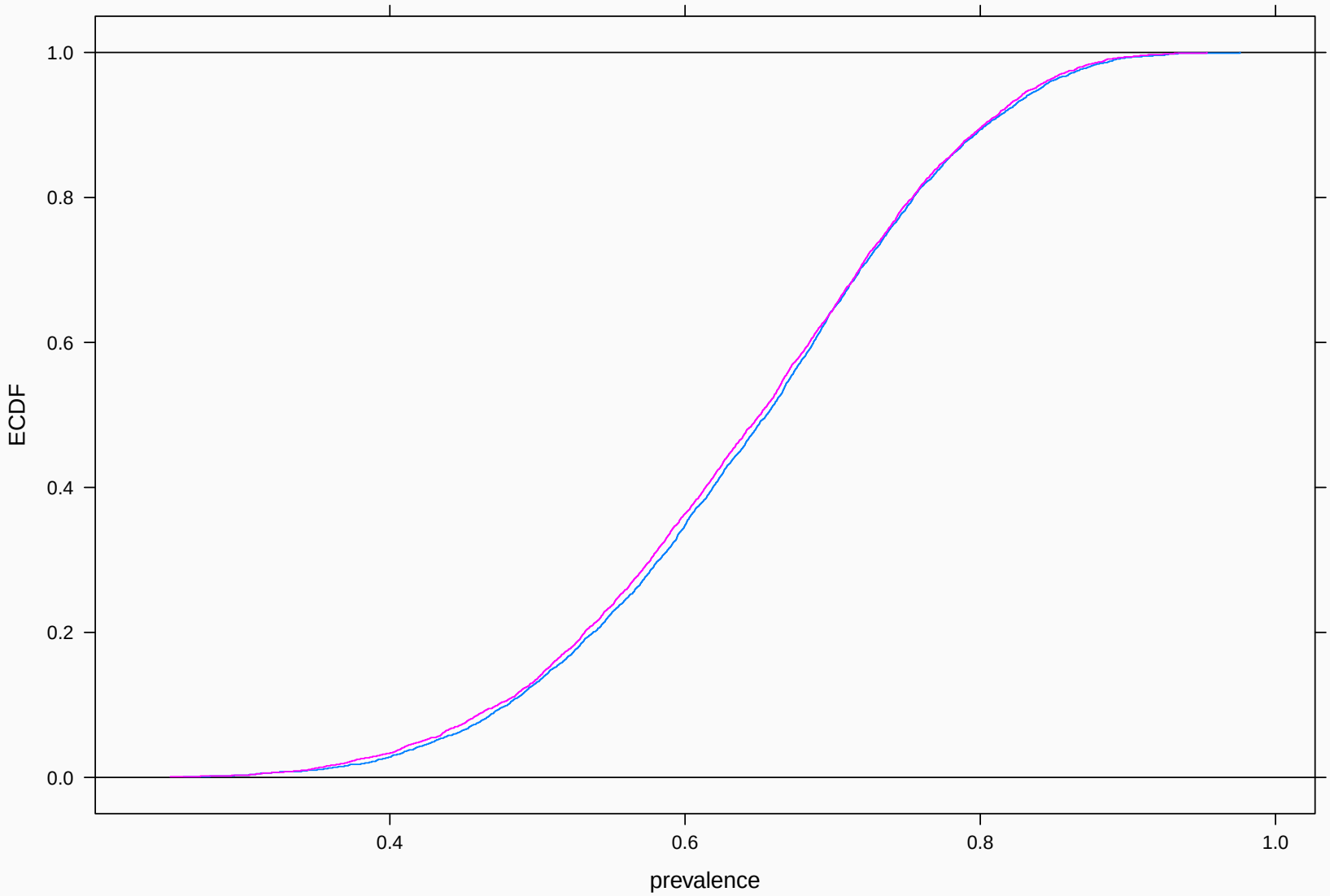


```
## Generating plots...
```

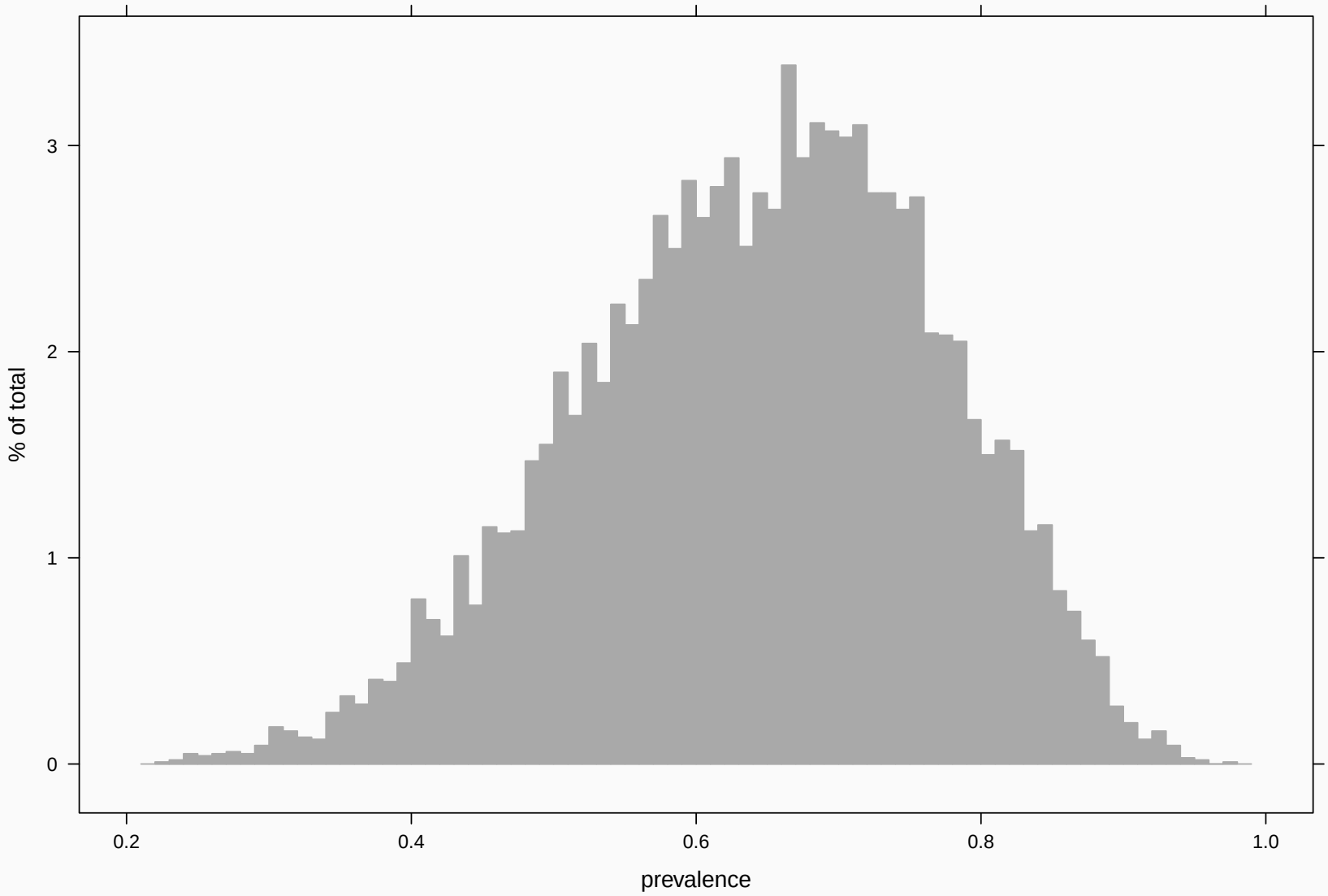
Trace plots: the two chains should be stationary:



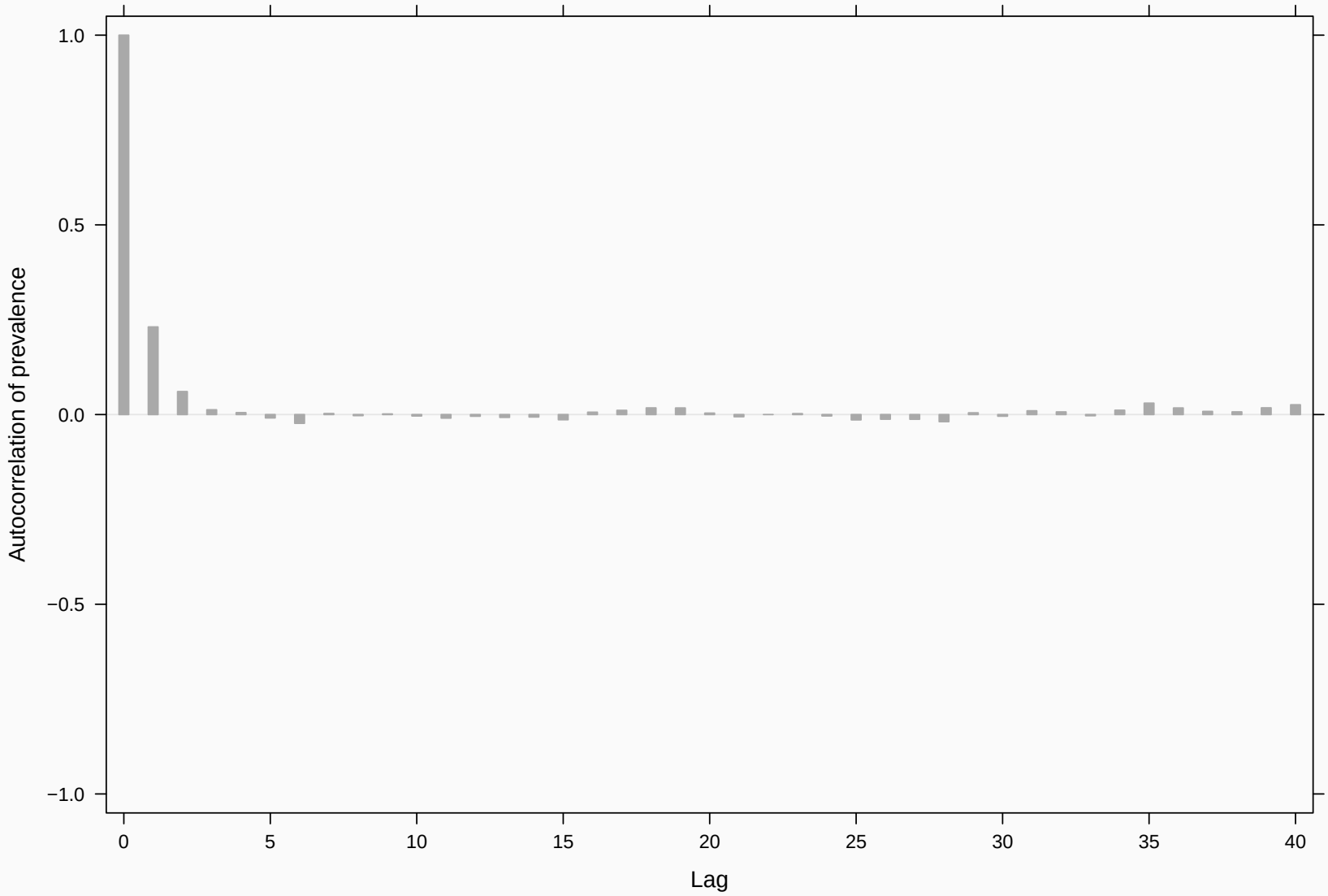
ECDF plots: the two chains should be very close to each other:



Histogram of the combined chains should appear smooth:



Autocorrelation plot tells you how well behaved the model is:



Then check the effective sample size (SSeff) and Gelman-Rubin statistic (psrf):

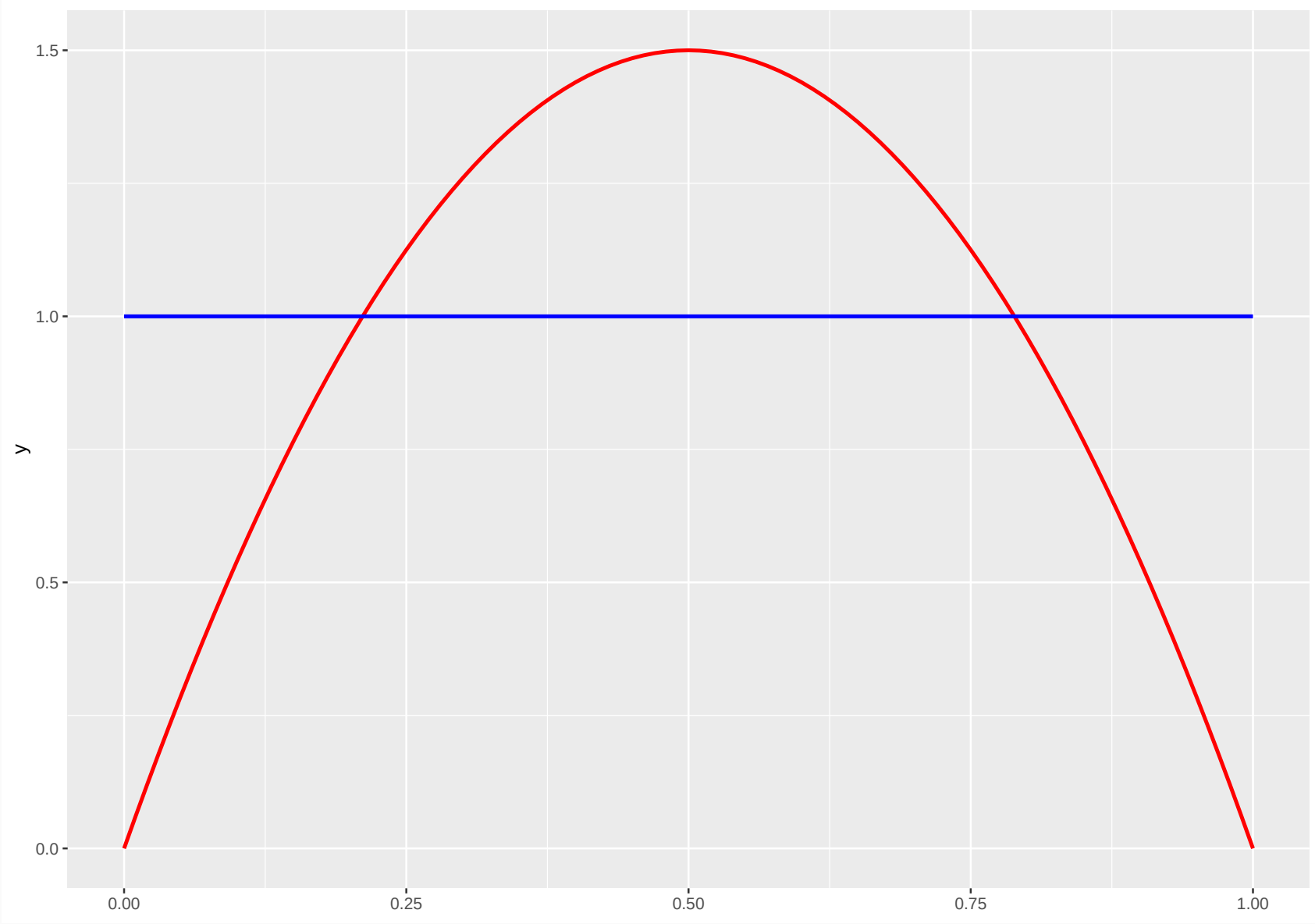
```
results
```

```
##
## JAGS model summary statistics from 10000 samples (chains
##
##            Lower95  Median Upper95    Mean      SD Mode
## prevalence 0.40088 0.65253 0.87294 0.64388 0.12506   --
##
##                AC.10   psrf
## prevalence -0.0042044 1.0003
##
## Total time taken: 0.8 seconds
```

Reminder: we want psrf < 1.05 and SSeff > 1000

Priors: the beta distribution

In blue $Beta(1, 1)$, in red $Beta(2, 2)$.



Exercise

- Run this model yourself in JAGS
- Change the initial values for the two chains and make sure it doesn't affect the results
- Reduce the burnin length - does this make a difference?
- Change the sample length - does this make a difference?
- Change the priors
- Increase the sample size