Introduction

Paolo Eusebi

16/09/2021

Introduction

Harmony

https://harmony-net.eu/

Figure 1: Harmony logo

Reproducibility

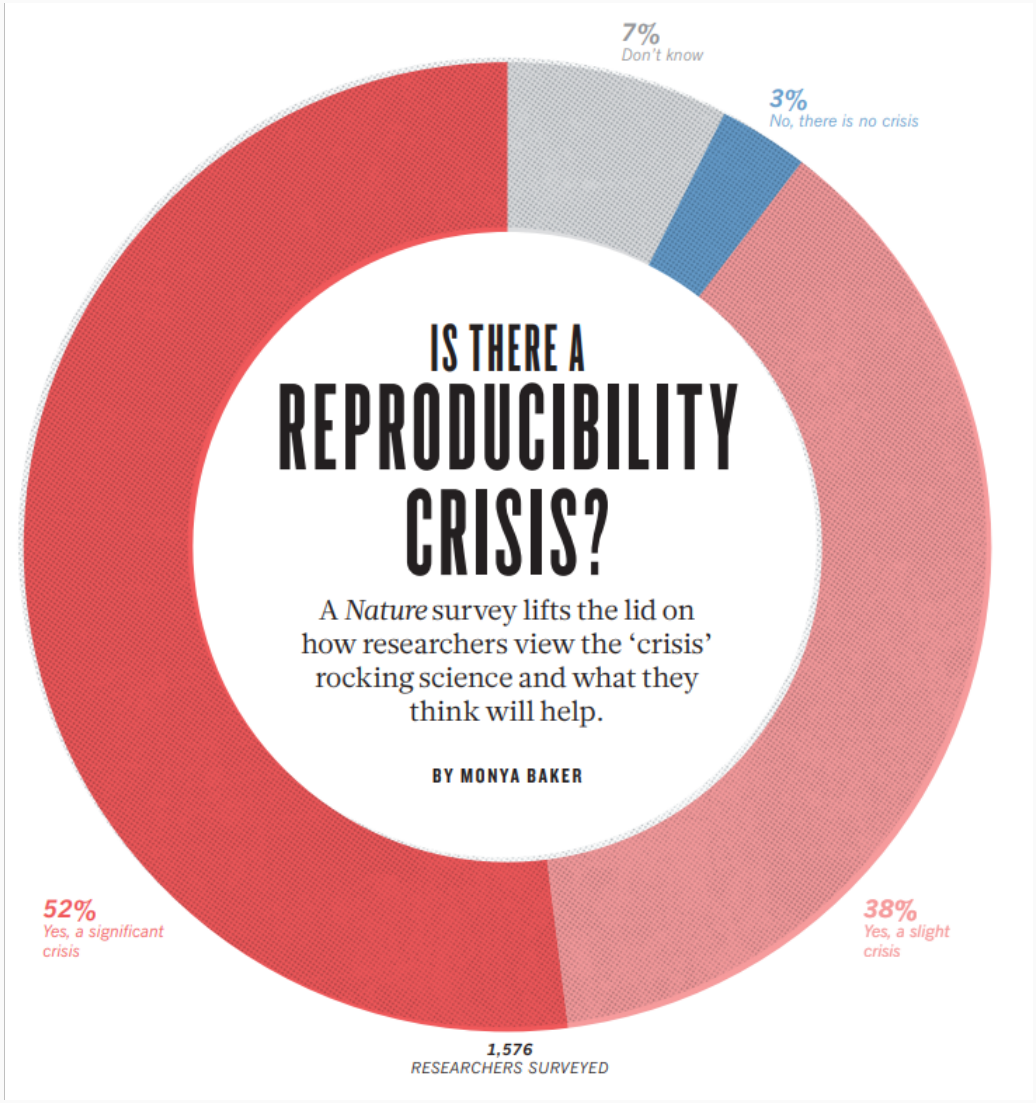


Figure 2: Baker Nature 2016

Reproducibility

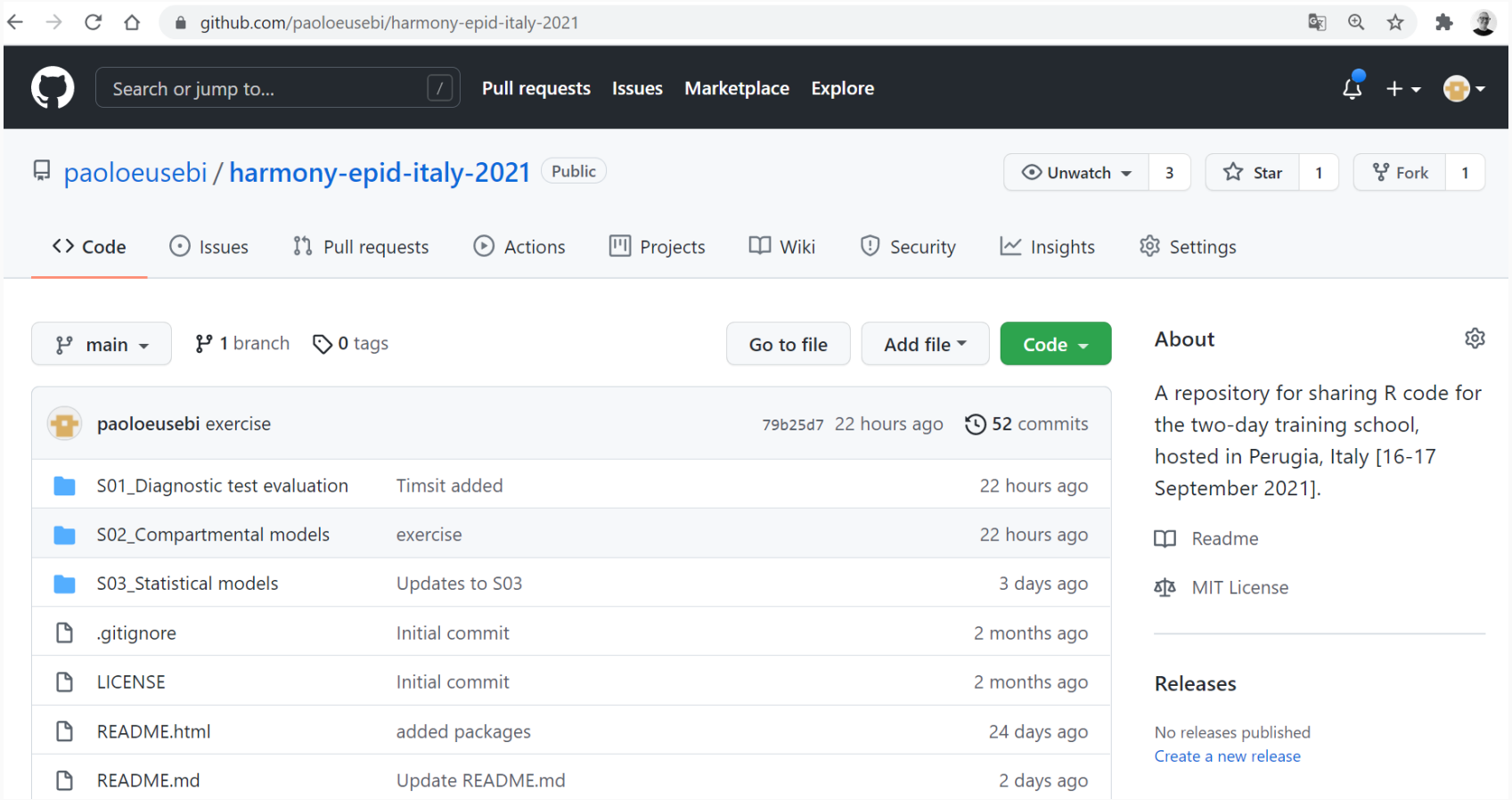


Figure 3: Snapshot of the GitHub Repo

Introduction to Bayesian Statistics

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 $P(\theta)$.

How to get a posterior distribution?

- Analytical derivation
- Numerical approximation

Analytical derivation

Analytical derivation could be feasible in some simple cases (Beta-binomial model)

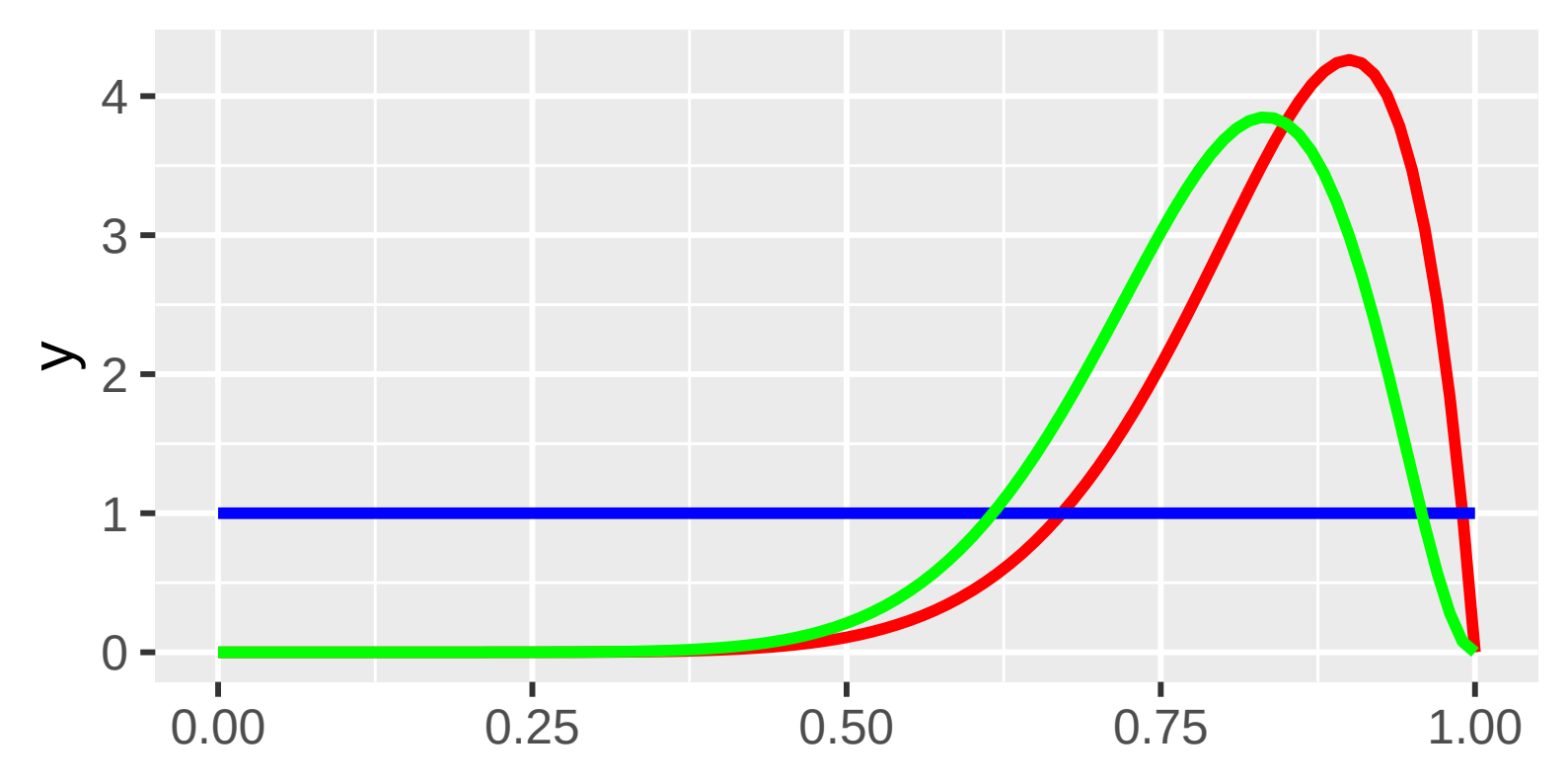
- Prior $p \sim Beta(\alpha, \beta)$
- Likelihood $Y|p \sim Bin(p, n)$
- Posterior $p|y \sim Beta(\alpha + y, \beta + n - y)$

Example: Suppose we are testing the fairness of a coin. We observed $n = 12$ flips and $y = 10$ heads. We assume an uninformative prior.

- Prior $p \sim Beta(\alpha = 1, \beta = 1)$
- Likelihood $Y|p \sim Bin(p, 12)$
- Posterior $p|y \sim Beta(\alpha = 11, \beta = 3)$

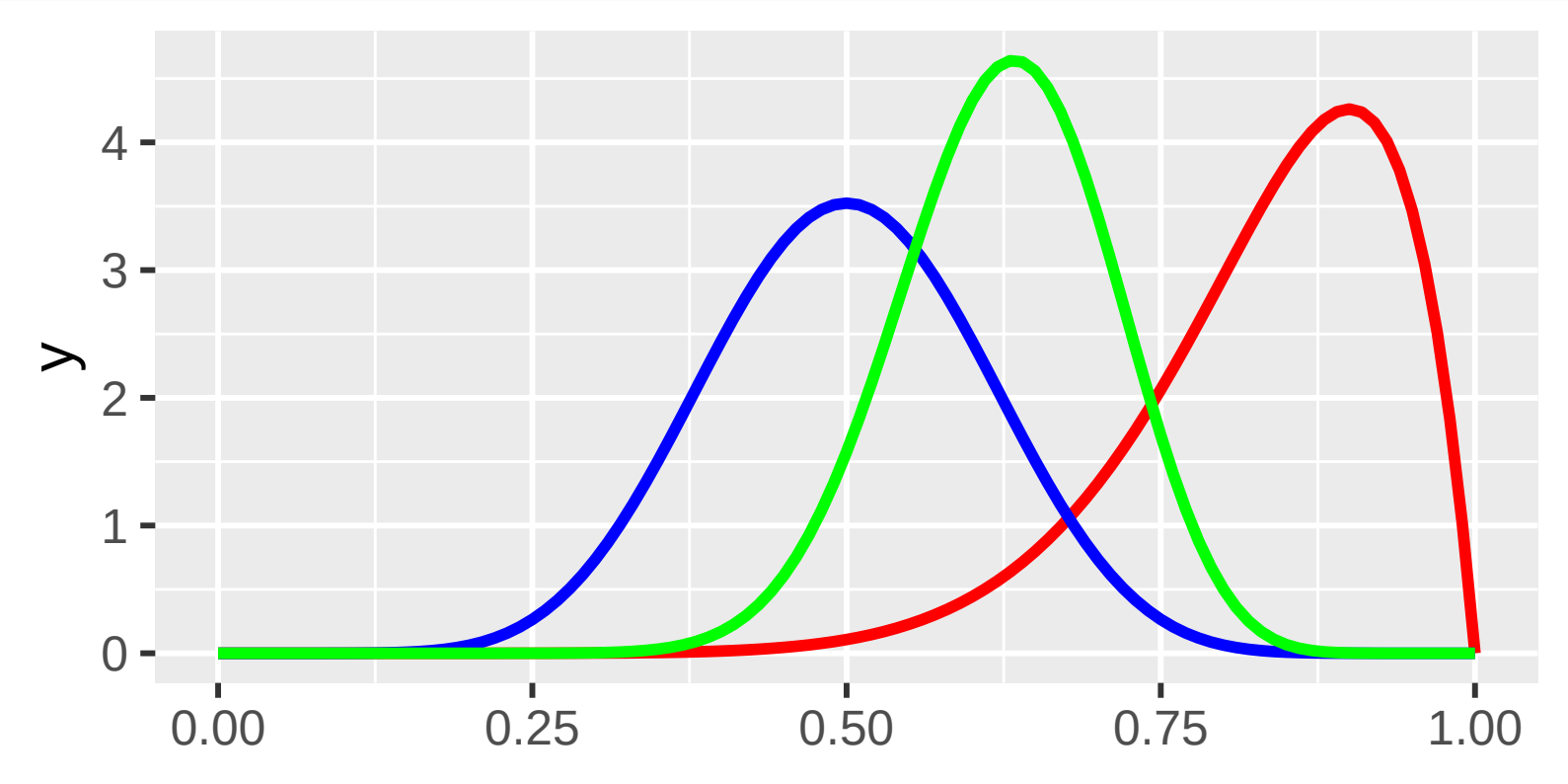
Analytical derivation

- Uniform Prior $Beta(1, 1)$
- Likelihood $Bin(p, 12)$
- Posterior $Beta(11, 3)$



Analytical derivation

- If I change the Prior to $Beta(10, 10)$
- Likelihood $Bin(p, 12)$
- Posterior $Beta(20, 12)$



A quick look at the Beta distribution.

- The Beta distribution is defined on the [0, 1] interval parameterized by two positive "shape" parameters α and β.
- Parameters of Beta distribution are commonly considered as "pseudo-counts" of successes (α) and failures (β)
- $Y \sim Beta(\alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$
- $E(Y) = \frac{\alpha}{\alpha+\beta}$
- $Var(Y) = \frac{\alpha\beta}{(\alpha+\beta)^2+(\alpha+\beta+1)}$

MCMC

- Markov chain Monte Carlo (MCMC) methods comprise a class of algorithms for sampling from a probability distribution.
- By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by recording states from the chain.
- The more steps are included, the more closely the distribution of the sample matches the actual desired distribution.
- Various algorithms exist for constructing chains, including the Metropolis–Hastings algorithm.

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–Hastings 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
- 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

Two model statements in this JAGS model:

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 a $Beta(2, 2)$

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

JAGS

The other lines in this model are automated hooks that are only used by runjags for:

1. Loading data

```
#data# Positives, TotalTests
```

2. Monitoring the posterior distribution of interest

```
#monitor# prevalence
```

3. Initializing the chains

```
#data# Positives, TotalTests
```

JAGS

This JAGS model is:

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

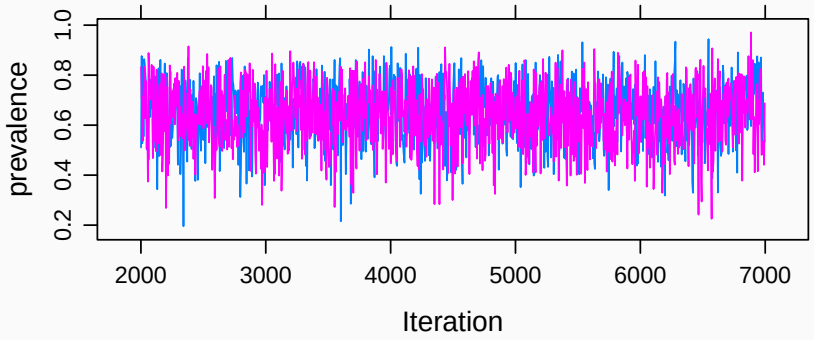
JAGS

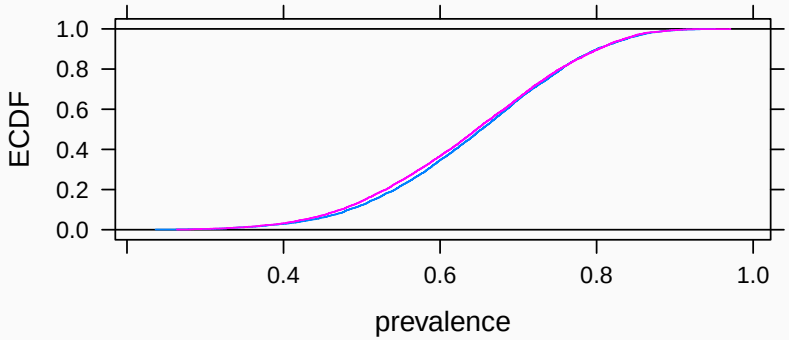
Let's run this model with some data.

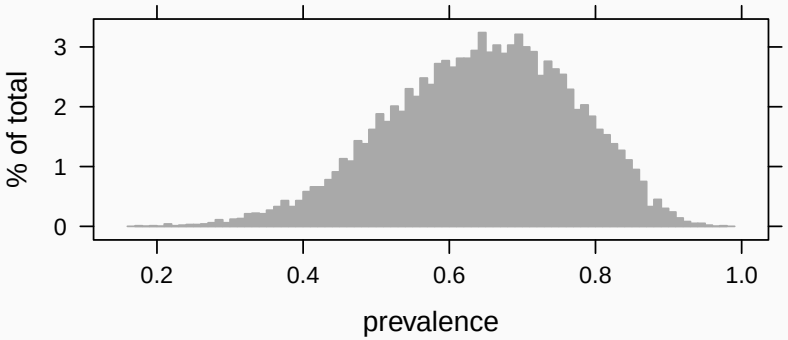
```
# data to be retrieved by runjags:
Positives <- 7
TotalTests <- 10
# initial values to be retrieved by runjags:
prevalence <- list(chain1=0.05, chain2=0.95)

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

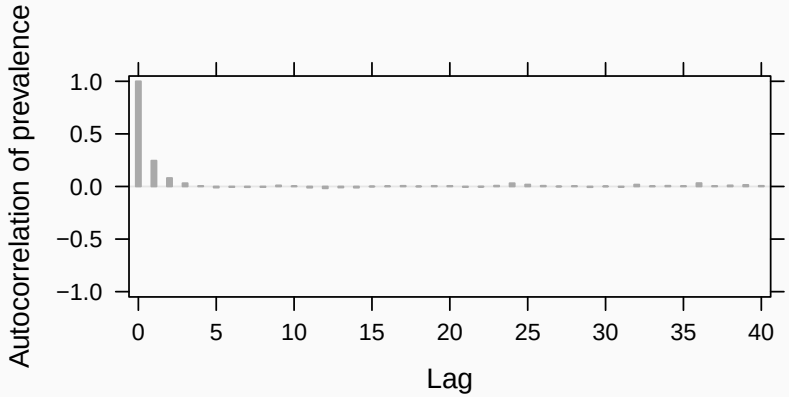
JAGS - Check the plots for convergence

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





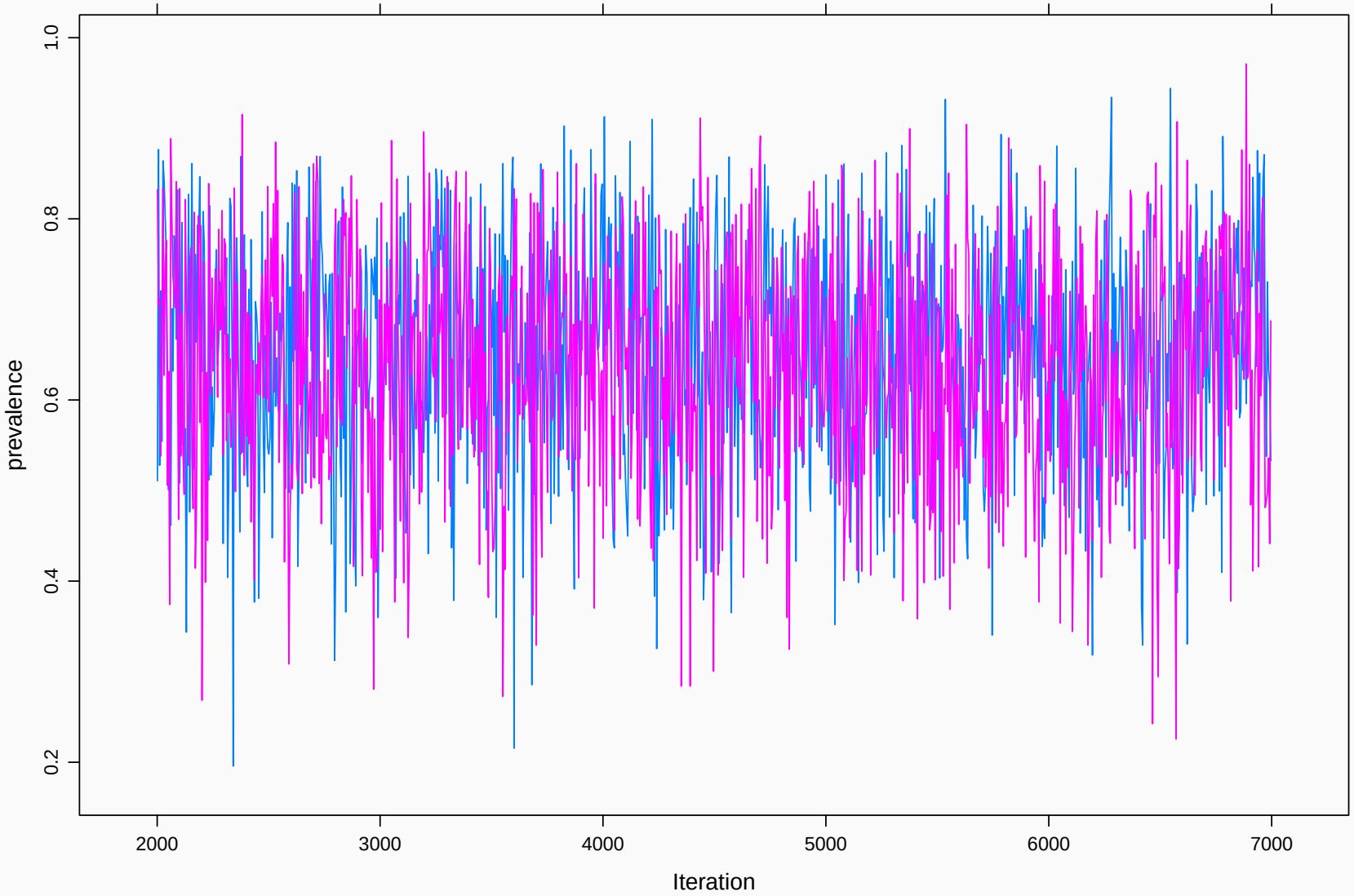





```
## Generating plots...
```

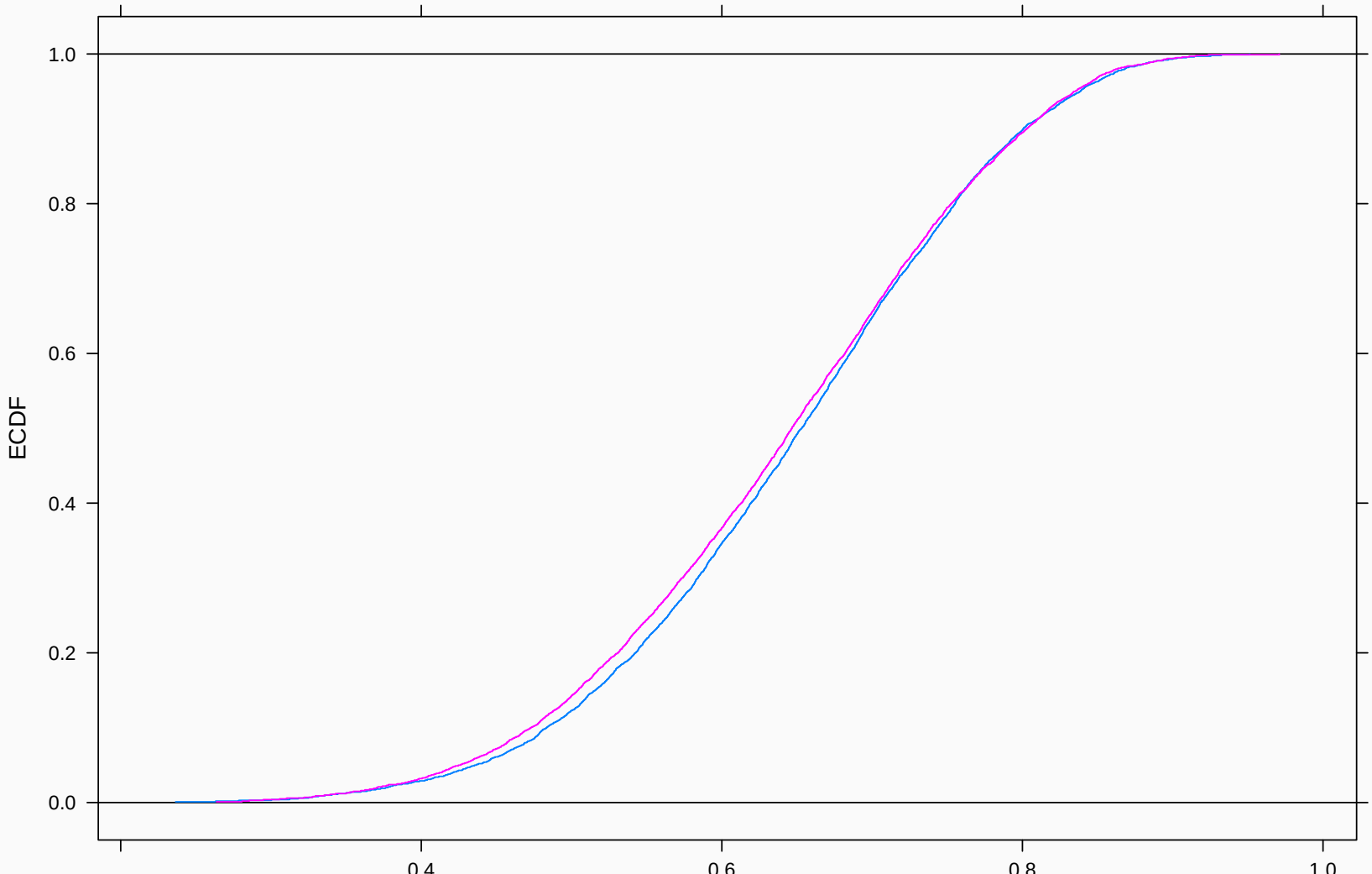
JAGS - Trace plots

The two chains should be stationary.



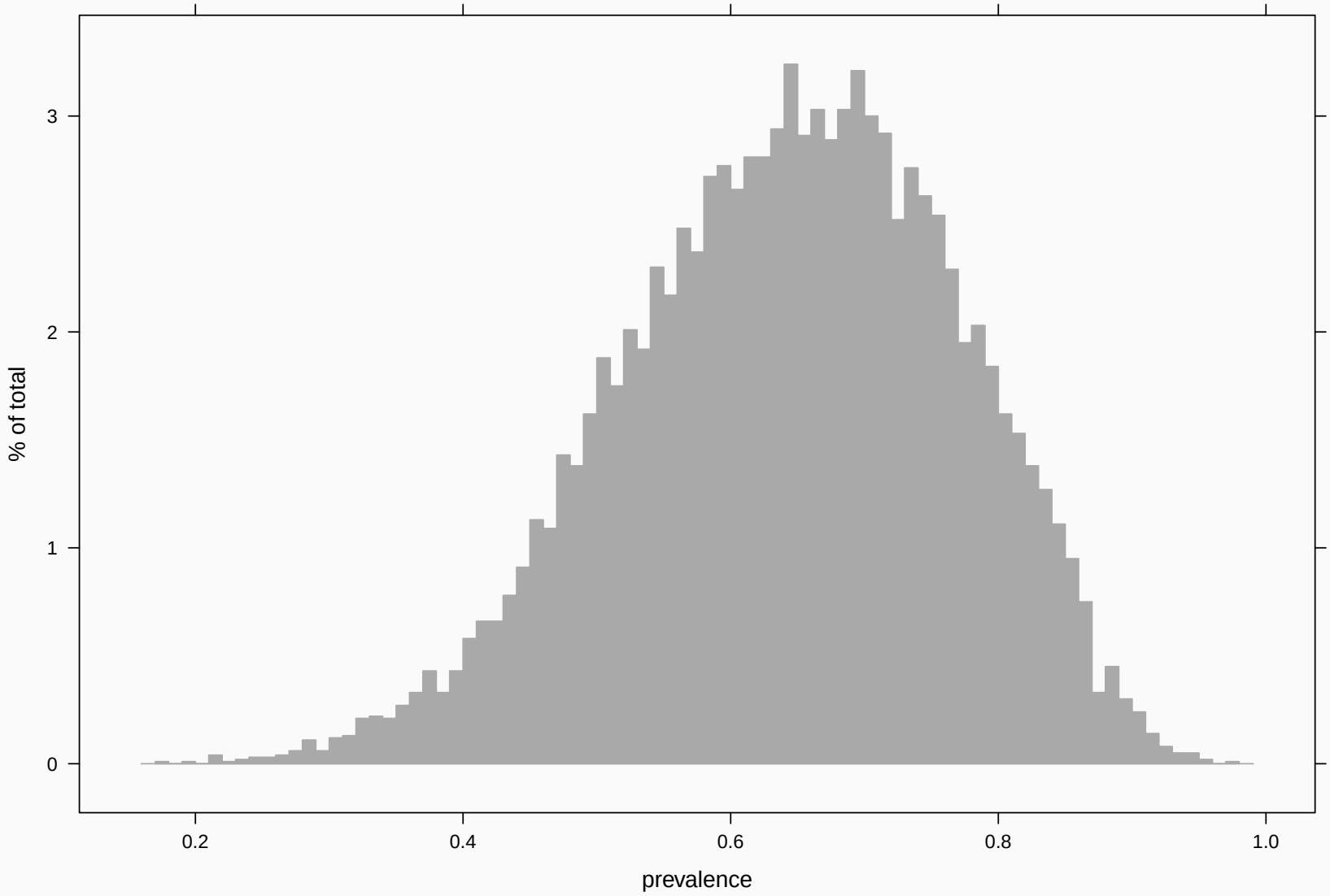
JAGS - Empirical Cumulative Distribution Function (ECDF) plots

The two chains should be very close to each other.



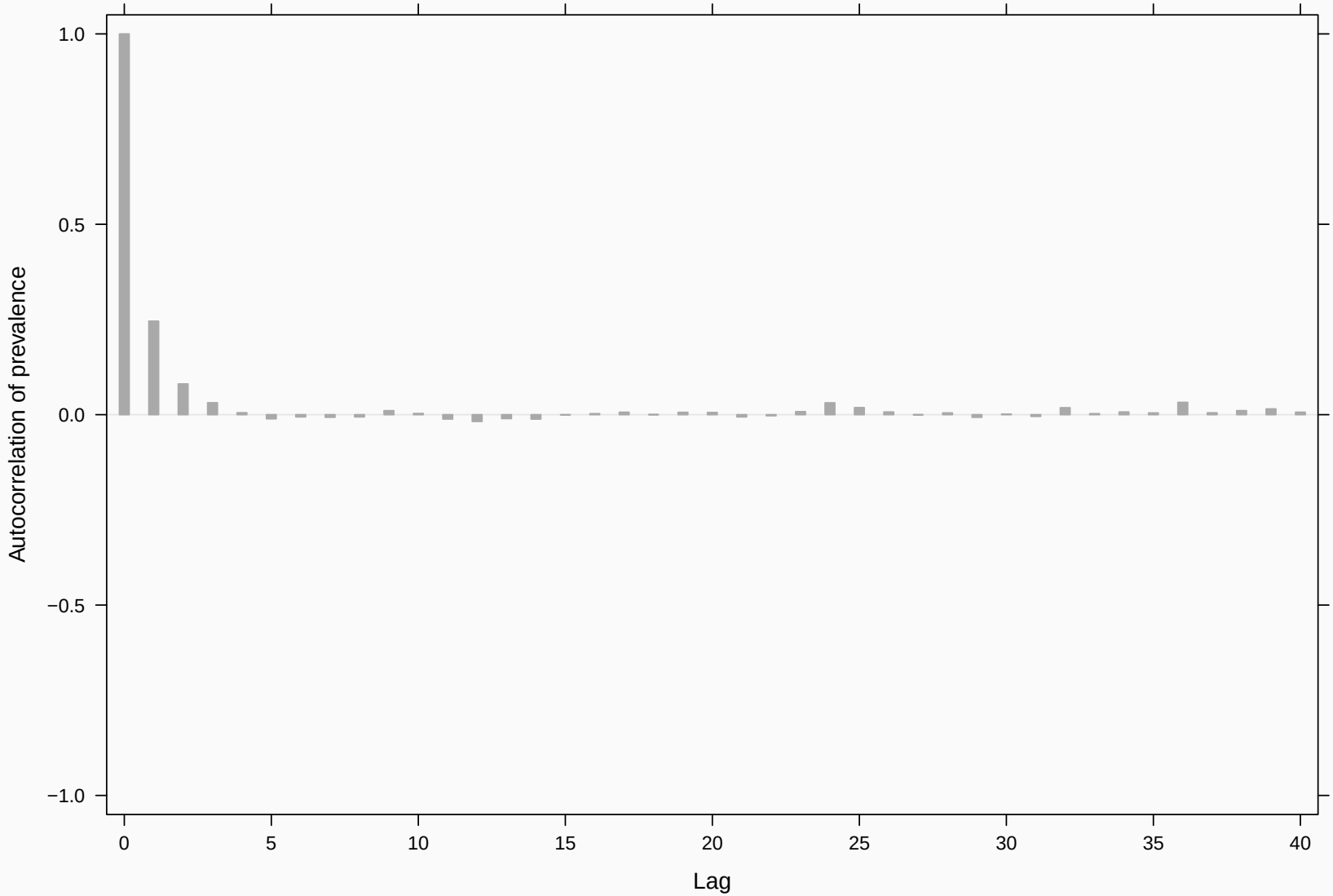
JAGS - Histograms

Histogram of the combined chains should appear smooth.



JAGS - Autocorrelation plots

Autocorrelation plot tells how well behaved the model.



JAGS - Diagnostics

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

```
##  Lower95   Median  Upper95    SSeff     psrf
##    0.403    0.649    0.868 5878.000    1.001
```

Reminder: we want psrf < 1.05 and SSeff > 1000

JAGS - Diagnostics

ESS definition:

$$ESS = \frac{n}{1 + 2\sum_{k=1}^{\infty} \rho(k)}$$

where n is the number of samples and $\rho(k)$ is the correlation at lag k.

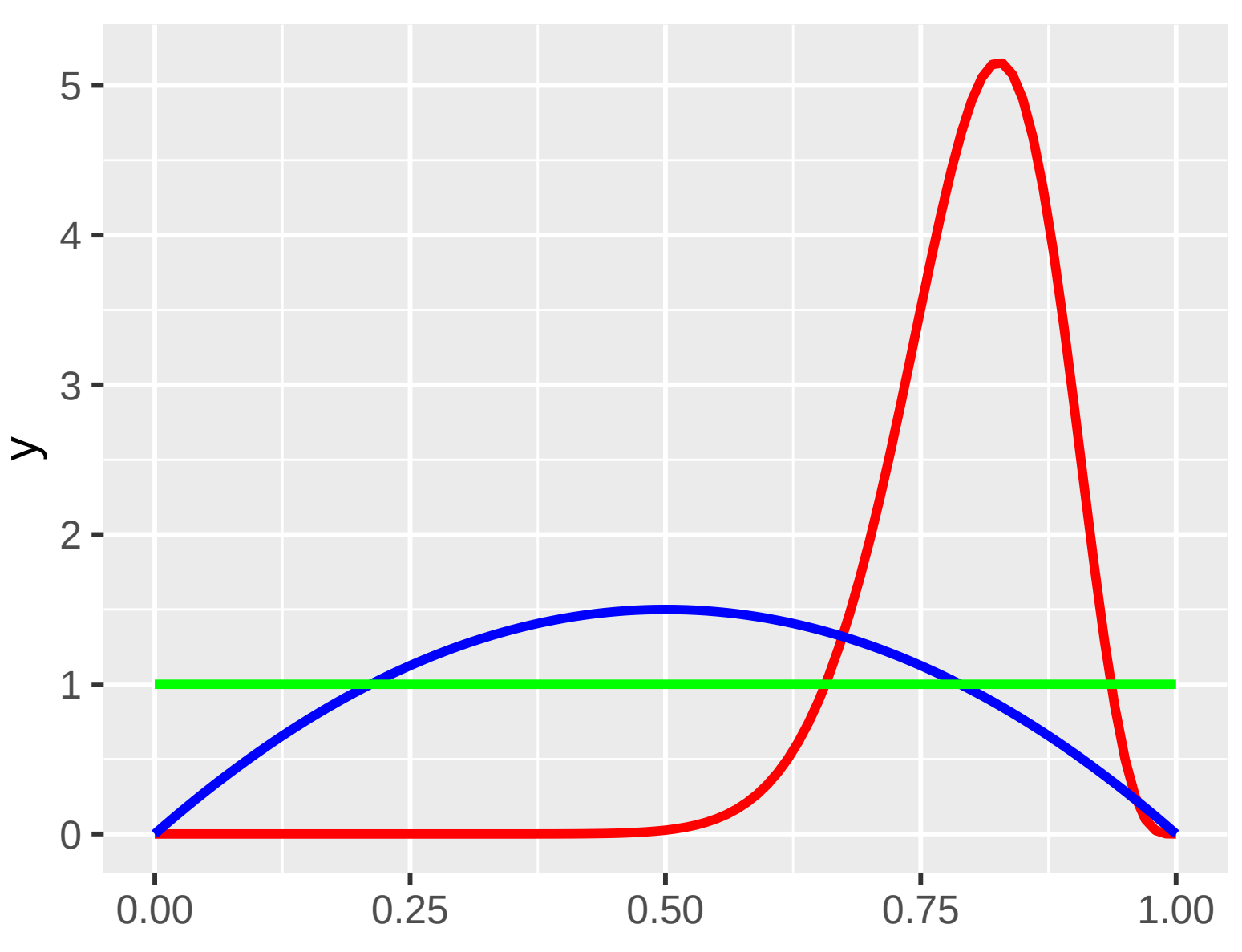
- If your samples are independent, your effective samples size equals the actual sample size.
- If the correlation at lag k decreases extremely slowly, so slowly that the sum in the denominator diverges, your effective sample size is zero.

JAGS - Diagnostics

- If psrf (potential scale reduction factor) is close to 1, we can conclude that each of the m sets of n simulated observations is close to the target distribution
- There is also a multivariate version mpsrf

Priors: the $Beta(\alpha, \beta)$ distribution

$Beta(2, 2)$ $Beta(20, 5)$ $Beta(1, 1)$



Priors: the $Beta(\alpha, \beta)$ distribution

Common choices:

- Uniform prior $Beta(1, 1)$ (Bayes-Laplace)
- Jeffreys prior $Beta(\frac{1}{2}, \frac{1}{2})$
- "Neutral" prior $Beta(\frac{1}{3}, \frac{1}{3})$ (Kerman 2011)
- Haldane prior $Beta(0, 0)$, or it's approximation $Beta(\epsilon > 0, \epsilon > 0)$

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