# MCMC - Stats and jAGS

## CA18208 HARMONY Zurich Training School - https://harmony-net.eu/

Sonja Hartnack  Valerie Hungerbühler  Eleftherios Meletis

2022-07-14

# Markov chain

- Definition: A random process that undergoes transitions from one state to another on a state space

https://setosa.io/blog/2014/07/26/markov-chains/

# Markov chain

S S S R R R S S S S R R R R R R R R R R R R R S S S S S S S S

One way to simulate this weather would be to just say "Half of the days are rainy. Therefore, every day in our simulation will have a fifty percent chance of rain." This rule would generate the following sequence in simulation:

S R R S R S R R S R S S S R S R S S R S R S S R S S S R R S S S R

# Markov chain

- Markov property: A Markov chain possesses a property that is characterised as “memoryless”: the probability of the next state depends only on the current state and not on the sequence of events that preceded it.

- Markov property defined as:

$$\begin{aligned} \Pr(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n) \\ = Pr(X_{n+1} = x | X_n = x_n) \end{aligned}$$

# Ergodicity

- This means that during numerous iterations, the chain will explore every point (or possible state) and will do so proportionally to its probability.
- To be considered ergodic, the Markov chain must be
  - irreducible: for every state there is a positive probability of moving to any other state
  - aperiodic: the chain must not get trapped in cycles

# Reversibility: reversible chains

- For a stationary distribution $\pi$ and a transition matrix $P$, the reversibility condition can be written as $\pi(x)P(x,y) = \pi(y)P(y,x)$, for all $(x,y) \in S$

# Link to Metropolis-Hastings and Gibbs

In his famous paper from 1953, Metropolis showed how to construct a Markov chain with stationary distribution $\pi$ such that

$$\pi(x) = p_x, x \in S$$

Here the first step to obtaining the stationary distribution of a Markov chain is to prove that the probabilities of a distribution satisfy the reversibility condition.

===

# 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 prior
  - You can only define each variable once!!!

# 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 prior
  - You can only define each variable once!!!
- Different ways to run JAGS from R:
  - rjags, runjags, R2jags, jagsUI
- See http://runjags.sourceforge.net/quickjags.html

A simple JAGS model might look like this:

```
model_definition <- "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
}
"
cat(model_definition, file='basicjags.bug')
```

There are two model statements:

```
# Likelihood part:
Positives ~ dbinom(prevalence, TotalTests)
```

- states that the number of *Positive* test samples is Binomially distributed with probability parameter *prevalence* and total trials *TotalTests*

There are two model statements:

```
# Likelihood part:
Positives ~ dbinom(prevalence, TotalTests)
```

- states that the number of *Positive* test samples is Binomially distributed with probability parameter *prevalence* and total trials *TotalTests*

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

- states that our prior probability distribution for the parameter *prevalence* is Beta(2,2)

The other lines in this model:

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

are automated hooks that are only used by runjags.

To run this model, copy/paste the code above into a new text file called "basicjags.bug" in the same folder as your current working directory. Then run:

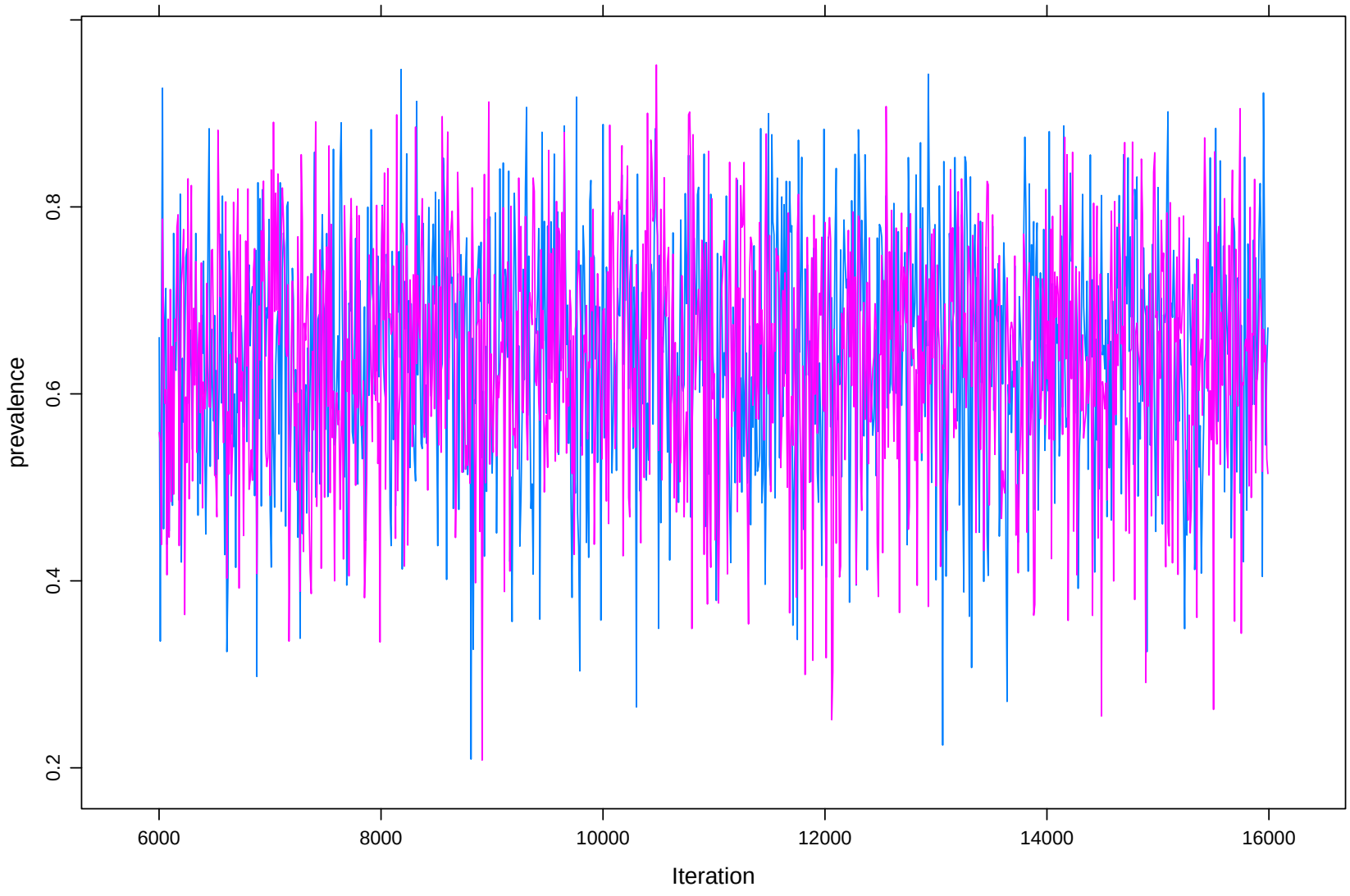
```
library('runjags')

# 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('basicjags.bug', n.chains=2, burnin=5000, sample=10000)
```

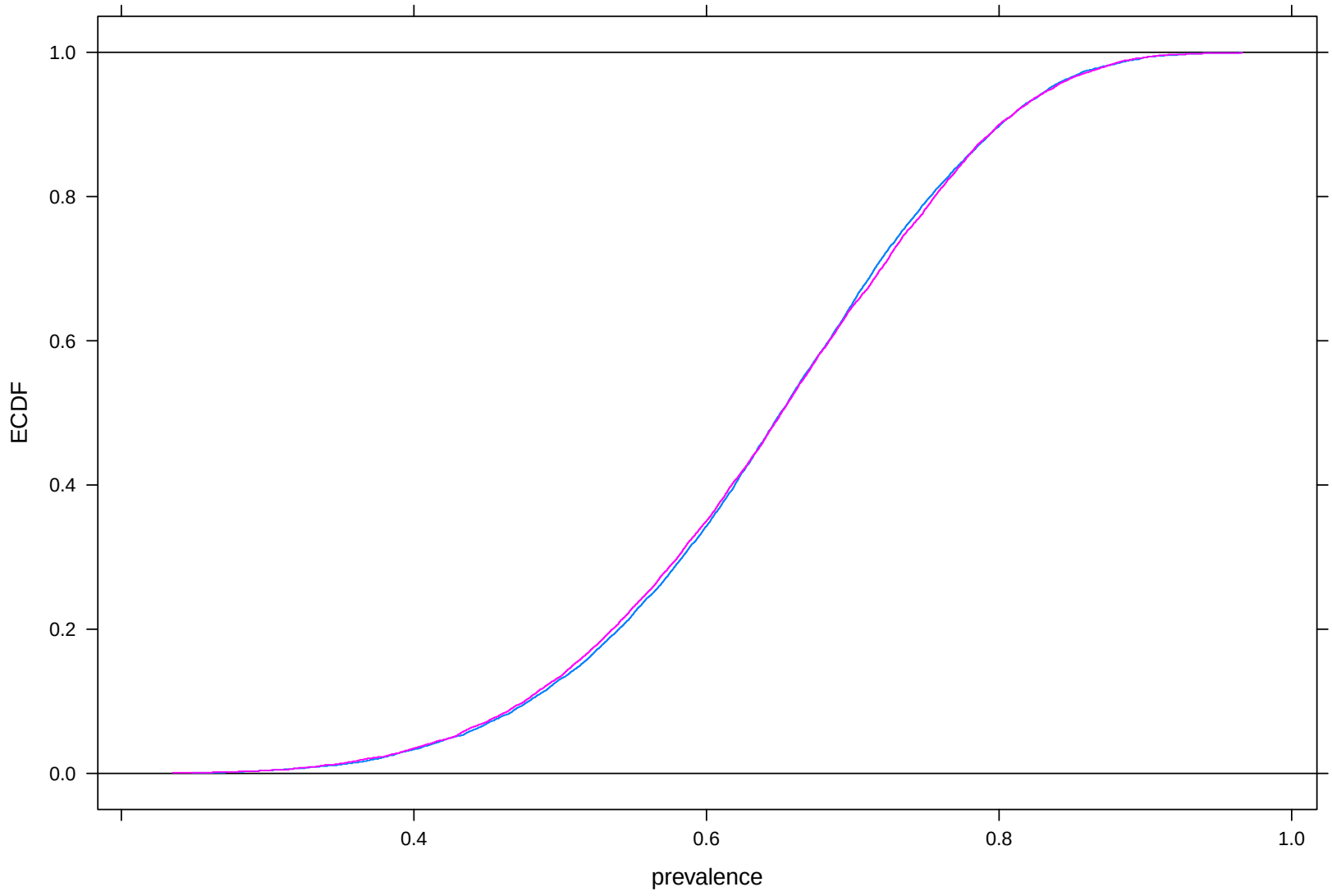
First check the plots for convergence:

```
plot(results)
```

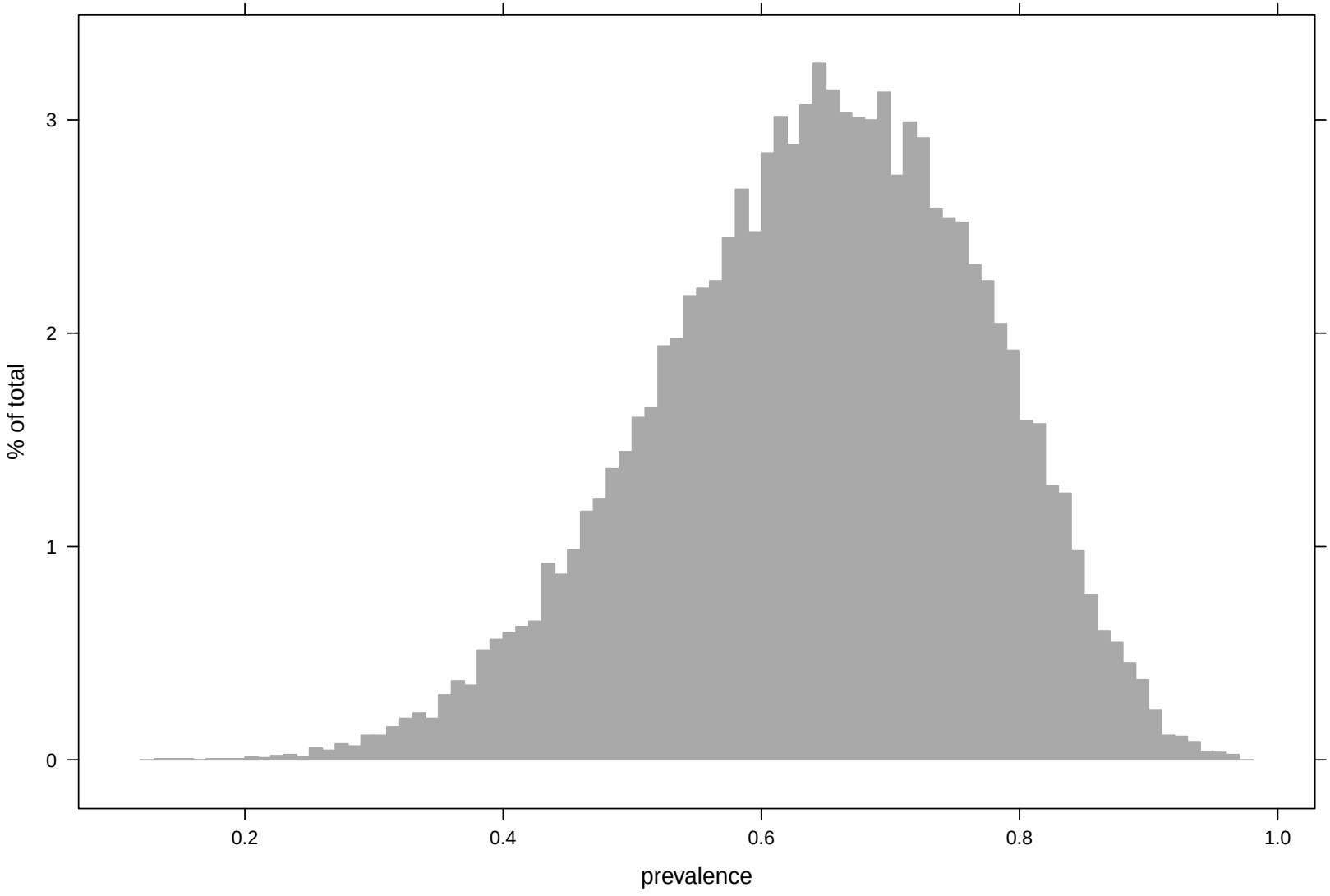
Trace plots: the two chains should be stationary:



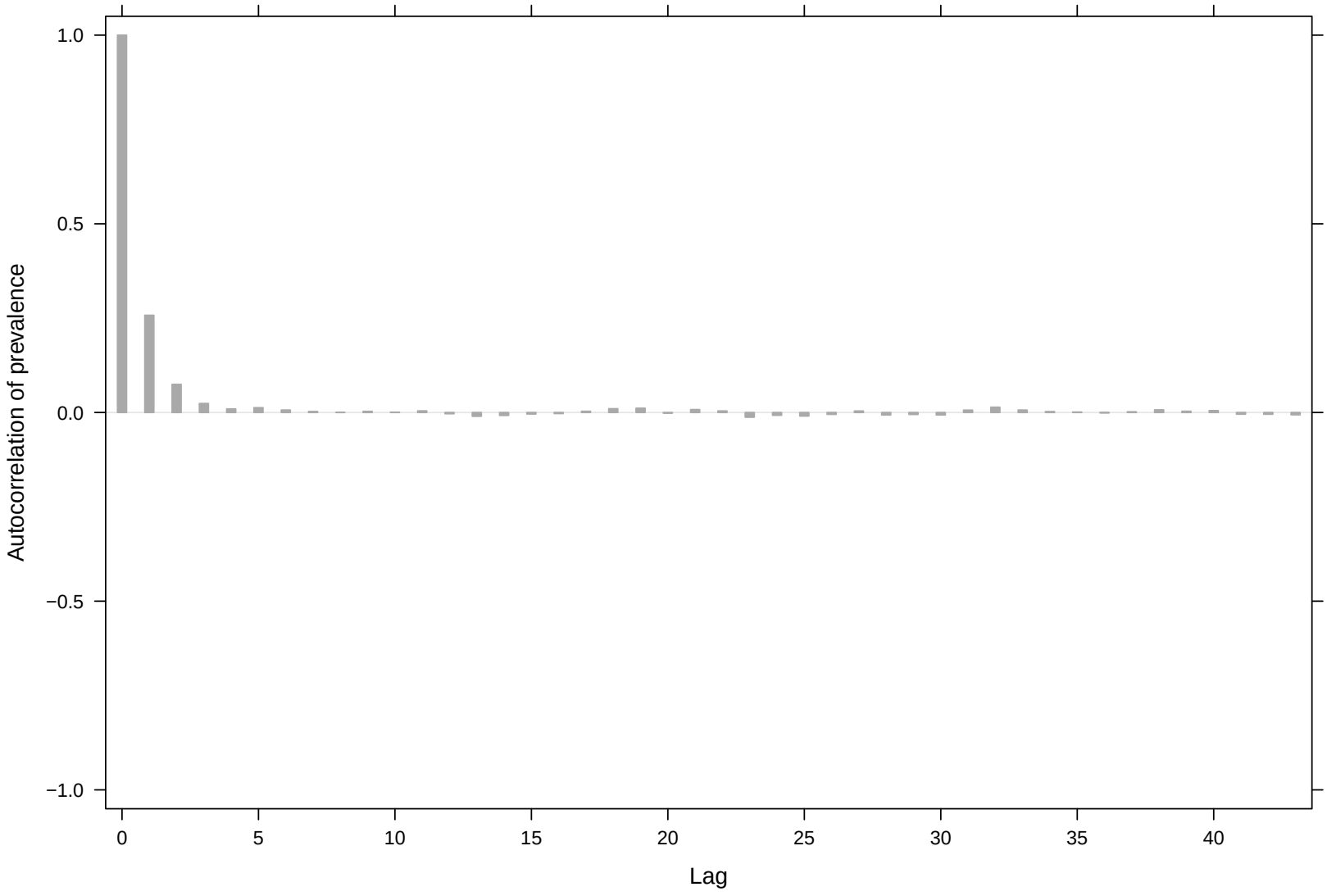
ECDF plots: the two chains should be very close to each other:



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 20000 samples (chains = 2; adapt+burnin =
##
##            Lower95 Median Upper95    Mean      SD Mode     MCerr MC%ofSD SSef
## prevalence 0.40257 0.6508 0.88056 0.64373 0.12469   -- 0.0011478     0.9 1180
##
##                 AC.10 psrf
## prevalence 0.00086095    1
##
## Total time taken: 0.2 seconds
```

Reminder: we want psrf $< 1.05$ and SSeff $> 1000$