Computer Lab 02 — rjags

Contents

1 Introduction

If you are on a system on which you cannot run OpenBUGS but the R package rjags and JAGS is available (e.g. a MAC OSX system or a Linux operating system), then the following shows how to run the two computer lab examples using the rjags package.

2 Douglas firs example with rags

As mentioned in the lecture, JAGS does not have the geometric distribution implemented. But, similarly to Stan, it has an implementation of the negative binomial distribution of which the geometric distribution is a special case. The parameterisation in JAGS is even more intuitive than the one used in Stan; see Plummer (2017, 51ff).

2.1 Flat prior

Assume we have the BUGS code for that example in a file called NegBin-flat.bug:

```
model{
  ## likelihood
  for(i in 1:N){
    y[i] ~ dnegbin(p, 1)
  }

  ## prior
  p ~ dbeta(1, 1)
}
```

Note that this code is using a flat prior for p.

First, we load the package rjags:

```
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

With JAGS, the model checking, data loading and compiling step is done in one step. Hence, we compile our model while at the same time specifying the data by the following command:

```
y.freq <- c(71, 28, 5, 2, 2, 1)
y.obs <- rep(0:5, times = y.freq)
mod <- jags.model("NegBin-flat.bug", data = list(y = y.obs, N = length(y.obs)), n.chains = 4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 109
##    Unobserved stochastic nodes: 1
##    Total graph size: 112
##
## Initializing model
```

If we want to run the chain for 10000 iterations while monitoring `p`, we have to issue the following command:

```
out <- coda.samples(mod, "p", n.iter = 10000)
```

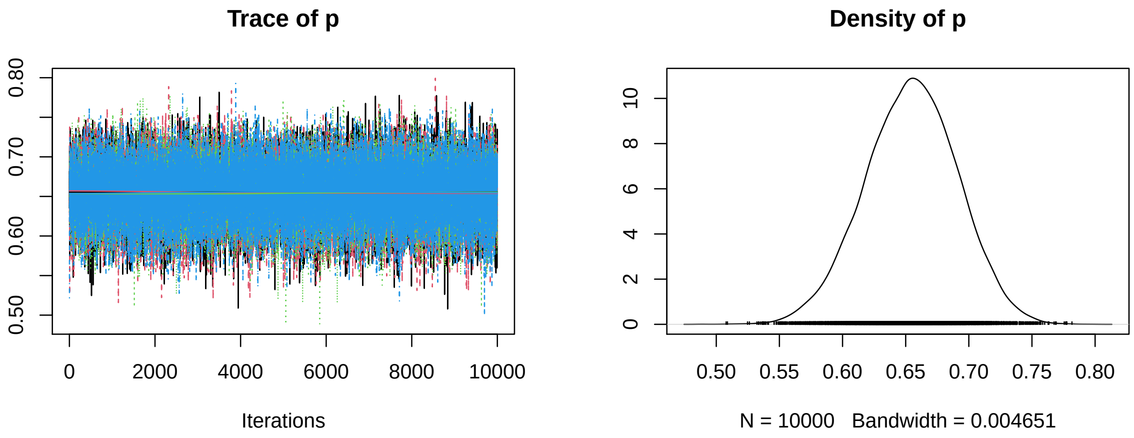
The statistics on each monitored node we can obtain as follows:

```
summary(out)
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##      0.6543773      0.0365327      0.0001827      0.0001791
##
## 2. Quantiles for each variable:
##
##   2.5%    25%    50%    75%  97.5%
## 0.5810 0.6298 0.6550 0.6795 0.7238
```

The trace plot and the empirical densities of the simulated values are produced by the first command in the following code snippet, while the estimated auto-correlation function of the chain is obtained and plotted by the second and third command, respectively:

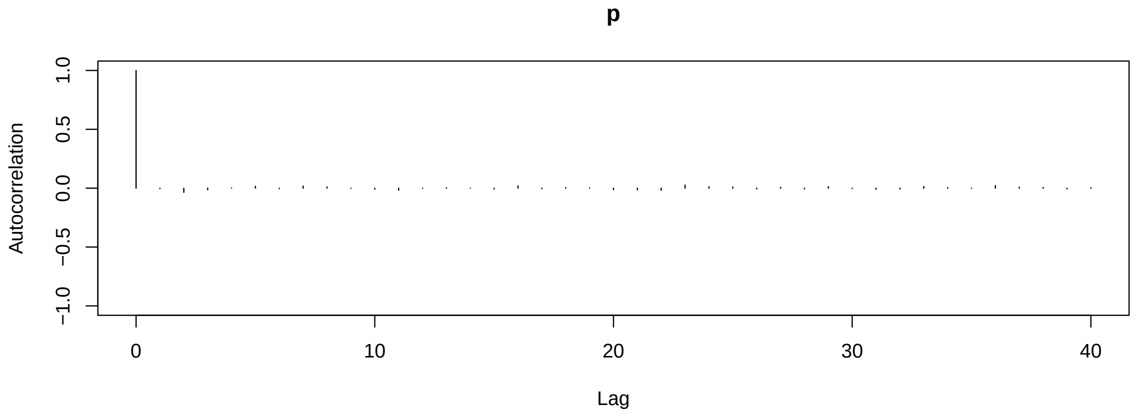
```
plot(out)
```

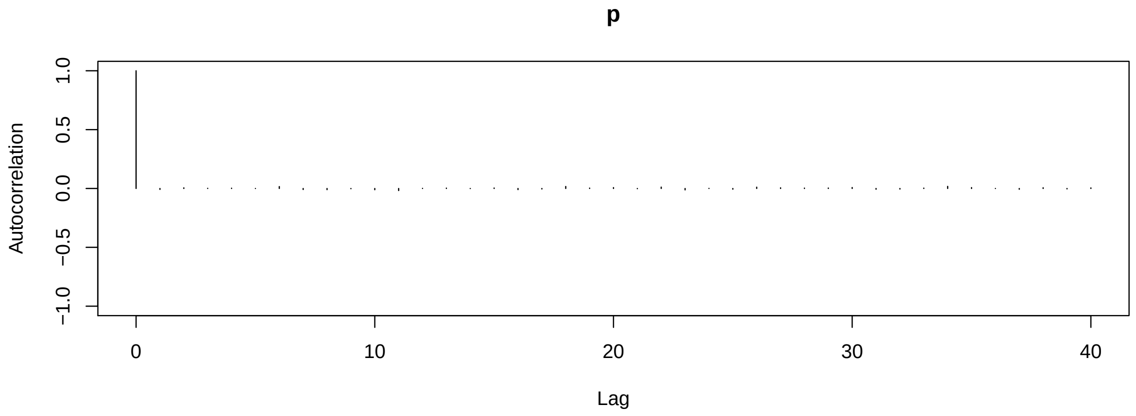


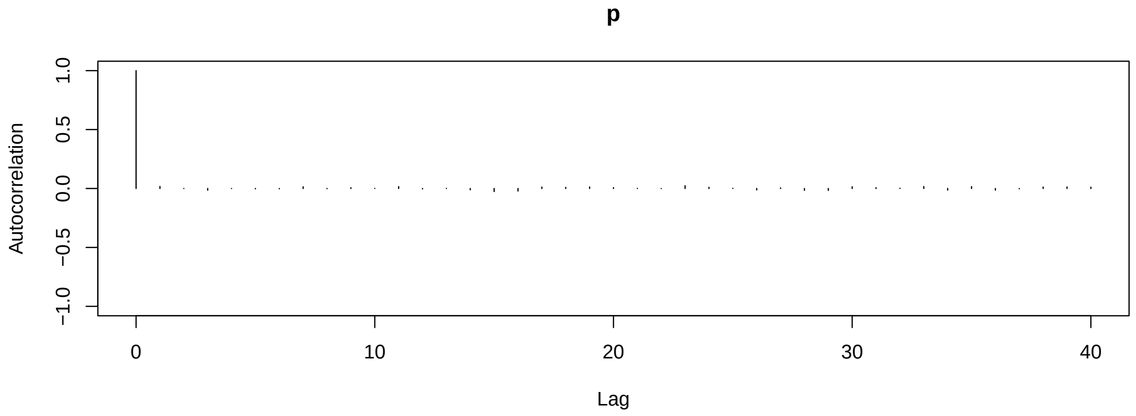
```
autocorr(out)
```

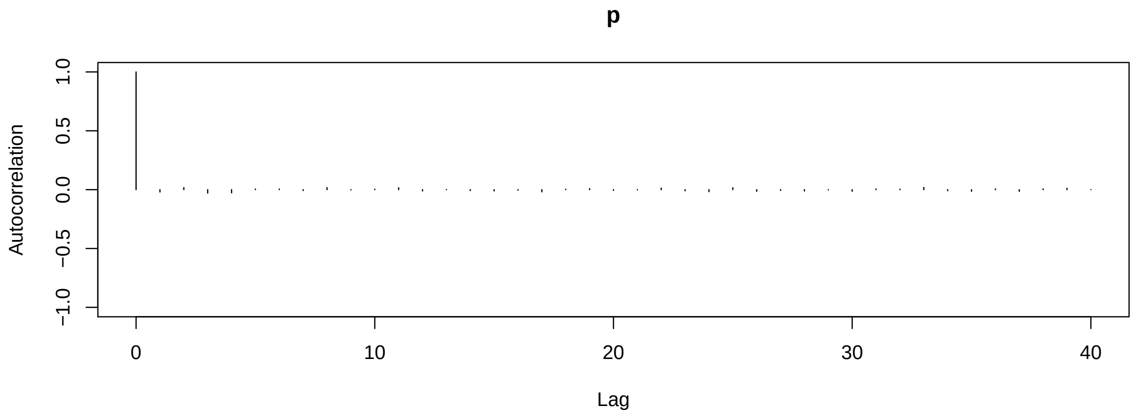
```
## [[1]]
## , , p
##
##                     p
## Lag 0   1.000000000
## Lag 1  -0.003744115
## Lag 5   0.016015233
## Lag 10 -0.009020191
## Lag 50  0.001126380
##
##
## [[2]]
## , , p
##
##                      p
## Lag 0   1.0000000000
## Lag 1  -0.0079385975
## Lag 5  -0.0001409474
## Lag 10 -0.0105601286
## Lag 50  0.0113669256
##
##
## [[3]]
## , , p
##
##                      p
## Lag 0   1.0000000000
## Lag 1   0.0170950467
## Lag 5  -0.0030608317
## Lag 10  0.0008841037
## Lag 50  0.0019768858
##
##
## [[4]]
## , , p
##
##                     p
## Lag 0   1.000000000
## Lag 1  -0.020056424
## Lag 5   0.004758279
## Lag 10  0.003972098
## Lag 50 -0.007391483
```

```
autocorr.plot(out)
```









2.2 Jeffreys' prior

Unfortunately, `JAGS` does not have a `dflat()` improper prior. We have to specify a proper prior, which can be done by using a uniform prior over a large enough region. Thus, assume we have the `BUGS` code for that example in a file called `NegBin-Jeffreys.bug`:

```
model{
  ## likelihood
  for(i in 1:N){
    y[i] ~ dnegbin(p, 1)
  }

  ## prior
  u ~ dunif(0, 1000)
  p <- 1 - pow( (1-exp(-u))/(1+exp(-u)), 2)
}
```

After that, we follow pretty much the same steps as before. Though, if we have a prior on `u` that is uniform on $[0, 1000]$ we might experience problems with `JAGS` not being able to produce samples as one or more of the chains start at bad initial values. Hence, here we also specify initial values, again via a list of lists:

```
data.in <- list(y = y.obs, N = length(y.obs))
inits <- list(list(u = 1), list(u = 2), list(u = 4), list(u = 6))
mod <- jags.model("NegBin-Jeffreys.bug", data = data.in, inits = inits, n.chains = 4)
```

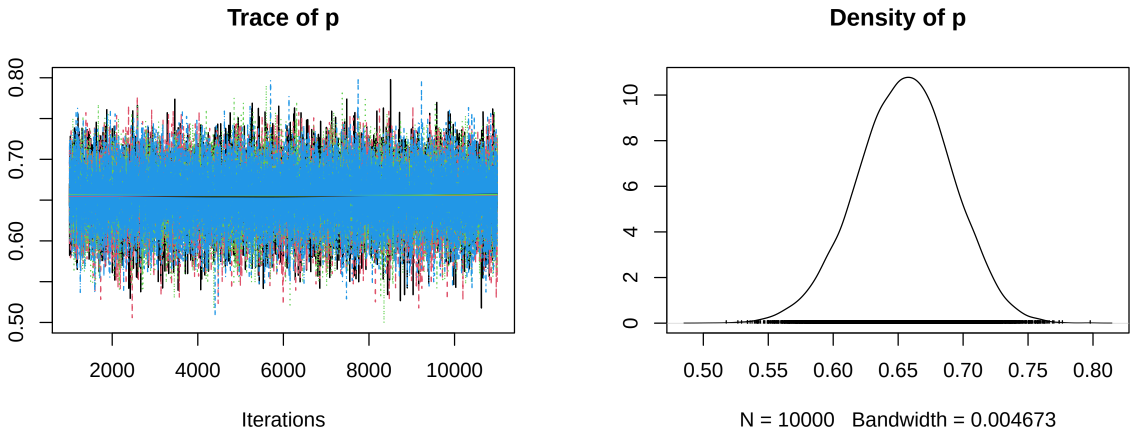
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 109
##    Unobserved stochastic nodes: 1
##    Total graph size: 122
##
## Initializing model
```

```
out <- coda.samples(mod, "p", n.iter = 10000)
summary(out)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 10000
```

```
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##      0.6549219      0.0367039      0.0001835      0.0002375
##
## 2. Quantiles for each variable:
##
##   2.5%    25%    50%    75%  97.5%
## 0.5811 0.6306 0.6556 0.6798 0.7247
```

```
plot(out)
```

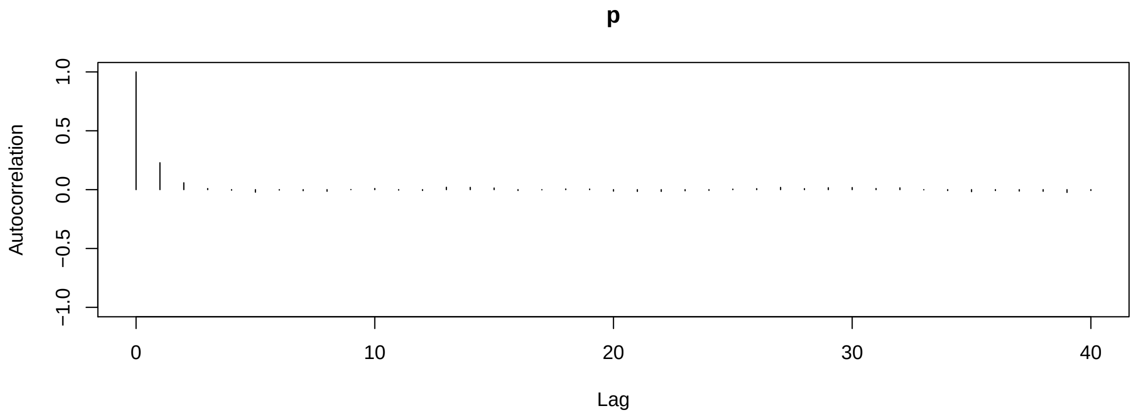


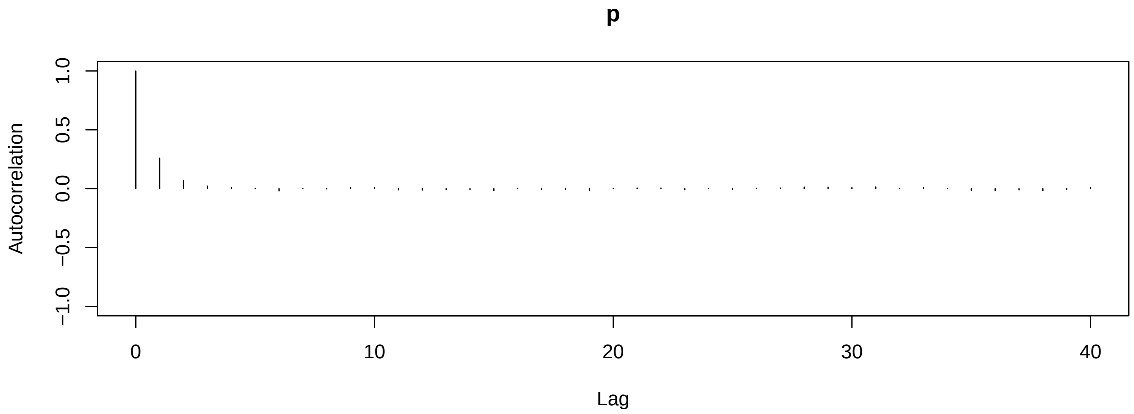
```
autocorr(out)
```

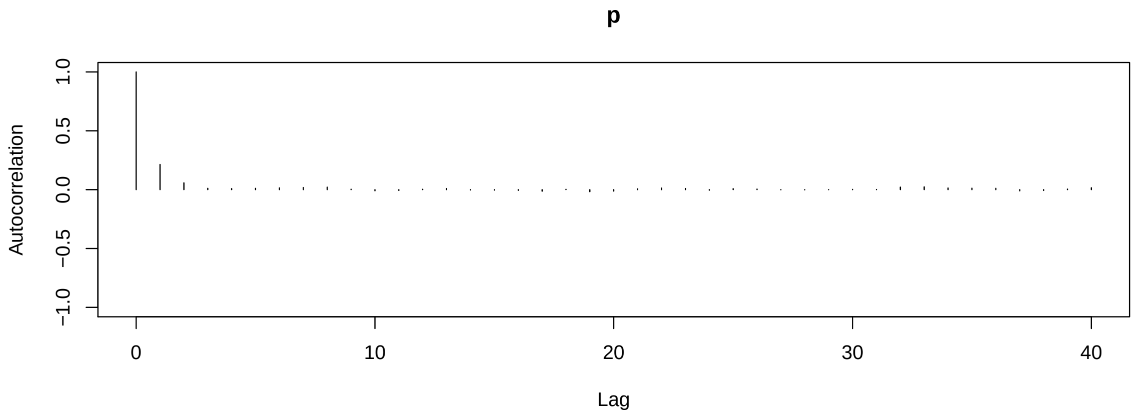
```
## [[1]]
## , , p
##
##                    p
## Lag 0   1.000000000
## Lag 1   0.228623859
## Lag 5  -0.021637079
## Lag 10  0.009864748
## Lag 50 -0.000977647
##
##
## [[2]]
## , , p
##
##                    p
## Lag 0   1.000000000
## Lag 1   0.259649468
## Lag 5   0.002031170
## Lag 10  0.008609066
## Lag 50 -0.012458566
##
##
## [[3]]
## , , p
##
```

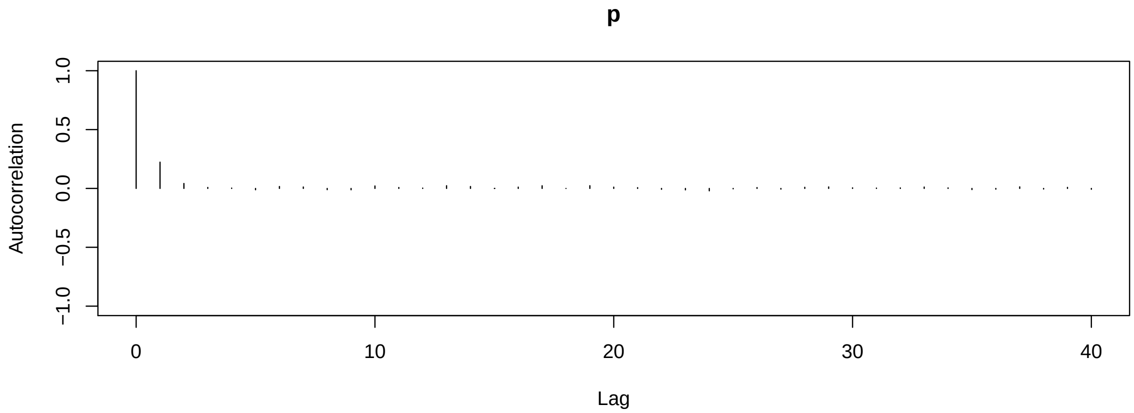
```
##                    p
## Lag 0   1.000000000
## Lag 1   0.213939160
## Lag 5   0.011143763
## Lag 10 -0.009890456
## Lag 50 -0.007617392
##
##
## [[4]]
## , , p
##
##                   p
## Lag 0   1.00000000
## Lag 1   0.22338866
## Lag 5  -0.01119032
## Lag 10  0.02130210
## Lag 50  0.01671218
```

```
autocorr.plot(out)
```









References

Plummer, M. 2017. *JAGS Version 4.3.0 User Manual*. https://sourceforge.net/projects/mcmc-jags/files/Manuals/4.x.