deBInfer logistic ODE example

Philipp H Boersch-Supan

2016-06-02

Preliminaries

First we install the deBInfer package. For this you need to install and load the devtools package first. You can do this from CRAN.

```r
install.packages("devtools")
```

```r
#Load the devtools package.
library(devtools)
```

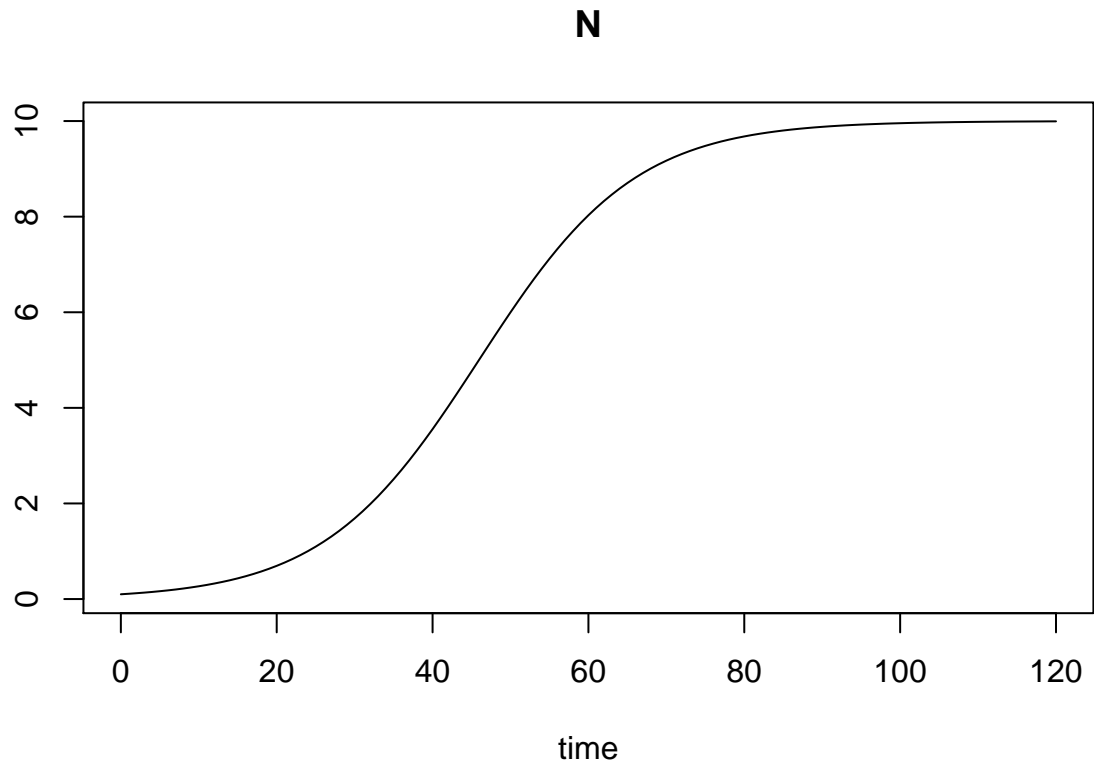Then you install deBInfer from github

```r
install_github("pboesu/debinfer")
```

Defining the DE model

We define a logistic growth ODE for deSolve.

```r
library(deSolve)
logistic_model <- function (time, y, parms) {
with(as.list(c(y, parms)), {
dN <- r * N * (1 - N / K)
list(dN)
})
}
y <- c(N = 0.1)
parms <- c(r = 0.1, K = 10)
times <- seq(0, 120, 1)
out <- ode(y, times, logistic_model, parms, method='lsoda')
```

Which gives us the numerical solution

N

Simulationg observations

Now we simulate a noisy dataset from this equation. We sample a random subset from the integration output

```r
set.seed(143)
N_obs <- as.data.frame(out[c(1,runif(35, 0, nrow(out))),]) #force include the first time-point (t=0)
```

and we "add" lognormal noise

```r
# add lognormal noise
  parms['loglogsd.N'] <- -4.6
  N_obs$N_noisy <- rlnorm(nrow(N_obs), log(N_obs$N),exp(parms['loglogsd.N']))
#observations must be ordered for solver to work
N_obs <- N_obs[order(N_obs$time),]

plot(N_obs$time, N_obs$N, ylim=c(0, max(N_obs$N,N_obs$N_noisy)))
points(N_obs$time, N_obs$N_noisy, col="red")
```
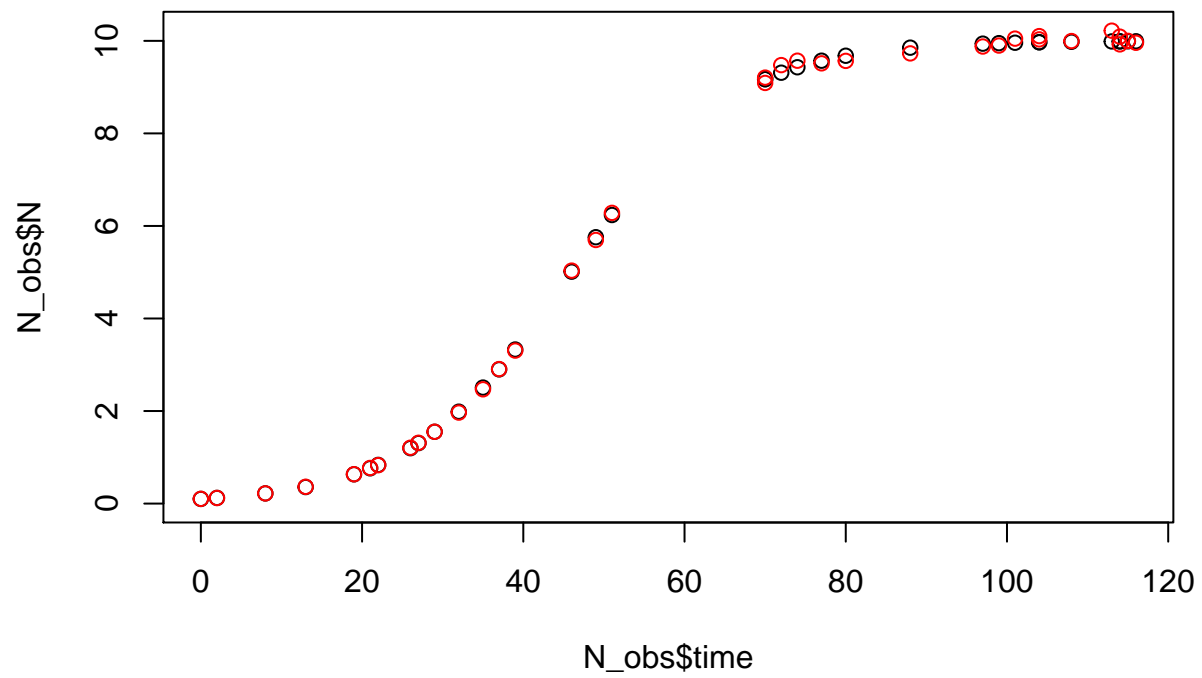
```
out_obs <- ode(y, c(0,N_obs$time), logistic_model, parms, method='lsoda')
plot(out_obs)
lines(out)
```

N
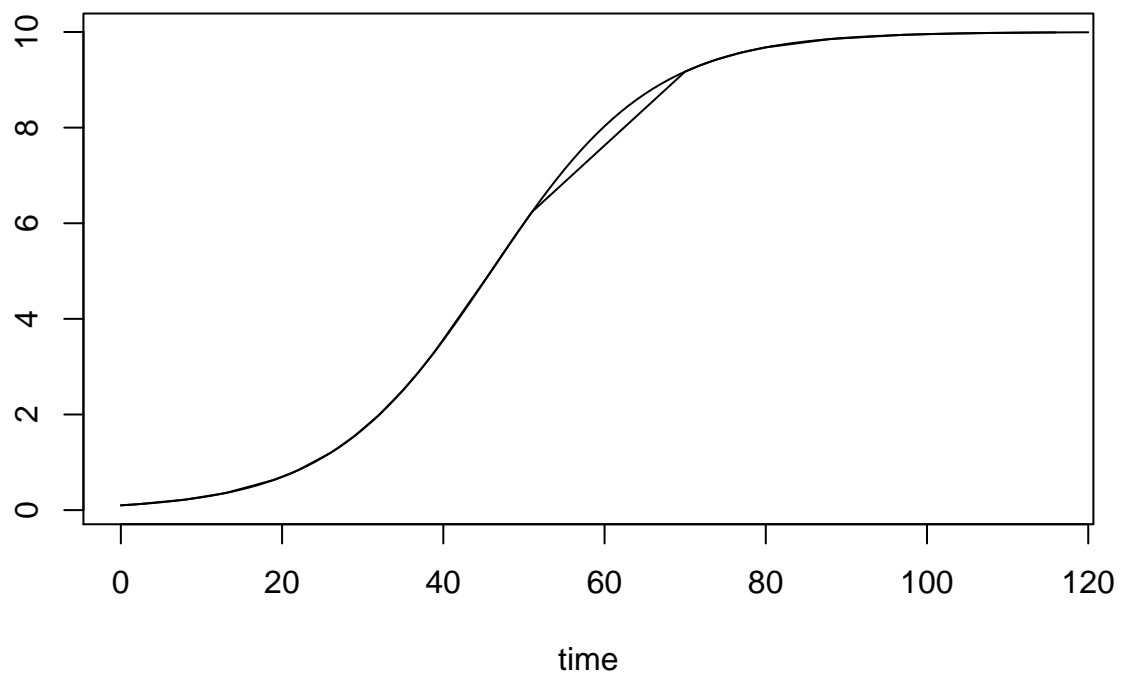
Defining an observation model and parameters for inference

We define an observation model. Note that we are sampling the log of the observation standard deviation, to ensure sampled values are strictly positive. We also use an epsilon correction for the meanlog, as the DE model can return values of 0 (or even less due to numerical precision).

```r
# the observation model
logistic_obs_model<-function(data, sim.data, samp){

  llik.N<-sum(dlnorm(data$N_noisy, meanlog=log(sim.data[,"N"] + 1e-6),
                    sdlog=exp(samp[['loglogsd.N']]), log=TRUE)
              )

  llik<-llik.N

  return(llik)
}
```

We declare the parameters for inference:

```r
library(deBInfer)
r <- debinfer_par(name = "r", var.type = "de", fixed = FALSE,
                 value = 0.5, prior="norm", hypers=list(mean = 0, sd = 1),
                 prop.var=0.005, samp.type="rw")

K <- debinfer_par(name = "K", var.type = "de", fixed = FALSE,
                 value = 5, prior="lnorm", hypers=list(meanlog = 1, sdlog = 1),
                 prop.var=0.1, samp.type="rw")

loglogsd.N <- debinfer_par(name = "loglogsd.N", var.type = "obs", fixed = FALSE,
                 value = -2, prior="norm", hypers=list(mean = 0, sd = 1),
                 prop.var=0.5, samp.type="rw")
```

and we also need to provide an initial condition for the differential equation:

```r
N <- debinfer_par(name = "N", var.type = "init", fixed = TRUE, value = 0.1)
```

All declared parameters are collated using the setup_debinfer function

```r
mcmc.pars <- setup_debinfer(r, K, loglogsd.N, N)
```
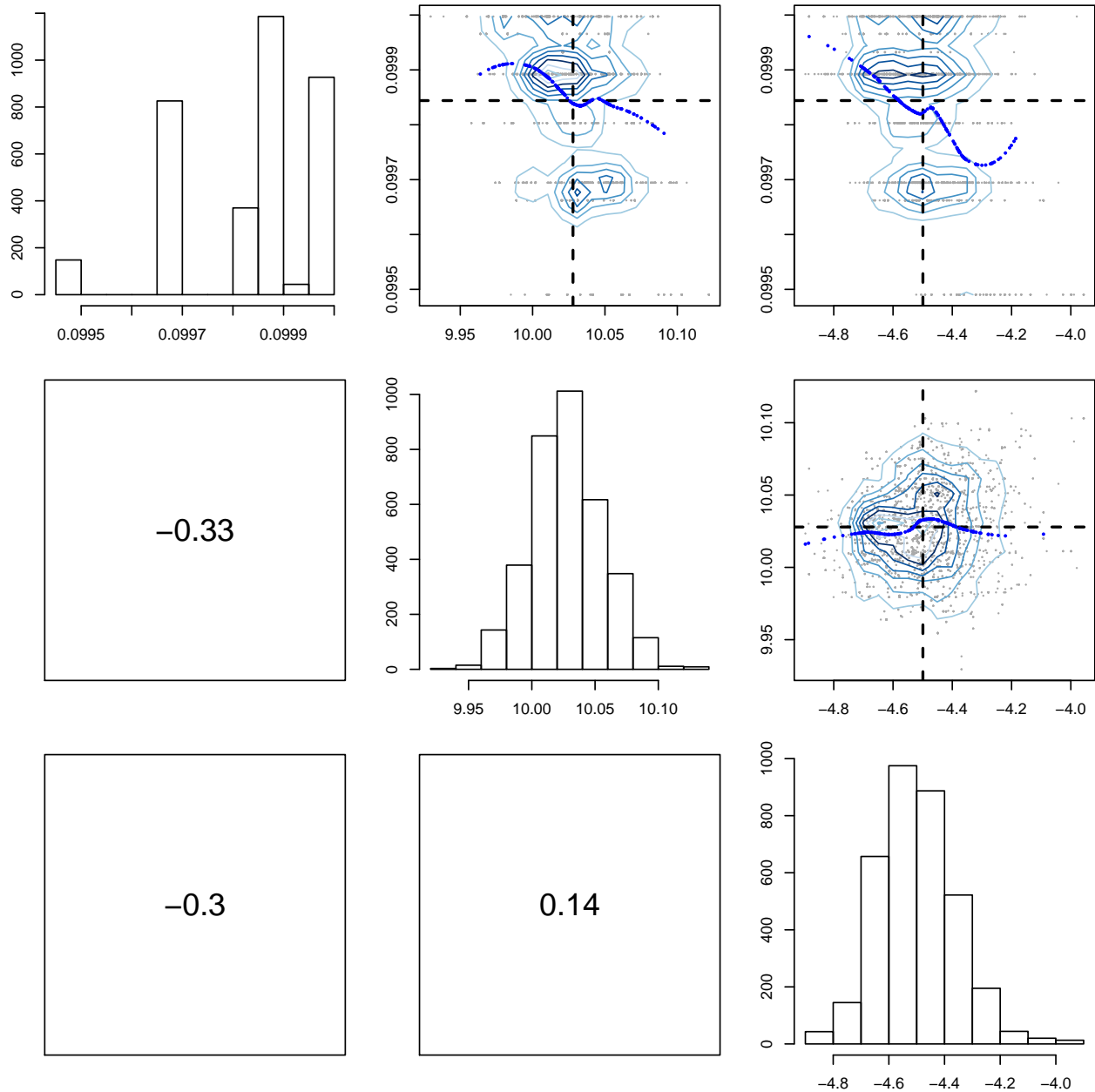
Conduct inference

Finally we use deBInfer to estimate the parameters of the original model.

```r
# do inference with deBInfer
  # MCMC iterations
  iter = 5000
  # inference call
  mcmc_samples <- de_mcmc(N = iter, data=N_obs, de.model=logistic_model,
```
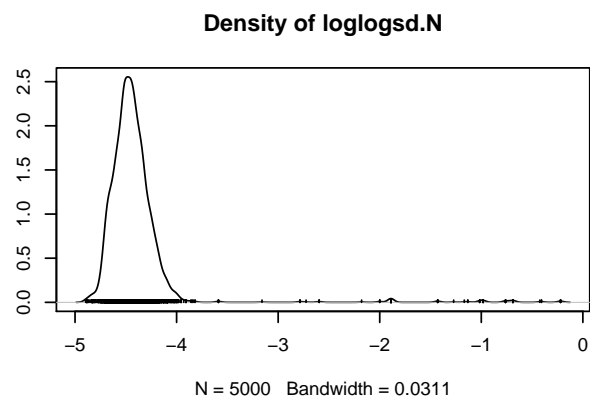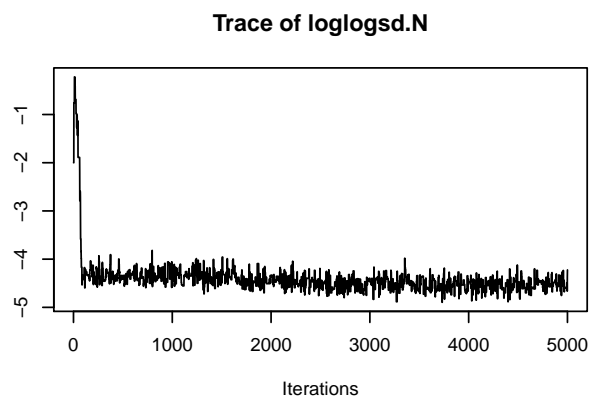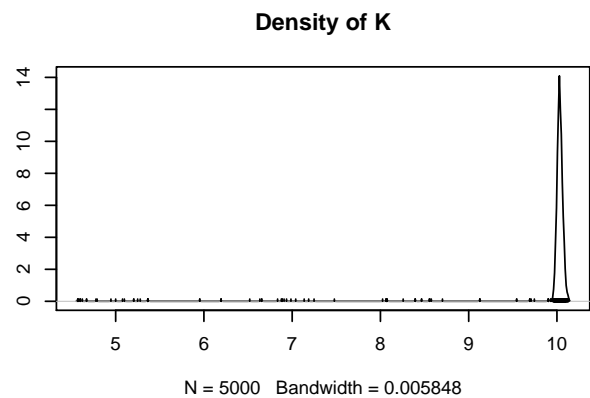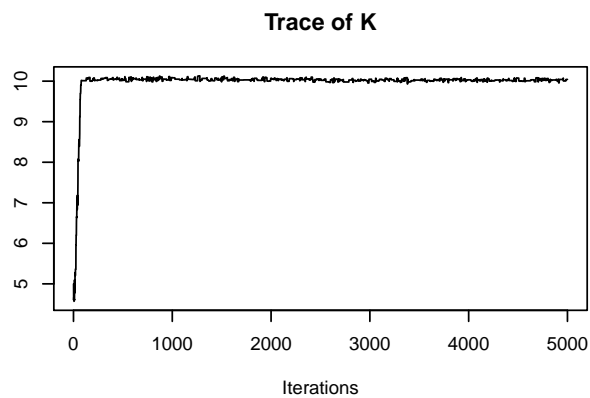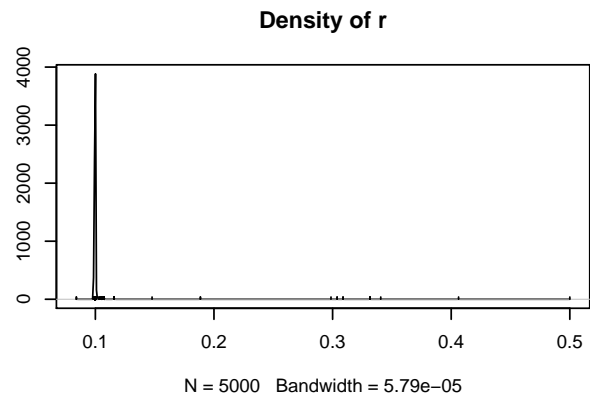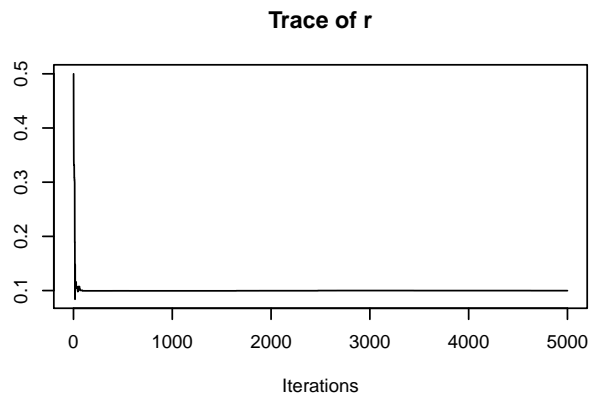
```
                     obs.model=logistic_obs_model, all.params=mcmc.pars,
                     Tmax = max(N_obs$time), data.times=N_obs$time, cnt=iter,
                     plot=FALSE, sizestep=0.1, solver="ode")
```

We plot the results

```
burnin = 1500
pairs(mcmc_samples, burnin = burnin, scatter=TRUE, trend=TRUE)
```



```
plot(mcmc_samples$samples)
```

Trace of r

Iterations

Density of r

N = 5000   Bandwidth = 5.79e−05

Trace of K

Iterations

Density of K

N = 5000   Bandwidth = 0.005848

Trace of loglogsd.N

Iterations

Density of loglogsd.N

N = 5000   Bandwidth = 0.0311

```r
summary(mcmc_samples$samples)
```

```
##
## Iterations = 1:5000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean      SD  Naive SE Time-series SE
```

```
## r           0.1004 0.01231 0.0001741        0.0006961
## K           9.9874 0.42553 0.0060179        0.0457233
## loglogsd.N -4.4080 0.40749 0.0057628        0.0577335
##
## 2. Quantiles for each variable:
##
##                  2.5%      25%     50%      75%  97.5%
## r           0.09946  0.09949  0.0998  0.09989  0.100
## K           9.96797 10.01091 10.0312 10.05152 10.096
## loglogsd.N -4.72698 -4.56125 -4.4567 -4.34532 -4.028
```

```r
#plot(N_obs$time,N_obs$N_noisy)

posterior.mean.sim <- solve_de(logistic_model, params=c(r=mean(mcmc_samples$samples[,"r"][burnin:iter])
plot(posterior.mean.sim)
points(N_obs$time,N_obs$N_noisy)
```

N

time
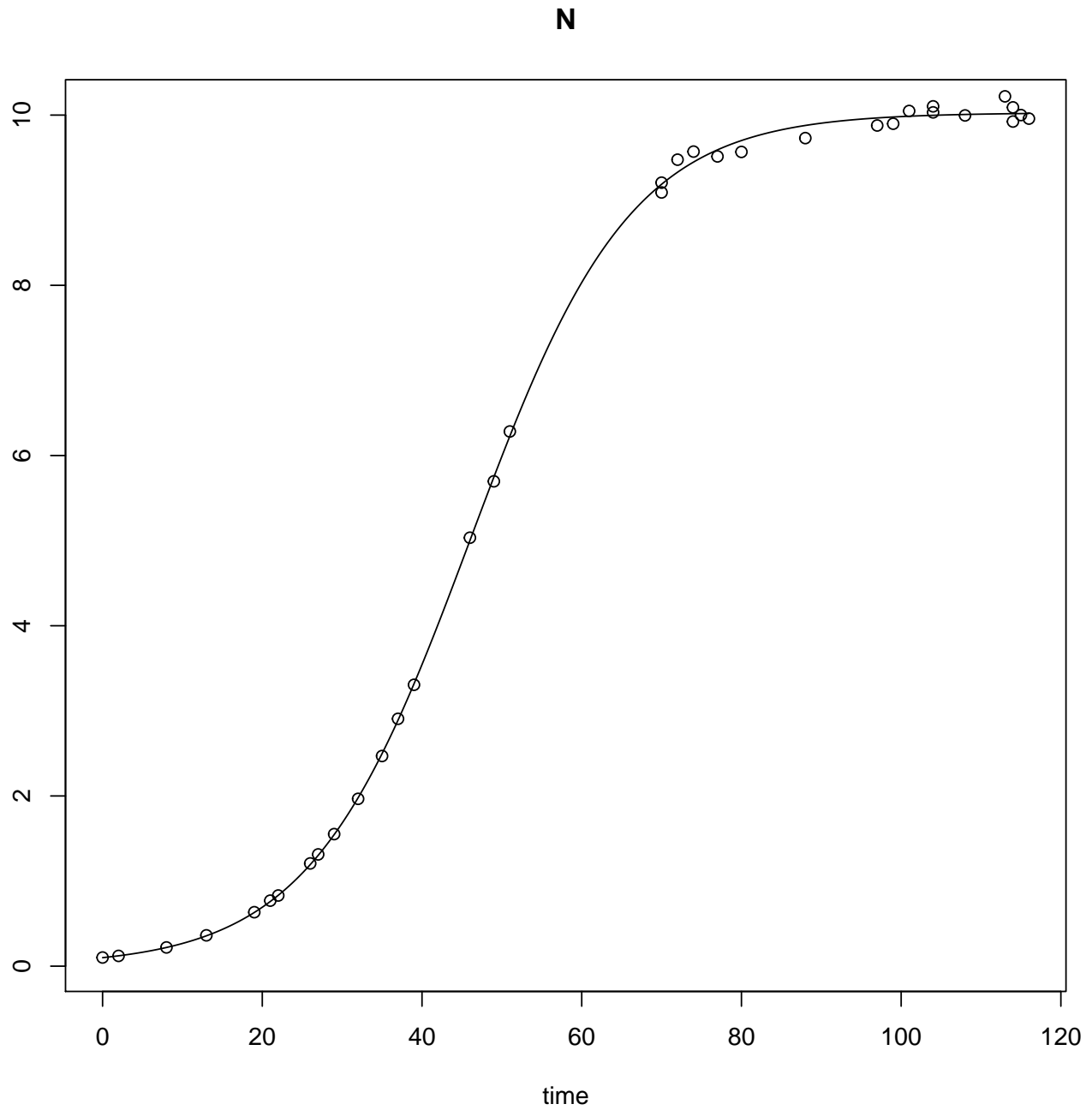
Posterior credible interval (equal-tailed)

```r
library(plyr)
simlist <- llply(sample(nrow(mcmc_samples$samples[burnin:iter,]), 1000),
                 function(x)solve_de(logistic_model, mcmc_samples$samples[burnin:iter,][x,],
                                     inits=c(N = 0.1), Tmax=120, numsteps=100))

sima <- simplify2array(simlist)
dim(sima)
```

```
## [1] 100    2 1000
```

```
simCI <- aaply(sima, .margins = c(1,2), quantile, probs=c(0.025,0.5,0.975))
for (p in 2:ncol(simlist[[1]])){
  plot(simCI[,,3][,c(1,p)], main="", type='n', ylim=c(0,11))

  #for (i in 1:length(simlist)){
  #  lines(simlist[[i]][,c(1,p)],lty=2,col='grey')
  #}
  for (i in 1:3){
    lines(simCI[,,i][,c(1,p)], lty=c(2,1,2)[i], col=c("red"))
  }
  if (dimnames(simlist[[1]])[[2]][p] == "N") points(N_obs$time,N_obs$N_noisy)
}
```
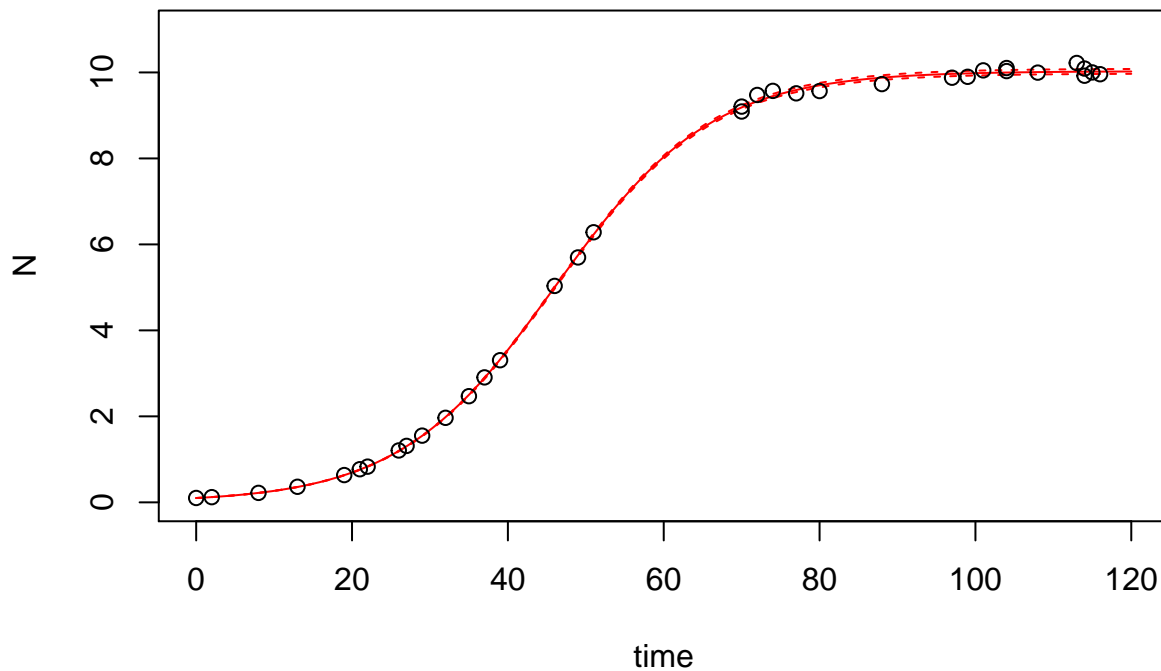


```
#str(out)
#lines(posterior.mean.sim, ylim=c(0,15))

#l_ply(simlist, function(x)  lines(x[,1], x[,2]))
#lines(out[,1],out[,2], col='blue')


#we could also use  asymmetric uniform proposals to ensure only positive values of logsd.N are sampled


logsd.Nunif <-  debinfer_par(name = "logsd.N", var.type = "obs", fixed = FALSE,
           value = 0.01, prior="norm", hypers=list(mean = 0, sd = 1),
           prop.var=c(1,2), samp.type="rw-unif")
```

```
## Warning in if (!is.numeric(prop.var) | prop.var < 0 | length(prop.var) != :
## the condition has length > 1 and only the first element will be used
```

```
#in that case we need to adjust the observation model slightly
```

```
# the observation model
  logistic_obs_model_unif<-function(data, sim.data, samp){

    llik.N<-sum(dlnorm(data$N_noisy, meanlog=log(sim.data[,"N"] + 1e-6),
                       sdlog=samp[['logsd.N']], log=TRUE)
               )

    llik<-llik.N

    return(llik)
  }

  mcmc.pars_unif <- setup_debinfer(r, K, logsd.Nunif, N)

  mcmc_samples_unif <- de_mcmc(N = iter, data=N_obs, de.model=logistic_model,
                          obs.model=logistic_obs_model_unif, all.params=mcmc.pars_unif,
                          Tmax = max(N_obs$time), data.times=N_obs$time, cnt=iter %/% 10,
                          plot=TRUE, sizestep=0.1, solver="ode")
```

```
## [1] "Order of initial conditions is  N"
## [1] "initial posterior probability = -197325.858568254"
## [1] "proposing logsd.N: prob.old = 63.174; prob.new = 56.276; A = 0.0013931"
## [1] "proposing K: prob.old = 63.174; prob.new = -235.39; A = 2.1667e-130"
## [1] "proposing r: prob.old = 63.174; prob.new = -820.95; A = 0"
## [1] "sample number 500"
```

Trace of r

Trace of K

Trace of logsd.N

Trace of N

Trace of lpost
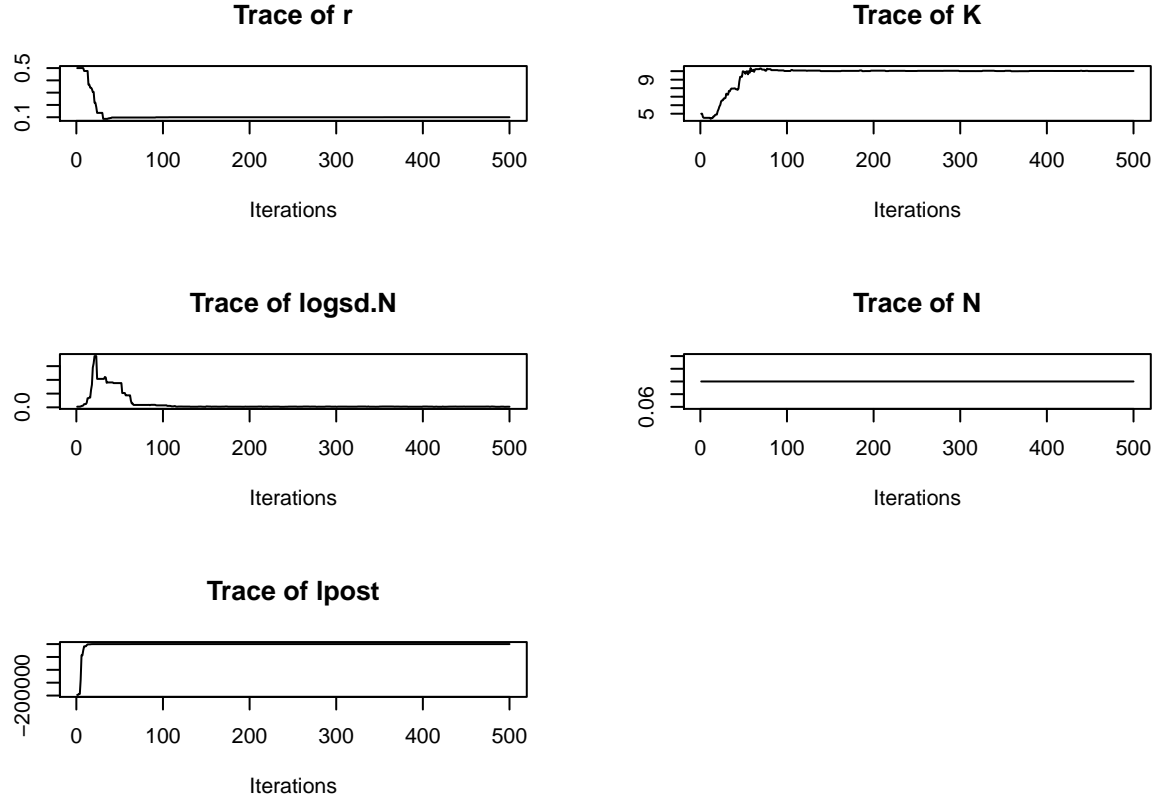

```
## [1] "proposing r: prob.old = 59.537; prob.new = -1131.6; A = 0"
```

```
## [1] "proposing logsd.N: prob.old = 59.537; prob.new = 53.965; A = 0.0024396"
## [1] "proposing K: prob.old = 59.537; prob.new = 2.3222; A = 1.4189e-25"
## [1] "sample number 1000"
```

Trace of r

Trace of K

Trace of logsd.N

Trace of N


Trace of lpost
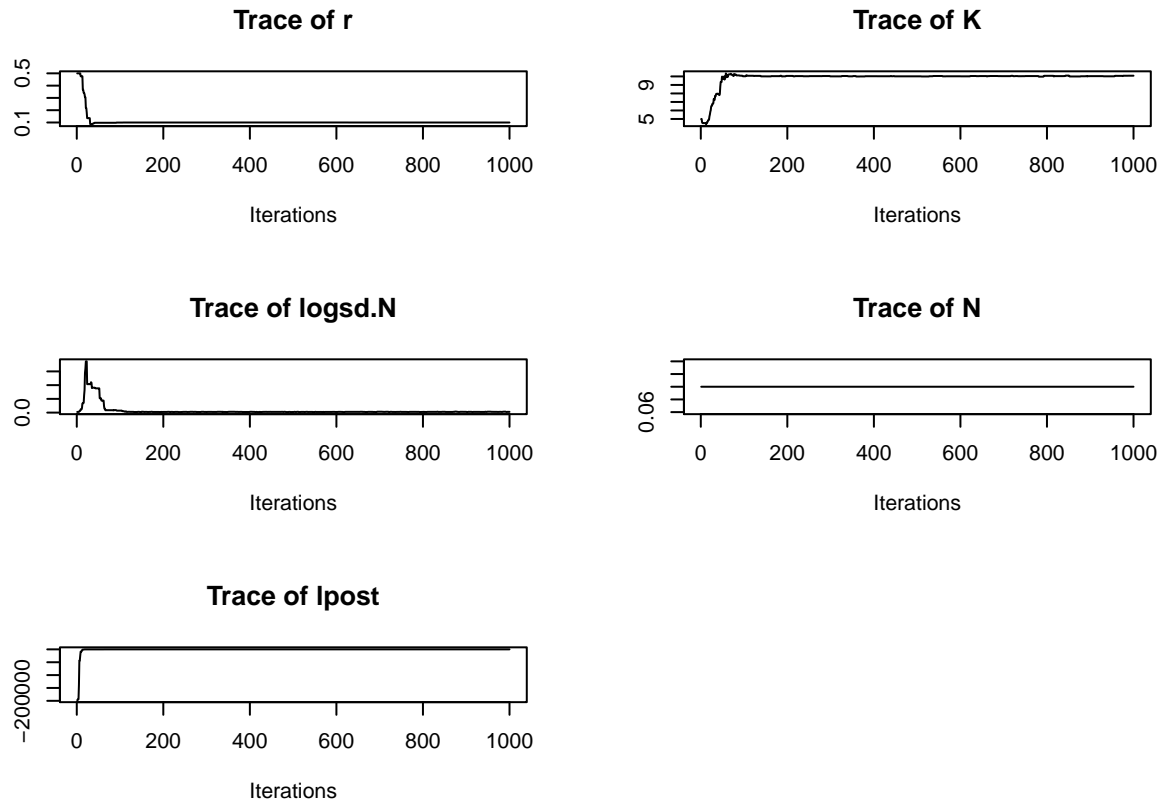

```
## [1] "proposing K: prob.old = 60.505; prob.new = 17.99; A = 3.4359e-19"
## [1] "proposing logsd.N: prob.old = 60.505; prob.new = 52.636; A = 0.00019229"
## [1] "proposing r: prob.old = 60.505; prob.new = -333820; A = 0"
## [1] "sample number 1500"
```
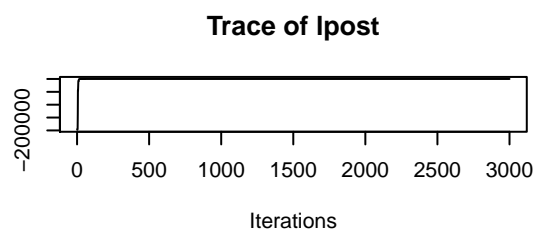
Trace of r

Trace of K

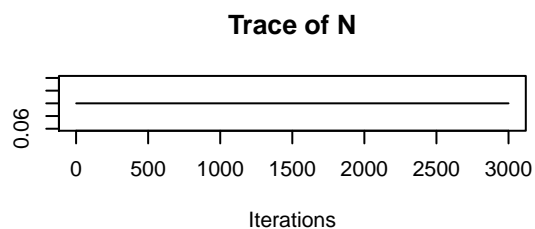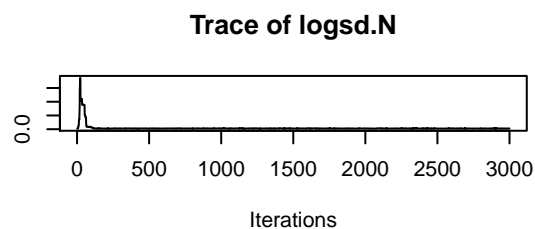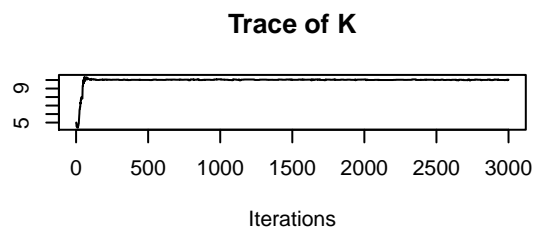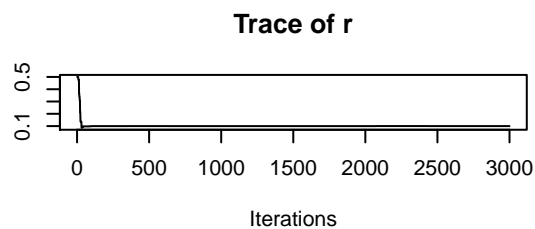**Trace of logsd.N**

Trace of N

Trace of Ipost
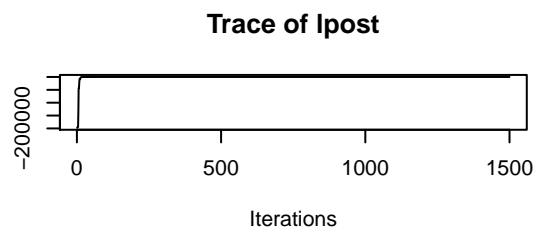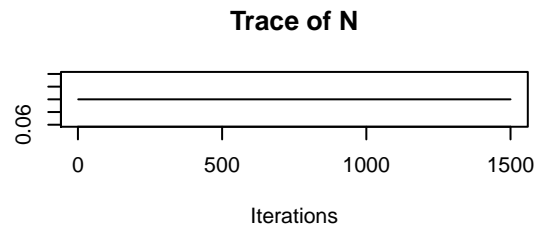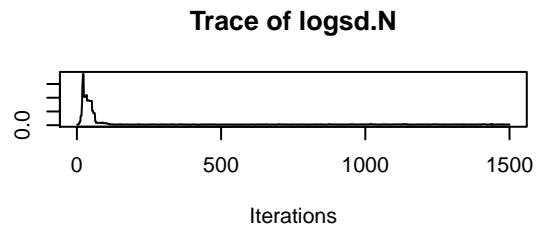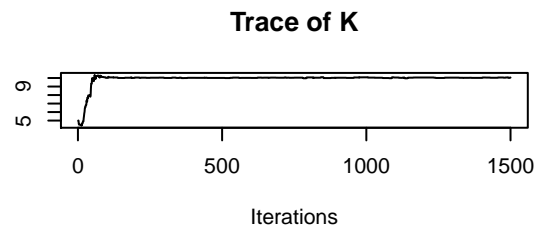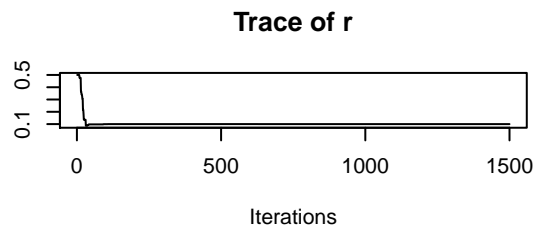
```
## [1] "proposing r: prob.old = 61.462; prob.new = -92176; A = 0"
## [1] "proposing K: prob.old = 61.462; prob.new = 60.9; A = 0.57048"
## [1] "proposing logsd.N: prob.old = 60.9; prob.new = 53.659; A = 0.00047369"
## [1] "sample number 2000"
```

Trace of r

Trace of K

**Trace of logsd.N**

Trace of N

Trace of Ipost
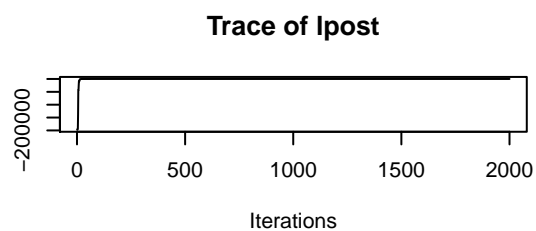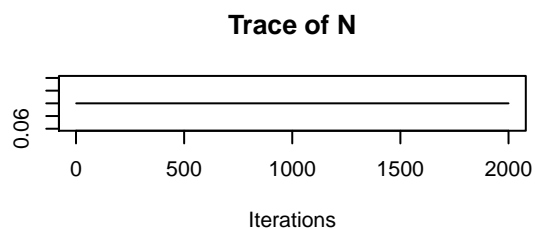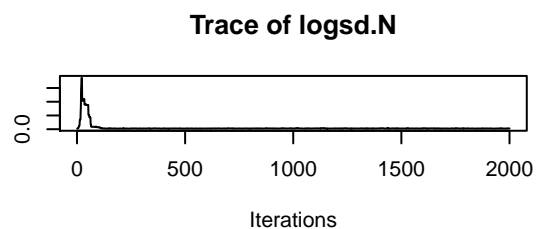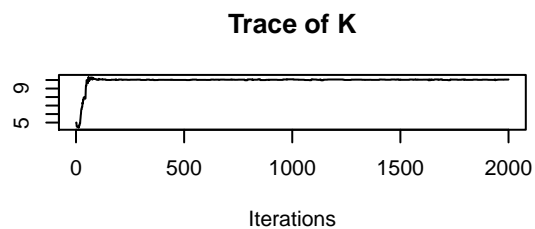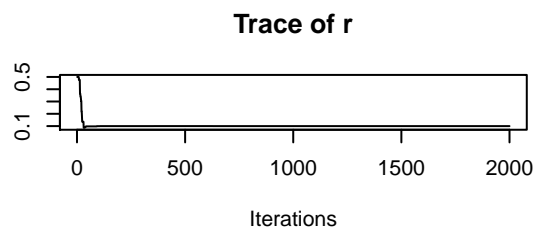
```
## [1] "proposing r: prob.old = 63.489; prob.new = -103.31; A = 3.6221e-73"
## [1] "proposing K: prob.old = 63.489; prob.new = -81.159; A = 1.5138e-63"
## [1] "proposing logsd.N: prob.old = 63.489; prob.new = 63.478; A = 0.99257"
## [1] "sample number 2500"
```

Trace of r

Trace of K

Trace of logsd.N

Trace of N

Trace of lpost
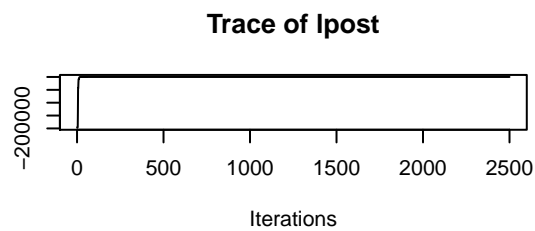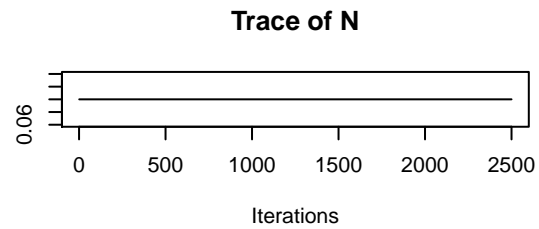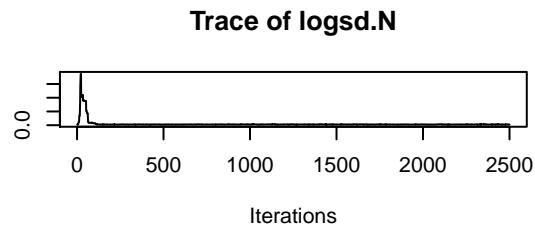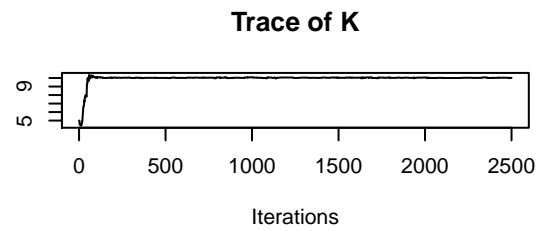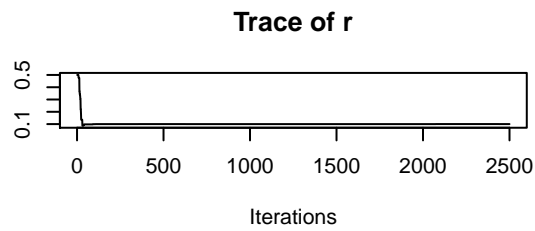

```
## [1] "proposing logsd.N: prob.old = 63.079; prob.new = 50.212; A = 1.3109e-06"
## [1] "proposing K: prob.old = 63.079; prob.new = 62.472; A = 0.54506"
## [1] "proposing r: prob.old = 63.079; prob.new = -32227; A = 0"
## [1] "sample number 3000"
```
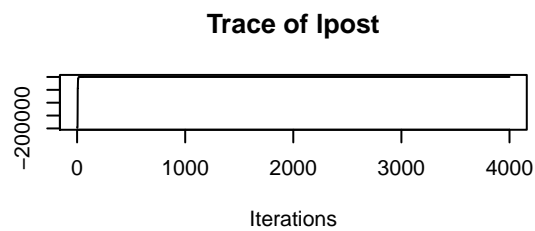
Trace of r

Iterations

Trace of K

Iterations

Trace of logsd.N

Iterations

Trace of N
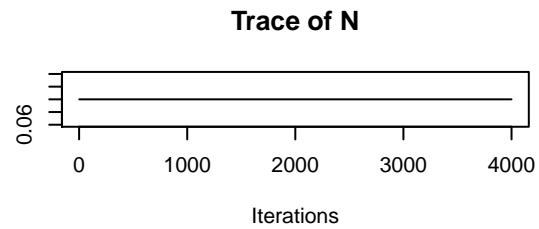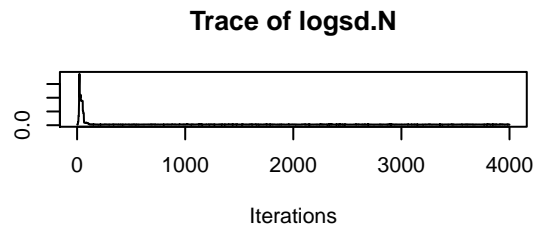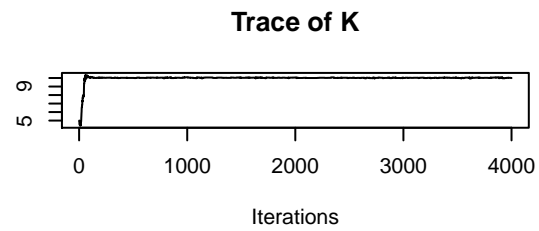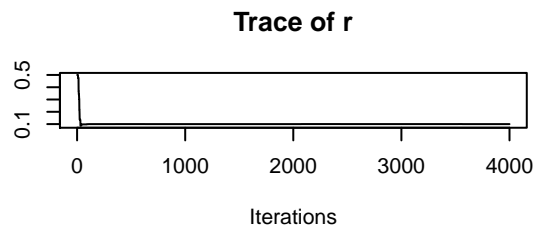

Iterations

Trace of lpost

Iterations

```
## [1] "proposing K: prob.old = 63.446; prob.new = 50.537; A = 2.4756e-06"
## [1] "proposing logsd.N: prob.old = 63.446; prob.new = 39.198; A = 5.5353e-11"
## [1] "proposing r: prob.old = 63.446; prob.new = -80342; A = 0"
## [1] "sample number 3500"
```

Trace of r

Trace of K

Trace of logsd.N


Trace of N

Trace of lpost
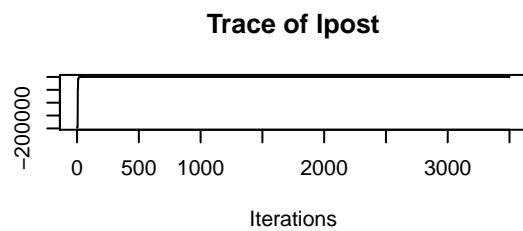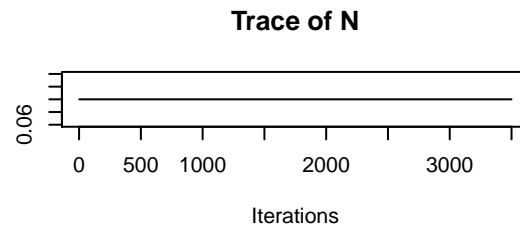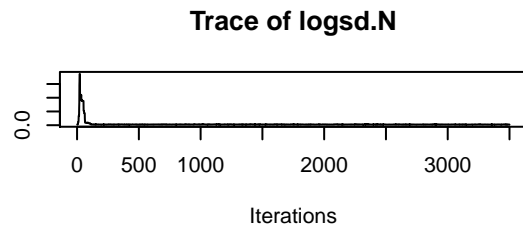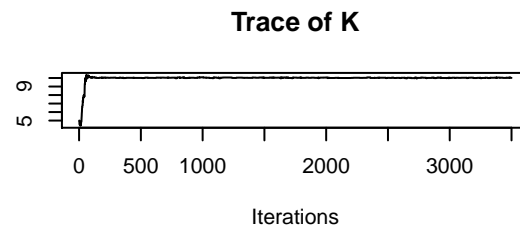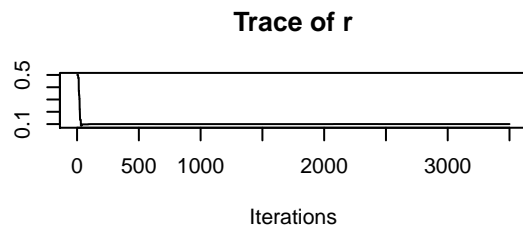

```
## [1] "proposing r: prob.old = 60.726; prob.new = -223110; A = 0"
## [1] "proposing logsd.N: prob.old = 60.726; prob.new = 58.709; A = 0.15692"
## [1] "proposing K: prob.old = 58.709; prob.new = -18.732; A = 2.3327e-34"
## [1] "sample number 4000"
```

Trace of r

Iterations

Trace of K

Iterations

Trace of logsd.N

Iterations

Trace of N

Iterations

Trace of lpost

Iterations

```
## [1] "proposing r: prob.old = 62.404; prob.new = -49424; A = 0"
## [1] "proposing logsd.N: prob.old = 62.404; prob.new = 62.416; A = 1.0673"
## [1] "proposing K: prob.old = 62.416; prob.new = 61.392; A = 0.35938"
## [1] "sample number 4500"
```

Trace of r

Trace of K

Trace of logsd.N

Trace of N

Trace of lpost
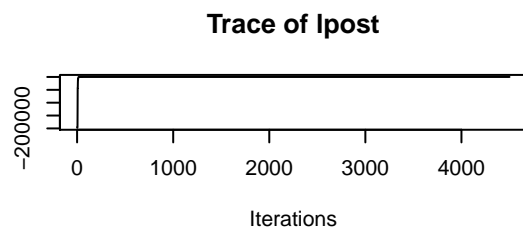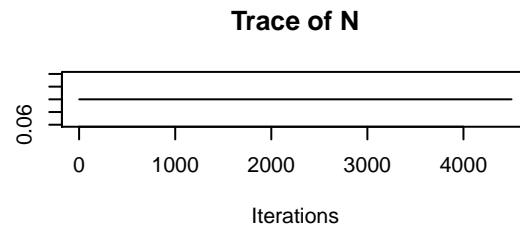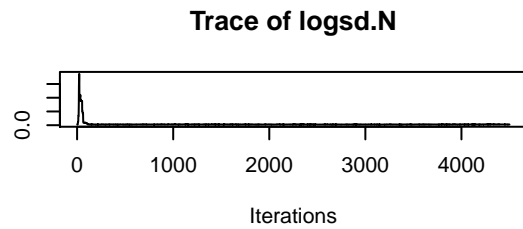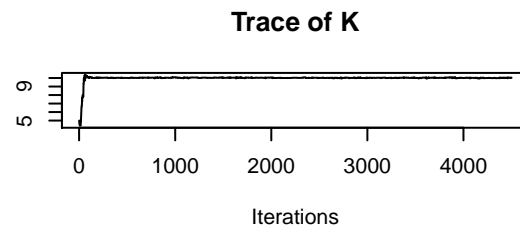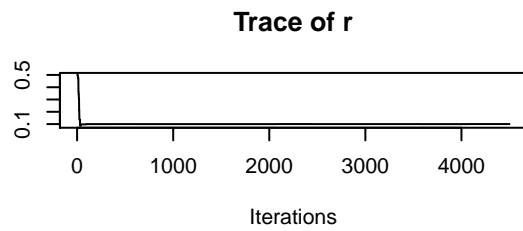

```
## [1] "proposing logsd.N: prob.old = 61.597; prob.new = 56.528; A = 0.0038677"
## [1] "proposing r: prob.old = 61.597; prob.new = -66996; A = 0"
## [1] "proposing K: prob.old = 61.597; prob.new = 22.872; A = 1.5202e-17"
## [1] "sample number 5000"
```

Trace of r

Iterations

Trace of K

Iterations

Trace of logsd.N

Iterations

Trace of N

Iterations

Trace of lpost

Iterations