# Analysing data with temporal dependence

*Peter Solymos and Subhash Lele*

*July 16, 2016 — Madison, WI — NACCB Congress*

Now that we are familiar with hierarchical models in the regression setup with independent data, we will now extend these ideas to dependent data situations. We will consider the linear and non-linear time series models used in modeling population dynamics as our examples. The principles, of course, apply to more general situations.

## Discrete time population dynamics models: General setup

We know that next year's population size depends on the current population size, recruitment rate and survival rate. In the case of meta-population models, we may also need to take into account immigration and emigration in the models. Almost all of these rates are stochastic, changing from one year to the next depending on the environment. We may also have to consider demographic stochasticity when the populations are closer to extinction. A general form for deterministic models is:

$$N_{t+1} = f(N_t, \theta)$$

There are a number of assumptions that we make to make statistical analysis feasible. For example, we assume the environmental noise is additive. We can also consider additive noise on the log-scale because we are interested in modeling growth

$$N_{t+1}/N_t$$

. This is because we tend to use differential equation models of the type: $\frac{1}{N}\frac{dN}{dt} = f(N)$. These models are models of growth.

Let $X_t = log(N_t)$. Then another way to write the population growth models in discrete time is:

$$log(N_{t+1}) - log(N_t) = X_{t+1} - X_t = f(N_t) + \varepsilon_{t+1}$$

Choice of different functional forms for $f(N_t, \theta)$ leads to different growth models. Each one has different dynamical properties. They may also lead to chaotic (or, bifurcation etc.) for different sets of parameters. Please make yourself familiar with these before using the models for estimation and forecasting. Our goal here is to show you why and how hierarchical models are used in this context.

This lecture is more bare-bones than the previous ones. Please feel free to contact the organizers if you have questions.

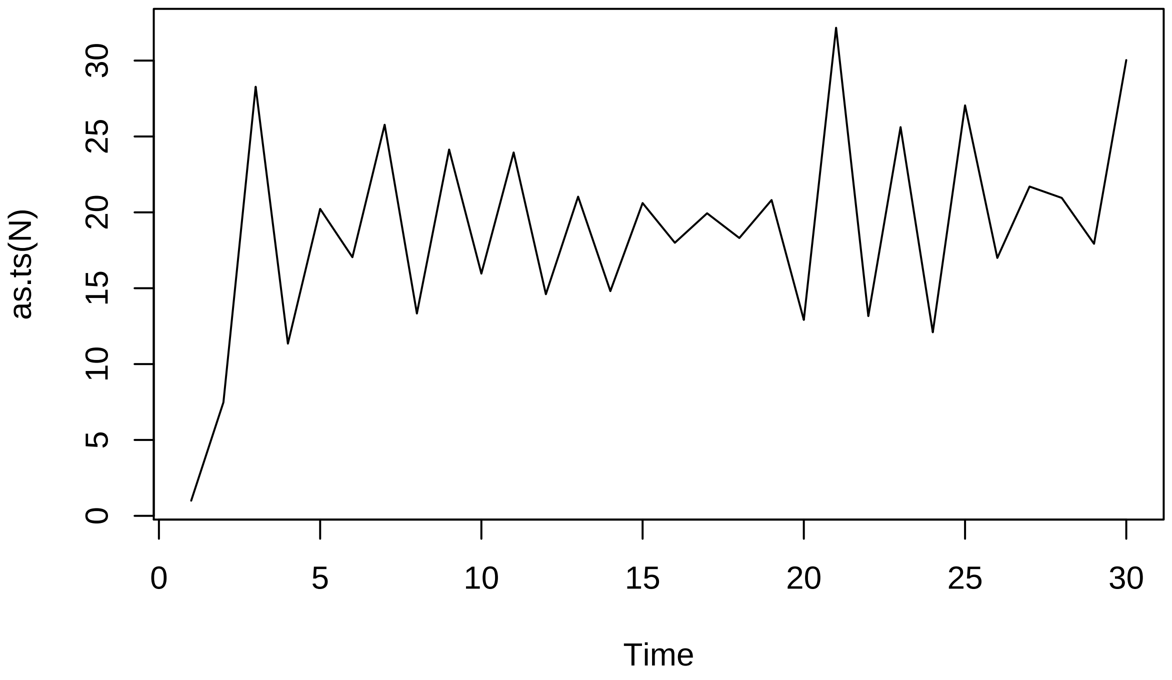
## Ricker growth model

### Ricker without observation error

a-b parametrization (similar to Gompertz and Kalman filter), data generation:

```r
set.seed(23432)
a <- 2
b <- -0.1
sigma_sq <- 0.05
T <- 30
```

```r
x <- numeric(T)
x[1] <- log(1)
for (t in 2:T)
    x[t] <- x[t - 1] + a + b * exp(x[t - 1]) +
        rnorm(1, 0, sqrt(sigma_sq))
N <- exp(x)
plot(as.ts(N))
```



Bayesian analysis

```r
library(dclone)
```

```
## Loading required package: coda

## Loading required package: parallel

## Loading required package: Matrix

## dclone 2.1-1      2016-01-11
```

```r
model <- custommodel("model {
    for (k in 1:kk) {
        N[1,k] <- exp(x[1,k])
        for (t in 2:T) {
            x[t,k] ~ dnorm(mu[t,k], 1 / sigma_sq)
            mu[t,k] <- a + b * N[t - 1,k] + x[t - 1,k]
            N[t,k] <- min(exp(x[t,k]), 10000)
        }
    }
    a ~ dnorm(1, 0.01)
    b ~ dnorm(0, 0.1)
    sigma_sq <- exp(log_sigma)^2
    log_sigma ~ dnorm(0, 1)
}")
dat <- list(x = data.matrix(x), kk = 1, T = T)
fit <- jags.fit(dat, c("a", "b", "log_sigma"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 3
##    Total graph size: 1119
##
## Initializing model
```

```
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a           1.95209 0.106680 8.710e-04      0.0036846
## b          -0.09903 0.005456 4.455e-05      0.0001874
## log_sigma -1.70056 0.142270 1.162e-03      0.0016416
##
## 2. Quantiles for each variable:
##
##               2.5%     25%      50%      75%   97.5%
## a           1.7378  1.8831  1.95124  2.02076  2.1618
## b          -0.1098 -0.1026 -0.09903 -0.09552 -0.0881
## log_sigma -1.9636 -1.7984 -1.70704 -1.60810 -1.4023
```

Data cloning

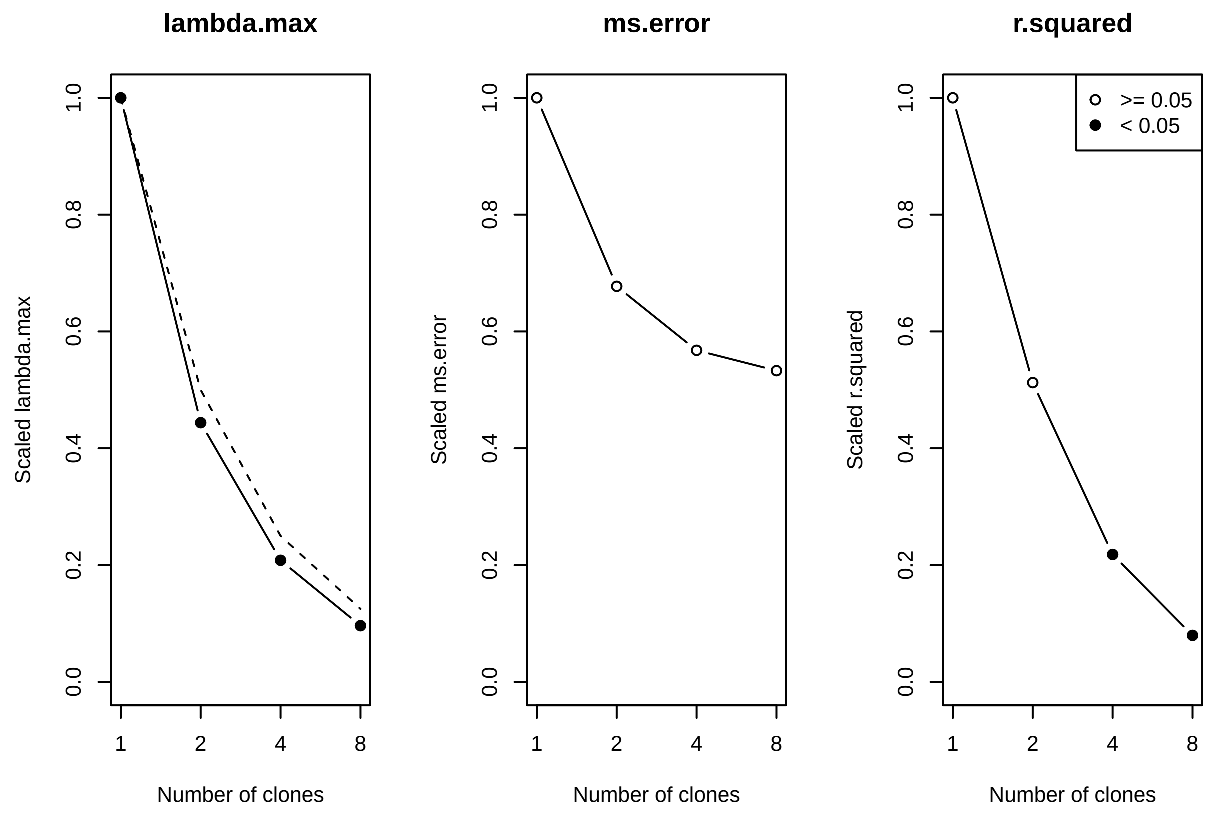
```
K <- c(1, 2, 4, 8)
dat <- list(x = dcdim(data.matrix(x)), kk = 1, T = T)
dcfit <- dc.fit(dat, c("a", "b", "sigma_sq"), model,
    n.clones = K,
    unchanged = "T", multiply = "kk")
```

```
##
## Fitting model with 1 clone
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 3
##    Total graph size: 1119
##
## Initializing model
##
##
## Fitting model with 2 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 58
##    Unobserved stochastic nodes: 3
##    Total graph size: 2106
##
## Initializing model
##
##
## Fitting model with 4 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 116
##    Unobserved stochastic nodes: 3
##    Total graph size: 4080
##
## Initializing model
##
##
## Fitting model with 8 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 232
##    Unobserved stochastic nodes: 3
##    Total graph size: 8028
##
## Initializing model
```

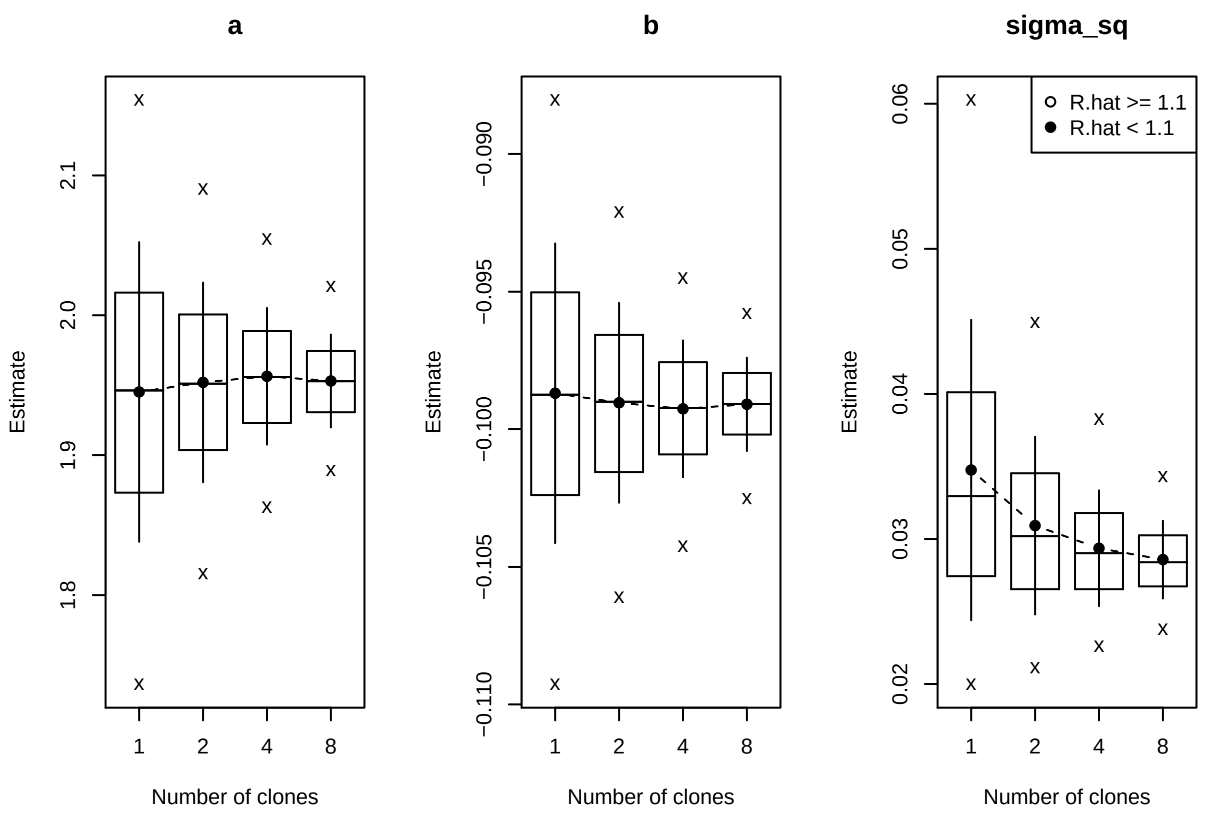
```r
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a           1.95209 0.106680 8.710e-04      0.0036846
## b          -0.09903 0.005456 4.455e-05      0.0001874
## log_sigma -1.70056 0.142270 1.162e-03      0.0016416
##
## 2. Quantiles for each variable:
##
##              2.5%     25%      50%      75%   97.5%
## a          1.7378  1.8831  1.95124  2.02076  2.1618
## b         -0.1098 -0.1026 -0.09903 -0.09552 -0.0881
## log_sigma -1.9636 -1.7984 -1.70704 -1.60810 -1.4023
```

```r
plot(dcdiag(dcfit))
```



```r
plot(dctable(dcfit))
```



Missing data

```r
dat$x[5,1] <- NA
```

Bayesian approach to predict missing observations and credible intervals

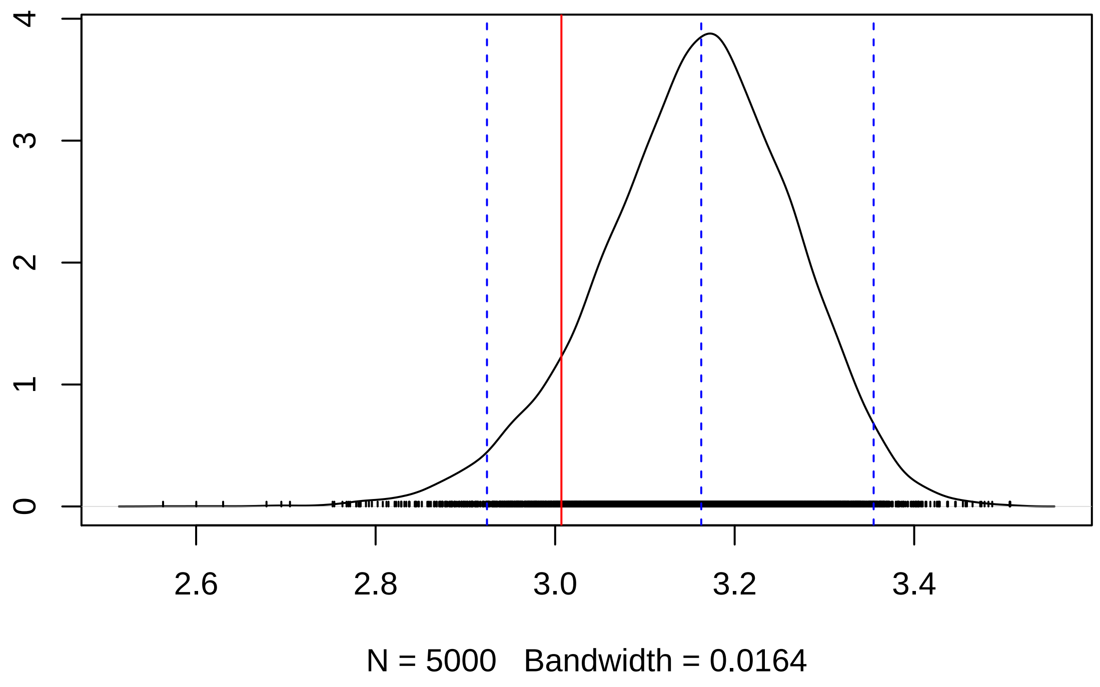
```r
fit <- jags.fit(dat, c("a", "b", "sigma_sq", "x[5,1]"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 28
##    Unobserved stochastic nodes: 4
##    Total graph size: 1119
##
## Initializing model
```

```r
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a          1.97206 0.101011 8.248e-04      0.0032556
## b         -0.09949 0.005106 4.169e-05      0.0001649
## sigma_sq   0.03315 0.010516 8.587e-05      0.0001346
## x[5,1]     3.15671 0.109516 8.942e-04      0.0012199
##
## 2. Quantiles for each variable:
##
##               2.5%      25%      50%      75%    97.5%
## a          1.77470  1.90590  1.97141  2.03663  2.17301
## b         -0.10968 -0.10279 -0.09947 -0.09617 -0.08952
## sigma_sq   0.01864  0.02596  0.03120  0.03815  0.05907
## x[5,1]     2.92406  3.08942  3.16274  3.23129  3.35482
```

```r
densplot(fit[,"x[5,1]"])
abline(v = x[5], col = 2)
abline(v = quantile(fit[,"x[5,1]"], probs = c(0.025, 0.5, 0.975)),
    col = 4, lty = 2)
```



```r
quantile(fit[,"x[5,1]"], probs = c(0.025, 0.5, 0.975))
```

```
##           var1
## 2.5%  2.924065
## 50%   3.162744
## 97.5% 3.354820
```

```r
x[5]
```

```
## [1] 3.006971
```

DC approach to predict missing observations

```r
pmodel <- custommodel("model {
    for (k in 1:kk) {
        N[1,k] <- exp(x[1,k])
        for (t in 2:T) {
            x[t,k] ~ dnorm(mu[t,k], 1 / sigma_sq)
            mu[t,k] <- a + b * N[t - 1,k] + x[t - 1,k]
            N[t,k] <- min(exp(x[t,k]), 10000)
        }
    }
    param[1:3] ~ dmnorm(cf[], Vinv[,])
    a <- param[1]
    b <- param[2]
    log_sigma <- param[3]
    sigma_sq <- exp(log_sigma)^2
}")
dcfit2 <- dc.fit(dat, c("a", "b", "log_sigma"), model,
    n.clones = 8,
    unchanged = "T", multiply = "kk")
```

```
##
## Fitting model with 8 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 224
##    Unobserved stochastic nodes: 11
##    Total graph size: 8056
##
## Initializing model
```

```r
(V <- vcov(dcfit2)[c("a", "b", "log_sigma"), c("a", "b", "log_sigma")])
```

```
##                        a             b     log_sigma
## a           9.077218e-03 -4.329772e-04 -8.923023e-05
## b          -4.329772e-04  2.324445e-05  3.430452e-06
## log_sigma -8.923023e-05  3.430452e-06  1.847909e-02
```

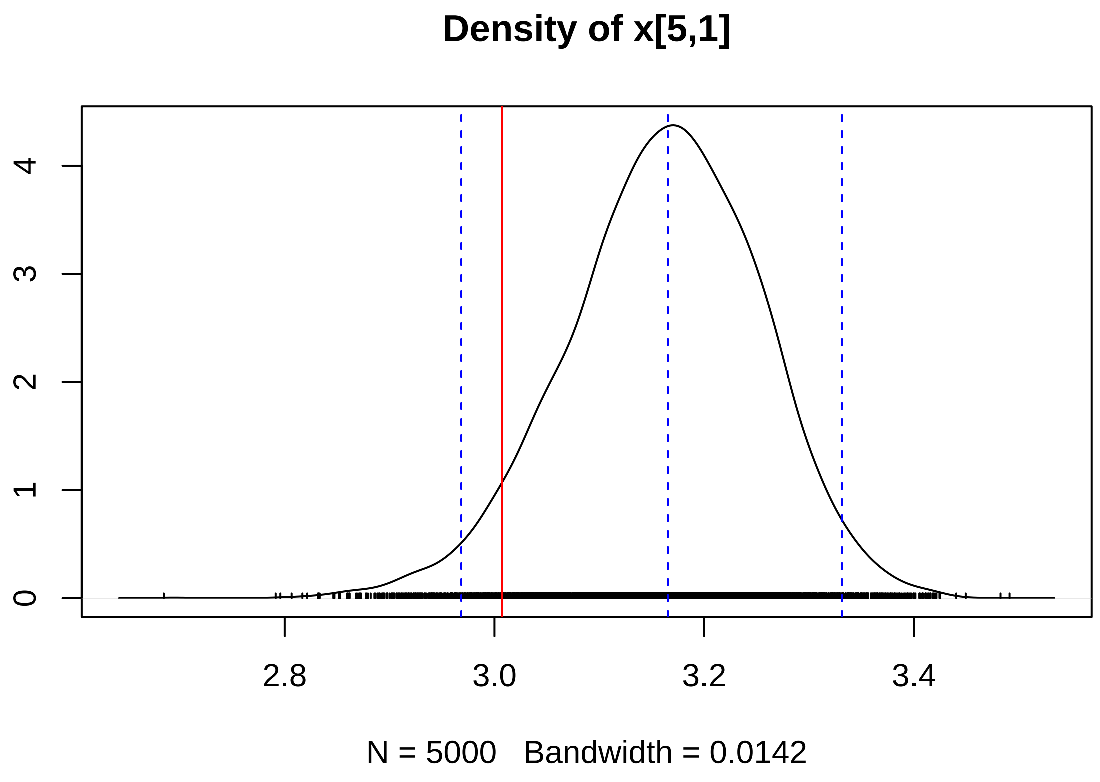
```r
(cf <- coef(dcfit2)[c("a", "b", "log_sigma")])
```

```
##           a           b   log_sigma
##  1.97006362 -0.09937436 -1.80536000
```

```r
prdat <- c(dat, list(cf = cf, Vinv = solve(V)))
pred <- jags.fit(prdat, "x[5,1]", pmodel)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 28
##    Unobserved stochastic nodes: 2
##    Total graph size: 1188
##
## Initializing model
```

```r
densplot(pred)
abline(v = x[5], col = 2)
abline(v = quantile(pred, probs = c(0.025, 0.5, 0.975)), col = 4, lty = 2)
```



```r
x[5]
```

```
## [1] 3.006971
```

```r
quantile(pred, probs = c(0.025, 0.5, 0.975))
```

```
##         x[5,1]
## 2.5%  2.968345
## 50%   3.165397
## 97.5% 3.331370
```

Predicting future states, Bayesian

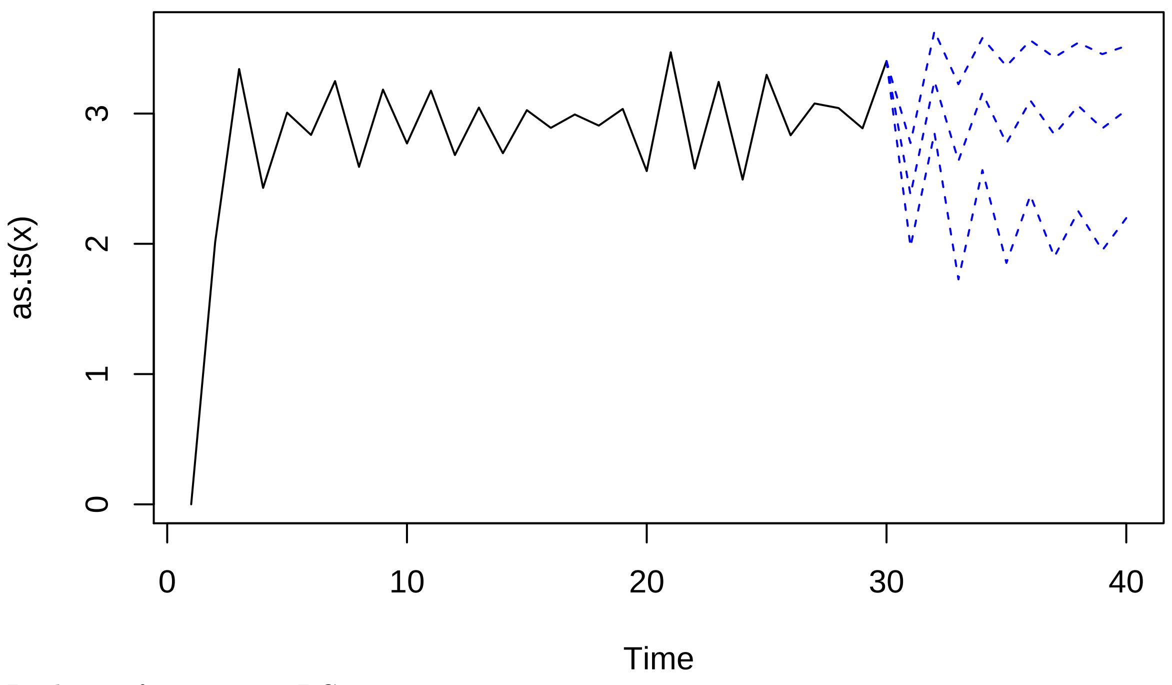
```r
T2 <- 10
dat2 <- list(x = data.matrix(c(x, rep(NA, T2))),
    kk = 1, T = T + T2)
fit <- jags.fit(dat2, c("x[31:40,1]"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 13
##    Total graph size: 1889
##
## Initializing model
```

```r
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## x[31,1] 2.377 0.2004 0.001636       0.002333
## x[32,1] 3.247 0.1967 0.001606       0.002173
## x[33,1] 2.596 0.3861 0.003152       0.004608
## x[34,1] 3.137 0.2568 0.002096       0.002533
## x[35,1] 2.728 0.3824 0.003122       0.004187
## x[36,1] 3.068 0.2997 0.002447       0.002605
## x[37,1] 2.794 0.3902 0.003186       0.003889
## x[38,1] 3.017 0.3316 0.002707       0.002856
## x[39,1] 2.839 0.3852 0.003145       0.003861
## x[40,1] 2.982 0.3397 0.002774       0.002747
##
## 2. Quantiles for each variable:
##
##          2.5%   25%   50%   75% 97.5%
## x[31,1] 1.977 2.247 2.377 2.510 2.774
## x[32,1] 2.850 3.118 3.249 3.376 3.633
## x[33,1] 1.727 2.383 2.637 2.856 3.225
## x[34,1] 2.566 2.998 3.157 3.305 3.580
## x[35,1] 1.853 2.510 2.772 2.987 3.364
## x[36,1] 2.372 2.908 3.100 3.267 3.562
## x[37,1] 1.900 2.576 2.839 3.062 3.431
## x[38,1] 2.250 2.843 3.059 3.239 3.544
## x[39,1] 1.950 2.622 2.886 3.106 3.456
## x[40,1] 2.197 2.799 3.026 3.214 3.519
```

```
pi1 <- quantile(fit, probs = c(0.025, 0.5, 0.975))

plot(as.ts(x), xlim=c(1,40), ylim = range(c(x, pi1)))
matlines(30:40, rbind(x[c(30, 30, 30)], t(pi1)), col=4, lty=2)
```



Predicting future states, DC

```
prdat2 <- c(dat2, list(cf = cf, Vinv = solve(V)))
pred2 <- jags.fit(prdat2, "x[31:40,1]", pmodel)
```

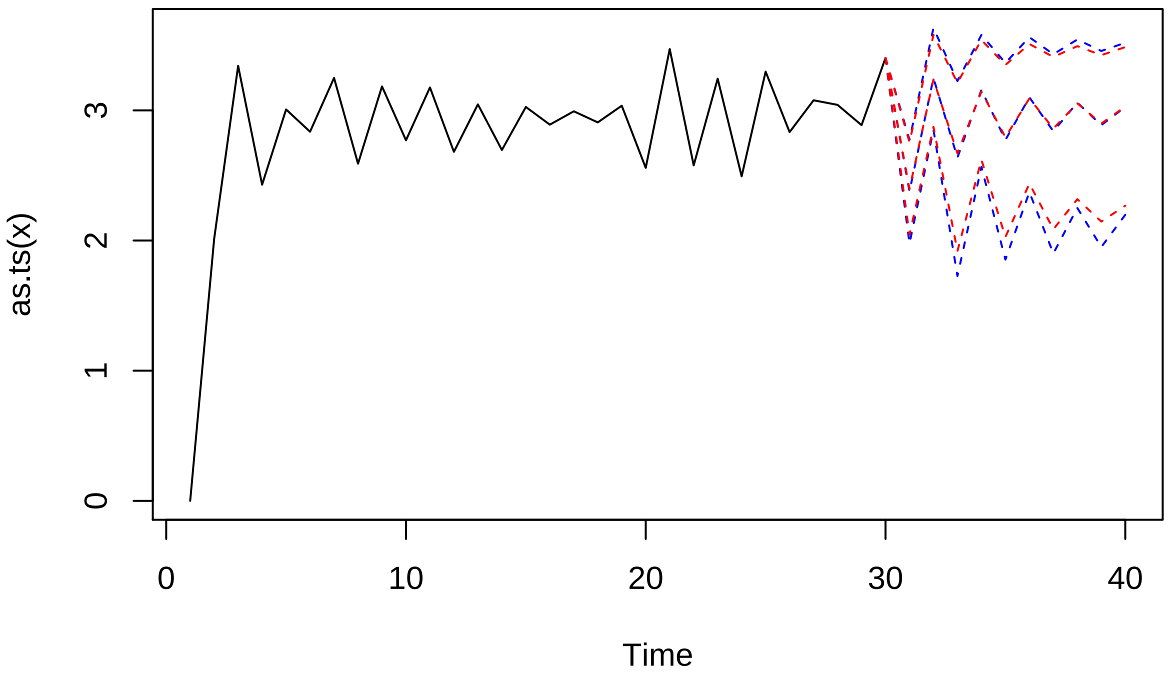
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 11
##    Total graph size: 1978
##
## Initializing model
```

```
summary(pred2)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## x[31,1] 2.399 0.1808 0.001476       0.001903
## x[32,1] 3.234 0.1797 0.001467       0.002126
## x[33,1] 2.643 0.3306 0.002699       0.003702
## x[34,1] 3.127 0.2344 0.001913       0.002594
## x[35,1] 2.768 0.3361 0.002744       0.004097
## x[36,1] 3.059 0.2752 0.002247       0.002480
## x[37,1] 2.830 0.3385 0.002764       0.002937
## x[38,1] 3.015 0.2993 0.002443       0.002618
## x[39,1] 2.867 0.3315 0.002707       0.003270
## x[40,1] 2.989 0.3097 0.002528       0.002862
##
## 2. Quantiles for each variable:
##
##           2.5%   25%   50%   75% 97.5%
## x[31,1] 2.044 2.277 2.396 2.521 2.755
## x[32,1] 2.873 3.116 3.237 3.355 3.580
## x[33,1] 1.918 2.441 2.670 2.874 3.212
## x[34,1] 2.621 2.990 3.145 3.285 3.541
## x[35,1] 2.027 2.568 2.793 3.003 3.349
## x[36,1] 2.438 2.903 3.092 3.251 3.510
## x[37,1] 2.086 2.623 2.860 3.067 3.411
## x[38,1] 2.318 2.848 3.054 3.225 3.493
## x[39,1] 2.146 2.661 2.898 3.103 3.424
## x[40,1] 2.268 2.809 3.029 3.209 3.486
```

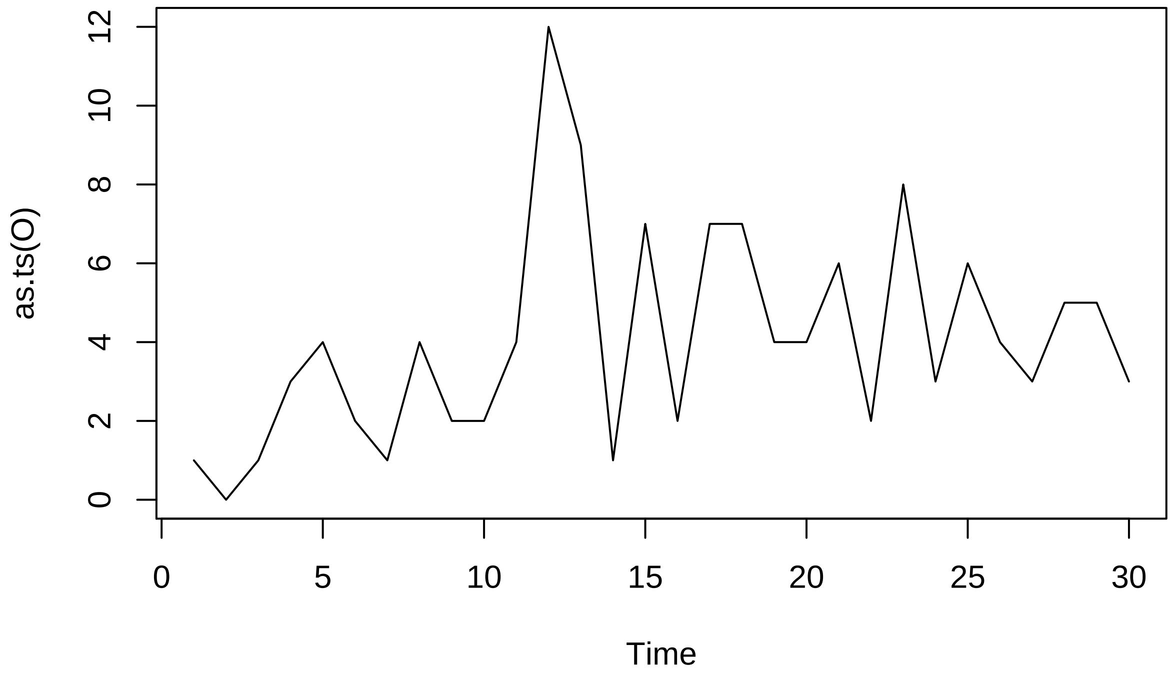
```
pi2 <- quantile(pred2, probs = c(0.025, 0.5, 0.975))

plot(as.ts(x), xlim=c(1,40), ylim = range(c(x, pi1, pi2)))
matlines(30:40, rbind(x[c(30, 30, 30)], t(pi1)), col=4, lty=2)
matlines(30:40, rbind(x[c(30, 30, 30)], t(pi2)), col=2, lty=2)
```



**Ricker with observation error**

Poisson observation error:

```r
set.seed(2345)
a <- 0.5
b <- -0.1
sigma_sq <- 0.05
T <- 30
x <- numeric(T)
x[1] <- log(1)
for (t in 2:T)
    x[t] <- x[t - 1] + a + b * exp(x[t - 1]) +
        rnorm(1, 0, sqrt(sigma_sq))
O <- rpois(T, lambda = exp(x))
plot(as.ts(O))
```



```r
library(dclone)
model <- custommodel("model {
    for (k in 1:kk) {
        x[1,k] <- log(O[1,k])
        N[1,k] <- O[1,k]
        for (t in 2:T) {
            x[t,k] ~ dnorm(mu[t,k], 1 / sigma_sq)
            mu[t,k] <- a + b * N[t - 1,k] + x[t - 1,k]
            N[t,k] <- min(exp(x[t,k]), 10000)
            O[t,k] ~ dpois(N[t,k])
        }
    }
    a ~ dnorm(1, 0.01)
    b ~ dnorm(0, 1)
    sigma_sq <- exp(log_sigma)^2
    log_sigma ~ dnorm(0, 1)
}")
dat <- list(O = data.matrix(O), kk = 1, T = T)
fit <- jags.fit(dat, c("a", "b", "sigma_sq"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 32
##    Total graph size: 1148
##
## Initializing model
```

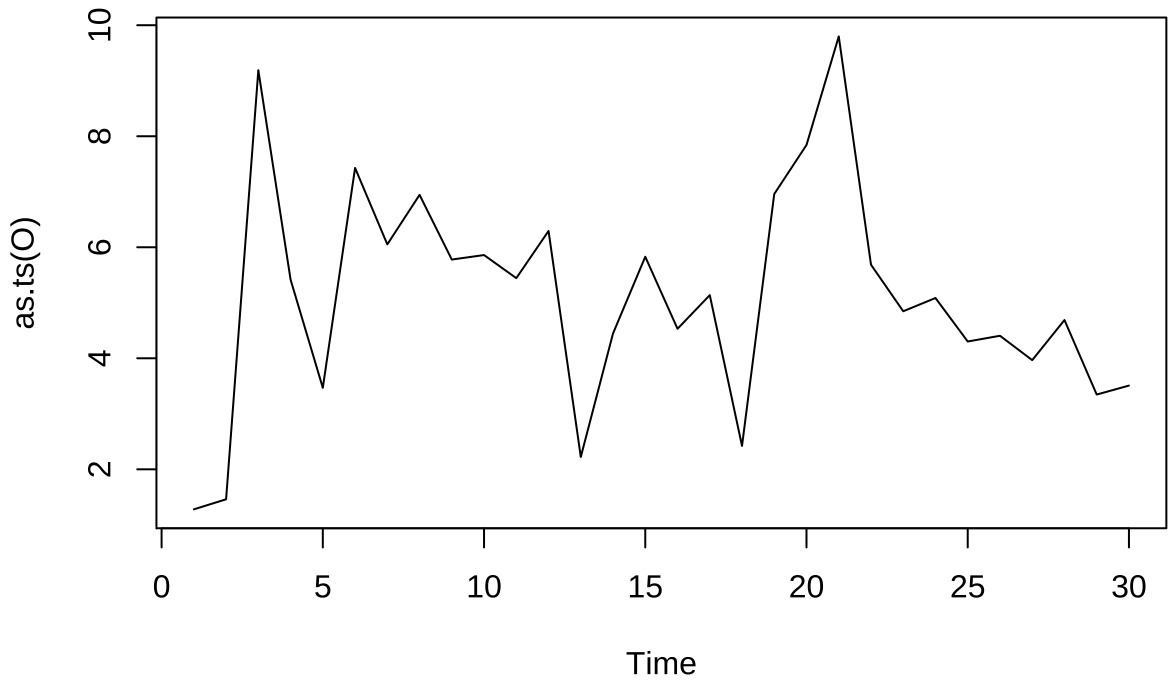
```
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a          0.6728 0.30487 0.0024893       0.020571
## b         -0.1536 0.07388 0.0006032       0.004921
## sigma_sq   0.1147 0.09949 0.0008123       0.004321
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%      75%    97.5%
## a          0.210558  0.43775  0.63358  0.86181  1.36122
## b         -0.322549 -0.19843 -0.14453 -0.09645 -0.04282
## sigma_sq   0.007249  0.04599  0.08814  0.15308  0.38098
```

Normal observation error: this comes with an additional parameter

```
set.seed(12321)
a <- 0.5
b <- -0.1
sigma_sq <- 0.05
tau_sq <- 0.1
T <- 30
x <- numeric(T)
x[1] <- log(1)
for (t in 2:T)
    x[t] <- x[t - 1] + a + b * exp(x[t - 1]) +
        rnorm(1, 0, sqrt(sigma_sq))
```

```r
O <- exp(rnorm(T, mean = x, sd = sqrt(tau_sq)))
plot(as.ts(O))
```



```r
model <- custommodel("model {
    for (k in 1:kk) {
        N[1,k] <- exp(O[1,k])
        x[1,k] <- O[1,k]
        for (t in 2:T) {
            x[t,k] ~ dnorm(mu[t,k], 1 / sigma_sq)
            mu[t,k] <- a + b * N[t-1,k] + x[t-1,k]
            N[t,k] <- min(exp(x[t,k]), 10000)
            O[t,k] ~ dnorm(x[t,k], 1 / tau_sq)
        }
    }
    a ~ dnorm(0, 0.01)
    b ~ dnorm(0, 10)
    sigma_sq <- exp(log_sigma)^2
    log_sigma ~ dnorm(0, 0.01)
    tau_sq <- exp(log_tau)^2
    log_tau ~ dnorm(0, 0.01)
}")
dat <- list(O = data.matrix(O), kk = 1, T = T)
fit <- jags.fit(dat, c("a", "b", "sigma_sq", "tau_sq"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 33
##    Total graph size: 1648
##
## Initializing model
```

```
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a         1.916269 0.311120 0.0025403      0.0503522
## b        -0.010776 0.002522 0.0000206      0.0004639
## sigma_sq  0.007601 0.025508 0.0002083      0.0039215
## tau_sq    3.662029 1.052776 0.0085959      0.0158622
##
## 2. Quantiles for each variable:
##
##                 2.5%        25%        50%       75%     97.5%
## a         1.184e+00  1.788e+00  2.063e+00  2.126217  2.240114
## b        -1.434e-02 -1.314e-02 -1.058e-02 -0.008412 -0.006649
## sigma_sq  3.093e-12  5.180e-08  1.623e-05  0.001980  0.065434
## tau_sq    2.127e+00  2.920e+00  3.486e+00  4.205296  6.211118
```

There is a package called PVAClone that can fit all of these models:

```
library(PVAClone)
```

```
## Loading required package: dcmle

## dcmle 0.3-0   2016-01-12

##
## Attaching package: 'dcmle'

## The following objects are masked from 'package:coda':
##
##     chanames, crosscorr.plot, gelman.diag, gelman.plot,
##     geweke.diag, heidel.diag, raftery.diag, varnames

## Loading required package: stats4

## PVAClone 0.1-5    2016-01-14
##     check out ?PVA for an overview
```

```r
m <- pva(O, model = ricker("normal"), n.clones=c(1, 2, 4))
```

```
## Warning in pva(O, model = ricker("normal"), n.clones = c(1, 2, 4)): non-
## integer values found in data
```

```
##
## Fitting model with 1 clone
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 33
##    Total graph size: 697
##
## Initializing model
##
##
## Fitting model with 2 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 58
##    Unobserved stochastic nodes: 62
##    Total graph size: 1363
##
## Initializing model
##
##
## Fitting model with 4 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 116
##    Unobserved stochastic nodes: 120
##    Total graph size: 2695
##
## Initializing model
```

```r
summary(m)
```

```
## PVA object:
## Ricker growth model with Normal observation error
##
## Time series with 30 observations (missing: 0)
##
## Call:
```

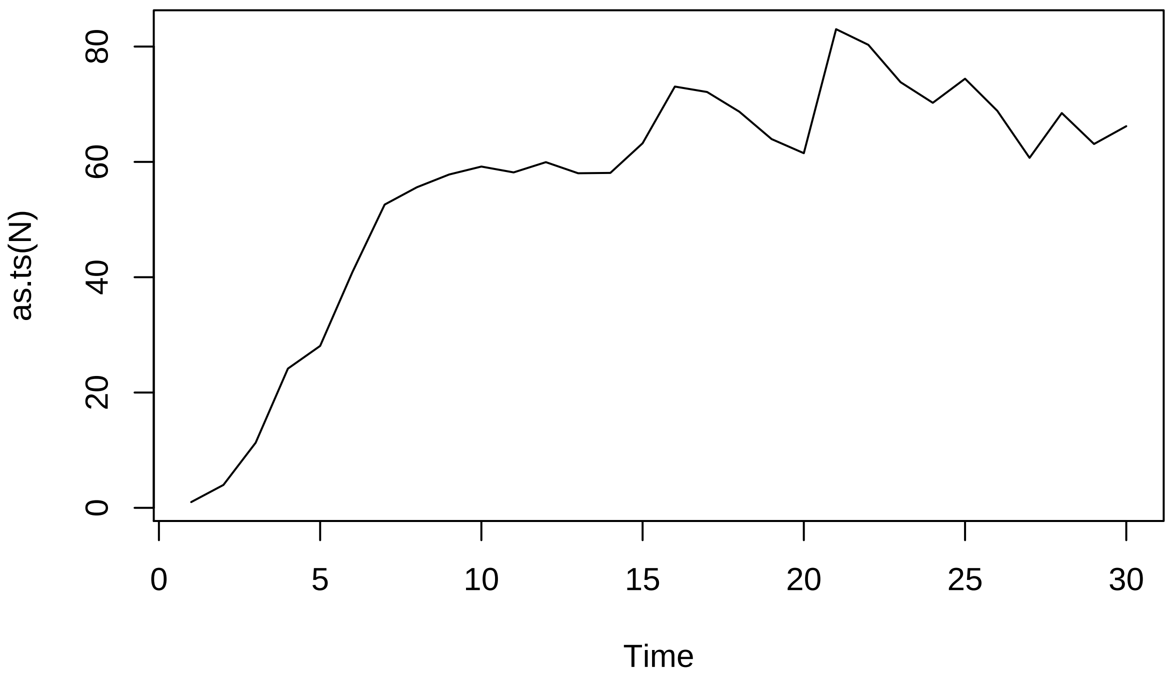
```
## pva(x = 0, model = ricker("normal"), n.clones = c(1, 2, 4))
##
## Settings:
##  start  end thin n.iter n.chains n.clones
##   2001 7000    1   5000        3        4
##
## Coefficients:
##       Estimate Std. Error z value Pr(>|z|)
## a      1.04294    0.24636   4.233 2.30e-05 ***
## b     -0.20393    0.04914  -4.150 3.33e-05 ***
## sigma  0.12770    0.10268   1.244    0.214
## tau    0.35268    0.06288   5.609 2.04e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Convergence:
##  n.clones lambda.max ms.error r.squared r.hat
##         1     0.5629    5.246   0.08557    NA
##         2     0.4235    4.562   0.04661    NA
##         4     0.3619    4.384   0.01779    NA
```

## Gompertz growth model

### Gompertz model without observation error

Data generation

```
set.seed(1234)
a <- 1.5
b <- -0.35
sigma_sq <- 0.01
T <- 30
x <- numeric(T)
x[1] <- log(1) # initial log abundance
for (t in 2:T)
    x[t] <- x[t - 1] + a + b * x[t - 1] + rnorm(1, 0, sqrt(sigma_sq))
N <- exp(x)
plot(as.ts(N))
```



Bayesiam analysis

```r
library(dclone)
model <- custommodel("model {
    for (k in 1:K) {
        x[1,k] ~ dnorm(a / (1 - b), (1 - b^2) / sigma_sq)
        for (t in 2:T) {
            x[t,k] ~ dnorm(mu[t,k], 1 / sigma_sq)
            mu[t,k] <- a + (1 + b) * x[t - 1,k]
        }
    }
    a ~ dnorm(0, 0.01)
    b ~ dunif(-0.999, -0.001)
    z <- 0.5 * log((1 + b) / (1 - b))
    sigma_sq <- exp(log_sigma)^2
    log_sigma ~ dnorm(0, 0.01)
}")
dat <- list(x = data.matrix(x), K = 1, T = T)
fit <- jags.fit(dat, c("a", "b", "sigma_sq"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 30
##    Unobserved stochastic nodes: 3
##    Total graph size: 239
##
## Initializing model
```

```r
summary(fit)
```

```
##
```

```
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a         1.20095 0.16242 0.0013261       0.008803
## b        -0.28446 0.04185 0.0003417       0.002361
## sigma_sq  0.04418 0.01291 0.0001054       0.000169
##
## 2. Quantiles for each variable:
##
##              2.5%      25%      50%     75%    97.5%
## a         0.88811  1.08914  1.20118  1.3130  1.51302
## b        -0.36553 -0.31327 -0.28440 -0.2554 -0.20275
## sigma_sq  0.02576  0.03512  0.04185  0.0507  0.07557
```

Data cloning

```
K <- c(1, 2, 4, 8)
dat <- list(x = dcdim(data.matrix(x)), K = 1, T = T)
dcfit <- dc.fit(dat, c("a", "b", "sigma_sq"), model,
    n.clones = K,
    unchanged = "T", multiply = "K")
```

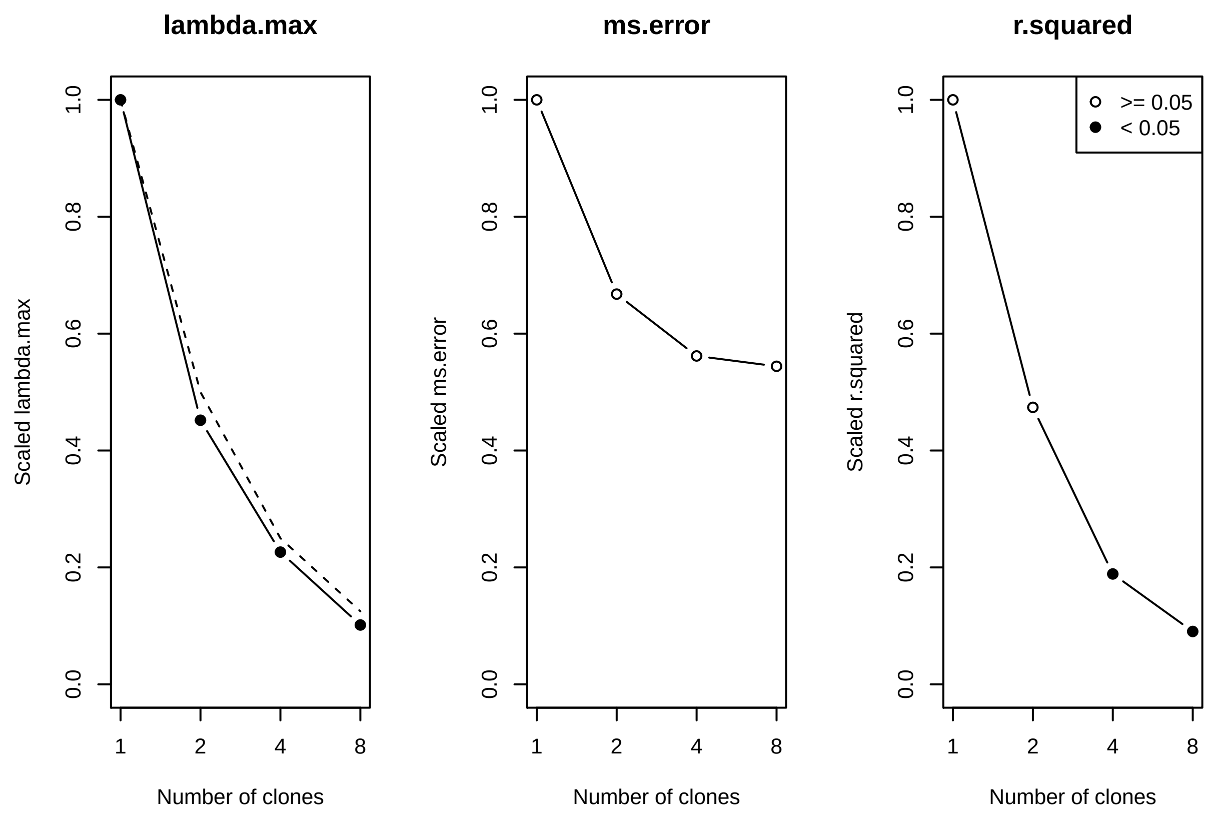
```
##
## Fitting model with 1 clone
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 30
##    Unobserved stochastic nodes: 3
##    Total graph size: 239
##
## Initializing model
##
##
## Fitting model with 2 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 60
##    Unobserved stochastic nodes: 3
##    Total graph size: 388
##
## Initializing model
```

```
##
##
## Fitting model with 4 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 120
##    Unobserved stochastic nodes: 3
##    Total graph size: 686
##
## Initializing model
##
##
## Fitting model with 8 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 240
##    Unobserved stochastic nodes: 3
##    Total graph size: 1282
##
## Initializing model
```

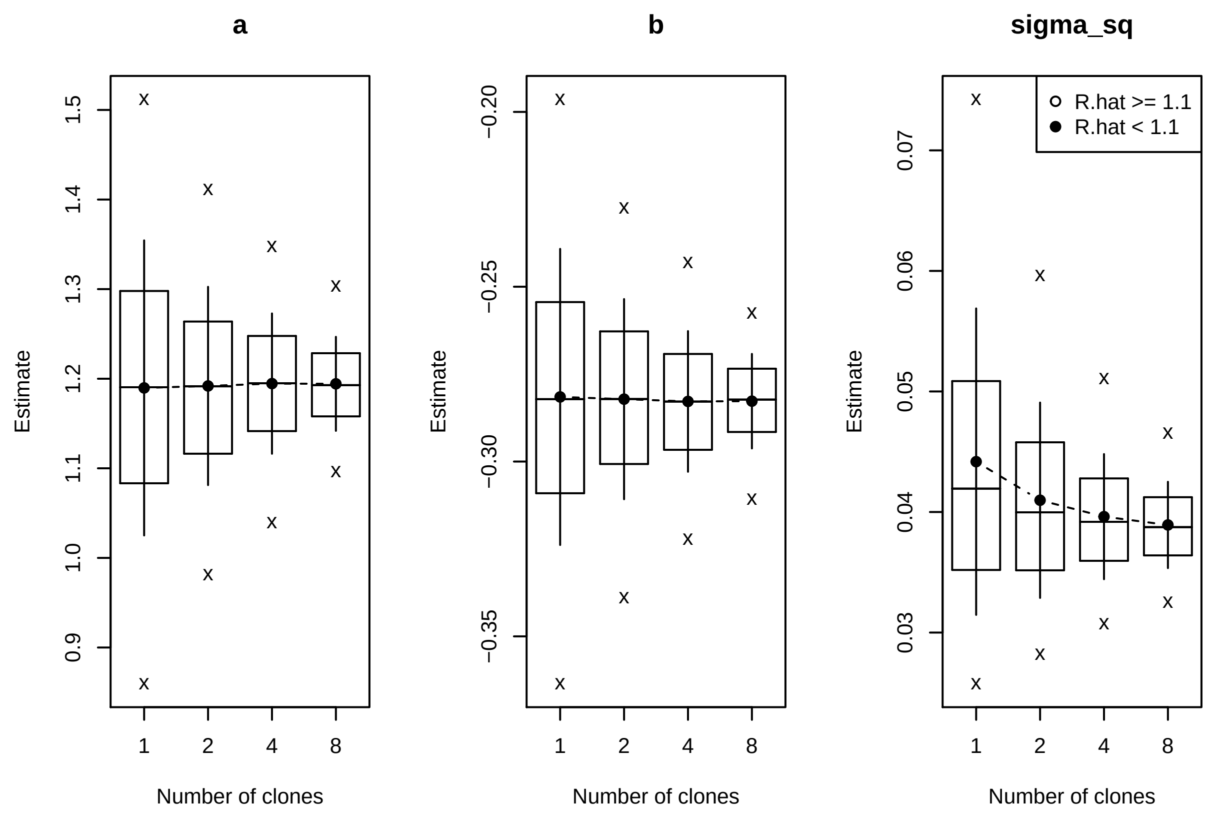
```
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a         1.20095 0.16242 0.0013261       0.008803
## b        -0.28446 0.04185 0.0003417       0.002361
## sigma_sq  0.04418 0.01291 0.0001054       0.000169
##
## 2. Quantiles for each variable:
##
##              2.5%      25%      50%     75%    97.5%
## a         0.88811  1.08914  1.20118  1.3130  1.51302
## b        -0.36553 -0.31327 -0.28440 -0.2554 -0.20275
## sigma_sq  0.02576  0.03512  0.04185  0.0507  0.07557
```

```
plot(dcdiag(dcfit))
```



```r
plot(dctable(dcfit))
```



Missing data

```
dat$x[5,1] <- NA
```

Bayesian approach to predict missing observations and credible intervals

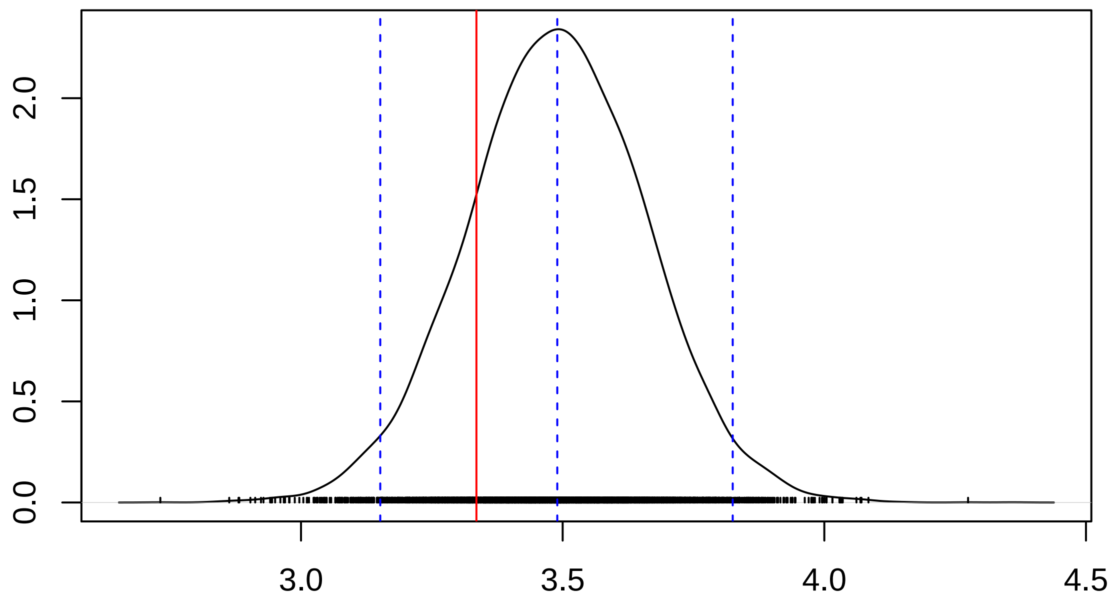
```
fit <- jags.fit(dat, c("a", "b", "sigma_sq", "x[5,1]"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 4
##    Total graph size: 239
##
## Initializing model
```

```
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a         1.19573 0.16112 0.0013155      0.0085089
## b        -0.28270 0.04141 0.0003381      0.0022432
## sigma_sq  0.04456 0.01316 0.0001075      0.0001622
## x[5,1]    3.48945 0.17116 0.0013975      0.0013627
##
## 2. Quantiles for each variable:
##
##              2.5%      25%      50%      75%    97.5%
## a         0.88494  1.08210  1.19655  1.30589  1.51204
## b        -0.36452 -0.31100 -0.28322 -0.25365 -0.20181
## sigma_sq  0.02566  0.03525  0.04239  0.05132  0.07673
## x[5,1]    3.15145  3.37576  3.48967  3.60438  3.82495
```

```
densplot(fit[,"x[5,1]"])
abline(v = x[5], col = 2)
abline(v = quantile(fit[,"x[5,1]"], probs = c(0.025, 0.5, 0.975)),
    col = 4, lty = 2)
```



N = 5000   Bandwidth = 0.02643

```r
quantile(fit[,"x[5,1]"], probs = c(0.025, 0.5, 0.975))
```

```
##           var1
## 2.5%  3.151454
## 50%   3.489670
## 97.5% 3.824950
```

```r
x[5]
```

```
## [1] 3.335179
```

DC approach to predict missing observations

```r
pmodel <- custommodel("model {
    for (k in 1:K) {
        x[1,k] ~ dnorm(a / (1 - b), (1 - b^2) / sigma_sq)
        for (t in 2:T) {
            x[t,k] ~ dnorm(mu[t,k], 1 / sigma_sq)
            mu[t,k] <- a + (1 + b) * x[t - 1,k]
        }
    }
    param[1:3] ~ dmnorm(cf[], Vinv[,])
    a <- param[1]
    z <- param[2]
    b <- (exp(2 * z) - 1) / (exp(2 * z) + 1)
    log_sigma <- param[3]
    sigma_sq <- exp(log_sigma)^2
}")
dcfit2 <- dc.fit(dat, c("a", "z", "log_sigma"), model,
    n.clones = 8,
    unchanged = "T", multiply = "K")
```

```
##
## Fitting model with 8 clones
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 232
##    Unobserved stochastic nodes: 11
##    Total graph size: 1296
##
## Initializing model
```

```
(V <- vcov(dcfit2)[c("a", "z", "log_sigma"), c("a", "z", "log_sigma")])
```

```
##                       a             z     log_sigma
## a          2.368075e-02 -6.453182e-03 -5.853515e-05
## z         -6.453182e-03  1.862645e-03  1.630518e-05
## log_sigma -5.853515e-05  1.630518e-05  1.780470e-02
```

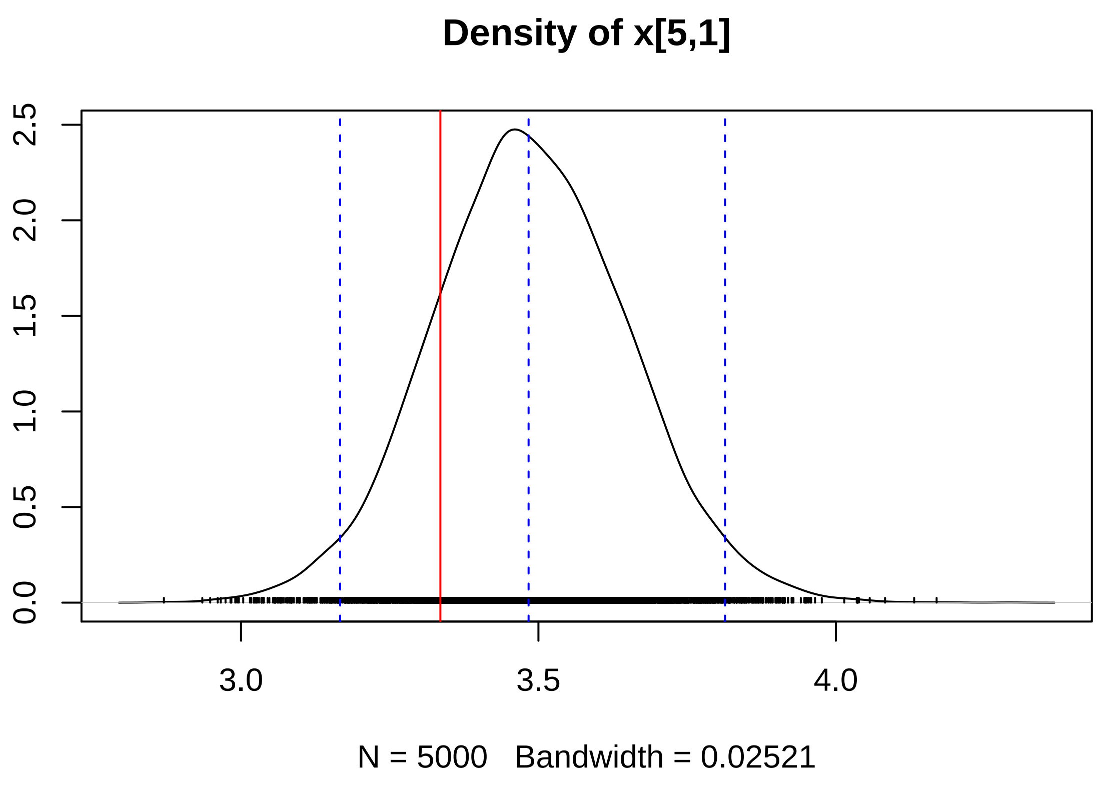
```
(cf <- coef(dcfit2)[c("a", "z", "log_sigma")])
```

```
##          a          z  log_sigma
##  1.1995412 -0.2918078 -1.6225151
```

```
prdat <- c(dat, list(cf = cf, Vinv = solve(V)))
pred <- jags.fit(prdat, "x[5,1]", pmodel)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 2
##    Total graph size: 378
##
## Initializing model
```

```
densplot(pred)
abline(v = x[5], col = 2)
abline(v = quantile(pred, probs = c(0.025, 0.5, 0.975)), col = 4, lty = 2)
```



```
x[5]
```

```
## [1] 3.335179
```

```
quantile(pred, probs = c(0.025, 0.5, 0.975))
```

```
##         x[5,1]
## 2.5%  3.166618
## 50%   3.483446
## 97.5% 3.813671
```

Predicting future states, Bayesian

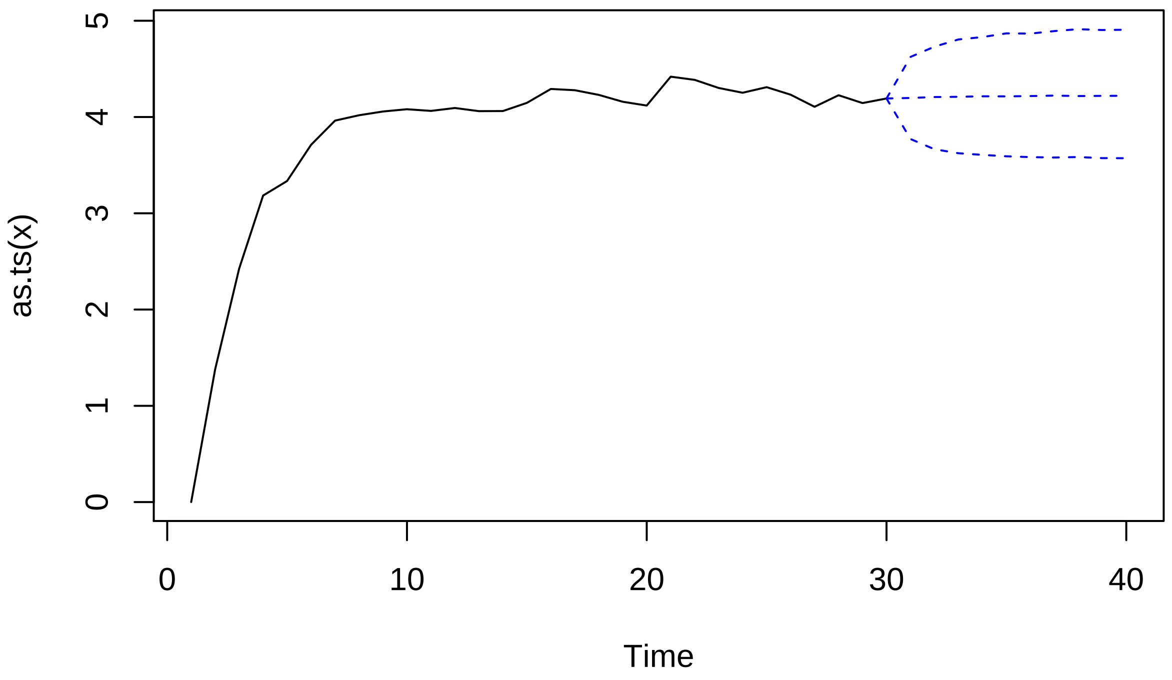
```
T2 <- 10
dat2 <- list(x = data.matrix(c(x, rep(NA, T2))),
    K = 1, T = T + T2)
fit <- jags.fit(dat2, c("x[31:40,1]"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 30
##    Unobserved stochastic nodes: 13
##    Total graph size: 309
##
## Initializing model
```

```
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## x[31,1] 4.198 0.2144 0.001751       0.001762
## x[32,1] 4.205 0.2697 0.002202       0.002219
## x[33,1] 4.212 0.2970 0.002425       0.002463
## x[34,1] 4.216 0.3108 0.002538       0.002590
## x[35,1] 4.221 0.3222 0.002631       0.002840
## x[36,1] 4.220 0.3262 0.002664       0.002664
## x[37,1] 4.225 0.3324 0.002714       0.002762
## x[38,1] 4.224 0.3369 0.002750       0.002890
## x[39,1] 4.224 0.3370 0.002752       0.003502
## x[40,1] 4.224 0.3399 0.002775       0.003270
##
## 2. Quantiles for each variable:
##
##          2.5%   25%   50%   75% 97.5%
## x[31,1] 3.771 4.058 4.198 4.340 4.624
## x[32,1] 3.665 4.028 4.207 4.383 4.731
## x[33,1] 3.624 4.015 4.210 4.410 4.806
## x[34,1] 3.607 4.010 4.215 4.421 4.830
## x[35,1] 3.592 4.006 4.214 4.433 4.869
## x[36,1] 3.583 4.003 4.218 4.427 4.866
## x[37,1] 3.580 4.001 4.222 4.438 4.893
## x[38,1] 3.584 3.998 4.218 4.443 4.912
## x[39,1] 3.573 4.001 4.219 4.442 4.904
## x[40,1] 3.572 4.000 4.220 4.438 4.907
```

```
pi1 <- quantile(fit, probs = c(0.025, 0.5, 0.975))

plot(as.ts(x), xlim=c(1,40), ylim = range(c(x, pi1)))
matlines(30:40, rbind(x[c(30, 30, 30)], t(pi1)), col=4, lty=2)
```



Predicting future states, DC

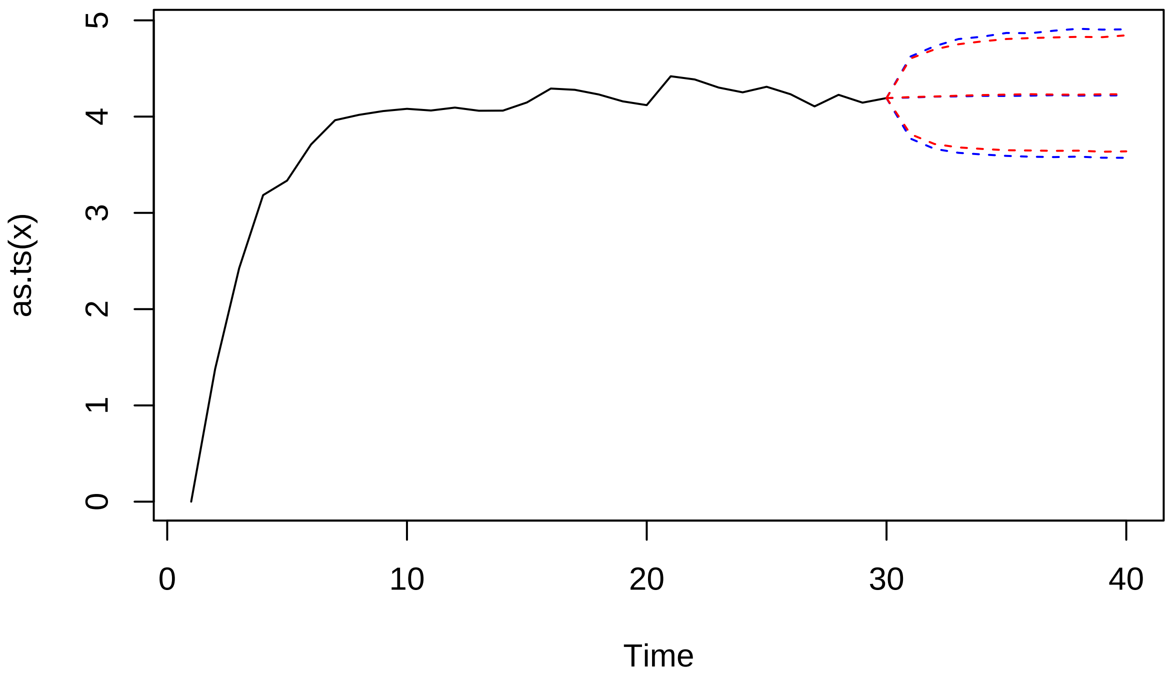
```
prdat2 <- c(dat2, list(cf = cf, Vinv = solve(V)))
pred2 <- jags.fit(prdat2, "x[31:40,1]", pmodel)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 30
##    Unobserved stochastic nodes: 11
##    Total graph size: 488
##
## Initializing model
```

```
summary(pred2)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## x[31,1] 4.204 0.2012 0.001643       0.001685
## x[32,1] 4.211 0.2500 0.002041       0.002097
## x[33,1] 4.217 0.2715 0.002217       0.002305
## x[34,1] 4.224 0.2839 0.002318       0.002611
## x[35,1] 4.228 0.2941 0.002402       0.002703
## x[36,1] 4.233 0.2985 0.002437       0.002863
## x[37,1] 4.231 0.2997 0.002447       0.002842
## x[38,1] 4.234 0.3021 0.002466       0.002812
## x[39,1] 4.233 0.3037 0.002480       0.003131
## x[40,1] 4.233 0.3054 0.002493       0.002972
##
## 2. Quantiles for each variable:
##
##           2.5%   25%   50%   75% 97.5%
## x[31,1] 3.817 4.069 4.203 4.339 4.604
## x[32,1] 3.715 4.044 4.209 4.380 4.698
## x[33,1] 3.680 4.035 4.217 4.400 4.752
## x[34,1] 3.664 4.032 4.224 4.414 4.782
## x[35,1] 3.650 4.033 4.228 4.422 4.806
## x[36,1] 3.648 4.031 4.233 4.433 4.816
## x[37,1] 3.644 4.029 4.229 4.432 4.823
## x[38,1] 3.646 4.031 4.228 4.436 4.829
## x[39,1] 3.635 4.030 4.232 4.434 4.825
## x[40,1] 3.639 4.026 4.231 4.435 4.844
```

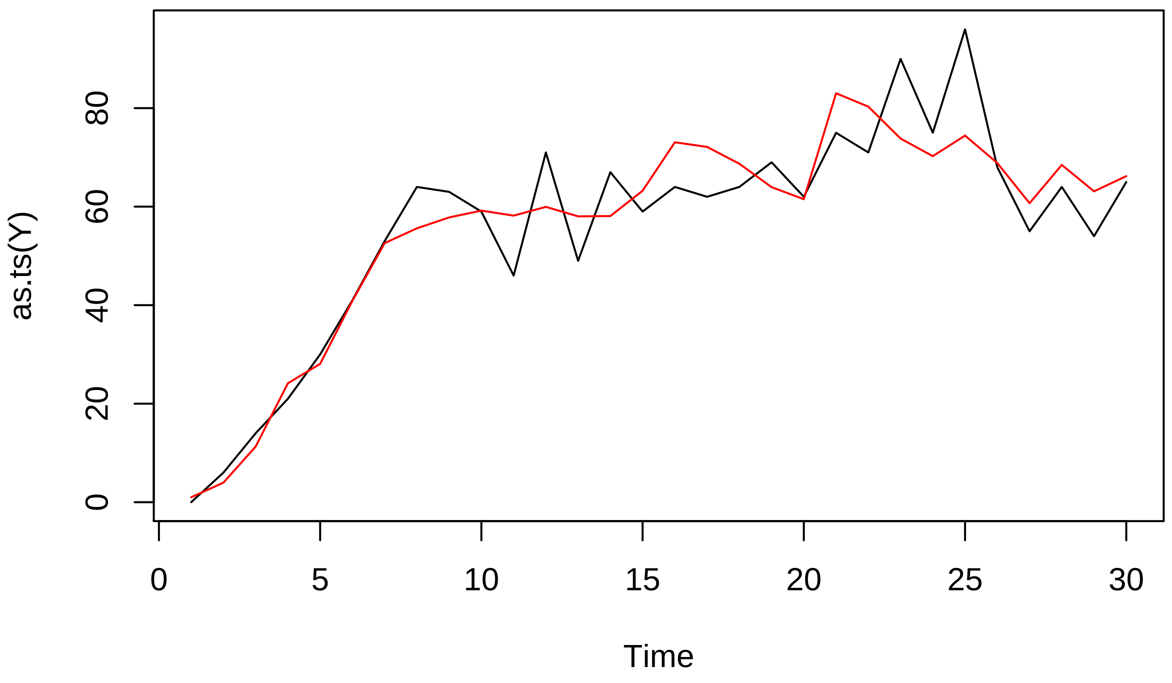
```r
pi2 <- quantile(pred2, probs = c(0.025, 0.5, 0.975))

plot(as.ts(x), xlim=c(1,40), ylim = range(c(x, pi1, pi2)))
matlines(30:40, rbind(x[c(30, 30, 30)], t(pi1)), col=4, lty=2)
matlines(30:40, rbind(x[c(30, 30, 30)], t(pi2)), col=2, lty=2)
```



### Gompertz model with observation error

Using Poisson distribution for observation error

```r
set.seed(1234)
a <- 1.5
b <- -0.35
sigma_sq <- 0.01
T <- 30
x <- numeric(T)
x[1] <- log(1)
for (t in 2:T)
    x[t] <- x[t - 1] + a + b * x[t - 1] + rnorm(1, 0, sqrt(sigma_sq))
N <- exp(x)
Y <- rpois(T, lambda = N)
plot(as.ts(Y))
lines(as.ts(N), col=2)
```



```r
model <- custommodel("model {
    for (k in 1:K) {
        Y[1,k] ~ dpois(exp(x[1,k]))
        x[1,k] ~ dnorm(a / (1 - b), (1 - b^2) / sigma_sq)
        for (t in 2:T) {
            Y[t,k] ~ dpois(exp(x[t,k]))
            x[t,k] ~ dnorm(mu[t,k], 1 / sigma_sq)
            mu[t,k] <- a + (1 + b) * x[t - 1,k]
        }
    }
    a ~ dnorm(0, 0.01)
    b ~ dunif(-0.999, -0.001)
    z <- 0.5 * log((1 + b) / (1 - b))
    sigma_sq <- exp(log_sigma)^2
    log_sigma ~ dnorm(0, 0.01)
}")
```

```r
dat <- list(Y = data.matrix(Y), K = 1, T = T)
## no need to monitor anything related to observation error
## because Poisson variance = mean
fit <- jags.fit(dat, c("a", "b", "sigma_sq"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 30
##    Unobserved stochastic nodes: 33
##    Total graph size: 299
##
## Initializing model
```

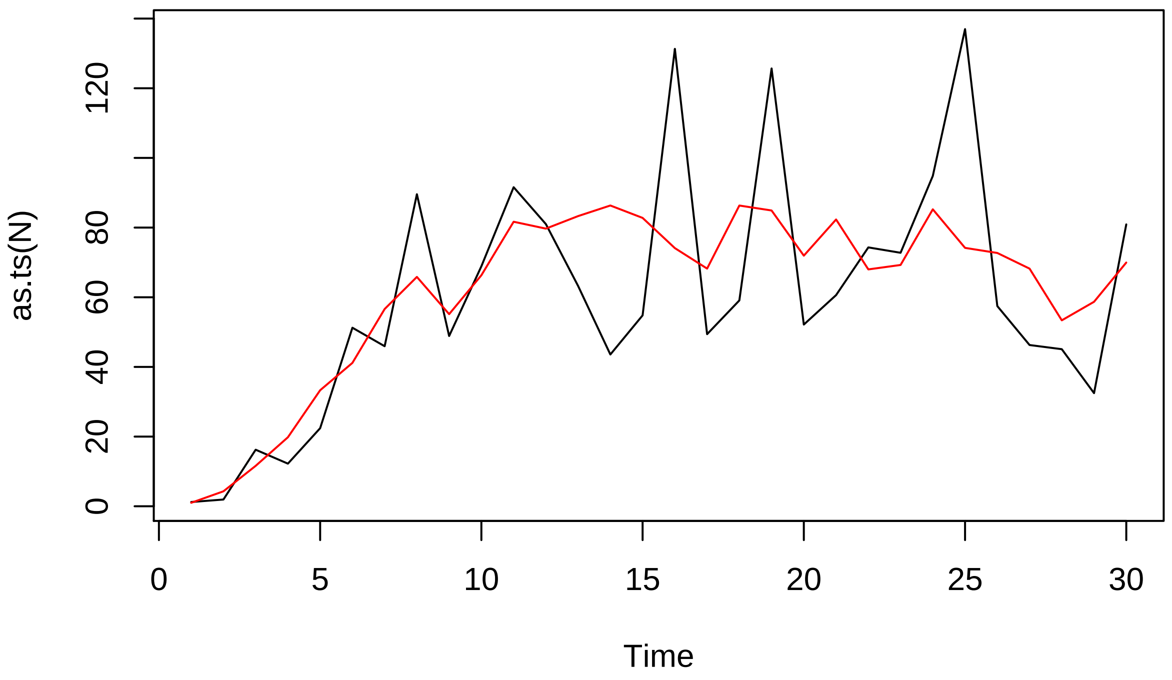
```r
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a         1.290379 0.093676 7.649e-04       0.007870
## b        -0.306690 0.023755 1.940e-04       0.002010
## sigma_sq  0.006728 0.005799 4.735e-05       0.000266
##
## 2. Quantiles for each variable:
##
##                 2.5%       25%       50%       75%    97.5%
## a         1.1148553  1.225980  1.288242  1.354982  1.48640
## b        -0.3569280 -0.322929 -0.305979 -0.290506 -0.26230
## sigma_sq  0.0000636  0.002746  0.005408  0.009086  0.02153
```

Normal error: this comes with an additional parameter

```r
set.seed(32123)
a <- 1.5
b <- -0.35
sigma_sq <- 0.01
tau_sq <- 0.1
T <- 30
x <- numeric(T)
x[1] <- log(1)
for (t in 2:T)
    x[t] <- x[t - 1] + a + b * x[t - 1] + rnorm(1, 0, sqrt(sigma_sq))
Y <- rnorm(T, mean = x, sd = sqrt(tau_sq))
N <- exp(Y)
```

```r
plot(as.ts(N))
lines(as.ts(exp(x)), col=2)
```



```r
model <- custommodel("model {
    for (k in 1:K) {
        Y[1,k] ~ dnorm(x[1,k], 1 / tau_sq)
        x[1,k] ~ dnorm(a / (1 - b), (1 - b^2) / sigma_sq)
        for (t in 2:T) {
            Y[t,k] ~ dnorm(x[t,k], 1 / tau_sq)
            x[t,k] ~ dnorm(mu[t,k], 1 / sigma_sq)
            mu[t,k] <- a + (1 + b) * x[t - 1,k]
        }
    }
    a ~ dnorm(0, 0.01)
    b ~ dunif(-0.999, -0.001)
    z <- 0.5 * log((1 + b) / (1 - b))
    sigma_sq <- exp(log_sigma)^2
    log_sigma ~ dnorm(0, 0.01)
    tau_sq <- exp(log_tau)^2
    log_tau ~ dnorm(0, 0.01)
}")
dat <- list(Y = data.matrix(Y), K = 1, T = T)
fit <- jags.fit(dat, c("a", "b", "sigma_sq", "tau_sq"), model)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 30
##    Unobserved stochastic nodes: 34
##    Total graph size: 335
##
## Initializing model
```

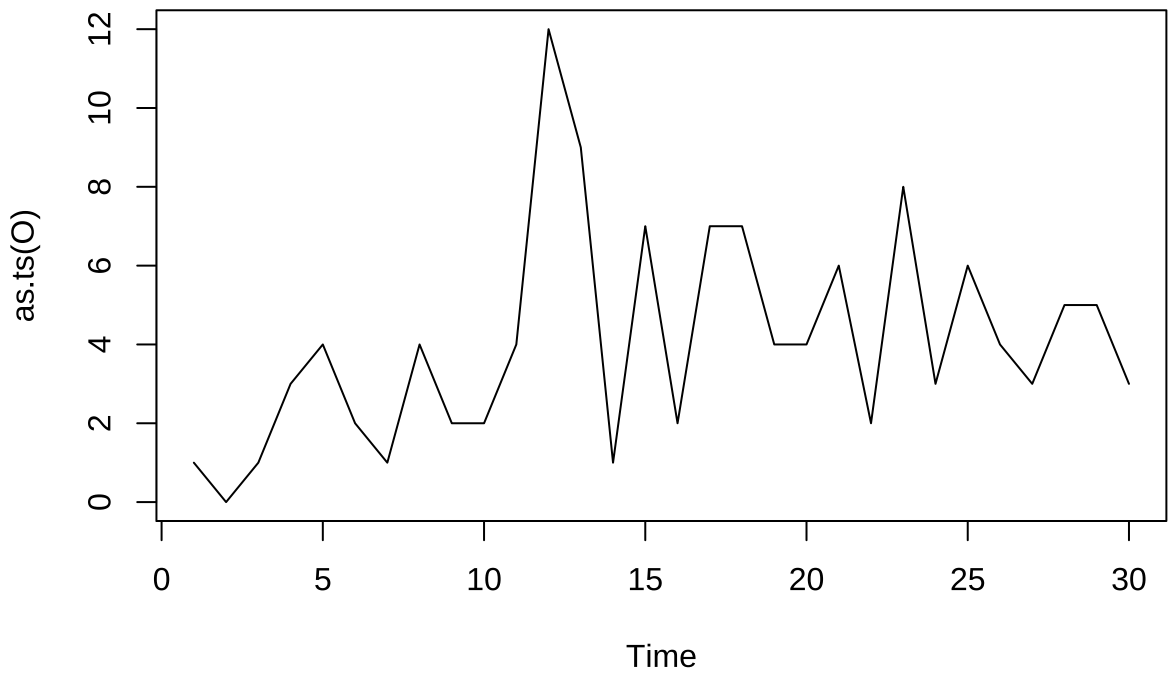
```
summary(fit)
```

```
##
## Iterations = 2001:7000
## Thinning interval = 1
## Number of chains = 3
## 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
## a         1.14566 0.18378 0.0015006       0.024657
## b        -0.27061 0.04743 0.0003873       0.006576
## sigma_sq  0.04316 0.08171 0.0006672       0.011162
## tau_sq    0.17403 0.08140 0.0006646       0.007722
##
## 2. Quantiles for each variable:
##
##                 2.5%        25%       50%      75%   97.5%
## a         8.806e-01  1.080e+00  1.089993  1.19754  1.6857
## b        -4.080e-01 -2.790e-01 -0.255866 -0.25502 -0.2037
## sigma_sq  7.234e-13  3.107e-09  0.002338  0.04498  0.3005
## tau_sq    1.438e-04  1.320e-01  0.175874  0.22099  0.3398
```

## Compare predictive distributions under different parametrizations of Ricker model

*a*-*b* vs. *r*-*K* (logistic) parametrization ($r = a$, $K = -a/b$)

```
set.seed(2345)
a <- 0.5
b <- -0.1
sigma_sq <- 0.05
T <- 30
x <- numeric(T)
x[1] <- log(1)
for (t in 2:T)
    x[t] <- x[t - 1] + a + b * exp(x[t - 1]) +
        rnorm(1, 0, sqrt(sigma_sq))
O <- rpois(T, lambda = exp(x))
plot(as.ts(O))
```



```r
library(dclone)
model_ab_ppi <- custommodel("model {
    x[1] <- log(O[1])
    N[1] <- O[1]
    for (t in 2:T) {
        x[t] ~ dnorm(mu[t], 1 / sigma_sq)
        mu[t] <- a + b * N[t - 1] + x[t - 1]
        N[t] <- min(exp(x[t]), 10000)
        O[t] ~ dpois(N[t])
    }
    for (i in (T + 1):(T + T2)) {
        x[i] ~ dnorm(mu[i], 1 / sigma_sq)
        mu[i] <- a + b * min(exp(x[i - 1]), 10000) + x[i - 1]
    }
    a ~ dnorm(1, 0.01)
    b ~ dnorm(0, 1)
    sigma_sq <- exp(log_sigma)^2
    log_sigma ~ dnorm(0, 1)
}")
T2 <- 10
dat_ppi <- list(O = O, T = T, T2 = T2)
ppi_ab_b <- jags.fit(dat_ppi, "x", model_ab_ppi)
```

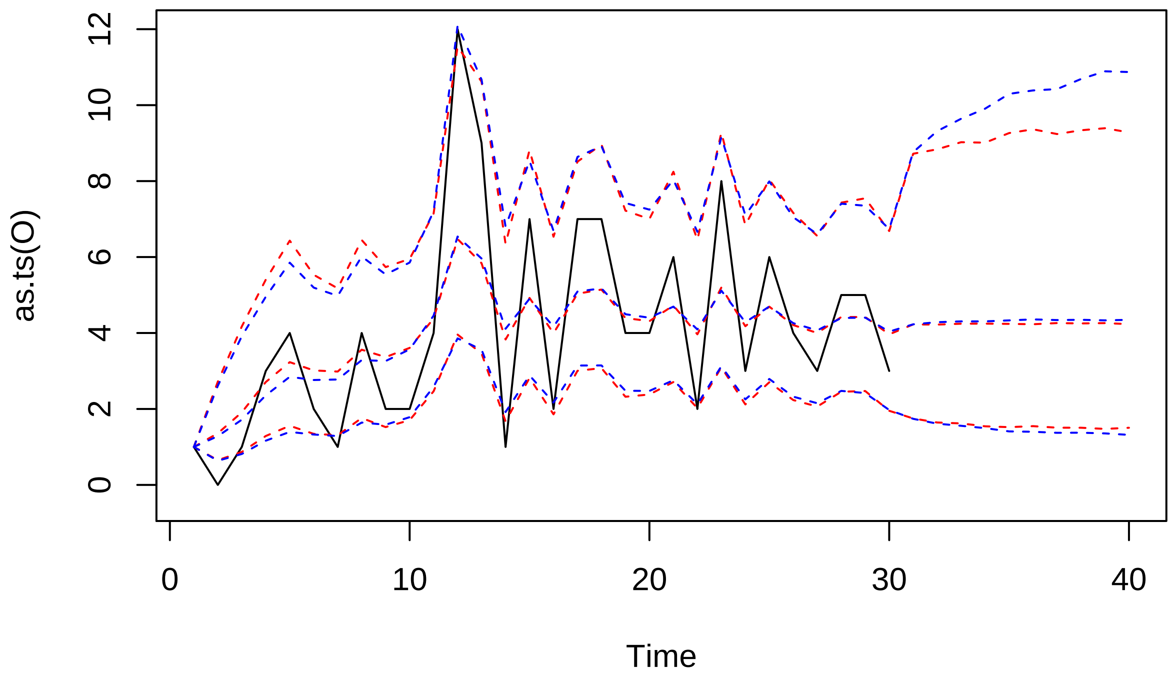
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 42
##    Total graph size: 1816
##
```

```
## Initializing model
```

```
model_rK_ppi <- custommodel("model {
    x[1] <- log(O[1])
    N[1] <- O[1]
    for (t in 2:T) {
        x[t] ~ dnorm(mu[t], 1 / sigma_sq)
        mu[t] <- r * (1 - N[t - 1] / K) + x[t - 1]
        N[t] <- min(exp(x[t]), 10000)
        O[t] ~ dpois(N[t])
    }
    for (i in (T + 1):(T + T2)) {
        x[i] ~ dnorm(mu[i], 1 / sigma_sq)
        mu[i] <- r * (1 - min(exp(x[i - 1]), 10000) / K) + x[i - 1]
    }
    r ~ dnorm(0, 0.1)
    K <- exp(log_K)
    log_K ~ dnorm(0, 0.1)
    sigma_sq <- exp(log_sigma)^2
    log_sigma ~ dnorm(0, 1)
}")
dat_ppi <- list(O = O, T = T, T2 = T2)
ppi_rK_b <- jags.fit(dat_ppi, "x", model_rK_ppi)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 29
##    Unobserved stochastic nodes: 42
##    Total graph size: 2669
##
## Initializing model
```

```
qppi_ab_b <- quantile(ppi_ab_b, probs = c(0.025, 0.5, 0.975))
qppi_rK_b <- quantile(ppi_rK_b, probs = c(0.025, 0.5, 0.975))

plot(as.ts(O), xlim=c(1,40), ylim = range(c(O, qppi_ab_b, qppi_rK_b)))
matlines(1:40, t(exp(qppi_ab_b)), col=2, lty=2)
matlines(1:40, t(exp(qppi_rK_b)), col=4, lty=2)
```



```
#matlines(30:40, rbind(O[c(30, 30, 30)], t(exp(qppi_ab_b[,31:40]))), col=4, lty=2)
#matlines(30:40, rbind(O[c(30, 30, 30)], t(exp(qppi_rK_b[,31:40]))), col=2, lty=2)
```