Lab_14_USA

Team USA: Yuting Deng, Kendra Gilbertson, Lily Durkee, Bennett Hardy

Hermione.Deng@colostate.edu, kendra01@colostate.edu, L.Durkee@colostate.edu,Bennett.Hardy@colosta

2022-11-30

Motivation

The Eurasian lynx (*Lynx lynx*) is a medium-sized predator with broad distribution in the boreal forests of Europe and Siberia. The lynx is classified as a threatened species throughout much of its range and there is controversy about the legal harvest of lynx in Sweden. Proponents of harvest argue that allowing hunting of lynx reduces illegal kill (poaching). Moreover, Sweden is committed to regulate lynx numbers to prevent excessive predation on reindeer because reindeer are critical to the livelihoods of indigenous pastoralists, the Sami. Many environmentalists oppose harvest, however, arguing that lynx are too rare to remove their fully protected status. A similar controversy surrounds management of wolves in the Western United States.

A forecasting model for the abundance of lynx helps managers make decisions that can be justified to citizens. The model you will develop today is not a toy. It is currently used in Sweden and Norway to manage Lynx (H. Andren, N. T. Hobbs, M. Aronsson, H. Broseth, G. Chapron, J. D. C. Linnell, J. Odden, J. Persson, and E. B. Nilsen. Harvest models of small populations of a large carnivore using Bayesian forecasting. Ecological Applications, 30(3):e02063, 2020.)

You have data on the number of lynx family groups censused in a managemengt unit as well as annual records of lynx harvested from the unit. You will model the population using the deterministic model:

$$N_t = \lambda(N_{t-1} - H_{t-1})$$

.

where N_t is the true, unobserved abundance of lynx and H_{t-1} is the number of lynx harvested during $t-1$ to t. The parentheses in this expression reflect the fact that harvest occurs immediately after census, such that the next years population increment comes from the post-harvest population size.

ADVANCED (for the population modelers) What would be the model if harvest occurred immediately before census? Three months after census? Continuously throughout the year?

Assume the harvest (H_t) is and the number of family groups (y_t) are observed without error. Harvest is closely regulated and all hunters who harvest a lynx are required by law to register the animal with the county. You are entitled to make the assumption that family groups are observed without error because your Scandinavian colleagues are amazing snow trackers and do a good job of estimating the number of family groups (if not the number of lynx) in a management region. The challenge in this problem is that the observations of lynx abundance (family groups) are not the same as the observation of harvest (number of lynx). Fortunately, you have prior information, hard won from radio-telemetry, on the proportional relationship between number of family groups and number of lynx in the population, i.e:

$$\phi = f/N$$

,

where f is the number of family groups and N is the population size, mean $\phi = 0.163$ with standard deviation of the mean $= 0.012$.

R libraries needed for this lab

You need to load the following libraries. Set the seed to 10 to compare your answers to ours. The data for this problem is located in the LynxFamilies data frame of the BayesNSF package.

```r
library(BayesNSF)
library(rjags)
library(MCMCvis)
library(HDInterval)
set.seed(10)
```

Generating an Informed Prior for ϕ

We've provided you with a useful moment matching function below for converting the mean and standard deviation of ϕ to the parameters for the beta distribution you will use as an informed prior on ϕ.

```r
# Function to get beta shape parameters from moments
shape_from_stats <- function(mu = mu.global, sigma = sigma.global) {
  a <-(mu^2 - mu^3 - mu * sigma^2) / sigma^2
  b <- (mu - 2 * mu^2 + mu^3 - sigma^2 + mu*sigma^2) / sigma^2
  shape_ps <- c(a, b)
  return(shape_ps)
}

# get parameters for distribution of population multiplier, 1/p
shapes = shape_from_stats(.163, .012)

# check prior on p using simulated data from beta distribution
x = seq(0, 1, .001)
p = dbeta(x, shapes[1], shapes[2])
plot(x, p, typ = "l", xlim = c(0, 1))
```

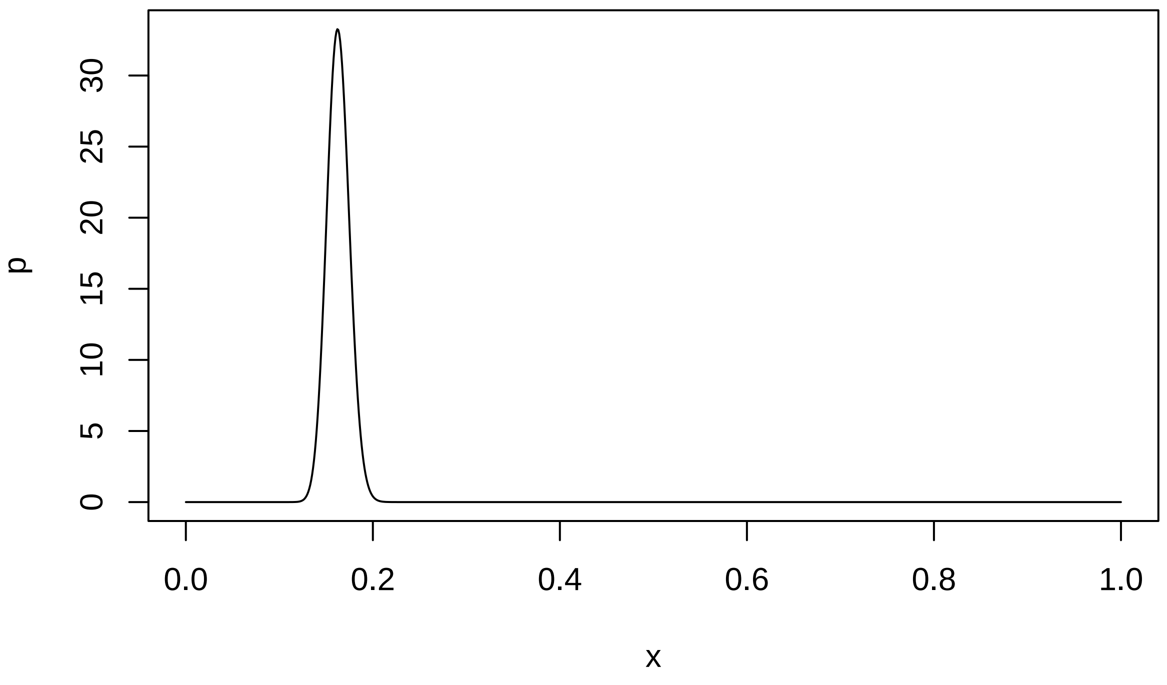
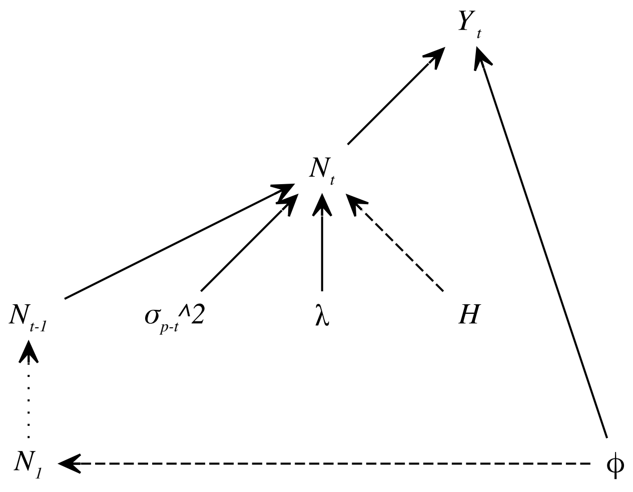


Diagram the Bayesian network

1. Develop a hierarchical Bayesian model (also called a state space model) of the lynx population in the management unit. Diagram the Bayesian network (the DAG) of knowns and unknowns and write out the posterior and factored joint distribution. Use a lognormal distribution to model the true lynx population size over time. Use a Poisson distribution for the data model relating the true, unobserved state (the total population size) to the observed data (number of family groups).



2. An alternative approach, which is slightly more difficult to code, is to model the process as:

$$\text{negative binomial}(N_t \mid \lambda(N_{t-1} - H_{t-1}, \rho))$$

,

and model the data as:

$$\text{binomial}(y_t \mid \text{round}(N_t\phi), p)$$

,

where p is a detection probability. Explain why this second formulation might be better than the formulation you are using. (It turns out they give virtually identical results.)

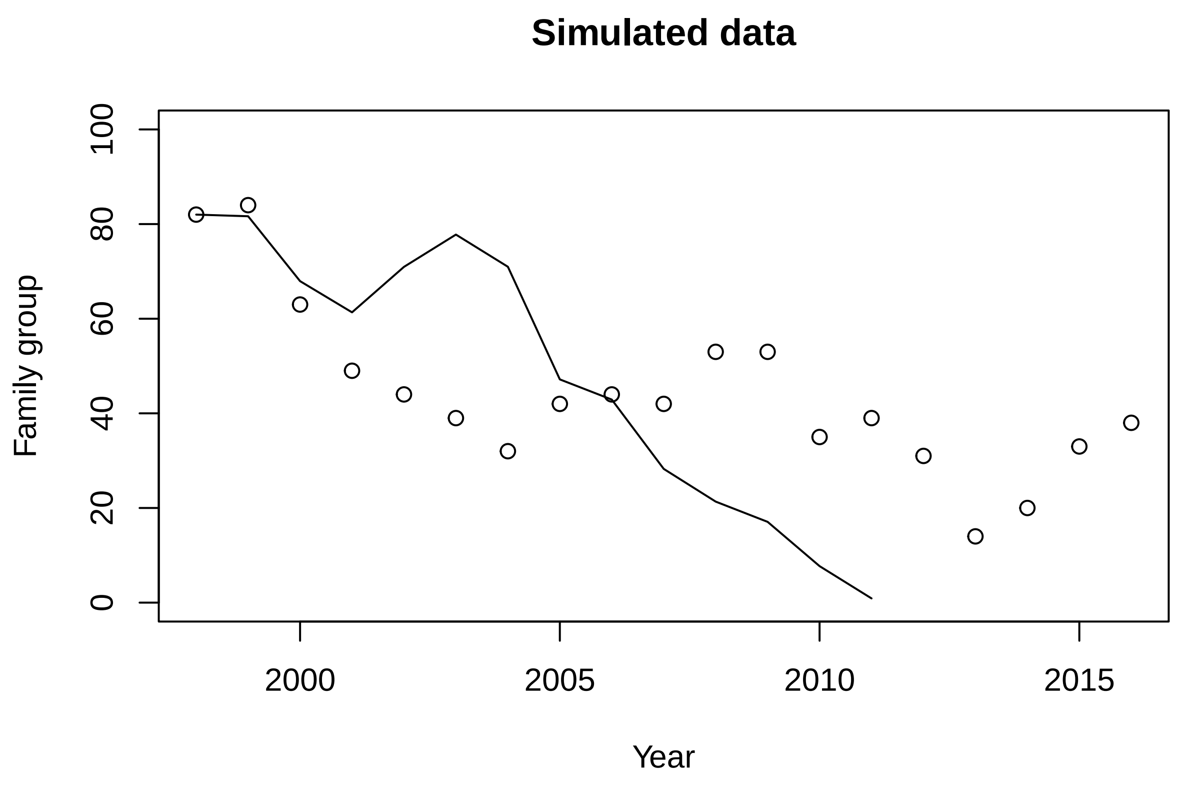
ANSWER

Fitting the Model

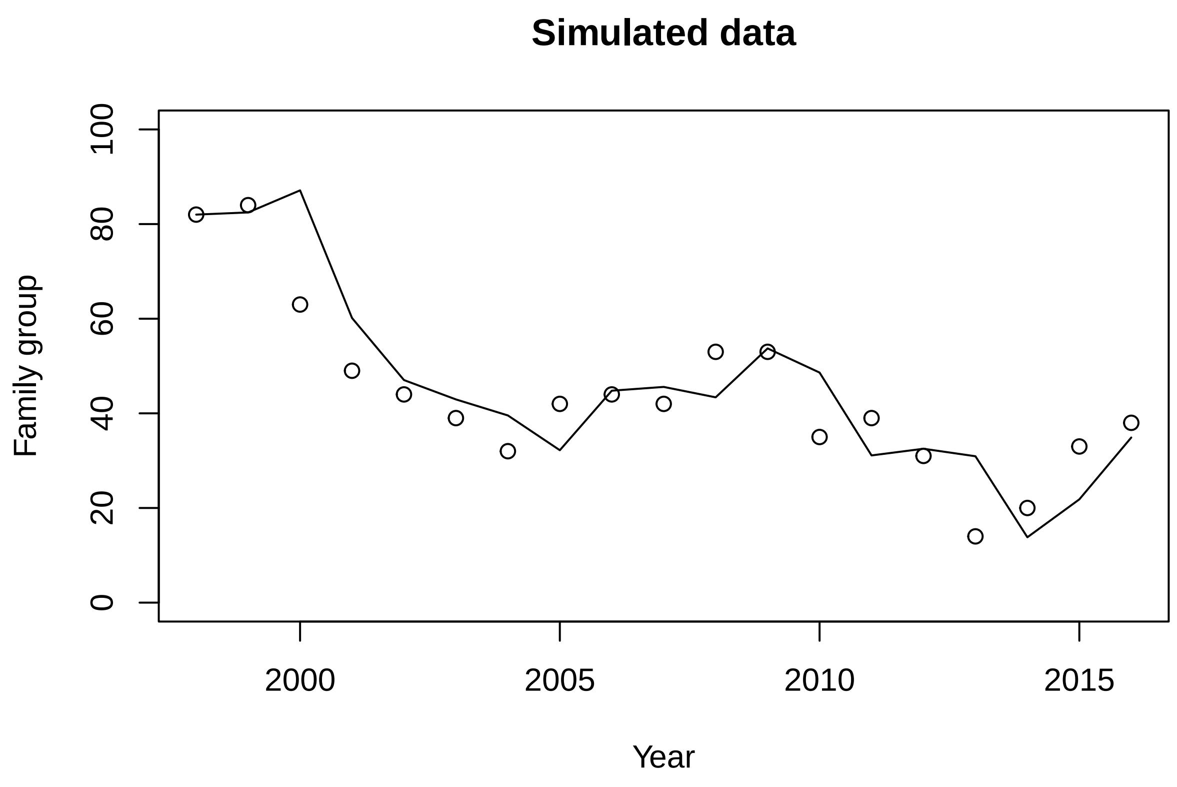
Now you'll estimate the marginal posterior distribution of the unobserved, true state over time ($\mathbf{N}$), the parameters in the model λ and ϕ as well as the process variance and observation variance. You'll also summarize the marginal posterior distributions of the parameters and unobserved states. A note about the data. Each row in the data file gives the observed number of family groups for that year in column 2 and that year's harvest in column 3. The harvest in each row influences the population size in the next row. So, for example, the 2016 harvest influences the 2017 population size.

Before you begin it's very helpful to use simulated data to the verify initial values and model. We simulate the true state by choosing some biologically reasonable values for model parameters and "eyeballing" the fit of the true state to the data. You can then use these simulated values for initial conditions (see the inits list below). This is of particular importance because failing to give reasonable initial conditions for dynamic models can cause problems in model fitting. Remember, supply initial conditions for *all* unobserved quantities in the posterior distribution (even those that do not have priors).

```r
y <- LynxFamilies
endyr <- nrow(y)
n <- numeric(endyr + 1)
mu <- numeric(endyr + 1)
fg <- numeric(endyr + 1)
phi <- 0.16
lambda <- 1.07
sigma.p <- 0.2

n[1] <- y$census[1] / phi # n in the unit of individuals
mu[1] <- n[1] # mean from deterministic model to simulate
fg[1] <- n[1] * phi # Nt in the unit of

for (t in 2:(endyr + 1)) {
  mu[t] <- lambda * (n[t - 1] - y$harvest[t - 1])
  n[t] <- rlnorm(1, log(mu[t]), sigma.p)
  fg[t] <- n[t] * phi
}

plot(y$year, y$census, ylim = c(0, 100), xlab = "Year", ylab = "Family group",
     main = "Simulated data")
lines(y$year, fg[1:length(y$year)])
```



```r
## visually match simulated data with observations for initial conditions
endyr = nrow(y)
n = numeric(endyr + 1)
mu = numeric(endyr + 1) #use this for family groups
lambda = 1.1
sigma.p = .00001
n[1] = y$census[1]

for(t in 2:(endyr + 1)) {
  n[t] <- lambda * (y$census[t - 1] - .16 * y$harvest[t - 1])  # use this for family groups
}

plot(y$year, y$census, ylim = c(0, 100), xlab = "Year", ylab = "Family group",
     main = "Simulated data")
lines(y$year, n[1:length(y$year)])
```



Here's your starting code:

```
data = list(
    y.endyr = endyr,
    y.a = shapes[1],
    y.b = shapes[2],
    y.H = y$harvest,
    y = y$census)

inits = list(
    list(lambda = 1.2, sigma.p = .01, N = n),
    list(lambda = 1.01,sigma.p = .2, N = n * 1.2),
    list(lambda = .95, sigma.p = .5, N = n * .5))
```

1. Write the JAGS model to estimate the marginal posterior distribution of the unobserved, true state over time (**N**), **the parameters in the model λ and ϕ as well as the process variance and observation variance. Include a summary the marginal posterior distributions of the parameters and unobserved states.**

```
{
sink("lynxmodel.R")
cat("
model{


```

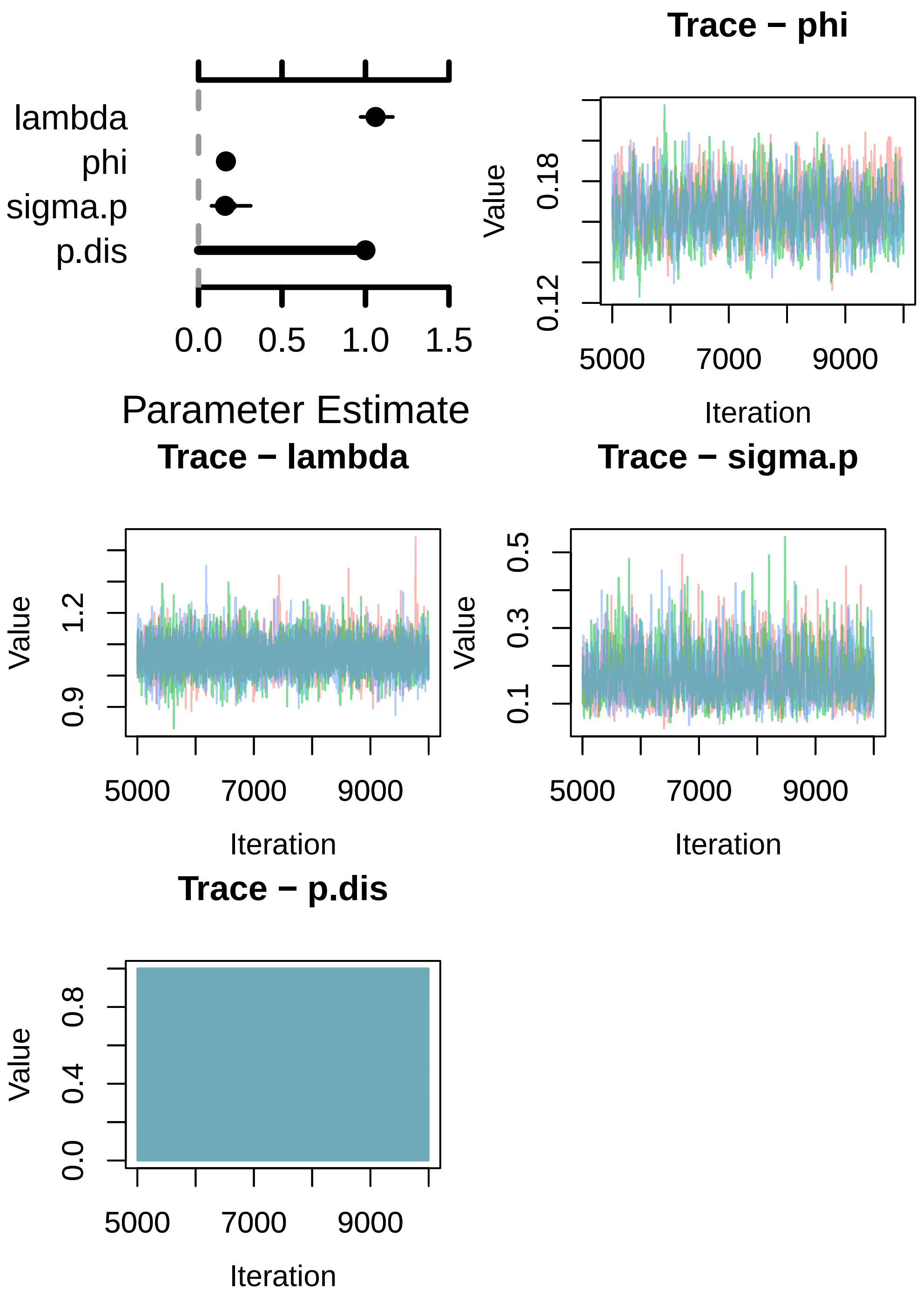
```
# Priors

phi ~ dbeta(y.a, y.b)
lambda ~ dgamma(.5, .001)
sigma.p ~ dunif(0,100)
tau.p <- 1/sigma.p^2
fg[1] ~ dpois(y[1])
N[1] ~ dlnorm(log(y[1] / phi), tau.p)


# Likelihood

for(t in 2:y.endyr){
# Data
  y[t] ~ dpois(phi * N[t])
}


# Process

for(t in 2:(y.endyr+1)){
  log.mu[t] <- log(max(0.0001,lambda * (N[t-1] - y.H[t-1])))
  N[t] ~ dlnorm(log.mu[t], tau.p)
  fg[t] <- N[t] * phi
}


# Bayesian p values

 for(t in 1:y.endyr){
  y.sim[t] ~ dpois(phi * N[t])
  sq.data[t] <- (y[t] - (phi*N[t]))^2
  sq.sim[t] <- (y.sim[t] - (phi*N[t]))^2
 }

# Predictive check for discrepancy
 dis.data <- sum(sq.data)
 dis.sim <- sum(sq.sim)
 p.dis <- step(dis.sim - dis.data)    # calcualte Bayesian test statistic

# Derived quantity to examine autocorrelation in residuals
  for(t in 2:y.endyr){
   e[t] <- y[t] - N[t]
   }

    }
      ", fill = TRUE)
  sink()
}
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
```

```
##    Observed stochastic nodes: 18
##    Unobserved stochastic nodes: 43
##    Total graph size: 309
##
## Initializing model
```

2. Check MCMC chains for model parameters, process variance, and latent states for convergence. This will probably require using the excl option in MCMCsummary.

```
##              mean      sd      2.5%       50%     97.5% Rhat n.eff
## N[1]     498.5102 78.8302  369.8333  490.0456  679.2531 1.00  2587
## N[2]     471.0153 56.8983  370.7223  467.1786  594.2512 1.01  1928
## N[3]     390.5288 44.8662  309.5329  387.9613  485.2600 1.01  2010
## N[4]     314.1365 38.8515  242.9704  312.2157  395.6017 1.01  1974
## N[5]     272.7012 34.4648  209.1106  271.2013  344.6715 1.01  1963
## N[6]     243.5696 32.3595  184.5179  241.8920  311.7830 1.01  2117
## N[7]     229.3634 31.2023  172.3762  227.8653  294.3737 1.00  2009
## N[8]     248.1195 32.5828  190.1056  245.9645  318.6538 1.00  2041
## N[9]     266.4227 33.8216  206.7627  264.5780  338.3328 1.00  2015
## N[10]    273.6805 34.0243  211.9451  271.9310  344.6694 1.00  2057
## N[11]    303.7044 37.2782  237.8119  301.1415  384.2679 1.00  2123
## N[12]    299.8663 36.1953  237.3985  297.0393  379.2211 1.00  2182
## N[13]    241.2155 30.1371  186.5551  239.7199  304.5715 1.00  2109
## N[14]    220.4485 27.9768  171.8551  218.4779  282.0419 1.01  2041
## N[15]    165.3553 24.2740  121.8874  164.0078  216.7763 1.01  1948
## N[16]    136.1992 22.2506   94.6656  135.5919  181.3177 1.01  2000
## N[17]    146.0732 22.5008  104.2602  145.2400  192.8278 1.01  2018
## N[18]    184.6193 26.6441  137.8896  182.6077  242.9341 1.00  2298
## N[19]    207.2283 33.4968  149.3377  204.2388  281.2687 1.00  3073
## N[20]    182.7110 52.5947  104.1715  174.3972  308.3178 1.00  5553
## e[2]    -387.0153 56.8983 -510.2512 -383.1786 -286.7223 1.01  1928
## e[3]    -327.5288 44.8662 -422.2600 -324.9613 -246.5329 1.01  2010
## e[4]    -265.1365 38.8515 -346.6017 -263.2157 -193.9704 1.01  1974
## e[5]    -228.7012 34.4648 -300.6715 -227.2013 -165.1106 1.01  1963
## e[6]    -204.5696 32.3595 -272.7830 -202.8920 -145.5179 1.01  2117
## e[7]    -197.3634 31.2023 -262.3737 -195.8653 -140.3762 1.00  2009
## e[8]    -206.1195 32.5828 -276.6538 -203.9645 -148.1056 1.00  2041
## e[9]    -222.4227 33.8216 -294.3328 -220.5780 -162.7627 1.00  2015
## e[10]   -231.6805 34.0243 -302.6694 -229.9310 -169.9451 1.00  2057
## e[11]   -250.7044 37.2782 -331.2679 -248.1415 -184.8119 1.00  2123
## e[12]   -246.8663 36.1953 -326.2211 -244.0393 -184.3985 1.00  2182
## e[13]   -206.2155 30.1371 -269.5715 -204.7199 -151.5551 1.00  2109
## e[14]   -181.4485 27.9768 -243.0419 -179.4779 -132.8551 1.01  2041
## e[15]   -134.3553 24.2740 -185.7763 -133.0078  -90.8874 1.01  1948
## e[16]   -122.1992 22.2506 -167.3177 -121.5919  -80.6656 1.01  2000
## e[17]   -126.0732 22.5008 -172.8278 -125.2400  -84.2602 1.01  2018
## e[18]   -151.6193 26.6441 -209.9341 -149.6077 -104.8896 1.00  2298
## e[19]   -169.2283 33.4968 -243.2687 -166.2388 -111.3377 1.00  3073
## lambda     1.0623  0.0478    0.9708    1.0604    1.1642 1.00  9658
## p.dis      0.5814  0.4933    0.0000    1.0000    1.0000 1.00 18461
## phi        0.1645  0.0121    0.1417    0.1642    0.1892 1.01  1014
## sigma.p    0.1684  0.0615    0.0775    0.1590    0.3127 1.00  1756
```



3. Conduct posterior predictive checks by simulating a new dataset for family groups (f_t) at every MCMC iteration. Calculate a Bayesian p value using the sums of squared discrepancy between the observed and the predicted number of family groups based on observed and simulated data,

$$T^{observed} = \sum_{t=1}^{n} (f_t^{observed} - N_t \phi)^2$$

$$T^{model} = \sum_{t=1}^{n} (f_t^{simulated} - N_t\phi)^2$$

.

The Bayesian p value is the proportion of MCMC iterations for which $T_{model} > T_{obs}$.

Assure yourself that the process model adequately accounts for temporal autocorrelation in the residuals—allowing the assumption that they are independent and identically distributed. To do this, include a derived quantity

$$e_t = y_t - N_t\phi$$

,

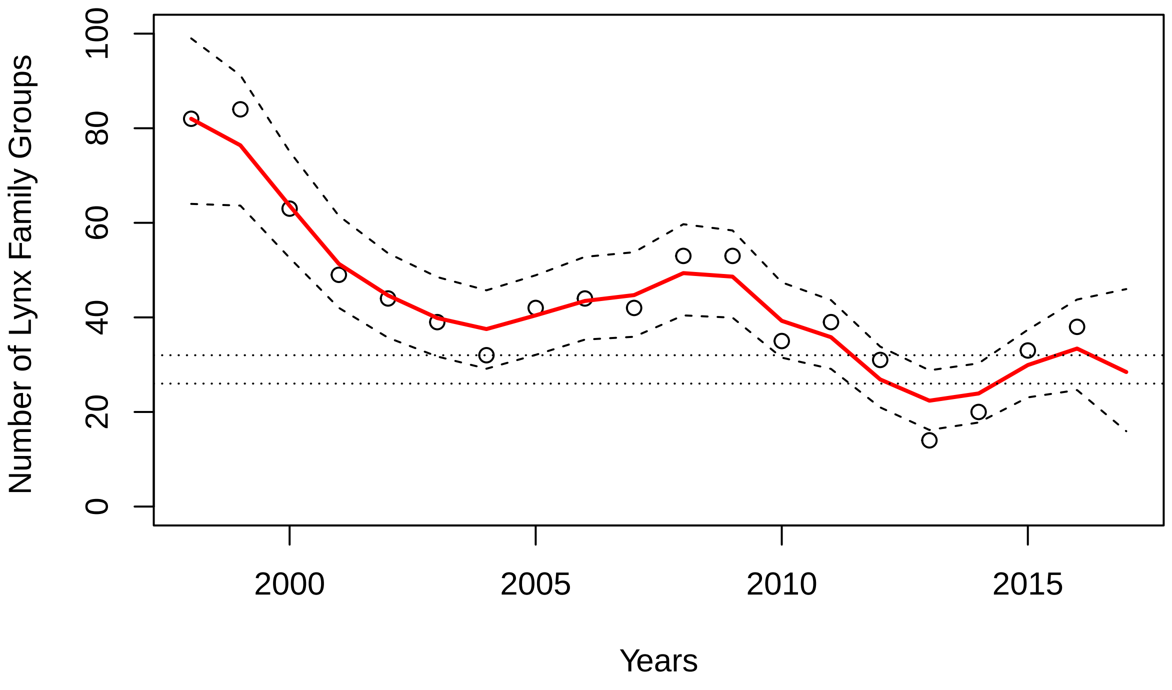
in your JAGS code and coda object. Use the following code or something like it to examine how autocorrelation in the residuals changes with time lag.

```r
acf(unlist(MCMCpstr(z, param = "e", func = mean)), main = "", lwd = 3, ci = 0)
```

4. Write a paragraph describing how to interpret the plot produced by this function.

ANSWER

5. Plot the median of the marginal posterior distribution of the number of lynx family groups over time (1998-2016) including a highest posterior density interval. Include your forecast for 2017 (the predictive process distribution) in this plot.



Code

```r
knitr::opts_chunk$set(
    echo = FALSE,
    message = FALSE,
    warning = FALSE,
    attr.source = ".numberLines"
)
library(BayesNSF)
library(rjags)
library(MCMCvis)
library(HDInterval)
set.seed(10)
library(BayesNSF)
library(rjags)
library(MCMCvis)
library(HDInterval)
set.seed(10)
# Function to get beta shape parameters from moments
shape_from_stats <- function(mu = mu.global, sigma = sigma.global) {
  a <-(mu^2 - mu^3 - mu * sigma^2) / sigma^2
  b <- (mu - 2 * mu^2 + mu^3 - sigma^2 + mu*sigma^2) / sigma^2
  shape_ps <- c(a, b)
  return(shape_ps)
}

# get parameters for distribution of population multiplier, 1/p
shapes = shape_from_stats(.163, .012)

# check prior on p using simulated data from beta distribution
x = seq(0, 1, .001)
p = dbeta(x, shapes[1], shapes[2])
plot(x, p, typ = "l", xlim = c(0, 1))
DiagrammeR::grViz("
      digraph mrdag {
      graph [rankdir=TB, layout=neato]

      node [shape=plaintext, height=0.3, width=0.3]
      Y     [label=<<I>Y@_{t}</I>>, pos='3,1!']

      N     [label=<<I>N@_{t}</I>>, pos='2,0!']

      phi   [label='&phi;', pos='4,-2!']

      H     [label=<<I>H</I>>, pos='3,-1!']

      lambda    [label='&lambda;', pos='2,-1!']

      sigma    [label='<I>&sigma;@_{p-t}^2</I>', pos='1,-1!']

      Nt1     [label=<<I>N@_{t-1}</I>>, pos='0,-1!']

      N1     [label=<<I>N@_{1}</I>>, pos='-0,-2!']

      edge [arrowhead='vee']
      N -> Y
      phi -> Y
      sigma -> N
      lambda -> N
      Nt1 -> N

      H -> N      [style=dashed];
      N1 -> Nt1  [style=dotted];
      phi -> N1 [style=dashed];
      }
      ", height = 190)
y <- LynxFamilies
endyr <- nrow(y)
n <- numeric(endyr + 1)
mu <- numeric(endyr + 1)
fg <- numeric(endyr + 1)
phi <- 0.16
lambda <- 1.07
sigma.p <- 0.2

n[1] <- y$census[1] / phi # n in the unit of individuals
mu[1] <- n[1] # mean from deterministic model to simulate
fg[1] <- n[1] * phi # Nt in the unit of

for (t in 2:(endyr + 1)) {
  mu[t] <- lambda * (n[t - 1] - y$harvest[t - 1])
  n[t] <- rlnorm(1, log(mu[t]), sigma.p)
  fg[t] <- n[t] * phi
}

plot(y$year, y$census, ylim = c(0, 100), xlab = "Year", ylab = "Family group",
     main = "Simulated data")
lines(y$year, fg[1:length(y$year)])
## visually match simulated data with observations for initial conditions
endyr = nrow(y)
n = numeric(endyr + 1)
mu = numeric(endyr + 1) #use this for family groups
lambda = 1.1
sigma.p = .00001
n[1] = y$census[1]

for(t in 2:(endyr + 1)) {
  n[t] <- lambda * (y$census[t - 1] - .16 * y$harvest[t - 1])  # use this for family groups
}

plot(y$year, y$census, ylim = c(0, 100), xlab = "Year", ylab = "Family group",
     main = "Simulated data")
lines(y$year, n[1:length(y$year)])
data = list(
    y.endyr = endyr,
    y.a = shapes[1],
    y.b = shapes[2],
    y.H = y$harvest,
    y = y$census)

inits = list(
    list(lambda = 1.2, sigma.p = .01, N = n),
    list(lambda = 1.01,sigma.p = .2, N = n * 1.2),
    list(lambda = .95, sigma.p = .5, N = n * .5))
{
sink("lynxmodel.R")
cat("
model{


# Priors

phi ~ dbeta(y.a, y.b)
lambda ~ dgamma(.5, .001)
sigma.p ~ dunif(0,100)
tau.p <- 1/sigma.p^2
fg[1] ~ dpois(y[1])
N[1] ~ dlnorm(log(y[1] / phi), tau.p)


# Likelihood

for(t in 2:y.endyr){
# Data
  y[t] ~ dpois(phi * N[t])
}


# Process

for(t in 2:(y.endyr+1)){
  log.mu[t] <- log(max(0.0001,lambda * (N[t-1] - y.H[t-1])))
  N[t] ~ dlnorm(log.mu[t], tau.p)
  fg[t] <- N[t] * phi
}


# Bayesian p values

 for(t in 1:y.endyr){
  y.sim[t] ~ dpois(phi * N[t])
  sq.data[t] <- (y[t] - (phi*N[t]))^2
  sq.sim[t] <- (y.sim[t] - (phi*N[t]))^2
 }

# Predictive check for discrepancy
 dis.data <- sum(sq.data)
 dis.sim <- sum(sq.sim)
 p.dis <- step(dis.sim - dis.data)    # calcualte Bayesian test statistic

# Derived quantity to examine autocorrelation in residuals
  for(t in 2:y.endyr){
   e[t] <- y[t] - N[t]
   }

    }
      ", fill = TRUE)
  sink()
}
# Setup MCMC chain
n.adapt = 3000
n.update = 10000
n.iter = 10000

jm = jags.model("lynxmodel.R",
                data = data, inits = inits, n.chains = length(inits), n.adapt = n.adapt)
update(jm, n.iter = n.update)

zm = coda.samples(jm, variable.names = c("lambda", "phi", "sigma.p", "N", "p.dis", "e"),
                  n.iter = n.iter, n.thin = 1)
MCMCsummary(zm, round = 4, n.eff = TRUE)
# caterpillar plot
MCMCplot(zm, params=c("lambda","phi", "sigma.p", "p.dis"))

# trace plots
MCMCtrace(zm, params = 'phi', type = 'trace', pdf = FALSE)
MCMCtrace(zm, params = 'lambda', type = 'trace', pdf = FALSE)
MCMCtrace(zm, params = 'sigma.p', type = 'trace', pdf = FALSE)
MCMCtrace(zm, params = 'p.dis', type = 'trace', pdf = FALSE)


acf(unlist(MCMCpstr(z, param = "e", func = mean)), main = "", lwd = 3, ci = 0)
zm1 <- coda.samples(jm, variable.names = c("N","fg"), n.iter = n.iter, n.thin = 1)
bound <- MCMCpstr(zm1, params = 'fg', func = function(x) hdi(x,0.95))

years <- seq(1998,2017,1)
plot(y$year, y$census, ylab = "Number of Lynx Family Groups ", xlab = "Years",
     xlim = c(1998,2017), ylim = c(0,100))
lines(years, MCMCpstr(zm1, params = 'fg', func = median)$fg, col="red", lwd = 2)
lines(bound$fg[,1] ~ years, lty=2)
lines(bound$fg[,2] ~ years, lty=2)
abline(h=26, lty=3)
abline(h=32, lty=3)
# this R markdown chunk generates a code appendix
```