

# ESS 575: JAGS Problems Lab

Team England

09 October, 2022

Team England:

- Caroline Blommel
- Carolyn Coyle
- Bryn Crosby
- George Woolsey

[cblommel@mail.colostate.edu](mailto:cblommel@mail.colostate.edu), [carolynm@mail.colostate.edu](mailto:carolynm@mail.colostate.edu), [brcrosby@rams.colostate.edu](mailto:brcrosby@rams.colostate.edu), [george.woolsey@colostate.edu](mailto:george.woolsey@colostate.edu)

## Setup

Download the R package [BayeNSF ver. 1.1](#) to your computer.

Run:

```
install.packages("<pathtoBayesNSF>/BayesNSF_1.1.tar.gz", repos = NULL, type = "source")
```

## Motivation

JAGS allows you to implement models of high dimension once you master its syntax and logic. It is a great tool for ecological analysis. The problems that follow challenge you to:

- Write joint distributions as a basis for writing JAGS code.
- Write JAGS code to approximate marginal posterior distributions of derived quantities.
- Plot model output in revealing ways.
- Understand the effect of vague priors on parameters and on predictions of non-linear models.

## Derived quantities with the logistic

One of the most useful features of MCMC is its equivariance property which means that any quantity that is a function of a random variable in the MCMC algorithm becomes a random variable. Consider two quantities of interest that are functions of our estimates of the random variables  $r$  and  $K$ :

- The population size where the population growth rate is maximum,  $\frac{K}{2}$
- The rate of population growth,  $\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right)$

You will now do a series of problems to estimate these quantities of interest. Some hints for the problems below:

- Include expressions for each derived quantity in your JAGS code.
- You will need to give JAGS a vector of  $N$  values to plot  $\frac{dN}{dt}$  vs  $N$ .
- Use a JAGS object for plotting the rate of population growth.
- Look into using the `ecdf()` function on a JAGS object. It is covered in the [JAGS Primer](#).

## Question 1

Approximate the marginal posterior distribution of the population size where the population growth rate is maximum and plot its posterior density. You may use the work you have already done in the JAGS Primer to speed this along.

```
#####
# insert JAGS model code into an R script
#####
{ # Extra bracket needed only for R markdown files - see answers
  sink("LogisticJAGS.R") # This is the file name for the jags code
  cat("
  ## Logistic example for Primer
  model{
    # priors
    K ~ dunif(0, 4000) # dunif(alpha = lower limit, beta = upper limit)
    r ~ dunif(0, 2) # dunif(alpha, beta)
    sigma ~ dunif(0, 2) # dunif(alpha, beta)
    tau <- 1/sigma^2
    # likelihood
    for(i in 1:n){
      mu[i] <- r - r/K * x[i]
      y[i] ~ dnorm(mu[i], tau) # dnorm(mu,tau)
    }
    ## quantities of interest
    # population size where the population growth rate is maximum
    N_max_pop_grwth_rt <- K/2
    # The rate of population growth
    for(j in 1:length(N)){
      pop_grwth_rt[j] <- r * N[j] * (1 - ( N[j] / K ))
    }
  }
  ", fill = TRUE)
  sink()
}
#####
# implement model
#####
# SESYNCBayes which has the data frame Logistic, which we then order by PopulationSize
# Logistic = SESYNCBayes::Logistic[order(Logistic$PopulationSize),]
Logistic = BayesNSF::Logistic %>% dplyr::arrange(PopulationSize)
# specify the initial conditions for the MCMC chain
inits = list(
  list(K = 1500, r = .2, sigma = 1),
  list(K = 1000, r = .15, sigma = .1),
```

```

list(K = 900, r = .3, sigma = .01)
)
# set up population size vector
N <- seq(
  0 # does it make sense to estimate the change in pop_grwth_rt for N<2?
  , round(
    max(Logistic$PopulationSize)
    + sd(Logistic$PopulationSize)*2
    , digits = -2 # round to the nearest 100
  )
  , 10
)
# specify the data that will be used by your JAGS program
#the execution of JAGS is about 5 times faster on double precision than on integers.
hey_data = list(
  n = nrow(BayesNSF::Logistic), # n is required in the JAGS program to index the for structure
  x = as.double(BayesNSF::Logistic$PopulationSize),
  y = as.double(BayesNSF::Logistic$GrowthRate),
  N = as.double(N)
)
# specify 3 scalars, n.adapt, n.update, and n.iter
# n.adapt = number of iterations that JAGS will use to choose the sampler
# and to assure optimum mixing of the MCMC chain
n.adapt = 1000
# n.update = number of iterations that will be discarded to allow the chain to
# converge before iterations are stored (aka, burn-in)
n.update = 10000
# n.iter = number of iterations that will be stored in the
# final chain as samples from the posterior distribution
n.iter = 10000
#####
# Call to JAGS
#####
set.seed(1)
jm = rjags::jags.model(
  file = "LogisticJAGS.R"
  , data = hey_data
  , inits = inits
  , n.chains = length(inits)
  , n.adapt = n.adapt
)

```

```

## Compiling model graph
##   Resolving undeclared variables
##   Allocating nodes
## Graph information:
##   Observed stochastic nodes: 50
##   Unobserved stochastic nodes: 3
##   Total graph size: 896
##
## Initializing model

```

```

stats::update(jm, n.iter = n.update)
# save the coda object (more precisely, an mcmc.list object) to R as "zm"
zm = rjags::coda.samples(
  model = jm
  , variable.names = c("K", "r", "sigma", "tau", "N_max_pop_grwth_rt", "pop_grwth_rt")
  , n.iter = n.iter
  , n.thin = 1
)
#####
# check output
#####
# summary
MCMCvis::MCMCsummary(zm, params = c("K", "r", "sigma", "tau", "N_max_pop_grwth_rt"))

```

```

##                mean                sd                2.5%                50%
## K                1.236876e+03 6.243228e+01 1.128811e+03 1.231665e+03
## r                2.008528e-01 9.727105e-03 1.815042e-01 2.008846e-01
## sigma            2.863662e-02 3.007144e-03 2.350638e-02 2.838879e-02
## tau              1.258991e+03 2.579033e+02 8.061866e+02 1.240813e+03
## N_max_pop_grwth_rt 6.184380e+02 3.121614e+01 5.644056e+02 6.158323e+02
##                97.5% Rhat n.eff
## K                1.373936e+03      1 7340
## r                2.199669e-01      1 7383
## sigma            3.521942e-02      1 14537
## tau              1.809791e+03      1 16491
## N_max_pop_grwth_rt 6.869681e+02      1 7340

```

```

# chain 1 first 6 iterations and specific columns
zm[[1]][1:6, c("K", "r", "sigma", "tau", "N_max_pop_grwth_rt")]

```

```

##           K           r          sigma          tau N_max_pop_grwth_rt
## [1,] 1191.938 0.1996379 0.02794900 1280.169      595.9691
## [2,] 1212.930 0.2070942 0.02833138 1245.847      606.4652
## [3,] 1209.625 0.2010009 0.02749981 1322.332      604.8124
## [4,] 1190.920 0.2078684 0.02446475 1670.777      595.4599
## [5,] 1208.301 0.2019632 0.03054199 1072.026      604.1503
## [6,] 1218.088 0.1978723 0.03029639 1089.478      609.0441

```

```

# The rate of population growth

```

```

MCMCvis::MCMCpstr(zm, params = "pop_grwth_rt", func = function(x) quantile(x, c(0.025, 0.5, 0.975))) %>%
  as.data.frame() %>%
  dplyr::bind_cols(N = N) %>%
  dplyr::slice_head(n = 6)

```

```

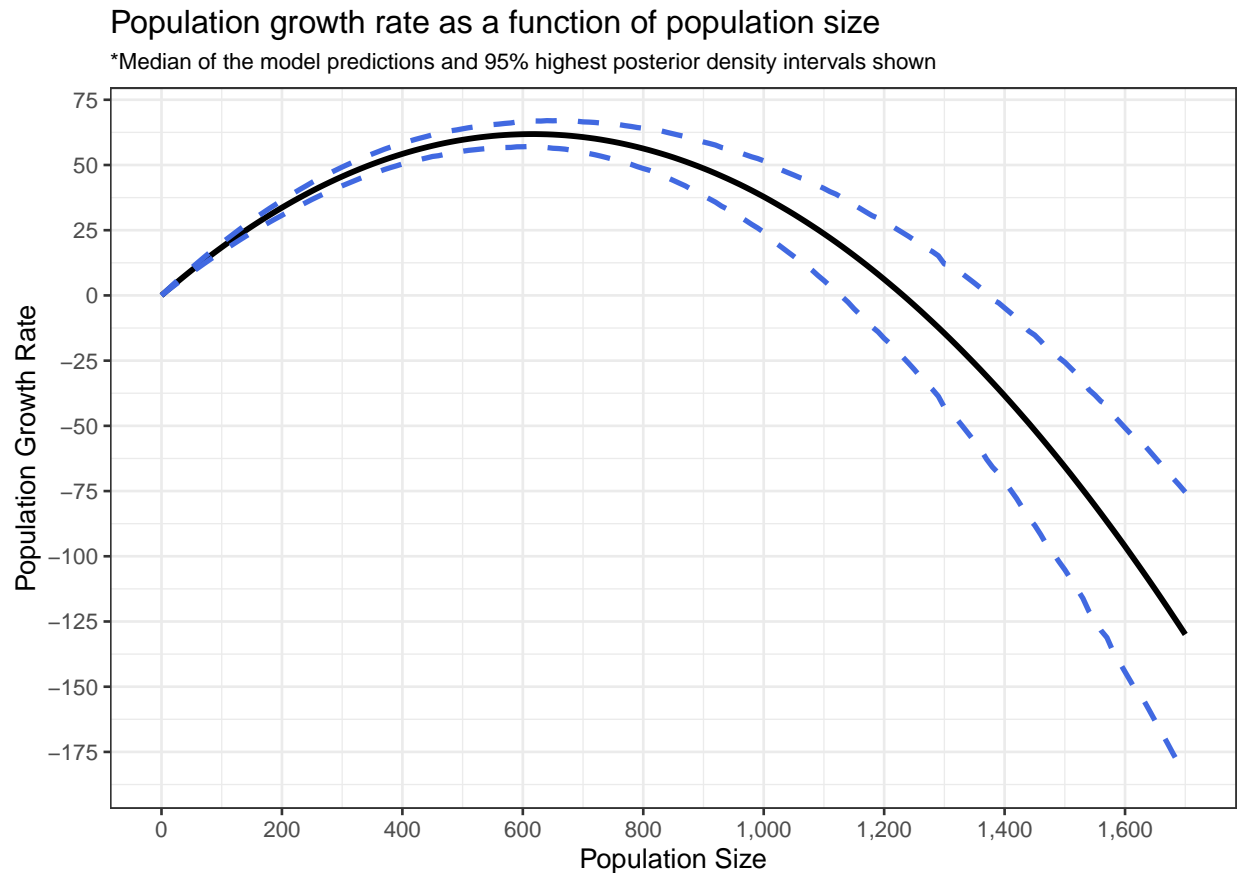
##           pop_grwth_rt.2.5. pop_grwth_rt.50. pop_grwth_rt.97.5.  N
## pop_grwth_rt[1]           0.000000           0.000000           0.000000  0
## pop_grwth_rt[2]           1.801219           1.992611           2.180681 10
## pop_grwth_rt[3]           3.575502           3.952697           4.324095 20
## pop_grwth_rt[4]           5.322428           5.880065           6.429662 30
## pop_grwth_rt[5]           7.041195           7.775308           8.498166 40
## pop_grwth_rt[6]           8.732861           9.637190          10.529233 50

```

## Question 2

Plot the median growth rate of the *population* (not the per-capita rate) rate and a 95% highest posterior density interval as a function of  $N$ . What does this curve tell you about the difficulty of sustaining harvest of populations?

```
dplyr::bind_cols(
  N = N
  , median_pop_grwth_rt = MCMCvis::MCMCpstr(zm, params = "pop_grwth_rt", func = median) %>% unlist()
  , MCMCvis::MCMCpstr(zm, params = "pop_grwth_rt", func = function(x) HDInterval::hdi(x, credMass = 0.95)) %>%
) %>%
# plot
ggplot(data = .) +
  geom_line(mapping = aes(x = N, y = median_pop_grwth_rt), color = "black", lwd = 1.1) +
  geom_line(mapping = aes(x = N, y = pop_grwth_rt.upper), color = "royalblue", lwd = 1, linetype = "dashed") +
  geom_line(mapping = aes(x = N, y = pop_grwth_rt.lower), color = "royalblue", lwd = 1, linetype = "dashed") +
  scale_y_continuous(breaks = scales::extended_breaks(n=10)) +
  scale_x_continuous(breaks = scales::extended_breaks(n=10), labels = scales::comma) +
  xlab("Population Size") +
  ylab("Population Growth Rate") +
  labs(
    title = "Population growth rate as a function of population size"
    , subtitle = "*Median of the model predictions and 95% highest posterior density intervals shown"
  ) +
  theme_bw() +
  theme(
    plot.subtitle = element_text(size = 9)
  )
```



### Question 3

What is the probability that the intrinsic rate of increase ( $r$ ) exceeds 0.22? What is the probability that  $r$  falls between 0.18 and 0.22?

```
# access data from MCMC list
temp_df <- MCMCvis::MCMCchains(zm, params = c("r")) %>% as.data.frame()
# probability that the intrinsic rate of increase  $r$  exceeds 0.22
temp_1 <- 1 - stats::ecdf(temp_df$r)(0.22)
# probability that  $r$  falls between 0.18 and 0.22
temp_2 <- stats::ecdf(temp_df$r)(0.22) - stats::ecdf(temp_df$r)(0.18)
```

The probability that the intrinsic rate of increase ( $r$ ) exceeds 0.22 is: 2.5%

The probability that  $r$  falls between 0.18 and 0.22 is: 95.7%

### Lizards on islands

This problem is courtesy of McCarthy (2007). Polis et al. (1998) analyzed the probability of occupancy of islands  $p$  by lizards as a function of the ratio of the islands' perimeter to area ratios. The data from this investigation are available in the data frame `BayesNSF::IslandsLizards`. The response data, as you will see, are 0 or 1: 0 if there were no lizards found on the island, 1 if there were 1 or more lizards observed. You are heroically assuming that if you fail to find a lizard, none are present on the island.

## Question 1

Construct a simple Bayesian model that represents the probability of occupancy as:

$$g(a, b, x_i) = \frac{e^{a+bx_i}}{1 + e^{a+bx_i}}$$

where  $x_i$  is the perimeter to area ratio of the  $i^{th}$  island. So, now that you have the deterministic model, the challenge is to choose the proper likelihood to link the data to the model. How do the data arise? What likelihood function is needed to represent the data?

The data – occupancy of islands  $p$  by lizards – arise from a Bernoulli distribution with the random variable  $p$  taking on the values 0 or 1. The likelihood function for the Bernoulli distribution is the inverse logit (i.e. the logistic function) with the form:

$$\text{inverse logit}(\phi_i) = \frac{\exp(\phi)}{1 + \exp(\phi)}$$

## Question 2

Write the expression for the posterior and joint distribution of the parameters and data, as we have learned how to do in lecture. Use the joint distribution as a basis for JAGS code needed to estimate the posterior distribution of  $a$  and  $b$ . Assume vague priors on the intercept and slope, e.g.,  $\beta_0 \sim \text{normal}(0, 10000)$ ,  $\beta_1 \sim \text{normal}(0, 10000)$ . Draw a DAG if you like. There doesn't appear to be any variance term in this model. How can that be?

$$[a, b | \mathbf{y}] \propto \prod_{i=1}^n \text{Bernoulli}(y_i | g(a, b, x_i)) \text{normal}(a | 0, 10000) \text{normal}(b | 0, 10000)$$
$$p = g(a, b, x_i) = \text{inverse logit}(a + bx_i) = \frac{\exp(a + bx_i)}{1 + \exp(a + bx_i)}$$

There doesn't appear to be any variance term in this model. How can that be?

The Bernoulli distribution is the discrete probability distribution of a random variable which takes the value 1 with probability  $p$  and the value 0 with probability  $q = 1 - p$  and variance  $\sigma^2 = pq = p(1 - p)$ . The model above includes  $p$  which determines the variance  $\sigma^2$ .

## Question 3

Using JAGS, run MCMC for three chains for the parameters  $a$  and  $b$  and the derived quantity  $p_i$ , the probability of occupancy. JAGS has a function, `ilogit` for the inverse logit that you might find helpful. Selecting initial conditions can be a bit tricky with the type of likelihood you will use here. You may get the message:

*Error in jags.model("IslandsJags.R", data = data, inits, n.chains = length(inits), : Error in node y[4]  
Observed node inconsistent with unobserved parents at initialization.*

To overcome this, try the following:

- Standardize the the perimeter to area ratio covariate using the `scale` function in R, which subtracts the mean of the data from every data point and divides by the standard deviation of the data. You want the default arguments for `center` and `scale` in this function.
- Choose initial values for  $a$  and  $b$  so that `inverselogit(a + b · standardized( $x_i$ ))` is between 0.01 and 0.99.

## Set up the data

```
data_df <- BayesNSF::IslandsLizards %>%
  # sort
  dplyr::arrange(desc(perimeterAreaRatio)) %>%
  # standardize
  dplyr::mutate(
    perim_area_ratio_z = scale(perimeterAreaRatio)
  )

inv_logit_fn <- function(a, b, x){
  exp(a + b*x) / (1 + exp(a + b*x))
}

a <- seq(0,50,.25)
b <- seq(0,50,.25)

# specify the initial conditions for the MCMC chain
inits = list(
  list(K = 1500, r = .2, sigma = 1),
  list(K = 1000, r = .15, sigma = .1),
  list(K = 900, r = .3, sigma = .01)
)

# specify the data that will be used by your JAGS program
#the execution of JAGS is about 5 times faster on double precision than on integers.
hey_data = list(
  n = nrow(SESYNCBayes::Logistic), # n is required in the JAGS program to index the for structure
  x = as.double(SESYNCBayes::Logistic$PopulationSize),
  y = as.double(SESYNCBayes::Logistic$GrowthRate)
)

# specify 3 scalars, n.adapt, n.update, and n.iter
# n.adapt = number of iterations that JAGS will use to choose the sampler
# and to assure optimum mixing of the MCMC chain
n.adapt = 1000
# n.update = number of iterations that will be discarded to allow the chain to
# converge before iterations are stored (aka, burn-in)
n.update = 10000
# n.iter = number of iterations that will be stored in the
# final chain as samples from the posterior distribution
n.iter = 10000
```

## JAGS Model

```
## Logistic example for Primer
model{
  # priors
  K ~ dunif(0, 4000) # dunif(alpha = lower limit, beta = upper limit)
  r ~ dunif(0, 2) # dunif(alpha, beta)
  sigma ~ dunif(0, 2) # dunif(alpha, beta)
  tau <- 1/sigma^2
  # likelihood
```



```
for(i in 1:n){  
  mu[i] <- r - r/K * x[i]  
  y[i] ~ dnorm(mu[i], tau) # dnorm(mu, tau)  
}  
}
```

\textcolor{violet}{We can be 95% confident that the true value of the random variable falls between the upper and lower limits.}