ESS 575: Model Selection Lab

Team England

16 November, 2022

Contents

Team England:

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

cblommel@mail.colostate.edu, carolynm@mail.colostate.edu, brcrosby@rams.colostate.edu, george.woolsey@colostate.edu

Motivation

There are a wide range of views about the value of model selection as a route to insight in science (e.g., Burnham and Anderson 2002, Gelman et al. 1995, Gelman et al. 2004, Gelman et al. 2013, Hobbs et al. 2012, Ver Hoef 2015, Gelman and Rubin 1995, Gelman and Shalizi 2013, Hooten and Hobbs 2015). Ecologists, particularly wildlife ecologists, have embraced the use of Akaike Information Criterion (AIC) to compare models, in part because it can be used with any likelihood-based model and has connections with the predictive ability of the model. AIC is not Bayesian (nor is BIC). Three Bayesian alternatives to AIC are DIC, WAIC, and Posterior Predictive Loss. They are only subtly different in their forumalation but usually lead to the same conclusions when comparing models. DIC and Posterior Predictive Loss are a good place to start learning about Bayesian model comparison because they do not require the assumption that data lack spatial or temporal structure, as with WAIC. See Hooten and Hobbs (2015) for details on the suite of other Bayesian model comparison approaches,

As always, it will be valuable to have you lecture notes close at hand to understand the math that stands behind the code.

A Note About: Matrix Specification of Linear Models

Specifying linear models in matrix notation is compact and convenient relative to writing out the full model as a scalar equation when there are several predictor variables. Consider the the typical, deterministic linear model:

$$\mu_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}$$

It can be written in matrix form as:

$$\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{\beta}$$

where $\boldsymbol{\beta}$ is a column vector, $(\beta_0, \beta_1, \beta_2)'$ with length = number of model coefficients, and $\mathbf{X}$ is a *design* matrix with the number of rows equal to the number of data points and the number of columns equal to the number of predictor variables + 1 (so, in this example, 3). Column one of $\mathbf{X}$ usually contains all 1s. Column two of $\mathbf{X}$ contains the covariate values of predictor variable 1, column three, predictor variable 2, and so on. The response variables are represented as $(\mathbf{y})$, a vector of dimension $n \times 1$. If you are unfamiliar with matrix multiplication, ask one of the lab instructors to explain how this works.

Matrix notation is handy because we can use a single JAGS file to specify several different models using the code below.

```
 z <- X %*% beta # the regression model in matrix form, returns a vector of length n
    for(i in 1:n)   {
    lambda[i] <- exp(z[i])
    y[i] ~ dpois(lambda[i])
}
```

Note that `%*%` is the symbol for matrix multiplication in JAGS and R.

The reason this is so handy is the R function, `model.matrix()`, for creating a design matrix, i.e., the $(\mathbf{X})$. Consider the following:

```
X = model.matrix(~as.numeric(scale(area)) + as.numeric(scale(temp)), data = bird.sm.df)
```

This creates a design matrix with 1s in column one and standardized data for area and temperature in columns two and three using the data, `bird.sm.df`.

R libraries needed for this lab

You need to load the following libraries. Set the seed to 10 to compare your answers to ours.

```r
# bread-and-butter
library(tidyverse)
library(lubridate)
library(viridis)
library(scales)
library(latex2exp)
# visualization
library(cowplot)
library(kableExtra)
# jags and bayesian
library(rjags)
library(MCMCvis)
library(HDInterval)
library(BayesNSF)
#set seed
set.seed(10)
```

Problem

We seek to model the response, bird species richness in 49 US states, using a set of predictor variables: area, temperature, and precipitation. Fit the following models and compare them using Deviance Information criterion (DIC) for each. Use a prior mean of 0 and variance of 100 for each regression coefficient (assuming independence}.

1) Model 1: Intercept and area as covariate.
2) Model 2: Intercept and area and temperature as covariates.

The matrix parameterization allows us to fit the models using the same code, but different design matrices input as covariate data.

Load data

The data for this problem is located in the `BayesNSF::RichnessBirds` data frame of the BayesNSF package. The species column give the number of different species of birds in a state.

```
BayesNSF::RichnessBirds %>%
  dplyr::glimpse()
```

```
## Rows: 49
## Columns: 7
## $ rank    <int> 17, 3, 22, 4, 7, 45, 32, 48, 5, 15, 36, 23, 43, 44, 11, 37, 18~
## $ state   <fct> Alabama, Arizona, Arkansas, California, Colorado, Connecticut,~
## $ st      <fct> AL, AZ, AR, CA, CO, CT, DE, DC, FL, GA, ID, IL, IN, IA, KS, KY~
## $ species <int> 326, 435, 312, 415, 371, 273, 295, 241, 386, 328, 284, 311, 27~
## $ area    <int> 135765, 295254, 137732, 423970, 269601, 14357, 6447, 177, 1703~
## $ temp    <dbl> 62.8, 60.3, 60.4, 59.4, 45.1, 49.0, 55.3, 48.0, 70.7, 63.5, 44~
## $ precip  <dbl> 58.3, 13.6, 50.6, 22.2, 15.9, 50.3, 45.7, 48.0, 54.5, 50.7, 18~
```

Some preliminaries

```
bird.df <- BayesNSF::RichnessBirds

####
####  Remove Outliers
####

idx.outlier=(1:51)[(bird.df$species==min(bird.df$species) | bird.df$area==max(bird.df$area))]
bird.sm.df=bird.df[-idx.outlier,]

####  Setup Data to Fit Model
####  Below will make the full design matrix from a data frame.  Automatically makes the first column =
X.1 = model.matrix(~as.numeric(scale(area)), data = bird.sm.df)
X.2 = model.matrix(~as.numeric(scale(area)) + as.numeric(scale(temp)), data = bird.sm.df)
y = bird.sm.df$species
M1.list <- list(y=y, X=as.matrix(X.1), n=length(y), p=dim(X.1)[2]) # p = # of columns in matrix (# beta
M2.list <- list(y=y, X=as.matrix(X.2), n=length(y), p=dim(X.2)[2]) # p = # of columns in matrix (# beta
set.seed(7)
```

Question 1

Write an expression for the posterior and joint distribution for the two models with pencil and paper. Use and exponential deterministic model. That said, what functional form might be more realistic if you want to consider the effect of area of a state on species richness?

Model 1

Deterministic model of y (number of bird species):

$$\begin{aligned} y_i &\sim \mathsf{Poisson}\big(g(\boldsymbol{\beta}, x_i)\big) \\ \lambda = g(\boldsymbol{\beta}, x_i) &= \exp\big(\beta_0 + \beta_1 x_{1i}\big) \\ x_1 &= \text{area} \end{aligned}$$

Posterior and Joint:

$$\begin{aligned} \big[\boldsymbol{\beta} \mid \boldsymbol{y}\big] \propto & \prod_{i=1}^{n=47} \mathsf{Poisson}\big(y_i \mid g(\boldsymbol{\beta}, \boldsymbol{x_i})\big) \\ & \times \ \mathsf{normal}\big(\boldsymbol{\beta} \mid 0, 10000\big) \end{aligned}$$

Model 2

Deterministic model of y (number of bird species):

$$\begin{aligned} y_i &\sim \mathsf{Poisson}\big(g(\boldsymbol{\beta}, \mathbf{X})\big) \\ \lambda = g(\boldsymbol{\beta}, \mathbf{X}) &= \exp(\boldsymbol{\beta} \cdot \mathbf{X}) = \exp\big(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}\big) \\ x_1 &= \text{area}; x_2 = \text{temperature} \end{aligned}$$

Posterior and Joint:

$$\begin{aligned} \big[\boldsymbol{\beta} \mid \boldsymbol{y}\big] \propto & \prod_{i=1}^{n=47} \mathsf{Poisson}\big(y_i \mid g(\boldsymbol{\beta}, \mathbf{X})\big) \\ & \times \ \mathsf{normal}\big(\boldsymbol{\beta} \mid 0, 10000\big) \end{aligned}$$

Question 2

Write the JAGS model statement and fit the model using both sets of covariates. Compute Bayesian p values for the mean and the standard deviation. Interpret the p values. What they tell you about the "do as I say, not as I do" nature of this exercise? What could you do to fix the problem?

JAGS Model

```
## JAGS Model
model{
  # priors
  for(i in 1:p){
    beta[i] ~ dnorm(0, (1/10000)) # dnorm(mu = mean, tau= precision)
  }
  # likelihood
  # %*% is the symbol for matrix multiplication in JAGS and R.
  b_x <- X %*% beta # the regression model in matrix form, returns a vector of length n
  for(i in 1:n){
    # Deterministic model of y
    lambda[i] <- exp(b_x[i])
    # likelihood
      # Stochastic model of y
      y[i] ~ dpois(lambda[i])
      y_sim[i]  ~ dpois(lambda[i])
  }
  # Derived quantities
    #posterior predictive checks
      # test statistics y
      mean_y <- mean(y)
      sd_y <- sd(y)

      # test statistics y_sim
      mean_y_sim <- mean(y_sim)
      sd_y_sim <- sd(y_sim)

      # p-values
      p_val_mean <- step(mean_y_sim - mean_y)
      p_val_sd <- step(sd_y_sim - sd_y)
}
```

Implement JAGS Model

```
## JAGS Model
jags_model_txt <- "model{
  # priors
  for(i in 1:p){
    beta[i] ~ dnorm(0, (1/10000)) # dnorm(mu = mean, tau= precision)
  }
  # likelihood
  # %*% is the symbol for matrix multiplication in JAGS and R.
  b_x <- X %*% beta # the regression model in matrix form, returns a vector of length n
  for(i in 1:n){
    # Deterministic model of y
    lambda[i] <- exp(b_x[i])
    # likelihood
      # Stochastic model of y
      y[i] ~ dpois(lambda[i])
      y_sim[i]  ~ dpois(lambda[i])
  }
  # Derived quantities
    #posterior predictive checks
      # test statistics y
      mean_y <- mean(y)
      sd_y <- sd(y)

      # test statistics y_sim
      mean_y_sim <- mean(y_sim)
      sd_y_sim <- sd(y_sim)

      # p-values
      p_val_mean <- step(mean_y_sim - mean_y)
      p_val_sd <- step(sd_y_sim - sd_y)
}"
#get DIC module for calculating deviance and DIC directly
load.module("dic")
set.seed(7)
```

Model 1

```
jags_model <- textConnection(jags_model_txt)
################################################################
# implement model
##################################################################
# specify the initial conditions for the MCMC chain
inits_1 = list(
  beta = c(
    mean(log(y))
    , rep(0, dim(X.1)[2]-1)
  )
)
# 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 = 3000
# n.iter = number of iterations that will be stored in the
  # final chain as samples from the posterior distribution
n.iter = 8000
######################
# Call to JAGS
######################
M1.model = rjags::jags.model(
  file = jags_model
  , data = M1.list
  , inits = inits_1
  , n.chains = length(inits_1)
  , n.adapt = n.adapt
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 47
##    Unobserved stochastic nodes: 49
##    Total graph size: 303
##
## Initializing model
```

```
stats::update(M1.model, n.iter = n.update, progress.bar = "none")
# save the coda object (more precisely, an mcmc.list object) to R as "zm"
M1.out = rjags::coda.samples(
  model = M1.model
  , variable.names = c(
      # parameters
      "beta"
      , "lambda"
      # test statistics
      , "mean_y"
      , "sd_y"
      , "mean_y_sim"
      , "sd_y_sim"
      # p-values
      , "p_val_mean"
      , "p_val_sd"
      # deviance from DIC module
      , "deviance"
    )
  , n.iter = n.iter
  , n.thin = 1
  , progress.bar = "none"
)
```

```
# summary
MCMCvis::MCMCsummary(M1.out, params = c("beta", "deviance"))
```

Estimates of the β and DIC

```
##                   mean          sd        2.5%          50%       97.5% Rhat
## beta[1]     5.75208446 0.008279916   5.7360191   5.75208216   5.7683604   NA
## beta[2]     0.07794577 0.008286288   0.0616604   0.07783368   0.0942639   NA
## deviance  542.86113493 2.039322536 540.8661718 542.25529405 548.4620001   NA
##          n.eff
## beta[1]   5080
## beta[2]   4813
## deviance  3361
```

```r
# summary
MCMCvis::MCMCsummary(M1.out, params = c(
      # test statistics
      "mean_y"
      , "mean_y_sim"
      , "sd_y"
      , "sd_y_sim"
    )
    , n.eff = FALSE
  )
```

Posterior predictive check - Test Statistics

```
##                  mean       sd      2.5%       50%     97.5% Rhat
## mean_y     315.82979 0.000000 315.82979 315.82979 315.82979   NA
## mean_y_sim 315.85180 3.686524 308.63777 315.82979 323.02128   NA
## sd_y        44.31131 0.000000  44.31131  44.31131  44.31131   NA
## sd_y_sim    30.67252 3.268560  24.41418  30.61135  37.23994   NA
```

```r
# summary
MCMCvis::MCMCsummary(M1.out, params = c(
      # p-values
      "p_val_mean"
      , "p_val_sd"
    )
  )
```

Posterior predictive check - p-values

```
##                mean        sd 2.5% 50% 97.5% Rhat n.eff
## p_val_mean 0.50225 0.5000262    0   1     1   NA  6923
## p_val_sd   0.00000 0.0000000    0   0     0   NA     0
```

Model 2

```r
jags_model <- textConnection(jags_model_txt)
################################################################
# implement model
##################################################################
# specify the initial conditions for the MCMC chain
inits_2 = list(
  beta = c(
    mean(log(y))
    , rep(0, dim(X.2)[2]-1)
  )
)
```

```
######################
# Call to JAGS
######################
M2.model = rjags::jags.model(
  file = jags_model
  , data = M2.list
  , inits = inits_2
  , n.chains = length(inits_1)
  , n.adapt = n.adapt
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 47
##    Unobserved stochastic nodes: 50
##    Total graph size: 351
##
## Initializing model
```

```
stats::update(M2.model, n.iter = n.update, progress.bar = "none")
# save the coda object (more precisely, an mcmc.list object) to R as "zm"
M2.out = rjags::coda.samples(
  model = M2.model
  , variable.names = c(
      # parameters
      "beta"
      , "lambda"
      # test statistics
      , "mean_y"
      , "sd_y"
      , "mean_y_sim"
      , "sd_y_sim"
      # p-values
      , "p_val_mean"
      , "p_val_sd"
      # deviance from DIC module
      , "deviance"
    )
  , n.iter = n.iter
  , n.thin = 1
  , progress.bar = "none"
)
```

```
# summary
MCMCvis::MCMCsummary(M2.out, params = c("beta", "deviance"))
```

Estimates of the β and DIC

```
##                  mean          sd         2.5%          50%         97.5% Rhat
## beta[1]    5.75038041 0.008143661   5.73466152   5.75039258   5.76637052   NA
## beta[2]    0.08120743 0.008227332   0.06498138   0.08119425   0.09712615   NA
## beta[3]    0.05848974 0.008234982   0.04230193   0.05847307   0.07429109   NA
## deviance 492.96001720 2.490729746 490.15918921 492.31372556 499.51679164   NA
##          n.eff
## beta[1]   4723
## beta[2]   4688
## beta[3]   4943
## deviance  3441
```

```
# summary
MCMCvis::MCMCsummary(M2.out, params = c(
      # test statistics
      "mean_y"
      , "mean_y_sim"
      , "sd_y"
      , "sd_y_sim"
    )
    , n.eff = FALSE
  )
```

Posterior predictive check - Test Statistics

```
##                 mean       sd      2.5%       50%     97.5% Rhat
## mean_y     315.82979 0.000000 315.82979 315.82979 315.82979   NA
## mean_y_sim 315.77966 3.630675 308.70213 315.68085 323.08511   NA
## sd_y        44.31131 0.000000  44.31131  44.31131  44.31131   NA
## sd_y_sim    36.08862 3.422858  29.51040  36.06088  42.88396   NA
```

```
# summary
MCMCvis::MCMCsummary(M2.out, params = c(
      # p-values
      "p_val_mean"
      , "p_val_sd"
    )
  )
```

Posterior predictive check - p-values

```
##                mean         sd 2.5% 50% 97.5% Rhat n.eff
## p_val_mean 0.484625 0.49979479    0   0     1   NA  7124
## p_val_sd   0.009625 0.09763991    0   0     0   NA  8000
```