ESS 575: JAGS Problems Lab

Team England

11 October, 2022

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

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
######################
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                  1.238814e+03 6.285769e+01 1.130739e+03 1.233826e+03
## r                  2.005766e-01 9.584878e-03 1.816990e-01 2.006146e-01
## sigma              2.868067e-02 3.060358e-03 2.350608e-02 2.838717e-02
## tau                1.256167e+03 2.596727e+02 7.969055e+02 1.240954e+03
## N_max_pop_grwth_rt 6.194069e+02 3.142884e+01 5.653694e+02 6.169129e+02
##                           97.5% Rhat n.eff
## K                  1.375482e+03    1  6974
## r                  2.191919e-01    1  7531
## sigma              3.542392e-02    1 13946
## tau                1.809838e+03    1 15987
## N_max_pop_grwth_rt 6.877409e+02    1  6974
```

```
# 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,] 1209.104 0.2006842 0.02444531 1673.436           604.5521
## [2,] 1252.009 0.1965960 0.02952278 1147.322           626.0045
## [3,] 1296.326 0.1896080 0.02978455 1127.244           648.1631
## [4,] 1327.639 0.1944488 0.02632647 1442.829           663.8193
## [5,] 1221.984 0.1916496 0.02992593 1116.618           610.9921
## [6,] 1206.285 0.1916397 0.02566341 1518.348           603.1424
```

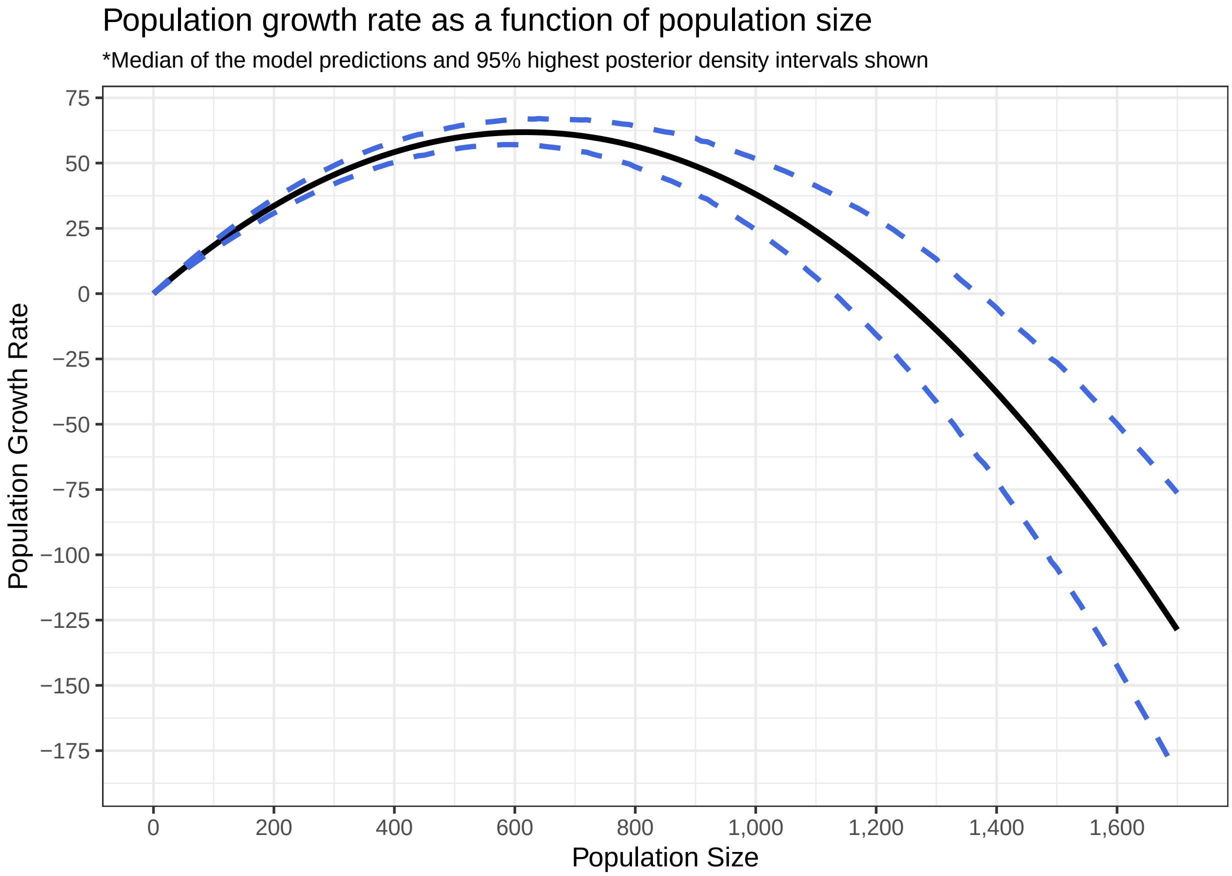
```
# The rate of population growth
MCMCvis::MCMCpstr(zm, params = "pop_grwth_rt", func = function(x) quantile(x, c(0.025, 0.5, 0.975))) %>%
  as.data.frame() %>%
  dplyr::bind_cols(N = N) %>%
  dplyr::slice_head(n = 6)
```

```
##                 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]          1.803364         1.989879           2.173085 10
## pop_grwth_rt[3]          3.579315         3.947190           4.308301 20
## pop_grwth_rt[4]          5.328023         5.872009           6.405803 30
## pop_grwth_rt[5]          7.049073         7.764078           8.467107 40
## pop_grwth_rt[6]          8.742418         9.623942          10.490991 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 = "das
  geom_line(mapping  = aes(x = N, y = pop_grwth_rt.lower), color = "royalblue", lwd = 1, linetype = "das
  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: 2.1%

The probability that $r$ falls between 0.18 and 0.22 is: 96.3%

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:

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

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., $\beta_0 \sim \mathsf{normal}(0, 10000)$, $\beta_1 \sim \mathsf{normal}(0, 10000)$. Draw a DAG if you like. There doesn't appear to be any variance term in this model. How can that be?

$$\left[a, b \mid \mathbf{y}\right] \propto \prod_{i=1}^{n} \mathsf{Bernoulli}\left(y_i \mid g(a, b, x_i)\right) \, \mathsf{normal}\left(a \mid 0, 10000\right) \, \mathsf{normal}\left(b \mid 0, 10000\right)$$

$$p = g(a, b, x_i) = \text{inverse logit}\left(a + bx_i\right) = \frac{\exp\left(a + bx_i\right)}{1 + \exp\left(a + bx_i\right)}$$

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 σ^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 $inverselogit(a + b \cdot standardized(x_i))$ 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 = as.numeric(scale(perimeterAreaRatio))
  )

# Choose initial values for a and b so that...
 # ...inverse logit(a+b*standardized(x_i)) is between 0.01 and 0.99
 inv_logit_fn <- function(a, b, x){
  exp(a + b*x) / (1 + exp(a + b*x))
 }

 a <- rnorm(10000, mean = 0, 20)
 b <- rnorm(10000, mean = 0, 20)
 y <- numeric(length(a))
 for(i in 1:length(a)){
  y[i] <- inv_logit_fn(a[i], b[i]
    , (dplyr::slice_sample(data_df, n=1))$perim_area_ratio_z
  )
 }
 temp_dta <- data.frame(
   a = a
   , b = b
   , y = y
  ) %>%
  dplyr::filter(
   y >= 0.01 & y <= 0.99
  )
 a_min <- floor(quantile(temp_dta$a, probs = 0.4)) %>% as.numeric()
 b_min <- floor(quantile(temp_dta$b, probs = 0.4)) %>% as.numeric()
 a_max <- ceiling(quantile(temp_dta$a, probs = 0.6)) %>% as.numeric()
 b_max <- ceiling(quantile(temp_dta$b, probs = 0.6)) %>% as.numeric()

 remove(temp_dta)
```

JAGS Model

```
## JAGS Model
model{
 # priors
 a ~ dnorm(0,1E-6)
 b ~ dnorm(0,1E-6)
 # likelihood
 for (i in 1:n) {
  p[i] <- ilogit(a + b*x[i])
  y[i] ~ dbern(p[i])
  }
}
```

Implement JAGS Model

```
###################################################################
# insert JAGS model code into an R script
###################################################################
{ # Extra bracket needed only for R markdown files - see answers
  sink("LizardsJAGS.R") # This is the file name for the jags code
  cat("
  model{
    # priors
    a ~ dnorm(0,1E-6)
    b ~ dnorm(0,1E-6)
    # likelihood
    for (i in 1:n) {
      p[i] <- ilogit(a + b*x[i])
      y[i] ~ dbern(p[i])
    }
  }
  ", fill = TRUE)
  sink()
}
#################################################################
# implement model
###################################################################
# specify the initial conditions for the MCMC chain
inits = list(
  list(a = runif(n = 1, min = a_min, max = a_max), b = runif(n = 1, min = b_min, max = b_max))
  , list(a = runif(n = 1, min = a_min, max = a_max), b = runif(n = 1, min = b_min, max = b_max))
  , list(a = runif(n = 1, min = a_min, max = a_max), b = runif(n = 1, min = b_min, max = b_max))
)
# 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(data_df) # n is required in the JAGS program to index the for structure
  , x = as.double(data_df$perim_area_ratio_z)
  , y = as.double(data_df$presence)
)

# 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
######################
```

```r
# Call to JAGS
######################
jm = rjags::jags.model(
  file = "LizardsJAGS.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: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 100
##
## Initializing model
```

```r
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("a", "b")
  # , variable.names = c("a", "b", "p")
  , n.iter = n.iter
  , n.thin = 1
)
#####################
# check output
#####################
# chain 1 first 6 iterations and specific columns
zm[[1]][1:6, c("a", "b")]
```

```
##                a         b
## [1,] -0.6613279 -4.834031
## [2,] -1.1926944 -4.592384
## [3,]  0.4163920 -3.659758
## [4,] -0.7683564 -4.880170
## [5,] -1.4041198 -5.566725
## [6,] -1.4724059 -5.111139
```

Question 4

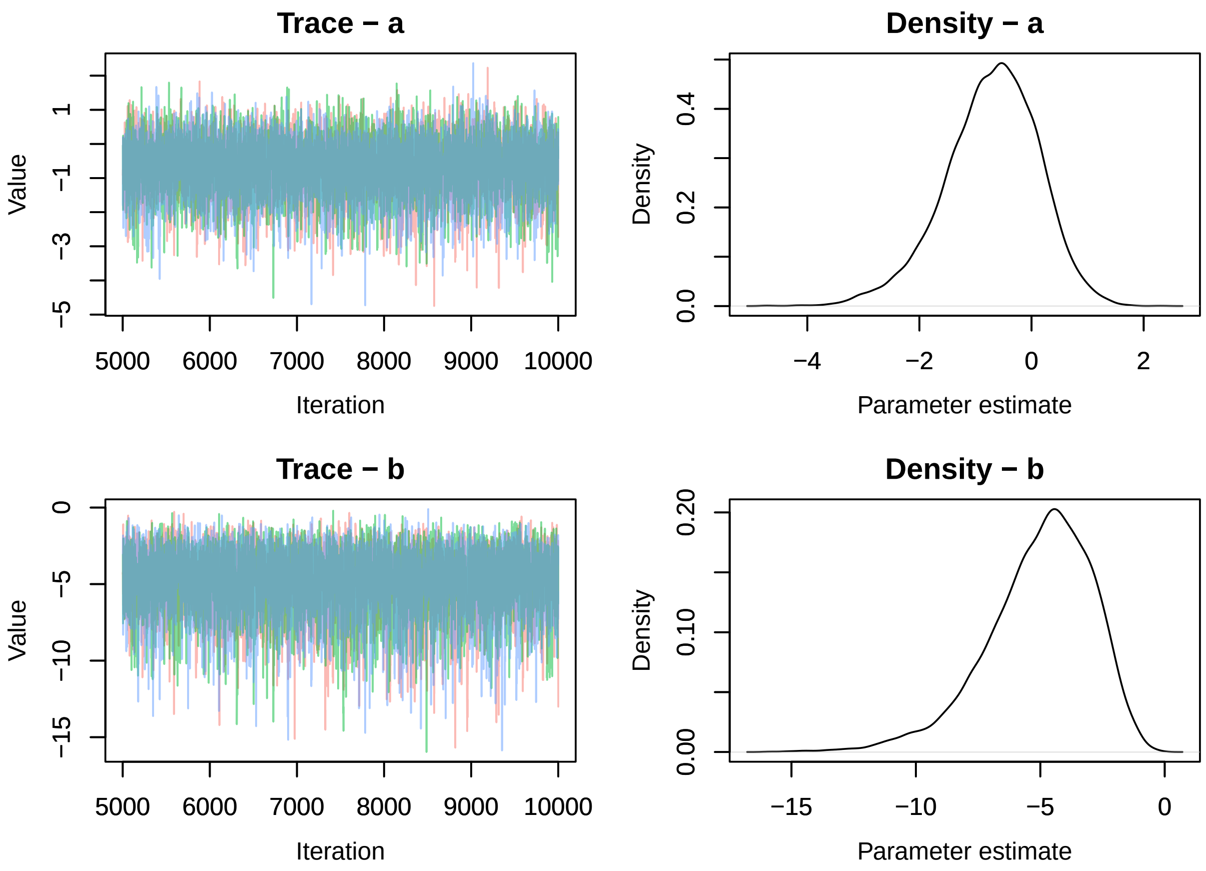
Do a summary table, a plot of the marginal posterior densities of the posterior density and a trace of the chain for parameters a and b. Does the trace indicate convergence? How can you tell? Use Gelman and Heidel diagnostics to check for convergence.

```r
# summary
MCMCvis::MCMCsummary(zm, params = c("a", "b")) %>%
    kableExtra::kable(
    caption = "Summary of simulations for parameters a and b"
    , digits = 5
  ) %>%
  kableExtra::kable_styling(font_size = 11) %>%
  kableExtra::column_spec(1, bold = TRUE) %>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

Table 1: Summary of simulations for parameters a and b

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
a	-0.69940	0.84324	-2.52851	-0.64593	0.80263	1	11604
b	-5.01017	2.15720	-10.15780	-4.71524	-1.68152	1	9094

```
# trace plot
MCMCvis::MCMCtrace(zm, params = c("a", "b"), pdf = FALSE)
```



```
# Gelman-Rubin diagnostic
coda::gelman.diag(zm)
```

```
## Potential scale reduction factors:
##
##   Point est. Upper C.I.
## a          1          1
## b          1          1
##
## Multivariate psrf
##
## 1
```

```
# Heidelberger and Welch diagnostic
coda::heidel.diag(zm)
```

```
## [[1]]
##
##   Stationarity start     p-value
##   test         iteration
## a passed       1         0.884
## b passed       1         0.718
##
##   Halfwidth Mean   Halfwidth
##   test
## a passed    -0.702 0.0271
## b passed    -4.992 0.0812
##
## [[2]]
##
##   Stationarity start     p-value
##   test         iteration
## a passed       1         0.817
## b passed       1         0.581
##
##   Halfwidth Mean   Halfwidth
##   test
## a passed    -0.675 0.0260
## b passed    -4.951 0.0731
##
## [[3]]
##
##   Stationarity start     p-value
##   test         iteration
## a passed       4001      0.0606
## b passed          1      0.2270
##
##   Halfwidth Mean   Halfwidth
##   test
## a passed    -0.751 0.0330
## b passed    -5.088 0.0764
```

Based on the trace plot and the Gelman-Ruban convergence diagnostic the chains have converged. This can be seen visually in the trace plot and the Gelman-Ruban convergance diagnostic value of '1' the indicates convergence. Values substantially above 1 would indicate lack of convergence.

Question 5

Plot the data as points. Overlay a line plot of the median and 95% highest posterior density intervals of the predicted probability of occurrence as a function of island perimeter to area ratios ranging from 1-60. Hint– create a vector of 1-60 in R, and use it as x values for an equation making predictions in your JAGS code. The curve is jumpy if you simply plot the predictions at the island perimeter to area data points. Remember, however, that the x's have been standardized to fit the coefficients, so you need to make predictions using standardized values in the sequence you create. You may plot these predictions against the un-standardized perimeter to area ratios, a plot that is more easily interpreted than plotting against the standardized ratios.

Data prep

```
# create a vector of 1-60...
  # the x's have been standardized to fit the coefficients...
  # so you need to make predictions using standardized values in the sequence you create
perim_area <- seq(1, 60, 0.25)
perim_area_z <- (perim_area - mean(data_df$perimeterAreaRatio)) / sd(data_df$perimeterAreaRatio)
```

JAGS Model

```
## JAGS Model
model{
  # priors
  a ~ dnorm(0,1E-6)
  b ~ dnorm(0,1E-6)
  # likelihood
  for (i in 1:n) {
    p[i] <- ilogit(a + b*x[i])
    y[i] ~ dbern(p[i])
  }
  ## quantities of interest
    # The predicted probability of occupancy
    for(j in 1:length(perim_area_z)){
      p_est[j] <- ilogit(a + b*perim_area_z[j])
    }

}
```

Implement JAGS Model

```
##################################################################
# insert JAGS model code into an R script
##################################################################
{ # Extra bracket needed only for R markdown files - see answers
  sink("LizardsJAGS.R") # This is the file name for the jags code
  cat("
  model{
    # priors
    a ~ dnorm(0,1E-6)
    b ~ dnorm(0,1E-6)
    # likelihood
    for (i in 1:n) {
      p[i] <- ilogit(a + b*x[i])
      y[i] ~ dbern(p[i])
    }
    ## quantities of interest
      # The predicted probability of occupancy
      for(j in 1:length(perim_area_z)){
        p_est[j] <- ilogit(a + b*perim_area_z[j])
      }

  }
  ", fill = TRUE)
  sink()
}
################################################################
# implement model
##################################################################
# specify the initial conditions for the MCMC chain
inits = list(
  list(a = runif(n = 1, min = a_min, max = a_max), b = runif(n = 1, min = b_min, max = b_max))
  , list(a = runif(n = 1, min = a_min, max = a_max), b = runif(n = 1, min = b_min, max = b_max))
  , list(a = runif(n = 1, min = a_min, max = a_max), b = runif(n = 1, min = b_min, max = b_max))
)
# 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(data_df) # n is required in the JAGS program to index the for structure
  , x = as.double(data_df$perim_area_ratio_z)
  , y = as.double(data_df$presence)
  , perim_area_z = as.double(perim_area_z)
)
# 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
######################
jm = rjags::jags.model(
  file = "LizardsJAGS.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: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 1048
##
## 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("a", "b", "p_est")
  , n.iter = n.iter
  , n.thin = 1
)
#####################
# check output
#####################
# summary
MCMCvis::MCMCsummary(zm, params = c("a", "b"))
```

```
##         mean        sd      2.5%        50%     97.5% Rhat n.eff
## a -0.6899837 0.8423686 -2.503918 -0.6407321  0.817215    1 11816
## b -4.9754189 2.1441057 -9.894814 -4.7001019 -1.649036    1  9245
```

```
# chain 1 first 6 iterations and specific columns
zm[[1]][1:6, c("a", "b")]
```

```
##               a         b
## [1,] -0.7093069 -3.149364
## [2,] -1.7848851 -3.243495
## [3,] -1.6477590 -6.158157
## [4,] -1.0914382 -2.671458
## [5,]  0.3667999 -6.829108
## [6,] -1.0538658 -3.613643
```

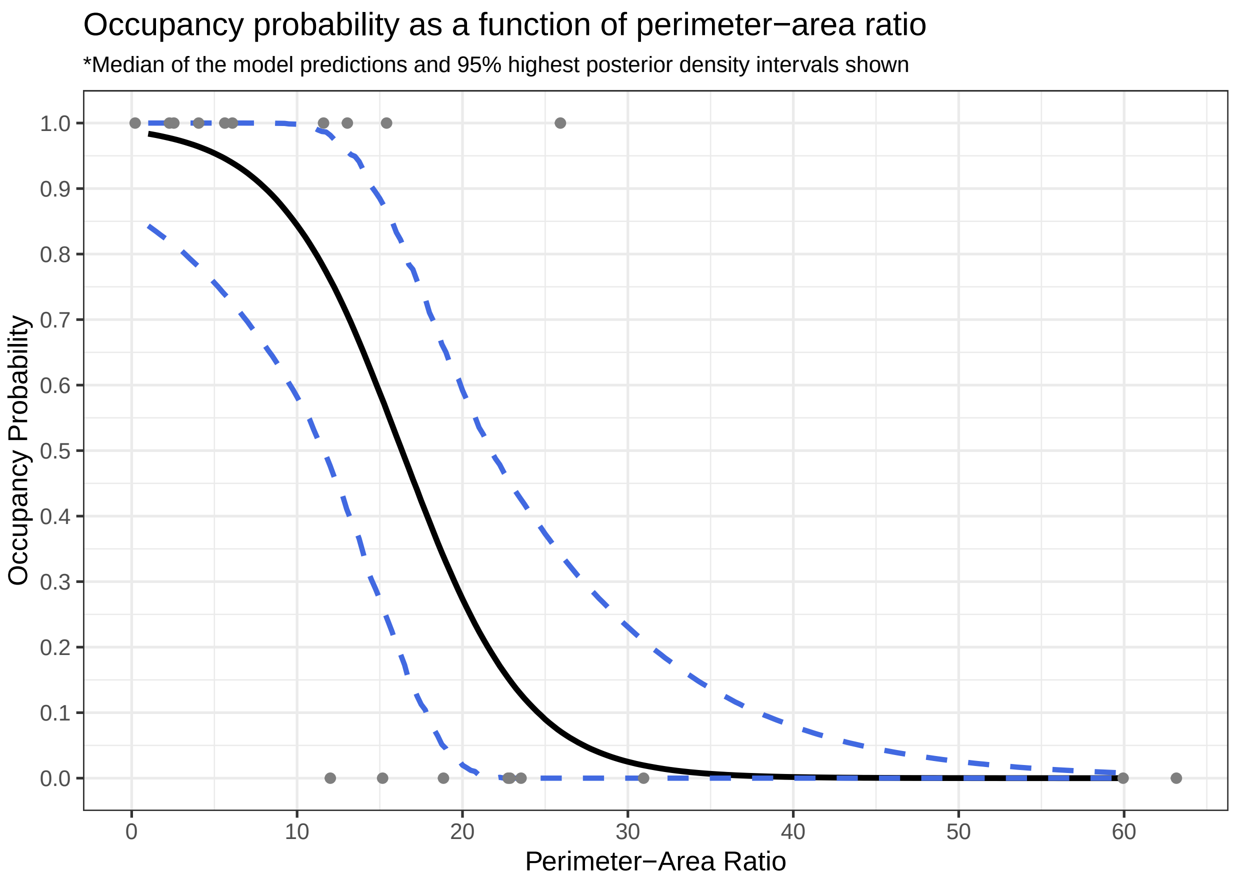
```
# The rate of occupancy
MCMCvis::MCMCpstr(zm, params = "p_est", func = function(x) quantile(x, c(0.025, 0.5, 0.975))) %>%
  as.data.frame() %>%
  dplyr::bind_cols(perim_area_z = perim_area_z) %>%
  dplyr::slice_head(n = 6)
```

```
##          p_est.2.5. p_est.50. p_est.97.5. perim_area_z
## p_est[1]  0.7887276 0.9837304   0.9998511   -1.0143901
## p_est[2]  0.7846504 0.9826104   0.9998287   -1.0000935
## p_est[3]  0.7799575 0.9814419   0.9998028   -0.9857970
## p_est[4]  0.7747998 0.9801480   0.9997746   -0.9715005
## p_est[5]  0.7695983 0.9787674   0.9997420   -0.9572040
## p_est[6]  0.7647383 0.9773822   0.9997025   -0.9429075
```

Plot

Plot the data as points. Overlay a line plot of the median and 95% highest posterior density intervals of the predicted probability of occurrence as a function of island perimeter to area ratios ranging from 1-60.

```
dplyr::bind_cols(
  perim_area = perim_area
  , median_p_est = MCMCvis::MCMCpstr(zm, params = "p_est", func = median) %>% unlist()
  , MCMCvis::MCMCpstr(zm, params = "p_est", func = function(x) HDInterval::hdi(x, credMass = 0.95)) %>%
) %>%
# plot
ggplot(data = .) +
  geom_line(mapping  = aes(x = perim_area, y = median_p_est), color = "black", lwd = 1.1) +
  geom_line(mapping  = aes(x = perim_area, y = p_est.upper), color = "royalblue", lwd = 1, linetype = "
  geom_line(mapping  = aes(x = perim_area, y = p_est.lower), color = "royalblue", lwd = 1, linetype = "
  # add sample data
  geom_point(
    data = data_df
    , mapping = aes(x = perimeterAreaRatio, y = presence)
    , color = "gray50"
  ) +
  scale_y_continuous(breaks = scales::extended_breaks(n=10)) +
  scale_x_continuous(breaks = scales::extended_breaks(n=10), labels = scales::comma) +
  xlab("Perimeter-Area Ratio") +
  ylab("Occupancy Probability") +
  labs(
    title = "Occupancy probability as a function of perimeter-area ratio"
    , subtitle = "*Median of the model predictions and 95% highest posterior density intervals shown"
  ) +
  theme_bw() +
  theme(
    plot.subtitle = element_text(size = 9)
  )
```



Question 6

Assume you are interested in 2 islands, one that has a perimeter to area ratio of 10, the other that has a perimeter to area ratio of 20. What is the 95% highest posterior density interval on the difference in the probability of occupancy of the two islands based on the analysis you did above? What is the probability that the difference exceeds 0? Remember that the data are standardized when you do this computation.

JAGS Model

```
## JAGS Model
model{
  # priors
  a ~ dnorm(0,1E-6)
  b ~ dnorm(0,1E-6)
  # likelihood
  for (i in 1:n) {
    p[i] <- ilogit(a + b*x[i])
    y[i] ~ dbern(p[i])
  }
  ## quantities of interest
    # The predicted probability of occupancy
    for(j in 1:length(perim_area_z)){
      p_est[j] <- ilogit(a + b*perim_area_z[j])
    }
    # different perimeter-area estimates
    p_x10 <- ilogit(a + b*x10)
    p_x20 <- ilogit(a + b*x20)
    diff_x10_x20 <- p_x10 - p_x20
}
```

Implement JAGS Model

```
##################################################################
# insert JAGS model code into an R script
##################################################################
{ # Extra bracket needed only for R markdown files - see answers
  sink("LizardsJAGS.R") # This is the file name for the jags code
  cat("
  model{
    # priors
    a ~ dnorm(0,1E-6)
    b ~ dnorm(0,1E-6)
    # likelihood
    for (i in 1:n) {
      p[i] <- ilogit(a + b*x[i])
      y[i] ~ dbern(p[i])
    }
    ## quantities of interest
      # The predicted probability of occupancy
      for(j in 1:length(perim_area_z)){
        p_est[j] <- ilogit(a + b*perim_area_z[j])
      }
      # different perimeter-area estimates
      p_x10 <- ilogit(a + b*x10)
      p_x20 <- ilogit(a + b*x20)
      diff_x10_x20 <- p_x10 - p_x20
  }
  ", fill = TRUE)
  sink()
}
################################################################
# implement model
##################################################################
# specify the initial conditions for the MCMC chain
inits = list(
  list(a = runif(n = 1, min = a_min, max = a_max), b = runif(n = 1, min = b_min, max = b_max))
  , list(a = runif(n = 1, min = a_min, max = a_max), b = runif(n = 1, min = b_min, max = b_max))
  , list(a = runif(n = 1, min = a_min, max = a_max), b = runif(n = 1, min = b_min, max = b_max))
)
# 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(data_df) # n is required in the JAGS program to index the for structure
  , x = as.double(data_df$perim_area_ratio_z)
  , y = as.double(data_df$presence)
  , perim_area_z = as.double(perim_area_z)
  , x10 = as.double((10 - mean(data_df$perimeterAreaRatio))/sd(data_df$perimeterAreaRatio))
  , x20 = as.double((20 - mean(data_df$perimeterAreaRatio))/sd(data_df$perimeterAreaRatio))
)
# 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
######################
jm = rjags::jags.model(
  file = "LizardsJAGS.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: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 1051
##
## 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("a", "b", "p_est", "p_x10", "p_x20", "diff_x10_x20")
  , n.iter = n.iter
  , n.thin = 1
)
#####################
# check output
#####################
# summary
MCMCvis::MCMCsummary(zm, params = c("p_x10", "p_x20", "diff_x10_x20"))
```

```
##                    mean        sd       2.5%       50%     97.5% Rhat n.eff
## p_x10         0.8198062 0.1226515 0.52943676 0.8444834 0.9837150    1 24887
## p_x20         0.2925769 0.1588310 0.04400136 0.2741659 0.6351609    1 11805
## diff_x10_x20 0.5272292 0.1735376 0.19663353 0.5262891 0.8618415    1 11425
```

```
# chain 1 first 6 iterations and specific columns
zm[[1]][1:6, c("a", "b", "p_x10", "p_x20", "diff_x10_x20")]
```

```
##                a         b     p_x10     p_x20 diff_x10_x20
## [1,]  0.69388281 -1.403339 0.8014137 0.6439705    0.1574432
## [2,] -0.86663191 -3.299986 0.6862020 0.2488587    0.4373433
## [3,] -0.54131502 -2.192677 0.6351559 0.3319220    0.3032339
## [4,] -1.09365578 -1.881900 0.4617645 0.2262821    0.2354824
## [5,] -0.88571518 -1.957226 0.5230690 0.2636819    0.2593871
## [6,] -0.05104999 -6.075087 0.9518812 0.3800502    0.5718310
```

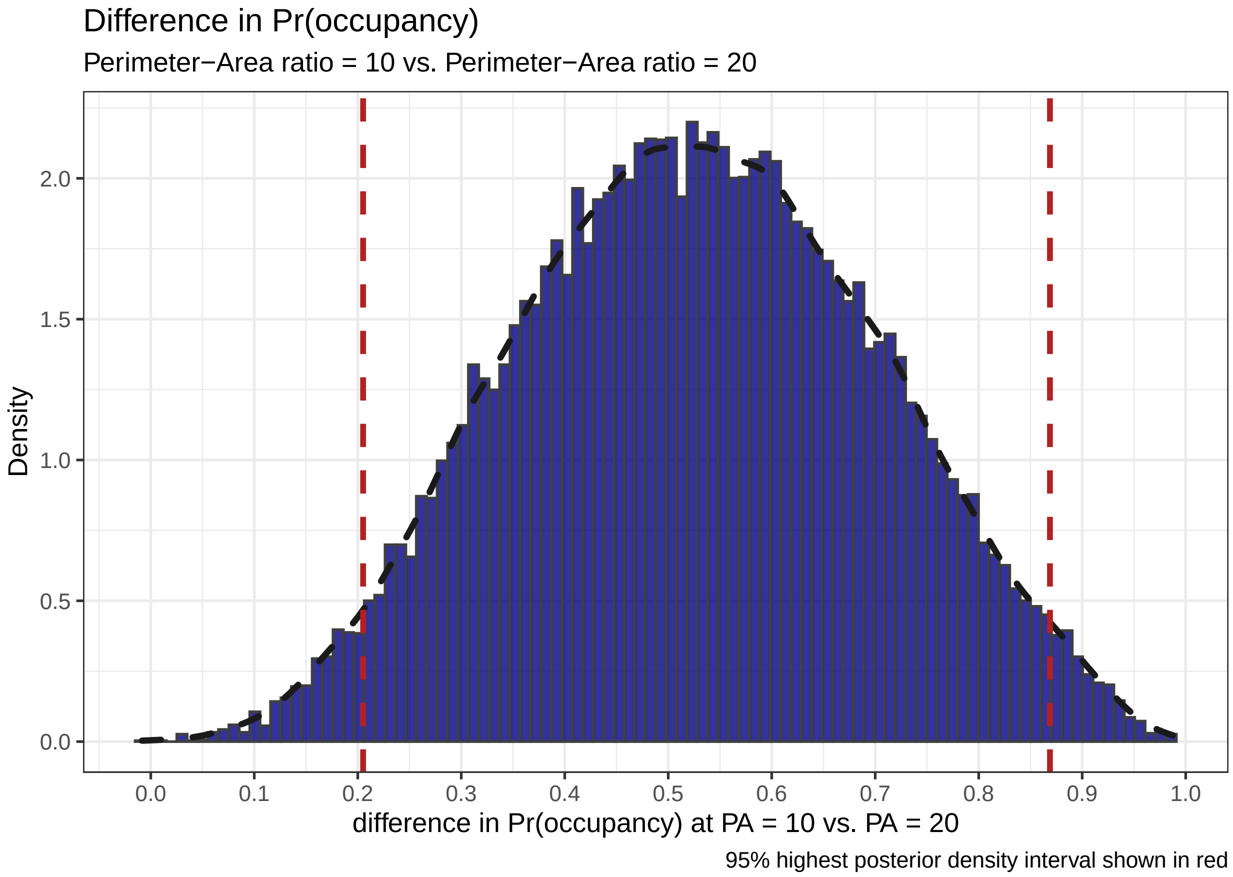
```
# The rate of occupancy
MCMCvis::MCMCpstr(zm, params = "p_est", func = function(x) quantile(x, c(0.025, 0.5, 0.975))) %>%
  as.data.frame() %>%
  dplyr::bind_cols(perim_area_z = perim_area_z) %>%
  dplyr::slice_head(n = 6)
```

```
##          p_est.2.5. p_est.50. p_est.97.5. perim_area_z
## p_est[1]  0.7900909 0.9836394   0.9998578   -1.0143901
## p_est[2]  0.7862607 0.9825541   0.9998361   -1.0000935
## p_est[3]  0.7810934 0.9813997   0.9998121   -0.9857970
## p_est[4]  0.7765930 0.9801148   0.9997851   -0.9715005
## p_est[5]  0.7719035 0.9788057   0.9997525   -0.9572040
## p_est[6]  0.7666836 0.9773677   0.9997163   -0.9429075
```

Plot

```
# extract data
temp_dta <- MCMCvis::MCMCchains(zm, params = c("a", "b", "p_x10", "p_x20", "diff_x10_x20")) %>%
  as.data.frame()
temp_hdi <- HDInterval::hdi(temp_dta$diff_x10_x20, credMass = 0.95)
temp_p_gt0 <- 1 - ecdf(temp_dta$diff_x10_x20)(0)

# the marginal posterior density of the difference
  # plot
  ggplot(data = temp_dta, mapping = aes(x = diff_x10_x20)) +
  geom_histogram(
    aes(y = ..density..)
    , bins = 100
    , fill = "navy"
    , alpha = 0.8
    , color = "gray25"
  ) +
  geom_density(
    aes(y = ..density..)
    , linetype = 2
    , lwd = 1.2
    , color = "gray10"
  ) +
  geom_vline(
    xintercept = temp_hdi
    , color = "firebrick"
    , linetype = "dashed"
    , lwd = 1.1
  ) +
  scale_x_continuous(breaks = scales::extended_breaks(n=9)) +
  xlab("difference in Pr(occupancy) at PA = 10 vs. PA = 20") +
  ylab("Density") +
  labs(
    title = "Difference in Pr(occupancy)"
    , subtitle = "Perimeter-Area ratio = 10 vs. Perimeter-Area ratio = 20"
    , caption = "95% highest posterior density interval shown in red"
  ) +
  theme_bw()
```



Short Answer

What is the 95% highest posterior density interval on the difference in the probability of occupancy of the two islands based on the analysis you did above? What is the probability that the difference exceeds 0?

The 95% highest posterior density interval on the difference in the probability of occupancy of the two islands (PA = 10 vs. PA = 20) is between **0.205** and **0.869**

The probability that the difference in the probability of occupancy between the two islands (PA = 10 vs. PA = 20) exceeds 0 is: **1.000**

Question 7

What fundamentally important source of error are we sweeping under the rug in all of these fancy calculations? What are the consequences of failing to consider this error for our estimates? Do you have some ideas about how we might cope with this problem?

A potential source of error in this analysis is in the collection of data. We are "assuming that if you fail to find a lizard, none are present on the island." It is possible that there is correlation between the detection of a lizard and the independent variable in our model perimeter to area ratio. For example, it might be more difficult to detect lizards are larger islands.

Vague priors in non-linear models

The priors you chose above were vague for the intercept and slope in the logistic regression but they were *not* vague for p_i. This is generally true for the output of nonlinear functions like the inverse logit (Lunn et al., 2012; Seaman et al., 2012), so you need to be careful about inference on the output of these non-linear function. See Hobbs and Hooten (2015) section 5.4.1 for an explanation of priors in logistic regression. It is prudent to to explore the effect of different values for priors on the shape of a the "prior" for quantities that are non-linear functions of model parameters, as demonstrated in the following exercise

Question 1

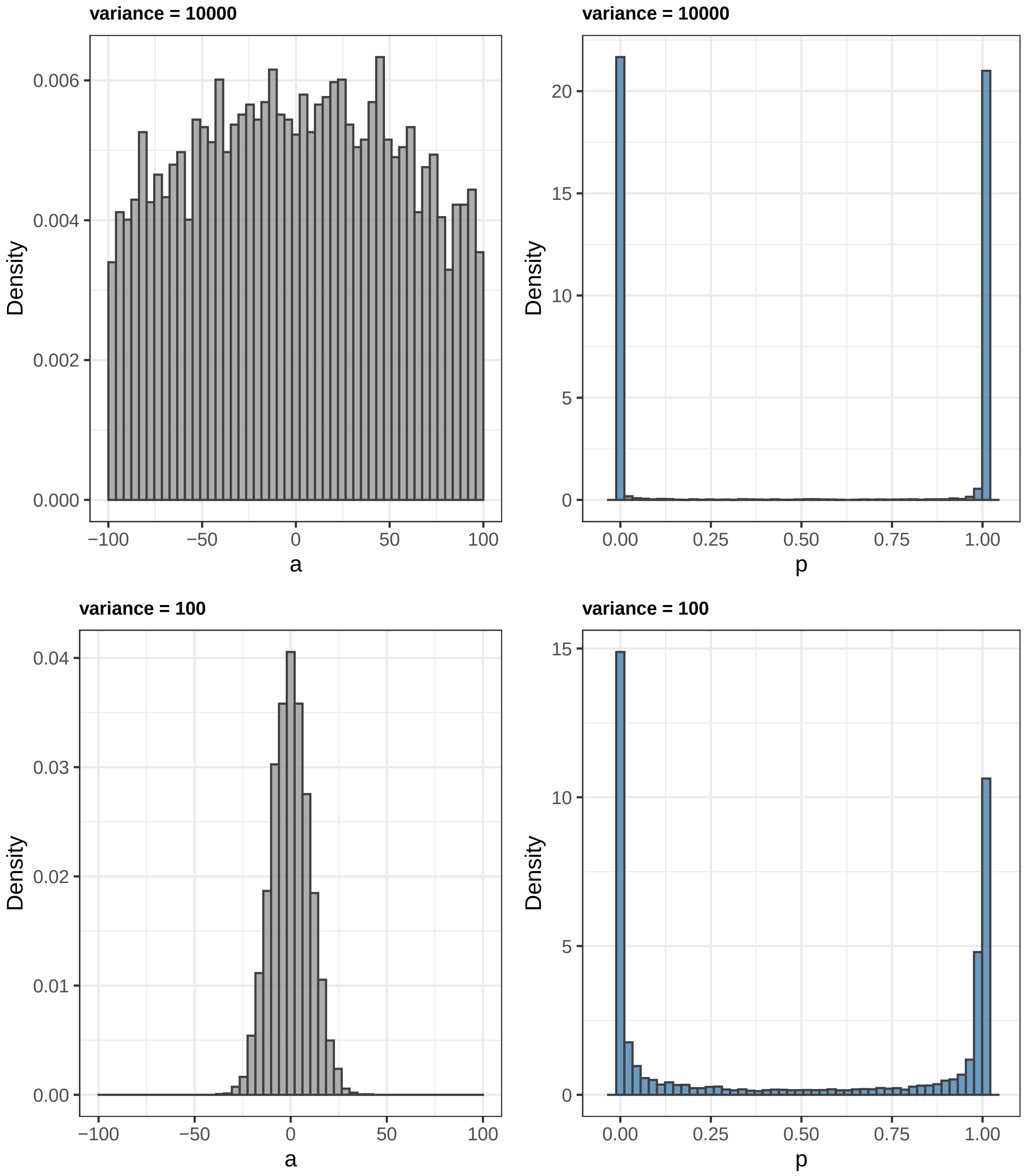
Write a function that takes an argument for the variance σ^2. The function should:

1) simulate 10000 draws from a normal distribution with mean 0 and variance σ^2 representing a prior on a, remembering, of course, that the argument to `rnorm` is the standard deviation (σ).
2) Plot histograms of the draws for a.
3) Plot a histogram of the inverse logit of the random draws, representing a "prior" on p at the mean of x (i.e., where the scaled value of x = 0).

Plotting these in side by side panels will facilitate comparison. Use your function to explore the effect of different variances ranging from 1 to 10000 on the priors for a and p. Find a value for the variance that produces a flat "prior" on p. The `boot` library contains an inverse logit function or you can write your own.

```
# inverse logit function
inv_logit_fn <- function(x){
  exp(x) / (1 + exp(x))
}
# plot function
my_plot_fn <- function(sigma_sq){
  a <- rnorm(n = 10000, mean = 0, sd = sqrt(sigma_sq))
  # histogram of a
    p1 <- ggplot(data = data.frame(a = a), mapping = aes(x = a)) +
      geom_histogram(
        aes(y = ..density..)
        , bins = 50
        , fill = "gray60"
        , alpha = 0.8
```

```r
        , color = "gray25"
      ) +
      scale_x_continuous(limits = c(-100, 100)) +
      xlab("a") +
      ylab("Density") +
      labs(
        title = paste0("variance = ", sigma_sq)
      ) +
      theme_bw() +
      theme(
        plot.title = element_text(size = 9, face = "bold")
      )
    # histogram of a
    p2 <- ggplot(data = data.frame(a = inv_logit_fn(a)), mapping = aes(x = a)) +
      geom_histogram(
        aes(y = ..density..)
        , bins = 50
        , fill = "steelblue"
        , alpha = 0.8
        , color = "gray25"
      ) +
      scale_x_continuous(limits = c(-0.05, 1.05)) +
      xlab("p") +
      ylab("Density") +
      labs(
        title = paste0("variance = ", sigma_sq)
      ) +
      theme_bw() +
      theme(
        plot.title = element_text(size = 9, face = "bold")
      )
    # combine
    cowplot::plot_grid(p1, p2)
}
# explore the effect of different variances ranging from 1 to 10000
# !!!!!!!!!!!!!! uncomment to see wider range
# c(
#   rev(seq(1000, 10000, 3000))
#   , rev(seq(100, 1000, 300)[1:3])
#   , rev(seq(10, 100, 30)[1:3])
#   , rev(seq(1, 10, 3)[1:3])
# ) %>%
# purrr::map(my_plot_fn)
c(
  10000
  , 100
  , 10
  , 5
  , 1
) %>%
purrr::map(my_plot_fn)
```

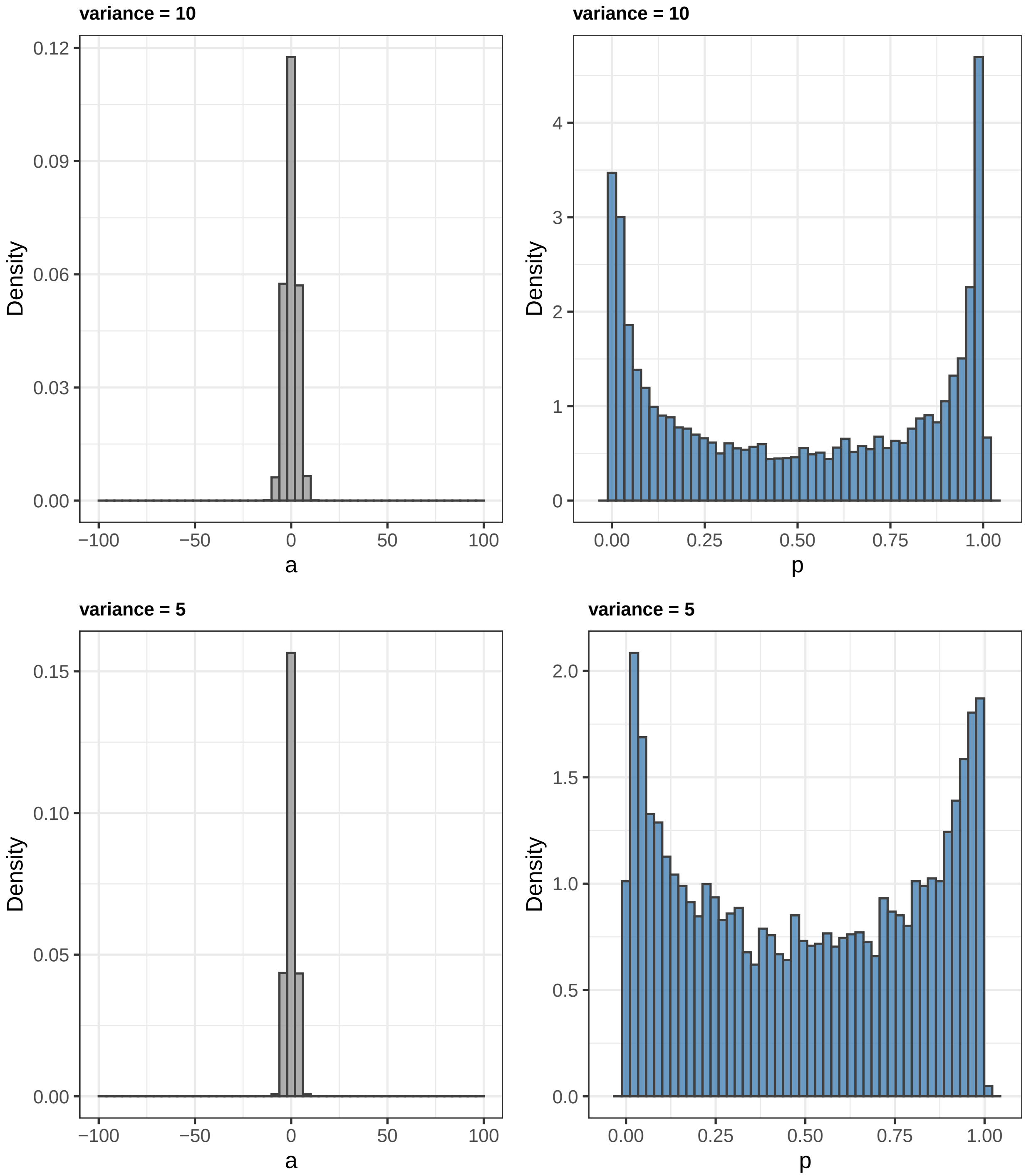


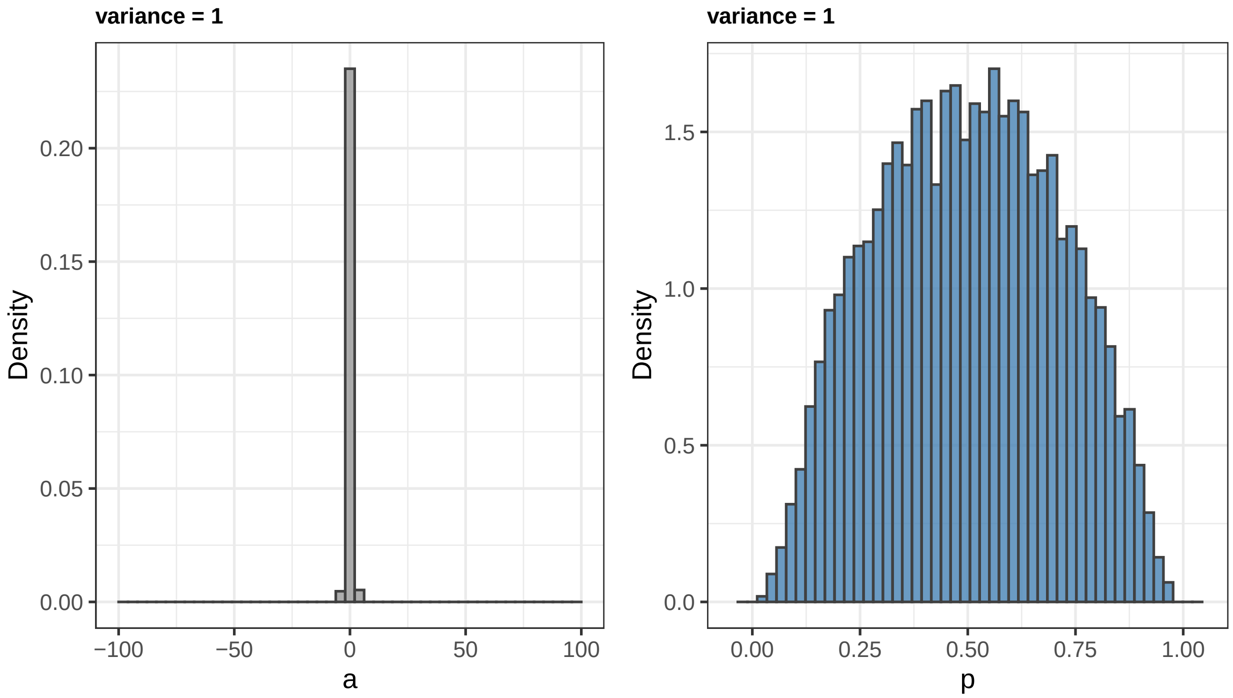
variance = 10000

variance = 10000

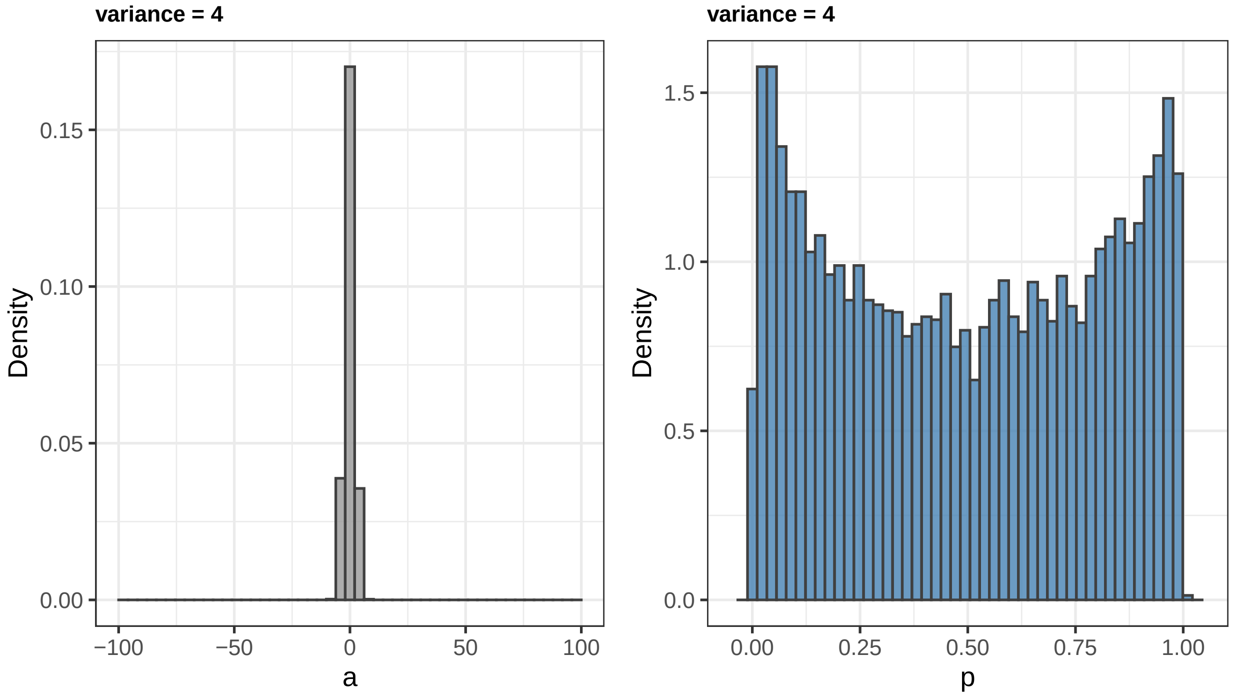
variance = 100

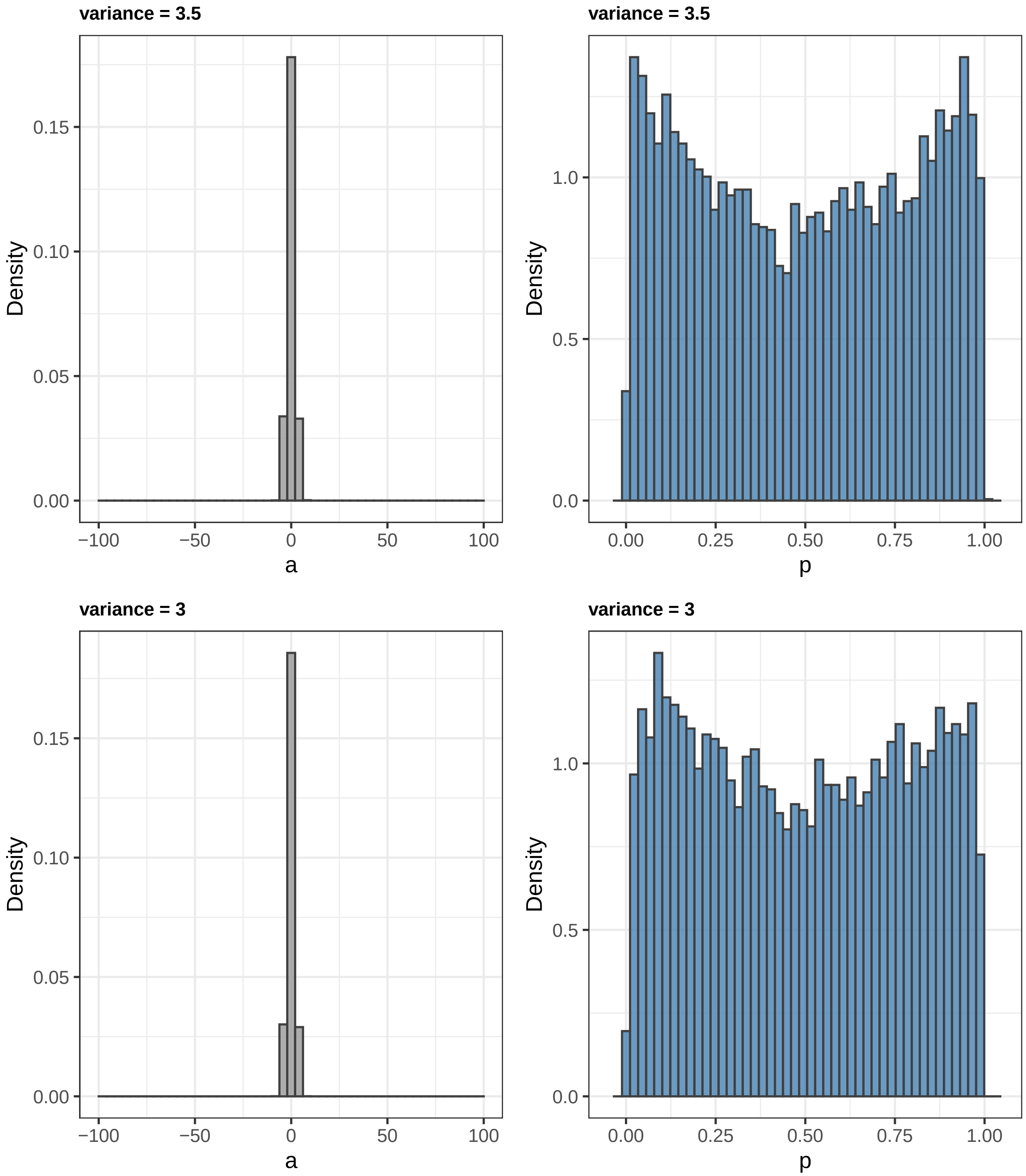
variance = 100

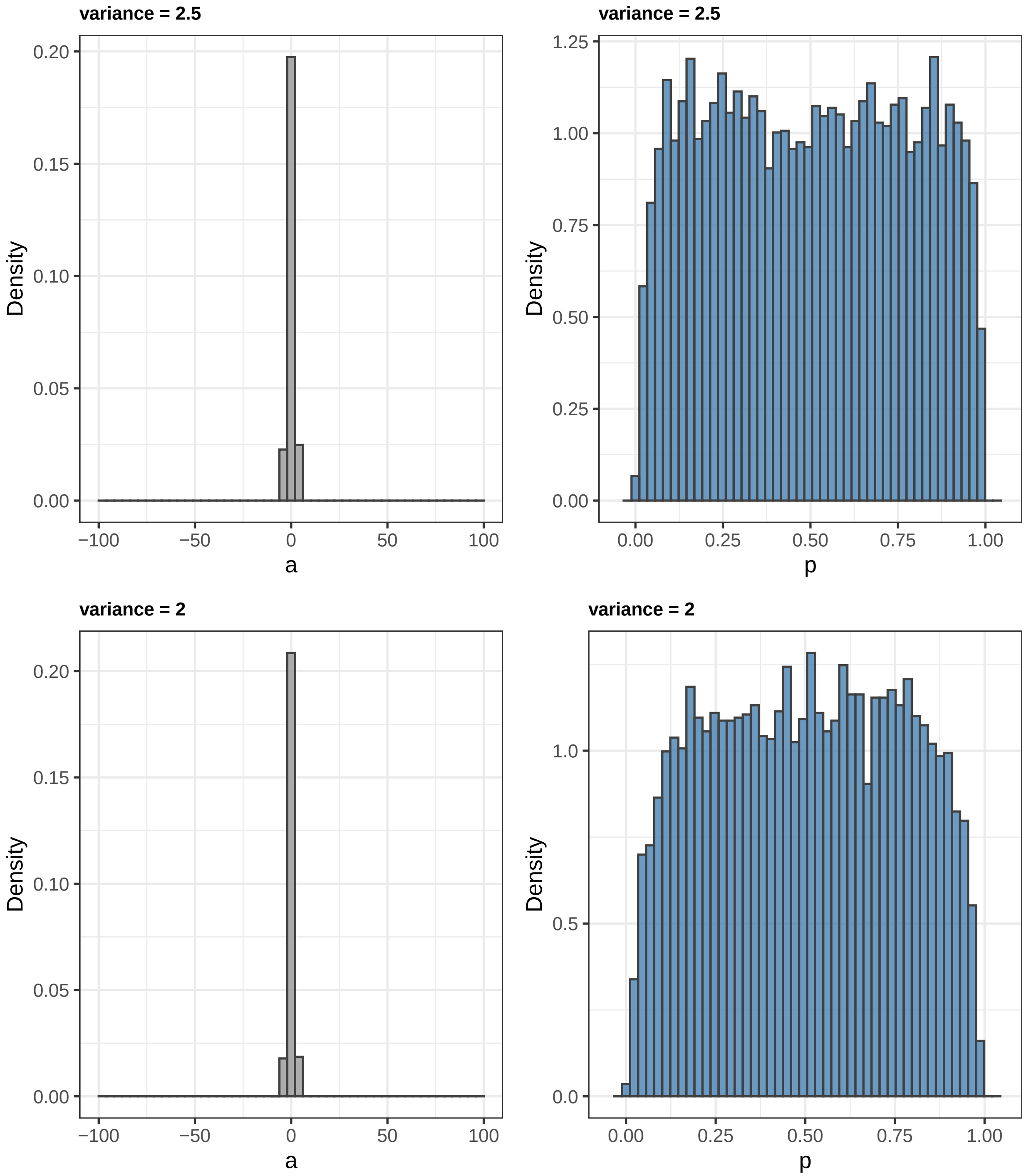

variance = 10
0.12
0.09
0.06
0.03
0.00
Density
−100
−50
0
50
100
a
variance = 10
4
3
2
1
0
Density
0.00
0.25
0.50
0.75
1.00
p
variance = 5
0.15
0.10
0.05
0.00
Density
−100
−50
0
50
100
a
variance = 5
2.0
1.5
1.0
0.5
0.0
Density
0.00
0.25
0.50
0.75
1.00
p



```r
# hone prior so that p is flat
c(
  rev(seq(2, 4, 0.5))
) %>%
purrr::map(my_plot_fn)
```




variance = 3.5
Density
0.15
0.10
0.05
0.00
−100
−50
0
50
100
a
variance = 3.5
Density
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
p
variance = 3
Density
0.15
0.10
0.05
0.00
−100
−50
0
50
100
a
variance = 3
Density
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
p



Visual inspection of the plots above indicates that a variance $\sigma_p^2 = 2.5$ is much less informative than a variance $\sigma_p^2 = 10000$ which was used in the above analysis.

"*The use of a normal prior with large, but finite, variance seems to work well without complications for parameters that are means and where the data contain plenty of information. However, for other types of parameters, say transformations of probabilities such as logit(p), the normal prior with large variance can have a dubious influence on the posterior.*"

Hobbs, N. Thompson, and Mevin B. Hooten. *Bayesian Models : A Statistical Primer for Ecologists*, Princeton University Press, 2015.

Question 2

Rerun your analysis using priors on the coefficients that are vague for inference on p based on what you learned in Hobbs and Hooten section 5.4.1 and in the previous exercise. (Be careful to convert variances to precision) Plot the probability of occupancy as a function of perimeter to area ratio using these priors and compare with the plot you obtained in exercise 5, above. You will see that the means of the pi changes and uncertainty about pi increases when you use appropriately vague priors for p.

Implement JAGS Model

Using a prior variance $\sigma_p^2 = 2.5$ to be much less informative:

```
variance <- 2.5
precision <- 1 / variance
model{
    # priors
    a ~ dnorm(0, precision)
    b ~ dnorm(0, precision)
    # likelihood
    for (i in 1:n) {
      p[i] <- ilogit(a + b*x[i])
      y[i] ~ dbern(p[i])
    }
    ## quantities of interest
      # The predicted probability of occupancy
      for(j in 1:length(perim_area_z)){
        new_p_est[j] <- ilogit(a + b*perim_area_z[j])
      }
  }
```

```
##################################################################
# insert JAGS model code into an R script
##################################################################
{ # Extra bracket needed only for R markdown files - see answers
  sink("temp_LizardsJAGS.R") # This is the file name for the jags code
  cat("
  model{
    # priors
    a ~ dnorm(0,0.4)
    b ~ dnorm(0,0.4)
    # likelihood
    for (i in 1:n) {
      p[i] <- ilogit(a + b*x[i])
      y[i] ~ dbern(p[i])
    }
    ## quantities of interest
      # The predicted probability of occupancy
      for(j in 1:length(perim_area_z)){
        new_p_est[j] <- ilogit(a + b*perim_area_z[j])
      }

  }
  ", fill = TRUE)
  sink()
}
######################
# Call to JAGS
######################
temp_jm = rjags::jags.model(
  file = "temp_LizardsJAGS.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: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 1048
##
## Initializing model
```

```
stats::update(temp_jm, n.iter = n.update)
# save the coda object (more precisely, an mcmc.list object) to R as "zm"
temp_zm = rjags::coda.samples(
  model = temp_jm
  , variable.names = c("a", "b", "new_p_est")
  , n.iter = n.iter
  , n.thin = 1
)
#####################
# check output
#####################
# summary
MCMCvis::MCMCsummary(temp_zm, params = c("a", "b"))
```

```
##          mean        sd      2.5%        50%      97.5% Rhat n.eff
## a -0.1770008 0.5654978 -1.315550 -0.1711633  0.9255501    1 16076
## b -2.3724611 0.9117601 -4.330344 -2.3211008 -0.7671327    1 14930
```

```
# chain 1 first 6 iterations and specific columns
temp_zm[[1]][1:6, c("a", "b")]
```

```
##                 a         b
## [1,] -0.07734108 -3.265032
## [2,] -0.41268075 -2.899948
## [3,]  0.16431162 -1.020199
## [4,]  0.06335756 -3.838069
## [5,] -0.63774554 -3.765747
## [6,] -0.09824442 -2.772216
```

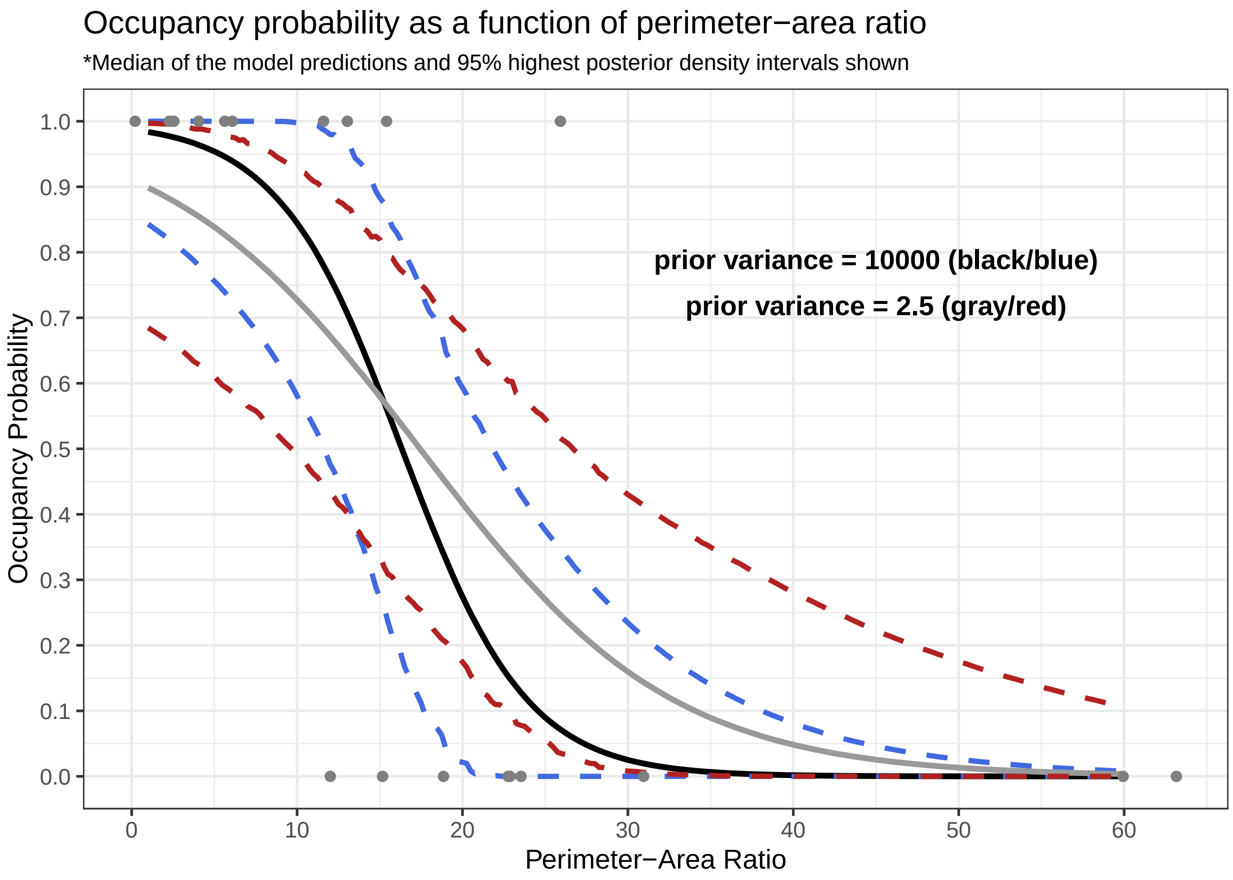
```r
# The rate of occupancy
MCMCvis::MCMCpstr(temp_zm, params = "new_p_est", func = function(x) quantile(x, c(0.025, 0.5, 0.975))) %
  as.data.frame() %>%
  dplyr::bind_cols(perim_area_z = perim_area_z) %>%
  dplyr::slice_head(n = 6)
```

```
##              new_p_est.2.5. new_p_est.50. new_p_est.97.5. perim_area_z
## new_p_est[1]      0.6262333     0.8986018       0.9855445   -1.0143901
## new_p_est[2]      0.6227128     0.8955321       0.9847497   -1.0000935
## new_p_est[3]      0.6196537     0.8922694       0.9839003   -0.9857970
## new_p_est[4]      0.6160793     0.8889079       0.9829540   -0.9715005
## new_p_est[5]      0.6138645     0.8855640       0.9819924   -0.9572040
## new_p_est[6]      0.6103771     0.8821901       0.9809148   -0.9429075
```

Plot

Plot the data as points. Overlay a line plot of the median and 95% highest posterior density intervals of the predicted probability of occurrence as a function of island perimeter to area ratios ranging from 1-60.

```r
dplyr::bind_cols(
  # original prior variance
  perim_area = perim_area
  , median_p_est = MCMCvis::MCMCpstr(zm, params = "p_est", func = median) %>% unlist()
  , MCMCvis::MCMCpstr(zm, params = "p_est", func = function(x) HDInterval::hdi(x, credMass = 0.95)) %>%
  # new prior variance
  , new_median_p_est = MCMCvis::MCMCpstr(temp_zm, params = "new_p_est", func = median) %>% unlist()
  , MCMCvis::MCMCpstr(temp_zm, params = "new_p_est", func = function(x) HDInterval::hdi(x, credMass = 0
) %>%
# plot
ggplot(data = .) +
  geom_line(mapping  = aes(x = perim_area, y = median_p_est), color = "black", lwd = 1.1) +
  geom_line(mapping  = aes(x = perim_area, y = p_est.upper), color = "royalblue", lwd = 1, linetype = "
  geom_line(mapping  = aes(x = perim_area, y = p_est.lower), color = "royalblue", lwd = 1, linetype = "
  # new
  geom_line(mapping  = aes(x = perim_area, y = new_median_p_est), color = "gray60", lwd = 1.1) +
  geom_line(mapping  = aes(x = perim_area, y = new_p_est.upper), color = "firebrick", lwd = 1, linetype
  geom_line(mapping  = aes(x = perim_area, y = new_p_est.lower), color = "firebrick", lwd = 1, linetype
  # add sample data
  geom_point(
    data = data_df
    , mapping = aes(x = perimeterAreaRatio, y = presence)
    , color = "gray50"
  ) +
  annotate("text"
           , x = 45
           , y = 0.75
           , label = 'atop(bold("prior variance = 10000 (black/blue)"), bold("prior variance = 2.5 (gra
           , parse = TRUE
  ) +
  scale_y_continuous(breaks = scales::extended_breaks(n=10)) +
  scale_x_continuous(breaks = scales::extended_breaks(n=10), labels = scales::comma) +
  xlab("Perimeter-Area Ratio") +
  ylab("Occupancy Probability") +
  labs(
    title = "Occupancy probability as a function of perimeter-area ratio"
    , subtitle = "*Median of the model predictions and 95% highest posterior density intervals shown"
    # , caption = "prior variance = 10000 (black/blue); prior variance = 2.5 (gray/red)"
  ) +
  theme_bw() +
  theme(
    plot.subtitle = element_text(size = 9)
    , plot.caption = element_text(size = 10, face = "bold")
  )
```



These is conflict between priors that are vague for the parameters and vague for the predictions of the model. If your primary inference is on p then you want to choose values for the priors on a and b that are minimally informative for p. The simulation exercise above shows a way to do that. However, what if you need inference on a, b, and p? There are two possibilities. First, get more data so that the influence of the prior becomes negligible. The best way to assure priors are vague is to collect lots of high quality data.

Second, use informative priors on the coefficients, even weakly informative ones. For example, you *know* that the slope should be negative and you *know* something about the probability of occupancy when islands are large. Centering the slope on a negative value rather than 0 makes sense because we know from many other studies that the probability of occupancy goes down as islands get smaller. Moreover, you could center the prior on the intercept on 3 using the reasoning that large islands are almost certainly occupied (when intercept = 3, p = .95 at PA = 0). Centering the priors on reasonable values (rather than 0) will make the results more precise and far less sensitive to the variance (or precision) chosen for the prior. Informative priors, even weakly informative ones, are helpful in many ways. We should use them.

You could explore these solutions on a Sunday afternoon using the code your wrote above.