ESS 575: JAGS Problems Lab

Team England

10 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                  1237.8408262 6.325654e+01 1128.4115503 1232.2687072
## r                     0.2006944 9.755559e-03    0.1813817    0.2007637
## sigma                 0.0286733 3.018552e-03    0.0234686    0.0284119
## tau                1256.0995519 2.585327e+02  802.7486971 1238.7948187
## N_max_pop_grwth_rt  618.9204131 3.162827e+01  564.2057751  616.1343536
##                           97.5% Rhat n.eff
## K                  1.377199e+03    1  7141
## r                  2.197134e-01    1  7518
## sigma              3.529476e-02    1 13985
## tau                1.815622e+03    1 15710
## N_max_pop_grwth_rt 6.885995e+02    1  7141
```

```
# 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,] 1377.859 0.1952503 0.02746323 1325.858           688.9294
## [2,] 1311.971 0.1916739 0.02803723 1272.125           655.9855
## [3,] 1226.860 0.1965458 0.02669876 1402.871           613.4299
## [4,] 1209.960 0.2087737 0.02518746 1576.272           604.9802
## [5,] 1173.551 0.2014230 0.02716596 1355.033           586.7754
## [6,] 1217.623 0.1944631 0.02891243 1196.274           608.8113
```

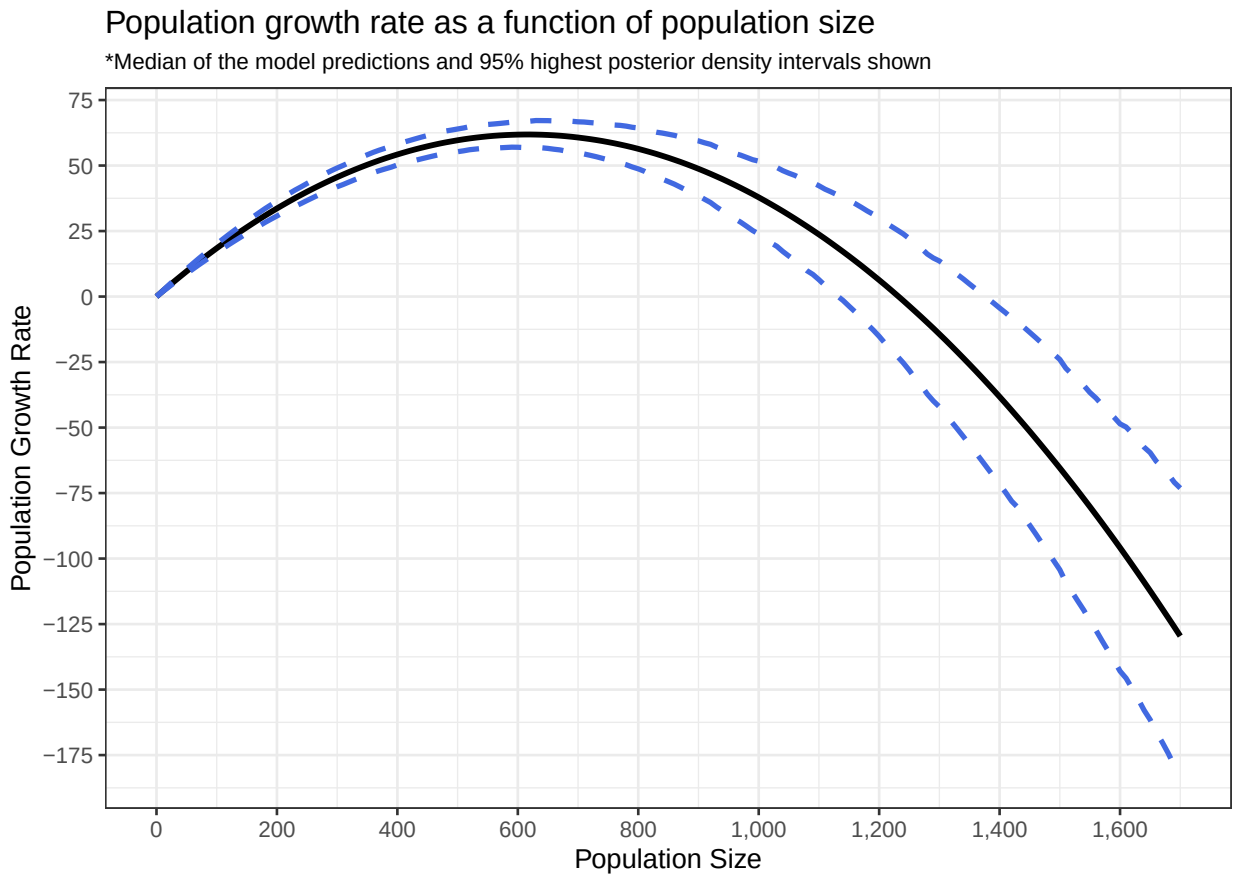
```
# The rate of population growth
MCMCvis::MCMCpstr(zm, params = "pop_grwth_rt", func = function(x) quantile(x, c(0.025, 0.5, 0.975))) %>%
  as.data.frame() %>%
  dplyr::bind_cols(N = N) %>%
  dplyr::slice_head(n = 6)
```

```
##                 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]          1.800575         1.991334           2.178275 10
## pop_grwth_rt[3]          3.573952         3.950145           4.318880 20
## pop_grwth_rt[4]          5.319473         5.876345           6.421964 30
## pop_grwth_rt[5]          7.037612         7.769489           8.486965 40
## pop_grwth_rt[6]          8.728376         9.629977          10.514581 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: 2.3%

The probability that r falls between 0.18 and 0.22 is: 95.8%

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?

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

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

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 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 $inverse logit(a+b \, \cdot \, standardized(x_i))$ is b
  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
#######################
# 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
```

```
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,] -0.8911061 -1.441693
## [2,]  0.8162864 -1.828787
## [3,]  0.5196094 -3.134151
## [4,] -0.1627937 -8.900142
## [5,] -0.3541986 -7.618712
## [6,] -1.2444863 -3.826821
```

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.

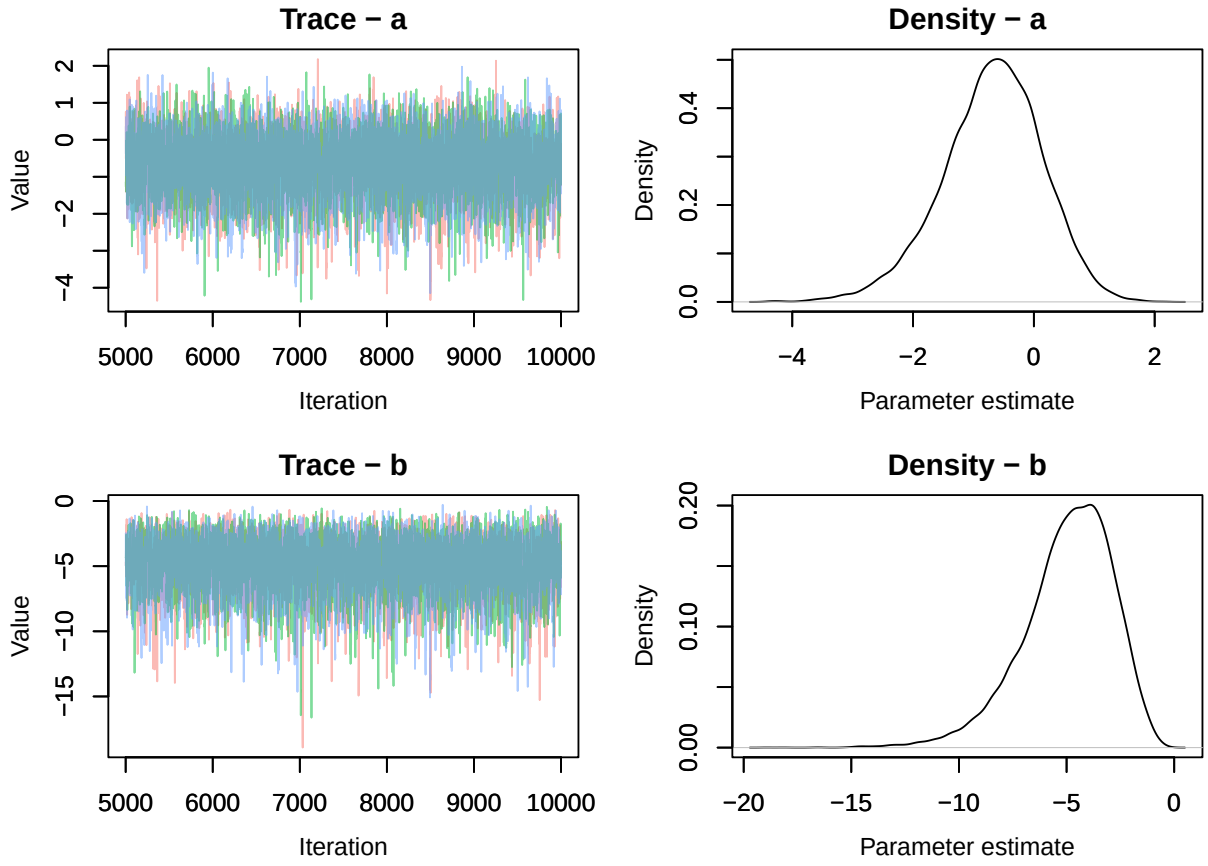
```
# summary
MCMCvis::MCMCsummary(zm, params = c("a", "b")) %>%
    kableExtra::kable(
    caption = "Summary of simulations for parameters a and b"
    , digits = 5
  ) %>%
  kableExtra::kable_styling(font_size = 11) %>%
```

```
  kableExtra::column_spec(1, bold = TRUE, width = "18em") %>%
  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	-0.69578	0.83143	-2.48581	-0.64920	0.80250	1	11431
b	-4.98331	2.10618	-9.86259	-4.71442	-1.66422	1	9180

```
# 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         0.529
## b passed       1         0.672
##
##   Halfwidth Mean   Halfwidth
##   test
## a passed    -0.698 0.0258
## b passed    -4.966 0.0732
##
## [[2]]
##
##   Stationarity start     p-value
##   test         iteration
## a passed       1         0.242
## b passed       1         0.219
##
##   Halfwidth Mean   Halfwidth
##   test
## a passed    -0.691 0.0270
## b passed    -4.968 0.0757
##
## [[3]]
##
##   Stationarity start     p-value
##   test         iteration
## a passed       1001      0.212
## b passed          1      0.957
##
##   Halfwidth Mean   Halfwidth
##   test
## a passed    -0.707 0.0281
## b passed    -5.017 0.0751
```

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 -0.6869656 0.8433019 -2.516125 -0.6283877  0.8127678    1 10801
## b -4.9832345 2.1359021 -9.949597 -4.7062101 -1.6348432    1  9615
```

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

```
##               a         b
## [1,]  0.6311295 -3.245100
## [2,]  0.3062076 -3.635709
## [3,]  0.2002269 -3.151254
## [4,] -0.3866619 -5.958355
## [5,] -1.1406050 -6.784476
## [6,] -1.9389496 -4.200398
```

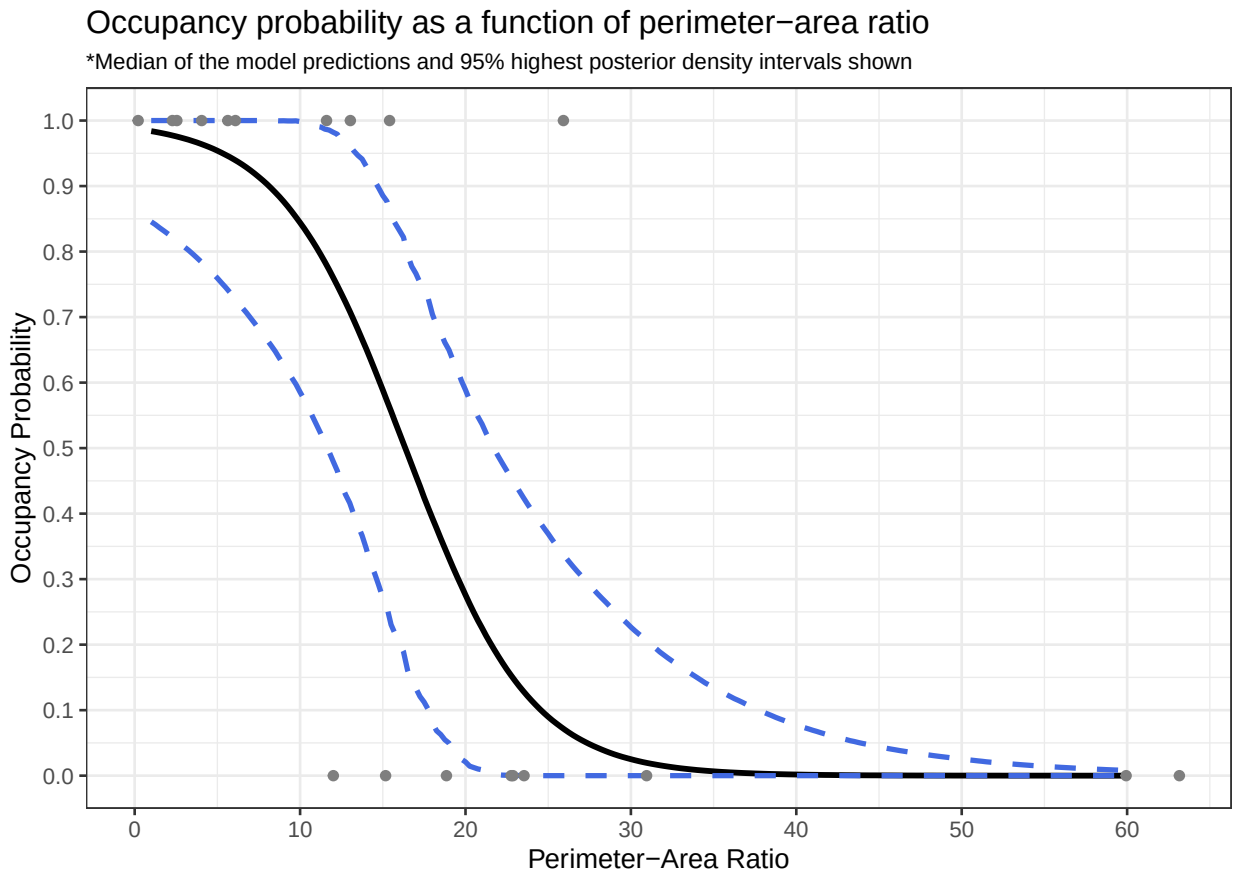
```
# The rate of occupancy
MCMCvis::MCMCpstr(zm, params = "p_est", func = function(x) quantile(x, c(0.025, 0.5, 0.975))) %>%
  as.data.frame() %>%
  dplyr::bind_cols(perim_area_z = perim_area_z) %>%
  dplyr::slice_head(n = 6)
```

```
##          p_est.2.5. p_est.50. p_est.97.5. perim_area_z
## p_est[1]  0.7901831 0.9838287   0.9998581   -1.0143901
## p_est[2]  0.7852558 0.9827166   0.9998380   -1.0000935
## p_est[3]  0.7805999 0.9815187   0.9998141   -0.9857970
## p_est[4]  0.7757114 0.9802447   0.9997892   -0.9715005
## p_est[5]  0.7700668 0.9789015   0.9997561   -0.9572040
## p_est[6]  0.7647121 0.9774437   0.9997199   -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(
  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       0.8196195 0.1224302 0.53062513 0.8438423 0.9833386    1 24622
## p_x20       0.2906231 0.1585867 0.04239939 0.2735999 0.6326536    1 11816
```

```
## diff_x10_x20 0.5289964 0.1731777 0.20188152 0.5276471 0.8544886    1 11254
```

```
# 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,] -0.4670135 -4.191170 0.8358104 0.3166123    0.5191980
## [2,] -0.7373616 -3.858915 0.7669196 0.2658538    0.5010658
## [3,] -0.2807892 -3.318942 0.7986275 0.3727977    0.4258298
## [4,] -0.3202651 -4.823301 0.8899288 0.3388909    0.5510379
## [5,] -1.1850000 -4.168259 0.7105269 0.1845653    0.5259616
## [6,] -1.3256669 -4.432477 0.7087481 0.1617253    0.5470228
```

```
# 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)
```

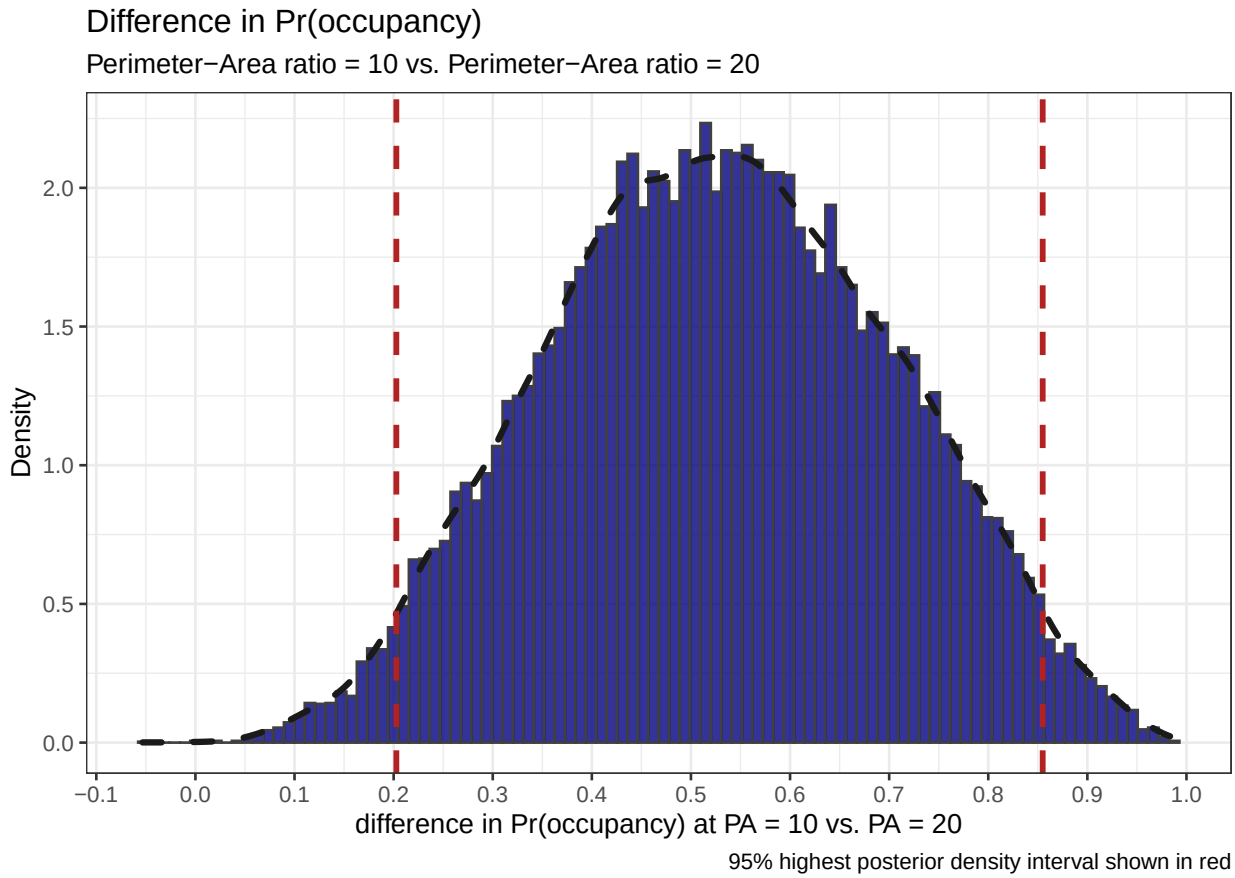
```
##          p_est.2.5. p_est.50. p_est.97.5. perim_area_z
## p_est[1]  0.7926358 0.9838226   0.9998606   -1.0143901
## p_est[2]  0.7881185 0.9827227   0.9998403   -1.0000935
## p_est[3]  0.7837109 0.9815681   0.9998166   -0.9857970
## p_est[4]  0.7788824 0.9802885   0.9997896   -0.9715005
## p_est[5]  0.7736639 0.9789585   0.9997574   -0.9572040
## p_est[6]  0.7689706 0.9775124   0.9997241   -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 N at the maximum population growth rate
  # 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 **0.203** and **0.855**

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?

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