ESS 575: Multi-Level Regression Lab

Team England

19 October, 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

Preliminaries

Motivation

Each section of this lab has two parts – a model *building* exercise and a model *coding* exercise. The material covered here is important and broadly useful – building multi-levels models is a true workhorse for understanding ecological processes because so many problems contain information at nested spatial scales, levels of organization, or categories. It will be worthwhile to dig in deeply to understand it. The big picture is to demonstrate the flexibility that you gain as a modeler by understanding basic principles of Bayesian analysis. To accomplish that, these exercises will reinforce the following:

1. Diagramming and writing hierarchical models
2. Using data to model parameters
3. JAGS coding
4. Creating index variables, a critically important and useful skill
5. Posterior predictive checks

Introduction

Ecological data are often collected at multiple scales or levels of organization in nested designs. Group is a catchall term for the upper level in many different types of nested hierarchies. Groups could logically be composed of populations, locations, species, treatments, life stages, and individual studies, or really, any sensible category. We have measurements within groups on individual organisms, plots, species, time periods, and so on. We may also have measurements on the groups themselves, that is, covariates that apply at the upper level of organization or spatial scale or the category that contains the measurements. Multilevel models represent the way that a quantity of interest responds to the combined influence of observations taken at the group level and within the group.

Nitrous oxide N_2O, a greenhouse gas roughly 300 times more potent than carbon dioxide in forcing atmospheric warming, is emitted when synthetic nitrogenous fertilizers are added to soils. Qian and colleagues (2010) conducted a Bayesian meta-analysis of N_2O emissions (g $N \cdot ha^{-1} \cdot d^{-1}$) from agricultural soils using data from a study conducted by Carey (2007), who reviewed 164 relevant studies. Studies occurred

at different locations, forming a group-level hierarchy (we will use only sites that have both nitrogen and carbon data, which reduces the number of sites to 107 in the analysis here). Soil carbon content (g · organic C · g^{-1} soil dry matter) was measured as a group-level covariate and is assumed to be measured without error. Observations of N_2O emission are also assumed to be measured without error and were paired with measurements of fertilizer addition (kg N· ha^{-1} · $year^{-1}$). The effect of different types of fertilizer was also studied.

You are going to use these data to build increasingly complex models of N_2O emission. The initial models will ignore some important covariates as well as how the data are structured hierarchically into sites. This is ok! When writing for a multi-level model like this one, do it incrementally, starting with a separate model for each site (the no-pool model) or a model that ignores sites entirely (the pooled model). After getting these models to work you can add complexity by drawing the intercept for each model from a distribution, before pursuing further refinements. We **strongly sugggest** this approach because it is always best to do the simple thing first: there is less to go wrong. Also, when things do go wrong it will be clearer as to what is causing the problem.

R libraries needed for this lab

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

```
# bread-and-butter
library(tidyverse)
library(lubridate)
library(viridis)
library(scales)
library(latex2exp)
# visualization
library(cowplot)
library(kableExtra)
# jags and bayesian
library(actuar)
library(rjags)
library(ggthemes)
library(gridExtra)
library(MCMCvis)
library(HDInterval)
library(BayesNSF)
library(reshape2)
#set seed
set.seed(10)
```

Pooled

Diagramming and writing the pooled model

Let's begin by ignoring the data on soil carbon and fertilizer type. In addition, we will ignore site, such that all observations are treated as independent from one another. This is what's known as complete pooling - see Gelman and Hill, (2007), or just a pooled model. You will use a linearized power function for your deterministic model of emissions as a function of nitrogen input:

$$\mu_i = \gamma x_i^{\beta}$$
$$\alpha = \log(\gamma)$$
$$\log(\mu_i) = \alpha + \beta(\log(x_i))$$
$$g(\alpha, \beta, \log(x_i)) = \alpha + \beta(\log(x_i))$$

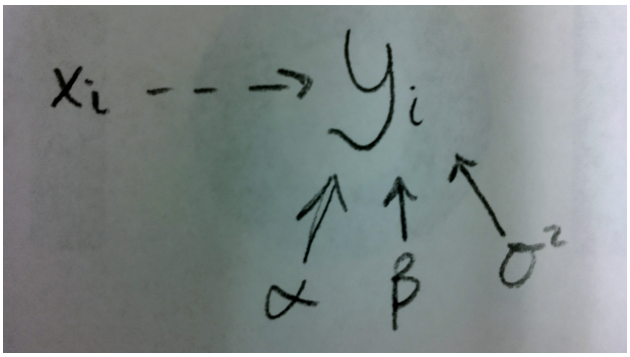
Question 1

Interpret the coefficients α, β, and γ in this model.

We are interested in modelling N_2O emission as a function of soil carbon content, fertilizer addition, and fertilizer type. We begin by ignoring the data on soil carbon and fertilizer type. In addition, we initially ignore site-level variations by pooling the data from different sites (i.e. a pooled model). In the model $\mu_i = \gamma x_i^{\beta}$, γ is the baseline scale factor for the fertilizer addition rate (x_i) impact to N_2O emission. The exponent β allows for the influence of fertilizer input on N_2O emission to vary with the rate of fertilizer input. Exponential regression models are used to model situations in which growth/change begins slowly and then accelerates rapidly without bound, or where decay begins rapidly and then slows down to get closer and closer to zero. The transformation $\alpha = \log(\gamma)$ allows for linear representation of the deterministic model.

Question 2

Draw a Bayesian network for a linear regression model of N_2O emission (y_i) on fertilizer addition (x_i).



DAG

Question 3

Write out the joint distribution for a linear regression model of N_2O emission (y_i) on fertilizer addition (x_i). Start by using generic `[ ]`. Use σ^2 to represent the uncertainty in your model realizing that you might need moment matching when you choose a specific distribution.

$$[\alpha, \beta, \sigma^2 \mid y_i] \propto \prod_{i=1}^{n} [\log(y_i) \mid g(\alpha, \beta, \log(x_i)), \sigma^2] [\alpha] [\beta] [\sigma]$$

Question 4

Finish by choosing specific distributions for likelihoods and priors. You will use the math in the answer as a template to code your model in the subsequent exercises. What are assuming about the distribution of the untransformed μ_i?

$$[\alpha, \beta, \sigma^2 \mid y_i] \propto \prod_{i=1}^{n} \text{normal}\big(\log(y_i) \mid g(\alpha, \beta, \log(x_i)), \sigma^2\big) \times \text{normal}\big(\alpha \mid 0, 10000\big) \times \text{normal}\big(\beta \mid 0, 10000\big) \times \text{uniform}\big(\sigma \mid 0, 100\big)$$

Question 5

What is the hypothesis represented by this model?

We are ignoring site-level variations by pooling the data from different sites (i.e. a pooled model). This means that we are assuming that the emissions response to nitrogen addition does not vary across sites. In this pooled model, we are allowing N_2O emission to increase exponentially with fertilizer application rate.

Visualizing the pooled data

It is always a good idea to look at the data. Examine the head of the data frame for emissions. Note that the columns `group.index` and `fert.index` contain indices for sites and fertilizer types. We are going to ignore these for now since the pooled model does not take these into account. Use the code below to plot N_2O emissions as a function of fertilizer input for both the logged and unlogged data.

```
# view the first few rows of data
BayesNSF::N2OEmission %>%
  head()
```

```
##   fertilizer group carbon n.input emission reps group.index fert.index
## 1          A    14    2.7     180    0.620   13          10          2
## 2          A    14    4.6     180    0.450   13          10          2
## 3          A    11    0.9     112    0.230   12           7          2
## 4          A    38    0.5     100    0.153   14          29          2
## 5          A     1    4.0     250    1.000    6           1          2
## 6          A    38    0.5     100    0.216   14          29          2
```

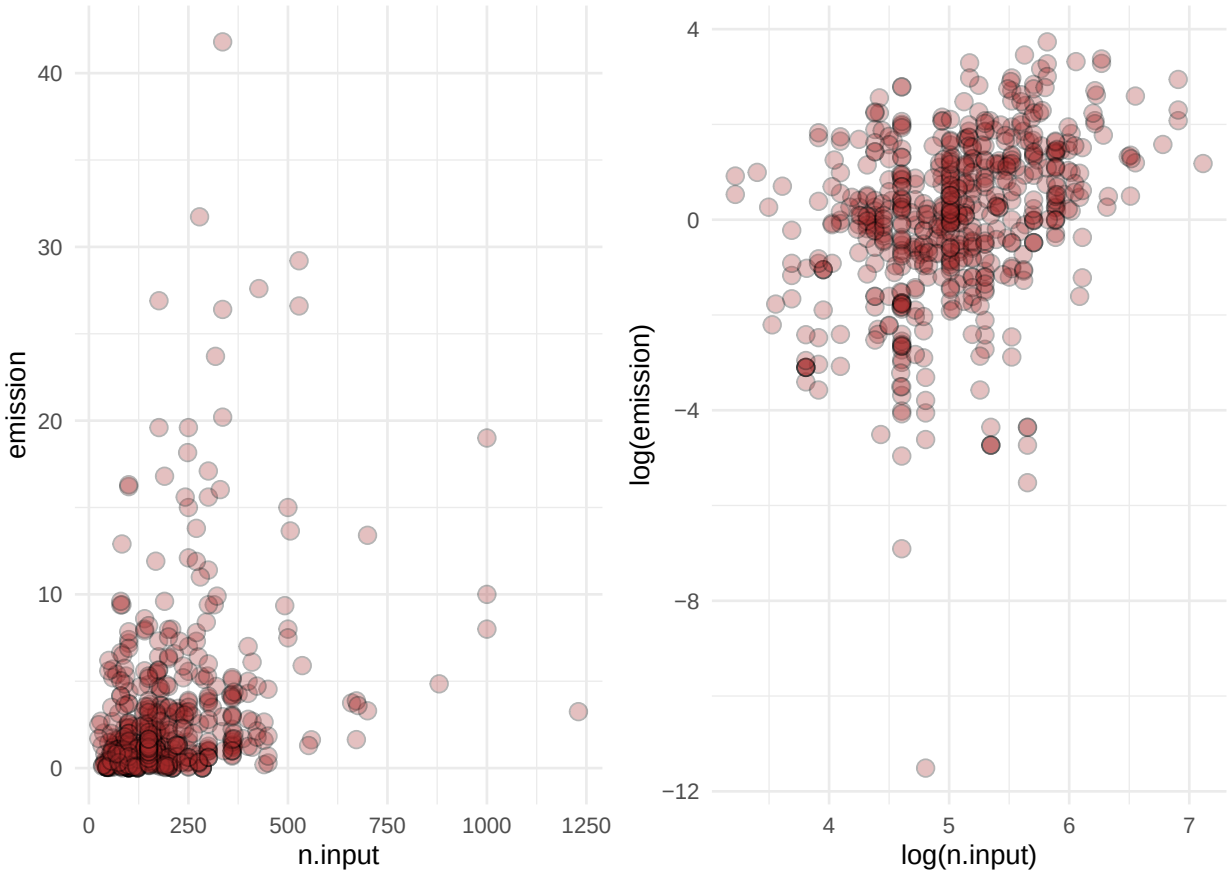
```
# data structure
BayesNSF::N2OEmission %>%
  dplyr::glimpse()
```

```
## Rows: 563
## Columns: 8
## $ fertilizer  <fct> A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A~
## $ group       <int> 14, 14, 11, 38, 1, 38, 11, 11, 11, 11, 13, 13, 13, 13, 1, ~
## $ carbon      <dbl> 2.70, 4.60, 0.90, 0.50, 4.00, 0.50, 0.90, 0.90, 0.90, 0.90~
## $ n.input     <dbl> 180, 180, 112, 100, 250, 100, 112, 112, 117, 82, 112, 112,~
## $ emission    <dbl> 0.620, 0.450, 0.230, 0.153, 1.000, 0.216, 0.240, 0.890, 0.~
## $ reps        <int> 13, 13, 12, 14, 6, 14, 12, 12, 12, 12, 14, 14, 14, 14, 6, ~
## $ group.index <int> 10, 10, 7, 29, 1, 29, 7, 7, 7, 7, 9, 9, 9, 9, 1, 9, 1, 7, ~
## $ fert.index  <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
```

We are going to use `ggplot` to visualize the data in this lab. If you are unfamiliar with this package, don't worry. We will provide you will all the codes you need and help your get oriented. We think you will find the plotting functions in `ggplot` very powerful and intuitive. We start by using `ggplot` to load the data frame we will plot data from. Then we add `geom_point` and use the `aes` argument (the aesthetic mappings) to define the x and y values for the points. All `ggplot` functions require you to define the aesthetic mappings

as needed. Here, they are the same as setting x and y in the normal plot functions. The other big difference is that `ggplot` allows you to add successive layers to the plot using the `+` operator. You will see later on that this offers a lot of flexibility. We add the `geom_line` feature and then set the theme to `minimal`. Lastly, we use the `grid.arrange` function to position multiple plots at once. This is similar to using `mfrow` with `par`.

```
# untransformed
g1 <- ggplot(data = BayesNSF::N2OEmission) +
  geom_point(
    mapping = aes(y = emission, x = n.input)
    , alpha = 3/10
    , shape = 21
    , colour = "black"
    , fill = "brown"
    , size = 3
  ) +
  theme_minimal()
# log transformed
g2 <- ggplot(data = BayesNSF::N2OEmission) +
  geom_point(
    mapping = aes(y = log(emission), x = log(n.input))
    , alpha = 3/10
    , shape = 21
    , colour = "black"
    , fill = "brown"
    , size = 3
  ) +
  theme_minimal()
# plot side by side
gridExtra::grid.arrange(g1, g2, nrow = 1)
```



Fitting the pooled model with JAGS

You will now write a simple, pooled model where you gloss over differences in sites and fertilizer types and lump everything into a set of x and y pairs using the R template provided below. It is imperative that you study the data statement and match the variable names in your JAGS code to the left hand side of the = in the data list. Call the intercept `alpha`, the slope `beta` and use `sigma` to name the standard deviation in the likelihood. Also notice, that we center the nitrogen input covariate to speed convergence. You could also standardize this as well.

In addition to fitting this model, we would like you to have JAGS predict the mean logged N_2O emissions and the median unlogged N_2O emissions as a function of soil fertilizer input. (Why median? Hint: think back to the distribution of the untransformed data above in question 3 above). To help you out we have provided the range of N_2O values to predict over as the third element in the `data` list. Make sure you understand how we chose these values.

Note that in this problem and the ones that follow we have set up the data and the initial conditions for you. This will save time and frustration, allowing you to concentrate on writing code for the model but you must pay attention to the names we give in the `data` and `inits` lists. These must agree with the variable names in your model. Please see any of the course instructors if there is anything that you don't understand about these lists.

```
n.input.pred <- seq(min(BayesNSF::N2OEmission$n.input), max(BayesNSF::N2OEmission$n.input), 10)

data = list(
  log.emission = log(BayesNSF::N2OEmission$emission) %>%
```

```
      as.double()
  , log.n.input.centered = log(BayesNSF::N2OEmission$n.input) -
      mean(log(BayesNSF::N2OEmission$n.input)) %>%
        as.double()
  , log.n.input.centered.pred = log(n.input.pred) -
      mean(log(BayesNSF::N2OEmission$n.input)) %>%
        as.double()
)

inits = list(
  list(alpha = 0, beta = .5, sigma = 50)
  , list(alpha = 1, beta = 1.5, sigma = 10)
  , list(alpha = 2, beta = .75, sigma = 20)
)
```

Question 6

Write the code for the model. Compile the model and execute the MCMC to produce a coda object. Produce trace plots of the chains for model parameters. Produce a summary table and caterpillar plot for the parameters and tests for convergence including the effective sample size.

```
## JAGS Model
model{

  # priors
  alpha ~ dnorm(0,1E-6)
  beta ~ dnorm(0,1E-6)
  sigma ~ dunif(0,100)
  tau <- 1/sigma^2

  # likelihood
  for (i in 1:length(log.emission)) {
    log_mu[i] <- alpha + beta * log.n.input.centered[i]
    log.emission[i] ~ dnorm(log_mu[i], tau)
  }

  ## quantities of interest
    # predicted emissions
    for (j in 1:length(log.n.input.centered.pred)) {
      log_mu_pred[j] <- alpha + beta * log.n.input.centered.pred[j]
      mu_pred[j] <- exp(log_mu_pred[j])
    }
}
```

JAGS Model

```
#######################################################################
# insert JAGS model code into an R script
#######################################################################
{ # Extra bracket needed only for R markdown files - see answers
  sink("NO2JAGS_pooled.R") # This is the file name for the jags code
  cat("
  model{
      # priors
      alpha ~ dnorm(0,1E-6)
      beta ~ dnorm(0,1E-6)
      sigma ~ dunif(0,100)
      tau <- 1/sigma^2

      # likelihood
      for (i in 1:length(log.emission)) {
        log_mu[i] <- alpha + beta * log.n.input.centered[i]
        log.emission[i] ~ dnorm(log_mu[i], tau)
      }

      ## quantities of interest
        # predicted emissions
        for (j in 1:length(log.n.input.centered.pred)) {
          log_mu_pred[j] <- alpha + beta * log.n.input.centered.pred[j]
          mu_pred[j] <- exp(log_mu_pred[j])
        }
  }
  ", fill = TRUE)
  sink()
}
#######################################################################
# implement model
#######################################################################
# 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 = "NO2JAGS_pooled.R"
  , data = data
  , inits = inits
  , n.chains = length(inits)
  , n.adapt = n.adapt
)
```

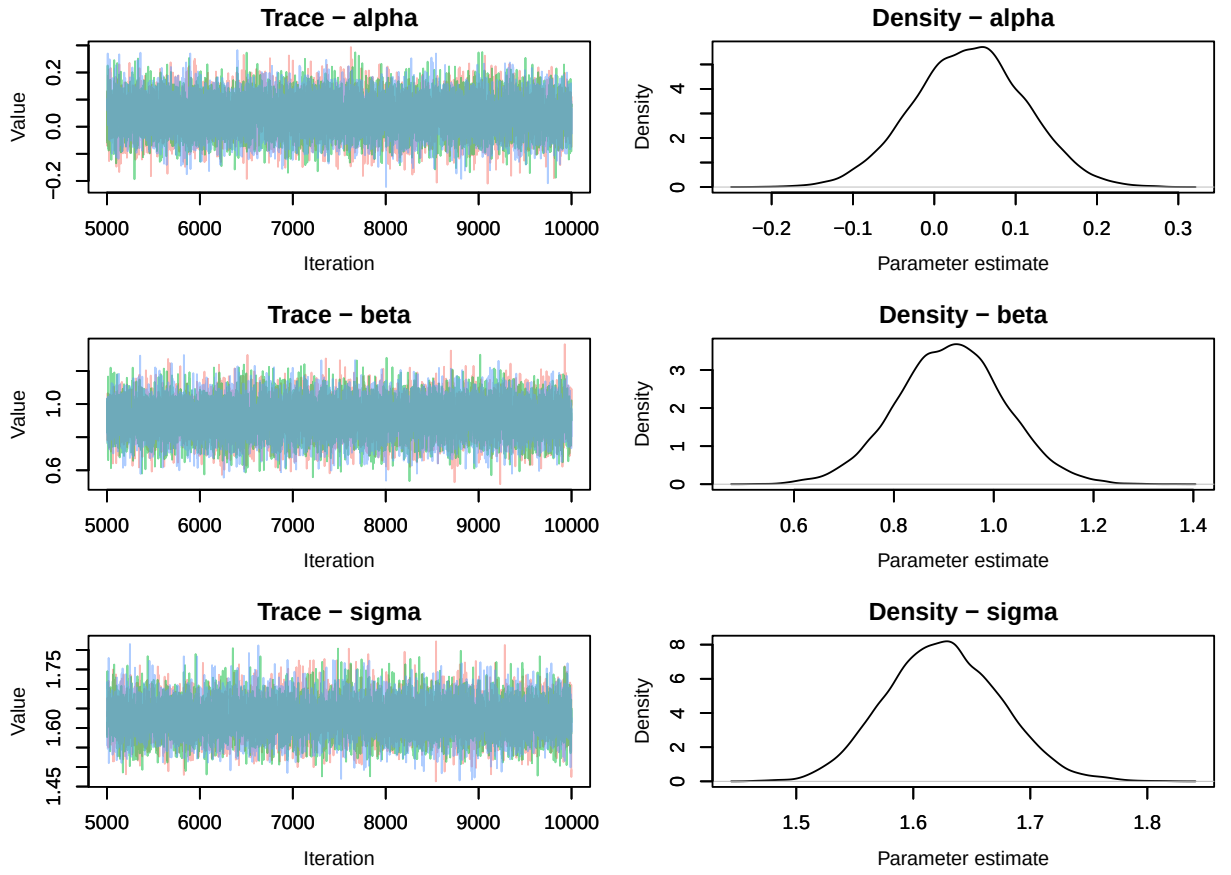
Implement JAGS Model

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 563
##    Unobserved stochastic nodes: 3
##    Total graph size: 1854
##
## Initializing model
```

```
stats::update(jm, n.iter = n.update)
# save the coda object (more precisely, an mcmc.list object) to R as "zc"
zc_pooled = rjags::coda.samples(
  model = jm
  , variable.names = c("alpha", "beta", "sigma", "tau", "log_mu_pred", "mu_pred")
  # , variable.names = c("a", "b", "p")
  , n.iter = n.iter
  , n.thin = 1
)
```

Model Output Produce trace plots of the chains for model parameters. Produce a summary table and caterpillar plot for the parameters and tests for convergence including the effective sample size.

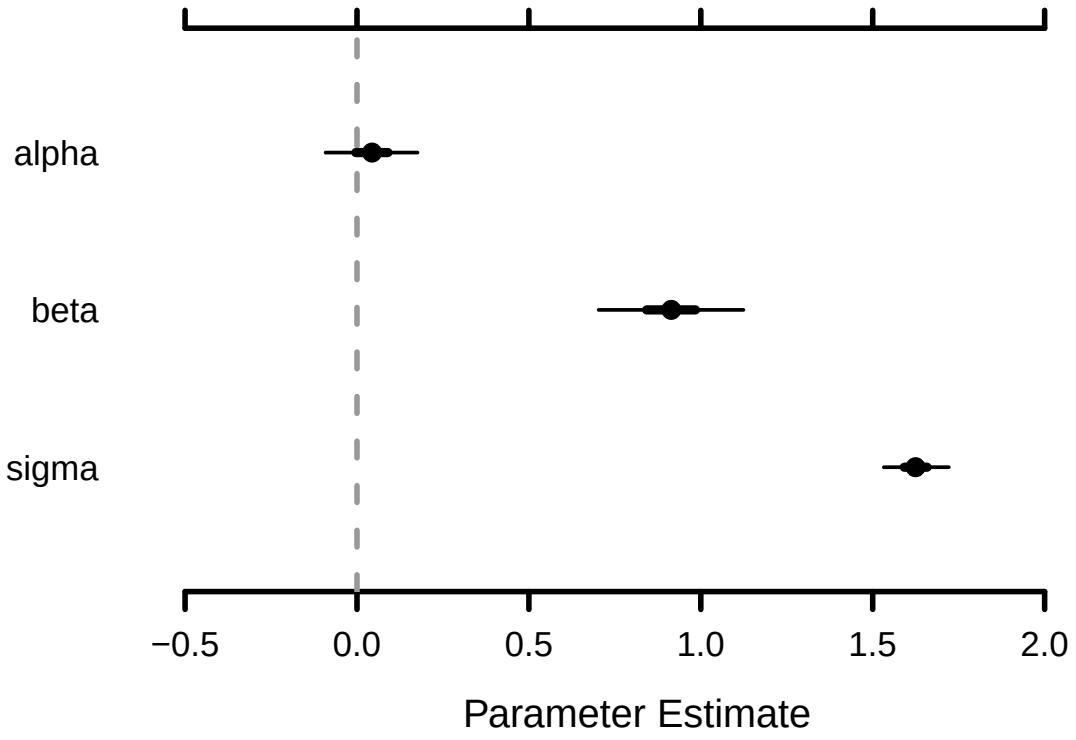
```
######################
# check output
######################
# trace plot
MCMCvis::MCMCtrace(zc_pooled, params = c("alpha", "beta", "sigma"), pdf = FALSE)
```




```
# summary
MCMCvis::MCMCsummary(zc_pooled, params = c("alpha", "beta", "sigma"))
```

```
##              mean         sd        2.5%         50%     97.5% Rhat n.eff
## alpha  0.0434061 0.06853169 -0.09139868 0.04381951 0.1761726    1 29686
## beta   0.9137811 0.10697482  0.70287836 0.91411629 1.1240026    1 29710
## sigma  1.6253459 0.04864886  1.53222722 1.62454020 1.7216007    1 19826
```

```
# Caterpillar plots
MCMCvis::MCMCplot(zc_pooled, params = c("alpha", "beta", "sigma"))
```



Visualizing the pooled model predictions

Let's overlay the predicted mean logged N_2O emissions and median unlogged N_2O emissions as a function of soil fertilizer input from the pooled model on top of the raw data. We summarize the predictions using `MCMCpstr()` twice - once to get the 95% HDPI intervals and a second time to get the posterior median for each fertilizer input value. We combine these predictions into two data frames, one for the logged N_2O emissions and one for untransformed N_2O emissions. We append our new graphical elements onto our old plots with the `+` operator. We plot the median of the posterior distribution as a black line with `geom_line()` and the 95% credible intervals as a yellow shaded region using the `geom_ribbon()` function. These data come from a different data frame than the one we used to plot the raw data so we need to add the `data` argument in the new `geom_line` and `geom_ribbon`. Again, we provide you with the code to do this to save time. You will need to modify this code to make similar plots for models you fit in later exercises.

```
# highest posterior density interval of predictions
pred1 <- MCMCvis::MCMCpstr(
  zc_pooled
  , params = c("mu_pred", "log_mu_pred")
  , func = function(x) HDInterval::hdi(x, .95)
)
# median of predictions
pred2 <- MCMCvis::MCMCpstr(
  zc_pooled
  , params = c("mu_pred", "log_mu_pred")
  , func = median
```

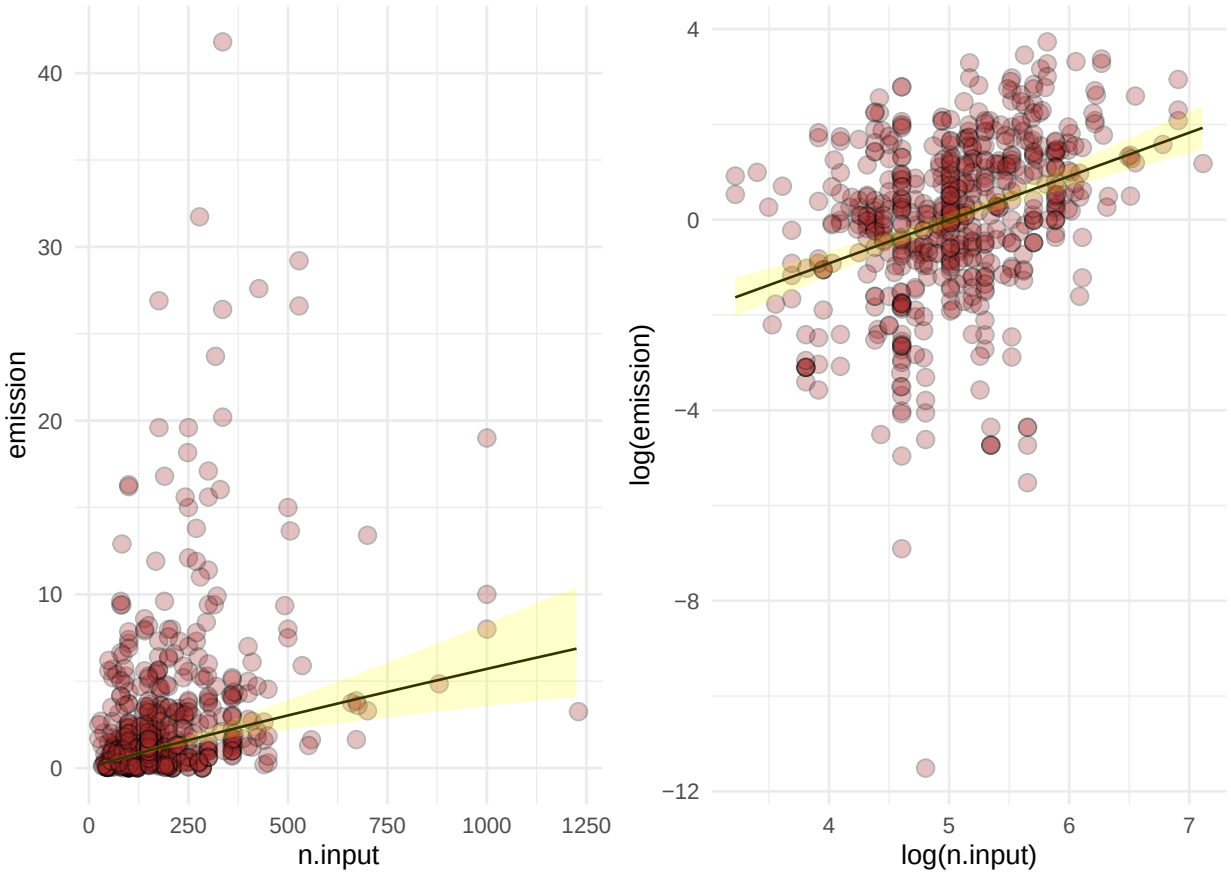
```
)
# put in data frame
pred.po.df <- dplyr::bind_cols(
  n.input.pred
  , data.frame(pred1$mu_pred)
  , median = pred2$mu_pred
)
lpred.po.df <- dplyr::bind_cols(
  log.n.input.pred = log(n.input.pred)
  , data.frame(pred1$log_mu_pred)
  , median = pred2$log_mu_pred
)
```

Plot the predictions

```
g3 <- g1 +
  geom_line(
    data = pred.po.df
    , mapping = aes(x = n.input.pred, y = median)
  ) +
  geom_ribbon(
    data = pred.po.df
    , mapping = aes(x = n.input.pred, ymin = lower, ymax = upper)
    , alpha = 0.2
    , fill = "yellow"
  )

g4 <- g2 +
  geom_line(
    data = lpred.po.df
    , mapping = aes(x = log.n.input.pred, y = median)
  ) +
  geom_ribbon(
    data = lpred.po.df
    , mapping = aes(x = log.n.input.pred, ymin = lower, ymax = upper)
    , alpha = 0.2
    , fill = "yellow"
  )

gridExtra::grid.arrange(g3, g4, nrow = 1)
```



Non-Pooled

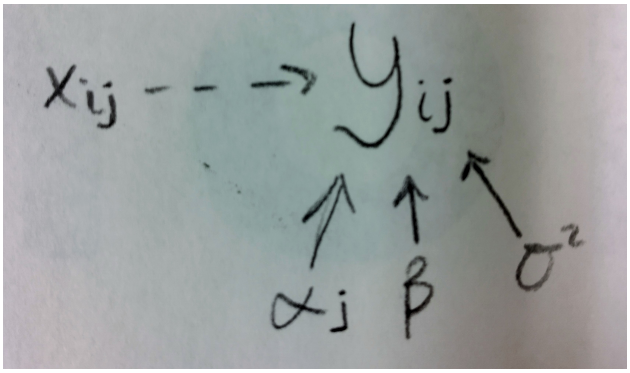
Diagramming and writing the no-pool model

Great! - you've got the pooled model fitted and made some predictions from it. However, perhaps the idea of ignoring the site effects is not sitting so well with you. Let's take this a step further by modeling the relationship between N_2O emission and fertilizer input such that the intercept α_j varies by site (we will again ignore the data on soil carbon and fertilizer type). This is the opposite of the pooled model where we completely ignored the effect of site as here we treat the intercept for each site as independent. This is commonly called a no-pool model. The deterministic portion of this model remains a linearized power function, but two subscripts are required: i which indexes the measurement within sites and j which indexes site itself.

$$\begin{aligned} \mu_{ij} &= \gamma_j x_{ij}^{\beta} \\ \alpha_j &= \log(\gamma_j) \\ \log(\mu_{ij}) &= \alpha_j + \beta(\log(x_{ij})) \\ g(\alpha_j, \beta, \log(x_{ij})) &= \alpha_j + \beta(\log(x_{ij})) \end{aligned}$$

Question 1

Draw a Bayesian network for a linear regression model of N_2O emission (y_{ij}) on fertilizer addition (x_{ij}).



DAG

Question 2

Write out the joint distribution for a linear regression model of N_2O emission (y_{ij}) on fertilizer addition (x_{ij}). Start by using generic `[ ]`. Use σ^2 to represent the uncertainty in your model realizing that you might need moment matching when you choose a specific distribution.

$$[\boldsymbol{\alpha}, \beta, \sigma^2 \mid \boldsymbol{y}] \propto \prod_{i=1}^{n} \prod_{j=1}^{J} [\log(y_{ij}) \mid g(\alpha_j, \beta, \log(x_{ij})), \sigma^2] \, [\alpha_j] \, [\beta] \, [\sigma]$$

Question 3

Finish by choosing specific distributions for likelihoods and priors. You will use the math in the answer as a template to code your model in the subsequent exercises.

$$\begin{aligned}[\boldsymbol{\alpha}, \beta, \sigma^2 \mid \boldsymbol{y}] \propto \prod_{i=1}^{n} \prod_{j=1}^{J} & \mathsf{normal}\big(\log(y_{ij}) \mid g(\alpha_j, \beta, \log(x_{ij})), \sigma^2\big) \\ & \times \mathsf{normal}\big(\alpha_j \mid 0, 10000\big) \\ & \times \mathsf{normal}\big(\beta \mid 0, 10000\big) \\ & \times \mathsf{uniform}\big(\sigma \mid 0, 100\big)\end{aligned}$$

Question 4

What is the hypothesis represented by this model?

fill this in!!!!

Visualizing the data

Let's visualize the data again, but this time highlighting the role site plays in determining the relationship between N_2O emission and fertilizer input. First, `head()` the data to see how groups are organized. You will use `group.index` to group the observations by site.

```
# view the first few rows of data
BayesNSF::N2OEmission %>%
  head()
```

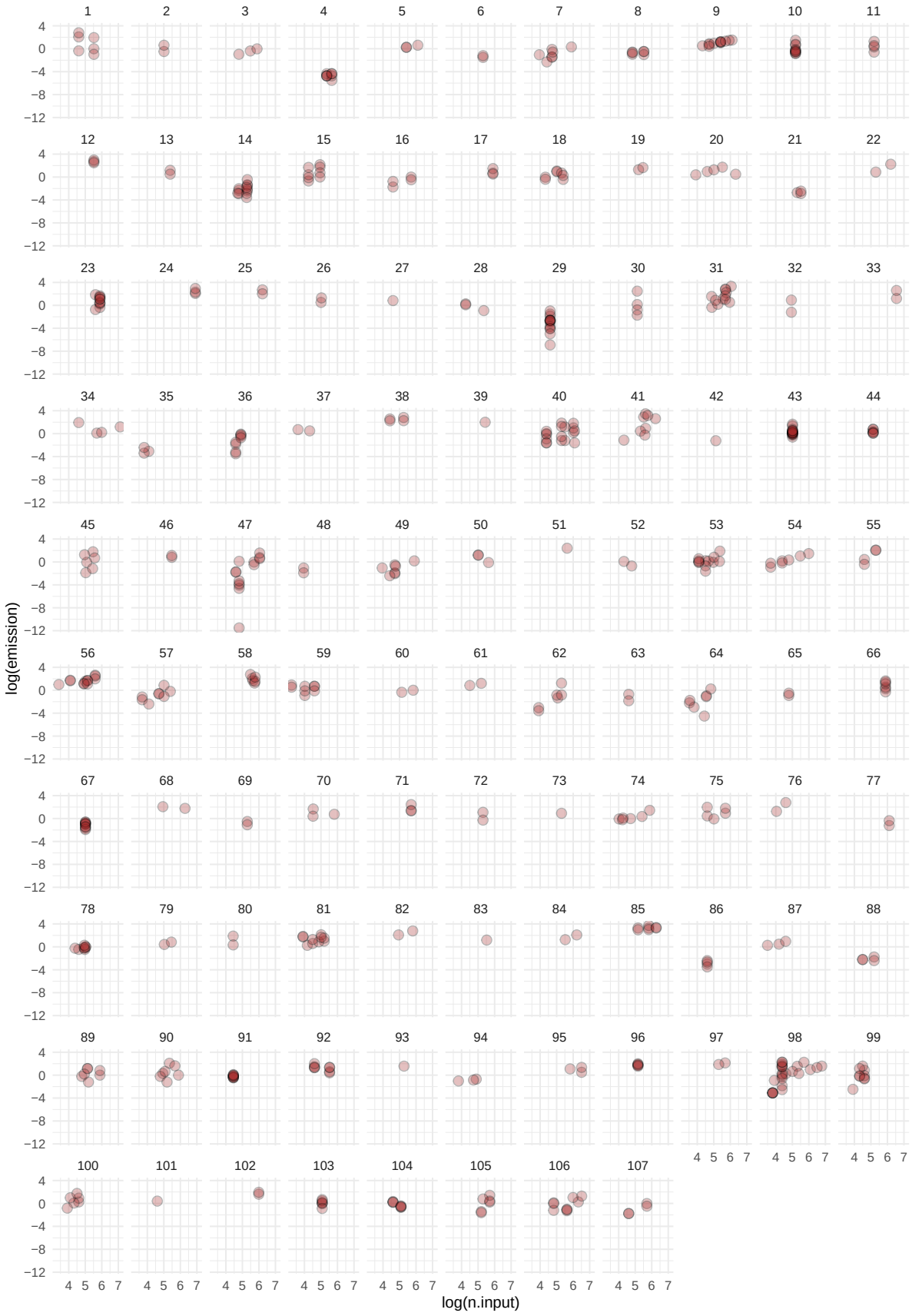
```
##   fertilizer group carbon n.input emission reps group.index fert.index
## 1          A    14    2.7     180    0.620   13          10          2
## 2          A    14    4.6     180    0.450   13          10          2
## 3          A    11    0.9     112    0.230   12           7          2
## 4          A    38    0.5     100    0.153   14          29          2
## 5          A     1    4.0     250    1.000    6           1          2
## 6          A    38    0.5     100    0.216   14          29          2
```

```
# data structure
BayesNSF::N2OEmission %>%
  dplyr::glimpse()
```

```
## Rows: 563
## Columns: 8
## $ fertilizer  <fct> A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A~
## $ group       <int> 14, 14, 11, 38, 1, 38, 11, 11, 11, 11, 13, 13, 13, 13, 1, ~
## $ carbon      <dbl> 2.70, 4.60, 0.90, 0.50, 4.00, 0.50, 0.90, 0.90, 0.90, 0.90~
## $ n.input     <dbl> 180, 180, 112, 100, 250, 100, 112, 112, 117, 82, 112, 112,~
## $ emission    <dbl> 0.620, 0.450, 0.230, 0.153, 1.000, 0.216, 0.240, 0.890, 0.~
## $ reps        <int> 13, 13, 12, 14, 6, 14, 12, 12, 12, 12, 14, 14, 14, 14, 6, ~
## $ group.index <int> 10, 10, 7, 29, 1, 29, 7, 7, 7, 7, 9, 9, 9, 9, 1, 9, 1, 7, ~
## $ fert.index  <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
```

Use the code below to plot logged N_2O emissions against logged fertilizer input. This is the same ggplot code as before except now we amend it to make plots for individual sites simply by adding the `facet_wrap` function and specifying the grouping variable(here it is `group.index`) as an argument.

```
g2 + facet_wrap(~group.index)
```


log(emission)
log(n.input)
1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63 64 65 66
67 68 69 70 71 72 73 74 75 76 77
78 79 80 81 82 83 84 85 86 87 88
89 90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 107
4 0 −4 −8 −12
4 5 6 7

Fitting the no-pool model with JAGS

You will now write a simple, no-pool model using the R template provided below. In addition to fitting this model, we would like you to have JAGS predict the mean logged N_2O emissions **for each site** as a function of soil fertilizer input. To help you out we have provided the range of N_2O values to predict over as the third element in the `data` list. **Note that you must use the index trick covered in lecture to align observations in each site with the appropriate intercept**. Here are the preliminaries to set up the model:

```
n.sites <- length(unique(BayesNSF::N2OEmission$group.index))
n.input.pred <- seq(min(BayesNSF::N2OEmission$n.input), max(BayesNSF::N2OEmission$n.input), 10)

data = list(
  log.emission = log(BayesNSF::N2OEmission$emission) %>% as.double()
  , log.n.input.centered = log(BayesNSF::N2OEmission$n.input) -
      mean(log(BayesNSF::N2OEmission$n.input)) %>%
        as.double()
  , log.n.input.centered.pred = log(n.input.pred) -
      mean(log(BayesNSF::N2OEmission$n.input)) %>%
        as.double()
  , group = BayesNSF::N2OEmission$group.index %>% as.double()
  , n.sites = n.sites
)

inits = list(
  list(alpha = rep(0, n.sites), beta = .5, sigma = 50)
  , list(alpha = rep(1, n.sites), beta = 1.5, sigma = 10)
  , list(alpha = rep(-1, n.sites), beta = .75, sigma = 20)
)
```

Question 5

Write the code for the model. Compile the model and execute the MCMC to produce a coda object. Produce trace plots of the chains for model parameters, excluding $\boldsymbol{\alpha}$ and a summary table of these same parameters. Assess convergence and look at the effective sample sizes for each of these parameters. Do you think any of the chains need to be run for longer and if so why? Make a horizontal caterpillar plot for the the $\boldsymbol{\alpha}$.

```
## JAGS Model
model{
  # priors
  # allow the intercept alpha to vary across sites
    for(j in 1:n.sites){
      alpha[j] ~ dnorm(0,1E-6)
    }
  # the slope beta is constant across sites
  beta ~ dnorm(0,1E-6)
  sigma ~ dunif(0,100)
  tau <- 1/sigma^2

  # likelihood
```

```
  for(i in 1:length(log.emission)) {
    log_mu[i] <- alpha[group[i]] + beta * log.n.input.centered[i]
    log.emission[i] ~ dnorm(log_mu[i], tau)
  }

  ## quantities of interest
    # predicted emissions
    ## from the JAGS primer:
      # If you have two product symbols in the conditional distribution with different indices
        # ...and two subscripts in the quantity of interest i.e. quantity[i, j]
        # ...then this dual product is specified in JAGS using nested for loops:
    for(i in 1:length(log.n.input.centered.pred)) {
      for(j in 1:n.sites){
        log_mu_site_pred[i, j] <- alpha[j] + beta * log.n.input.centered.pred[i]
      } # end j
    } # end i
}
```

JAGS Model

```
#####################################################################
# insert JAGS model code into an R script
#####################################################################
{ # Extra bracket needed only for R markdown files - see answers
  sink("NO2JAGS_nopooled.R") # This is the file name for the jags code
  cat("
  model{
    # priors
    # allow the intercept alpha to vary across sites
      for(j in 1:n.sites){
        alpha[j] ~ dnorm(0,1E-6)
      }
    # the slope beta is constant across sites
    beta ~ dnorm(0,1E-6)
    sigma ~ dunif(0,100)
    tau <- 1/sigma^2

    # likelihood
    for(i in 1:length(log.emission)) {
      log_mu[i] <- alpha[group[i]] + beta * log.n.input.centered[i]
      log.emission[i] ~ dnorm(log_mu[i], tau)
    }

    ## quantities of interest
      # predicted emissions
      ## from the JAGS primer:
        # If you have two product symbols in the conditional distribution with different indices
          # ...and two subscripts in the quantity of interest i.e. quantity[i, j]
          # ...then this dual product is specified in JAGS using nested for loops:
      for(i in 1:length(log.n.input.centered.pred)) {
        for(j in 1:n.sites){
```

```
          log_mu_site_pred[i, j] <- alpha[j] + beta * log.n.input.centered.pred[i]
        } # end j
      } # end i
  }
  ", fill = TRUE)
  sink()
}
#######################################################################
# implement model
#######################################################################
# 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 = "NO2JAGS_nopooled.R"
  , data = data
  , inits = inits
  , n.chains = length(inits)
  , n.adapt = n.adapt
)
```

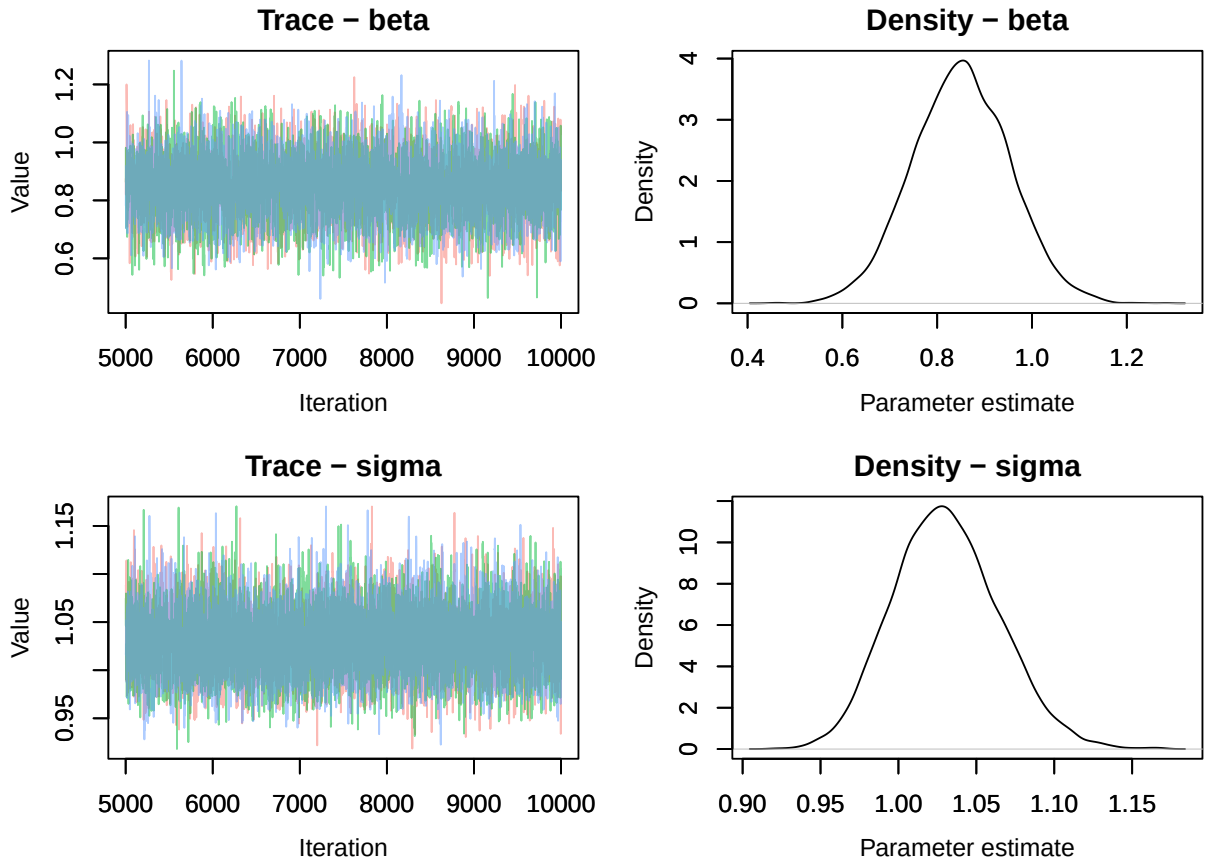
Implement JAGS Model

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 563
##    Unobserved stochastic nodes: 109
##    Total graph size: 15372
##
## Initializing model
```

```
stats::update(jm, n.iter = n.update)
# save the coda object (more precisely, an mcmc.list object) to R as "zc"
zc_nopooled = rjags::coda.samples(
  model = jm
  , variable.names = c("alpha", "beta", "sigma", "tau", "log_mu_site_pred")
  # , variable.names = c("a", "b", "p")
  , n.iter = n.iter
  , n.thin = 1
)
```

Model Output Produce trace plots of the chains for model parameters, excluding $\boldsymbol{\alpha}$ and a summary table of these same parameters. Assess convergence and look at the effective sample sizes for each of these parameters. Do you think any of the chains need to be run for longer and if so why?

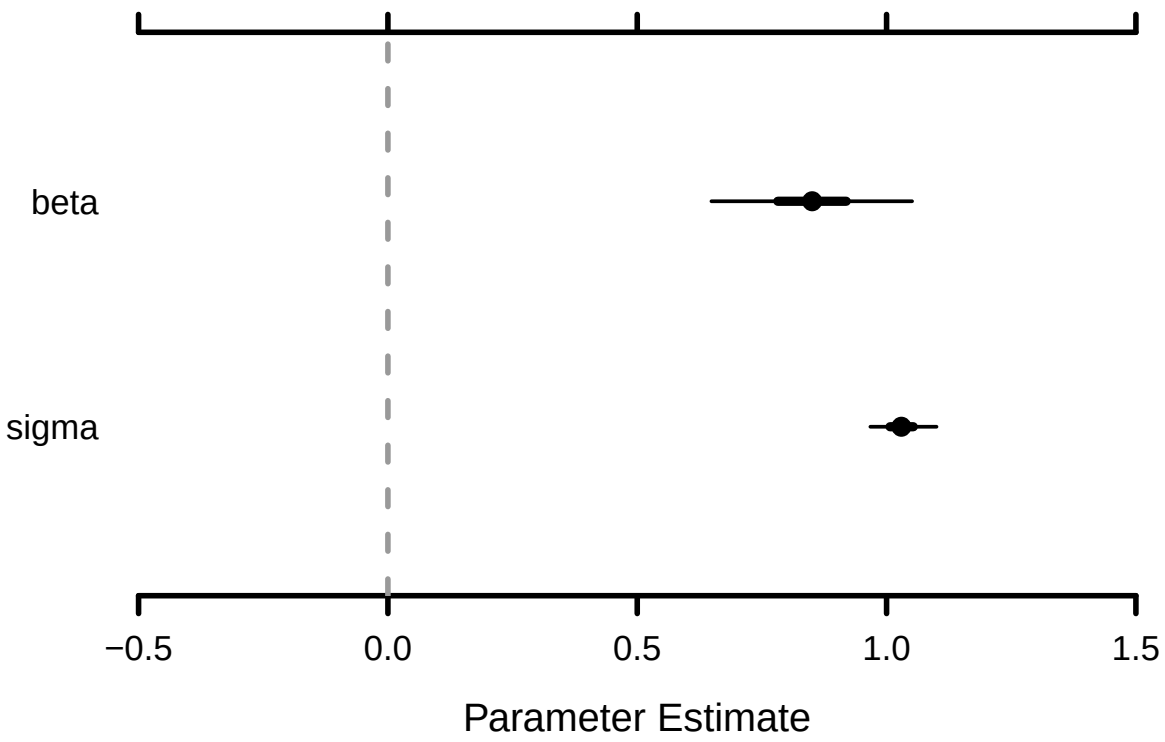
```
#####################
# check output
#####################
# trace plot
MCMCvis::MCMCtrace(zc_nopooled, params = c("beta", "sigma"), pdf = FALSE)
```




```
# summary
MCMCvis::MCMCsummary(zc_nopooled, params = c("alpha", "beta", "sigma")) %>%
  data.frame() %>%
  dplyr::slice_tail(n = 6)
```

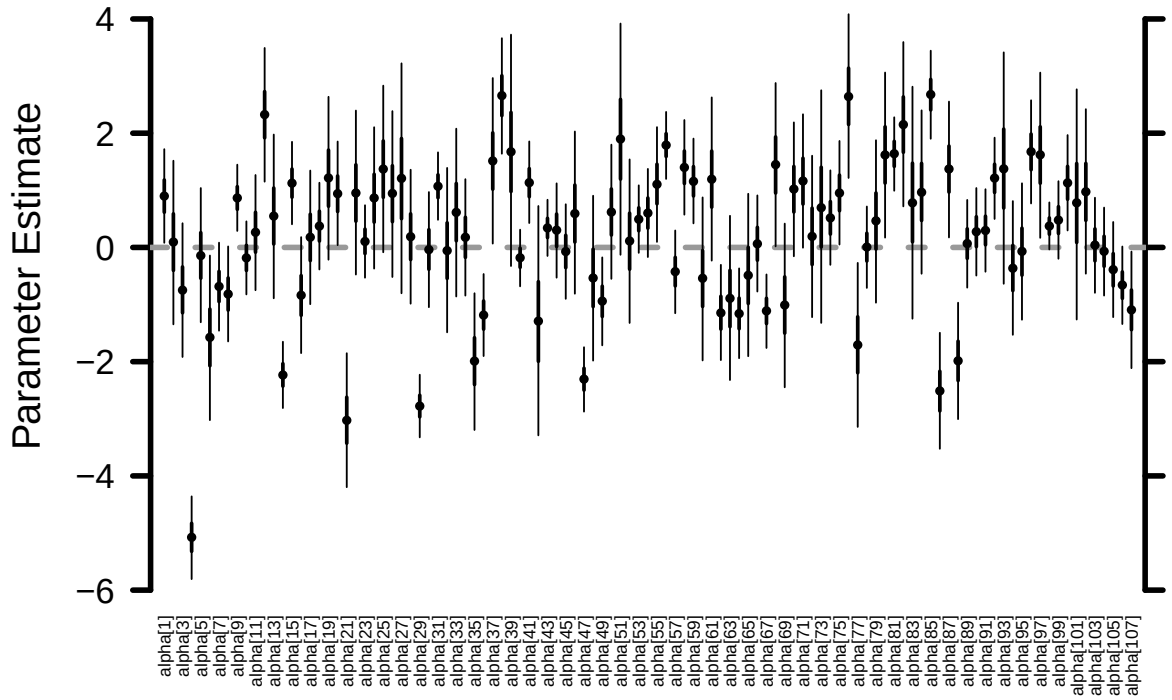
```
##                   mean         sd      X2.5.       X50.      X97.5. Rhat n.eff
## alpha[104] -0.06797236 0.39141160 -0.8427865 -0.0685102  0.69704330    1 30234
## alpha[105] -0.38886282 0.42393480 -1.2204560 -0.3889302  0.44525504    1 29544
## alpha[106] -0.65969429 0.34576157 -1.3411195 -0.6580501  0.01567793    1 28068
## alpha[107] -1.09360187 0.51804488 -2.1137363 -1.0934902 -0.07299432    1 30000
## beta        0.85024669 0.10202474  0.6487365  0.8507817  1.05119934    1  8405
## sigma       1.03086570 0.03421321  0.9674888  1.0297186  1.10022916    1 14147
```

```
# Caterpillar plots
MCMCvis::MCMCplot(zc_nopooled, params = c("beta", "sigma"), xlim = c(-0.5,1.5) )
```



Make a horizontal caterpillar plot for the the $\boldsymbol{\alpha}$.

```
# Caterpillar plots
MCMCvis::MCMCplot(
  zc_nopooled
  , params = c("alpha")
  , horiz = FALSE
  , ylim = c(-6,5)
  # Number specifying size of text for parameter labels on axis.
  , sz_labels = 0.6
  # Number specifying size of points represents posterior medians.
  , sz_med = 0.7
  # Number specifying thickness of 50 percent CI line (thicker line).
  , sz_thick = 2
  # Number specifying thickness of 95 percent CI line (thinner line).
  , sz_thin = 1
)
```



Question 6

How is the model able to estimate intercepts for sites where there is only a single x value, or even sites where there is only a single observation at all?

The model is able to estimate intercepts for sites where there is only a single data record because, in this model, the slope (β) is calculated using data from all sites. That is, the slope is assumed to be constant across sites. If we also allowed the slope (β) to vary across sites, then the model would require more than one data point to estimate site-specifice intercept and slope.

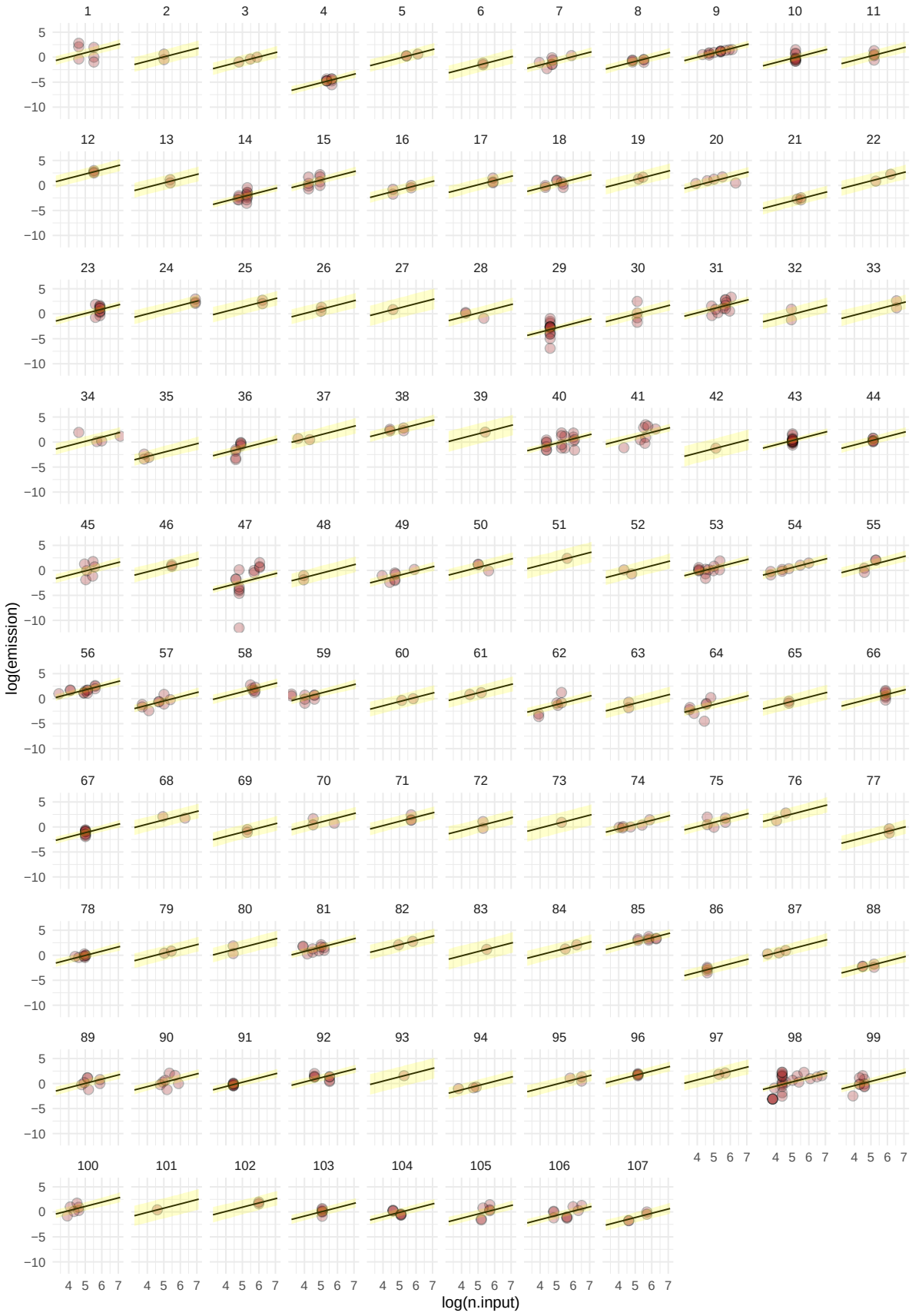
Visualizing the no-pool model predictions

We modify the `MCMCpstr` code from the previous model to produce a data frame of the median and 95% HDPI credible intervals of N_2O emission predictions for each site. `MCMCpstr` preserves the shape of the parameter from your JAGS model, which can be very handy in certain situations. Here, `pred1` is a list whose first element is a 3D-array. This array's rows are fertilizer inputs, columns are sites, and z-values are the quantities produced by the `hdi` function, which in this case is the lower and upper credible interval. You can `str` the `pred1[[1]]` object to see this for yourself. For plotting purposes though, we would like a data frame with columns for site, fertilizer input, the posterior's median emission, and the posterior's lower and upper HDPI credible intervals. This can be made easily using the `melt` function to go from wide to long followed by the `spread` function to make separate columns for the lower and upper bounds. Then we rely on `select` and `arrange` to order the data properly and keep the relevant columns. Lastly, we use `cbind` to make the data frame we seek, taking advantage of the fact that `n.input.pred` will repeat each site, which is exactly what we want it to do.

```
# HDI
pred1 <- MCMCvis::MCMCpstr(
  zc_nopooled
  , params = "log_mu_site_pred"
  , func = function(x) HDInterval::hdi(x, .95)
)
# median
pred2 <- MCMCvis::MCMCpstr(
  zc_nopooled
  , params = "log_mu_site_pred"
  , func = median
)
# create data frame
pred1.df <- melt(pred1[[1]], as.is = TRUE, varnames = c("x", "group.index", "metric")) %>%
  spread(metric, value) %>%
  arrange(group.index, x) %>%
  dplyr::select(group.index, lower, upper)
pred2.df <- melt(pred2[[1]], as.is = TRUE, varnames = c("x", "group.index"), value.name = "median") %>%
  arrange(group.index, x) %>%
  dplyr::select(median)
lpred.snp.df <- cbind(log.n.input.pred = log(n.input.pred), pred1.df, pred2.df)
```

To add the predictions to the plots for each site we use `geom_line` and `geom_ribbon` again, in combination with `facet_wrap`.

```
g2 +
  geom_line(
    data = lpred.snp.df
    , mapping = aes(x = log.n.input.pred, y = median)
  ) +
  geom_ribbon(
    data = lpred.snp.df
    , mapping = aes(x = log.n.input.pred, ymin = lower, ymax = upper)
    , alpha = 0.2
    , fill = "yellow"
  ) +
  facet_wrap(~group.index)
```


log(emission)
log(n.input)

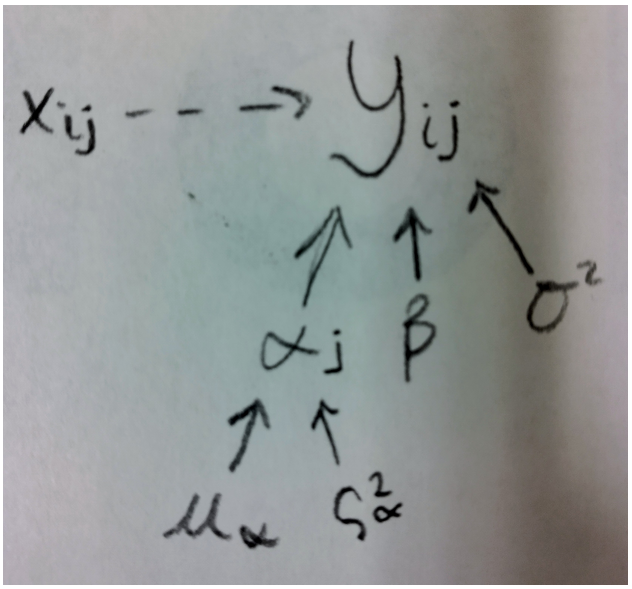
Random Intercepts

Diagramming and writing the random intercepts model

So far you have either ignored site completely (the pooled model) or treated all the site intercepts as independent from one another (the no-pool model). Now you are going to treat the site intercepts as partially pooled, meaning you will model them as coming from a common distribution. In other words, you will treat these intercepts in your model as a group level effect (aka, random effect). Hence, this model is often called a random-intercepts model. Like in the no-pool model, the deterministic portion of this model remains a linearized power function, but two subscripts are required: i which indexes the measurement within sites and j which indexes site itself. However, unlike the no-pool model, assume that these intercepts are drawn from a distribution with mean μ_α and variance ς_α^2.

Question 1

Draw a Bayesian network for a linear regression model of N_2O emission (y_{ij}) on fertilizer addition (x_{ij}).



DAG

Question 2

Write out the posterior and joint distribution for a linear regression model of N_2O emission (y_{ij}) on fertilizer addition (x_{ij}). Start by using generic `[ ]`. Use σ^2 and, ς^2 to represent the uncertainty in your model realizing that you might need moment matching when you choose a specific distribution.

$$g\big(\alpha_j, \beta, \log(x_{ij})\big) = \alpha_j + \beta\big(\log(x_i j)\big)$$

Joint:

$$\big[\alpha_j, \beta, \mu_\alpha, \sigma^2, \varsigma_\alpha^2 \mid \boldsymbol{y}\big] \propto \prod_{i=1}^{n}\prod_{j=1}^{J}\big[\log(y_{ij}) \mid g\big(\alpha_j, \beta, \log(x_{ij})\big), \sigma^2\big]\big[\alpha_j \mid \mu_\alpha, \varsigma_\alpha^2\big]\big[\beta\big]\big[\sigma\big]\big[\varsigma\big]$$

Question 3

Finish by choosing specific distributions for likelihoods and priors. You will use the math in the answer as a template to code your model in the subsequent exercises.

$$\begin{aligned}\left[\alpha_j, \beta, \mu_\alpha, \sigma^2, \varsigma_\alpha^2 \mid \boldsymbol{y}\right] \propto \prod_{i=1}^{n}\prod_{j=1}^{J} \mathsf{normal}\big(\log(y_{ij}) \mid g\big(\alpha_j, \beta, \log(x_{ij})\big), \sigma^2\big) \\ \times \mathsf{normal}\big(\alpha_j \mid \mu_\alpha, \varsigma_\alpha^2\big) \\ \times \mathsf{normal}\big(\beta \mid 0, 10000\big) \\ \times \mathsf{uniform}\big(\sigma \mid 0, 100\big) \\ \times \mathsf{uniform}\big(\varsigma \mid 0, 100\big)\end{aligned}$$

Fitting the random intercepts model with JAGS

Now you will implement the random-intercepts model that allows the intercept α_j to vary by site, where each intercept is drawn from a common distribution. Use the `data` and initial values for JAGS provided below to allow you to concentrate on writing JAGS code for the model.

In addition to fitting this model, we would like you to have JAGS predict the mean logged N_2O emissions **for each site** as a function of soil fertilizer input, just like you did in the no-pool model. We also would like you to predict the mean logged N_2O emissions and the median unlogged N_2O emissions as a function of soil fertilizer input, just like you did in the pooled model. However, these predictions should take into account **the uncertainty associated with site**. This is equivalent to asking you to make a prediction for a new site whose intercept α_j is drawn from the same distribution as the intercepts are for the actual sites themselves. To help you out we have provided the range of N_2O values to predict over as the third element in the `data` list.

```
n.input.pred <- seq(min(N2OEmission$n.input), max(N2OEmission$n.input), 10)
n.sites <- length(unique(N2OEmission$group.index))

data = list(
  log.emission = log(N2OEmission$emission),
  log.n.input.centered = log(N2OEmission$n.input) - mean(log(N2OEmission$n.input)),
  log.n.input.centered.pred = log(n.input.pred) - mean(log(N2OEmission$n.input)),
  group = N2OEmission$group.index,
  n.sites = n.sites)

inits = list(
  list(alpha = rep(0, n.sites), beta = .5, sigma = 50, mu.alpha= 0, sigma.alpha = 10),
  list(alpha = rep(1, n.sites), beta = 1.5, sigma = 10, mu.alpha= 2, sigma.alpha = 20),
  list(alpha = rep(-1, n.sites), beta = .75, sigma = 20, mu.alpha= -1, sigma.alpha = 12))
```

Question 5

Write the code for the model. Compile the model and execute the MCMC to produce a coda object. Produce trace plots of the chains for model parameters, excluding $\boldsymbol{\alpha}$ and a summary table of these same parameters. Assess convergence and look at the effective sample sizes for each of these parameters. Do you think any of the chains need to be run for longer and if so why? Make a horizontal caterpillar plot for the the $\boldsymbol{\alpha}$.

```
## JAGS Model
model{
  # priors
  beta ~ dnorm(0,1E-6)
  sigma ~ dunif(0,100)
  tau_y <- 1/sigma^2
  # alpha priors
  mu_alpha ~ dnorm(0,1E-6)
  varsigma ~ dunif(0,100)
  tau_alpha <- 1/varsigma^2
  # allow the intercept alpha to vary across sites
    for(j in 1:n.sites){
      alpha[j] ~ dnorm(mu_alpha, tau_alpha)
    }

  # likelihood
  for(i in 1:length(log.emission)) {
    log_mu[i] <- alpha[group[i]] + beta * log.n.input.centered[i]
    log.emission[i] ~ dnorm(log_mu[i], tau_y)
  }

  ## quantities of interest
    # predicted emissions FOR EACH SITE
      ## from the JAGS primer:
        # If you have two product symbols in the conditional distribution with different indices
          # ...and two subscripts in the quantity of interest i.e. quantity[i, j]
          # ...then this dual product is specified in JAGS using nested for loops:
      for(i in 1:length(log.n.input.centered.pred)) {
        for(j in 1:n.sites){
          log_mu_site_pred[i, j] <- alpha[j] + beta * log.n.input.centered.pred[i]
        } # end j
      } # end i

    # predicted emissions ACROSS SITES
      alpha_pred ~ dnorm(mu_alpha, tau_alpha)
      for(i in 1:length(log.n.input.centered.pred)){
        log_mu_pred[i] <- alpha_pred + beta * log.n.input.centered.pred[i]
        mu_pred[i] <- exp(log_mu_pred[i])
      }
}
```

JAGS Model

```
#####################################################################
# insert JAGS model code into an R script
#####################################################################
{ # Extra bracket needed only for R markdown files - see answers
  sink("NO2JAGS_randomints.R") # This is the file name for the jags code
  cat("
  model{
    # priors
```

```
    beta ~ dnorm(0,1E-6)
    sigma ~ dunif(0,100)
    tau_y <- 1/sigma^2
    # alpha priors
    mu_alpha ~ dnorm(0,1E-6)
    varsigma ~ dunif(0,100)
    tau_alpha <- 1/varsigma^2
    # allow the intercept alpha to vary across sites
      for(j in 1:n.sites){
        alpha[j] ~ dnorm(mu_alpha, tau_alpha)
      }

    # likelihood
    for(i in 1:length(log.emission)) {
      log_mu[i] <- alpha[group[i]] + beta * log.n.input.centered[i]
      log.emission[i] ~ dnorm(log_mu[i], tau_y)
    }

    ## quantities of interest
      # predicted emissions FOR EACH SITE
        ## from the JAGS primer:
          # If you have two product symbols in the conditional distribution with different indices
            # ...and two subscripts in the quantity of interest i.e. quantity[i, j]
            # ...then this dual product is specified in JAGS using nested for loops:
        for(i in 1:length(log.n.input.centered.pred)) {
          for(j in 1:n.sites){
            log_mu_site_pred[i, j] <- alpha[j] + beta * log.n.input.centered.pred[i]
          } # end j
        } # end i

        # predicted emissions ACROSS SITES
          alpha_pred ~ dnorm(mu_alpha, tau_alpha)
          for(i in 1:length(log.n.input.centered.pred)){
            log_mu_pred[i] <- alpha_pred + beta * log.n.input.centered.pred[i]
            mu_pred[i] <- exp(log_mu_pred[i])
          }
  }
  ", fill = TRUE)
  sink()
}
#####################################################################
# implement model
#####################################################################
# 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 = "NO2JAGS_randomints.R"
  , data = data
  , inits = inits
  , n.chains = length(inits)
  , n.adapt = n.adapt
)
```

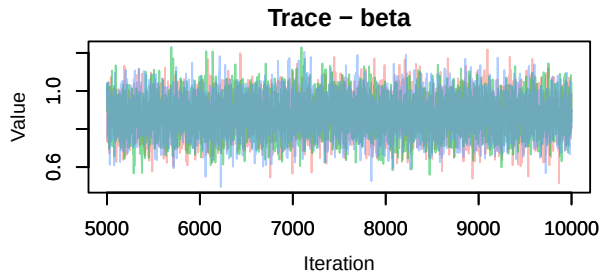
Implement JAGS Model

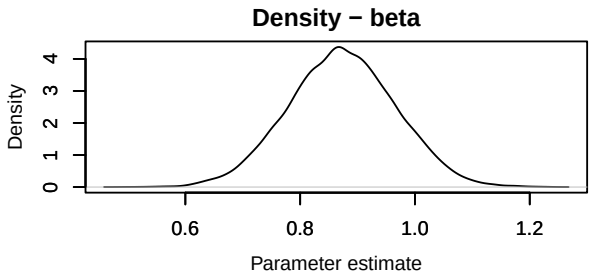
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 563
##    Unobserved stochastic nodes: 112
##    Total graph size: 15619
##
## Initializing model
```

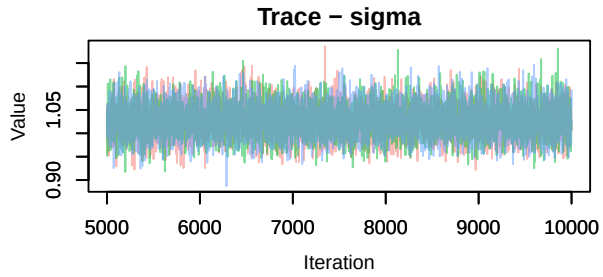
```
stats::update(jm, n.iter = n.update)
# save the coda object (more precisely, an mcmc.list object) to R as "zc"
zc_randomints = rjags::coda.samples(
  model = jm
  , variable.names = c("alpha", "beta", "sigma", "mu_alpha", "varsigma", "log_mu_site_pred", "log_mu_pr
  , n.iter = n.iter
  , n.thin = 1
)
```

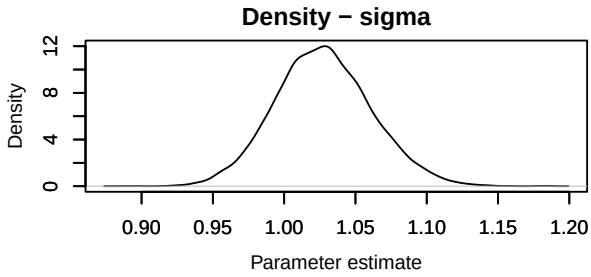
Model Output Produce trace plots of the chains for model parameters, excluding $\boldsymbol{\alpha}$ and a summary table of these same parameters. Assess convergence and look at the effective sample sizes for each of these parameters. Do you think any of the chains need to be run for longer and if so why? Make a horizontal caterpillar plot for the the $\boldsymbol{\alpha}$.

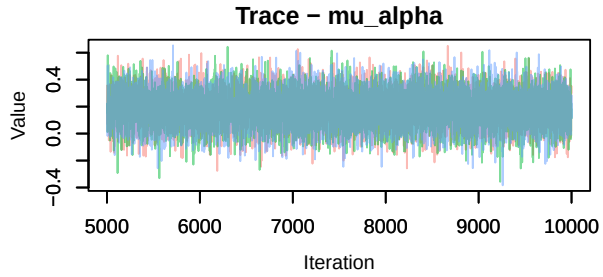
```
######################
# check output
######################
# trace plot
MCMCvis::MCMCtrace(zc_randomints, params = c("beta", "sigma", "mu_alpha", "varsigma"), pdf = FALSE)
```

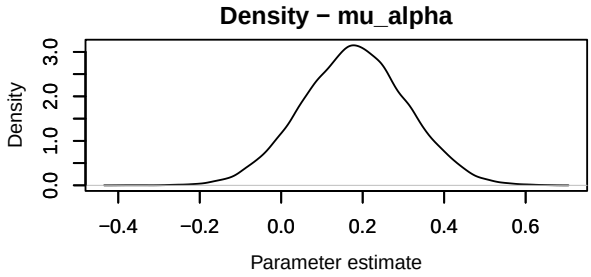

Trace − beta
Value
0.6
1.0
5000
6000
7000
8000
9000
10000
Iteration

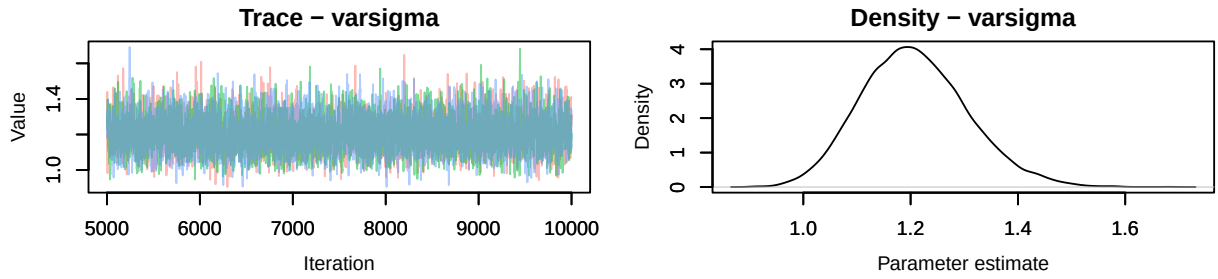

Density − beta
Density
0
1
2
3
4
0.6
0.8
1.0
1.2
Parameter estimate


Trace − sigma
Value
0.90
1.05
5000
6000
7000
8000
9000
10000
Iteration


Density − sigma
Density
0
4
8
12
0.90
0.95
1.00
1.05
1.10
1.15
1.20
Parameter estimate


Trace − mu_alpha
Value
−0.4
0.0
0.4
5000
6000
7000
8000
9000
10000
Iteration

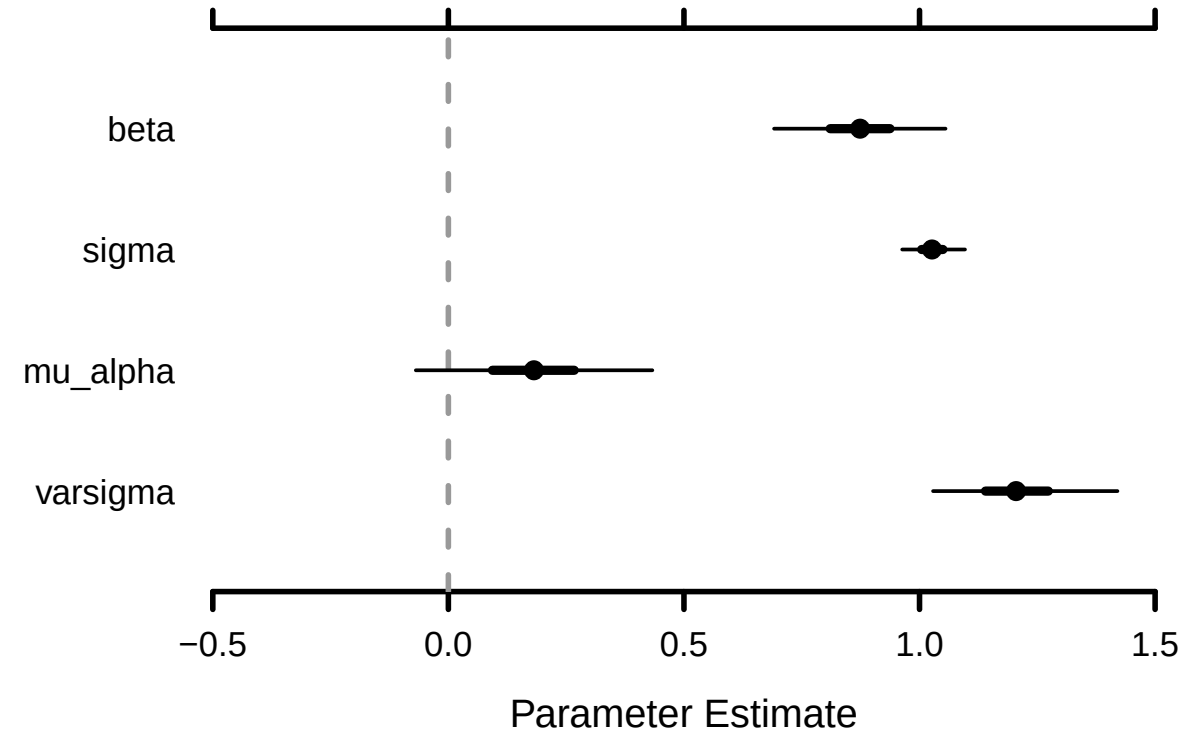

Density − mu_alpha
Density
0.0
1.0
2.0
3.0
−0.4
−0.2
0.0
0.2
0.4
0.6
Parameter estimate




```
# summary
MCMCvis::MCMCsummary(zc_randomints, params = c("alpha", "beta", "sigma")) %>%
  data.frame() %>%
  dplyr::slice_tail(n = 6)
```

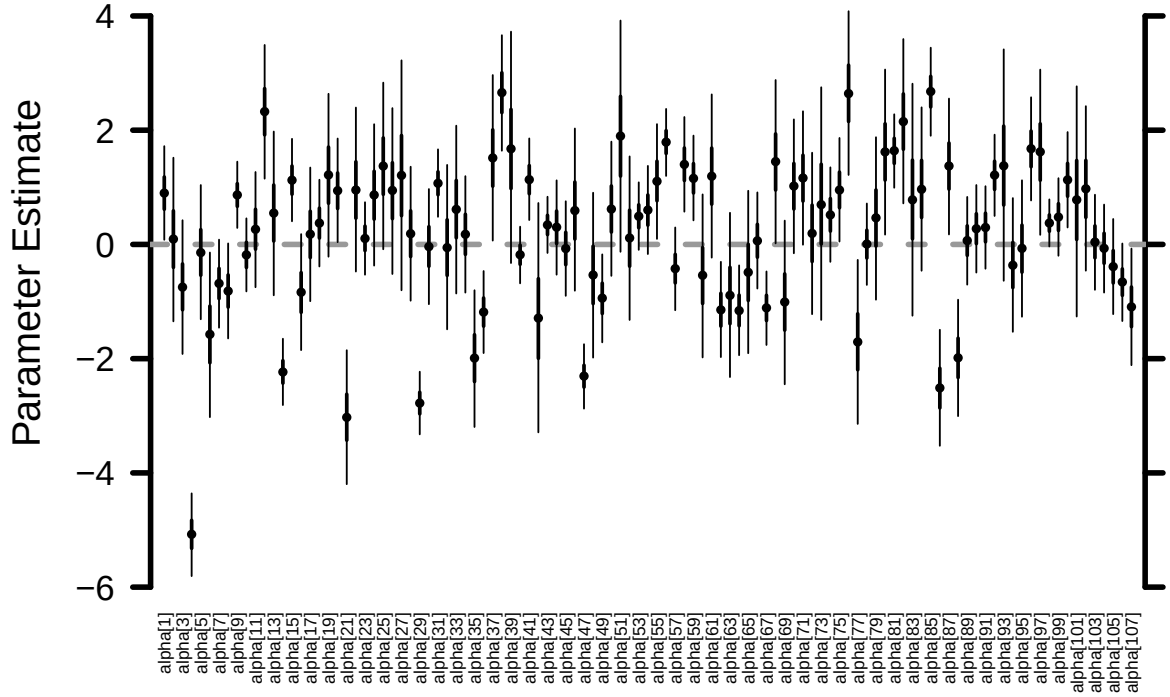
```
##                   mean         sd      X2.5.        X50.     X97.5. Rhat n.eff
## alpha[104] -0.04690785 0.37088318 -0.7686312 -0.04887137 0.68407711    1 29242
## alpha[105] -0.33431285 0.39816759 -1.1104522 -0.33698570 0.45197934    1 30497
## alpha[106] -0.60660977 0.33234687 -1.2531950 -0.60804313 0.03750739    1 28000
## alpha[107] -0.89346960 0.47789212 -1.8299594 -0.89179225 0.04828906    1 30000
## beta        0.87403388 0.09386744  0.6912836  0.87382708 1.05520764    1 10450
## sigma       1.02740983 0.03375885  0.9632187  1.02648151 1.09652271    1 14042
```

```
# Caterpillar plots
MCMCvis::MCMCplot(zc_randomints, params = c("beta", "sigma", "mu_alpha", "varsigma"), xlim = c(-0.5,1.5)
```



Make a horizontal caterpillar plot for the the $\boldsymbol{\alpha}$.

```
# Caterpillar plots
MCMCvis::MCMCplot(
  zc_nopooled
  , params = c("alpha")
  , horiz = FALSE
  , ylim = c(-6,5)
  # Number specifying size of text for parameter labels on axis.
  , sz_labels = 0.6
  # Number specifying size of points represents posterior medians.
  , sz_med = 0.7
  # Number specifying thickness of 50 percent CI line (thicker line).
  , sz_thick = 2
  # Number specifying thickness of 95 percent CI line (thinner line).
  , sz_thin = 1
)
```


Parameter Estimate
4
2
0
−2
−4
−6
alpha[1]
alpha[3]
alpha[5]
alpha[7]
alpha[9]
alpha[11]
alpha[13]
alpha[15]
alpha[17]
alpha[19]
alpha[21]
alpha[23]
alpha[25]
alpha[27]
alpha[29]
alpha[31]
alpha[33]
alpha[35]
alpha[37]
alpha[39]
alpha[41]
alpha[43]
alpha[45]
alpha[47]
alpha[49]
alpha[51]
alpha[53]
alpha[55]
alpha[57]
alpha[59]
alpha[61]
alpha[63]
alpha[65]
alpha[67]
alpha[69]
alpha[71]
alpha[73]
alpha[75]
alpha[77]
alpha[79]
alpha[81]
alpha[83]
alpha[85]
alpha[87]
alpha[89]
alpha[91]
alpha[93]
alpha[95]
alpha[97]
alpha[99]
alpha[101]
alpha[103]
alpha[105]
alpha[107]