Appendix S2: Example of informed priors using simulated data.

This appendix is a tutorial for using prior aggregation to include external sources of information in multi-species occupancy models (MSOMs). Running the code included in this tutorial requires the software JAGS, which can be downloaded here. All code in this tutorial can be found in the accompanying R script (Appendix S3).

Simulate the community

We will begin by simulating a community consisting of 10 species. We will assume that we surveyed this community by sampling 20 sites over a period of 4 surveys each. We will also simulate a covariate that is correlated with the occupancy rates of some species: half of the species in the community will respond negatively to the covariate, while the other half are not affected.

```r
# Global variables
nspec <- 10 # number of species
nsite <- 20 # number of sites
nsurvey <- 4 # surveys per site

Ks <- rep(nsurvey, nsite) # vector of surveys at each site

# Vector of covariate responses: half of species respond negatively
resp2cov <- c(rnorm(n = 5, sd = 0.25),
              rnorm(n = 5, mean = -3, sd = 0.25))

resp2cov <- sample(resp2cov)

# Covariate values for sites
cov <- sort(rnorm(n = nsite))
```

To simulate site-level occupancy, we will first draw species-level occupancy probabilities from a beta distribution $\psi_i \sim Beta(\alpha = 2, \beta = 3)$. This distribution generates a wide range of occupancy probabilities (95% interval 0.067586 - 0.8058796), a situation in which data augmentation is known to work well. We will use a logit link function to account for covariate effects on site-level occupancy probability of each species. Finally, the true occupancy state for each species at each site will be the result of a Bernoulli trial with the site-level probability as the probability of success.

```r
# Get probs from a beta distribution
sim.occ <- rbeta(n = nspec, shape1 = 2, shape2 = 3)

# Write function to simulate true occupancy state
tru.mats <- function(spec=nspec, site=nsite,
                     alpha1=resp2cov){

  #Get site-level psi to account for covariates
  alpha0 <- logit(sim.occ)

  #Create empty matrix to store occupancy probs
  logit.psi <- matrix(NA, nrow = spec, ncol = site)
  # Generate occupancy probs
  for(i in 1:spec){
    logit.psi[i,] <- alpha0[i] + alpha1[i]*cov
  }

  # Transform
  psi <- plogis(logit.psi)

  # Generate true occupancy state
  nlist<-list()
  for(a in 1:spec){
    nlist[[a]] <- rbinom(n = site, size = 1, prob = psi[a,])
  }

  #Turn abundance vectors into abundance matrix
  ns<-do.call(rbind, nlist)

  return(ns)
}
```

The true occupancy state for each species at each site is as follows:

Similarly to species-level occupancy probabilities, species-level detection probabilities will be drawn from a beta distribution $p_i \sim Beta(\alpha = 2, \beta = 8)$. This will generate low-to-mid detection probabilites (95% interval 0.028145-0.4824965), another situation in which data augmentation performs well. Environmental or survey covariates that may influence detectability can be added using the logit link function; however, for this example we will assume detectability does not vary across sites and surveys.

Simulated survey data will be the result of a Bernoulli trial with the species-level detection probability as the probability of encountering that species at a given site during a given survey.

```r
# Generate mean detection probabilities from beta dist
mean.p <- rbeta(n = nspec, shape1 = 2, shape2 = 8)
mean.p <- sort(mean.p, decreasing = T)

# Generate detection histories
get.obs <- function(mat, specs){
  #Detection intercept and cov responses
  beta0<-logit(mean.p) #put it on logit scale

  #Logit link function
  logit.p <- array(NA, dim = c(nsite, nsurvey, specs))
  for(i in 1:specs){
    for(j in 1:nsite){
      for(k in 1:nsurvey){
        logit.p[j,,i] <- beta0[i]
      }
    }
  }

  p <- plogis(logit.p)

  #Simulate observation data
  L<-list()
  for(b in 1:specs){
    y<-matrix(NA, ncol = nsite, nrow = nsurvey)
    for(a in 1:nsurvey){
      y[a,]<-rbinom(n = nsite, size = 1, prob = p[,,b]*mat[b,])
    }
    L[[b]]<-t(y)
  }

  #Smash it into array
  obs<-array(as.numeric(unlist(L)),
               dim=c(nsite, nsurvey, specs))

  return(obs)
}

obs.data <- get.obs(mat = tru, specs = nspec)

# Look at observed occurrence
maxobs <- apply(obs.data, c(1,3), max)
```

By calculating the column sums, we can see that one species went undetected in the simulated survey:

To make the JAGS script easier to write and the figures more readable, the undetected species was moved to the last position in the observed data. Code for this procedure can be found in Appendix S3.

Define the informed prior

Next, we will define the informed species-level prior distribution for the undetected species. We know the true value of the covariate is:

```
# Get true covariate value
resp2cov[10]
```

```
## [1] -2.745199
```

We will use this value as the mean of the informed prior distribution.

We will use the Markov chain Monte Carlo (MCMC) sampler JAGS to analyze the model. JAGS is compatible with most operating systems and the language is similar to R. The package R2jags will allow us to call JAGS directly from R.

In order for JAGS to analyze the model, we have to write a text file to send to JAGS. Begin by writing a character object that defines the mean and variance of the informed prior distribution:

```
# Write script for priors in JAGS language
priors <- "#Info for species-level prior distribution
            inf.mean <- -3 #mean of distribution
            inf.var <- 0.5 #variance of distribution"
```

Next, define the relative weights of the community-level hyperprior and the informed species-level prior. The weight is a value between 0 and 1 that determines the relative contribution of each prior to the aggregated prior (weights of each prior must sum to 1). To assign weights, create a vector with the weight of the community-level prior as the first element and the species-level prior as the second:

```
priors <- paste(priors,
          "#Define prior weights: how much each distribution
          #contributes to the final aggregate
```

```
           #Hyperprior first, then informed
             weights <- c(0.5, 0.5) #these are equal weights")
```

Next, pool the distributions. For normal distributions, the pooled mean is defined as:

$\mu_{pooled} = \sum(\mathbf{w}\mu) * v_{pooled}$

where μ is a vector of raw means and v_{pooled} the pooled variance. The pooled variance v_{pooled} is defined as:

$v_{pooled} = 1/\sum \mathbf{w}$

The term $\mathbf{w}$ is defined as $\mathbf{w} = \alpha/v$, where α is the vector of weights and $\mathbf{v}$ is a vector of raw variances.

Because $\mathbf{w}$ typically represents the regional occupancy of a species in MSOM notation, we will use the term 'lb' to calculate the pooled mean and variance. The terms 'a1.mean' and '1/tau.a1' are the mean and variance, respectively, of the community-level hyperprior, which we will define later.

```
priors <- paste(priors,
            "#Pool the distributions
            lb[1] <- weights[1]/(1/tau.a1)
            #1/tau.a0 is the variation of hyperprior
            lb[2] <- weights[2]/inf.var

            pooled.var <- 1/sum(lb)
            pooled.mean <- sum(lb*c(a1.mean,inf.mean))
              *pooled.var")
```

Finally, we will use the pooled mean and variance of the aggregated prior above when we define species-level priors:

```
priors <- paste(priors,
                "for(i in 1:spec){
                  #Create priors from hyperpriors/aggregated prior
                  w[i] ~ dbern(omega)
                  #w=1 means species was available for sampling

                  a0[i] ~ dnorm(a0.mean, tau.a0)
                  #a0 is the occupancy intercept

                  a1[i] ~ dnorm(ifelse(i==10,pooled.mean,a1.mean),
                            ifelse(i==10,(1/pooled.var),tau.a1))
                  #Use ifelse() here because detected species
                  #are still drawn from hyperprior

                  b0[i] ~ dnorm(b0.mean, tau.b0)
                  #b0 is detection intercept")
```

Write the JAGS script

Next, we write the full model script in the JAGS language:

```
# Function to create text file
write.model <- function(priors){
  mod <- paste("
    model{
    # Define hyperprior distributions: intercepts
    omega ~ dunif(0,1)
    mean.a0 ~ dunif(0,1)
    a0.mean <- log(mean.a0)-log(1-mean.a0)
    tau.a0 ~ dgamma(0.1, 0.1)

    mean.a1 ~ dunif(0,1)
    a1.mean <- log(mean.a0)-log(1-mean.a0)
    tau.a1 ~ dgamma(0.1, 0.1)

    mean.b0 ~ dunif(0,1)
    b0.mean <- log(mean.b0)-log(1-mean.b0)
    tau.b0 ~ dgamma(0.1, 0.1)

    ",priors,"

      #Estimate occupancy of species i at point j
      for (j in 1:J){
        logit(psi[j,i]) <- a0[i] + a1[i]*cov[j]
        Z[j,i] ~ dbern(psi[j,i]*w[i])

        #Estimate detection of i at point j during survey k
        for(k in 1:K[j]){
          logit(p[j,k,i]) <-  b0[i]
          obs[j,k,i] ~ dbern(p[j,k,i]*Z[j,i])
    }
    }
    }

    #Estimate total richness by adding observed and unobserved species
    n0<-sum(w[(spec+1):(spec+aug)])
    N<-spec+n0

    }
    ")
  writeLines(mod, "samplemod.txt")
}

write.model(priors = priors)
```

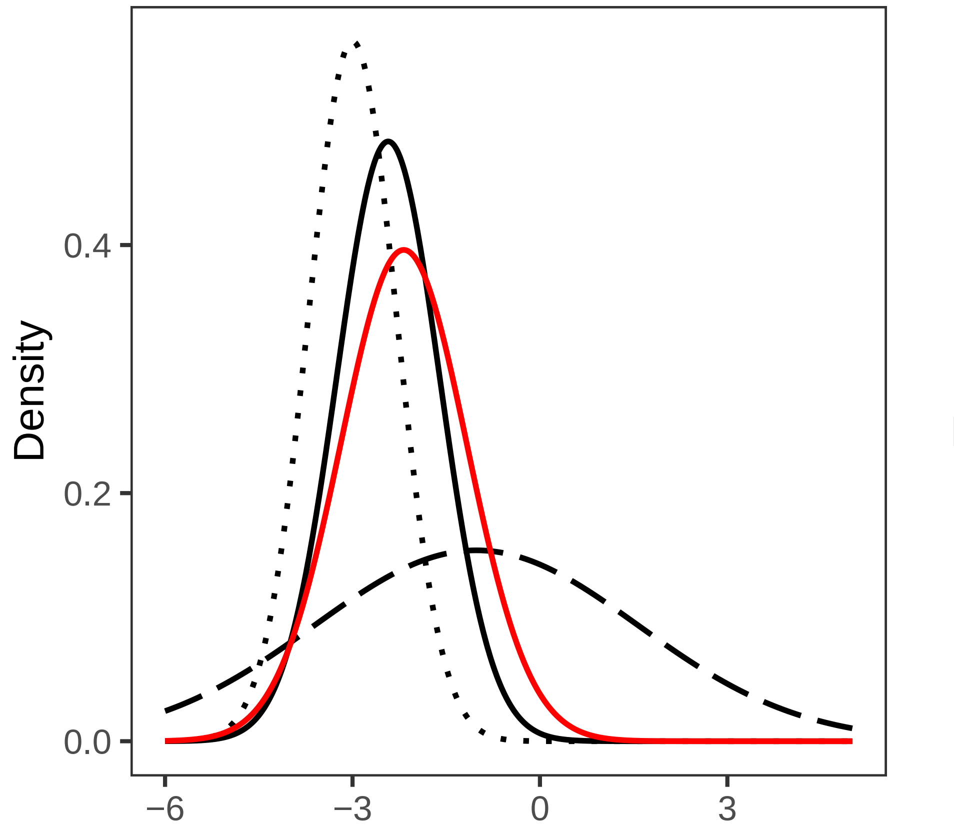
Run model

```
# List of data to send to model
datalist <- list(J = nsite, K = Ks, obs = obs.aug,
                 spec = 9, aug = 1, cov = cov)

# Parameters to save after model is analyzed
parms <- c('N', 'a0', 'b0', 'a1', 'Z', 'a1.mean','tau.a1', 'pooled.mean',
           'pooled.var')

# Initial values for the Markov chains
init.values<-function(){
  maxobs <- apply(obs.aug, c(1,3), max)
  inits <- list(
    w = rep(1,nspec),
```

```r
    a0 = rnorm(n = nspec),
    a1 = rnorm(n = nspec),
    b0 = rnorm(n = nspec),
    Z = maxobs)
}

# Send model to JAGS
# model <- jags(model.file = 'samplemod.txt', data = datalist,
#               n.chains = 3, parameters.to.save = parms,
#               inits = init.values, n.burnin = 1000,
#               n.iter = 5000, n.thin = 3)

# Save/load model
# saveRDS(model, file = "sample_mod.rds")
model <- readRDS(file = "sample_mod.rds")
```

Figures

```r
# Get values from aggregated prior
pooled.mean <- median(model$BUGSoutput$sims.list$pooled.mean)
pooled.sd <- median(sqrt(model$BUGSoutput$sims.list$pooled.var))
# Medians used because posterior is asymmetrical

# Create objects from informed values used in priors
inf.mean <- -3
inf.var <- 0.5

# Pull community distribution priors from model
comm.mean <- median(model$BUGSoutput$sims.list$a1.mean)
comm.sd <- median(sqrt(1/model$BUGSoutput$sims.list$tau.a1))
# These are symmetrical but using median for consistency

# Pull posteriors from model
post.mean <- mean(model$BUGSoutput$sims.list$a1[,10])
post.sd <- sd(model$BUGSoutput$sims.list$a1[,10])

# Plot the distributions
ggplot()+
  stat_function(fun = dnorm, n = 1000,
                args = list(mean = pooled.mean, sd = pooled.sd),
                size = 1, aes(linetype = "Aggregated", color = "Prior"))+
  stat_function(fun = dnorm, n = 1000,
                args = list(mean = inf.mean, sd = sqrt(inf.var)),
                size = 1, aes(linetype = "Informed", color = "Prior"))+
  stat_function(fun = dnorm, n = 1000,
                args = list(mean = comm.mean, sd = comm.sd),
                size = 1, aes(linetype = "Community", color = "Prior"))+
  stat_function(fun = dnorm, n = 1000,
                args = list(mean = post.mean, sd = post.sd),
                size = 1, aes(linetype = "Aggregated", color = "Posterior"))+
  xlim(c(-6, 5))+
  scale_linetype_manual(breaks = c("Aggregated", "Informed", "Community"),
                         values = c(1, 3, 5), name = "Prior")+
  scale_color_manual(breaks = c("Prior", "Posterior"),
                      values = c("black", "red"), name = "")+
  labs(y = "Density")+
  theme_bw(base_size = 16)+
  theme(panel.grid = element_blank(),
        axis.title.x = element_blank())
```



Check to see if aggregation worked

```
# Extract regional species richness N from model
Ns <- as.vector(model$BUGSoutput$sims.list$N)

# Create table of counts for each estimate
Ns %>%
  table() %>%
  data.frame() %>%
  {. ->> ns.frame}
colnames(ns.frame) <- c("N_Species", "Freq")

# Look at mean an median estimates
mean(Ns)
```

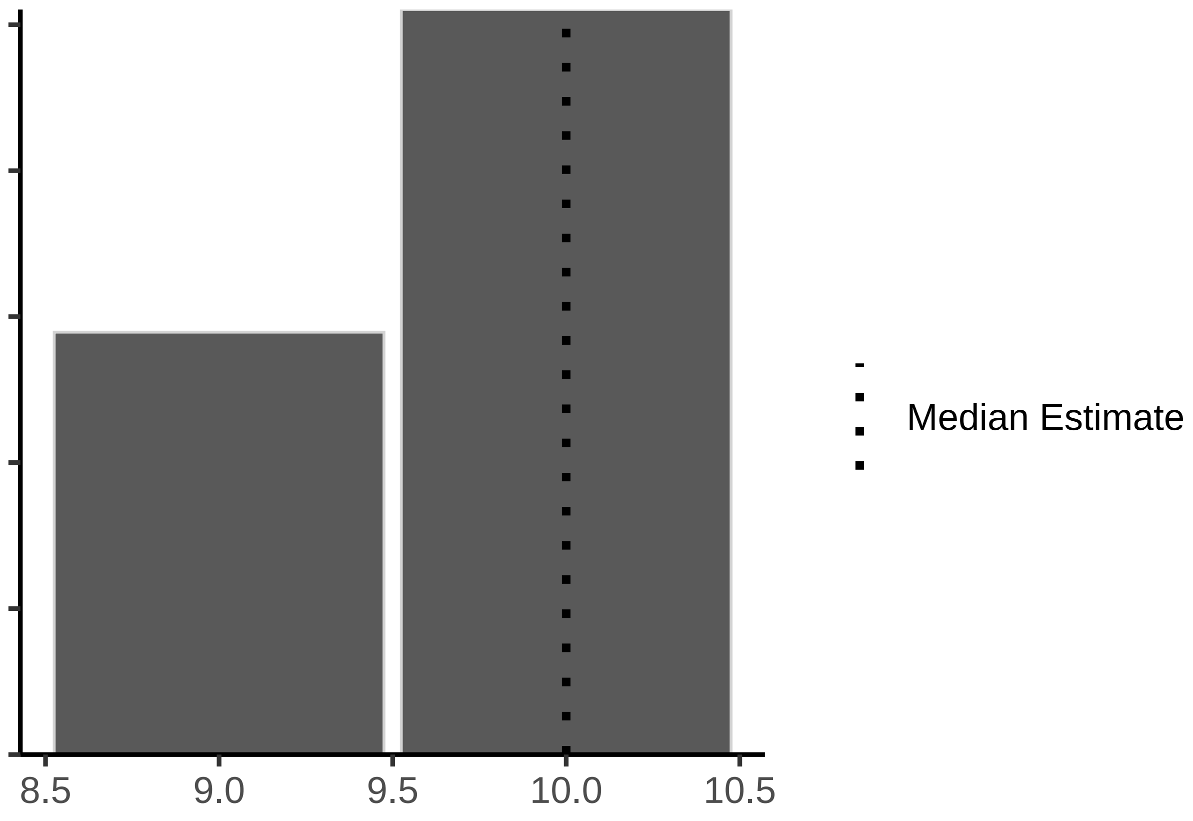
Regional richness estimates

```
## [1] 9.63816
```

```r
median(Ns)
```

```
## [1] 10
```

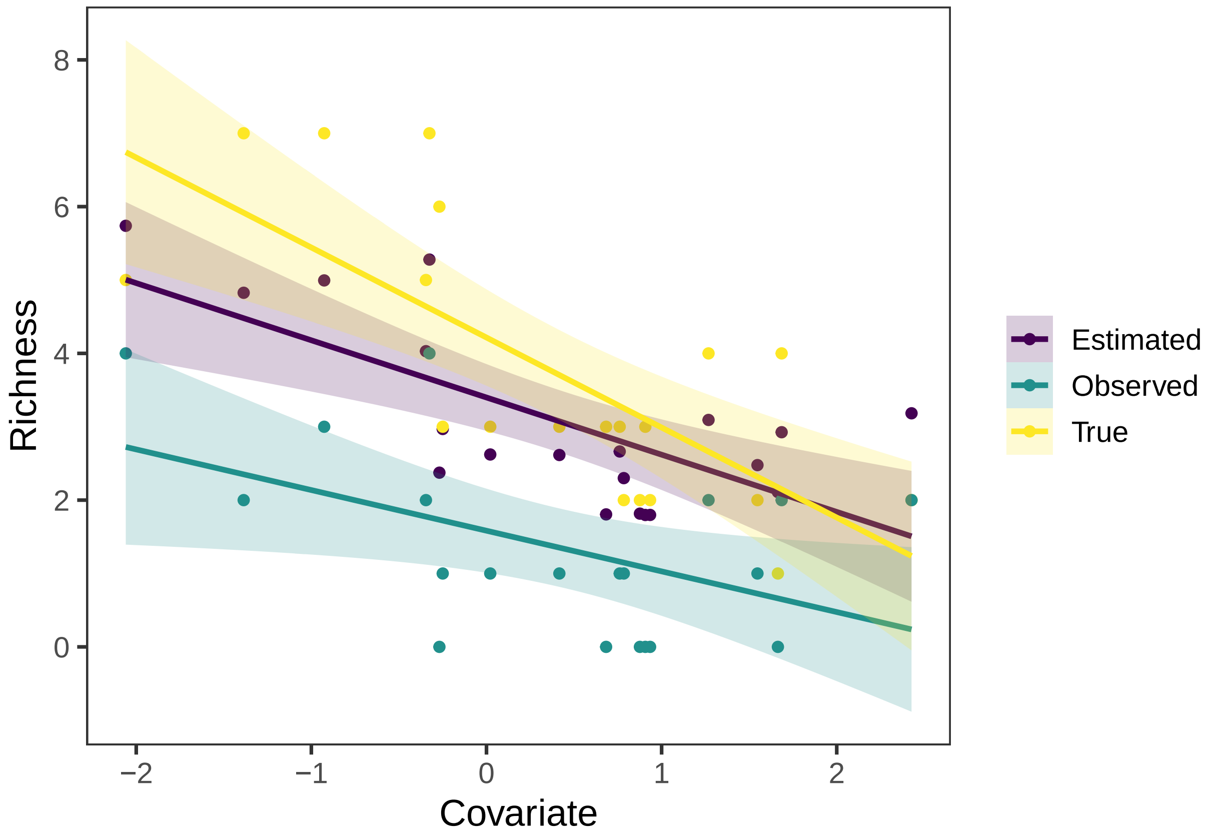
```r
# Suggests most models accounted for the missing species

# Check it graphically
Ns.median <- median(Ns)
ggplot(data = ns.frame, aes(x = as.integer(as.character(N_Species)),
                            y = Freq))+
  geom_col(width = 0.95, color = 'lightgray')+
  geom_vline(aes(xintercept = Ns.median, linetype = "Estimated"),
             size = 1.5)+
  scale_linetype_manual(values = c("Estimated"="dotted"),
                        name = "",
                        labels = c("Median Estimate"))+
  labs(x = "Estimated Species", y = "Frequency")+
  scale_y_continuous(expand = c(0,0))+
  theme_classic(base_size = 18)+
  theme(axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        axis.title.x = element_blank(),
        legend.key.height = unit(40, units = 'pt'),
        aspect.ratio = 1/1)
```



```r
# Pull estimated occurrences from model
Zs <- model$BUGSoutput$sims.list$Z

# Get avg. occurrence matrices
Zs.mean <- apply(Zs, c(2,3), mean)

# Get site-level richness
site.rich <- data.frame(Estimated = rowSums(Zs.mean))

# Get true and observed values for comparison
site.rich$True <- colSums(tru)
site.rich$Observed <- rowSums(apply(obs.data, c(1,3), max))
site.rich$Cov <- cov # Covariate makes for a cleaner plot

# Coerce to long format
rich.long <-pivot_longer(site.rich, cols = Estimated:Observed,
                         values_to = "Richness", names_to = "Source")

# Plot it
ggplot(data = rich.long, aes(x = Cov, y = Richness, color = Source))+
  geom_point()+
  geom_smooth(aes(fill = Source), method = 'lm', alpha = 0.2)+
  labs(x = "Covariate")+
  expand_limits(y = 0)+
  scale_color_viridis_d()+
  scale_fill_viridis_d()+
  theme_bw(base_size = 14)+
  theme(panel.grid = element_blank(), legend.title = element_blank())
```

Site-level richness estimates

```
## `geom_smooth()` using formula 'y ~ x'
```

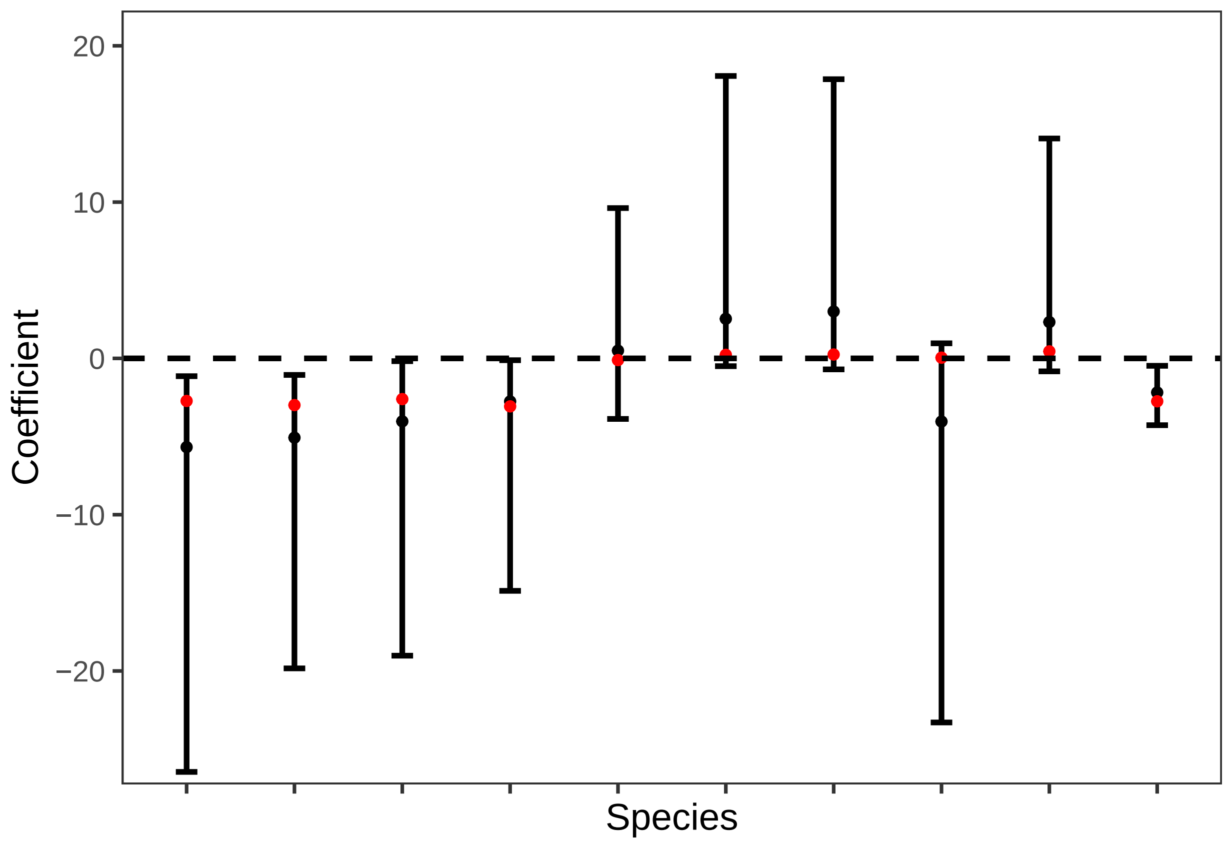


```r
# Extract covariate estimates from jags object
a1s <- model$BUGSoutput$sims.list$a1

a1s <- as.data.frame(a1s)

# Create a vector of species names
specnames <- logical()
for(i in 1:nspec){
  specnames[i] <- paste("Spec", i, sep = "")
}

colnames(a1s) <- specnames

# Pivot data frame for plotting
a1.long <- a1s %>%
  pivot_longer(cols = everything(), names_to = "Spec",
               values_to = "a1")

a1.long$Spec <- factor(a1.long$Spec, levels = specnames)

# Get summary stats
a1.stat <- a1.long %>%
  group_by(Spec) %>%
  summarise(mean = mean(a1), lo = quantile(a1, 0.025),
            hi = quantile(a1, 0.975)) %>%
  mutate(tru.resp = resp2cov)
```

Covariate responses

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
# Make interval plot
ggplot(data = a1.stat, aes(x = Spec, y = mean))+
  geom_point(size = 1.5)+
  geom_errorbar(ymin = a1.stat$lo, ymax = a1.stat$hi,
                size = 1, width = 0.2)+
  geom_point(aes(y = tru.resp), color = "red", size = 1.5)+
  geom_hline(yintercept = 0, linetype = "dashed", size = 1)+
  scale_y_continuous(limits = c(-25, 20))+
  labs(x = "Species", y = "Coefficient")+
  theme_bw(base_size = 14)+
  theme(panel.grid = element_blank(), axis.text.x = element_blank())
```


Coefficient
Species