```
## Fit an integrated population model (IPM) for Mourning Doves to data collected in Missouri

## Band data informs population age structure at banding, survival, recovery, vulnerability, and
harvest (Brownie model)
## Wing data informs productivity, with a correction for vulnerability
## Call count index is linked to population change

# Packages ------------------------------------------------------------------
library(tidyverse) ## data wrangling/piping
library(nimble) ## run models
library(coda) ## create mcmc.list object
library(parallel) ## run in parallel
library(MCMCvis) ## summarise output

# rm(list = ls())

# Data ----------------------------------------------------------------------

## Productivity info
wings_total <- readRDS(file = "../Data/IPM_input/merged_wing_total_PP+D_MAP_OP_DT.rds")
wings_hatchyear <- readRDS(file = "../Data/IPM_input/merged_wing_hy_PP+D_MAP_OP_DT.rds")

means_t <- round(colMeans(wings_total), 0)
means_j <- round(colMeans(wings_hatchyear), 0)


# Convert mean (0.446) and sd (0.034) for report rate to parameters for Beta-distribution (alpha
and beta)
mean_rr <- 0.446
sd_rr <- 0.034
alpha_rr <- (mean_rr^2) * (((1 - mean_rr) / (sd_rr^2)) - (1 / mean_rr))
beta_rr <- alpha_rr * ((1 / mean_rr) - 1)


## Dead m-array data
m_arr_ad_PP <- readRDS(file =
"../Data/IPM_input/m_array_dead_PrairiePeninsula+Dent_adult_2024.rds")
m_arr_jv_PP <- readRDS(file =
"../Data/IPM_input/m_array_dead_PrairiePeninsula+Dent_juvenile_2024.rds")

m_arr_ad_MAP <- readRDS(file = "../Data/IPM_input/m_array_dead_MAV_adult_2024.rds")
m_arr_jv_MAP <- readRDS(file = "../Data/IPM_input/m_array_dead_MAV_juvenile_2024.rds")

m_arr_ad_OP <- readRDS(file = "../Data/IPM_input/m_array_dead_OsagePlains_adult_2024.rds")
m_arr_jv_OP <- readRDS(file = "../Data/IPM_input/m_array_dead_OsagePlains_juvenile_2024.rds")

m_arr_ad_DT <- readRDS(file = "../Data/IPM_input/m_array_dead_DissectedTill_adult_2024.rds")
m_arr_jv_DT <- readRDS(file = "../Data/IPM_input/m_array_dead_DissectedTill_juvenile_2024.rds")


# Lump into age-specific m-arrays
library(abind)
m_arr_j <- abind(m_arr_jv_PP, m_arr_jv_MAP, m_arr_jv_OP, m_arr_jv_DT, along = 3)
m_arr_a <- abind(m_arr_ad_PP, m_arr_ad_MAP, m_arr_ad_OP, m_arr_ad_DT, along = 3)

# Number of released at each occasions (here rowsums)
rel_j <- as.matrix(cbind(rowSums(m_arr_j[, , 1]), rowSums(m_arr_j[, , 2]), rowSums(m_arr_j[, ,
3]), rowSums(m_arr_j[, , 4])))
rel_a <- as.matrix(cbind(rowSums(m_arr_a[, , 1]), rowSums(m_arr_a[, , 2]), rowSums(m_arr_a[, ,
3]), rowSums(m_arr_a[, , 4])))


## density
```

```
densities_PP <- readRDS(file = "../Data/IPM_input/densities_PrairiePeninsula+D.rds") %>%
as.data.frame()
densities_MAP <- readRDS(file = "../Data/IPM_input/densities_MAP.rds") %>% as.data.frame()
densities_OP <- readRDS(file = "../Data/IPM_input/densities_OsagePlains.rds") %>% as.data.frame()
densities_DT <- readRDS(file = "../Data/IPM_input/densities_DissectedTill.rds") %>%
as.data.frame()


## 1999 to 2024, 25 rows total
dens_mat <- densities_PP %>%
  mutate(
    dens_map = densities_MAP$density,
    dens_op = densities_OP$density,
    dens_dt = densities_DT$density
  ) %>%
  select(density, dens_map, dens_op, dens_dt) %>%
  as.matrix()


# Model ------------------------------------------------------------------

model_code <- nimbleCode({
  #-------------------------------------------------------------------------------------------------
----
  ## Productivity sub-model
  ## -------------------------------------------------------------------------------------------------
-----

  # Priors
  ratio_mean ~ dunif(0, 20)
  log_ratio_mean <- log(ratio_mean)

  ## priors for deviation from grand mean
  # site
  ratio_sigma_site ~ T(dt(0, pow(2.5, -2), 1), 0, )
  ratio_tau_site <- pow(ratio_sigma_site, -2)

  # year
  ratio_sigma_year ~ T(dt(0, pow(2.5, -2), 1), 0, )
  ratio_tau_year <- pow(ratio_sigma_year, -2)

  # region
  for (s in 1:4) {
    # Draw site-specific mean ratio from a common distribution (i.e. spatial random-effect for
ratio)
    log_ratio_site[s] ~ dnorm(log_ratio_mean, ratio_tau_site)
    ratio_site[s] <- exp(log_ratio_site[s])

    for (t in 1:(n.occasions)) {
      # Draw annual values for ratio at each site from a site-specific distribution (i.e. temporal
random-effect for ratio)
      log_ratio[t, s] ~ dnorm(log_ratio_site[s], ratio_tau_year)
      ratio[t, s] <- exp(log_ratio[t, s])

      # Predicted proportion of juveniles in harvest bag
      q_obs[t, s] <- ratio[t, s] * kill_j[t, s] / (ratio[t, s] * kill_j[t, s] + kill_a[t, s])

      # Likelihood for harvest bag data
      wing_j[t, s] ~ dbin(q_obs[t, s], wing_t[t, s])
    }
  }


  ## -------------------------------------------------------------------------------------------------
-------
```

```
  ##        Harvest sub-model
  ## ------------------------------------------------------------------------------------------
-------

  # Harvest priors (do not vary by site and year)
  rr ~ dbeta(alpha_rr, beta_rr) ## reporting rate
  cr ~ dunif(0.25, 0.29) ## crippling rate (i.e. 1 - recovery rate); strong prior from Schulz et
al 2013 https://doi.org/10.1002/wsb.274


  # Harvest priors (vary by site and year)
  kill_j_grand ~ dunif(0, 1) ## overall juvenile kill rate
  logit_kill_j_grand <- logit(kill_j_grand)

  kill_a_grand ~ dunif(0, 1) ## overall adult kill rate
  logit_kill_a_grand <- logit(kill_a_grand)

  ## site kill rate precision
  sigma_kill_j_site ~ T(dt(0, pow(2.5, -2), 1), 0, )
  sigma_kill_a_site ~ T(dt(0, pow(2.5, -2), 1), 0, )

  tau_kill_j_site <- pow(sigma_kill_j_site, -2)
  tau_kill_a_site <- pow(sigma_kill_a_site, -2)

  ## year kill rate precision
  sigma_kill_j_year ~ T(dt(0, pow(2.5, -2), 1), 0, )
  sigma_kill_a_year ~ T(dt(0, pow(2.5, -2), 1), 0, )

  tau_kill_j_year <- pow(sigma_kill_j_year, -2)
  tau_kill_a_year <- pow(sigma_kill_a_year, -2)


  for (s in 1:4) {
    logit_kill_j_site[s] ~ dnorm(logit_kill_j_grand, tau_kill_j_site)
    logit_kill_a_site[s] ~ dnorm(logit_kill_a_grand, tau_kill_a_site)

    kill_j_site[s] <- ilogit(logit_kill_j_site[s])
    kill_a_site[s] <- ilogit(logit_kill_a_site[s])

    for (t in 1:n.occasions) {
      logit_kill_j[t, s] ~ dnorm(logit_kill_j_site[s], tau_kill_j_year)
      logit_kill_a[t, s] ~ dnorm(logit_kill_a_site[s], tau_kill_a_year)

      kill_j[t, s] <- ilogit(logit_kill_j[t, s])
      kill_a[t, s] <- ilogit(logit_kill_a[t, s])

      # Recovery rate f (Brownie)
      f[1, t, s] <- kill_j[t, s] * (1 - cr) * rr # juvenile recovery rate
      f[2, t, s] <- kill_a[t, s] * (1 - cr) * rr # adult recovery rate
    }
  }


  ## ------------------------------------------------------------------------------------------
  ## Survival sub-model
  ## ------------------------------------------------------------------------------------------

  # Priors for survival probability
  sj_grand ~ dunif(0, 1) # overall juvie survival is between 0 and 1
  logit_sj_grand <- logit(sj_grand) ## logit transformation

  sa_grand ~ dunif(0, 1) # overall adult survival
  logit_sa_grand <- logit(sa_grand)

  # priors on tau for site level deviation from the grand mean
```

```
  sigma_sj ~ T(dt(0, pow(2.5, -2), 1), 0, )
  sigma_sa ~ T(dt(0, pow(2.5, -2), 1), 0, )

  tau_sj_site <- pow(sigma_sj, -2)
  tau_sa_site <- pow(sigma_sa, -2)

  # priors on tau for year level deviation from the grand mean
  sigma_sjy ~ T(dt(0, pow(2.5, -2), 1), 0, )
  sigma_say ~ T(dt(0, pow(2.5, -2), 1), 0, )

  tau_sj_year <- pow(sigma_sjy, -2)
  tau_sa_year <- pow(sigma_say, -2)


  for (s in 1:4) {
    logit_sj_site[s] ~ dnorm(logit_sj_grand, tau_sj_site) ## draws for juvie region specific
precision
    logit_sa_site[s] ~ dnorm(logit_sa_grand, tau_sa_site) ## draws for adult region specific
precision

    ## back transform from logit for interpretation
    sj_site[s] <- ilogit(logit_sj_site[s])
    sa_site[s] <- ilogit(logit_sa_site[s])

    for (t in 1:n.occasions) {
      logit_sj[s, t] ~ dnorm(logit_sj_site[s], tau_sj_year)
      logit_sa[s, t] ~ dnorm(logit_sa_site[s], tau_sa_year)

      sj[s, t] <- ilogit(logit_sj[s, t])
      sa[s, t] <- ilogit(logit_sa[s, t])
    }
  }

  # Loop over the regions
  for (s in 1:4) {
    # Likelihood
    for (t in 1:(n.occasions)) {
      m_arr_j[t, 1:(n.occasions + 1), s] ~ dmulti(pr.j[t, 1:(n.occasions + 1), s], rel_j[t, s])
      m_arr_a[t, 1:(n.occasions + 1), s] ~ dmulti(pr.a[t, 1:(n.occasions + 1), s], rel_a[t, s])
    }

    # Cell probabilities of the juvenile m-array

    # Main diagonal
    for (t in 1:n.occasions) {
      pr.j[t, t, s] <- f[1, t, s]
    }

    # One above main diagonal
    for (t in 1:(n.occasions - 1)) {
      pr.j[t, t + 1, s] <- sj[s, t] * f[2, t + 1, s]
    }

    # Two or further above main diagonal
    for (t in 1:(n.occasions - 2)) {
      for (j in (t + 2):n.occasions) {
        pr.j[t, j, s] <- sj[s, t] * prod(sa[s, (t + 1):(j - 1)]) * f[2, j, s]
      }
    }
    # Below main diagonal
    for (t in 2:n.occasions) {
      for (j in 1:(t - 1)) {
        pr.j[t, j, s] <- 0
      }
    }
```

```
    # Last column: probability of non-recovery
    for (t in 1:n.occasions) {
      pr.j[t, (n.occasions + 1), s] <- 1 - sum(pr.j[t, 1:n.occasions, s])
    } # t


    # Cell probabilities of the adult m-array

    # Main diagonal
    for (t in 1:n.occasions) {
      pr.a[t, t, s] <- f[2, t, s]
    }

    # Above main diagonal
    for (t in 1:(n.occasions - 1)) {
      for (j in (t + 1):n.occasions) {
        pr.a[t, j, s] <- prod(sa[s, t:(j - 1)]) * f[2, j, s]
      }
    }

    # Below main diagonal
    for (t in 2:n.occasions) {
      for (j in 1:(t - 1)) {
        pr.a[t, j, s] <- 0
      }
    }

    # Last column: probability of non-recovery
    for (t in 1:n.occasions) {
      pr.a[t, ((n.occasions) + 1), s] <- 1 - sum(pr.a[t, 1:(n.occasions), s])
    }
  } # s

  ## ------------------------------------------------------------------------------
  ## Density sub-model
  ## ------------------------------------------------------------------------------
  # Priors
  for (s in 1:4) {
    sigma.y[s] ~ T(dt(0, pow(2.5, -2), 1), 0, )
    tau.y[s] <- pow(sigma.y[s], -2)
  }

  ## initial population density
  for (s in 1:4) {
    N[1, 1, s] ~ dgamma(1.1, 3)
    N[2, 1, s] ~ dgamma(1.1, 3)
  }

  ## Likelihood for population density data (state-space model)
  # Process model
  for (s in 1:4) {
    for (t in 1:(n.occasions)) {
      N[1, t + 1, s] <- ratio[t, s] * sj[s, t] * (N[1, t, s] + N[2, t, s])
      N[2, t + 1, s] <- sa[s, t] * (N[1, t, s] + N[2, t, s])
    }
  }

  # Combining juveniles and adults
  for (s in 1:4) {
    for (t in 1:n.occasions + 1) {
      Ntot[t, s] <- N[1, t, s] + N[2, t, s] ## first term is juvies that survived, second is
adults that survived
    }
  }
```

```
  # Observation model
  for (s in 1:4) {
    for (t in 1:n.occasions + 1) {
      dens[t, s] ~ T(dnorm(Ntot[t, s], tau.y[s]), 0, ) ## true density is a function of observed
density with associated error
    }
  }
}) # end NIMBLE model code


n.occasions <- dim(m_arr_a)[1]
n.age <- 2
n.site <- 4


constants <- list(
  n.occasions = n.occasions,
  alpha_rr = alpha_rr,
  beta_rr = beta_rr
)



data_nimble <- list(
  wing_t = rbind(matrix(750, ncol = 4, nrow = 6), as.matrix(wings_total), matrix(750, ncol = 4,
nrow = 1)),
  wing_j = rbind(matrix(NA, ncol = 4, nrow = 6), as.matrix(wings_hatchyear), matrix(NA, ncol = 4,
nrow = 1)),
  m_arr_j = m_arr_j,
  m_arr_a = m_arr_a,
  rel_j = rel_j,
  rel_a = rel_a,
  dens = dens_mat
)

# initiate the N array
init.N <- array(NA, dim = c(2, n.occasions + 1, 4))
init.N[1, 1, ] <- 5
init.N[2, 1, ] <- 5


# Initial values
initial_values <- function() {
  list( ## random effects
    sigma_sj = 1,
    sigma_sa = 1,
    sigma_sjy = 1,
    sigma_say = 1,
    ratio_sigma_site = 1,
    ratio_sigma_year = 1,
    sigma_kill_j_site = 1,
    sigma_kill_a_site = 1,
    sigma_kill_j_year = 1,
    sigma_kill_a_year = 1,
    sigma.y = rep(1, 4),
    cr = 0.26,
    rr = 0.45,
    f = array(0.2, dim = c(n.age, n.occasions, n.site)),
    kill_j_grand = 0.2,
    kill_a_grand = 0.2,
    # inititiate unobserved values of wing_j
    wing_j = rbind(matrix(500, ncol = 4, nrow = 6), matrix(NA, ncol = 4, nrow = 18), matrix(500,
ncol = 4, nrow = 1)),
    logit_kill_j_site = rep(0.2, 4),
```

```
    logit_kill_a_site = rep(0.2, 4),
    logit_kill_j = matrix(0.2, ncol = 4, nrow = n.occasions),
    logit_kill_a = matrix(0.2, ncol = 4, nrow = n.occasions),
    ratio_mean = 0.5,
    log_ratio_site = rep(0.5, 4),
    log_ratio = matrix(1, ncol = 4, nrow = n.occasions),
    q_obs = matrix(0.8, ncol = 4, nrow = n.occasions),
    sj_grand = 0.3,
    sa_grand = 0.5,
    logit_sj_site = plogis(rep(0.3, 4)),
    logit_sa_site = plogis(rep(0.5, 4)),
    logit_sj = plogis(matrix(0.3, nrow = 4, ncol = n.occasions)),
    logit_sa = plogis(matrix(0.5, nrow = 4, ncol = n.occasions)),
    N = init.N,
    # initiate the values that are NA for dens
    dens = ifelse(is.na(dens_mat), 1.5, NA)
  )
}

# Check
model <- nimbleModel(
  code = model_code,
  data = data_nimble,
  inits = initial_values(),
  constants = constants,
  calculate = FALSE
)

## model checks
model$check()
model$calculate()

# Parameters monitored
parameters <- c(
  "ratio", "ratio_site", "ratio_mean", "q_obs",
  "ratio_sigma_site", "ratio_sigma_year",
  "sigma_kill_j_site", "sigma_kill_a_site", "sigma_kill_j_year", "sigma_kill_a_year",
  "sigma.y", "sigma_sj", "sigma_sa", "sigma_sjy", "sigma_say",
  "kill_j_grand", "kill_a_grand",
  "kill_j_site", "kill_a_site", "kill_j", "kill_a", "cr", "rr", "f",
  "f", "sj", "sa", "sj_grand", "sa_grand", "sj_site", "sa_site",
  "N", "Ntot"
)

# Parallelize and sample
n.chains <- 4
n.iter <- 800000
cl <- makeCluster(n.chains)


run.mcmc <- function(code, data, consts, inits, params, params2, niter) {
  library(nimble)
  library(coda)


  model <- nimbleModel(
    code = code,
    constants = consts,
    data = data,
    inits = inits
  )

  model_comp <- compileNimble(model)
  mcmc_model <- buildMCMC(model_comp, useConjugacy = FALSE, monitors = params)
  mcmc_model_c <- compileNimble(mcmc_model, project = model)
```

```
  results <- runMCMC(mcmc_model_c, niter = niter, thin = 10, nburnin = 50000)
  return(results)
}


## Run model
## set path for C++ compiler

system.time(
  output <- parLapply(
    cl = cl, X = 1:n.chains,
    fun = run.mcmc,
    code = model_code,
    data = data_nimble,
    consts = constants,
    inits = initial_values(),
    params = parameters,
    niter = n.iter
  )
)

stopCluster(cl)

# saveRDS(output, file = "../results/MODO.rds")

# Extract samples
samples <- list(chain1 = output[[1]], chain2 = output[[2]], chain3 = output[[3]], chain4 =
output[[4]])

# Remove burn-in (recall that the sample is already thinned by 10)
n.burnin <- 20000
samples <- list(
  chain1 = samples$chain1[-c(1:n.burnin), ],
  chain2 = samples$chain2[-c(1:n.burnin), ],
  chain3 = samples$chain3[-c(1:n.burnin), ],
  chain4 = samples$chain4[-c(1:n.burnin), ]
)


mcmc_list <- as.mcmc.list(lapply(samples, mcmc))
sum <- MCMCsummary(mcmc_list) ## convergence check

paste(
  "burnin", n.burnin, "low_neff", sum %>% filter(n.eff < 1000) %>% nrow(),
  "bad_rhat", nrow(sum[which(sum$Rhat > 1.1), ])
)

## print any parameters with low n effective
sum %>% filter(n.eff < 1000)


## check traceplots manually for specific parameters
param <- "sj_grand" # "ratio_site[4]" "sj_site[1]"
col_idx <- which(colnames(samples$chain1) == param)

# Years and site to add into plots
sites <- c(rep("PP", 25), rep("MAP", 25), rep("OP", 25), rep("DT", 25))
years <- c(1999:2023, 1999:2023, 1999:2023, 1999:2023)

## For survival
sites_survival <- rep(c("PP", "MAP", "OP", "DT"), 25)
years_survival <- rep(1999:2023, each = 4)
```

```
## For density
year_full <- c(1999:2024, 1999:2024, 1999:2024, 1999:2024)
sitesN <- c(rep("PP", 26), rep("MAP", 26), rep("OP", 26), rep("DT", 26))
yearsN <- c(1:26, 1:26, 1:26, 1:26)


## Extract parameter values from output
## Harvest
kill_j <- subset(sum, grepl("kill_j[", rownames(sum), fixed = TRUE)) %>% mutate(site = sites, year
= years)
kill_a <- subset(sum, grepl("kill_a[", rownames(sum), fixed = TRUE)) %>% mutate(site = sites, year
= years)
kill_a_grand <- subset(sum, grepl("kill_a_grand", rownames(sum), fixed = TRUE))
kill_j_grand <- subset(sum, grepl("kill_j_grand", rownames(sum), fixed = TRUE))
f_j <- subset(sum, grepl("f[1", rownames(sum), fixed = TRUE)) %>% mutate(site = sites, year =
years)
f_a <- subset(sum, grepl("f[2", rownames(sum), fixed = TRUE)) %>% mutate(site = sites, year =
years)

cr <- subset(sum, grepl("cr", rownames(sum), fixed = TRUE))
rr <- subset(sum, grepl("rr", rownames(sum), fixed = TRUE))

## Survival
sj <- subset(sum, grepl("sj[", rownames(sum), fixed = TRUE)) %>% mutate(site = sites_survival,
year = years_survival)
sa <- subset(sum, grepl("sa[", rownames(sum), fixed = TRUE)) %>% mutate(site = sites_survival,
year = years_survival)
sa_grand <- subset(sum, grepl("sa_grand", rownames(sum), fixed = TRUE))
sj_grand <- subset(sum, grepl("sj_grand", rownames(sum), fixed = TRUE))

## Productivity
q_obs <- subset(sum, grepl("q_obs", rownames(sum), fixed = TRUE)) %>% mutate(site = sites, year =
years)

ratio <- subset(sum, grepl("ratio[", rownames(sum), fixed = TRUE)) %>% mutate(site = sites, year =
years, y = years)
ratio_mean <- subset(sum, grepl("ratio_mean", rownames(sum), fixed = TRUE))

## Density
N1 <- subset(sum, grepl("N[1", rownames(sum), fixed = TRUE)) %>% mutate(years = yearsN, site =
sitesN)
N2 <- subset(sum, grepl("N[2", rownames(sum), fixed = TRUE)) %>% mutate(years = yearsN, site =
sitesN)
Ntot <- subset(sum, grepl("Ntot[", rownames(sum), fixed = TRUE)) %>% mutate(years = yearsN, site =
sitesN, year = year_full)

## sigmas
sigma_kill_a_site <- subset(sum, grepl("sigma_kill_a_site", rownames(sum), fixed = TRUE))
sigma_kill_a_year <- subset(sum, grepl("sigma_kill_a_year", rownames(sum), fixed = TRUE))
sigma_kill_j_site <- subset(sum, grepl("sigma_kill_j_site", rownames(sum), fixed = TRUE))
sigma_kill_j_year <- subset(sum, grepl("sigma_kill_j_year", rownames(sum), fixed = TRUE))
sigma_sa <- subset(sum, grepl("sigma_sa", rownames(sum), fixed = TRUE))
sigma_say <- subset(sum, grepl("sigma_say", rownames(sum), fixed = TRUE))
sigma_sj <- subset(sum, grepl("sigma_sj", rownames(sum), fixed = TRUE))
sigma_sjy <- subset(sum, grepl("sigma_sjy", rownames(sum), fixed = TRUE))
ratio_sigma_site <- subset(sum, grepl("ratio_sigma_site", rownames(sum), fixed = TRUE))
ratio_sigma_year <- subset(sum, grepl("ratio_sigma_year", rownames(sum), fixed = TRUE))
sigma_dens_site <- subset(sum, grepl("sigma.y", rownames(sum), fixed = TRUE))



# Posterior correlations -------------------------------------------------

all_chains <- rbind(samples$chain1, samples$chain2, samples$chain3, samples$chain4)
```

```
Ntot <- all_chains[, grepl("Ntot[", colnames(mcmc_list$chain1), fixed = TRUE)]

Ntot_PP <- Ntot[, grepl("1]", colnames(Ntot))]
Ntot_MAP <- Ntot[, grepl("2]", colnames(Ntot))]
Ntot_OP <- Ntot[, grepl("3]", colnames(Ntot))]
Ntot_DT <- Ntot[, grepl("4]", colnames(Ntot))]

lamba_PP <- Ntot_PP[, 2:26] / Ntot_PP[, 1:25]
lamba_MAP <- Ntot_MAP[, 2:26] / Ntot_MAP[, 1:25]
lamba_OP <- Ntot_OP[, 2:26] / Ntot_OP[, 1:25]
lamba_DT <- Ntot_DT[, 2:26] / Ntot_DT[, 1:25]

## The first Ntot 26 columns are PP, the next 26 are MAP, then OP, then PP

## N1
N1 <- all_chains[, grepl("N[1", colnames(mcmc_list$chain1), fixed = TRUE)]

N1_PP <- N1[, grepl("1]", colnames(N1))]
N1_MAP <- N1[, grepl("2]", colnames(N1))]
N1_OP <- N1[, grepl("3]", colnames(N1))]
N1_DT <- N1[, grepl("4]", colnames(N1))]


## N2
N2 <- all_chains[, grepl("N[2", colnames(mcmc_list$chain1), fixed = TRUE)]

N2_PP <- N2[, grepl("1]", colnames(N2))]
N2_MAP <- N2[, grepl("2]", colnames(N2))]
N2_OP <- N2[, grepl("3]", colnames(N2))]
N2_DT <- N2[, grepl("4]", colnames(N2))]


## ratio is blocks of 25 columns (one less year), same order of regions
ratio <- all_chains[, grepl("ratio[", colnames(mcmc_list$chain1), fixed = TRUE)]

ratio_PP <- ratio[, grepl("1]", colnames(ratio))]
ratio_MAP <- ratio[, grepl("2]", colnames(ratio))]
ratio_OP <- ratio[, grepl("3]", colnames(ratio))]
ratio_DT <- ratio[, grepl("4]", colnames(ratio))]


## survival is grouped by year, so first 4 columns are year 1 for the four sites
sa <- all_chains[, grepl("sa[", colnames(mcmc_list$chain1), fixed = TRUE)]

sa_PP <- sa[, grepl("\\[1", colnames(sa))]
sa_MAP <- sa[, grepl("\\[2", colnames(sa))]
sa_OP <- sa[, grepl("\\[3", colnames(sa))]
sa_DT <- sa[, grepl("\\[4", colnames(sa))]

sj <- all_chains[, grepl("sj[", colnames(mcmc_list$chain1), fixed = TRUE)]

sj_PP <- sj[, grepl("\\[1", colnames(sj))]
sj_MAP <- sj[, grepl("\\[2", colnames(sj))]
sj_OP <- sj[, grepl("\\[3", colnames(sj))]
sj_DT <- sj[, grepl("\\[4", colnames(sj))]


corr_sa_lambda_PP <- sapply(1:nrow(lamba_PP), function(i) {
  cor(lamba_PP[i, ], sa_PP[i, ])
})
corr_sj_lambda_PP <- sapply(1:nrow(lamba_PP), function(i) {
  cor(lamba_PP[i, ], sj_PP[i, ])
})
corr_ratio_lambda_PP <- sapply(1:nrow(lamba_PP), function(i) {
  cor(lamba_PP[i, ], ratio_PP[i, ])
```

```
})
all_corr_PP <- cbind(corr_sa_lambda_PP, corr_sj_lambda_PP, corr_ratio_lambda_PP)
colnames(all_corr_PP) <- c("adult survival", "juvies survival", "ratio")

corr_sa_lambda_MAP <- sapply(1:nrow(lamba_MAP), function(i) {
  cor(lamba_MAP[i, ], sa_MAP[i, ])
})
corr_sj_lambda_MAP <- sapply(1:nrow(lamba_MAP), function(i) {
  cor(lamba_MAP[i, ], sj_MAP[i, ])
})
corr_ratio_lambda_MAP <- sapply(1:nrow(lamba_MAP), function(i) {
  cor(lamba_MAP[i, ], ratio_MAP[i, ])
})
all_corr_MAP <- cbind(corr_sa_lambda_MAP, corr_sj_lambda_MAP, corr_ratio_lambda_MAP)
colnames(all_corr_MAP) <- c("adult survival", "juvies survival", "ratio")

corr_sa_lambda_OP <- sapply(1:nrow(lamba_OP), function(i) {
  cor(lamba_OP[i, ], sa_OP[i, ])
})
corr_sj_lambda_OP <- sapply(1:nrow(lamba_OP), function(i) {
  cor(lamba_OP[i, ], sj_OP[i, ])
})
corr_ratio_lambda_OP <- sapply(1:nrow(lamba_OP), function(i) {
  cor(lamba_OP[i, ], ratio_OP[i, ])
})
all_corr_OP <- cbind(corr_sa_lambda_OP, corr_sj_lambda_OP, corr_ratio_lambda_OP)
colnames(all_corr_OP) <- c("adult survival", "juvies survival", "ratio")

corr_sa_lambda_DT <- sapply(1:nrow(lamba_DT), function(i) {
  cor(lamba_DT[i, ], sa_DT[i, ])
})
corr_sj_lambda_DT <- sapply(1:nrow(lamba_DT), function(i) {
  cor(lamba_DT[i, ], sj_DT[i, ])
})
corr_ratio_lambda_DT <- sapply(1:nrow(lamba_DT), function(i) {
  cor(lamba_DT[i, ], ratio_DT[i, ])
})
all_corr_DT <- cbind(corr_sa_lambda_DT, corr_sj_lambda_DT, corr_ratio_lambda_DT)
colnames(all_corr_DT) <- c("adult survival", "juvies survival", "ratio")


apply(all_corr_PP, 2, mean)
apply(all_corr_MAP, 2, mean)
apply(all_corr_OP, 2, mean)
apply(all_corr_DT, 2, mean)

summarize_corr <- function(mat) {
  apply(mat, 2, function(x) {
    c(
      mean = mean(x),
      lwr = quantile(x, 0.025),
      med = quantile(x, 0.5),
      upr = quantile(x, 0.975)
    )
  })
}



# Combine all correlation data into one tidy data frame
to_tidy <- function(df, site_name) {
  as.data.frame(df) %>%
    setNames(c("adult_survival", "juvenile_survival", "productivity")) %>%
    pivot_longer(cols = everything(), names_to = "parameter", values_to = "correlation") %>%
    group_by(parameter) %>%
```

```
    summarise(
      mean = mean(correlation),
      mode = {
        d <- density(correlation, na.rm = TRUE)
        d$x[which.max(d$y)]
      },
      lower = quantile(correlation, 0.025), ## 95% CI
      upper = quantile(correlation, 0.975),
      .groups = "drop"
    ) %>%
    mutate(site = site_name)
}

# Apply to all sites
df_PP <- to_tidy(all_corr_PP, "PP")
df_MAP <- to_tidy(all_corr_MAP, "MAP")
df_OP <- to_tidy(all_corr_OP, "OP")
df_DT <- to_tidy(all_corr_DT, "DT")
```