Appendix S3. Extended multievent model: Arnason-Schwarz without the robust design

Matthijs Hollanders & J. Andrew Royle

Manuscript title: Know what you don't know: Embracing state uncertainty in disease-structured multistate models

Table of contents

Introduction **2**

Simulating data **3**
- Input parameters . . . 3
- Transition probability matrices . . . 3
- Generating capture histories . . . 5

Extended multievent model **7**
- Model code . . . 7
- Initial values for latent ecological and observed states . . . 11
- Composing infection intensity data . . . 12
- Data, constants, initial values, and parameters to monitor . . . 12
- Run model . . . 13

Results **15**
- Traceplots and summary statistics . . . 15
- Posterior distributions . . . 18

References **20**

Introduction

In the first section, we simulate multievent mark-recapture Arnason-Schwarz (Arnason, 1973, 1972; Pradel, 2005; Schwarz et al., 1993) data for animals with two pathogen infection states using R 4.2.0 (R Core Team, 2022). We simulate the ecological process, consisting of survival and infection state transitions, in discrete-time. The observation process is simulated without the robust design using "traditional" single surveys (i.e., one survey at a time). During each survey, multiple samples (e.g., swabs) are collected from individuals to detect the pathogen, which happens imperfectly. Each sample is subjected to multiple diagnostic runs (e.g., qPCR), which also features imperfect pathogen detection, yielding infection status (binary) and infection intensity (continuous) data.

This model differs from the robust design formulation in that the *observation process* of the robust design, consisting of both the recapture and the sampling pathogen detection probabilities, is now split into two separate processes (note that matrix multiplication of the two transition probability matrices [TPMs] below yields the observation process of the robust design). First, there is the recapture process where the observed state (o) of individual i during survey t is conditional on ecological state ($z_{i,t}$):

$$o_{i,t} | z_{i,t} \sim \text{Categorical} \begin{array}{c} \\ \\ \end{array} \begin{array}{cccc} & & o_{i,t} & \\ & \text{Seen, } Bd\text{–} & \text{Seen, } Bd\text{+} & \text{Not seen} \end{array}$$

$$o_{i,t} | z_{i,t} \sim \text{Categorical} \left(\begin{array}{ccc|ll} \text{Seen, } Bd\text{–} & \text{Seen, } Bd\text{+} & \text{Not seen} & & \\ p_1 & 0 & 1 - p_1 & \text{Alive, } Bd\text{–} & \\ 0 & p_2 & 1 - p_2 & \text{Alive, } Bd\text{+} & z_{i,t} \\ 0 & 0 & 1 & \text{Dead} & \end{array} \right)$$

Then in the sampling process, the sample state (s) of individual i, survey t, and replicate sample k is conditional on the observed state:

$$s_{i,t,k} | o_{i,t} \sim \text{Categorical} \left(\begin{array}{ccc|ll} \text{Sample } Bd\text{–} & \text{Sample } Bd\text{+} & \text{Not sampled} & & \\ 1 - \delta_{21} & \delta_{21} & 0 & \text{Seen, } Bd\text{–} & \\ 1 - \delta_{22} & \delta_{22} & 0 & \text{Seen, } Bd\text{+} & o_{i,t} \\ 0 & 0 & 1 & \text{Not seen} & \end{array} \right)$$

In the second section, we demonstrate how to analyze datasets with Bayesian inference and Markov chain Monte Carlo (MCMC) methods using NIMBLE 0.12.2 (de Valpine et al., 2017). We write the model code in the NIMBLE dialect of the BUGS language (which is easily adapted to work in BUGS or JAGS), generate suitable initial values for latent ecological, observed, and sample states, and condense the diagnostic run infection intensities to "long" format. We then run the MCMC, show traceplots and summary statistics, and plot the parameter estimates with the simulated values.

We created this document with the intention to show that the observation process with single surveys is unable to recover input parameter values, unlike the robust design. We postulate that the model structure has issues with identifiability and/or multimodality.

Simulating data

We simulate individuals experiencing changes in their in ecological states (z), observed states (o) conditional on these ecological states, sample states (s) from each recaptured individual, and diagnostic run states (y) conditional on the sample states. Ecological states (z) are (1) alive and uninfected, (2) alive and infected, and (3) dead (Kéry & Schaub, 2012). Individuals can transition between these states, represented by survival (conditional on infection intensity) and infection state transitions. Observed states (o) are (1) seen/captured while uninfected, (2) seen/captured while infected, (3) and not seen/captured. Sample states (s) are (1) sample uninfected, (2) sample infected, and (3) no sample, the last of which is only possible when the individual was not captured. For every sample, multiple diagnostic runs are conducted which let us estimate the probability of detecting an infection on an infected sample. There are three diagnostic run states (y): (1) uninfected, (2) infected, and (3) no diagnostic run, the last of which is only possible when there is no sample collected and therefore no capture of an individual. Pathogen detection probabilities on samples and diagnostic runs are modeled with occupancy-like submodels (multiple sample units with replicated measurements, MacKenzie et al., 2002) conditional on infection intensity. We simulate the occurrence of false-positives in both the sampling and diagnostic processes.

Input parameters

We specify the sampling condition and define parameter values for the simulation.

```r
# Number of ecological (z), observed (o), sample (s), and diagnostic (y) states
n.states <- 3

# Sampling conditions
n.ind <- 200        # Number of individuals in simulation
n.surv <- 8         # Number of survey occasions
n.samp <- 3         # Number of samples collected (per capture)
n.diag <- 3         # Number of diagnostic runs (per sample)

# Parameters
pi <- 0.4           # Probability of entering the study as infected
phi1 <- 0.9         # Survival probability if uninfected
phi2.alpha <- 0.9   # Survival probability if infected (intercept)
phi2.beta <- -0.5   # Effect of one "unit" of pathogen on survival (log odds)
psi12 <- 0.6        # Probability of gaining infection
psi21 <- 0.3        # Probability of clearing infection
p1 <- 0.6           # Recapture probability if uninfected
p2 <- 0.7           # Recapture probability if infected
r.delta <- 0.4      # Probability of detecting one "unit" of pathogen on a sample
delta21 <- 0.05     # False-positive probability of sampling process (e.g., swab)
r.lambda <- 0.6     # Probability of detecting one "unit" of pathogen on a diagnostic run
lambda21 <- 0.10    # False-positive probability of diagnostic process (e.g., qPCR)
mu <- 4             # Average population infection intensity
mu.sigma <-         # Population, sampling process, and diagnostic process infection
  c(0.4, 0.3, 0.2)  # intensity standard deviations
```

Transition probability matrices

We start with creating the TPMs of the ecological, observation, sampling, and diagnostic processes. We simulate the survival probability of infected individuals ($\phi_{2_{i,t}}$) as a logit-linear function of infection intensity, and the pathogen detection probabilities ($\delta_{22_{i,t}}$ and $\lambda_{22_{i,t}}$) following abundance-induced heterogeneity models (Lachish et al., 2012; Royle & Nichols, 2003). Note that, because Arnason-Schwarz models condition on first capture, we do not model the capture probabilities within the survey that an individual was first caught.

```r
# Arrays for ecological, observation, sampling, and diagnostic process TPMs
TPM.z <- TPM.o <- array(NA, c(n.states, n.states, n.ind, n.surv - 1))
TPM.s <- array(NA, c(n.states, n.states, n.ind, n.surv))
TPM.d <- array(NA, c(n.states, n.states, n.ind, n.surv, n.samp))

# Arrays for survival, pathogen detection, and infection intensities
phi2 <- array(NA, c(n.ind, n.surv - 1))
m <- delta22 <- array(NA, c(n.ind, n.surv))
n <- lambda22 <- array(NA, c(n.ind, n.surv, n.samp))

# Primary occasion that individuals were first captured
first <- sort(sample(1:(n.surv - 1), n.ind, replace = T))

# TPMs
for(i in 1:n.ind){
  for(t in first[i]:n.surv){

    # Individual infection intensity (lognormal)
    m[i,t] <- rlnorm(1, log(mu), mu.sigma[1])

  } # t

  for(t in first[i]:(n.surv - 1)){

    # Survival probability as a function of infection intensity
    phi2[i,t] <- plogis(qlogis(phi2.alpha) + phi2.beta * m[i,t])

    # Ecological process TPM
    TPM.z[,,i,t] <-
      matrix(c(phi1 * (1 - psi12), phi1 * psi12,           1 - phi1,
               phi2[i,t] * psi21,  phi2[i,t] * (1 - psi21), 1 - phi2[i,t],
               0,                  0,                       1),
             nrow = n.states, ncol = n.states, byrow = T)

    # Observation process TPM
    TPM.o[,,i,t] <-
      matrix(c(p1, 0,  1 - p1,
               0,  p2, 1 - p2,
               0,  0,  1),
             nrow = n.states, ncol = n.states, byrow = T)

  } # t

  for(t in first[i]:n.surv){

    # Sample pathogen detection
    delta22[i,t] <- 1 - (1 - r.delta) ^ m[i,t]

    # Sampling process TPM
    TPM.s[,,i,t] <-
    matrix(c(1 - delta21,      delta21,      0,
             1 - delta22[i,t], delta22[i,t], 0,
             0,                0,            1),
           nrow = n.states, ncol = n.states, byrow = T)

    for(k in 1:n.samp){

      # Sample infection intensity
      n[i,t,k] <- rnorm(1, m[i,t], mu.sigma[2])

      # Diagnostic pathogen detection
      lambda22[i,t,k] <- 1 - (1 - r.lambda) ^ n[i,t,k]

      # Diagnostic process TPM
      TPM.d[,,i,t,k] <-
        matrix(c(1 - lambda21,        lambda21,        0,
                 1 - lambda22[i,t,k], lambda22[i,t,k], 0,
                 0,                   0,               1),
               nrow = n.states, ncol = n.states, byrow = T)

    } # k
  } # t
} # i
```

Generating capture histories

We run the simulation, which yields the two 4-dimensional capture histories (y and x) that we use as data for the model.

```
# Arrays for ecological, observed, sample, and diagnostic states, and infection intensities
z <- o <- array(NA, c(n.ind, n.surv))
s <- array(NA, c(n.ind, n.surv, n.samp))
y <- x <- array(NA, c(n.ind, n.surv, n.samp, n.diag))

# We start using NIMBLE's functionality here with the rcat() function
library(nimble)

# Simulation
for(i in 1:n.ind){

  # Ecological and observed state at first capture
  z[i,first[i]] <- o[i,first[i]] <- rcat(1, c(1 - pi, pi))

  for(t in (first[i] + 1):n.surv){

    # Ecological process
    z[i,t] <- rcat(1, TPM.z[z[i,t-1],,i,t-1])

    # Observation process
    o[i,t] <- rcat(1, TPM.o[z[i,t],,i,t-1])
  } # t

  for(t in first[i]:n.surv){
    for(k in 1:n.samp){

      # Sampling process
      s[i,t,k] <- rcat(1, TPM.s[o[i,t],,i,t])

      for(l in 1:n.diag){

        # Diagnostic process
        y[i,t,k,l] <- rcat(1, TPM.d[s[i,t,k],,i,t,k])

        # Diagnostic run infection intensity
        if(y[i,t,k,l] == 2){
          x[i,t,k,l] <- rnorm(1, n[i,t,k], mu.sigma[3])
        }

      } # l
    } # k
  } # t
} # i
```

Extended multievent model

Model code

We write the model code using NIMBLE. We specify vague Beta and weakly informative (half-)Cauchy priors for parameters, define functions for model parameters, write out the TPMs, specify the likelihood for the multievent model, and use “long” format without NAs for the infection intensity model. We use a mix of centered and non-centered parameterization for modeling infection intensities to improve MCMC mixing (Papaspiliopoulos & Roberts, 2003). Note that all parameters can be modeled as functions of covariates with appropriate link functions.

```
MEcode <- nimbleCode({

  # PRIORS

  # Constrain false-positive and true-positive probabilities to avoid multimodality
  constraint ~ dconstraint(delta21 < r.delta & lambda21 < r.lambda)

  # Multievent
  pi.alpha ~ dbeta(1, 1)
  phi1.alpha ~ dbeta(1, 1)
  phi2.alpha ~ dbeta(1, 1)
  phi2.beta ~ dt(0, sigma = 1, df = 1)
  psi12.alpha ~ dbeta(1, 1)
  psi21.alpha ~ dbeta(1, 1)
  p1.alpha ~ dbeta(1, 1)
  p2.alpha ~ dbeta(1, 1)
  r.delta ~ dbeta(1, 1)
  delta21 ~ dbeta(1, 1)
  r.lambda ~ dbeta(1, 1)
  lambda21 ~ dbeta(1, 1)

  # Infection intensity
  mu.alpha ~ dt(mu.prior, sigma = 1, df = 1)
  for(i in 1:3){
    mu.sigma[i] ~ T(dt(0, sigma = 1, df = 1), 0, )
  }

  # PARAMETERS

  for(i in 1:n.ind){

    # Probability of entering as infected
    pi[i] <- pi.alpha

    for(t in first[i]:(n.surv - 1)){

      # Mortality hazard rates
      phi1[i,t] <- phi1.alpha
      logit(phi2[i,t]) <- logit(phi2.alpha) + phi2.beta * m[i,t]

      # Infection state transition hazard rates
```

```
        psi12[i,t] <- psi12.alpha
        psi21[i,t] <- psi21.alpha

        # Recapture probabilities
        p1[i,t+1] <- p1.alpha
        p2[i,t+1] <- p2.alpha

      } # t

      # Recapture probabilities at first capture
      p1[i,first[i]] <- 1
      p2[i,first[i]] <- 1

      for(t in first[i]:n.surv){

        # Population infection intensity
        mu[i,t] <- mu.alpha

        # Individual infection intensity (lognormal)
        m[i,t] ~ dlnorm(mu[i,t], sdlog = mu.sigma[1])     # Centered parameterization
#        m.z[i,t] ~ dnorm(0, 1)
#        log(m[i,t]) <- mu[i,t] + m.z[i,t] * mu.sigma[1]  # Non-centered parameterization

        # Sample pathogen detection
        delta22[i,t] <- 1 - (1 - r.delta) ^ m[i,t]

        for(k in 1:n.samp){

          # Sample infection intensity
#          n[i,t,k] ~ dnorm(m[i,t], sd = mu.sigma[2])    # Centered parameterization
          n.z[i,t,k] ~ dnorm(0, 1)
          n[i,t,k] <- m[i,t] + n.z[i,t,k] * mu.sigma[2]  # Non-centered parameterization

          # Diagnostic pathogen detection
          lambda22[i,t,k] <- 1 - (1 - r.lambda) ^ n[i,t,k]

        } # k
      } # t

      # TRANSITION PROBABILITY MATRICES

      # ECOLOGICAL PROCESS (survival and infection state transitions)

      # First capture
      TPM.z.first[1,i] <- 1 - pi[i]  # Enters as uninfected
      TPM.z.first[2,i] <- pi[i]      # Enters as infected

      for(t in first[i]:(n.surv - 1)){

        # Alive, uninfected
        TPM.z[1,1,i,t] <- phi1[i,t] * (1 - psi12[i,t])  # Remains uninfected
        TPM.z[1,2,i,t] <- phi1[i,t] * psi12[i,t]        # Transitions to infected
        TPM.z[1,3,i,t] <- 1 - phi1[i,t]                 # Dies
```

```
    # Alive, infected
    TPM.z[2,1,i,t] <- phi2[i,t] * psi21[i,t]        # Transitions to uninfected
    TPM.z[2,2,i,t] <- phi2[i,t] * (1 - psi21[i,t])  # Remains infected
    TPM.z[2,3,i,t] <- 1 - phi2[i,t]                 # Dies

    # Dead
    TPM.z[3,1,i,t] <- 0                             # Transitions to uninfected
    TPM.z[3,2,i,t] <- 0                             # Transitions to infected
    TPM.z[3,3,i,t] <- 1                             # Remains dead

  } # t

  for(t in first[i]:n.surv){

    # OBSERVATION PROCESS (recapture)

    # Alive, uninfected
    TPM.o[1,1,i,t] <- p1[i,t]      # Seen, uninfected
    TPM.o[1,2,i,t] <- 0            # Seen, infected
    TPM.o[1,3,i,t] <- 1 - p1[i,t]  # Not seen

    # Alive, infected
    TPM.o[2,1,i,t] <- 0            # Seen, uninfected
    TPM.o[2,2,i,t] <- p2[i,t]      # Seen, infected
    TPM.o[2,3,i,t] <- 1 - p2[i,t]  # Not seen

    # Dead
    TPM.o[3,1,i,t] <- 0            # Seen, uninfected
    TPM.o[3,2,i,t] <- 0            # Seen, infected
    TPM.o[3,3,i,t] <- 1            # Not seen

    # SAMPLING PROCESS (sample pathogen detection)

    # Seen, uninfected
    TPM.s[1,1,i,t] <- 1 - delta21       # Sample uninfected (true -)
    TPM.s[1,2,i,t] <- delta21           # Sample infected (false +)
    TPM.s[1,3,i,t] <- 0                 # Not sampled

    # Seen, infected
    TPM.s[2,1,i,t] <- 1 - delta22[i,t]  # Sample uninfected (false -)
    TPM.s[2,2,i,t] <- delta22[i,t]      # Sample infected (true +)
    TPM.s[2,3,i,t] <- 0                 # Not sampled

    # Not seen
    TPM.s[3,1,i,t] <- 0                 # Sample uninfected
    TPM.s[3,2,i,t] <- 0                 # Sample infected
    TPM.s[3,3,i,t] <- 1                 # Not sampled

    for(k in 1:n.samp){

      # DIAGNOSTIC PROCESS (diagnostic pathogen detection)
```

```
        # Sample uninfected
        TPM.d[1,1,i,t,k] <- 1 - lambda21          # Uninfected (true -)
        TPM.d[1,2,i,t,k] <- lambda21              # Infected (false +)
        TPM.d[1,3,i,t,k] <- 0                     # No diagnostic run

        # Sample infected
        TPM.d[2,1,i,t,k] <- 1 - lambda22[i,t,k]   # Uninfected (false -)
        TPM.d[2,2,i,t,k] <- lambda22[i,t,k]       # Infected (true +)
        TPM.d[2,3,i,t,k] <- 0                     # No diagnostic run

        # Not sampled
        TPM.d[3,1,i,t,k] <- 0                     # Uninfected
        TPM.d[3,2,i,t,k] <- 0                     # Infected
        TPM.d[3,3,i,t,k] <- 1                     # No diagnostic run

      } # k
    } # t

    # LIKELIHOOD

    # Ecological state at first capture
    z[i,first[i]] ~ dcat(TPM.z.first[1:2,i])

    for(t in (first[i] + 1):n.surv){

      # Ecological process
      z[i,t] ~ dcat(TPM.z[z[i,t-1],1:3,i,t-1])

    } # t

    for(t in first[i]:n.surv){

      # Observation process
      o[i,t] ~ dcat(TPM.o[z[i,t],1:3,i,t])

      for(k in 1:n.samp){

        # Sampling process
        s[i,t,k] ~ dcat(TPM.s[o[i,t],1:3,i,t])

        for(l in 1:n.diag){

          # Diagnostic process
          y[i,t,k,l] ~ dcat(TPM.d[s[i,t,k],1:3,i,t,k])

        } # l
      } # k
    } # t
  } # i

  # INFECTION INTENSITY

  for(j in 1:n.x){
    # Likelihood
    x[j] ~ dnorm(n[ind[j],surv[j],samp[j]], sd = mu.sigma[3])

  } # j

})
```

Initial values for latent ecological and observed states

We generate suitable initial values for latent ecological (z), observed (o), and sample (s) states from our diagnostic capture history (y).

```
# Lowest observed state per individual per primary
observed <- apply(y, c(1, 2), function(x) min(x, na.rm = T))

# First and last captures
first.cap <- apply(observed, 1, function(x) min(which(x < 3)))
last.cap <- apply(observed, 1, function(x) max(which(x < 3)))

# Sample states
s.init <- array(NA, c(n.ind, n.surv, n.samp))
for(i in 1:n.ind){
  for(t in first.cap[i]:n.surv){
    for(k in 1:n.samp){
      s.init[i,t,k] <-
        # If missing survey, sample is "not sampled"
        ifelse(length(which(is.na(y[i,t,k,]))) == n.diag, 3,
               # If no diagnostic runs, sample is "not sampled"
               ifelse(min(y[i,t,k,], na.rm = T) == 3, 3,
                      # Otherwise, assign highest observed diagnostic state
                      max(y[i,t,k,][y[i,t,k,] < 3], na.rm = T)))
    } # k
  } # t
} # i

# Observed and ecological states
o.init <- z.init <- array(NA, c(n.ind, n.surv))
for(i in 1:n.ind){
  for(t in first.cap[i]:last.cap[i]){
    # If not captured during survey
    if(min(s.init[i,t,]) == 3){
      o.init[i,t] <- 3                # "Not captured"
      z.init[i,t] <- sample(1:2, 1)   # Randomly assign alive state
    } else {
      # If captured, assign highest sample state
      o.init[i,t] <- z.init[i,t] <- max(s.init[i,t,][s.init[i,t,] < 3])
    }
  } # t
  # If not captured during the last survey
  if(last.cap[i] < n.surv){
    # Assign "not captured" or "dead" state after last capture
    for(t in (last.cap[i] + 1):n.surv){
      o.init[i,t] <- z.init[i,t] <- 3
    } # t
  }
} # i
```

Composing infection intensity data

We convert our 4-dimensional infection intensity array (x) into tidy ("long") format without NAs to improve MCMC mixing and speed.

```
x.long <- na.omit(reshape2::melt(x,
                                 value.name = "x",
                                 varnames = c("ind", "surv", "samp", "diag")))
```

Data, constants, initial values, and parameters to monitor

We package the data, constants, initial values, and parameters to monitor to feed to NIMBLE.

```
# Data
str(MEdata <- list(y = y,
                   x = x.long$x,
                   constraint = 1))
```

```
List of 3
 $ y         : int [1:200, 1:8, 1:3, 1:3] 1 1 1 2 2 1 2 1 2 2 ...
 $ x         : num [1:1947] 8.77 2.39 6.09 6.51 3.15 ...
 $ constraint: num 1
```

```
# Constants
mu.prior <- mean(log(x), na.rm = T)
str(MEconsts <- list(# Multievent
                     n.ind = dim(y)[1],
                     n.surv = dim(y)[2],
                     n.samp = dim(y)[3],
                     n.diag = dim(y)[4],
                     first = first.cap,

                     # Infection intensity
                     n.x = nrow(x.long),
                     ind = x.long$ind,
                     surv = x.long$surv,
                     samp = x.long$samp,
                     mu.prior = mu.prior))
```

```
List of 10
 $ n.ind   : int 200
 $ n.surv  : int 8
 $ n.samp  : int 3
 $ n.diag  : int 3
 $ first   : int [1:200] 1 1 1 1 1 1 1 1 1 1 ...
 $ n.x     : int 1947
 $ ind     : int [1:1947] 4 5 7 9 10 14 16 19 20 22 ...
 $ surv    : int [1:1947] 1 1 1 1 1 1 1 1 1 1 ...
 $ samp    : int [1:1947] 1 1 1 1 1 1 1 1 1 1 ...
 $ mu.prior: num 1.46
```

```
# Initial values for latent states, pathogen detection, and infection intensity
MEinits <- list(z = z.init, o = o.init, s = s.init,
                r.delta = 3/4, delta21 = 1/4,
                r.lambda = 3/4, lambda21 = 1/4,
                mu.alpha = mu.prior, mu.sigma = runif(3))

# Parameters to monitor
MEmons <- c("pi.alpha",
            "phi1.alpha", "phi2.alpha", "phi2.beta",
            "psi12.alpha", "psi21.alpha",
            "p1.alpha", "p2.alpha",
            "r.delta", "delta21",
            "r.lambda", "lambda21",
            "mu.alpha", "mu.sigma")
```

Run model

We run the model with NIMBLE's default MCMC configuration.

```
MEstart <- Sys.time()
# Model
MEmodel <- nimbleModel(MEcode, MEconsts, MEdata, MEinits, calculate = F, check = F)
cMEmodel <- compileNimble(MEmodel)

# MCMC
MEconf <- configureMCMC(MEmodel, monitors = MEmons)
MEmcmc <- buildMCMC(MEconf)
cMEmcmc <- compileNimble(MEmcmc, project = cMEmodel)

# Run MCMC
nchains <- 3
niter <- 5000
nburnin <- 1000
MEsamples <- runMCMC(cMEmcmc, nchains = nchains, niter = niter)
MEend <- Sys.time()
```

```
MEend - MEstart
```

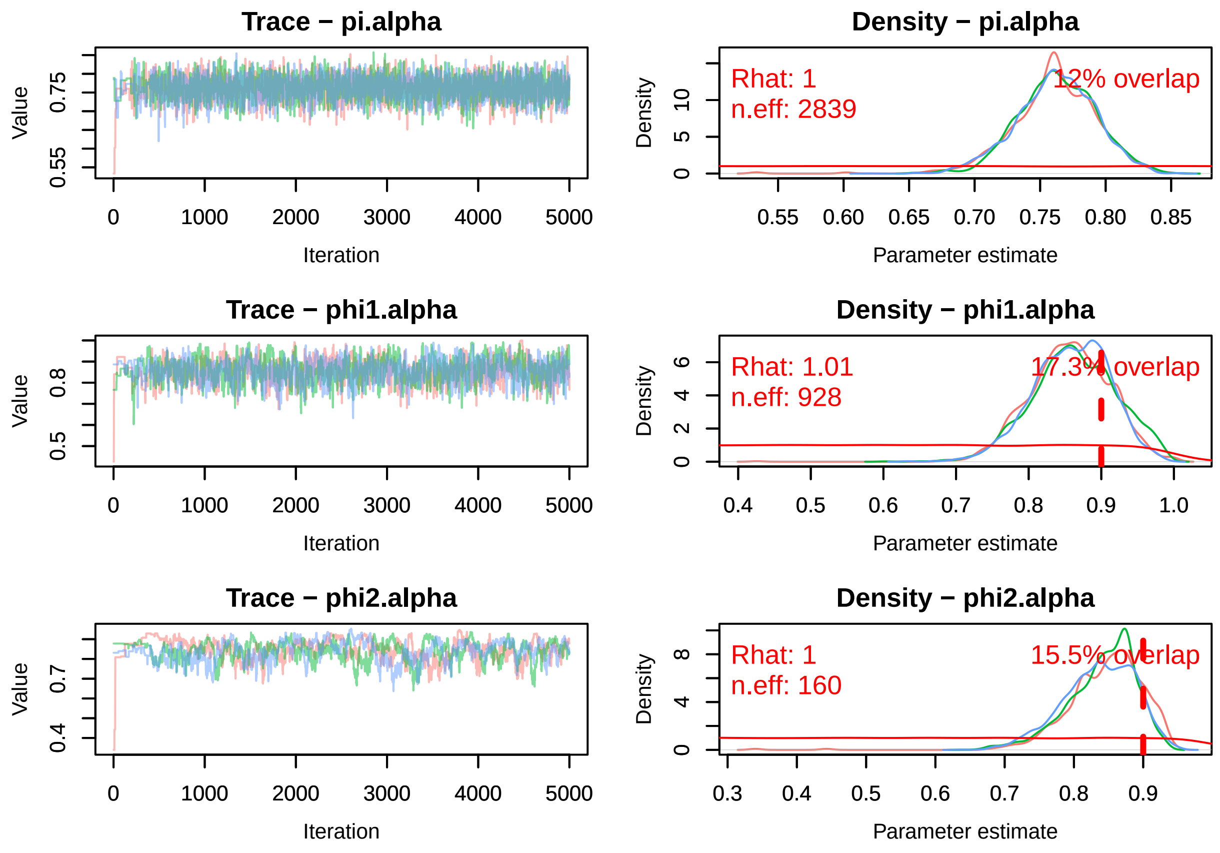
```
Time difference of 15 mins
```

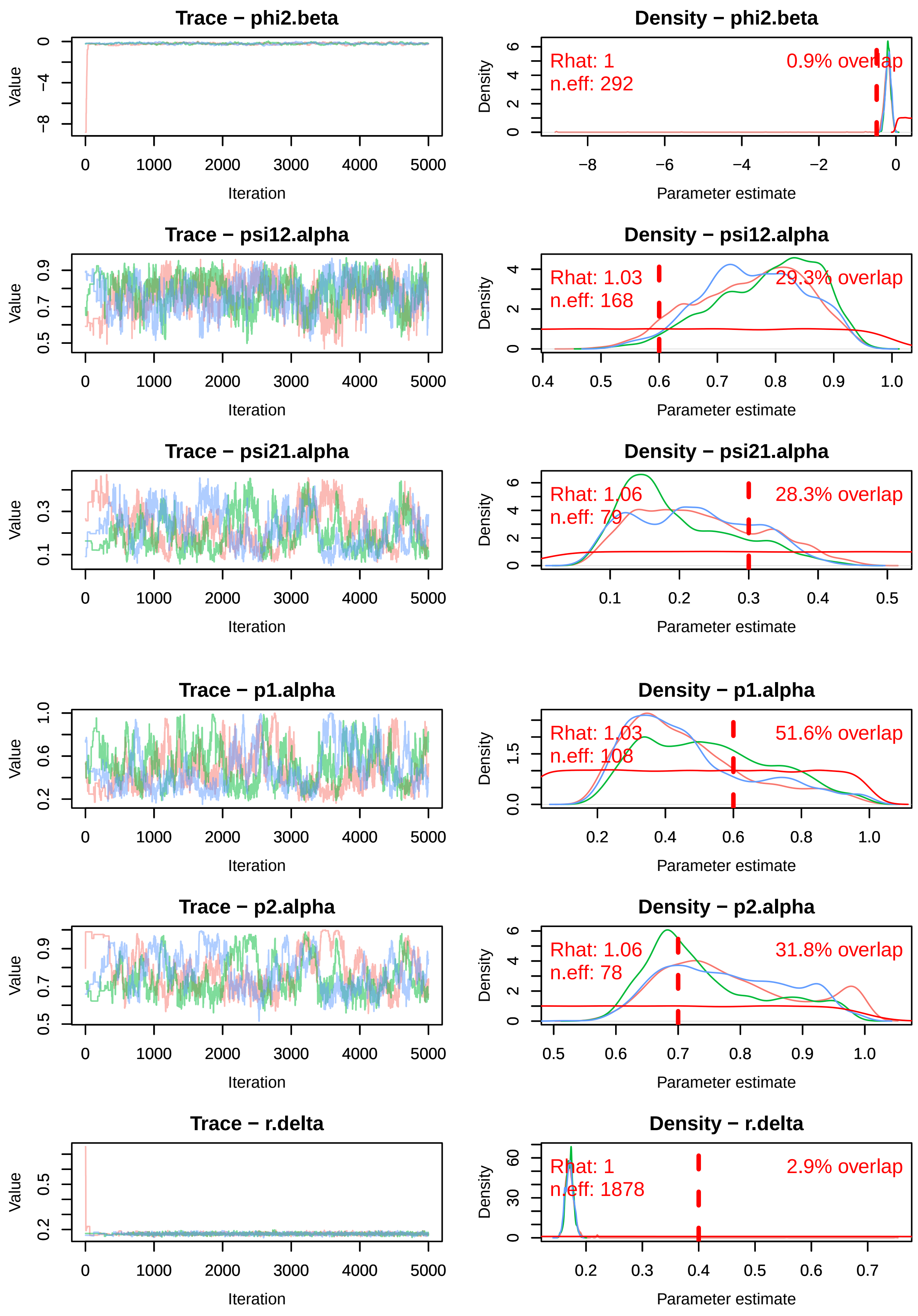
Results

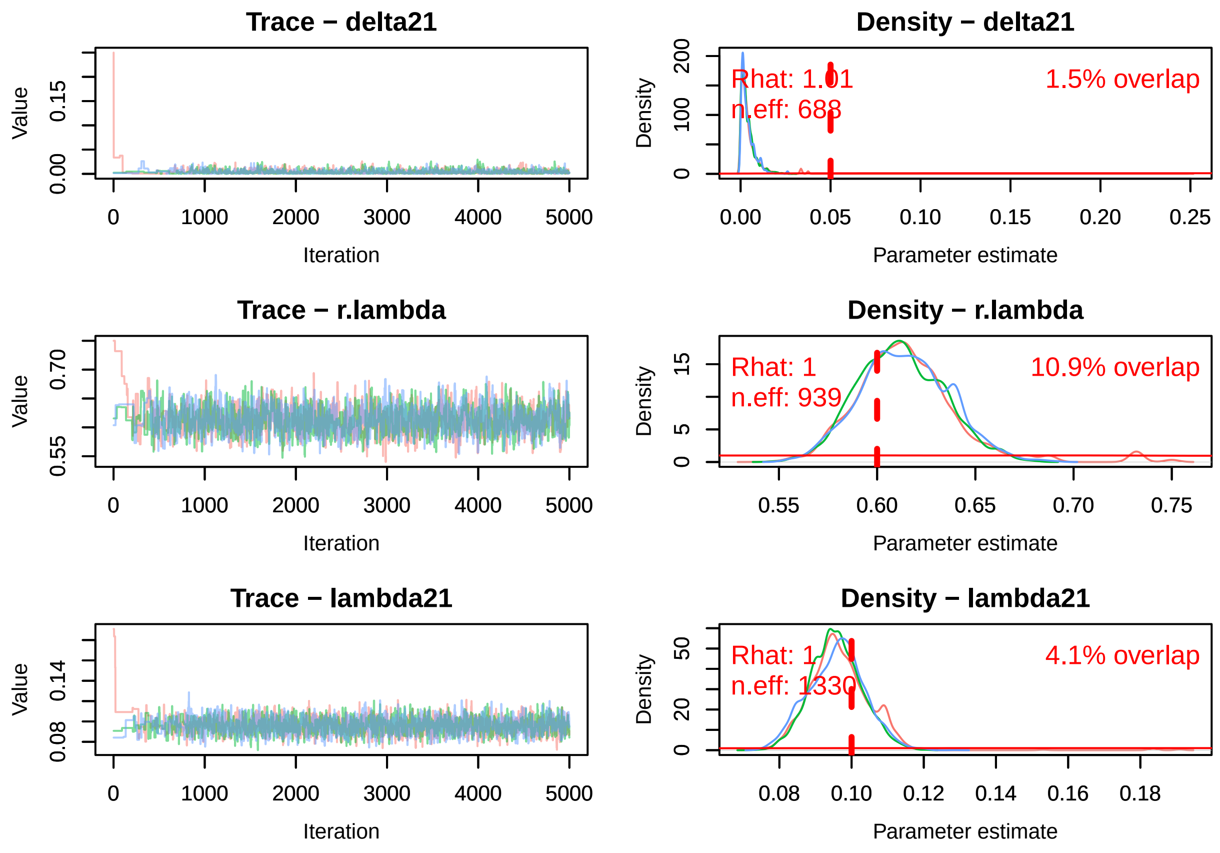
Traceplots and summary statistics

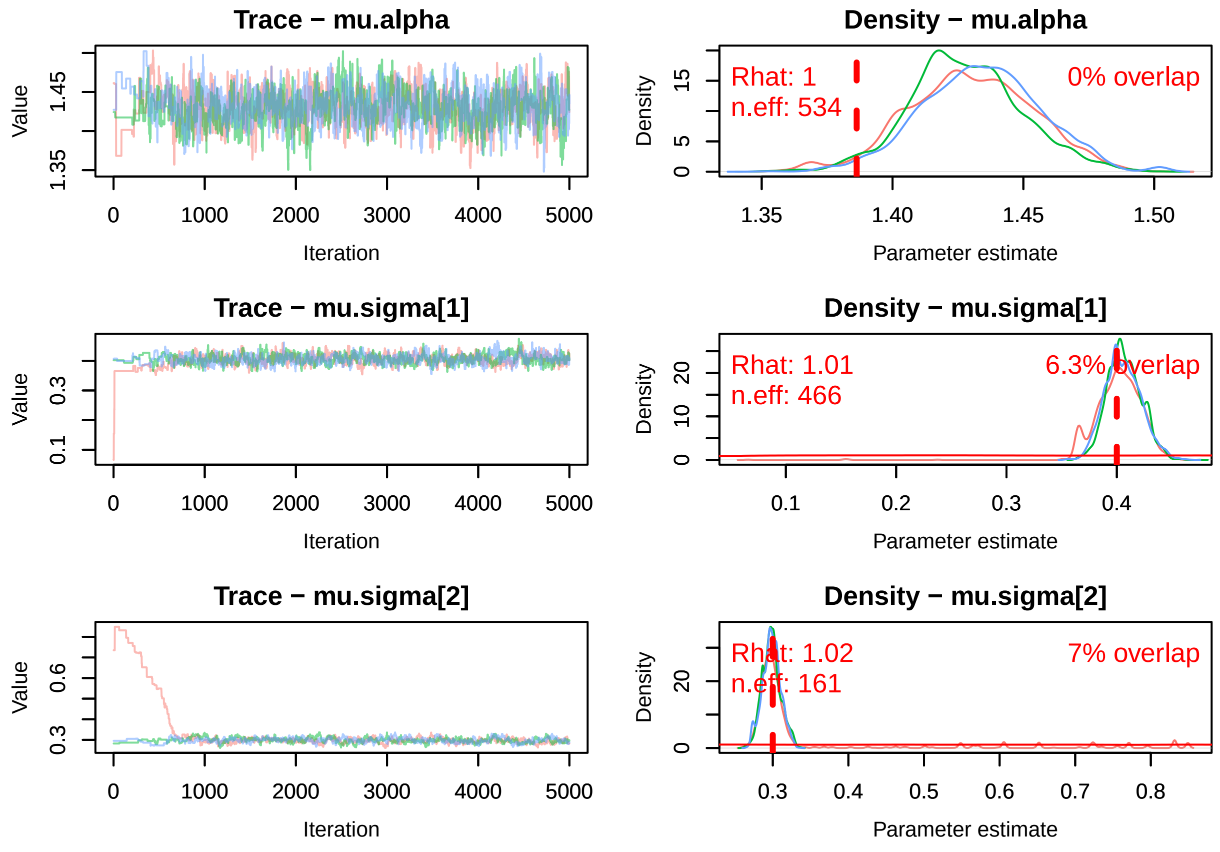
We see that parameters mix fairly well and that convergence is achieved quickly (Youngflesh, 2018), but that parameters are not recovered properly. Additionally, the recapture and state transition probabilities have considerable percent overlap between the prior and posterior distributions (PPO), suggestive of identifiability issues (Gimenez et al., 2009). The red horizontal lines in the density plots are Beta(1, 1) priors, which were used for all probabilities, and the red dashed lines mark the simulation input parameter values.

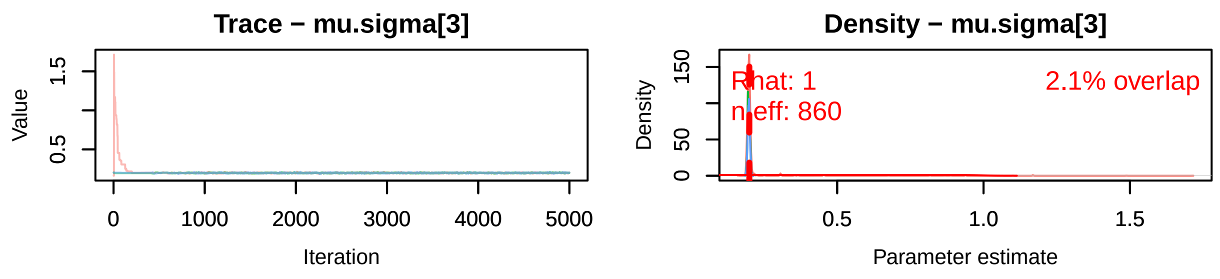
```
# Parameters
params <- c(pi, phi1, phi2.alpha, phi2.beta, psi12, psi21, p1, p2,
            r.delta, delta21, r.lambda, lambda21, log(mu), mu.sigma)

# Traceplots and summary
library(MCMCvis)
MCMCtrace(MEsamples, params = MEmons, Rhat = T, n.eff = T, ind = T, pdf = F,
          gvals = params, priors = rbeta(nchains * niter, 1, 1))
```




Trace − phi2.beta
Value
Iteration
Density − phi2.beta
Density
Parameter estimate
Rhat: 1
n.eff: 292
0.9% overlap
Trace − psi12.alpha
Density − psi12.alpha
Rhat: 1.03
n.eff: 168
29.3% overlap
Trace − psi21.alpha
Density − psi21.alpha
Rhat: 1.06
n.eff: 79
28.3% overlap
Trace − p1.alpha
Density − p1.alpha
Rhat: 1.03
n.eff: 108
51.6% overlap
Trace − p2.alpha
Density − p2.alpha
Rhat: 1.06
n.eff: 78
31.8% overlap
Trace − r.delta
Density − r.delta
Rhat: 1
n.eff: 1878
2.9% overlap


Trace − delta21
Density − delta21
Rhat: 1.01
n.eff: 688
1.5% overlap
Trace − r.lambda
Density − r.lambda
Rhat: 1
n.eff: 939
10.9% overlap
Trace − lambda21
Density − lambda21
Rhat: 1
n.eff: 1330
4.1% overlap
Value
Iteration
Density
Parameter estimate


Trace − mu.alpha
Density − mu.alpha
Rhat: 1
n.eff: 534
0% overlap
Trace − mu.sigma[1]
Density − mu.sigma[1]
Rhat: 1.01
n.eff: 466
6.3% overlap
Trace − mu.sigma[2]
Density − mu.sigma[2]
Rhat: 1.02
n.eff: 161
7% overlap
Value
Iteration
Density
Parameter estimate

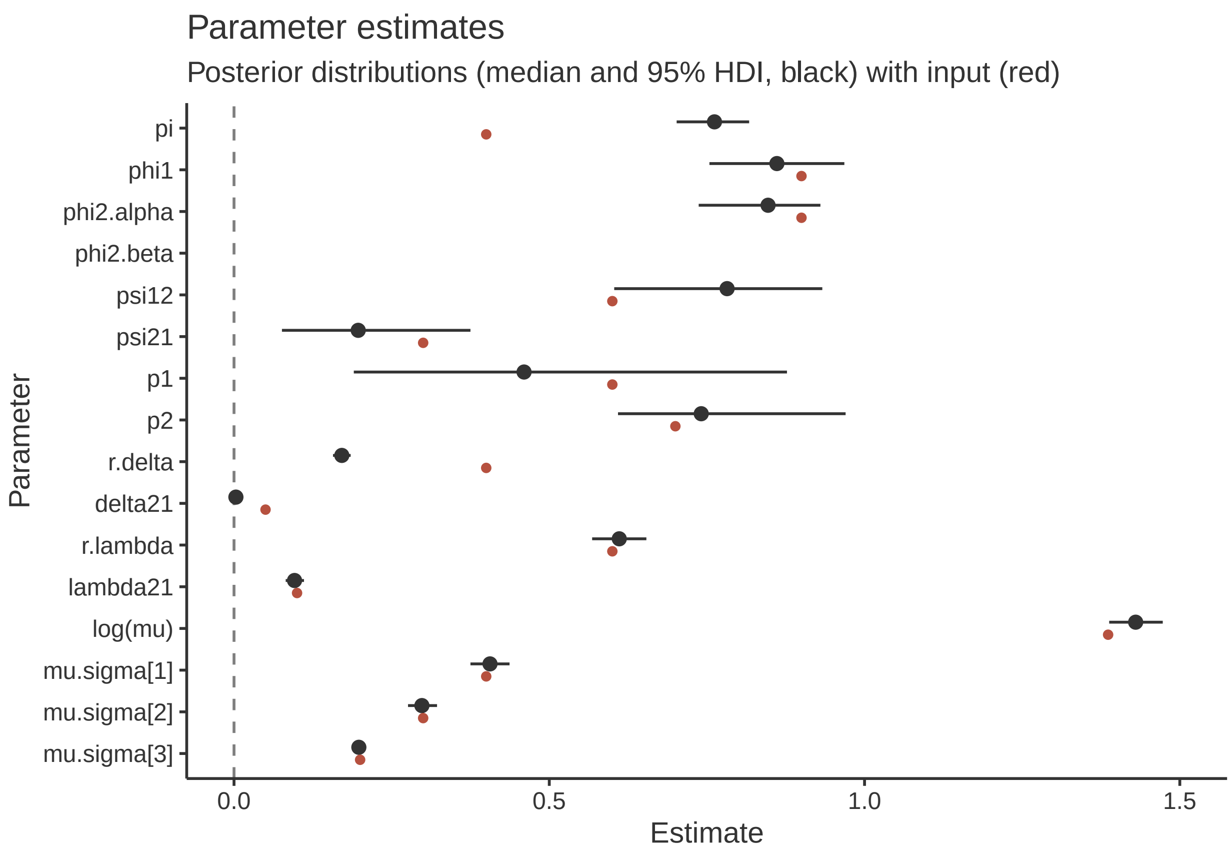


```
print(MEsum <- MCMCsummary(lapply(MEsamples, function(x) x[-(1:nburnin),]),
                           params = MEmons, HPD = T))
```

```
                 mean      sd    95%_HPDL 95%_HPDU Rhat n.eff
pi.alpha      0.76253 0.02989  0.70153865   0.8174 1.00  2942
phi1.alpha    0.86005 0.05593  0.75432472   0.9682 1.03   800
phi2.alpha    0.84035 0.05169  0.73734171   0.9303 1.00   129
phi2.beta    -0.20238 0.07380 -0.35513095  -0.0696 1.01   137
psi12.alpha   0.77385 0.08979  0.60327625   0.9328 1.06   153
psi21.alpha   0.21155 0.08730  0.07639695   0.3755 1.13    65
p1.alpha      0.49729 0.19312  0.19008609   0.8774 1.09    88
p2.alpha      0.76361 0.10200  0.60924934   0.9701 1.13    58
r.delta       0.17110 0.00704  0.15740498   0.1851 1.01  2025
delta21       0.00404 0.00392  0.00000126   0.0119 1.02  1191
r.lambda      0.61219 0.02185  0.56844550   0.6538 1.00  1161
lambda21      0.09592 0.00745  0.08158176   0.1111 1.00  1606
mu.alpha      1.43032 0.02188  1.38762926   1.4731 1.00   531
mu.sigma[1]   0.40673 0.01607  0.37477918   0.4374 1.01   448
mu.sigma[2]   0.29789 0.01175  0.27572806   0.3217 1.02   247
mu.sigma[3]   0.19824 0.00405  0.19089752   0.2063 1.00  1305
```

Posterior distributions

The input values of the simulation are not recovered within the posterior distributions of model parameters. We summarized posteriors as medians and 95% highest density intervals (HDIs) after omitting the first 1000 MCMC samples as burn-in.


Parameter estimates
Posterior distributions (median and 95% HDI, black) with input (red)
Parameter
pi
phi1
phi2.alpha
phi2.beta
psi12
psi21
p1
p2
r.delta
delta21
r.lambda
lambda21
log(mu)
mu.sigma[1]
mu.sigma[2]
mu.sigma[3]
0.0
0.5
1.0
1.5
Estimate

References

Arnason, A. N. (1973). The estimation of population size, migration rates and survival in a stratified population. *Researches on Population Ecology*, *15*(2), 1–8. https://doi.org/10.1007/BF02510705

Arnason, A. N. (1972). Parameter estimates from mark-recapture experiments on two populations subject to migration and death. *Researches on Population Ecology*, *13*(2), 97–113.

de Valpine, P., Turek, D., Paciorek, C. J., Anderson-Bergman, C., Lang, D. T., & Bodik, R. (2017). Programming with models: Writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*, *26*(2), 403–413. https://doi.org/10.1080/10618600.2016.1172487

Gimenez, O., Morgan, B. J. T., & Brooks, S. P. (2009). Weak identifiability in models for mark-recapture-recovery data. In D. L. Thomson, E. G. Cooch, & M. J. Conroy (Eds.), *Modeling Demographic Processes in Marked Populations* (pp. 8–48). Springer. https://doi.org/10.1007/978-0-387-78151-8_48

Kéry, M., & Schaub, M. (2012). Chapter 9. Estimation of survival and movement from capture-recapture data using multistate models. In *Bayesian Population Analysis Using WinBUGS: A Hierarchical Perspective* (pp. 264–315). Academic Press.

Lachish, S., Gopalaswamy, A. M., Knowles, S. C. L., & Sheldon, B. C. (2012). Site-occupancy modelling as a novel framework for assessing test sensitivity and estimating wildlife disease prevalence from imperfect diagnostic tests. *Methods in Ecology and Evolution*, *3*(2), 339–348. https://doi.org/10.1111/j.2041-210X.2011.00156.x

MacKenzie, D., Nichols, J. D., Lachman, G. B., Droege, S., Royle, J. A., & Langtimm, C. A. (2002). Estimating site occupancy rates when detection probabilities are less than one. *Ecology*, *83*(8), 2248–2255. https://doi.org/10.1890/0012-9658(2002)083%5B2248:ESORWD%5D2.0.CO;2

Papaspiliopoulos, O., & Roberts, G. O. (2003). Non-centered parameterisations for hierarchical models and data augmentation. *Bayesian Statistics*, *7*, 307–326.

Pradel, R. (2005). Multievent: An extension of multistate capture-recapture models to uncertain states. *Biometrics*, *61*(2), 442–447. https://doi.org/10.1111/j.1541-0420.2005.00318.x

R Core Team. (2022). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing.

Royle, J. A., & Nichols, J. D. (2003). Estimating abundance from repeated presence-absence data or point counts. *Ecology*, *84*(3), 777–790. https://doi.org/10.1890/0012-9658(2003)084%5B0777:EAFRPA%5D2.0.CO;2

Schwarz, C. J., Schweigert, J. F., & Arnason, A. N. (1993). Estimating migration rates using tag-recovery data. *Biometrics*, *49*(1), 177–193. https://doi.org/10.2307/2532612

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis.* Springer-Verlag New York.

Youngflesh, C. (2018). MCMCvis: Tools to visualize, manipulate, and summarize MCMC output. *Journal of Open Source Software*, *3*(24), 640. https://doi.org/10.21105/joss.00640