Appendix S3

Supplementary model code for:

Simulating telemetry studies that estimate component mortality rates of imperiled juvenile salmonids

Authors: E. M. Greenheck, M. Peterson, T. Pilger, M. A. Djokic, J. Eschenroeder, T. R. Nelson

Table of Contents

Installing and loading packages, clearing the R workspace

```r
1 # Clear workspace
2 rm(list = ls())
3
4 # Required packages:
5 packs <- c("R2OpenBUGS", "jagsUI", "tidyverse", "purrr")
6
7 # Install packages install.packages(packs, repos =
8 # 'https://cloud.r-project.org')
9
10 # Load packages
11 lapply(packs, library, character.only = TRUE)
12
13 # Setting a working directory
14
15 # setwd('yourworkingdirectory')
```

Model Scenario 4: High detection probability (passive acoustic telemetry array and active tracking); four-state and four-observation multistate mark-recapture model (MSMR)

Section 1: Creating the simulation data

See Appendix S1 (Model Scenarios 1 & 2) for information regarding the MSMRs. The following code demonstrates how to increase the number of model states from three observation states and three true states to four observation states and four true states.

The below code does not change from Appendix S1:

```r
16 # Defining our number of reaches (columns)
17 n.reach <- 6
18 n.reach.obs <- n.reach + 1
19
20 # Defining our number of individuals to be modeled
21 ind_sims <- c(50, 100, 150, 500, 1000)  # our individual scenarios
22 nind_sims <- as.numeric(length(ind_sims))  # the number of individual scenarios
```

Setting up our state process

Here, we increase our number of states (n.states, Line 24) to 4, and we estimate fish predation (Pf) and avian predation (Pa) instead of a single predation source (P; see Appendix S1). See Lines 26-36.

We assume that we detect fish predation with predation-detection acoustic tags (PDATs) and detect avian predation with combined-acoustic radio tags (CARTS). Thus, our Pf value is the same as it was in Model Scenarios 1 & 2, where predation that is observed in aquatic environments is attributed to fish predation. Using the CARTS, we are able to partition our unobserved state into avian predation, Pa, and unknown mortality, M.

We also increase our PSI.STATE matrix from a 3x3 matrix to a 4x4 matrix (Lines 57-62)

```r
23 # Defining our number of model states
24 n.states <- 4
25
26 # Defining our estimates for fish predation mortality (Pf)
27
28 # Here, Pf is 0 in the first reach, reach 2 has high Pf, Pf is
29 # moderate in reaches 3 and 5, and low in reaches 4 and 6
30 Pf <- c(1e-05, 0.26, 0.05, 0.02, 0.05, 0.02)  # instantaneous Pf, sum = 0.4
31
32 # Defining our estimates for avian predation mortality (Pa)
33
34 # Here, Pa is 0 in the first reach, elevated in reach 4, moderate
35 # in reaches 3 and 5, and low in reaches 2 and 6
36 Pa <- c(1e-05, 0.01, 0.025, 0.1, 0.025, 0.01)  # instantaneous Pa, sum = 0.17
37
38 # Defining our estimates for unknown mortality (M)
39
40 # Here, M is 0 in the first reach, slightly elevated in reach 4,
41 # moderate in reaches 3 and 5, and low in reaches 2 and 6
42 M <- c(1e-05, 0.01, 0.025, 0.06, 0.025, 0.01)  # instantaneous M, sum = 0.13
43
44 # Calculating reach-specific total mortality (Z) and survival (S)
45 Z <- Pf + Pa + M  # reach-specific total instantaneous mortality
46 S <- exp(-Z)  # reach-specific discrete survival
47
```

```
48 # Calculating discrete and/or total estimates for all reaches
49 Z_D <- sum(Z)  # total instantaneous mortality
50 A_D <- 1 - exp(-Z_D)  # total discrete mortality; 0.503; roughly 50% of fish die
51 S_D <- exp(-sum(Z))  # total discrete survival; 0.497; roughly 50% of fish survive
52 Pf_D <- (sum(Pf) * A_D)/Z_D  # total discrete fish predation probability for all reaches
53 Pa_D <- (sum(Pa) * A_D)/Z_D  # total discrete avian predation probability for all reaches
54 M_D <- (sum(M) * A_D)/Z_D  # total discrete unknown mortality for all reaches
55
56 # Creating the state process matrix
57 PSI.STATE <- array(NA, dim = c(n.states, n.states, n.reach))
58 for (r in 1:(n.reach)) {
59     PSI.STATE[, , r] <- matrix(c(S[r], Pf[r] * (1 - S[r])/Z[r], Pa[r] *
60         (1 - S[r])/Z[r], M[r] * (1 - S[r])/Z[r], 0, 1, 0, 0, 0, 0, 1,
61         0, 0, 0, 0, 1), nrow = n.states, byrow = TRUE)
62 }  # r
63
64 # Printing the first two reaches for PSI.STATE
65 PSI.STATE[, , 1:2]
```

```
## , , 1
##
##         [,1]        [,2]        [,3]        [,4]
## [1,] 0.99997 9.99985e-06 9.99985e-06 9.99985e-06
## [2,] 0.00000 1.00000e+00 0.00000e+00 0.00000e+00
## [3,] 0.00000 0.00000e+00 1.00000e+00 0.00000e+00
## [4,] 0.00000 0.00000e+00 0.00000e+00 1.00000e+00
##
## , , 2
##
##           [,1]      [,2]        [,3]        [,4]
## [1,] 0.7557837 0.2267722 0.008722009 0.008722009
## [2,] 0.0000000 1.0000000 0.000000000 0.000000000
## [3,] 0.0000000 0.0000000 1.000000000 0.000000000
## [4,] 0.0000000 0.0000000 0.000000000 1.000000000
```

PSI.STATE should now produce an array of 4x4 matrices, and each matrix denotes the probability of a fish transitioning between true states.

Setting up our observation process

Here, PSI.OBS is converted from a 3x3 matrix to a 4x4 matrix. We edit our number of observations (n.obs) to 4 (Line 67) and update our PSI.OBS matrix (Lines 73-77).

```
66 # Defining our number of observation states
67 n.obs <- 4
68
69 # Defining our reach-specific detection probability (p)
70 p <- c(1, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9)
71
72 # Creating the observation process matrix
73 PSI.OBS <- array(NA, dim = c(n.states, n.obs, n.reach.obs))
74 for (r in 1:n.reach.obs) {
75     PSI.OBS[, , r] <- matrix(c(p[r], 0, 0, 1 - p[r], 0, 1, 0, 0, 0, 0,
76         1, 0, 0, 0, 0, 1), nrow = n.states, byrow = TRUE)
77 }  # r
```

```r

# Printing the first two reaches for PSI.OBS
PSI.OBS[, , 1:2]
```

```
## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]  0.9    0    0  0.1
## [2,]  0.0    1    0  0.0
## [3,]  0.0    0    1  0.0
## [4,]  0.0    0    0  1.0
```

PSI.STATE should now produce an array of 4x4 matrices, and each matrix denotes the probability of detecting a fish in each state.

Define the function to simulate multistate capture-recapture data

The below code does not change from Appendix S1:

```r
simul.ms <- function(PSI.STATE, PSI.OBS, nind, unobservable = NA) {
    n.reach.obs <- dim(PSI.STATE)[3] + 1
    CH <- CH.TRUE <- matrix(NA, ncol = n.reach.obs, nrow = nind)
    for (i in 1:nind) {
        CH[, 1] <- CH.TRUE[, 1] <- 1  # first column all fish are released alive
        for (r in (2:n.reach.obs)) {
            # Multinomial trials for state transitions
            state <- which(rmultinom(1, 1, PSI.STATE[CH.TRUE[i, r - 1],
                , r - 1]) == 1)
            # which() = which state gets the 1 random draw at reach
            # r given state at r-1
            CH.TRUE[i, r] <- state  # then fills in true states
            # Multinomial trials for observation process
            event <- which(rmultinom(1, 1, PSI.OBS[CH.TRUE[i, r], , r]) ==
                1)
            # which observation gets the 1 random draw, given true
            # reach r state.
            CH[i, r] <- event
        }  # r
    }  # i
    return(list(CH = CH, CH.TRUE))
}
```

Setting our number of simulation repetitions

The below code does not change from Appendix S1:

```
103 # 100 is a good number of repetitions, and has been used in other
104 # studies (e.g., Hightower & Harris 2017)
105 Reps <- 100
```

Data setup check code

This code ensures that the vector lengths of the simulation data are equal - the state-process code is edited to include Pf and Pa (Line 115)

```
106 # Data set-up check
107
108 # For our number of individuals
109 if (nind_sims != length(ind_sims)) {
110     print("Length mismatch for number of individuals")
111 } else {
112     print("individual setup OK")
113 }
```

```
## [1] "individual setup OK"
```

```
114 # For the state-process
115 if (n.reach != length(Pf) | n.reach != length(Pa) | n.reach != length(M)) {
116     print("Length mismatch for state estimates")
117 } else {
118     print("estimates setup OK")
119 }
```

```
## [1] "estimates setup OK"
```

```
120 # For p
121 if (n.reach.obs != length(p)) {
122     print("Length mismatch for p")
123 } else {
124     print("p setup OK")
125 }
```

```
## [1] "p setup OK"
```

Section 2: Four-observation four-state capture-recapture model

The four-state capture-recapture model

The below model code estimates instantaneous reach-specific fish predation (Pf), avian predation (Pf), unknown (M), and total mortality (Z), as well as reach-specific survival (S), using the capture histories simulated above and the probability of an individual transitioning from the alive state to either mortality source within each reach. Additionally, total study wide discrete estimates of fish predation (Pf_D), avian predation (Pa_D), unknown mortality (M_D), and survival (S) are derived. The model has uninformative priors and was developed using Kéry & Schaub 2012 as well as Hightower and Harris 2017. Code annotation describe each aspect of the model.

Here, we add additional priors for the added state, add additional derived sums, and expand our state-transition (ps) and observation matrices (po). The detection probability setup (Lines 160-164) and likelihood process (Lines 216-228) are not edited when an additional state is added.

```
# Model code for JAGS
mod = function() {
    # Priors

    # We are estimating our parameters on the reach-specific level
    # (r), e.g., our Pf estimate is Pf[r]. In models where
    # individual-level (i) indexing is necessary, this can be
    # expanded to Pf[r,i].

    # We use an uninformative prior for Pf, Pa, and M on the
    # natural log scale.  We do this instead of the response scale
    # (e.g., dunif(0,1)) because on the response scale,
    # reach-specific mortality is overestimated when values are
    # very low (e.g. 0).  We recommend playing around with your
    # priors, even if uninformative.
    for (r in 1:(Reaches)) {
        ln.Pf[r] ~ dunif(-10, 1)  # uninformative prior for Pf on the natural log scale
        Pf[r] <- exp(ln.Pf[r])  # transforming the Pf prior to the response scale
        ln.Pa[r] ~ dunif(-10, 1)  # uninformative prior for Pa on the natural log scale
        Pa[r] <- exp(ln.Pa[r])  # transforming the Pa prior to the response scale
        ln.M[r] ~ dunif(-10, 1)  # uninformative prior for M on the natural log scale
        M[r] <- exp(ln.M[r])  # transforming the M prior to the response scale
        Z[r] <- M[r] + Pf[r] + Pa[r]  # total instantaneous reach-specific mortality
        S[r] <- exp(-Z[r])  # total reach-specific survival
    }  # r

    # Derived sums of discrete mortality for reaches 1-6
    Z_D <- sum(Z[1:(Reaches)])  # total instantaneous mortality
    S_D <- exp(-Z_D)  # total discrete survival
    A_D <- 1 - exp(-Z_D)  # total discrete mortality
    Pf_D <- (sum(Pf[1:(Reaches)]) * A_D)/Z_D  # discrete total fish predation
    Pa_D <- (sum(Pa[1:(Reaches)]) * A_D)/Z_D  # discrete total avian predation
    M_D <- (sum(M[1:(Reaches)]) * A_D)/Z_D  # discrete total other mortality

    # Detection probability
    p[1] <- 1  # we assume we detect 100% of fish in reach 1, so p = 100% (or 1)
    for (r in 2:(Reaches + 1)) {
        p[r] ~ dunif(0, 1)  # uninformative prior for p on the response scale
    }  # r

    # Define state-transition (ps) and observation matrices (po)
    for (i in 1:nFish) {
        # Define probabilities of State (r+1) given State (r).
        # First index is state at reach r, next is state at r+1.

        # The first[i]:(last[i]-1) allows for staggered entry into
        # the system.  In this scenario, a staggered entry would be
        # releasing fish in reach 0 and reach 1 (for example).  We
        # release all fish in reach 0 in this study, but we retain
        # this code for flexibility in model application.
        for (r in first[i]:(last[i] - 1)) {
            ps[1, i, r, 1] <- S[r]  # alive fish remains alive
            ps[1, i, r, 2] <- Pf[r] * (1 - S[r])/Z[r]  # alive fish experiences fish pred.
            ps[1, i, r, 3] <- Pa[r] * (1 - S[r])/Z[r]  # alive fish experiences avian pred.
            ps[1, i, r, 4] <- M[r] * (1 - S[r])/Z[r]  # alive fish dies of unknown mort.
            ps[2, i, r, 1] <- 0
            ps[2, i, r, 2] <- 1
            ps[2, i, r, 3] <- 0
            ps[2, i, r, 4] <- 0
            ps[3, i, r, 1] <- 0
            ps[3, i, r, 2] <- 0
            ps[3, i, r, 3] <- 1
            ps[3, i, r, 4] <- 0
            ps[4, i, r, 1] <- 0
            ps[4, i, r, 2] <- 0
            ps[4, i, r, 3] <- 0
            ps[4, i, r, 4] <- 1
        }  # r
        for (r in first[i]:(last[i])) {
            # Define probabilities of Observed (r) given State (r).
            # First index is state, last index is observed
            po[1, i, r, 1] <- p[r]  # alive fish is detected alive
            po[1, i, r, 2] <- 0
            po[1, i, r, 3] <- 0
            po[1, i, r, 4] <- 1 - p[r]  # alive fish is not detected
            po[2, i, r, 1] <- 0
            po[2, i, r, 2] <- 1  # fish is a '2' (fish predation) it stays a 2
            po[2, i, r, 3] <- 0
            po[2, i, r, 4] <- 0
            po[3, i, r, 1] <- 0
            po[3, i, r, 2] <- 0
            po[3, i, r, 3] <- 1  # fish is a '3' (avian predation) it stays a 3
            po[3, i, r, 4] <- 0
            po[4, i, r, 1] <- 0
            po[4, i, r, 2] <- 0
            po[4, i, r, 3] <- 0
            po[4, i, r, 4] <- 1  # fish is a 'f' (unknown mortality) it stays a 4
        }  # r
    }  # i

    # Likelihood process
    for (i in 1:nFish) {
        z[i, first[i]] <- 1  # individuals are alive at first occasion in study
        for (r in (first[i] + 1):last[i]) {
            z[i, r] ~ dcat(ps[z[i, r - 1], i, r - 1, ])
            # State process: draw State (r) given State (r-1)
        }  # r
        for (r in first[i]:last[i]) {
            y[i, r] ~ dcat(po[z[i, r], i, r, ])
            # Observation process: draw Observed (r) given State
            # (r)
        }  # r
    }  # i
}  # final bracket

# Write model to a text file
model.file = "model4.txt"
R2OpenBUGS::write.model(mod, model.file)
```

Setting up our model data and storing the estimates

Below, we create empty vectors to store our model outputs for each repetition (Reps) and each number of individuals simulated (nind_sims). We now add an additional empty vector for our added state.

```
234 # Vector to record the start & end time of each model run
235 starttime <- vector("list", nind_sims)
236 endtime <- vector("list", nind_sims)
237
238 # Setting up our model data and writing a loop to iterate through
239 # our Reps and nind_sims
240 Ests_D <- vector("list", nind_sims)  # empty vector for all discrete estimates
241 Ests_Pf <- vector("list", nind_sims)  # empty vector to store reach-specific Pf
242 Ests_Pa <- vector("list", nind_sims)  # empty vector to store reach-specific Pa
243 Ests_M <- vector("list", nind_sims)  # empty vector to store reach-specific M
244 Ests_S <- vector("list", nind_sims)  # empty vector to store reach-specific S
245 det_p <- vector("list", nind_sims)  # empty vector to store our detection probabilities (p)
246 n_state_recs <- vector("list", nind_sims)
247
248 for (sim in 1:nind_sims) {
249     starttime[[sim]] <- vector("list", Reps)
250     endtime[[sim]] <- vector("list", Reps)
251     Ests_D[[sim]] <- vector("list", Reps)
252     Ests_Pf[[sim]] <- vector("list", Reps)
253     Ests_Pa[[sim]] <- vector("list", Reps)
254     Ests_M[[sim]] <- vector("list", Reps)
255     Ests_S[[sim]] <- vector("list", Reps)
256     det_p[[sim]] <- vector("list", Reps)
257     n_state_recs[[sim]] <- vector("list", Reps)
258 }  #sim
```

Main simulation loop and extracting results for each simulation and repetition

Finally, the model will estimate survival (S), fish predation (Pf), avian predation (Pa) and unknown mortality (M) using our simulated capture histories. The empty vectors created above will store all model outputs. We will need to edit multiple aspects of this code to account for an additional state, including storage of the reach-specific number of individuals in each state (Lines 279-294), initial function, m.init.z (Lines 305-338), jags.inits (Lines 340-344), monitored parameters (Lines 352-353), and our storage of the additional parameter (Lines 396-434).

```
259 for (sim in 1:nind_sims) {
260     nind <- ind_sims[sim]
261     for (rep in 1:Reps) {
262         simCH <- simul.ms(PSI.STATE, PSI.OBS, nind)
263         CH <- simCH$CH
264         CH.TRUE <- simCH[[2]]
265         y = CH
266         nFish = dim(CH)[1]
267         Reaches = dim(CH)[2] - 1
268         first <- numeric()
269         for (i in 1:dim(y)[1]) {
270             first[i] <- min(which(y[i, ] != 0))
271         }  # i
272         last <- numeric()
273         for (i in 1:dim(y)[1]) {
274             last[i] <- max(which(y[i, ] != 0))
275         }  # i
```

```r
        f <- first
        l <- last

        # Initialize reach-state count matrix Rows: periods
        # (reaches), Cols: states (1,2,3,4)
        n_state_record <- matrix(NA, nrow = ncol(CH.TRUE), ncol = 4,
            dimnames = list(paste0("R", 1:ncol(CH.TRUE)), c("S", "Pf",
                "Pa", "M")))

        for (r in 1:ncol(CH.TRUE)) {
            n_state_record[r, 1] <- sum(CH.TRUE[, r] == 1, na.rm = TRUE)  # alive (survival)
            n_state_record[r, 2] <- sum(CH.TRUE[, r] == 2, na.rm = TRUE)  # fish predation
            n_state_record[r, 3] <- sum(CH.TRUE[, r] == 3, na.rm = TRUE)  # avian predation
            n_state_record[r, 4] <- sum(CH.TRUE[, r] == 4, na.rm = TRUE)  # unknown mort.
            # if you have NA, they won't be counted
        }  # r

        # Store in list
        n_state_recs[[sim]][[rep]] <- n_state_record

        jags.data <- list(y = y, nFish = nFish, first = first, last = last,
            Reaches = Reaches)

        # Initial function - this aids in model speed by creating
        # rules around how the unobserved state should be
        # 'back-filled'. For example, if a fish has a capture
        # history of 1-3-3-1-1-1, we know the fish in reaches 2 and
        # 3 is actually alive, so this function fills the 3s as 1s
        # and dings our detection probability in reaches 2 and 3.
        ms.init.z <- function(y, f) {
            y.iv <- y
            y.iv[y.iv == 4] <- NA  # change this to the unobserved state

            for (i in 1:nrow(y.iv)) {
                if (max(y.iv[i, ], na.rm = TRUE) == 1) {
                  y.iv[i, (f[i] + 1):l[i]] <- 1
                }
                # not detected dead so initialize as alive

                if (max(y.iv[i, ], na.rm = TRUE) == 2)
                  {
                    m <- min(which(y.iv[i, ] == 2))
                    y.iv[i, f[i]:(m - 1)] <- 1
                    # fish predation not detected so initialize as
                    # alive
                    y.iv[i, m:l[i]] <- 2
                  }  # fish predation detected, terminal until end of CH

                if (max(y.iv[i, ], na.rm = TRUE) == 3)
                  {
                    m <- min(which(y.iv[i, ] == 3))
                    y.iv[i, f[i]:(m - 1)] <- 1
                    # avian predation not detected so initialize as
                    # alive
                    y.iv[i, m:l[i]] <- 3
                  }  # avian predation detected, terminal until end of CH
            }  # i

            for (i in 1:dim(y.iv)[1]) {
                y.iv[i, 1:f[i]] <- NA
            }  # i
            return(y.iv)
        }  # final initial function contains 1s, 2s, and 3s

        jags.inits <- function() {
            list(ln.M = runif(Reaches, -10, 1), ln.Pf = runif(Reaches,
                -10, 1), ln.Pa = runif(Reaches, -10, 1), z = ms.init.z(y,
                f))
        }

        # Parameters to monitor during model run

        # For this model, we are interested in our final estimates
        # for Pf, Pa, M, and S (Pf_D, Pa_D, M_D, S_D), our
        # reach-specific estimates (Pf, Pa, M, S), and detection
        # probability (p).
        params <- c("Pf_D", "Pa_D", "M_D", "S_D", "p", "Pf", "Pa", "M",
            "S")

        # Run model in JAGS

        # NOTE: R2Jags and jagsUI both use a function called
        # 'autojags'. We used the jagsUI package,the syntax will
        # give an error if the R2Jags package is used instead.
        starttime[[sim]][[rep]] <- Sys.time()  # record start time
        jagsfit <- jagsUI::autojags(data = jags.data, inits = jags.inits,
            parameters.to.save = params, model.file, n.chains = 3, n.adapt = NULL,
            iter.increment = 3000, n.burnin = 10000, n.thin = 1, save.all.iter = FALSE,
            modules = c("glm"), parallel = TRUE, DIC = TRUE, store.data = FALSE,
            codaOnly = FALSE, bugs.format = FALSE, Rhat.limit = 1.1,
            max.iter = 12000, verbose = TRUE)
        endtime[[sim]][[rep]] <- Sys.time()  # record end time

        # For quantiles & parameter extracts
        est_names <- c("q2.5", "q25", "q50", "q75", "q97.5")
        param_names_D <- str_subset(params, "_D$")

        Est_D <- array(NA, dim = c(length(est_names), length(param_names_D)),
            dimnames = list(est_names, param_names_D))

        for (param in param_names_D) {
            for (quant in est_names) {
                Est_D[quant, param] <- jagsfit[[quant]][[param]]
            }  # quant
        }  # param

        Ests_D[[sim]][[rep]] <- Est_D

        # For p extract R0 is a place holder for the transition
        # from release to reach 2
        reaches <- c("R0", "R1", "R2", "R3", "R4", "R5", "R6")
        det_p_ <- matrix(NA, nrow = length(est_names), ncol = n.reach.obs,
```

```r
            dimnames = list(est_names, reaches))

        for (quant in est_names) {
            det_p_[quant, ] <- jagsfit[[quant]]$p
        }  # quant

        det_p[[sim]][[rep]] <- det_p_

        # For Pf extract
        Est_Pf <- matrix(NA, nrow = length(est_names), ncol = n.reach,
            dimnames = list(est_names, reaches[-1]))

        for (quant in est_names) {
            Est_Pf[quant, ] <- jagsfit[[quant]]$Pf
        }  # quant

        Ests_Pf[[sim]][[rep]] <- Est_Pf

        # For Pa extract
        Est_Pa <- matrix(NA, nrow = length(est_names), ncol = n.reach,
            dimnames = list(est_names, reaches[-1]))

        for (quant in est_names) {
            Est_Pa[quant, ] <- jagsfit[[quant]]$Pa
        }  # quant

        Ests_Pa[[sim]][[rep]] <- Est_Pa

        # For M extract
        Est_M <- matrix(NA, nrow = length(est_names), ncol = n.reach,
            dimnames = list(est_names, reaches[-1]))

        for (quant in est_names) {
            Est_M[quant, ] <- jagsfit[[quant]]$M
        }  # quant

        Ests_M[[sim]][[rep]] <- Est_M

        # For S extract
        Est_S <- matrix(NA, nrow = length(est_names), ncol = n.reach,
            dimnames = list(est_names, reaches[-1]))

        for (quant in est_names) {
            Est_S[quant, ] <- jagsfit[[quant]]$S
        }  # quant

        Ests_S[[sim]][[rep]] <- Est_S

    }  # rep
}  # sim

save.image(file = "model4.RData")
```

Converting outputs to exportable flat files (.csv).

Here, we need to adapt our code to obtain the true values (Lines 497-499), ests_list (Line 517), and trueests_p (Lines 536-538).

```r
440 # The individual simulations code as 1-length(nind_sims); so this
441 # codes them back into the actual number of individuals
442 nind_ <- tibble(sim = c(1:length(ind_sims)), ind_sims)
443
444 # Storing run times for each model
445 i <- 1
446 starttime_df <- list()
447 for (r in 1:Reps) {
448     for (s in 1:nind_sims) {
449         df <- tibble(starttime = as.POSIXct(starttime[[s]][[r]])) %>%
450             mutate(ind_sims = ind_sims[s], rep = r)
451         starttime_df[[i]] <- df
452         i <- i + 1
453     }  # s
454 }  # r
455
456 starttime_df <- do.call("rbind", starttime_df)
457
458 i <- 1
459 endtime_df <- list()
460 for (r in 1:Reps) {
461     for (s in 1:nind_sims) {
462         df <- tibble(endtime = as.POSIXct(endtime[[s]][[r]])) %>%
463             mutate(ind_sims = ind_sims[s], rep = r)
464         endtime_df[[i]] <- df
465         i <- i + 1
466     }  # s
467 }  # r
468
469 endtime_df <- do.call("rbind", endtime_df)
470 runtimes <- full_join(starttime_df, endtime_df)
471 runtimes <- runtimes %>%
472     mutate(runtime_seconds = as.numeric(endtime - starttime))
473
474 # Unpacking the reach-specific number of individuals in each state
475 N_State_recs_df <- purrr::map2_dfr(n_state_recs, seq_along(n_state_recs),
476     function(s_list, s) {
477         purrr::map2_dfr(s_list, seq_along(s_list), function(est_array,
478             r) {
479             as.data.frame(as.table(est_array)) %>%
480                 rename(reach = Var1, parameter = Var2, N = Freq) %>%
481                 mutate(sim = s, rep = r)
482         })
483     }) %>%
484     left_join(nind_)
485
486 # Unpacking the model estimates and calculating percent relative
487 # error (MRE) and the coefficient of variation (CV)
488 Ests_df <- purrr::map2_dfr(Ests_D, seq_along(Ests_D), function(s_list,
489     s) {
490     purrr::map2_dfr(s_list, seq_along(s_list), function(est_array, r) {
491         as.data.frame(as.table(est_array)) %>%
            dplyr::rename(quantile = Var1, parameter = Var2, value = Freq) %>%
            mutate(sim = s, rep = r)
    })
}) %>%
    dplyr::rename(Est = value) %>%
    mutate(true_value = case_when(parameter == "S_D" ~ S_D, parameter ==
        "Pf_D" ~ Pf_D, parameter == "Pa_D" ~ Pa_D, parameter == "M_D" ~
        M_D, )) %>%
    mutate(MRE = (Est - true_value)/true_value * 100) %>%
    dplyr::group_by(sim, quantile, parameter) %>%
    mutate(CV = ((sd(Est)/mean(Est)) * 100)) %>%
    mutate(df = "Est") %>%
    left_join(nind_)

# Unpacking the reach-specific detection probability estimates
p_df <- purrr::map2_dfr(det_p, seq_along(det_p), function(s_list, s) {
    purrr::map2_dfr(s_list, seq_along(s_list), function(est_array, r) {
        as.data.frame(as.table(est_array)) %>%
            dplyr::rename(quantile = Var1, reach = Var2, value = Freq) %>%
            mutate(sim = s, rep = r)
    })
}) %>%
    mutate(df = "p") %>%
    left_join(nind_)

ests_list <- list(M = Ests_M, Pf = Ests_Pf, Pa = Ests_Pa, S = Ests_S)

# Function for processing each list
process_ests <- function(x) {
    map2_dfr(x, seq_along(x), function(s_list, s) {
        map2_dfr(s_list, seq_along(s_list), function(est_array, r) {
            as.data.frame(as.table(est_array)) %>%
                rename(quantile = Var1, reach = Var2, value = Freq) %>%
                mutate(sim = s, rep = r)
        })
    })
}

# Apply to all reach-specific estimates, output named list: M_df,
# Pd_df, Pa_df, S_df
result <- imap(ests_list, ~process_ests(.x) %>%
    mutate(parameter = .y))
result <- do.call("rbind", result)

trueests_p <- tibble(parameter = rep(c("M", "Pf", "Pa", "S"), each = 6),
    true_value = c(M, Pf, Pa, S), reach = rep(c("R1", "R2", "R3", "R4",
        "R5", "R6"), times = 4))
Ests_P_df <- result %>%
    dplyr::rename(Est = value) %>%
    left_join(trueests_p) %>%
    mutate(MRE = (Est - true_value)/true_value * 100) %>%
    dplyr::group_by(sim, quantile, parameter, reach) %>%
    mutate(CV = ((sd(Est)/mean(Est)) * 100)) %>%
    mutate(df = "Est") %>%
    left_join(nind_)

```

```r
# Saving data
Ests_df %>%
    write_csv("model4_estimates.csv")
Ests_P_df %>%
    write_csv("model4_reachestimates.csv")
p_df %>%
    write_csv("model4_detectionprobability.csv")
N_State_recs_df %>%
    write_csv("model4_Nfishstate.csv")
runtimes %>%
    write_csv("model4_runtimes.csv")
```