#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Supplement B. Commented JAGS Model Definition coded and run in R.

# Perry et al. Flow-mediated effects on travel time, survival, and routing of

# juvenile Chinook salmon in a spatially complex, tidally forced river delta

#

# 21 November 2017

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

###### User input (directory names, filenames, etc.) #########

# Input for user's desired working directory

directory <- "Your working directory path here"

# Name of .RData object containing processed capture histories etc.

datafile <- "./Your local path/Your data filename.RData"

# Directory containing JAGS .exe file

jagspath <- "C:/Program Files/JAGS/JAGS-4.2.0/x64/bin/"

# Desired name for JAGS model definition textfile

jagsModel <- "Your JAGS model filename.txt"

# Desired name for JAGS saved model output

mcmcObject <- "./Your local path/Your output filename.RData"

##### Set working directory, load libraries, load data, strings as factors=F ###

setwd(directory)

require(runjags)

require(mcmcplots)

require(gplots)

library(Hmisc)

options(stringsAsFactors=F)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

load(datafile)

###### Specify model in JAGS language ########

sink(jagsModel)

cat("

model {

#---- Constraints ----------------------------------------------------------

#---- Define a special case of precision parameter for travel time in lower

#---- Sutter/Steamboat Sloughs, since no detection station there. This will

#---- be set equal to the precision parameter in the upper half of Sutter/Steamboat.

mean.tau.r.stm[1] <- 1

mean.tau.r.stm[2] <- 1

mean.tau.r.stm[3] <- 1

mean.tau.r.stm[4] <- mean.tau.r[2,3]

mean.tau.r.stm[5] <- 1

mean.tau.r.stm[6] <- 1

for (i in 1:nind){ # Loop through all fish

#---- Define a special case of survival parameter in lower

#---- Sutter/Steamboat Sloughs, since no detection station there. This will

#---- be set equal to the survival parameter in the upper half of Sutter/Steamboat.

phi.stm[i,1] <- 0

phi.stm[i,2] <- 0

phi.stm[i,3] <- sqrt(ilogit(max(min(alpha.cov[2,3] + beta.flow[2,3] \* FPT.flow[FPT.index[i,3]] +

beta.size \* size[i] + r.eff[release[i],2,3]\*sd.r.eff[2,3], 999), -999)))

phi.stm[i,4] <- phi.stm[i,3]

phi.stm[i,5] <- 0

phi.stm[i,6] <- 0

#----------------

#---- Define a special case of survival parameter in Sacramento River from junction

#---- with Sutter/Steamboat Sloughs to junction with Georgiana Slough, since no detection

#---- station there for certain release groups. This will be set equal to the survival

#---- parameter in the Sacramento River from Freeport to junction

#---- with Sutter/Steamboat for those release groups.

phi.J1[i,1] <- 0

phi.J1[i,2] <- 0

phi.J1[i,3] <- ilogit(max(min(alpha.cov[1,2] + beta.flow[1,2] \* FPT.flow[FPT.index[i,3]] +

beta.size \* size[i] + r.eff[release[i],1,2]\*sd.r.eff[1,2], 999), -999))

phi.J1[i,4] <- 0

phi.J1[i,5] <- 0

phi.J1[i,6] <- 0

#----------------

#---- Define a special case of mean parameter for travel time in lower

#---- Sutter/Steamboat Sloughs, since no detection station there. This will

#---- be set equal to the mean parameter in the upper half of Sutter/Steamboat.

mur.stm[i,1] <- 0

mur.stm[i,2] <- 0

mur.stm[i,3] <- alpha.time[2,3] + beta.time[2,3] \* FPT.flow[FPT.index[i,3]] +

r.eff.mut[release[i],2,3]\*sd.r.eff.mut[2,3]

mur.stm[i,4] <- mur.stm[i,3]

mur.stm[i,5] <- 0

mur.stm[i,6] <- 0

#----------------

#---- Define a special case of detection parameter for at entrance to the Delta Cross Channel.

#---- Because of the nature of downstream detection stations, detection probabilities at the DCC

#---- entrance and the Georgiana Slough entrance are inseparable. These will be constrained to be equal.

lp.geo[i,4] <- max(min(alpha.p.flow[year[i],3,4] + beta.p.flow[3,4] \* FPT.flow[FPT.index[i,4]], 999), -999)

lp.geo[i,2] <- 0

lp.geo[i,3] <- 0

lp.geo[i,5] <- 0

lp.geo[i,6] <- 0

lp.geo[i,7] <- 0

#----- End special case definitions -------------------------------------------

#--- Define constraints on p and construct observation matrix

for (t in (f[i]+1):n.occasions){ # Loop through occasions for each fish

for (j in Pl[Ps[t]:Pe[t],1]){ # Loop through available detection locations for each occasion

# Detection probability (p) defined via series of if/else statements to capture special cases

# set p for DCC (state 4) equal to geo (state 3) -- see definition of lp.geo above

p[i,j,t] <- ifelse(equals(t,4) && equals(j,4), ilogit(lp.geo[i,t]),

# set detection p for release 1 at J1 Sac (state 1) to 0, since no detection station there

# set detection p to zero at J1 Sac (state 1) for year 2 and phantom site in Sut/Stm (state 2)

ifelse((equals(t,3) && equals(j,1) && equals(release[i],1) && totTime[i,t] < 47.667) ||

(equals(t,3) && equals(j,1) && equals(year[i],2)) ||

(equals(t,4) && equals(j,2)), 0,

# set detection p at J2 Sac (state 1) in year 2 = p at Sut/Stm

ifelse(equals(t,4) && equals(j,1) && equals(year[i],2),

ilogit(max(min(alpha.p.flow[year[i],2,3] + beta.p.flow[2,3] \*

FPT.flow[FPT.index[i,t]], 999), -999)),

# otherwise p is a fn of year and flow

ilogit(lp[i,j,t]))))

lp[i,j,t] <- max(min(alpha.p.flow[year[i],j,t] + beta.p.flow[j,t] \* FPT.flow[FPT.index[i,t]], 999), -999)

#--- Construct observation matrix from detection probabilities -----------

# Prob of unobserved for live fish in a given state

po[j,i,t,nstates] <- 1 - p[i,j,t]

# Prob of observing live fish in a given state

po[j,i,t,j] <- p[i,j,t]

} #end j loop

# Prob of unobserved for dead fish

po[nstates,i,t,nstates] <- 1

} #end t loop

#------------------------------------------------------------------------------------------

#--- Define constraints on travel time, phi, and psi, and construct state transition matrix

for (t in f[i]:(n.occasions-1)){ #runs from either 1:6 or 4:6, depending on release site

for (j in sfl[sfs[t]:sfe[t],1]){ #runs through possible state transitions for occasion t

# Mean travel time (mu.r) defined via series of if/else statements to capture all cases

mu.r[i,j,t] <-

# if in Sut/Stm then default to mur.stm -- see definition for mur.stm above

ifelse((equals(t,4) || equals(t,3)) && equals(j,2), mur.stm[i,t],

# else mu.r is fn of flow, release-specific random effect, DCC effect

# for DCC reaches, and release site effect

alpha.time[j,t] + beta.time[j,t] \* FPT.flow[FPT.index[i,t]] +

r.eff.mut[release[i],j,t]\*sd.r.eff.mut[j,t] + DCC[i,t]\*DCC.reach[j,t] \* DCC.offset.time[j,t] +

Feather[i]\*equals(j,1)\*equals(t,1) \* Feather.offset +

SacElk[i]\*equals(j,1)\*equals(t,1) \* SacElk.offset)

# Precision parameter for travel time (tau.r) defined by ifelse to capture each case;

# if in lower half of SutStm, then mean tau for stm, else reach-specific tau --

# see definition for mean.tau.r.stm above

tau.r[i,j,t] <- ifelse(equals(t,4) && equals(j,2), mean.tau.r.stm[t], mean.tau.r[j,t])

# Survival probability (phi) defined via if/else to capture all cases

phi[i,j,t] <-

# if in Sut/Stm the default to phi.stm -- see definition for phi.stm above

ifelse((equals(t,4) || equals(t,3)) && equals(j,2), phi.stm[i,t],

# if in Sac. R. from J1 to J2 default to phi.J1 -- see definition of phi.J1 above

ifelse(equals(j,1) && equals(t,3), phi.J1[i,t],

# otherwise define phi for each reach

ilogit(lphi[i,j,t])))

# define logit of survival for most reaches as fn of flow, fish size,

# release-specific random effect, and DCC effect for DCC reaches

lphi[i,j,t] <- max(min(alpha.cov[j,t] + beta.flow[j,t] \* FPT.flow[FPT.index[i,t]] +

beta.size \* size[i] + r.eff[release[i],j,t]\*sd.r.eff[j,t] + DCC[i,t]\*DCC.reach[j,t] \*

DCC.offset.surv[j,t], 999), -999)

# transition to 'death state' equals 1 minus survival probability

ps[j,i,t,nstates] <- (1 - phi[i,j,t])

} #end j loop

# define travel time probability for dead individuals -- this does not impact

# other parameters but the model requires a definition

mu.r[i,nstates,t] <- dead.mu.r[t]

tau.r[i,nstates,t] <- mean.tau.r[nstates,t]

# death is an absorbing state

ps[nstates,i,t,nstates] <- 1

} #end t loop

} #end i loop

# Transition probability (ps) defined for constrained transitions

for(i in georel){ # loop through fish released into Georgiana Slough

for(t in 4:6){ # loop through occasions available to Geo. Sl. released fish

for (h in Ts[t]:Te[t]){ # loop through available transition states other than 'death state'

# Geo. Sl. released fish are locked into a single route to Chipps Island and so transition = survival

ps[Tl[h,1],i,t,Tl[h,2]] <- phi[i,Tl[h,1],t]

} #end h loop

} #end t loop

} #end i loop for Geo. Sl. released fish

for(i in sacrel){ # loop through fish released above Sacramento

for(t in c(1,4:6)){ # loop through occasions where only one transition is possible

for (h in Ts[t]:Te[t]){ # loop through start states where only one transition possible for each occasion

# Transition probabilities for occasions other than 2 and 3 are constrained to transition into a single state

# (other than the 'death state') -- for these occasions transition = survival

ps[Tl[h,1],i,t,Tl[h,2]] <- phi[i,Tl[h,1],t]

} #end h loop

} #end t loop

#--------Transition probabilities for occasion 2->3 (Sutter/Steamboat junction J1)

# Logit of prob. of entering Sutter/Steamboat is a fn of flow and release-specific random effect

lpsi.J1[i, 2] <- b0.J12 + b1.J12\*FPT.flow[FPT.index[i,3]] + r.eff.J12[release[i]]\*sd.J12

# Generalized logistic function allows estimation of an upper asymptote L

psi.J1[i, 2] <- L.J12/(1+exp(-lpsi.J1[i,2]))

# Staying in Sac. R. is 1 minus prob. of entering Sutter/Steamboat

psi.J1[i, 1] <- 1 - psi.J1[i, 2]

# Transition probabilities are product of entrainment prob. and survival prob.

ps[1,i,2,1] <- psi.J1[i, 1]\*phi[i,1,2]

ps[1,i,2,2] <- psi.J1[i, 2]\*phi[i,1,2]

#--------Transition probabilities for occ. 3->4 (either lower half od Sutter/Steamboat or Geo./DCC junction J2)

# Individuals in Sutter/Steamboat must remain in Sutter/Steamboat

ps[2,i,3,2] <- phi[i,2,3]

# unconditional probability of entering DCC; fn of flow and release-specific random effect

lpsi.J2[i, 3] <- ifelse(equals(s[i,3],1), ifelse(equals(DCC[i,4],0), -10,

b0.J24 + b1.J24\*FPT.flow[FPT.index[i,4]] + r.eff.J24[release[i]]\*sd.J24), 0)

psi.J2[i, 3] <- ilogit(lpsi.J2[i, 3])

# Probability of entering Geo conditional on not entering DCC; fn of flow, DCC open or closed,

# and release-specific random effect

lpsiG.notD[i] <- b0.J23 + b1.J23\*FPT.flow[FPT.index[i,4]] + b2.J23\*DCC[i,4] + r.eff.J23[release[i]]\*sd.J23

# Generalized logistic function allows estimation of lower asymptote A

psiG.notD[i] <- A.J23 + (1-A.J23)/(1+exp(-lpsiG.notD[i]))

# Define unconditional probability of entering Georgiana Slough

psi.J2[i, 2] <- (1-psi.J2[i, 3])\*psiG.notD[i]

# Define unconditional probability of remaining in Sacramento River

psi.J2[i, 1] <- (1-psi.J2[i, 3])\*(1-psiG.notD[i])

# Transition probabilities are product of entrainment prob. and survival prob.

ps[1,i,3,1] <- psi.J2[i, 1] \* phi[i,1,3]

ps[1,i,3,3] <- psi.J2[i, 2] \* phi[i,1,3]

ps[1,i,3,4] <- psi.J2[i, 3] \* phi[i,1,3]

} #end i loop for Sacramento released fish

#----- Priors --------------------------------------------------------------

# Entrainment parameters

# Intercepts: t dist. w/ k=1, mu=0, & tau=0.01 is a Cauchy(0,10) (see Gelman)

b0.J12 ~ dt(0,0.01,1)

b0.J24 ~ dt(0,0.01,1)

b0.J23 ~ dt(0,0.01,1)

# Slopes: scale parameter (sigma) = 2.5 per Gelman, corresponds to tau = (1/sigma)^2 = (0.4)^2 = 0.16

b1.J12 ~ dt(0,0.16, 7)

b1.J24 ~ dt(0,0.16, 7)

b1.J23 ~ dt(0,0.16, 7)

b2.J23 ~ dt(0,0.16, 7)

# Logistic function asymptotes

L.J12 ~ dbeta(1,1)

A.J23 ~ dbeta(1,1)

# Random effects

sd.J12 ~ dnorm(0,1) T(0,)

sd.J23 ~ dnorm(0,1) T(0,)

sd.J24 ~ dnorm(0,1) T(0,)

# Survival Parameters

# Slope on fish size

beta.size ~ dt(0,0.16, 7)

# Travel time parameters

# Release site offsets

Feather.offset ~ dnorm(0,0.01)

SacElk.offset ~ dnorm(0,0.01)

# Parameters defined by state and/or occasion

for (t in 1:(n.occasions-1)){ # Loop through occasions

for (j in sfl[sfs[t]:sfe[t],1]){ # Loop through available states

# Survival parameters

# Intercept: t dist. w/ k=1, mu=0, & tau=0.01 is a Cauchy(0,10)

alpha.cov[j,t] ~ dt(0,0.01,1)

# Slope: scale parameter (sigma) = 2.5 per Gelman, corresponds to tau = (1/sigma)^2 = (0.4)^2 = 0.16

beta.flow[j,t] ~ dt(0,0.16,7)

# Random effect

sd.r.eff[j,t] ~ dnorm(0,1) T(0,)

# DCC effect offset

DCC.offset.surv[j,t] ~ dnorm(0,0.01)

# Travel Time parameters

# Intercept

alpha.time[j,t] ~ dnorm(0,0.01)

# Slope

beta.time[j,t] ~ dnorm(0,0.01)

# Random effect

sd.r.eff.mut[j,t] ~ dunif(0,10)

# DCC effect offset

DCC.offset.time[j,t] ~ dnorm(0,0.01)

# Precision/Dispersion

mean.tau.r[j,t] <- pow(mean.sigma.r[j,t],-2)

mean.sigma.r[j,t] ~ dunif(0,10)

} #end j loop

# Travel Time parameters for 'death state'

mean.tau.r[nstates,t] <- pow(mean.sigma.r[nstates,t],-2)

mean.sigma.r[nstates,t] ~ dunif(0,10)

dead.mu.r[t] ~ dnorm(0, 1.0E-2)

} #end t loop

# Occasion and state loops for detection parameters (locations) are defined over a different index

# than for survival/travel time parameters (reaches)

# Detection parameters

for (t in 2:n.occasions){ # Loop through location occasions

for (j in Pl[Ps[t]:Pe[t],1]){ # Loop through available location states

for (yr in 1:Nyear){ # Loop through years

# Intercept: t dist. w/ k=1, mu=0, & tau=0.01 is a Cauchy(0,10)

alpha.p.flow[yr,j,t] ~ dt(0,0.01,1)

} #end yr loop

# Slope: scale parameter (sigma) = 2.5 per Gelman, corresponds to tau = (1/sigma)^2 = (0.4)^2 = 0.16

beta.p.flow[j,t] ~ dt(0,0.16,7)

} #end j loop

} #end t loop

#

for(rel in 1:nrel){ # Loop through release groups

# Random effects

r.eff.J12[rel] ~ dnorm(0,1)

r.eff.J23[rel] ~ dnorm(0,1)

r.eff.J24[rel] ~ dnorm(0,1)

for (t in 1:(n.occasions-1)){ # Loop through occasions

for (j in sfl[sfs[t]:sfe[t],1]){ # Loop through available states

# Random effects for entrainment

r.eff[rel,j,t] ~ dnorm(0,1)

r.eff.mut[rel,j,t] ~ dnorm(0,1)

} #end j loop

} #end t loop

} #end rel loop

#------------- Likelihood --------------------------------------------------

# Fish released at Georgiana Slough don't appear until occasion 4, but we must define

# certain indices for occasions 1-3 so JAGS does not throw an error

for (i in georel){ # Loop through fish released at Geo. Sl.

for (t in 1:(f[i]-1)){ # Loop through occasions before release

FPT.index[i,t] <- 1

DCC[i,t] <- 1

} #end t loop

} #end i loop

for (i in 1:nind){ # Loop through all fish

# Define true state at first capture

s[i,1:f[i]] <- y[i,1:f[i]]

# Define true state for replicate dataset at first capture

s.rep[i,1:f[i]] <- y[i,1:f[i]]

# Generate replicate data for first capture

y.rep[i,1:f[i]] <- y[i,1:f[i]]

# Use nested index to assign DCC open/closed on release occasion. See comments at assignment

# of DCC open/closed on occasions after release below.

DCC.open[i,f[i]] <- sum(step(totTime[i,f[i]]-DCC.int))

DCC[i,f[i]] <- DCC.ops[DCC.open[i,f[i]]]

for (t in (f[i]+1):n.occasions){ #Loop through occasions after release. t= 2:7 for sac or 5:7 for geo

# DCC.int is input vector of DCC ops times.

# IMPORTANT: DCC.int[1] should be earlier than any fish can possibly arrive

DCC.open[i,t] <- sum(step(totTime[i,t]-DCC.int))

# DCC.ops is input vector of DCC ops states (1=open, 0=closed). Indexing matches times for DCC.int

DCC[i,t] <- DCC.ops[DCC.open[i,t]]

# Assign julian date of time at arrival to obtain index of daily covariates

FPT.index[i,t-1] <- sum(step(totTime[i,t-1]-FPT.ind[1:Ndays]))

# Travel time between adjacent locations is distributed as truncated lognormal

TTime[i,t-1] ~ dlnorm(mu.r[i,s[i,t-1],t-1],tau.r[i,s[i,t-1],t-1]) T(0.000001,90)

# Constrain individual reach travel times to sum to observed times from release

totTime[i,t] ~ dsum(totTime[i,t-1], TTime[i,t-1])

# State process: draw S(t) given S(t-1)

s[i,t] ~ dcat(ps[s[i,t-1], i, t-1, 1:nstates])

# Observation process: draw O(t) given S(t)

y[i,t] ~ dcat(po[s[i,t], i, t, 1:nstates])

# Replicate dataset is used to calculate Bayesian p.value

TTime.rep[i,t-1] ~ dlnorm(mu.r[i,s[i,t-1],t-1], tau.r[i,s[i,t-1],t-1]) T(0.00001,90)

# Constrain replicate travel times to sum to observed times from release

totTime.rep[i,t] ~ dsum(totTime.rep[i,t-1], TTime.rep[i,t-1])

# State process for replicate data: draw S(t) given S(t-1)

s.rep[i,t] ~ dcat(ps[s[i,t-1], i, t-1, 1:nstates])

# Observation process for replicate data: draw O(t) given S(t)

y.rep[i,t] ~ dcat(po[s.rep[i,t], i, t, 1:nstates])

} #end t loop

# Assign julian date of time at arrival on final occasion to obtain index of daily covariates

FPT.index[i,n.occasions] <- sum(step(totTime[i,n.occasions]-FPT.ind[1:Ndays]))

} #end i loop

for (t in 1:(n.occasions-1)){ # Loop through occasions

for (j in Pl[Ps[t+1]:Pe[t+1],1]){ # Loop through available detection locations for each occasion

for (yr in 1:Nyear){ # Loop through study years

# Derived parameter mean.p represents detection probability; used for post-analysis evaluation

mean.p[yr,j,t+1] <- sum(p[1:nind,j,t+1]\*equals(s[1:nind,t+1],j)\*equals(year[1:nind],yr))/

(sum(equals(s[1:nind,t+1],j)\*equals(year[1:nind],yr)) +

equals(sum(equals(s[1:nind,t+1],j)\*equals(year[1:nind],yr)),0))

} #end yr loop

} #end j loop

} #end t loop

#\*\*\*\* Auxilliary likelihood for J1 routing data from 2014 \*\*\*\*

for(rel in 1:nRel2014){ # Loop through 2014 release groups

# Prior on random effect

r.eff.J12.2014[rel] ~ dnorm(0,1)

for(i in relInd[rel,1]:relInd[rel,2]){ # Loop through individuals in each 2014 release group

# form of relationship is identical to that for parameter psi.J1 above

lpsi.J1.2014[i] <- b0.J12 + b1.J12\*FPT2014[i] + r.eff.J12.2014[rel]\*sd.J12

psi.J1.2014[i] <- L.J12/(1+exp(-lpsi.J1.2014[i]))

#--- Likelihood

SSI[i] ~ dbern(psi.J1.2014[i])

} #end i loop

} #end rel loop

} #end model definition

",fill = TRUE)

sink()

############### Function to create known latent states s #############

#uses architecture of multistate model to infer known state based on past or future detections

#this architecture is model-specific and code must be changed accordingly for each new model

known.state.cond <- list(

list(occasion=numeric(), condition=numeric(), state=numeric()),

list(occasion=3:n.occasions, condition=rep(list(1:(nstates-2)),n.occasions-2), state=rep(1,n.occasions-2)),

list(occasion=4:5, condition=list(c(1,3,4),3), state=c(1,1)),

list(occasion=numeric(), condition=numeric(), state=numeric()),

list(occasion=numeric(), condition=numeric(), state=numeric()),

list(occasion=n.occasions, condition=1, state=1))

# Given a particular capture history, an individual may be constrained to be in a particular state even if

# it was not detected on that occasion (see description of capture history in paper). This function creates

# a latent known state history to pass to the JAGS model based on these constraints.

known.state.ms <- function(ms, notseen, f){

# notseen : number representing non-observations in capture histories (in lieu of NA)

# ms : capture histories -- matrix of i individuals by j occasions

# f : occasion of release for each individual -- vector of length i

state <- ms

# Loop through some occasions where state may be known despite non-detection (2, 3, and 6).

# Use architecture of 'known.state.cond' list created above to assign state to these occasions,

# if warranted by capture history.

for (j in c(2,3,6)){

unknown.j <- which(state[,j]==notseen & j>f)

if (length(known.state.cond[[j]]$occasion)>=1){

for (j2 in 1:length(known.state.cond[[j]]$occasion)){

state[unknown.j,j] <- ifelse(state[unknown.j,known.state.cond[[j]]$occasion[j2]] %in%

known.state.cond[[j]]$condition[[j2]],

known.state.cond[[j]]$state[j2], state[unknown.j,j])}

}

}

# For occasion 4, state is known to be 1 (Sac.R.) only if also known to be 1 on occasions 3 and 5

unknown.j <- which(state[,4]==notseen & 4>f)

state[unknown.j,4] <- ifelse(state[unknown.j,3]==1 & state[unknown.j,5]==1,1,state[unknown.j,4])

# If unobserved states are still unknown, re-label as NA to pass to JAGS

# Also, assign occasion of release as NA (for JAGS, since this is passed in as data)

state[state==notseen] <- NA

for (i in 1:dim(ms)[1]){

m <- min(which(!is.na(state[i,])))

state[i,m] <- NA

}

return(state)

}

################ End known states function ################

############### Function to create initial values for latent states ###########

# Can't let JAGS assign initial states, since they may violate model transition constraints

init.state.ms <- function(ms.known, f){

# ms.known : matrix of known latent states created by known.state.ms function above

# f : occasion of release for each individual -- vector of length i

ms.init <- ms.known

for (i in 1:(dim(ms.known)[1])){ # Loop through individual capture histories

if (is.na(ms.known[i,4])){ #only valid for DCC closed (see known.state.cond definition above)

# Start with occasion 4. Case 1: Unknown

# Can be in states 1 or 2 on occ. 3 if occs. 3 & 4 both unknown

ms.init[i,3] <- ifelse(is.na(ms.known[i,3]), sample(c(1,2),1), NA)

# State on occ. 4 is same as occ 3

ms.init[i,4] <- ifelse(is.na(ms.known[i,3]), ms.init[i,3], NA)

if (!is.na(ms.known[i,3])){ # If we know where it is on occ. 3

if (ms.known[i,3] == 2) { # If we know it's in Sutter/Steamboat

ms.init[i,4] <- ms.known[i,3]

ms.init[i,5] <- ifelse(is.na(ms.known[i,5]), 1, NA)

} else { # If we know it's in the Sac. R. (occ. 3)

ms.init[i,3] <- NA

# Can be in Sac. R. or Geo. Sl.

ms.init[i,4] <- ifelse(is.na(ms.known[i,5]),sample(c(1,3),1),ifelse(ms.known[i,5]==1,1,3))

# State on occ. 5 is same as occ. 4

ms.init[i,5] <- ifelse(is.na(ms.known[i,5]),ifelse(ms.init[i,4]==1,1,3),NA)

}

} else if (ms.init[i,3] == 2){

# If we don't know where it is on occ. 3 but it's initialized as being in Sutter/Steam. Sl.

ms.init[i,5] <- ifelse(is.na(ms.known[i,5]), 1, NA)

} else if (ms.init[i,3]==1) {

# If we don't know where it is on occ. 3 but it's initialized as being in Sac R.

ms.init[i,5] <- ifelse(is.na(ms.known[i,5]),ifelse(ms.init[i,4]==1,1,3),NA)

}

} else if (ms.known[i,4] == 2){ # Case 2: Known in Sutter Steamboat on occ. 4

ms.init[i,3] <- ifelse(is.na(ms.known[i,3]), ms.known[i,4], NA)

ms.init[i,4] <- NA

ms.init[i,5] <- ifelse(is.na(ms.known[i,5]), 1, NA)

} else if (ms.known[i,4] %in% c(3,4)){ # Case 3: in Geo. Sl. or DCC on occ. 4

ms.init[i,5] <- ifelse(is.na(ms.known[i,5]), 3, NA)

ms.init[i,4] <- ms.init[i,3] <- NA

} else if (ms.known[i,4]==1){ # Case 4: in Sac. R. on occ. 4

ms.init[i,3] <- ms.init[i,4] <- NA

ms.init[i,5] <- ifelse(is.na(ms.known[i,5]), 1, NA)

}

# Initialize other occasions where unknown, assign NA where data exists

ms.init[i,2] <- ifelse(is.na(ms.known[i,2]), 1, NA)

if (f[i]==4) ms.init[i,5] <- ifelse(is.na(ms.known[i,5]), 3, NA)

ms.init[i,6] <- ifelse(is.na(ms.known[i,6]), 1, NA)

ms.init[i,7] <- ifelse(is.na(ms.known[i,7]), 1, NA)

ms.init[i,1:f[i]] <- NA

}

ms.init[!is.na(ms.known)] <- NA

return(ms.init)

}

################# End initial values for latent states function #################

############### Function to create initial values for unknown cumulative travel times #########

cjs.init.t <- function(ch,TTime,totTime,f,rep=FALSE) {

# ch : capture history matrix (observed)

# TTime : observed reach-specific travel time matrix

# totTime : observed cumulative travel time matrix

# f : occasion of release for each individual -- vector of length i

tot.inits <- array(NA,dim=c(dim(ch)[1],dim(ch)[2]))

t.out <- array(NA,dim=c(dim(ch)[1],dim(ch)[2]-1))

for (i in 1:dim(ch)[1]) { # Loop through each individual

tot.inits[i,f[i]]<-totTime[i,f[i]]

for (j in (f[i]+1):(dim(ch)[2])) { # Loop through each occasion of capture history

tot.inits[i,j]<-totTime[i,j]

if (is.na(tot.inits[i,j])){

prev.time <- tot.inits[i,j-1]

if (j < (dim(ch)[2])) {

# If there is another observed travel time after this missing one

if (length(which(!is.na(totTime[i,(j+1):(dim(ch)[2])])))>0) {

next.occ <- min(which(!is.na(totTime[i,(j+1):(dim(ch)[2])])))+j

next.time <- totTime[i,next.occ]

# Initialize missing times as evenly distributed between observations

tot.inits[i,j] <- round((next.time - prev.time)/(next.occ - (j-1)),5) + prev.time

} else tot.inits[i,j] <- (prev.time + 1.1)/1

# If no more observed times after this missing one, initialize as some fixed time after last observation

} else tot.inits[i,j] <- (prev.time + 1.1)/1

}

}

t.out[i,] <- diff(tot.inits[i,])

if (rep==FALSE) {t.out[i,which(!is.na(TTime[i,1:(dim(ch)[2]-1)]))]<-NA}

}

return(t.out)

}

############ End initial values for missing travel times function #################

#----------- End model and function definition

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

########## Create list containing data to pass to MCMC

jags.data <- list(

# Data dimensions

n.occasions = dim(rCH)[2], # Length of capture history vector

nstates = nstates, # Total number of available states

nind = dim(rCH)[1], # Number of individuals

nrel = nrel, # Number of release groups

Nyear = Nyear, # Total number of years in study

# Individual detection history data

y = rCH, # Observed capture histories

s = known.state.ms(rCH, 6, f), # Known latent state histories

totTime = rtotTime, # Observed cumulative travel times

totTime.rep = rtotTime, # Replicate of tottime for Bayesian p-value calculation

TTime = rTTime, # Observed reach-specific travel times

# Release information and data

f = f, # Release occasion for each fish

release = release, # Release group for each fish

sacrel = sacrel, # Indices of Sacramento released fish

georel = georel, # Indices of Georgiana Sl. released fish

Feather = Feather, # Vector indicating fish released at Feather R. site

SacElk = SacElk, # Vector indicating fish released at upstream Elk Landing site

year = year, # Study year for each fish

# Covariates

Ndays = Ndays, # Total number of days from start to finish of multiyear study

FPT.flow = fpt.dat$cfs.std, # Vector of daily Freeport flow covariates

FPT.ind = as.numeric(fpt.dat$studyDate),# Vector of dates to match to Freeport flows (FPT.flow) above

size=size.std, # Size covariate for each fish

DCC.int = DCC.int, # Vector of dates when DCC opened or closed

DCC.ops = DCC.ops, # Vector of DCC states (open/closed)

DCC.reach=DCC.reach, # Which reaches need DCC offset

# Matrix dimension and index information for JAGS loops and lookups

sfl = sfl, sfs = sfs, sfe = sfe, # Index of available states at each occasion

Tl = Tl, Ts = Ts, Te = Te, # Index of available transitions at each occasion

Pl = Pl, Ps = Ps, Pe = Pe, # Index of available detection states at each occasion

# Definition of trivial/undefined nodes: this is a bookkeeping excersize to ensure JAGS loops run contiguously

ps = ps, po = po, # Set undefined transition and observation matrices

mu.r = mu.r, tau.r = tau.r, phi = phi, # Set undefined parameter matrices

# Set undefined detection matrix to 0, so that derived parameter mean.p is unaffected by undefined nodes

p = p,

# J1 2014 auxilliary data for routing model

SSI = J1.df$SSI, # Indicator for J1 routing of individual 2014 fish

FPT2014 = FPT2014, # Daily Freeport flow covariate for 2014

relInd = relInd, # Index of release groups for 2014 fish

nRel2014 = nRel2014 # Total number of 2014 fish released

)

######### Parameters monitored and returned by MCMC

parameters.waic <- c(

# Travel time parameters

"alpha.time", # Intercept: mean

"beta.time", # Slope for flow covariate: mean

"DCC.offset.time", # Effect of DCC open/closed: mean

"Feather.offset", # Effect of release at Feather R. site: mean (Reach 0 only)

"SacElk.offset", # Effect of release at Elk Landing site: mean (reach 0 only)

"sd.r.eff.mut", # Dispersion of release-specific random effects on the mean

"mean.sigma.r", # Dispersion of reach-specific travel times

# Detection probability parameters

"alpha.p.flow", # Intercept

"beta.p.flow", # Slope for flow covariate

"mean.p", # Derived mean site- and year- specific detection probability

# Survival parameters

"alpha.cov", # Intercept

"beta.flow", # Slope for flow covariate

"beta.size", # Slope for fish size covariate

"DCC.offset.surv", # Effect of DCC open/closed

"sd.r.eff", # Dispersion of release-specific random effects

# Routing parameters

# Junction of Sacramento R. with Sutter/Steamboat Slough (J1)

"b0.J12", # Intercept

"b1.J12", # Slope

"b2.J12", # Effect of DCC open/closed

"sd.J12", # Dispersion of release-specific random effects

"L.J12", # Upper asymptote for entrainment into Sutter/Steamboat

# Junction of Sacramento R. with DCC (J2)

"b0.J24", # Intercept

"b1.J24", # Slope

"sd.J24", # Dispersion of release-specific random effects

# Junction of Sacramento R. with Georgiana Slough (J2)

"b0.J23", # Intercept

"b1.J23", # Slope

"b2.J23", # Effect of DCC open/closed

"A.J23", # Lower asymptote for entrainment into Geogiana

"sd.J23", # Dispersion of release-specific random effects

# Random effects (Draws from Normal(0, sigma) included for posterior checks and plots)

"r.eff.J12", "r.eff.J12.2014", "r.eff.J23", "r.eff.J24", "r.eff", "r.eff.mut",

# Latent and replicate parameters for Bayesian p-value computation

"s", "s.rep", "y.rep", "TTime", "TTime.rep"

)

######### MCMC settings

ni <- 1000 # Number of iterations we want to keep in posterior sample for each chain

nt <- 20 # Thin rate -- take 1 posterior sample per this many iterations

nb <- 30000 # Number of burn-in iterations

nc <- 3 # Number of chains to run

######### Create list of initial values to pass to MCMC

inits <- as.list(rep(NA,nc))

init.s <- init.state.ms(known.state.ms(rCH, 6, f),f)

tTime.init <- cjs.init.t(rCH,rTTime,rtotTime, f)

tTime.rep.init <- cjs.init.t(rCH, rTTime, rtotTime, f, rep=T)

# Initial values

for (i in 1:nc) {

inits[[i]] <- list(

TTime = tTime.init,

TTime.rep = tTime.rep.init,

s = init.s)

}

######### Call JAGS from R using Package runjags

(start=Sys.time())

runjags::runjags.options(jagspath=jagspath)

ms.sim.vem.pQ.ttQ.phiQ.psiQreff.wJ12014 <-

runjags::run.jags(model=jagsModel,

monitor = parameters.waic, data=jags.data, n.chains = nc, inits=inits, burnin = nb,

sample = ni, adapt = 1000, thin = nt, method="parallel", summarise = F)

(end=Sys.time())

(elapsed=end-start)

save.image(mcmcObject)