Supplement 2

R code for: *Low impact of first-time spawners on population growth in a brown trout population*

Marlene Wæge Stubberud[1,*], Chloé R. Nater[1,2], Yngvild Vindenes[1], L. Asbjørn Vøllestad[1], Øystein Langangen[1].

[1] Department of Biosciences, University of Oslo, NO-0316 Oslo, Norway
[2] Centre for Biodiversity Dynamics (CBD), NTNU, NO-7491 Trondheim, Norway
*Author for correspondence: m.w.stubberud@ibv.uio.no

Contents

General Notes:

This integral projection model (IPM) is based on the model for Hunder trout found here:

https://github.com/ChloeRN/HunderTroutIPM

The model distinguishes between first-time and repeat spawners and assumes that all mortality for mature adults in spawning and non-spawning years happens before the spawning-non-spawning transition (splitting mortality hazard rate 100:0 over two years). The census is placed just *after* mature trout ascend or do not ascend the ladder.

Required packages

```
library(dplyr)
library(ggplot2)
library(viridis)
library(cowplot)
library(Matrix)
```

1. Setup

Baseline parameters

```
## Lower and upper length boundaries for the individual state variable (length, in
↪  cm)
# l.limit = lower limit below which to set harvest mortality to 0
# u.limit = upper limit above which to set harvest mortality to 0
l.limit = Lx = 0
u.limit = Ux = 1300

## Number of "mesh points", for discretization of the integral projection model
# n = 300 # used for final run
n = 100 # used to save time

## Vector of lengths
x = seq(Lx, Ux, length=n)

## Bin size
dx = x[2] - x[1]

## Bin width
h <- (Ux-Lx)/n


## General settings for ignoring year and individual variation
chosenYear = 1991
test.year = no.years = chosenYear - 1951 # 40 years
no.inds = 1
dam = 1

## Set random effects and environmental covariates
## Setting individual random effect levels to 0
epsilon.grR.i <- rnorm(no.inds, 0, 0)
epsilon.grL.i <- rnorm(no.inds, 0, 0)
epsilon.muI.i <- rnorm(no.inds, 0, 0)

## Setting year random effect levels to 0
epsilon.grR <- rnorm(no.years, 0, 0)
epsilon.grL <- rnorm(no.years, 0, 0)
epsilon.mH <- rnorm(no.years, 0, 0)
epsilon.mO <- rnorm(no.years, 0, 0)
epsilon.pS <- rnorm(no.years, 0, 0)
epsilon.pM <- rnorm(no.years, 0, 0)
epsilon.pL1 = rnorm(no.years, 0, 0) # ladder probability for first-time spawners
epsilon.pL <- rnorm(no.years, 0, 0) # ladder probability for repeat spawners

## Setting discharge covariate to 0
discF.std <- rep(0, no.years)
discS.std <- rep(0, no.years)
```

2. Vital rate prediction functions

River growth

```
## Parameter values
mu0 <- 1.548 # Average length at hatching
h0 <- 69.206 # Baseline river growth increment (time = 0, year = 1951)
betaYR <- -0.224 # Time trend in river growth (times = 1:51, years = 1952:2002)
varR.i <- 59.521 # Variance of among-individual differences in river growth
varR.t <- 10.375 # Variance of among-year differences in river growth
varR.R <- 188.074 # Residual variance in river growth


## Prediction function for average river growth
mean.grR.pred <- function(size, t, i){

  # Calculate increment (linear)
  inc <- h0 + betaYR*t + epsilon.grR[t] + epsilon.grR.i[i]

  # Add increment to size
  size.next <- size + inc
  return(size.next)
}


## Prediction function for river growth distribution
dist.grR.pred <- function(size, size.next, t, i){

  # Calculate growth and set variance parameter
  mu <- mean.grR.pred(size, t, i)
  var <- varR.R

  # Make a switch to account for potential negative growth
  z <- ifelse(size.next-size > 0, size.next, 0)

  # Calculate densities
  y <- dnorm(z, mean = mu, sd = sqrt(var))

  # Scale and return densities
  if(sum(y*dx)==0){
    return (c(rep(0,n-1),1/dx))
  }
  else return(y/sum(y*dx))
  y/sum(y*dx)
}
```

Lake growth

```r
## Parameter values (1st value = wild, 2nd value = stocked)
k0 <- c(0.155, 0.139) # Baseline lake growth rate
mu_inf0 <- c(1295.570, 1435.445) # Average asymptotic length
betaS <- -1.770 # Effect of spawning status on log(k)
varL.i.k <- 0.024 # Variance of among-individual differences in log(k)
varL.i.mu <- 0 # Variance of among-individual differences in asymptotic size
varL.t <- 0.014 # Variance of among-year differences in log(k)
varL.R <-  396.822 # Residual variance in lake growth increment

## Growth reduction factor for mature fish
redFactor <- 0.63

## Prediction function for river growth
mean.grL.pred <- function(size, t, i, spawn.year, orig, mature){

    # Calculate lake growth rate and asymptotic size
    k <- exp(log(k0[orig]) + betaS*spawn.year + epsilon.grL[t] + epsilon.grL.i[i])
    mu_inf <- mu_inf0[orig] + epsilon.muI.i[i]

    # Calculate increment (von Bertalanffy)
    if(mature == 0){
        inc <- (mu_inf - size)*(1-exp(-k))
    }else{
        inc <- (mu_inf - size)*(1-exp(-k))*redFactor
    }

    # Include residual variation in increment and add to size
    #size.next <- size + rnorm(1, inc, sqrt(varL.R))
    size.next <- size + inc
    return(size.next)
}

## Prediction function for lake growth distribution
dist.grL.pred <- function(size, size.next, t, i, spawn.year, orig, mature){

  # Calculate growth and set variance parameter
  mu <- mean.grL.pred(size, t, i, spawn.year, orig, mature)
  var <- varL.R

  # Make a switch to account for potential negative growth
  #z <- ifelse(size.next-size > 0, size.next, 0)
  z <- size.next

  # Calculate densities
  y <- dnorm(z, mean = mu, sd = sqrt(var))

  # Scale and return densities
  if(sum(y*dx)==0){
    return (c(rep(0,n-1),1/dx))
  }
  else return(y/sum(y*dx))
  y/sum(y*dx)
}
```

Adult survival

```
## Parameter values (1st value = wild, 2nd value = stocked)
Mu.mH <- c(1.272, 1.258) # Median harvest mortality hazard rate
Mu.mO1 <- c(0.104, 0.103) # Median background mortality hazard rate (above-dam)
Mu.mO2 <- c(0.158, 0.157) # Median background mortality hazard rate (below-dam)
beta2.mH <- -0.345 # Linear size effect on log(mH)
beta4.mH <- -0.153 # Quadratic size effect on log(mH)
beta1.mO1 <- 0.639 # Linear discharge effect on log(mO1)
beta1.mO2 <- 0.063 # Linear discharge effect on log(mO2)
beta2.mO1 <- -0.315 # Linear size effect on log(mO1)
beta2.mO2 <- 0.781 # Linear size effect on log(mO2)
sigma.mH <- 0.135 # SD of random year variation in log(mH)
sigma.mO <- 1.322 # SD of random year variation in log(mO1) and log(mO2)

## Prediction function for harvest mortality (2 years)
mH.2yr.pred <- function(size, t, orig, l.limit, u.limit, mH.factor){

    # Scale length
    size.std <- (size - 669.1941)/108.6021

    # Make prediction
    y.mH <- log(Mu.mH[orig]) + beta2.mH*size.std + beta4.mH*(size.std^2) +
↪   epsilon.mH[t]
    mH <- exp(y.mH)*mH.factor

    # Set mH to 0 below/above limits
    mH[which(size < l.limit)] <- 0
    mH[which(size > u.limit)] <- 0

    return(mH)
}

## Prediction function for background mortality (2 years)
mO.2yr.pred <- function(size, dam, t, orig, mO1size.factor){

    # Scale length
    size.std <- (size - 669.1941)/108.6021

    # Make prediction (based on spawning location)
    if(dam == 1){
        y.mO <- log(Mu.mO1[orig]) + beta1.mO1*discF.std[t] + beta2.mO1*size.std +
↪   epsilon.mO[t]
        y.mO[which(size.std < 0)] <- log(Mu.mO1[orig]) + beta1.mO1*discF.std[t] +
↪   beta2.mO1*size.std[which(size.std < 0)]*mO1size.factor + epsilon.mO[t]

    }
    if(dam == 0){
        y.mO <- log(Mu.mO2[orig]) + beta1.mO2*discF.std[t] + beta2.mO2*size.std +
↪   epsilon.mO[t]

    }
    mO <- exp(y.mO)
    return(mO)
}
```

```r
## Prediction function for adult survival (repeat spawners)
Sa.pred <- function(size, dam, t, orig, min.size, max.size, spawn.year, l.limit,
↪  u.limit, mH.factor, mO1size.factor){
    # Calculate the vector of survival probabilities (whole size range)
    mTOT <- mH.2yr.pred(size, t, orig, l.limit, u.limit, mH.factor) +
↪  mO.2yr.pred(size, dam, t, orig, mO1size.factor)
    Sa <- exp(-mTOT)

    # Calculate minimum and maximum size survival
    mTOT.min <- (mH.2yr.pred(min.size, t, orig, l.limit, u.limit, mH.factor) +
↪  mO.2yr.pred(min.size, dam, t, orig, mO1size.factor))
    mTOT.max <- (mH.2yr.pred(max.size, t, orig, l.limit, u.limit, mH.factor) +
↪  mO.2yr.pred(max.size, dam, t, orig, mO1size.factor))
    Sa.min <- exp(-mTOT.min)
    Sa.max <- exp(-mTOT.max)

    # Replace survival probabilites for sizes beyond thresholds
    Sa[which(size < min.size)] <- Sa.min
    Sa[which(size > max.size)] <- Sa.max

    if(spawn.year==1){Sa.final <- Sa}
    if(spawn.year!=1){Sa.final <- rep(1, length(Sa))}
    return(Sa.final)
}

## Prediciton function for adult survival (first-time spawners)
Sa1.pred <- function(size, dam, t, orig, min.size, max.size, spawn.year, l.limit,
↪  u.limit, mH.factor, mO1size.factor){

if(dam == 1){ # if above dam, same survival as repeat spawners
  Sa1 <- Sa.pred(size, dam, t, orig, min.size, max.size, spawn.year = 1, l.limit,
↪  u.limit, mH.factor, mO1size.factor)
} else { # if below dam, subadult survival, adjusted for 2 year interval
  mOs <- mOs.pred(size, Mu.mOs, beta2.mOs)*2
  mHs <- mH.2yr.pred(size, t, orig, l.limit, u.limit, mH.factor)
  mTOT <- mHs + mOs
  Sa1 <- exp(-mTOT)
}
return(Sa1)
}
```

Subadult survival

```
## Prediction function for subadult background mortality
mOs.pred <- function(size, Mu.mOs, beta2.mOs){

    size.std <- (size-437)/150 # 437 chosen as median (halfway between length at
↪  smolting/maturation based on growth data), 150 chosen as scaling factor

  log.mOs <- log(Mu.mOs) + beta2.mOs*size.std
  mOs <- exp(log.mOs)

  return(mOs)
}

# Suggestions for constant mOs
# Average: 0.7
# Low: 0.2
# High: 1.7

## Prediction function for annual subadult survival
# `mO.s` is the median background mortality hazard rate for subadults.
# Orig: 1 = wild, 2 = stocked.
Ss.pred <- function(size, t, orig, Mu.mOs, beta2.mOs, l.limit, u.limit, mH.factor){

    mOs <- mOs.pred(size, Mu.mOs, beta2.mOs)
    mTOT <- (mH.2yr.pred(size, t, orig, l.limit, u.limit, mH.factor)/2) + mOs
    Ss <- exp(-mTOT)
    return(Ss)
}
```

Juvenile survival

Annual survival, after first year from egg to parr before smolting.

```
Sj.pred <- function(size, Mu.mj, beta2.mj){

    size.std <- (size-157)/50 # 157 chosen as median (halway between length at age 1
↪  and length at smolting)

  log.mj <- log(Mu.mj) + beta2.mj*size.std
  mj <- exp(log.mj)
  return(exp(-mj))
}
```

Smolting probability

```
## Parameter values
mu.pS <- -2.482 # Intercept of logit(pS)
beta2.pS <- 2.935 # Size effect on logit(pS)
betaY2.pS <- 0.294 # Size:time effect on logit(pS)
var.pS <- 0.205 # Among-year variance in logit(pS)

## Prediction function for smolting probability
pS.pred <- function(size, t){

    # Calculate relevant year from t (t=0 corresponds to 1951)
    year <- t + 1951

    # Scale length and year
    size.std <- (size - 164.5047)/78.71436
    year.std <- (year - 1977.933)/12.48213

    # Make prediction
    y.pS <- mu.pS + beta2.pS*size.std + betaY2.pS*size.std*year.std + epsilon.pS[t]
    pS <- plogis(y.pS)

    return(pS)
}
```

Maturation probability

```
## Parameter values (1st dimension = origin (1 = wild, 2 = stocked), 2nd dimension =
↪  sex (1 = female, 2 = male))
betaMale <- 0.334
betaWild <- -0.068
int0 <- -1.165

betaMale.size <- -0.944
betaWild.size <- -0.380
slp0.size <- 2.556

mu.pM <- cbind(c(int0+betaWild, int0), c(int0+betaWild+betaMale, int0+betaMale)) #
↪  Intercept of logit(pS)
beta2.pM <- cbind(c(slp0.size+betaWild.size, slp0.size),
↪  c(slp0.size+betaWild.size+betaMale.size, slp0.size+betaMale.size)) # Size effect
↪  on logit(pS)
betaY.pM <- 0.413 # Time trend in logit(pS)
var.pM <- 0.547 # Among-year variance in logit(pM)

## Prediction function for smolting probability
pM.pred <- function(size, t, sex, orig){

    # Calculate relevant year from t (t=0 corresponds to 1951)
    year <- t + 1951

    # Scale length and year
    size.std <- (size - 511.4021)/128.2904
```

```
    year.std <- (year - 1984.539)/11.64529

    # Make prediction
    y.pM <- mu.pM[orig,sex] + beta2.pM[orig,sex]*size.std + betaY.pM*year.std +
↪   epsilon.pM[t]
    pM <- plogis(y.pM)

    return(pM)
}
```

Ladder usage

```
## Parameter values (1st value = wild, 2nd value = stocked)
Mu.pL <- c(0.532, 0.476) # Average ladder usage probability
beta1.pL <- 0.271 # Linear discharge effect on logit(p)
beta2.pL <- 0.155 # Linear size effect on logit(p)
beta3.pL <- -0.281 # Interactive discharge:size effect on logit(p)
beta4.pL <- -0.622 # Quadratic size effect on logit(p)
sigma.pL <- 0.383 # SD of random year variation in logit(p)

## Prediction function for annual ladder usage probability
pL.pred <- function(size, t, orig){

    # Scale length
    size.std <- (size - 669.1941)/108.6021

    # Make prediction
    y.pL <- qlogis(Mu.pL[orig]) + beta1.pL*discS.std[t] + beta2.pL*size.std +
↪   beta3.pL*discS.std[t]*size.std + beta4.pL*(size.std^2) + epsilon.pL[t]
    pL <- plogis(y.pL)

    return(pL)
}
```

Fecundity

Fecundity; the potential reproductive output.

```
## Parameter values
mean.log.fec = -7.0218 # Intercept on the log scale
size.eff.fec = 2.3422 # Effect of log(size) on log(fecundity)

fec.pred = function(size){

  y.fec = mean.log.fec + size.eff.fec*log(size)
  fec = exp(y.fec)
  return(fec)
}
```

Early survival

First year, from fertilised egg to parr.

```
m0 <- -log(0.082) # egg background mortality hazard rate

## With an added penalty for being below the dam.
S0.pred <- function(m0, dPenalty.0){

  m0.new <- m0*dPenalty.0
  return(exp(-m0.new))
}


## With an added penalty for having fist-time spawner as parent
S01.pred <- function(m0, dPenalty.0, r){

  m0.new <- m0*dPenalty.0
  return(exp(-m0.new)*r)
}
```

Offspring size

```
## Prediction function for average offspring size (Option 2 - Growth model)
mean.sizeOff.pred <- function(t, i){

  # Calculate increment (linear)
  inc <- h0 + betaYR*t + epsilon.grR[t] + epsilon.grR.i[i]

  # Add increment to size at hatching
  size.next <- mu0 + rnorm(1, inc, 0)
  return(size.next)
}


## Prediction function for river growth distribution
dist.sizeOff.pred <- function(size.next, t, i){

  # Calculate growth and set variance parameter
  mu <- mean.sizeOff.pred(t, i)
  var <- varR.R

  # Calculate densities
  y <- dnorm(size.next, mean = mu, sd = sqrt(var))

  # Scale and return densities
  if(sum(y*dx)==0){
    return (c(rep(0,n-1),1/dx))
  }
  else return(y/sum(y*dx))
  y/sum(y*dx)
}
```

Dam survival

```
## Prediction function for dam survival of smolts
Sdam.pred <- function(size, Mu.mdam, beta2.mdam){

  size.std <- (size-250)/50

  log.mdam <- log(Mu.mdam) + beta2.mdam*size.std
  mdam <- exp(log.mdam)
  return(exp(-mdam))
}
```

3. Create kernels for all transitions

a) Juvenile to Juvenile (upriver and downriver)

```
# in river, pre-smolt
# juveniles remaining juveniles

K.JJ.fun  <-function(t, dam, Mu.mj, beta2.mj){

  # Define matrix
  Smat <- matrix(0, n, n)

  # Fill matrices using vital rate functions
  # survival
  if(dam == 1){ # above dam
    Svec <- Sj.pred(x, Mu.mj, beta2.mj) # juvenile survival above dam, pre-smolt
  }else{ # below dam
    Svec <- Sj.pred(x, Mu.mj, beta2.mj) # same survival as above the dam
  }
  # transition from juvenile to juvenile (do not smolt)
  Tvec <- 1-(pS.pred(x, t)) # 1-smolting probability

  for(i in 1:n){
    Smat[,i] <- Svec[i]*Tvec[i]*dist.grR.pred(x[i], x, t, 1)*dx # those that did not
↪  smolt, still juveniles
  }
  return(Smat)
}
```

b) Juvenile to Subadult (upriver and downriver)

```
# river to lake, at smolting
# juveniles (river) becoming subadults (lake)

K.JS.fun  <-function(t, orig, dam, Mu.mj, beta2.mj, Mu.mdam, beta2.mdam){

  # Define matrix
  Smat <- matrix(0, n, n)

  # Fill matrices using vital rate functions
  # survival
  if(dam == 1){ # above dam
    Svec <- Sj.pred(x, Mu.mj, beta2.mj)*Sdam.pred(x, Mu.mdam, beta2.mdam) # juvenile
↪  surival with added dam mortality going down
  }else{ # below dam
    Svec <- Sj.pred(x, Mu.mj, beta2.mj) # only juvenile survival, as they do NOT
↪  cross the dam going down to the lake
  }
  # transition, juvenile to subadult (those that do smolt)
  Tvec <- pS.pred(x, t) # smolting probability

  for (i in 1:n){
```

```
    Smat[,i] <- Svec[i]*Tvec[i]*dist.grL.pred(x[i], x, t, 1, 0, orig, mature=0)*dx #
↪  those that DID smolt and are now subadults

  }
  return(Smat)
}
```

c) Subadult to Subadult

```
# in lake, pre-maturity
# subadults remaining subadults

K.SS.fun  <-function(t, sex, orig, Mu.mOs, beta2.mOs, l.limit, u.limit, mH.factor){

  # Define matrix
  Smat <- matrix(0, n, n)

  # Fill matrices using vital rate functions
  #survival
  Svec <- Ss.pred(x, t, orig, Mu.mOs, beta2.mOs, l.limit, u.limit, mH.factor) #
↪  subadult survival in lake
  # transition, subadult to subadult (do not mature)
  Tvec <- 1-(pM.pred(x, t, sex, orig)) # 1-maturation probability

  for (i in 1:n){
    Smat[,i] <- Svec[i]*Tvec[i]*dist.grL.pred(x[i], x, t, 1, 0, orig, mature=0)*dx #
↪  those that did NOT mature, and are still subadults
  }
  return(Smat)
}
```

d) Subadult to Spawner (upriver and downriver), first-time spawners

```
# lake to river, after maturation
# subadults becoming first-time spawners

K.SAs1.fun  <-function(t, dam, sex, orig, Mu.mOs, beta2.mOs, l.limit, u.limit,
↪  mH.factor){

  # Define matrix
  Smat <- matrix(0, n, n)

  # Fill matrices using vital rate functions
  # survival
  Svec <- Ss.pred(x, t, orig, Mu.mOs, beta2.mOs, l.limit, u.limit, mH.factor) #
↪  subadult survival

  # transititon number 1, those that do mature
  Tvec1 <- pM.pred(x, t, sex, orig) # maturation probability

  # transition number 2, ladder usage
```

```
  if(dam == 1){ # above dam
    Tvec2 <- pL.pred(x, t, orig) # ladder probability
  }else{ # below dam
    Tvec2 <- 1-pL.pred(x, t, orig)
  }

  # 1. Survive, mature, and grow (given current size, with reproduction cost)
  for (i in 1:n){
    Smat[,i] <- Svec[i]*Tvec1[i]*dist.grL.pred(x[i], x, t, 1, 1, orig, mature = 1)
  }

  # 2. Ascend the ladder or not (given next size, i.e. already grown), those that
  ↪  survive and mature and come up the river
  for(j in 1:n){
    Smat[j,] <- Smat[j,]*Tvec2[j]*dx
  }
  return(Smat)
}
```

e) Spawners to Juvenile (upriver and downriver), first-time spawners

```
# first-time spawners producing juveniles

K.As1J.fun  <-function(t, m0, dam, orig, dPenalty.0, r){

  # Define matrix
  Bmat <- matrix(0, n, n)

  # Fill matrices using vital rate functions
  if(dam == 1){
    Bvec <- fec.pred(x)*S01.pred(m0, 1, r)*0.5
  }else{
    Bvec <- fec.pred(x)*S01.pred(m0, dPenalty.0, r)*0.5
  }

  # Reproduction, egg survival, growth to 1-year size
  for (i in 1:n){
    Bmat[,i] <- Bvec[i]*dist.sizeOff.pred(x, t, 1) * dx
  }
  return(Bmat)
}
```

f) First-time spawner to Non-spawner (upriver and downriver)

```
# river to lake
# first-time spawners (river) becoming non-spawners (lake)
# keep subadult survival for the first-time spawners, Ss.pred (and not adult
↪  survival Sa.pred)

K.As1An.fun  <-function(t, dam, sex, orig, min.size, max.size, l.limit, u.limit,
↪  mH.factor, m01size.factor){
```

```
  # Define matrix
  Smat <- matrix(0, n, n)

  # Fill matrices using vital rate functions
  # transition from first-time spawner to non-spawner (rest year)
  Tvec <- rep(1, n) # 1 because the transition is deterministic (and independent of
↪  the dam)

  # survival
  # above dam: adult survival, below dam: subadult survival
  Svec <- Sa1.pred(x, dam, t, orig, min.size, max.size, spawn.year = 1, l.limit,
↪  u.limit, mH.factor, mO1size.factor)

  # Deterministic transition, survive, and grow (without reproduction cost, i.e. the
  ↪  resting year)
  for (i in 1:n){
    Smat[,i] <- Tvec[i]*Svec[i]*dist.grL.pred(x[i], x, t, 1, 0, orig, mature = 1)*dx
↪  # those that survive spawning and go down to lake
  }
  return(Smat)
}
```

g) Non-spawner to Spawner (upriver and downriver), repeat spawners

```
# lake to river
# adult non-spawners (lake) becoming repeat spawners (river)

K.AnAs.fun  <-function(t, dam, sex, orig, min.size, max.size, l.limit, u.limit,
↪  mH.factor, mO1size.factor){

  # Define matrix
  Smat <- matrix(0, n, n)

  # Fill matrices using vital rate functions
  # transition, going up the ladder or not
  if(dam == 1){ # above dam
    Tvec <- pL.pred(x, t, orig) # the original ladder probability
  }else{ # below dam
    Tvec <- 1-pL.pred(x, t, orig)
  }

  # survival, going from non-spawner to spawner
    Svec <- Sa.pred(x, dam, t, orig, min.size, max.size, spawn.year = 0, l.limit,
↪  u.limit, mH.factor, mO1size.factor) # Survival is 1 here, higher survival for
↪  below dam spawners (dam = 0)

  # 1. Survive and grow (given current size, with reproduction cost, i.e. the year
  ↪  they go up to spawn)
  for (i in 1:n){
    Smat[,i] <- Svec[i]*dist.grL.pred(x[i], x, t, 1, 1, orig, mature = 1)
  }

  # 2. Ascend the ladder or not (given next size, i.e. already grown) - those that
  ↪  survive their resting year and return to the river
```

```
  for(j in 1:n){
    Smat[j,] <- Smat[j,]*Tvec[j]*dx # those that survive the resting year and come
↪   back to spawn
  }

  return(Smat)
}
```

h) Spawners to Juvenile (upriver and downriver)

```
# repeat spawners producing juveniles

K.AsJ.fun  <-function(t, m0, dam, orig, dPenalty.0){

  # Define matrix
  Bmat <- matrix(0, n, n)

  # Fill matrices using vital rate functions
  if(dam == 1){ # above dam
    Bvec <- fec.pred(x)*S0.pred(m0, 1)*0.5 # 0.5 sex ratio
  }else{ # below dam
    Bvec <- fec.pred(x)*S0.pred(m0, dPenalty.0)*0.5 # with an added penalty for
↪   spawning below
  }

  for (i in 1:n){
    Bmat[,i] <- Bvec[i]*dist.sizeOff.pred(x, t, 1) * dx
  }
  return(Bmat)
}
```

i) Repeat spawner to Non-spawner (upriver and downriver)

```
# from dam to the lake
# repeat spawners becoming non-spawners
# then they switch between g) and i) until death

K.AsAn.fun <- function(t, dam, sex, orig, min.size, max.size, l.limit, u.limit,
↪   mH.factor, mO1size.factor){

  # Define matrix
  Smat <- matrix(0, n, n)

  # Fill matrices using vital rate functions
  # transition, going from repeat spawner to non-spawner (rest year)
  Tvec <- rep(1, n) # 1 because the transition is deterministic (and independent of
↪   the dam)

  # survival, depends on if spawning above or below dam
  Svec <- Sa.pred(x, dam, t, orig, min.size, max.size, spawn.year = 1, l.limit,
↪   u.limit, mH.factor, mO1size.factor)
```

```
  # Deterministic transition, survive, and grow (without reproduction cost, i.e.
  ↪  rest year growth)
  for (i in 1:n){
    Smat[,i] <- Tvec[i]*Svec[i]*dist.grL.pred(x[i], x, t, 1, 0, orig, mature = 1)*dx
↪  # those that survive spawning to go back to the lake again
  }
  return(Smat)
}
```

4. Define the transition matrix

```
# 9 kernels (a-i)
# 8 by 8 matrix, the An's are the same from first and repeat spawners
# All entries in the transition matrix are 1, because the size-specific transition
↪  probabilities are already included in the 9 kernels (a-i)

## Create matrix
stg.mat <- matrix(c(1, 0, 0, 1, 0, 0, 1, 0,
                    0, 1, 0, 0, 1, 0, 0, 1,
                    1, 1, 1, 0, 0, 0, 0, 0,
                    0, 0, 1, 0, 0, 0, 0, 0,
                    0, 0, 1, 0, 0, 0, 0, 0,
                    0, 0, 0, 1, 1, 0, 1, 1,
                    0, 0, 0, 0, 0, 1, 0, 0,
                    0, 0, 0, 0, 0, 1, 0, 0), nrow=8, ncol=8, byrow=TRUE)

## Create an additional component matrix filled with only 0's
K0 <- matrix(0, nrow=n, ncol=n)
K.sp <- as(K0, 'sparseMatrix')

## Fill the kernels into the transition matrix
trout.megamatrix = function(K.JJ.u, K.JJ.d, K.JS.u, K.JS.d, K.SS, K.SAs1.u,
↪  K.SAs1.d, K.As1J.u, K.As1J.d, K.As1An.u, K.As1An.d, K.AsAn.u, K.AsAn.d,
↪  K.AnAs.u, K.AnAs.d, K.AsJ.u, K.AsJ.d){ # Kernels updated for first and repeat
↪  spawners

  # Complete IPM kernel - updated
  J.u   <- cbind(K.JJ.u, K0, K0, K.As1J.u, K0, K0, K.AsJ.u, K0)
  J.d   <- cbind(K0, K.JJ.d, K0, K0, K.As1J.d, K0, K0, K.AsJ.d)
  S     <- cbind(K.JS.u, K.JS.d, K.SS, K0, K0, K0, K0, K0)
  As1.u <- cbind(K0, K0, K.SAs1.u, K0, K0, K0, K0, K0)
  As1.d <- cbind(K0, K0, K.SAs1.d, K0, K0, K0, K0, K0)
  An    <- cbind(K0, K0, K0, K.As1An.u, K.As1An.d, K0, K.AsAn.u, K.AsAn.d)
  As.u  <- cbind(K0, K0, K0, K0, K0, K.AnAs.u, K0, K0)
  As.d  <- cbind(K0, K0, K0, K0, K0, K.AnAs.d, K0, K0)

  mega.K <- rbind(J.u, J.d, S, As1.u, As1.d, An, As.u, As.d)

  # Pmatrix only - updated
  J2.u <- cbind(K.JJ.u, K0, K0, K0, K0, K0, K0, K0)
  J2.d <- cbind(K0, K.JJ.d, K0, K0, K0, K0, K0, K0)
  mega.P <- rbind(J2.u, J2.d, S, As1.u, As1.d, An, As.u, As.d)

  # Complete IPM kernel with NA's for empty spots - updated
  y <- matrix(NA, nrow=n, ncol=n)

  J3.u   <- cbind(K.JJ.u, y, y, K.As1J.u, y, y, K.AsJ.u, y)
  J3.d   <- cbind(y, K.JJ.d, y, y, K.As1J.d, y, y, K.AsJ.d)
  S3     <- cbind(K.JS.u, K.JS.d, K.SS, y, y, y, y, y)
  As13.u <- cbind(y, y, K.SAs1.u, y, y, y, y, y)
  As13.d <- cbind(y, y, K.SAs1.d, y, y, y, y, y)
  An3    <- cbind(y, y, y, K.As1An.u, K.As1An.d, y, K.AsAn.u, K.AsAn.d)
  As3.u  <- cbind(y, y, y, y, y, K.AnAs.u, y, y)
  As3.d  <- cbind(y, y, y, y, y, K.AnAs.d, y, y)
```

```
  mega.K.na <- rbind(J3.u, J3.d, S3, As13.u, As13.d, An3, As3.u, As3.d)

  # Full Kernel without 0's in the possible transitions (for plotting) - updated
  x <- 1e-200

  Jx.u   <- cbind(K.JJ.u+x, K0, K0, K.As1J.u+x, K0, K0, K.AsJ.u+x, K0)
  Jx.d   <- cbind(K0, K.JJ.d+x, K0, K0, K.As1J.d+x, K0, K0, K.AsJ.d+x)
  Sx     <- cbind(K.JS.u+x, K.JS.d+x, K.SS+x, K0, K0, K0, K0, K0)
  As1x.u <- cbind(K0, K0, K.SAs1.u+x, K0, K0, K0, K0, K0)
  As1x.d <- cbind(K0, K0, K.SAs1.d+x, K0, K0, K0, K0, K0)
  Anx    <- cbind(K0, K0, K0, K.As1An.u+x, K.As1An.d+x, K0, K.AsAn.u+x, K.AsAn.d+x)
  Asx.u  <- cbind(K0, K0, K0, K0, K0, K.AnAs.u+x, K0, K0)
  Asx.d  <- cbind(K0, K0, K0, K0, K0, K.AnAs.d+x, K0, K0)

  mega.Kx <- rbind(Jx.u, Jx.d, Sx, As1x.u, As1x.d, Anx, Asx.u, Asx.d)

  return(list(IPM = mega.K, Pmatrix = mega.P, IPMna = mega.K.na, IPMx = mega.Kx))
}
```

Function for building IPMs

```
# The following function builds an IPM for a specific size range and binwidth

build.IPM <- function(t, sex, orig, m0, Mu.mj, Mu.mdam, Mu.mOs, beta2.mj,
↪  beta2.mdam, beta2.mOs, dPenalty.0, r, l.limit, u.limit, mH.factor,
↪  mO1size.factor){

  ## Build separate kernels
  # juveniles
  K.JJ.u    <- K.JJ.fun(t, 1, Mu.mj, beta2.mj)
  K.JJ.d    <- K.JJ.fun(t, 0, Mu.mj, beta2.mj)
  # subadults
  K.JS.u    <- K.JS.fun(t, orig, 1, Mu.mj, beta2.mj, Mu.mdam, beta2.mdam)
  K.JS.d    <- K.JS.fun(t, orig, 0, Mu.mj, beta2.mj, Mu.mdam, beta2.mdam)
    K.SS      <- K.SS.fun(t, sex, orig, Mu.mOs, beta2.mOs, l.limit, u.limit,
↪  mH.factor)
  # first-time spawners
  K.SAs1.u  <- K.SAs1.fun(t, 1, sex, orig, Mu.mOs, beta2.mOs, l.limit, u.limit,
↪  mH.factor) # new spawner kernel
  K.SAs1.d  <- K.SAs1.fun(t, 0, sex, orig, Mu.mOs, beta2.mOs, l.limit, u.limit,
↪  mH.factor) # new spawner kernel
  K.As1J.u  <- K.As1J.fun(t, m0, 1, orig, dPenalty.0, r) # new spawner kernel
  K.As1J.d  <- K.As1J.fun(t, m0, 0, orig, dPenalty.0, r) # new spawner kernel
  # non-spawners
  K.As1An.u <- K.As1An.fun(t, 1, sex, orig, Lx, Ux, l.limit, u.limit, mH.factor,
↪  mO1size.factor) # new spawner kernel
  K.As1An.d <- K.As1An.fun(t, 0, sex, orig, Lx, Ux, l.limit, u.limit, mH.factor,
↪  mO1size.factor) # new spawner kernel
    K.AsAn.u  <- K.AsAn.fun(t, 1, sex, orig, Lx, Ux, l.limit, u.limit, mH.factor,
↪  mO1size.factor)
    K.AsAn.d  <- K.AsAn.fun(t, 0, sex, orig, Lx, Ux, l.limit, u.limit, mH.factor,
↪  mO1size.factor)
  # repeat spawners
```

```
  K.AnAs.u  <- K.AnAs.fun(t, 1, sex, orig, Lx, Ux, l.limit, u.limit, mH.factor,
↪  mO1size.factor)
  K.AnAs.d  <- K.AnAs.fun(t, 0, sex, orig, Lx, Ux, l.limit, u.limit, mH.factor,
↪  mO1size.factor)
  K.AsJ.u   <- K.AsJ.fun(t, m0, 1, orig, dPenalty.0)
  K.AsJ.d   <- K.AsJ.fun(t, m0, 0, orig, dPenalty.0)


  ## Build mega-matrix kernel
  IPM <- trout.megamatrix(K.JJ.u, K.JJ.d, K.JS.u, K.JS.d, K.SS, K.SAs1.u, K.SAs1.d,
↪  K.As1J.u, K.As1J.d, K.As1An.u, K.As1An.d, K.AsAn.u, K.AsAn.d, K.AnAs.u,
↪  K.AnAs.d, K.AsJ.u, K.AsJ.d) # Kernels updated

  return(IPM)
}
```

Function for eigen analysis

```
## Function returning lambda, stable structure w and reproductive values v scaled so
↪  that: sum(v*w)=1
wvlambda <- function(Kmat){
  ev <- eigen(Kmat)
  tev <- eigen(t(Kmat))
  lmax <- which.max(Re(ev$values))
  W <- ev$vectors
  V <- tev$vectors
  w <- as.matrix(abs(Re(W[, lmax]))/sum(abs(Re(W[, lmax]))))
  w <- w/(sum(w))
  v <- as.matrix(abs(Re(V[, lmax])))
  v <- v/sum(w*v )
  v <- ifelse(w*v <= 0, 0, v)
  return(list("lambda"=max(Re(ev$values)),"w"=w,"v"=v))
}
```

Function for returning results on lambda, SSD, and elasticities

```
popAnalysis <- function(IPM, t, orig, sex, m0, Mu.mj, Mu.mdam, Mu.mOs, beta2.mj,
↪  beta2.mdam, beta2.mOs, dPenalty.0, r, l.limit, u.limit, mH.factor,
↪  mO1size.factor){

  ## Do eigen analysis
  eigenA <- wvlambda(IPM$IPMx)

  return(list(lambda = eigenA$lambda, SSD = eigenA$w, v = eigenA$v))
}
```

Selected “blackbox” parameter values

```
m0 <- -log(0.082) # egg background mortality hazard rate
Mu.mj <- -log(0.353) # juvenile background mortality
Mu.mdam <- -log(0.75) # smolt dam mortality
Mu.mOs <- 0.7 # subadult background mortality

# size-dependence in juvenile survival
beta2.mj <- -0.2
beta2.mdam <- 0.05

# size-dependence in subadult survival
beta2.mOs <- -0.6

# factor by which to modify background mortality of smaller than average above-dam
↪  spawners (1 = no change, < 1 = reduction, > 1 = increase)
mO1size.factor = 1
```

Adjustable parameters

```
# with below-dam penalties on early survival (1 = no change, < 1 = reduction, > 1 =
↪  increase)
dPenalty.0 = 1

# with first-time spawners fertility penalty (1 = no change, < 1 = reduction, > 1 =
↪  increase)
r = 1

# factor by which to modify harvest mortality (1 = no change, < 1 = reduction, > 1 =
↪  increase)
mH.factor = 0 # no harvest in baseline scenario
```

5. Build the IPMs

```
IPM = build.IPM(t=test.year, sex=1, orig=1, m0, Mu.mj, Mu.mdam, Mu.mOs, beta2.mj,
↪  beta2.mdam, beta2.mOs, dPenalty.0, r, l.limit, u.limit, mH.factor,
↪  mO1size.factor)

# eigen analysis of the IPM
eigen = popAnalysis(IPM = IPM, t, orig, sex, m0, Mu.mj, Mu.mdam, Mu.mOs, beta2.mj,
↪  beta2.mdam, beta2.mOs, dPenalty.0, r, l.limit, u.limit, mH.factor,
↪  mO1size.factor)
```

a) Population growth rate

```
eigen$lambda
```

```
## [1] 1.046184
```

b) Stable size-by-stage distribution

Look at the SSDs of the spawners.

```
# SSD = eigen$SSD # calculate
# colnames(SSD) = "density"
# write.csv(SSD, "SSD_baseline.csv", row.names = F) # save time later
SSD = read.csv("SSD_baseline.csv", header = T, check.names=FALSE)

dist = data.frame(density=SSD,size=rep(x,8),stage=c(rep("Juvenile (above)",n),
↪  rep("Juvenile (below)",n), rep("Subadult",n), rep("First-time spawner
↪  (above)",n), rep("First-time spawner (below)",n), rep("Non-spawner",n),
↪  rep("Repeat spawner (above)",n), rep("Repeat spawner (below)",n)))

dist$stage = factor(dist$stage, levels = c("Juvenile (above)", "Juvenile (below)",
↪  "Subadult", "First-time spawner (above)", "First-time spawner (below)",
↪  "Non-spawner", "Repeat spawner (above)", "Repeat spawner (below)"))

df = as.data.frame(cbind(size = x, harvest = mH.2yr.pred(x, t = 1, orig = 1,l.limit,
↪  u.limit, mH.factor = 1)))
```

```
dist %>%
  filter(grepl('R', stage, ignore.case = T) & !grepl('N', stage, ignore.case = F))
↪  %>% # select only the spawners
  inner_join(df, by = "size") %>% # join the two data frames
  ggplot(., aes(x = size, y = density, fill = stage, col = stage)) +
  geom_density(stat = "identity", alpha = 0.6) +
  scale_fill_viridis_d(option = "viridis", direction = -1, begin = 0.3) +
  scale_color_viridis_d(option = "viridis", direction = -1, begin = 0.3) +
  labs(x = "Size (mm)", y = "SSD", fill = "", col = "") +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0), limits = c(0, 1.1e-4)) +
  theme_cowplot() +
  theme(legend.position = c(0.01,0.9))
```

c) Proportion of spawners and juveniles

```
prop = dist %>%
  filter(grepl('R', stage, ignore.case = T) & !grepl('N', stage, ignore.case = F))
↪  %>% # select only the spawners
  group_by(stage) %>%
  mutate(n = density*fec.pred(x)) %>% # ad hoc number based on SSD
  summarise(offspring = sum(n), SSD = sum(density)) %>%
  mutate(prop.spawner = SSD/sum(SSD), prop.juv = offspring/sum(offspring)) %>%
  ungroup()

ad = prop %>%
  select(stage, prop.spawner) %>%
  ggplot(., aes(x = stage, y = prop.spawner, fill = stage)) +
  geom_bar(stat = "identity", position = position_dodge(), width = 1) +
  geom_text(aes(label = round(prop.spawner, 2)), position = position_dodge(width =
↪  0.9), vjust = -0.3) +
  labs(x = "", y = "Proportion of spawners", fill = "") +
  scale_fill_viridis_d(option = "viridis", direction = -1, begin = 0.3) +
  scale_y_continuous(limits = c(0,1.05), expand = c(0, 0)) +
  theme_cowplot()+
  theme(plot.margin = margin(2,7,-5,2), legend.position = c(0.01, 0.92), legend.text
↪  = element_text(size = 12), axis.text.x = element_blank(), axis.ticks.x =
↪  element_blank())

juv = prop %>%
  select(stage, prop.juv) %>%
  ggplot(., aes(x = stage, y = prop.juv, fill = stage)) +
  geom_bar(stat = "identity", position = position_dodge(), width = 1) +
  geom_text(aes(label = round(prop.juv, 2)), position = position_dodge(width = 0.9),
↪  vjust = -0.3) +
  labs(x = "", y = "Proportion of juveniles", fill = "")+
  scale_fill_viridis_d(option = "viridis", direction = -1, begin = 0.3) +
  scale_y_continuous(limits = c(0,1.05), expand = c(0, 0)) +
  theme_cowplot()+
  theme(plot.margin = margin(2,7,-5,0), legend.position = "none", axis.text.x =
↪  element_blank(), axis.ticks.x = element_blank())

plot_grid(ad, juv,labels = "AUTO", align = "hv", ncol = 2)
```