Appendix S2. Model definitions, model fitting, and model evaluation.

Supporting information for Skagit Chinook IPM

Contents

This is version 0.20.10.23.

1 Background

This appendix describes how we fit the models and evaluated their relative performances. It demonstrates how to load the fish data and environmenal covariates, specify the different models in the **JAGS** software, and fit each one.

All analyses require the R software (v3.4.3 or later) for data retrieval, data processing, and summarizing model results, and the JAGS software (v4.2.0) for Markov chain Monte Carlo (MCMC) simulation. Please note that some of the **R** code below may not work with older versions of **JAGS** due to some changes in the ways that arrays are handled.

We also need a few packages that are not included with the base installation of **R**, so we begin by installing them (if necessary) and then loading them.

```
if(!require("here")) {
  install.packages("here")
  library("here")
}
if(!require("readr")) {
  install.packages("readr")
  library("readr")
}
if(!require("rjags")) {
  install.packages("rjags")
  library("rjags")
}
if(!require("loo")) {
  install.packages("loo")
  library("loo")
}
if(!require("ggplot2")) {
  install.packages("ggplot2")
  library("ggplot2")
}
## set directory locations
datadir <- here("data")
jagsdir <- here("jags")
analdir <- here("analysis")
savedir <- here("analysis/cache")
```

We also need a couple of helper functions.

```
## better round
Re2prec <- function(x, fun = "round", prec = 1) {
  ## 'fun' can be "round", "floor", or "ceiling"
  ## 'prec' is nearest value
  ## (eg, 0.1 is to nearest tenth; 1 is to nearest integer)
  if(prec<=0) { stop("\"prec\" cannot be less than or equal to 0") }
  do.call(map,list(x/prec))*prec
}

## wrapper function to fit JAGS models & rearrange output
fit_jags <- function(model, data, params, inits, ctrl, dir = jagsdir) {
  jm <- jags.model(file.path(jagsdir, model),
                   data,
                   inits,
                   ctrl$chains,
                   ctrl$burn,
                   quiet = TRUE)
  return(coda.samples(jm, params, ctrl$length, ctrl$thin))
}
```

2 User inputs

We begin by supplying values for the following parameters, which we need for model fitting and evaluation.

```
## first & last years of fish data
yr_frst <- 1993
yr_last <- 2016

## min & max adult age classes
age_min <- 2
age_max <- 5
## years (if any) of age-comp to skip; see below
age_skip <- 0

## number of years for run forecasts
n_fore <- 0

## upper threshold for Gelman & Rubin's potential scale reduction factor (Rhat).
Rhat_thresh <- 1.1
```

Next we specify the names of three necessary data files containing the following information:

1. observed total number of adult spawners (escapement) by year;
2. observed total number of subyearling chinook smolts by year
3. observed age composition of adult spawners by year;
4. observed total harvest by year;

```
## 1. file with escapement data
## [n_yrs x 2] matrix of obs counts; 1st col is calendar yr
fn_esc <- "skagit_chinook_esc.csv"

## 2. file with smolt data
## [n_yrs x 2] matrix of obs counts; 1st col is calendar yr
fn_smolt <- "skagit_chinook_smolt.csv"

## 3. file with age comp data
## [n_yrs x (1+A)]; 1st col is calendar yr
fn_age <- "skagit_chinook_age.csv"

## 4. file with harvest data
## [n_yrs x 2] matrix of obs catch; 1st col is calendar yr
fn_harv <- "skagit_chinook_aeq_harvest.csv"
```

3 Loading the fish data

Here we load in the first three data files and do some simple calculations and manipulations. First the spawner data:

```
## escapement
dat_esc <- read_csv(file.path(datadir, fn_esc))
##
dat_esc <- dat_esc[which(dat_esc$year %in% seq(yr_frst,yr_last,1)),]
## years of data
dat_yrs <- dat_esc$year
## number of years of data
n_yrs <- length(dat_yrs)
## log of escapement
ln_dat_esc <- c(log(dat_esc$escapement),rep(NA,n_fore))
```

Next the smolt production data:

```
## subyearling smolt abundance
dat_smolt <- read_csv(file.path(datadir, fn_smolt))
##
dat_smolt <- dat_smolt[which(dat_smolt$year %in% seq(yr_frst,yr_last,1)),]

## log of subyearling smolt abundance
ln_dat_smolt <- c(log(dat_smolt$smolt),rep(NA,n_fore))
```

Next the age composition data:

```
## age comp data
dat_age <- read_csv(file.path(datadir, fn_age))
##
dat_age <- dat_age[which(dat_age$year %in% seq(yr_frst,yr_last,1)),]
## drop year col & first (age_min + age_skip) rows
dat_age <- dat_age[-(1:(age_min+age_skip)),-1]
## num of age classes
A <- age_max - age_min + 1
## add row(s) of NA's for forecast years
if(n_fore > 0) {
  dat_age <- rbind(dat_age,
                   matrix(0, n_fore, A,
                          dimnames = list(n_yrs+seq(n_fore),
                                          colnames(dat_age))))
}
## total num of age obs by cal yr
dat_age[,"sum"] <- apply(dat_age, 1, sum)
## row indices for any years with no obs age comp
```

```
idx_NA_yrs <- which(dat_age$sum<A, TRUE)
## replace 0's in yrs w/o any obs with NA's
dat_age[idx_NA_yrs,(1:A)] <- NA
## change total in yrs w/o any obs from 0 to A to help dmulti()
dat_age[idx_NA_yrs,"sum"] <- A
## convert class
dat_age <- as.matrix(dat_age)
```

And then the harvest data:

```
## harvest
dat_harv <- read_csv(file.path(datadir, fn_harv))
##
dat_harv <- dat_harv[which(dat_harv$year %in% seq(yr_frst,yr_last,1)),]
## drop year col & first age_max rows
dat_harv <- c(dat_harv$tot_aeq, rep(0,n_fore))
```

4 Loading the covariates

Our analysis investigates the effects of 4 covariates on the population's instrinic growth rate:

1. Maximum river discharge in winter;
2. Minimum river discharge in summer;
3. North Pacific Gyre Oscillation;

All of the covariates are contained in the file `/data/skagit_sthd_covars.csv`. We will load and then standardize them to have zero-mean and unit-variance.

```
## covariate(s)
dat_cvrs <- read_csv(file.path(datadir, "skagit_chinook_covars.csv"))
## drop year col
dat_cvrs <- dat_cvrs[,-1]
## transform the covariates to z-scores
scl_cvrs <- as.matrix(scale(dat_cvrs))
## total number of covariates
n_cov <- dim(scl_cvrs)[2]
```

5 Specifying the models in JAGS

Now we can specify the various models in JAGS. We fit a total of 4 different models, which we outline below, based on the 2 different process models with and without and covariates.

5.1 Models without covariates

5.1.1 Beverton-Holt

```
cat("

model {

  ##--------
  ## PRIORS
  ##--------

  ## 1. adult to smolt productivity
  pas ~ dnorm(0,0.001) T(0,);
  mu_pas <- log(pas)

  ## 2. smolt to adult recruit productivity
  psa ~ dnorm(0,0.001) T(0,);
  mu_BH_psa <- log(psa)
  E_BH_psa <- mu_BH_psa + sigma_r/(2 - 2*phi^2);

  ## strength of dens depend for smolt to adult stage
  beta_inv ~ dnorm(0, 1e-9) T(0,);
  beta <- 1/beta_inv;

  ## AR(1) coef for proc errors
  phi ~ dunif(-0.999,0.999);

  ## process variance for adult recruits model
  sigma_r ~ dnorm(0, 2e-2) T(0,);
  tau_r <- 1/sigma_r;

  ## innovation in first year
  innov_1 ~ dnorm(0,tau_r*(1-phi*phi));

  ## obs variance for spawners
  tau_sp <- 1/sigma_sp;
  sigma_sp ~ dnorm(0, 0.001) T(0,);

  ## obs variance for spawners
  tau_sm <- 1/sigma_sm;
  sigma_sm ~ dnorm(0, 0.001) T(0,);

  ## unprojectable early recruits;
  ## hyper mean across all popns
  Rec_mu ~ dnorm(0,0.001);
  ## hyper SD across all popns
  Rec_sig ~ dunif(0,100);
```

```
## precision across all popns
Rec_tau <- pow(Rec_sig,-2);
## multipliers for unobservable total runs
  ttl_run_mu ~ dunif(1,5);
  ttl_run_tau ~ dunif(1,20);

## get total cal yr returns for first age_min yrs
for(i in 1:(age_min+age_skip)) {
      ln_tot_Run[i] ~ dnorm(ttl_run_mu*Rec_mu,Rec_tau/ttl_run_tau);
      tot_Run[i] <- exp(ln_tot_Run[i]);
}

## maturity schedule
## unif vec for Dirch prior
theta <- c(1,10,10,5)
## hyper-mean for maturity
pi_eta ~ ddirch(theta);
## hyper-prec for maturity
pi_tau ~ dnorm(0, 0.01) T(0,);
for(t in 1:(n_yrs-age_min+n_fore)) { pi_vec[t,1:A] ~ ddirch(pi_eta*pi_tau) }

## estimated harvest rate
for(t in 1:(n_yrs+n_fore)) { h_rate[t] ~ dunif(0,1) }

##------------
## LIKELIHOOD
##------------
## 1st brood yr requires different innovation
## predicted adult recruits in BY t. Modelling as density dependent relationship given that

ln_BH_psa[1] <- mu_BH_psa;
E_ln_Rec[1] <- ln_BH_psa[1] + E_ln_smolt[1] - log(1 + beta*smolt[1]) + phi*innov_1;
tot_ln_Rec[1] ~ dnorm(E_ln_Rec[1],tau_r);
res_ln_Rec[1] <- tot_ln_Rec[1] - E_ln_Rec[1];

## median of total recruits
tot_Rec[1] <- exp(tot_ln_Rec[1]);

## survival = Rec/Smolt
ln_RS[1] <- tot_ln_Rec[1] - E_ln_smolt[1];

## brood-yr recruits by age
for(a in 1:A) {
  Rec[1,a] <- tot_Rec[1] * pi_vec[1,a];
}

## brood years 2:(n_yrs-age_min)
for(t in 2:(n_yrs-age_min+n_fore)) {
```

```
## predicted adult recruits in BY t. Modelling as density dependent relationship given that
ln_BH_psa[t] <- mu_BH_psa;
E_ln_Rec[t] <- ln_BH_psa[t] + E_ln_smolt[t] - log(1 + beta*smolt[t]) + phi*res_ln_Rec[t-1];
tot_ln_Rec[t] ~ dnorm(E_ln_Rec[t],tau_r)
res_ln_Rec[t] <- tot_ln_Rec[t] - E_ln_Rec[t];
## median of total recruits
tot_Rec[t] <- exp(tot_ln_Rec[t]);

## survival = Rec/Smolt
ln_RS[t] <- tot_ln_Rec[t] - E_ln_smolt[t];
  ## brood-yr recruits by age
  for(a in 1:A) {
    Rec[t,a] <- tot_Rec[t] * pi_vec[t,a];
  }
} ## end t loop over year

## get predicted calendar year returns by age
## matrix Run has dim [(n_yrs-age_min) x A]
## step 1: incomplete early broods
## first cal yr of this grp is first brood yr + age_min + age_skip
for(i in 1:(age_max-age_min-age_skip)) {
  ## projected recruits
  for(a in 1:(i+age_skip)) {
    Run[i,a] <- Rec[(age_skip+i)-a+1,a];
  }
  ## imputed recruits
  for(a in (i+1+age_skip):A) {
    lnRec[i,a] ~ dnorm(Rec_mu,Rec_tau);
    Run[i,a] <- exp(lnRec[i,a]);
  }
  ## total run size
  tot_Run[i+age_min+age_skip] <- sum(Run[i,1:A]);
  ## predicted age-prop vec for multinom
  for(a in 1:A) {
    age_v[i,a] <- Run[i,a] / tot_Run[i+age_min];
  }
  ## multinomial for age comp
  dat_age[i,1:A] ~ dmulti(age_v[i,1:A],dat_age[i,A+1]);
  lp_age[i] <- logdensity.multi(dat_age[i,1:A],age_v[i,1:A],dat_age[i,A+1]);
}

## step 2: info from complete broods
## first cal yr of this grp is first brood yr + age_max
for(i in (A-age_skip):(n_yrs-age_min-age_skip+n_fore)) {
  for(a in 1:A) {
    Run[i,a] <- Rec[(age_skip+i)-a+1,a];
  }
```

```
    ## total run size
    tot_Run[i+age_min+age_skip] <- sum(Run[i,1:A]);
    ## predicted age-prop vec for multinom
    for(a in 1:A) {
      age_v[i,a] <- Run[i,a] / tot_Run[i+age_min];
    }
    ## multinomial for age comp
    dat_age[i,1:A] ~ dmulti(age_v[i,1:A],dat_age[i,A+1]);
    lp_age[i] <- logdensity.multi(dat_age[i,1:A],age_v[i,1:A],dat_age[i,A+1]);
  }

  ## get predicted calendar year spawners and smolts
  ## first cal yr is first brood yr
  for(t in 1:(n_yrs+n_fore)) {
    ## obs model for spawners
    # Sp[t] <- max(10,tot_Run[t] - dat_harv[t]);
    est_harv[t] = h_rate[t] * tot_Run[t];
    dat_harv[t] ~ dlnorm(log(est_harv[t]), 20);
    Sp[t] = tot_Run[t] - est_harv[t];
    ln_Sp[t] <- log(Sp[t]);
    ln_dat_esc[t] ~ dnorm(ln_Sp[t], tau_sp);
    lp_esc[t] <- logdensity.norm(ln_dat_esc[t],ln_Sp[t], tau_sp);

    ## obs model for smolts
    ## predicted smolts in BY t
    ## predicted recruits in BY t
    ln_pas[t] <- mu_pas;
    E_ln_smolt[t] <- ln_pas[t] + ln_Sp[t]
    smolt[t] <- exp(E_ln_smolt[t])

    ## observation  model for smolt production
    ln_dat_smolt[t] ~ dnorm(E_ln_smolt[t], tau_sm);
    lp_smolt[t] <- logdensity.norm(ln_dat_smolt[t],E_ln_smolt[t], tau_sm);

    ## survival = smolts/spawners
    ln_ss[t] <- E_ln_smolt[t] - ln_Sp[t]

  }

} ## end model description

", file=file.path(jagsdir, "IPM_BH_AR.txt"))
```

6 Fitting the models

Before fitting the model in JAGS, we need to specify:

1. the data and indices that go into the model;
2. the model parameters and states that we want JAGS to return;
3. the MCMC control parameters.

```
## 1. Data to pass to JAGS:
dat_jags <- list(dat_age = dat_age,
                 ln_dat_esc = ln_dat_esc,
                 ln_dat_smolt = ln_dat_smolt,
                 dat_harv = dat_harv,
                 A = A,
                 age_min = age_min,
                 age_max = age_max,
                 age_skip = age_skip,
                 n_yrs = n_yrs,
                 n_fore = n_fore)

## 2. Model params/states for JAGS to return:
##
##    These are specific to the process model,
##    so we define them in 'par_jags' below.

## 3. MCMC control params:
mcmc_ctrl <- list(
  chains = 4,
  length = 5e5,
  burn = 2e5,
  thin = 400
)
## total number of MCMC samples after burnin
mcmc_samp <- mcmc_ctrl$length*mcmc_ctrl$chains/mcmc_ctrl$thin
```

6.1 Models without covariates

Please note that the following code takes ~80 min to run on a quad-core machine with 3.5 GHz Intel processors.

```
## empty list for fits

## function for inits
init_vals_AR <- function() {
    list(pas = 8, psa = 2,
         beta_inv = exp(mean(ln_dat_smolt, na.rm = TRUE)),
         pi_tau = 10,
         pi_eta = rep(1,A),
         pi_vec = matrix(c(0.020,0.219,0.581,0.179),
                         n_yrs-age_min+n_fore, A,
                         byrow = TRUE),
```

```
          Rec_mu = log(1000),
          Rec_sig = 0.1,
          tot_ln_Rec = rep(log(1000), n_yrs - age_min + n_fore),
          innov_1 = 0,
          phi = 0.5)
}
```

6.1.1 Beverton-Holt

```
## params/states to return
par_jags <- c("pas","mu_pas","psa","E_BH_psa","mu_BH_psa",
              "beta",
              "Sp","smolt","Rec","tot_ln_Rec","ln_RS",
              "pi_eta","pi_tau",
              "sigma_r","sigma_sp","sigma_sm","res_ln_Rec",
              "lp_age","lp_esc","lp_smolt")
## fit model & save it
mod_fit <- fit_jags("IPM_BH_AR.txt", dat_jags, par_jags, init_vals_AR, mcmc_ctrl)
```