R2. Model fitting and evaluation

2020 - 2021 Skagit River steelhead forecast.

Contents

This is version 0.20.12.07.

```
## [1] TRUE
```

Requirements

All analyses require the R software (v3.4.3) 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, map = "round", prec = 1) {
  ## 'map' can be "round", "floor", or "ceiling"
  ## 'prec' is nearest value (eg, 0.1 means to nearest tenth; 1 gives normal behavior)
  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))
}
```

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 <- 1978
yr_last <- 2020

## min & max adult age classes
age_min <- 3
age_max <- 8
## years (if any) of age-comp to skip; see below
age_skip <- 0

## number of years ahead for run forecasts
n_fore <- 1

## number of recent year forecasts
n_forecasts <- 5

## 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 age composition of adult spawners by year;
3. 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_sthd_esc.csv"

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

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

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))
## 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 age composition data:

```
## age comp data
dat_age <- read_csv(file.path(datadir, fn_age))
## num of age classes
A <- age_max - age_min + 1
## drop year col & first age_min+age_skip rows
dat_age <- dat_age[-(1:(age_min+age_skip)),-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))
## drop year col & first age_max rows
dat_harv <- c(dat_harv$catch,rep(NA,n_fore))
```

Loading the covariates

Our analysis investigates 5 covariates as possible drivers of 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.

```
dat_cvrs <- read_csv(file.path(datadir, "skagit_sthd_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(dat_cvrs)[2]
```

Specifying models in JAGS

Now we can specify the model in JAGS. We fit a total one model, which we outline below, based on a beverton holt process model with covariates.

Beverton-Holt with covars

```
cat("
    model {

    ##--------
    ## PRIORS
    ##--------
    ## alpha = intrinsic productivity
    alpha ~ dnorm(0,0.001) T(0,);
    mu_BH_a <- log(alpha);
    E_BH_a <- mu_BH_a + sigma_r/(2 - 2*phi^2);

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

    ## covariate effects
    for(i in 1:n_cov) { gamma[i] ~ dnorm(0,0.01) }

    ## AR(1) coef for proc errors
    phi ~ dunif(-0.999,0.999);
    #phi <- 0;

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

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

    ## obs variance for spawners
    tau_s <- 1/sigma_s;
    sigma_s ~ 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(1,10,10,5,1,1)
## 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
##------------
## predicted recruits in BY t
covar[1] <- inprod(gamma,mod_cvrs[1,]);
ln_BH_a[1] <- mu_BH_a + covar[1];
E_ln_Rec[1] <- ln_BH_a[1] + ln_Sp[1] - log(1 + beta*Sp[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]);

## R/S
ln_RS[1] <- tot_ln_Rec[1] - ln_Sp[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 recruits in BY t
covar[t] <- inprod(gamma, mod_cvrs[t,]);
ln_BH_a[t] <- mu_BH_a + covar[t];
E_ln_Rec[t] <- ln_BH_a[t] + ln_Sp[t] - log(1 + beta*Sp[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]);

## R/S
ln_RS[t] <- tot_ln_Rec[t] - ln_Sp[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] <- ifelse(i < n_yrs-age_min-age_skip+n_fore, logdensity.multi(dat_age[i,1:A],age_v[i,1:A]
}
```

```
    ## get predicted calendar year spawners
    ## 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] = ifelse(t > n_yrs,1,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_s);

    lp_esc[t] <- ifelse(t < n_yrs + 1,logdensity.norm(ln_dat_esc[t],ln_Sp[t], tau_s),0);
    }
  } ## end model description

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

Beverton-Holt with covars

```
cat("
model {

  ##--------
  ## PRIORS
  ##--------
  ## alpha = intrinsic productivity
  alpha ~ dnorm(0,0.001) T(0,);
  mu_BH_a <- log(alpha);
  E_BH_a <- mu_BH_a + sigma_r/(2 - 2*phi^2);

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

  ## covariate effects
  for(i in 1:n_cov) { gamma[i] ~ dnorm(0,0.01) }

  ## AR(1) coef for recruitment residual
  # phi ~ dunif(-0.999,0.999);
  #phi <- 0;
  phi_prior ~ dbeta(2,2);
  phi <- phi_prior*2-1;

  ## MA(1) coef recruitment residual
  theta_res_prior ~ dbeta(2,2);
  theta_res <- theta_res_prior*2-1;

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

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

## obs variance for spawners
tau_s <- 1/sigma_s;
sigma_s ~ 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(1,10,10,5,1,1)
## 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
##------------
## predicted recruits in BY t
covar[1] <- inprod(gamma,mod_cvrs[1,]);
ln_BH_a[1] <- mu_BH_a + covar[1];
res_ln_Rec[1] <- 0;
w[1] <- innov_1;
tot_ln_Rec[1] <- ln_BH_a[1] + ln_Sp[1] - log(1 + beta*Sp[1]) + w[1];
## median of total recruits
tot_Rec[1] <- exp(tot_ln_Rec[1]);

## R/S
ln_RS[1] <- tot_ln_Rec[1] - ln_Sp[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 recruits in BY t
  covar[t] <- inprod(gamma, mod_cvrs[t,]);
  ln_BH_a[t] <- mu_BH_a + covar[t];
  res_ln_Rec[t] ~ dnorm(0, tau_r);
  w[t] <- phi * w[t-1] + theta_res * res_ln_Rec[t-1] + res_ln_Rec[t] * sigma_r;
  tot_ln_Rec[t] <- ln_BH_a[t] + ln_Sp[t] - log(1 + beta*Sp[t]) + w[t];
  ## median of total recruits
  tot_Rec[t] <- exp(tot_ln_Rec[t]);
  ## R/S
  ln_RS[t] <- tot_ln_Rec[t] - ln_Sp[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]);
    lp_age[i] <- ifelse(i < n_yrs-age_min-age_skip+n_fore,
    logdensity.multi(dat_age[i,1:A],age_v[i,1:A],dat_age[i,A+1]),0)
  }

  ## get predicted calendar year spawners
  ## 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_s);
    lp_esc[t] <- ifelse(t < n_yrs + 1,logdensity.norm(ln_dat_esc[t],ln_Sp[t], tau_s),0);
  }

} ## end model description

  ", file=file.path(jagsdir, "IPM_BH_cov_MA1_AR1.txt"))
```

Fitting the models and generating the one year ahead forecasts

Before fitting the model in JAGS, we need to specify the MCMC control parameters.

```
## empty list for fits
n_mods <- 2
mod_fits <- vector("list", n_mods)
```

```
## 1. Data to pass to JAGS
dat_jags <- list(dat_age = dat_age,
                 ln_dat_esc = ln_dat_esc,
                 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. 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
```

Model with all covariates

Please note that the following code takes ~20 min to run on a quad-core machine with 3.5 GHz Intel processors.

```
## set of multi-covariate models
cset <- colnames(scl_cvrs)
dat_jags$n_cov <- length(cset)
dat_jags$mod_cvrs <- scl_cvrs[, cset]
```

First, we will fit a beverton holt model assuming MA1 and AR1 errors

```
## function for inits
init_vals_cov <- function() {
  list(alpha = 5,
       beta_inv = exp(mean(ln_dat_esc, na.rm = TRUE)),
       gamma = rep(0, 3),
       pi_tau = 10,
       pi_eta = rep(1,A),
       pi_vec = matrix(c(0.01,0.35,0.47,0.15,0.01,0.01),
                       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),
       phi_prior = 0.75,theta_res_prior = 0.75,
       innov_1 = 0)
}

## params/states to return
par_jags <- c("alpha","E_BH_a","ln_BH_a",
              "beta",
              "gamma",
              "Sp","Rec","tot_ln_Rec","ln_RS",
              "pi_eta","pi_tau",
              "sigma_r","sigma_s","res_ln_Rec","w","theta_res","phi",
              "lp_age","lp_esc"
              )

cat("Count =", 1, "; Time =", round(((proc.time()-timer_start)/60)["elapsed"], 1), "\n",
          file="cnt_time.txt", append=TRUE)
```

```
## fit model & save it
mod_fits[[1]] <- fit_jags("IPM_BH_cov_MA1_AR1.txt", dat_jags, par_jags,
                          init_vals_cov, mcmc_ctrl)
```

Model diagnostics

Here is a table of the Gelman & Rubin statistics (R_{hat}) for the estimated parameters. Recall that we set an upper threshold of 1.1, so values larger than that deserve some additional inspection.

```
## params of interest
par_conv <- c("alpha","beta",paste0("gamma[",seq(3),"]"),
              "sigma_r","sigma_s","pi_tau","theta_res",paste0("pi_eta[",seq(A-1),"]"))
## Gelman-Rubin
gelman.diag(mod_fits[[1]][,par_conv])
```

```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## alpha           1.10       1.25
## beta            1.11       1.27
## gamma[1]        1.00       1.00
## gamma[2]        1.00       1.01
## gamma[3]        1.00       1.00
## sigma_r         1.01       1.03
## sigma_s         1.00       1.01
## pi_tau          1.01       1.03
## theta_res       1.03       1.09
## pi_eta[1]       1.00       1.01
## pi_eta[2]       1.00       1.00
## pi_eta[3]       1.00       1.00
## pi_eta[4]       1.00       1.00
## pi_eta[5]       1.00       1.01
##
## Multivariate psrf
##
## 1.11
```

```
## Autocorrelation
t(round(autocorr.diag(mod_fits[[1]][,par_conv],
                      lags = seq(mcmc_ctrl$thin, 4*mcmc_ctrl$thin, mcmc_ctrl$thin),
                      relative=FALSE), 2))
```

```
##           Lag 400 Lag 800 Lag 1200 Lag 1600
## alpha        0.93    0.86     0.80     0.75
## beta         0.89    0.82     0.76     0.71
## gamma[1]     0.00   -0.01     0.00    -0.01
## gamma[2]     0.07    0.05     0.05     0.02
## gamma[3]     0.00    0.01     0.01     0.01
## sigma_r      0.53    0.39     0.32     0.23
## sigma_s      0.04    0.01     0.01     0.00
## pi_tau       0.10    0.07     0.05     0.05
## theta_res    0.81    0.71     0.65     0.59
```

```
## pi_eta[1]    0.23    0.14     0.08     0.09
## pi_eta[2]    0.02    0.02    -0.01     0.01
## pi_eta[3]    0.01    0.02    -0.03     0.03
## pi_eta[4]    0.02   -0.01    -0.03     0.01
## pi_eta[5]    0.17    0.10     0.05     0.06
```

next we will fit a beverton holt model assuming AR1 errors only

```
## function for inits
init_vals_cov <- function() {
  list(alpha = 5,
       beta_inv = exp(mean(ln_dat_esc, na.rm = TRUE)),
       gamma = rep(0, 3),
       pi_tau = 10,
       pi_eta = rep(1,A),
       pi_vec = matrix(c(0.01,0.35,0.47,0.15,0.01,0.01),
                       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),
       phi = 0.5,
       innov_1 = 0)
}

## params/states to return
par_jags <- c("alpha","E_BH_a","ln_BH_a",
              "beta",
              "gamma",
              "Sp","Rec","tot_ln_Rec","ln_RS",
              "pi_eta","pi_tau",
              "sigma_r","sigma_s","res_ln_Rec",
              "lp_age","lp_esc","phi"
              )

cat("Count =", 2, "; Time =", round(((proc.time()-timer_start)/60)["elapsed"], 1), "\n",
          file="cnt_time.txt", append=TRUE)

## fit model & save it
mod_fits[[2]] <- fit_jags("IPM_BH_cov_AR.txt", dat_jags, par_jags,
                          init_vals_cov, mcmc_ctrl)
```

Model diagnostics

Here is a table of the Gelman & Rubin statistics (R_{hat}) for the estimated parameters. Recall that we set an upper threshold of 1.1, so values larger than that deserve some additional inspection.

```
## params of interest
par_conv <- c("alpha","beta",paste0("gamma[",seq(3),"]"),
              "sigma_r","sigma_s","pi_tau","phi",paste0("pi_eta[",seq(A-1),"]"))
## Gelman-Rubin
gelman.diag(mod_fits[[2]][,par_conv])
```

```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## alpha           1.00       1.00
## beta            1.00       1.00
## gamma[1]        1.00       1.00
## gamma[2]        1.00       1.00
## gamma[3]        1.00       1.00
## sigma_r         1.00       1.00
## sigma_s         1.00       1.00
## pi_tau          1.01       1.03
## phi             1.00       1.00
## pi_eta[1]       1.01       1.03
## pi_eta[2]       1.00       1.00
## pi_eta[3]       1.00       1.00
## pi_eta[4]       1.00       1.00
## pi_eta[5]       1.00       1.01
##
## Multivariate psrf
##
## 1.2
```

```
## Autocorrelation
t(round(autocorr.diag(mod_fits[[2]][,par_conv],
                      lags = seq(mcmc_ctrl$thin, 4*mcmc_ctrl$thin, mcmc_ctrl$thin),
                      relative=FALSE), 2))
```

```
##           Lag 400 Lag 800 Lag 1200 Lag 1600
## alpha        0.28    0.08     0.04     0.03
## beta         0.28    0.07     0.04     0.03
## gamma[1]     0.01    0.00     0.00     0.00
## gamma[2]     0.00   -0.01     0.03    -0.02
## gamma[3]     0.03   -0.02     0.00     0.00
## sigma_r      0.01    0.02    -0.01     0.00
## sigma_s      0.06    0.03     0.05     0.03
## pi_tau       0.13    0.09     0.07     0.07
## phi          0.00   -0.02     0.01     0.01
## pi_eta[1]    0.24    0.14     0.10     0.07
## pi_eta[2]   -0.02    0.01     0.01     0.03
## pi_eta[3]    0.01    0.00    -0.01     0.02
## pi_eta[4]    0.01    0.00     0.00     0.02
## pi_eta[5]    0.13    0.09     0.04     0.03
```

Model selection

Via `loo()` and `compare()` with full table of results. Note that `elpd_diff` will be negative (positive) if the expected predictive accuracy for the first (second) model is higher.

```
LOOIC <- vector("list", n_mods)
## extract log densities from JAGS objects
for(i in 1:n_mods) {
  ## convert mcmc.list to matrix
```

```
  tmp_lp <- as.matrix(mod_fits[[i]])
  ## extract pointwise likelihoods
  tmp_lp <- tmp_lp[,grepl("lp_", colnames(tmp_lp))]
  ## if numerical underflows, convert -Inf to 5% less than min(likelihood)
  if(any(is.infinite(tmp_lp))) {
    tmp_lp[is.infinite(tmp_lp)] <- NA
    tmp_min <- min(tmp_lp, na.rm = TRUE)
    tmp_lp[is.na(tmp_lp)] <- tmp_min * 1.05
  }
  ## calculate LOOIC
  LOOIC[[i]] <- loo(tmp_lp)
}

## compute pseudo weights
model_weights <- loo_model_weights(LOOIC, method = "pseudobma",optim_method = "BFGS", optim_control = li

## LOOIC for all data
tbl_LOOIC <- round(loo_compare(x = LOOIC), 2)
rownames(tbl_LOOIC) <- sub("model", "", rownames(tbl_LOOIC))
tbl_LOOIC <- tbl_LOOIC[order(as.numeric(rownames(tbl_LOOIC))), ]
tbl_LOOIC <- cbind(model = c("B-H","B-H"),
                   error = c("MA1_AR1","AR1"),
                   as.data.frame(tbl_LOOIC),pseudo_bma_weight = as.matrix(model_weights))
tbl_LOOIC[order(tbl_LOOIC[,"looic"]), ]
```

```
##   model   error elpd_diff se_diff elpd_loo se_elpd_loo  p_loo se_p_loo  looic se_looic
## 2   B-H     AR1      0.00    0.00  -401.95       49.27 146.41    11.48 803.90    98.54
## 1   B-H MA1_AR1     -4.94    4.18  -406.89       49.69 145.41    13.80 813.77    99.38
##   pseudo_bma_weight
## 2          0.892781
## 1          0.107219
```

```
## best model
best_i <- which(tbl_LOOIC[,"looic"] == min(tbl_LOOIC[,"looic"]))
best_fit <- mod_fits[[best_i]]
```

These results show that the Beverton-Holt model with AR1 error has the lowest LOOIC value. All results will be derived from model averaging based on pseudo bayesian model average weights.