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

Contents

This is version 0.19.03.21.

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("knitr")) {
  install.packages("knitr")
  library("knitr")
}
if(!require("kableExtra")) {
  install.packages("kableExtra")
  library("kableExtra")
}
## set directory locations
datadir <- here("data")
jagsdir <- here("jags")
```

We also need a couple of helper functions.

```
## better round
Re2prec <- function(x, map = "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 a JAGS model
fit_jags <- function(model, data, params, inits, ctrl, dir = jagsdir) {
  jm <- jags.model(file.path(jagsdir, model),
                   data,
                   inits,
                   n.chains = ctrl$chains,
                   n.adapt = 0,
                   quiet = TRUE)
  adp <- FALSE
  while(!adp) {
    adp <- adapt(jm, n.iter = 1000)
  }
  update(jm, ctrl$burn, progress.bar = "none")
  return(coda.samples(jm, params, ctrl$length, ctrl$thin))
}

## inits function for base model
init_vals_AR <- function() {
  list(alpha = 5,
```

```
       beta_inv = exp(mean(ln_dat_esc, na.rm = TRUE)),
       pi_tau = 10,
       pi_eta = rep(1,A),
       pi_vec = matrix(c(0.05,0.5,0.4,0.05),
                       n_yrs-age_min, A,
                       byrow = TRUE),
       Rec_mu = log(1000),
       Rec_sig = 0.1,
       sigma_r = 0.5,
       sigma_s = 0.1,
       tot_ln_Rec = rep(log(1000), n_yrs - age_min),
       innov_1 = 0,
       phi = 0.5)
}

## inits function for cov models
init_vals_cov <- function() {
  list(alpha = 5,
       beta_inv = exp(mean(ln_dat_esc, na.rm = TRUE)),
       gamma = 0,
       pi_tau = 10,
       pi_eta = rep(1,A),
       pi_vec = matrix(c(0.05,0.5,0.4,0.05),
                       n_yrs-age_min, A,
                       byrow = TRUE),
       Rec_mu = log(1000),
       Rec_sig = 0.1,
       tot_ln_Rec = rep(log(1000), n_yrs - age_min),
       # phi = 0.5,
       innov_1 = 0)
}

## estimate LOOIC
looic <- function(jags_obj, mcmc_ctrl) {
  ## convert mcmc.list to matrix
  tmp_lp <- as.matrix(jags_obj)
  ## 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
  }
  ## effective sample size
  r_eff <- relative_eff(exp(tmp_lp),
                        chain_id = rep(seq(mcmc_ctrl$chains),
                                       each = mcmc_ctrl$length / mcmc_ctrl$thin))
  ## calculate LOOIC
  looic <- loo(tmp_lp, r_eff = r_eff)
  return(looic)
}
```

2 User inputs

We begin by supplying values for the minimum and maximum ages of spawning adults, plus some information for the model code and evaluation.

```
## min & max adult age classes
age_min <- 3
age_max <- 6

## file where to save JAGS model
fn_jags <- "Willamette_Chin_SR_flow_models_mainstem_JAGS.txt"

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

Next we specify the names of five 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;
4. flow covariates by year;
5. metadata for flow covariates.

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

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

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

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

## 5. covariate metadata
cov_meta_file <- "chin_cov_metadata.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))
## use total counts
```

```
dat_esc <- dat_esc[dat_esc$group=="total",-1]
## years of data
dat_yrs <- dat_esc$year
## number of years of data
n_yrs <- length(dat_yrs)
## get first & last years
yr_frst <- min(dat_yrs)
yr_last <- max(dat_yrs)
## log of escapement
ln_dat_esc <- log(dat_esc[,-1])
```

Next the age composition data:

```
## age comp data
dat_age <- read.csv(file.path(datadir, fn_age))
## drop first age_min rows; drop site & year col
dat_age <- dat_age[-(1:(age_min)), -1]
## num of age classes
A <- age_max-age_min+1
## 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)
if(length(idx_NA_yrs) > 0) {
  ## 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))
## trim to correct years & drop year col
dat_harv <- dat_harv[dat_harv$year>=yr_frst & dat_harv$year<=yr_last,-1]
```

4 Loading the covariates

Load the metadata file containing all of the specifications for the covariates to be used.

```
cov_meta <- read.csv(file.path(datadir, cov_meta_file), stringsAsFactors = FALSE)
cov_meta$code <- gsub("\"","",cov_meta$code)
cov_meta$begin <- gsub("\"","",cov_meta$begin)
cov_meta$end <- gsub("\"","",cov_meta$end)
```

Load the saved covariates.

```
cov_flow <- read.csv(file.path(datadir, fn_cov))[,-1]
n_cov <- dim(cov_flow)[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 Ricker model without covariates

```
cat("

model {

  ##--------
  ## PRIORS
  ##--------
  ## alpha = exp(a) = intrinsic productivity
  alpha ~ dnorm(0,0.01) T(0,);
  mu_Rkr_a <- log(alpha);
  E_Rkr_a <- mu_Rkr_a + sigma_r/(2 - 2*phi^2);

  ## strength of dens depend
  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 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_s <- 1/sigma_s;
  sigma_s ~ dnorm(0, 0.001) T(0,);

  ## maturity schedule
  ## unif vec for Dirch prior
  theta <- c(2,20,20,1)
  ## 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)) { pi_vec[t,1:A] ~ ddirch(pi_eta*pi_tau) }

  ## 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)) {
      ln_tot_Run[i] ~ dnorm(ttl_run_mu*Rec_mu,Rec_tau/ttl_run_tau);
      tot_Run[i] <- exp(ln_tot_Run[i]);
}

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

##------------
## LIKELIHOOD
##------------
## 1st brood yr requires different innovation
## predicted recruits in BY t
ln_Rkr_a[1] <- mu_Rkr_a;
E_ln_Rec[1] <- ln_Rkr_a[1] + ln_Sp[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)) {
  ## predicted recruits in BY t
  ln_Rkr_a[t] <- mu_Rkr_a;
  E_ln_Rec[t] <- ln_Rkr_a[t] + ln_Sp[t] - 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
  for(i in 1:(age_max-age_min)) {
    ## projected recruits
    for(a in 1:i) {
      Run[i,a] <- Rec[i-a+1,a];
    }
    ## imputed recruits
    for(a in (i+1):A) {
      lnRec[i,a] ~ dnorm(Rec_mu,Rec_tau);
      Run[i,a] <- exp(lnRec[i,a]);
    }
    ## total run size
    tot_Run[i+age_min] <- 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:(n_yrs-age_min)) {
    for(a in 1:A) {
      Run[i,a] <- Rec[i-a+1,a];
    }
    ## total run size
    tot_Run[i+age_min] <- 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
  ## first cal yr is first brood yr
  for(t in 1:n_yrs) {
    ## 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] - dat_harv[t];
    ln_Sp[t] <- log(Sp[t]);
    ln_dat_esc[t] ~ dnorm(ln_Sp[t], tau_s);
    lp_esc[t] <- logdensity.norm(ln_dat_esc[t],ln_Sp[t], tau_s);
  }

} ## end model description
```

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

5.2 Ricker model with covariates

```
cat("

model {

  ##--------
  ## PRIORS
  ##--------
  ## alpha = exp(a) = intrinsic productivity
  alpha ~ dnorm(0,0.01) T(0,);
  mu_Rkr_a <- log(alpha);
  E_Rkr_a <- mu_Rkr_a + sigma_r/(2 - 2*phi^2);

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

  ## covariate effect
  gamma ~ dnorm(0,0.01)

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

  ## process variance for 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_s <- 1/sigma_s;
  sigma_s ~ dnorm(0, 0.001) T(0,);

  ## maturity schedule
  ## unif vec for Dirch prior
  theta <- c(2,20,20,1)
  ## 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)) { pi_vec[t,1:A] ~ ddirch(pi_eta*pi_tau) }

  ## 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)) {
      ln_tot_Run[i] ~ dnorm(ttl_run_mu*Rec_mu,Rec_tau/ttl_run_tau);
      tot_Run[i] <- exp(ln_tot_Run[i]);
}

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

##------------
## LIKELIHOOD
##------------
## 1st brood yr requires different innovation
## predicted recruits in BY t
covar[1] <- gamma * mod_cvrs[1];
ln_Rkr_a[1] <- mu_Rkr_a + covar[1];
E_ln_Rec[1] <- ln_Rkr_a[1] + ln_Sp[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)) {
  ## predicted recruits in BY t
  covar[t] <- gamma * mod_cvrs[t];
  ln_Rkr_a[t] <- mu_Rkr_a + covar[t];
  E_ln_Rec[t] <- ln_Rkr_a[t] + ln_Sp[t] - 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
for(i in 1:(age_max-age_min)) {
  ## projected recruits
  for(a in 1:i) {
    Run[i,a] <- Rec[i-a+1,a];
  }
  ## imputed recruits
  for(a in (i+1):A) {
    lnRec[i,a] ~ dnorm(Rec_mu,Rec_tau);
    Run[i,a] <- exp(lnRec[i,a]);
  }
  ## total run size
  tot_Run[i+age_min] <- 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:(n_yrs-age_min)) {
  for(a in 1:A) {
    Run[i,a] <- Rec[i-a+1,a];
  }
  ## total run size
  tot_Run[i+age_min] <- 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
## first cal yr is first brood yr
for(t in 1:n_yrs) {
  ## 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] - dat_harv[t];
  ln_Sp[t] <- log(Sp[t]);
  ln_dat_esc[t] ~ dnorm(ln_Sp[t], tau_s);
  lp_esc[t] <- logdensity.norm(ln_dat_esc[t],ln_Sp[t], tau_s);
}
```

```
} ## end model description

", file=file.path(jagsdir, "IPM_RK_cov_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,
                 dat_harv = dat_harv,
                 A = A,
                 age_min = age_min,
                 age_max = age_max,
                 n_yrs = n_yrs)

## 2. Model params/states for JAGS to return:
par_jags <- c("alpha","E_Rkr_a","ln_Rkr_a",
              "beta",
              "Sp","Rec","tot_ln_Rec","ln_RS",
              "pi_eta","pi_tau",
              "sigma_r","sigma_s","res_ln_Rec",
              "lp_age","lp_esc")

## 3. MCMC control params:
mcmc_ctrl <- list(
  chains = 4,
  length = 1.25e5,
  burn = 5e4,
  thin = 100
)
## total number of MCMC samples after burnin
mcmc_samp <- mcmc_ctrl$length*mcmc_ctrl$chains/mcmc_ctrl$thin
```

Please note that the following code takes ~60 min to run on a quad-core machine with 3.5 GHz Intel processors.

```
## total number of models to fit
n_mods <- 1 + n_cov

## empty list for LOOIC values
LOOIC <- vector("list", n_mods)

## fit base model (if not already saved)
if(!file.exists(file.path(jagsdir, "fit_ricker_base.rds"))) {
  mod_fit <- fit_jags("IPM_RK_AR.txt", dat_jags, par_jags, init_vals_AR, mcmc_ctrl)
  ## save results to file
```

```
  saveRDS(mod_fit, file.path(jagsdir, "fit_ricker_base.rds"))
  ## compute LOOIC
  LOOIC[[1]] <- looic(mod_fit, mcmc_ctrl)
}

## fit models with covariates
par_jags <- c(par_jags, "gamma")
for(i in seq(n_mods-1)) {
  if(!file.exists(file.path(jagsdir, paste0("fit_ricker_cov_", i, ".rds")))) {
    dat_jags$mod_cvrs <- cov_flow[,i]
    mod_fit <- fit_jags("IPM_RK_cov_AR.txt", dat_jags, par_jags,
                        init_vals_cov, mcmc_ctrl)
    ## save results to file
    saveRDS(mod_fit, file.path(jagsdir, paste0("fit_ricker_cov_", i, ".rds")))
    ## compute LOOIC
    LOOIC[[i+1]] <- looic(mod_fit, mcmc_ctrl)
  }
}
if(!file.exists(file.path(jagsdir, "LOOIC_values.rds"))) {
      saveRDS(LOOIC, file.path(jagsdir, "LOOIC_values.rds"))
} else {
      LOOIC <- readRDS(file.path(jagsdir, "LOOIC_values.rds"))
}
```

6.0.0.1 Convergence checks

```
base_mod <- readRDS(file.path(jagsdir, "fit_ricker_base.rds"))

par_conv <- c("alpha", "beta",
              "sigma_r", "sigma_s",
              "pi_tau", paste0("pi_eta[",seq(A),"]"))

## Gelman-Rubin
gelman.diag(base_mod[,par_conv])
```

```
## Potential scale reduction factors:
##
##            Point est. Upper C.I.
## alpha            1.06       1.06
## beta             1.11       1.29
## sigma_r          1.07       1.14
## sigma_s          1.64       6.51
## pi_tau           1.00       1.00
## pi_eta[1]        1.00       1.00
## pi_eta[2]        1.00       1.00
## pi_eta[3]        1.00       1.00
## pi_eta[4]        1.00       1.01
##
## Multivariate psrf
##
## 1.37
```

```
## autocorrelation
t(round(autocorr.diag(base_mod[,par_conv],
```

```
                     lags = seq(mcmc_ctrl$thin, 4*mcmc_ctrl$thin, mcmc_ctrl$thin),
                     relative=FALSE), 2))
```

```
##             Lag 100 Lag 200 Lag 300 Lag 400
## alpha          0.10    0.06    0.06    0.09
## beta           0.27    0.23    0.21    0.21
## sigma_r        0.23    0.22    0.19    0.18
## sigma_s        0.38    0.27    0.21    0.18
## pi_tau         0.06    0.04    0.05    0.03
## pi_eta[1]      0.03    0.01    0.01    0.03
## pi_eta[2]      0.03    0.03    0.03    0.02
## pi_eta[3]      0.03    0.04    0.03    0.03
## pi_eta[4]      0.16    0.12    0.12    0.08
```

7 Model selection

Here is a table of LOOIC results as estimated with `loo()`.

```
## data frame of LOOIC values
tbl_LOOIC <- as.data.frame(round(compare(x = LOOIC), 1))
tbl_LOOIC$d_looic <- -2 * tbl_LOOIC$elpd_diff
tbl_LOOIC <- tbl_LOOIC[, c("p_loo", "se_p_loo", "looic", "se_looic", "d_looic")]
rownames(tbl_LOOIC) <- sub("model", "", rownames(tbl_LOOIC))
tbl_LOOIC <- tbl_LOOIC[order(as.numeric(rownames(tbl_LOOIC))),]
tbl_LOOIC <- data.frame(life_stage = c("base", cov_meta$life_stage),
                        variable = c("NA", sub(" of 7-day mean","",cov_meta$long_name)),
                        begin = c("NA", cov_meta$begin),
                        end = c("NA", cov_meta$end),
                        lag = c(NA, cov_meta$lag_1),
                        tbl_LOOIC)
## best model; need to subtract 1 from index to acct for base model
best_i <- which(tbl_LOOIC[,"looic"] == min(tbl_LOOIC[,"looic"])) - 1
best_fit <- readRDS(file.path(jagsdir, paste0("fit_ricker_cov_", best_i, ".rds")))
## table of LOOIC values
kable(tbl_LOOIC[order(tbl_LOOIC[,"looic"]),], "latex", booktabs = TRUE)
```

	life_stage	variable	begin	end	lag	p_loo	se_p_loo	looic	se_looic	d_looic
29	2+ outmigrants	Range	02-01	04-30	2	53.1	3.8	335.0	63.8	0.0
22	rearing	Min	07-01	09-30	1	53.3	3.9	336.8	63.8	1.8
26	2+ outmigrants	Max	02-01	04-30	2	53.6	4.0	336.9	63.9	2.0
6	prespawn	Max	04-01	06-30	0	53.8	4.1	340.0	63.8	5.0
9	prespawn	Min	05-01	05-31	0	51.9	4.0	340.0	63.2	5.0
10	prespawn	Max	05-01	05-31	0	52.5	4.0	340.5	63.3	5.6
15	1+ outmigrants	Min	04-01	06-30	1	52.2	4.3	340.5	63.7	5.4
30	2+ outmigrants	Max	04-01	04-30	2	52.1	4.0	340.7	63.0	5.6
7	prespawn	Range	04-01	06-30	0	53.7	4.2	340.8	63.8	5.8
18	1+ outmigrants	Range	04-01	06-30	1	52.1	4.1	341.0	63.5	6.0
28	2+ outmigrants	Min	02-01	04-30	2	53.8	4.0	341.2	63.5	6.2
11	prespawn	Min	07-01	09-30	0	53.2	4.2	342.0	63.5	7.0
16	1+ outmigrants	Median	04-01	06-30	1	54.0	3.9	342.4	63.1	7.4
13	incubation	Max	11-01	03-31	0	54.3	4.2	342.5	64.0	7.4
33	2+ outmigrants	Range	04-01	04-30	2	53.5	4.0	342.6	63.3	7.6
5	prespawn	Median	04-01	06-30	0	53.6	4.0	342.8	63.1	7.8
14	incubation	Median	11-01	03-31	0	53.9	4.2	342.9	63.6	7.8
2	prespawn	Max	11-01	03-31	-1	53.6	4.3	343.3	63.8	8.4
24	rearing	Max	07-01	09-30	1	54.9	4.4	343.9	64.2	8.8
27	2+ outmigrants	Median	02-01	04-30	2	54.2	4.4	344.1	64.0	9.0
23	rearing	Median	07-01	09-30	1	55.1	4.6	344.2	64.7	9.2
25	rearing	Range	07-01	09-30	1	54.9	4.5	344.2	64.3	9.2
8	prespawn	Min	04-01	04-30	0	58.2	4.4	344.6	65.2	9.6
3	prespawn	Median	11-01	03-31	-1	54.7	4.6	344.7	64.3	9.8
12	prespawn	Median	07-01	09-30	0	54.0	4.5	344.8	63.8	9.8
32	2+ outmigrants	Min	04-01	04-30	2	54.4	4.3	345.2	63.7	10.2
31	2+ outmigrants	Median	04-01	04-30	2	53.4	4.3	345.8	63.3	10.8
17	1+ outmigrants	Max	04-01	06-30	1	55.4	4.6	346.1	64.4	11.0
20	1+ outmigrants	Max	05-01	05-31	1	54.9	4.7	346.2	64.2	11.2
4	prespawn	Min	04-01	06-30	0	55.3	4.3	346.3	63.8	11.4
19	1+ outmigrants	Min	04-01	04-30	1	56.0	4.3	346.6	64.0	11.6
21	1+ outmigrants	Min	05-01	05-31	1	56.4	4.7	348.6	64.3	13.6
1	base	NA	NA	NA	NA	67.6	9.1	374.5	61.1	39.4

8 Model diagnostics

8.1 Gelman & Rubin statistic

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","gamma",
              "sigma_r","sigma_s",
              "pi_tau",paste0("pi_eta[",seq(A-1),"]"),
              paste0("Sp[",seq(n_yrs),"]"),
              paste0("tot_ln_Rec[",seq(n_yrs-age_min),"]"))
```

```r
## Gelman-Rubin
gelman.diag(best_fit[,par_conv])
```

```
## Potential scale reduction factors:
##
##                  Point est. Upper C.I.
## alpha                     1       1.00
## beta                      1       1.00
## gamma                     1       1.00
## sigma_r                   1       1.00
## sigma_s                   1       1.00
## pi_tau                    1       1.00
## pi_eta[1]                 1       1.00
## pi_eta[2]                 1       1.00
## pi_eta[3]                 1       1.00
## Sp[1]                     1       1.00
## Sp[2]                     1       1.00
## Sp[3]                     1       1.00
## Sp[4]                     1       1.01
## Sp[5]                     1       1.00
## Sp[6]                     1       1.00
## Sp[7]                     1       1.00
## Sp[8]                     1       1.00
## Sp[9]                     1       1.00
## Sp[10]                    1       1.00
## Sp[11]                    1       1.00
## Sp[12]                    1       1.00
## Sp[13]                    1       1.00
## Sp[14]                    1       1.00
## Sp[15]                    1       1.00
## Sp[16]                    1       1.00
## Sp[17]                    1       1.00
## Sp[18]                    1       1.00
## Sp[19]                    1       1.00
## tot_ln_Rec[1]             1       1.01
## tot_ln_Rec[2]             1       1.00
## tot_ln_Rec[3]             1       1.00
## tot_ln_Rec[4]             1       1.00
## tot_ln_Rec[5]             1       1.00
## tot_ln_Rec[6]             1       1.00
## tot_ln_Rec[7]             1       1.00
## tot_ln_Rec[8]             1       1.00
## tot_ln_Rec[9]             1       1.00
## tot_ln_Rec[10]            1       1.00
## tot_ln_Rec[11]            1       1.00
## tot_ln_Rec[12]            1       1.00
## tot_ln_Rec[13]            1       1.00
## tot_ln_Rec[14]            1       1.00
## tot_ln_Rec[15]            1       1.00
## tot_ln_Rec[16]            1       1.00
##
## Multivariate psrf
##
## 1.01
```

8.2 Autocorrelation

```
t(round(autocorr.diag(best_fit[,par_conv],
                      lags = seq(mcmc_ctrl$thin, 3*mcmc_ctrl$thin, mcmc_ctrl$thin),
                      relative=FALSE), 2))
```

```
##                  Lag 100 Lag 200 Lag 300
## alpha               0.02    0.04   -0.01
## beta                0.02    0.01   -0.01
## gamma               0.02    0.03    0.01
## sigma_r             0.00   -0.02   -0.03
## sigma_s             0.22    0.06    0.02
## pi_tau              0.05    0.04    0.03
## pi_eta[1]           0.00    0.02    0.02
## pi_eta[2]           0.03   -0.01    0.01
## pi_eta[3]           0.03    0.00    0.01
## Sp[1]               0.02    0.02    0.00
## Sp[2]               0.01   -0.02   -0.03
## Sp[3]               0.01    0.00    0.01
## Sp[4]               0.01    0.02    0.01
## Sp[5]               0.04    0.01   -0.01
## Sp[6]               0.01   -0.02   -0.01
## Sp[7]               0.08    0.01    0.01
## Sp[8]               0.05    0.01    0.01
## Sp[9]               0.08    0.00   -0.03
## Sp[10]              0.09    0.04    0.02
## Sp[11]              0.09    0.01    0.00
## Sp[12]              0.18    0.05    0.01
## Sp[13]              0.11    0.03    0.01
## Sp[14]              0.10    0.03    0.00
## Sp[15]              0.02    0.01    0.01
## Sp[16]              0.07    0.01   -0.02
## Sp[17]              0.03    0.00   -0.03
## Sp[18]              0.07    0.04    0.00
## Sp[19]              0.03    0.02    0.01
## tot_ln_Rec[1]      -0.01   -0.02    0.00
## tot_ln_Rec[2]       0.00   -0.02   -0.01
## tot_ln_Rec[3]       0.02   -0.01    0.01
## tot_ln_Rec[4]       0.06    0.02    0.00
## tot_ln_Rec[5]       0.05    0.01   -0.01
## tot_ln_Rec[6]       0.04    0.01    0.00
## tot_ln_Rec[7]       0.06    0.01   -0.01
## tot_ln_Rec[8]       0.15    0.05    0.01
## tot_ln_Rec[9]       0.10    0.03    0.01
## tot_ln_Rec[10]      0.05    0.01   -0.01
## tot_ln_Rec[11]      0.01    0.00    0.00
## tot_ln_Rec[12]      0.05    0.00   -0.03
## tot_ln_Rec[13]      0.03    0.01   -0.01
## tot_ln_Rec[14]      0.10    0.06    0.04
## tot_ln_Rec[15]      0.04    0.00   -0.01
## tot_ln_Rec[16]      0.00    0.01    0.01
```

8.3 Effective sample sizes

```
floor(effectiveSize(best_fit))
```

```
##        E_Rkr_a       Rec[1,1]       Rec[2,1]       Rec[3,1]       Rec[4,1]       Rec[5,1]
##           4459           4807           4493           3792           4389           4566
##       Rec[6,1]       Rec[7,1]       Rec[8,1]       Rec[9,1]      Rec[10,1]      Rec[11,1]
##           4708           4428           4536           3986           5166           4370
##      Rec[12,1]      Rec[13,1]      Rec[14,1]      Rec[15,1]      Rec[16,1]       Rec[1,2]
##           4999           4442           4969           4748           4999           4087
##       Rec[2,2]       Rec[3,2]       Rec[4,2]       Rec[5,2]       Rec[6,2]       Rec[7,2]
##           4894           4689           4350           4358           4159           4279
##       Rec[8,2]       Rec[9,2]      Rec[10,2]      Rec[11,2]      Rec[12,2]      Rec[13,2]
##           3590           3619           3992           4872           4363           5070
##      Rec[14,2]      Rec[15,2]      Rec[16,2]       Rec[1,3]       Rec[2,3]       Rec[3,3]
##           4540           5195           5146           5006           4584           4822
##       Rec[4,3]       Rec[5,3]       Rec[6,3]       Rec[7,3]       Rec[8,3]       Rec[9,3]
##           4508           3992           4277           3498           4017           4293
##      Rec[10,3]      Rec[11,3]      Rec[12,3]      Rec[13,3]      Rec[14,3]      Rec[15,3]
##           4918           4703           4870           4328           4893           4619
##      Rec[16,3]       Rec[1,4]       Rec[2,4]       Rec[3,4]       Rec[4,4]       Rec[5,4]
##           5000           4760           4639           4919           4615           4450
##       Rec[6,4]       Rec[7,4]       Rec[8,4]       Rec[9,4]      Rec[10,4]      Rec[11,4]
##           3521           4104           4716           3940           4311           4438
##      Rec[12,4]      Rec[13,4]      Rec[14,4]      Rec[15,4]      Rec[16,4]          Sp[1]
##           4339            972           1432           2445           4030           4689
##          Sp[2]          Sp[3]          Sp[4]          Sp[5]          Sp[6]          Sp[7]
##           4999           5000           5000           4157           4880           4508
##          Sp[8]          Sp[9]         Sp[10]         Sp[11]         Sp[12]         Sp[13]
##           4670           4302           4071           4194           3393           3882
##         Sp[14]         Sp[15]         Sp[16]         Sp[17]         Sp[18]         Sp[19]
##           4012           4783           4315           4915           4486           4826
##          alpha           beta          gamma       ln_RS[1]       ln_RS[2]       ln_RS[3]
##           4728           5069           4507           5113           5000           4894
##       ln_RS[4]       ln_RS[5]       ln_RS[6]       ln_RS[7]       ln_RS[8]       ln_RS[9]
##           4805           4444           4576           4173           4477           4033
##      ln_RS[10]      ln_RS[11]      ln_RS[12]      ln_RS[13]      ln_RS[14]      ln_RS[15]
##           3987           4716           3544           3603           4262           4909
##      ln_RS[16]      lp_age[1]      lp_age[2]      lp_age[3]      lp_age[4]      lp_age[5]
##           5227           4997           5330           4725           5195           4716
##      lp_age[6]      lp_age[7]      lp_age[8]      lp_age[9]     lp_age[10]     lp_age[11]
##           4795           5376           5465           5122           4999           5000
##     lp_age[12]     lp_age[13]     lp_age[14]     lp_age[15]     lp_age[16]      lp_esc[1]
##           5000           4502           4268           5000           1849           3853
##      lp_esc[2]      lp_esc[3]      lp_esc[4]      lp_esc[5]      lp_esc[6]      lp_esc[7]
##           3923           3509           3727           3861           3680           3541
##      lp_esc[8]      lp_esc[9]     lp_esc[10]     lp_esc[11]     lp_esc[12]     lp_esc[13]
##           3513           3519           3459           3529           3329           3933
##     lp_esc[14]     lp_esc[15]     lp_esc[16]     lp_esc[17]     lp_esc[18]     lp_esc[19]
##           3337           3740           3545           3597           3526           3792
##       mu_Rkr_a      pi_eta[1]      pi_eta[2]      pi_eta[3]      pi_eta[4]         pi_tau
##           4478           5000           4977           4854           2236           4190
## res_ln_Rec[1]  res_ln_Rec[2]  res_ln_Rec[3]  res_ln_Rec[4]  res_ln_Rec[5]  res_ln_Rec[6]
##           5000           5119           4902           4775           4772           5000
```

```
##  res_ln_Rec[7]  res_ln_Rec[8]  res_ln_Rec[9] res_ln_Rec[10] res_ln_Rec[11] res_ln_Rec[12]
##           4520           4999           4861           4662           4810           4912
## res_ln_Rec[13] res_ln_Rec[14] res_ln_Rec[15] res_ln_Rec[16]        sigma_r        sigma_s
##           4262           4773           4920           5214           5254           3203
##  tot_ln_Rec[1]  tot_ln_Rec[2]  tot_ln_Rec[3]  tot_ln_Rec[4]  tot_ln_Rec[5]  tot_ln_Rec[6]
##           5330           5000           5734           4361           4511           4620
##  tot_ln_Rec[7]  tot_ln_Rec[8]  tot_ln_Rec[9] tot_ln_Rec[10] tot_ln_Rec[11] tot_ln_Rec[12]
##           4844           3506           3928           4424           5000           4714
## tot_ln_Rec[13] tot_ln_Rec[14] tot_ln_Rec[15] tot_ln_Rec[16]
##           4661           3589           4786           5187
```