

Bayesian Estimation of Recruitment Trends in Alberta (BERTA) Tutorial

Christopher L. Cahill

31 December 2021

Goals

- Understand general structure of and build intuition for the age-structured population dynamics model used in Cahill et al. (2021).
- Gain an understanding of available modeling options, including tweaking priors and MCMC run parameters.
- Become familiar with tidyverse sub-setting, `get_fit()`, `future_pwalk()`, and `plan()`.
- Learn how .R and .stan scripts are working together to subset data, fit a Bayesian stock reduction analysis model to those data, and then save the model fit with a unique file name identifier.
- Practice debugging using `browser()`.

Packages

Let's load the packages we will use:

```
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.1 --

## v ggplot2 3.3.5      v purrr  0.3.4
## v tibble  3.1.3      v dplyr  1.0.7
## v tidyr   1.1.3      v stringr 1.4.0
## v readr   2.0.1      v forcats 0.5.1

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(rstan)

## Warning: package 'rstan' was built under R version 4.1.2

## Loading required package: StanHeaders

## rstan (Version 2.21.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:tidyr':
##
##      extract
```

```
library(furrr)
```

```
## Loading required package: future
```

```
library(future)
library(ggplot2)
```

Data

We will work with the Fall Walleye Index Netting (FWIN) dataset used in Cahill et al. (2021), which included all Alberta lakes with ≥ 3 FWIN surveys during 2000-2018. Life history parameters ω , A_{50} , L_{∞} , vb_k , and β_{wl} were obtained using hierarchical modeling methods described in Cahill et al. (2020), and these values represent lake-specific averages.

```
data <- readRDS(here::here("data/BERTA-wide-0-25.rds"))
glimpse(data)
```

```
## Rows: 236
## Columns: 46
## Groups: name [55]
## $ WBID      <int> 3526, 3526, 3526, 3526, 3916, 3916, 3916, 3969, 3969, 3969,~
## $ year      <dbl> 5, 6, 13, 19, 6, 11, 14, 9, 12, 15, 17, 7, 12, 17, 4, 11, 1~
## $ name      <chr> "milk river ridge reservoir", "milk river ridge reservoir",~
## $ nnet      <int> 18, 20, 11, 12, 8, 12, 10, 11, 12, 12, 12, 18, 15, 15, 4, 6~
## $ n         <int> 201, 283, 232, 158, 189, 117, 132, 357, 171, 186, 201, 373,~
## $ effort    <dbl> 18.0, 20.0, 11.0, 12.0, 8.0, 6.0, 5.0, 11.0, 6.0, 6.0, 6.0,~
## $ X_TTM_c   <dbl> 676088.4, 676088.4, 676088.4, 676088.4, 652638.4, 652638.4,~
## $ Y_TTM_c   <dbl> 5469124, 5469124, 5469124, 5469124, 6050150, 6050150, 60501~
## $ p_aged    <dbl> 1.0000000, 1.0000000, 1.0000000, 0.9430380, 1.0000000, 0.98~
## $ omega     <dbl> 12.22278, 12.22278, 12.22278, 12.22278, 13.93477, 13.93477,~
## $ linf      <dbl> 56.72357, 56.72357, 56.72357, 56.72357, 51.38603, 51.38603,~
## $ vbk       <dbl> 0.2154797, 0.2154797, 0.2154797, 0.2154797, 0.2711781, 0.27~
## $ a50       <dbl> 7, 7, 7, 7, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 5, 5,~
## $ beta_wl   <dbl> 3.409773, 3.409773, 3.409773, 3.409773, 3.100920, 3.100920,~
## $ X_long    <dbl> -112.5735, -112.5735, -112.5735, -112.5735, -112.6363, -112~
## $ Y_lat     <dbl> 49.36904, 49.36904, 49.36904, 49.36904, 54.59751, 54.59751,~
```

```
## $ Area_Ha      <dbl> 1355.0, 1355.0, 1355.0, 1355.0, 527.1, 527.1, 527.1, 970.7, ~
## $ DD5          <int> 1605, 1605, 1605, 1605, 1293, 1293, 1293, 1293, 1293, 1293, ~
## $ Max_Depth    <dbl> NA, NA, NA, NA, 27.5, 27.5, 27.5, 27.4, 27.4, 27.4, 27.4, 1~
## $ Mean_Depth   <dbl> NA, NA, NA, NA, 14.3, 14.3, 14.3, 9.2, 9.2, 9.2, 9.2, 6.9, ~
## $ '1'          <dbl> 23, 21, 5, 2, 0, 4, 4, 2, 10, 3, 23, 24, 5, 6, 1, 18, 0, 0, ~
## $ '2'          <dbl> 13, 67, 43, 6, 0, 4, 7, 24, 4, 14, 41, 11, 5, 10, 7, 14, 0, ~
## $ '3'          <dbl> 18, 55, 13, 8, 5, 17, 44, 53, 5, 12, 14, 16, 5, 4, 15, 13, ~
## $ '4'          <dbl> 30, 46, 10, 11, 3, 16, 13, 78, 4, 8, 17, 50, 11, 2, 7, 5, 1~
## $ '5'          <dbl> 27, 35, 17, 13, 41, 3, 8, 15, 12, 0, 8, 55, 5, 37, 17, 9, 1~
## $ '6'          <dbl> 26, 20, 16, 30, 80, 1, 23, 6, 26, 11, 7, 38, 5, 18, 0, 7, 7~
## $ '7'          <dbl> 16, 11, 18, 40, 45, 2, 11, 8, 37, 1, 4, 27, 6, 15, 0, 5, 8, ~
## $ '8'          <dbl> 23, 10, 14, 11, 9, 1, 3, 4, 4, 12, 4, 39, 4, 5, 2, 1, 6, 45~
## $ '9'          <dbl> 16, 7, 20, 8, 2, 1, 1, 14, 1, 23, 0, 7, 8, 12, 0, 4, 4, 85, ~
## $ '10'         <dbl> 6, 1, 14, 3, 2, 18, 1, 26, 3, 38, 7, 10, 17, 7, 3, 4, 3, 3, ~
## $ '11'         <dbl> 0, 1, 3, 6, 0, 30, 1, 38, 6, 9, 16, 23, 6, 1, 0, 5, 3, 1, 0~
## $ '12'         <dbl> 0, 0, 0, 3, 0, 12, 0, 45, 2, 3, 25, 39, 1, 3, 0, 2, 1, 0, 0~
## $ '13'         <dbl> 0, 0, 1, 2, 1, 0, 5, 3, 14, 2, 5, 3, 9, 0, 2, 0, 0, 0, 0, 0~
## $ '14'         <dbl> 0, 1, 1, 0, 1, 1, 6, 18, 7, 5, 3, 1, 0, 4, 0, 1, 1, 1, 0, 0~
## $ '15'         <dbl> 0, 0, 0, 1, 0, 0, 2, 10, 5, 4, 1, 0, 1, 5, 0, 0, 0, 0, 0, 0~
## $ '16'         <dbl> 0, 0, 0, 1, 0, 0, 2, 2, 3, 8, 2, 0, 1, 4, 0, 0, 0, 0, 0, 0, ~
## $ '17'         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 12, 3, 0, 7, 3, 0, 0, 0, 0, 0, ~
## $ '18'         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 10, 3, 0, 1, 4, 0, 0, 0, 0, 0, ~
## $ '19'         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 3, 0, 0, 0, 0, 0, 0, ~
## $ '20'         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 4, 0, 0, 1, 0, 0, 0, 0, 0, ~
## $ '21'         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0, ~
## $ '22'         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 0, 0, ~
## $ '23'         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ '24'         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ '25'         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ lake         <dbl> 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 5, 6, 6, ~
```

Now read in the stocking data, which was used for plotting and not fitted in the .stan model. Note these stocking records go from 1980-2018, and values represent the number of Walleye stocked per hectare:

```
stocking <- readRDS(here::here("data/stocking_matrix_ha.rds"))
glimpse(stocking)
```

```
##   num [1:106, 1:39] 0 0 0 0 0 0 0 0 0 0 0 ...
##   - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:106] "berry creek reservoir" "jensen reservoir" "milk river ridge reservoir" "travers
##   ..$ : NULL
```

Create a wrapper function

Once the data are read into R, we can write a wrapper function called `get_fit()` that does the following:

- subsets all the data to data for a specific lake
- creates the appropriate tagged list data structures as input into the Stan model
- creates appropriate input for parameters for our Stan model
- runs the stan model for a particular combination of priors (e.g., which α_r), structural control parameters (e.g., Ricker vs. Beverton-Holt stock-recruit), and Stan run parameter values (i.e., how many iterations?)

- saves this model run with a clever name (e.g., pigeon_lake_ricker_cr6.rds)

This may seem like a pain, but coding this way will help us later on when we need to run multiple models on different data sets.

```
get_fit <- function(which_lake = "pigeon lake",
                    rec_ctl = c("bev-holt", "ricker"),
                    cr_prior = c(6, 12),
                    n_iter = n_iter, n_chains = n_chains,
                    n_warmup = n_iter / 2,
                    ...) {
  rec_ctl <- match.arg(rec_ctl)
  cat(
    crayon::green(
      clisymbols::symbol$tick
    ),
    fitted = "model fitted = ", which_lake, rec_ctl,
    sep = " "
  )
  cat("\n")

  #filter the run data from all data, re-order it
  run_data <- data %>% filter(name %in% which_lake)
  run_data <-
    within(run_data, lake <-
      as.numeric(interaction(
        run_data$WBID,
        drop = TRUE, lex.order = F
      )))
  run_data <- run_data[order(run_data$lake), ]

  # stocking stuff was run in different versions, now just for plotting:
  run_stocking <- stocking[which(rownames(stocking) %in% which_lake), ]

  # Add ten years of zero for short term projections
  proj_stock <- rep(0, 10)
  run_stocking <- round(c(run_stocking, proj_stock)) #add to stocking data (for plots)

  # Set up the Rbar years
  suppressMessages(
    survey_yrs <- run_data %>%
      group_by(lake) %>%
      summarise(
        min_yr = min(year) + length(initial_yr:(t - 1)),
        max_yr = max(year) + length(initial_yr:(t - 1))
      ) %>%
      as.numeric()
  )
  survey_yrs <- survey_yrs[2:3]
  # summarize the life history relationships
  suppressMessages(
    life_hist <- run_data %>%
      group_by(lake) %>%
      summarize(
```

```

    a50 = unique(a50),
    vbk = unique(vbk),
    linf = unique(linf),
    wl_beta = unique(beta_wl)
  )
)

Fseq <- seq(from = 0.01, to = 1.0, by = 0.01)

# declare the tagged data list for stan
stan_data <- list(
  n_surveys = nrow(run_data),
  n_ages = length(Ages),
  n_obs = nrow(run_data) * length(Ages),
  n_years = length(initial_yr:2028),
  n_lakes = length(unique(run_data$lake)),
  caa = run_data[, which(colnames(run_data) %in% Ages)],
  prop_aged = run_data$p_aged,
  effort = run_data$effort,
  lake = run_data$lake,
  year = run_data$year + length(initial_yr:(t - 1)),
  ages = Ages,
  survey_yrs = survey_yrs,
  which_year = 1996 - initial_yr + 2, # which integer corresponds to year = 1997
  v_prior_early = 0.3,
  v_prior_late = 0.1,
  prior_sigma_v = c(0.1, 0.5),
  R0_mean = log(6),
  R0_sd = log(3),
  ar_sd = 0.1,
  prior_mean_w = 0,
  prior_sigma_w = 1.2,
  vbk = life_hist$vbk,
  linf = life_hist$linf,
  a50 = life_hist$a50,
  wl_beta = life_hist$wl_beta,
  lbar = 57.57, # From cahill et al. 2020
  M = 0.1,
  theta = 0.85, # Lorenzen M exponent
  phi = 2.02, # vulnerability parameter (nets)
  psi = 2, # vulnerability parameter (angling)
  G_bound = c(0, Inf),
  get_SSB_obs = 1L,
  obs_cv_prior = 0.15,
  SSB_penalty = 0,
  prior_sigma_G = 1,
  Rinit_ctl = 0,
  length_Fseq = length(Fseq),
  Fseq = Fseq,
  rec_ctl = ifelse(rec_ctl == "ricker", 0, 1),
  cr_prior = cr_prior
)

```

```

# create a function for start values
vk <- c(0.3, 0.3)
inits <- function() {
  list(
    v = jitter(vk, amount = 0.1),
    R0 = jitter(15, amount = 2),
    G = jitter(1, amount = 0.1),
    w = jitter(rep(0,
                  stan_data$n_years - 2),
              amount = 0.1),
    sigma_w = jitter(0.5, amount = 0.05),
    ar = jitter(0.5, amount = 0.01)
  )
}

#run the model
fit <-
  rstan::sampling(
    m,
    data = stan_data,
    pars =
      c(
        "ar_mean_kick", "F_ratio", "Fmsy", "MSY",
        "G", "cr", "ar", "SPR", "br",
        "SBR", "sbr0_kick", "R0", "v", "SSB",
        "R2", "SSB_obs", "caa_pred", "b_ratio", "w"
      ),
    iter = n_iter,
    warmup = n_warmup,
    chains = n_chains,
    init = inits,
    control = list(
      adapt_delta = 0.999,
      max_treedepth = 15
    )
  )

# create name and save .rds files for each run
if(rec_ctl=="ricker"){
  my_name <- paste0(which_lake, "_ricker.rds")
}
if(rec_ctl=="bev-holt"){
  my_name <- paste0(which_lake, "_bh.rds")
}
stan_file <- "fits/"
stan_file <- str_c(stan_file, my_name)
stan_file <- stan_file %>% gsub(" ", "_", .)
if(cr_prior == 6){
  stan_file <- stan_file %>% gsub(".rds", "_cr_6.rds", .)
}
if(cr_prior == 12){
  stan_file <- stan_file %>% gsub(".rds", "_cr_12.rds", .)
}
if (file.exists(stan_file)) {

```

```
    return(NULL)
  } else {
    saveRDS(fit, file = stan_file)
  }
}
```