STADEM Vignette

Kevin E. See

2017-10-05

Contents

Introduction

The STADEM package was developed with the goal of estimating total adult escapement of spring/summer Chinook salmon and steelhead that cross Lower Granite dam (LGD). In addition, to meet desired management and research objectives, total escapement has to include estimates of uncertainty and be parsed into weekly strata by three origin groups; wild, hatchery and hatchery no-clip. To reach this goal, we have developed the **ST**ate space **A**dult **D**am **E**scapement **M**odel (STADEM) model that incorporates fish ladder window counts, data from sampled fish at the LGD adult trap, and observations of previously PIT tagged fish at LGD adult detection sites.

Some of the data needed for STADEM is available at other dams, and the package developers are currently working to develop the ability to query all of the necessary data at other locations. Currently however, the focus remains on Lower Granite dam, the furthest upstream dam returning salmonids encounter on the journey up the Snake River. The following example will show how to run STADEM for one species and one year, and what some of the output looks like.

System requirements

STADEM relies on the following R packages which can be downloaded via CRAN or by using the function `install.packages()`:

- `dplyr`, `lubridate`, `httr`: can all be installed by installing the `tidyverse` package
- `rjags`
- `jagsUI`
- `boot`

In addition, STADEM requires the JAGS software (Just Another Gibbs Sampler). Please download version >= 4.0.0 via the former link.

Data sources

STADEM relies on several pieces of data, which must be compiled from multiple sources. Many of them are accessed through the Columbia Basin Research Data Access in Real Time (DART) website.

- **Window counts** are available through DART for many dams within the Columbia River basin.
- **Trap data** comes from an adult fish trap. This provides biological information (e.g., origin, genetic stock, age, sex) to allow the decomposition of total escapement into specific groups, as well as a secondary estimate of total escapement, if the trap rate is known or can be reliably estimated.
- **PIT-tag data** comes from fish that were previously PIT tagged, either as juveniles or as adults at one of the dams downstream of Lower Granite. These fish provide information about the proportion of the run that crosses the dam at night, when the counting window is closed, as well as the proportion of the run that has fallen back and re-ascended the dam. This data is also available through DART.

High-level overview

STADEM estimates the total number of fish crossing the dam each week, based on two major data sources: the window counts and the total fish in the trap, while also accounting for two known biological processes: night-time passage and fallback-and-reacension. Using a state-space approach, STADEM assumes that the window counts and the estimates from the trap (fish in the trap divided by trap rate that week) are generated by processes with observation error. In the case of the window counts, there is some inherent error in the counting process, and it fails to account for fish that cross the dam while the window is closed. In the case of the trap, there is sampling variation and uncertainty in what the realized trap rate is. In addition, STADEM accounts for potential double-counting of fish that have fallen back and re-ascended the dam. It then partitions the estimate of total fish over the dam by origin, to provide the needed data for management goals (Figure 1).

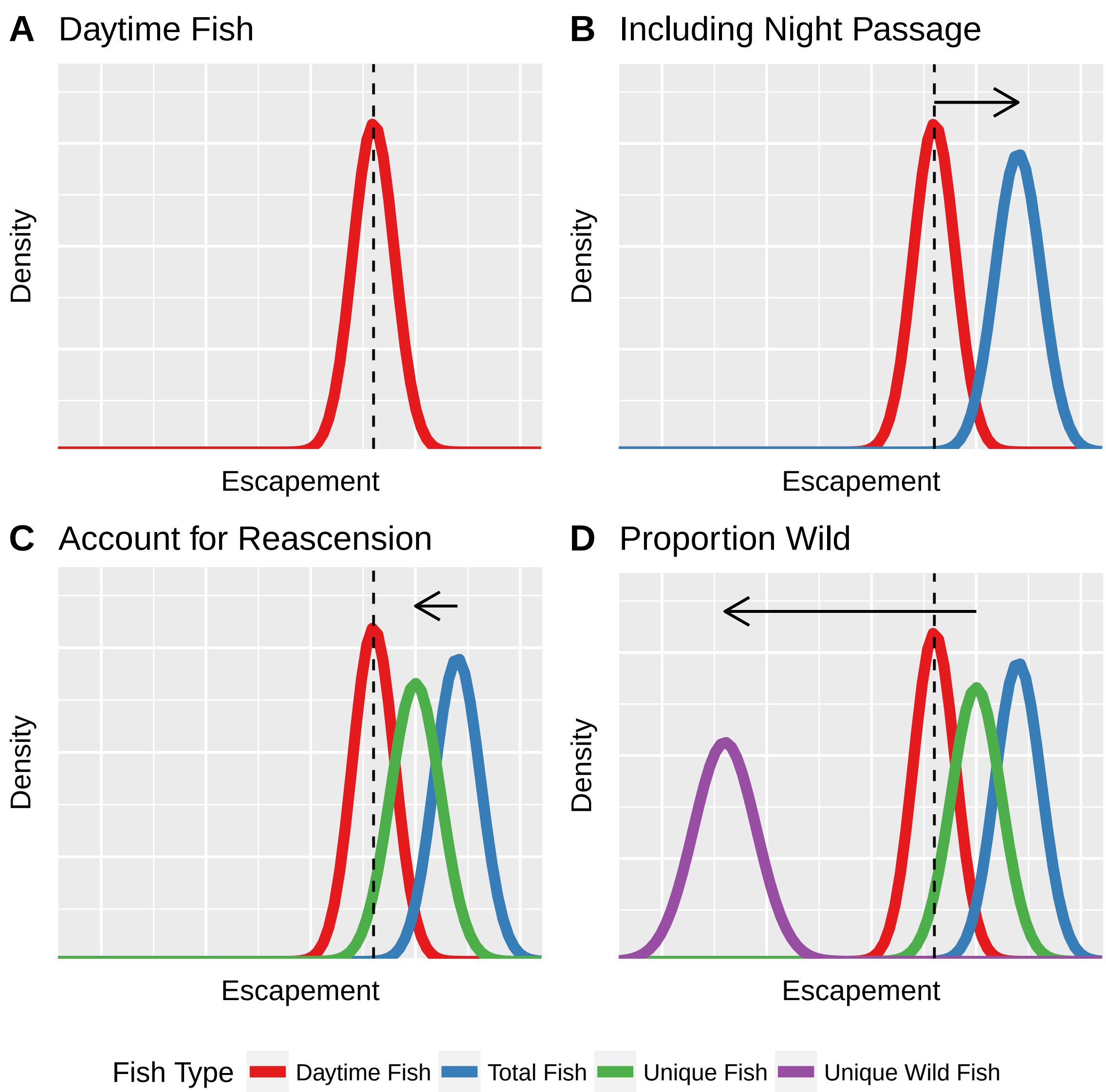

Figure 1: Figure 1: Schematic of how the STADEM model works. Panel A shows the posterior of the estimate of fish crossing while the window is open (dashed line shows observed window counts). That estimate is divided by the nighttime passage rate (B). The total fish is then discounted by the reascension rate to estimate unique fish (C). Those unique fish are then multiplied by the proportion of wild fish (D), to estimate unique wild fish.

Compiling data

The STADEM package relies on many functions from two other packages, `tidyverse` for many data manipulation functions and `lubridate` for dates and intervals.

```
library(tidyverse)
library(lubridate)
library(STADEM)
library(jagsUI)
```

The STADEM package makes it easy to compile all the necessary data in one convenient function, `compileGRAdata`. The user provides the spawn year and species (either "Chinook" or "Steelhead") they are interested in. STADEM operates on a weekly time-step, and the user has the option to determine the day of the week that most strata will begin on, using the `strata_beg` arguement. There are periodic messages describing what is being done within `compileGRAdata`. Within `compileGRAdata`, several internal functions are being called, which may be utilized with the STADEM package for other purposes. They include:

- `getWindowCounts`: query DART for window counts at various dams
- `queryPITtagData`: query DART for data about PIT tags exhibiting night-time passage and fall-back/reascension behavior
- `weeklyStrata`: divide season into weekly strata, with the user defining the day of the week each strata should begin with
- `summariseLGRtrapDaily`: summarise on a daily time-step the .csv file contained within the Lower Granite Dam trap database
- `tagTrapRate`: estimate the fish trap rate based on proportion of PIT tags known to have crossed the dam were caught in the adult trap
- `queryTrapRate`: query DART for intended and realized trap rates, based on time the trap is open

```
# what spawning year? (2010 - 2016)
yr = 2014

# what species?
spp = c('Chinook', 'Steelhead')[1]

# pull together all data
stadem_list = compileGRAdata(yr = yr,
                             spp = spp,
                             strata_beg = 'Sun')
```

The `compileGRAdata()` function returns several pieces of information, consolidated into a named list we have called `stadem_list`:

- `weekStrata`: weekly strata for STADEM, which are `interval` objects from the `lubridate` package.
- `trapData`: data from adult fish trap.
- `dailyData`: data.frame of data, summarised by day.
- `weeklyData`: data.frame of data, summarised by weekly strata.

To run STADEM, only `weeklyData` is needed. STADEM also includes a function to transform all relevant data from the weekly summary to a list ready to be passed to JAGS.

```
# compile everything into a list to pass to JAGS
jags_data_list = prepJAGS(stadem_list[['weeklyData']])
```

Run STADEM

Part of the function runJAGSmodel writes the JAGS model. This requires a filename, and the type of statistical distribution the user would like to use to model the window counts. The options are Poisson (pois), negative binomial (neg_bin), or a more flexible type of negative binomial, described in Lindén and Mäntyniemi (2011) (neg_bin2), or quasi-Poisson (quasi_pois). Once those have been set, use the runSTADEMmodel function to run the model in JAGS. Some of the inputs to this function are:

- file_name name of text file to write the JAGS model to (should end in .txt)
- mcmc_chainlength length of MCMC chain
- mcmc_burn length of burn-in period for MCMC chain
- mcmc_thin thinning rate for with samples of MCMC chain to keep
- mcmc_chains how many independent chains to run
- jags_data list of data compiled for JAGS, returned by prepJAGS function
- seed input to set.seed function to make the results exactly reproducible
- weekly_params Should weekly estimates of escapement be saved, or only season-wide totals?
- win_model statistical distribution used to model the window counts

```
# name of JAGS model to write
model_file_nm = 'STADEM_JAGS_model.txt'

# what distribution to use for window counts?
win_model = c('pois', 'neg_bin', 'neg_bin2', 'quasi_pois')[2]

#-----------------------------------------------------------------
# run STADEM model
#-----------------------------------------------------------------
mod = runSTADEMmodel(file_name = model_file_nm,
                     mcmc_chainLength = 40000,
                     mcmc_burn = 10000,
                     mcmc_thin = 30,
                     mcmc_chains = 4,
                     jags_data = jags_data_list,
                     seed = 5,
                     weekly_params = T,
                     win_model = win_model)
```

STADEM output

The JAGS object returned by runSTADEMmodel contains many parameter estimates. Some of the most important are the total escapement of various fish.

```
mod$summary[grep('X.tot', rownames(mod$summary)),] %>%
  pander::pander()
```

Table 1: Table continues below

	mean	sd	2.5%	25%	50%	75%	97.5%
X.tot.all	98839	3999	91467	96104	98708	101396	107068
X.tot.day	91817	3687	85044	89268	91663	94108	99469
X.tot.night	7022	714	5913	6548	6944	7412	8546
X.tot.reasc	9442	856.1	7997	8850	9367	9962	11256
X.tot.new.wild	25650	1165	23534	24835	25594	26395	28142

	mean	sd	2.5%	25%	50%	75%	97.5%
X.tot.new.hatch	55329	2471	50852	53589	55256	56952	60353
X.tot.new.hnc	8418	462.4	7571	8098	8395	8721	9364
X.tot.night.wild	1823	203.3	1513	1687	1797	1930	2280
X.tot.reasc.wild	2906	325	2382	2683	2864	3083	3636

	Rhat	n.eff	overlap0	f
X.tot.all	1.003	860	0	1
X.tot.day	1.003	802	0	1
X.tot.night	1.007	1217	0	1
X.tot.reasc	1.002	1202	0	1
X.tot.new.wild	1.004	602	0	1
X.tot.new.hatch	1.001	1824	0	1
X.tot.new.hnc	1.003	719	0	1
X.tot.night.wild	1.008	876	0	1
X.tot.reasc.wild	1.002	1179	0	1

A user might also like to make time-series plots of estimates, to compare with window counts and/or trap estimates (Figure 2).

```
library(stringr)
week_est = mod$summary[grep('^X.all', rownames(mod$summary)),] %>%
  as.data.frame() %>%
  mutate(var = rownames(.),
         week = as.integer(str_extract(var, "[0-9]+")),
         param = str_extract_all(var, "[:alpha:]+", simplify = T)[,3],
         param = ifelse(param == '', 'all', param)) %>%
  tbl_df() %>%
  select(var, param, week, everything()) %>%
  left_join(stadem_list[['weeklyData']] %>%
              filter(window_open | trap_open) %>%
              mutate(week = 1:n()))
```

```
# plot time-series of model estimates, window counts and trap estimates
week_est %>%
  filter(param == 'all') %>%
  filter(tot_tags > 0) %>%
  ggplot(aes(x = Start_Date,
             y = `50%`)) +
  geom_ribbon(aes(ymin = `2.5%`,
                  ymax = `97.5%`),
              alpha = 0.2) +
  geom_line(aes(y = win_cnt / (day_tags / tot_tags),
                color = 'Window (adj)')) +
  geom_point(aes(y = win_cnt / (day_tags / tot_tags),
                 color = 'Window (adj)')) +
  geom_line(aes(y = win_cnt,
                color = 'Window (raw)')) +
  geom_point(aes(y = win_cnt,
                 color = 'Window (raw)')) +
  geom_line(aes(y = trap_est,
```

```
                 color = 'Trap')) +
  geom_point(aes(y = trap_est,
                  color = 'Trap')) +
  geom_line(aes(color = 'Model')) +
  geom_point(aes(color = 'Model')) +
  scale_color_manual(values = c('Model' = 'black',
                                'Window (raw)' = 'lightblue',
                                'Window (adj)' = 'blue',
                                'Trap' = 'red')) +
  theme_bw() +
  theme(legend.position = 'bottom') +
  labs(x = 'Date',
       y = 'Estimate',
       color = 'Source',
       title = paste('All', spp, 'in', yr))
```

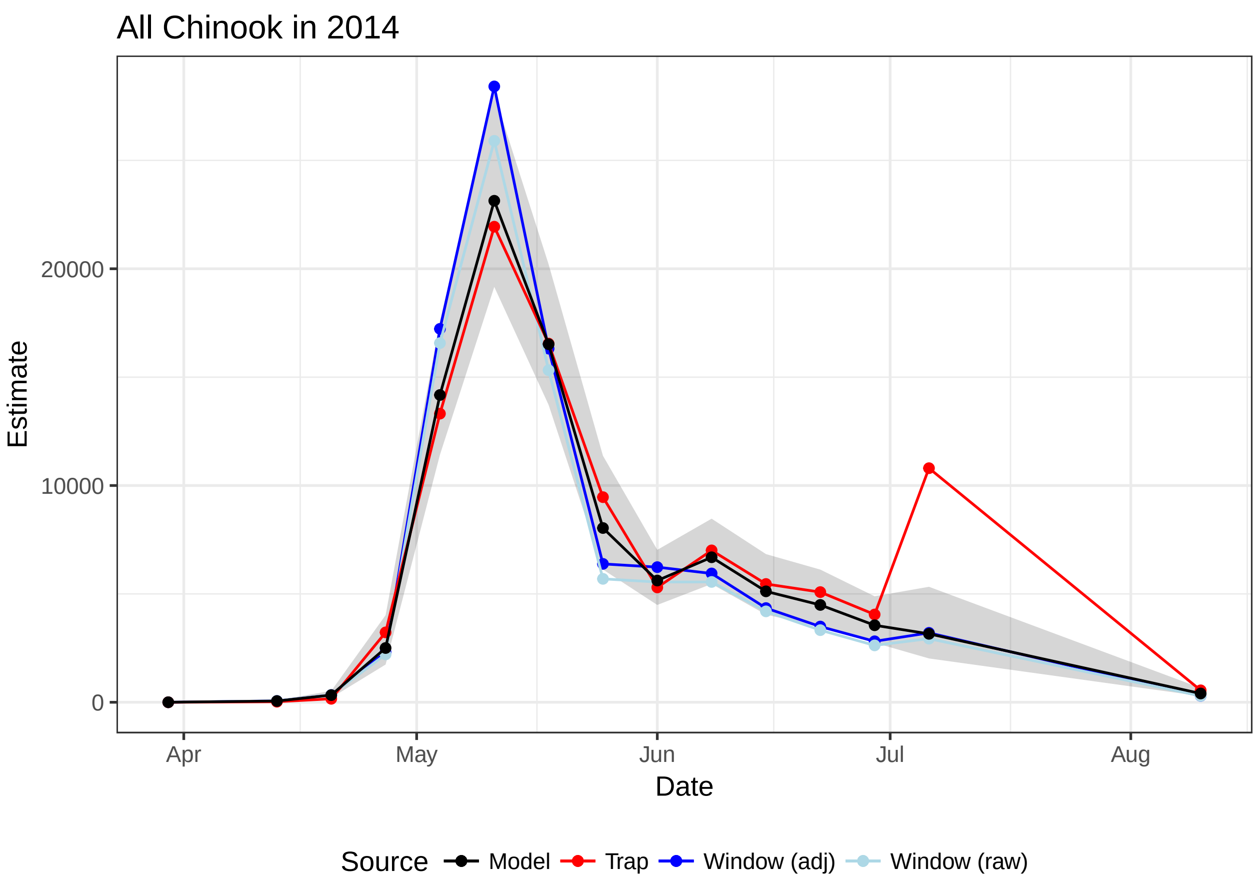

Figure 2: Figure 2: Time-series plot showing estimates of total escapement for Chinook in 2014 , including raw window counts, window counts adjusted for nighttime passage, trap estimates and STADEM estimates.

References

Lindén, Andreas, and Samu Mäntyniemi. 2011. "Using the Negative Binomial Distribution to Model Overdispersion in Ecological Count Data." *Ecology* 92 (4). Ecologial Society of America: 1414–21.