How to Use the Bayesian Yukon River Canadian-origin Chinook Salmon Inseason Projection Model

Aaron Lambert

University of Alaska Fairbanks CFOS

May 22, 2024

Contents

Purpose

This document describes how to run the Yukon River Canadian-origin Chinook salmon inseason projection model. This model can be run one day at a time, one day at a time generating a PDF projection report, or retrospectively to test different model variants. It is recommended that the user be familiar with R.

Folder structure

This model is called from R. Use the following folder structure:

- Working Directory (R project)

- R
- data
- stan
- figs
- output

R scripts will call data, stan models, functions, etc. from these folders.

Run the model for one day in one year

1. Open the R script titled, “Yukon Inseason Forecast Leave One Out ADFG.R”.
2. Set the working directory to the correct location on your computer. This directory should have folders titled R, data, stan, figs, and output.

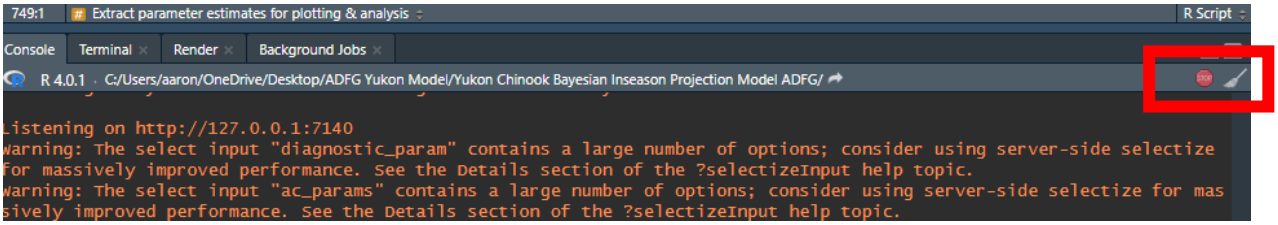
```
46  # set to working directory (Set to directory with all the necessary folders)
47  wd <- "C:/Users/aaron/OneDrive/Desktop/ADFG Yukon Model/Yukon Chinook Bayesian Inseason Projection Model ADFG"
48
49  setwd(wd)
```

3. In the control section specify the model (model.version), year (myYear), and day (myDay). One can specify other inputs such as number of chains, iterations, and thinning rate. Other inputs should not be changed unless they are to be aspects to be tested, such as length of time-series to use in the model for different data inputs.

```
# Control Section ###################################################################
# This is where users can input dates, stan model, and stan model controls.

# Model Version
model.version <- "PSSnormal_ESprop"

# Range of years 1995 to current year
myYear <- 2022

# Range of days 152 -243
myDay <- 200

# MCMC Parameters
n.chains <- 4    # Number of chains to run
n.iter <- 3000   # Number of iterations to run
n.thin <- 2      # How many iterations to thin

# Start Years for Predictors
startYearPF <- 2007    # Preseason forecast
startYearGSI <- 2005   # Genetic stock identification
startYearPSS <- 2005   # Pilot station sonar
startYearEagle <- 2005 # Eagle Sonar

# Wont typically change;
#  *Day 152 is June 1*
startDayPSS <- 148
endDayPSS <- 250
```

4. Run the entire script down to the stan model. The model will take somewhere between 30 seconds and 5 minutes to run depending on the model and how many iterations the model runs for. Typically, 5000 iterations should be sufficient.
5. After running the stan model, one may look at some diagnostic plots. The easiest way to do this is to use SHINYSTAN. Uncomment shinystan::launch_shinystan(as.shinystan(fit)) and run the line of code. This will take you to a web browser pop-up window and the shiny interface.

6. To exit out of shiny stan, make sure to hit the stop button in the r console.



7. To generate plots of outputs such as regressions and density plots, extract the parameter estimates by running the line of code:

```
# Extract parameter estimates for plotting & analysis ##########################
pars <- rstan::extract(fit)
```

8. Now you can run the rest of the script to get various plots. There are numerous plots that can be generated that are not in this script. Have fun playing around with the outputs! The web site https://mc-stan.org/bayesplot/ is a great place to get started with plots.

Run the model inseason and generate a PDF document

When generating reports inseason and in real time, it is convenient to use an r-markdown document that runs the model and generates plots and updates in-text values automatically. Currently, there are two r-markdowns that will run either 1) *PSSnormal_ESprop* or 2) *PSSprop_ESprop*. These are the two models that we recommend that ADF&G use to generate inseason estimates of the Yukon River Canadian-origin Chinook salmon abundance.

1. First, go to https://www.adfg.alaska.gov/index.cfm?adfg=commercialbyareayukon.salmon_escapement and download PSS and Eagle daily observations up to the day that the projection is made.
    a. For Pilot Station, make sure to pull years 1995 – the current year.
    b. For Eagle, 2005 – to the current year
    c. Save the Daily Escapement Table as excel files.
        i. Save to the *Daily Reports* subfolder in the *data* folder.

ii. For PSS, save as: Yukon Escapement Daily 9Aug23 (change to correct date)

iii. For Eagle, save as: Yukon Escapement Daily Eagle 9Aug23 (Change to correct date)

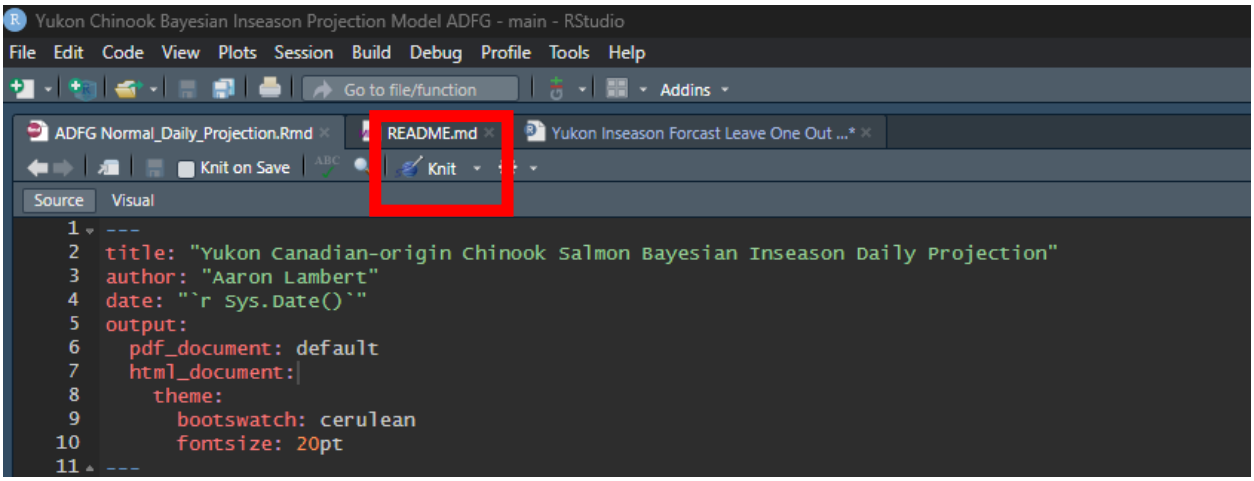
2. Open either ADFG *Normal_Daily_Projection.RMD* or ADFG Proportion Model Daily *Projection.RMD*. These are not located in folders.

3. Change the Pilot Station and Eagle data inputs to the most current escapement tables downloaded in step 1.

```
# PSS and ES data
# Read in the file as xlsx !!!CHANGE FILE HERE Daily!!!!!!!
PSS <- read_xlsx(file.path(dir.data,"ADFG PSS Daily Reports/Yukon Escapement Daily 9Aug23.xlsx"),skip = 3)

ESS <- read_xlsx(file.path(dir.data,"ADFG Eagle Daily Reports/Yukon Escapement Daily Eagle 9Aug23.xlsx"),skip = 3)
```

4. Scroll down to the control section. Here you must select the followings:
   a. projection_day: This is the day the most recent data ends on. Typically, this will be the day prior to when this model is used inseason. This will be the text that will be inserted into the body of the generated document.
   b. myYear: The year the projection is made in.
   c. myDay: In myDay_func, put the month and day the data goes to. For example, if Pilot Station observations run to August 9, then month = 8 and day = 9. This will convert myDay to day of year for running the model.

5. Next, click the knit button at the top. Knit to PDF works well.



6. This will take a few minutes to run and will open in your PDF app. This will get an error if rerunning and the previous output is still open in the PDF app, so close the PDF before running.

Retrospective testing models

When creating new stan models or assessing which years to include in the projection model, use the "Retrospective Testing With Plots ADFG Version.R" script.

1. In the control section, select:
    a. End.year (the last year to include in retro testing
    b. the model's name
    c. iterations to run,
    d. the days and years to test.

```
# Control Section ###################################################

# Year to conduct retro testing through
end.year = 2022

# Input model for retro testing
model.version <- "PSSreg"

# MCMC Parameters
n.chains <- 4   # Number of chains to run
n.iter <- 5000  # Number of iterations per chain
n.thin <- 2     # Thinning rate

# Days to use in retrospective testing runs ##############
# Test days used in full season run (every 5 days starting June 2)
testDays <- seq(from = 153, to = 243, by = 5)

# Years included in retrospective testing
testYears <- c(2007:end.year)

```

2. Next run the function. Note that there are many options in the function call that can drastically alter the model runs. Please look throught th function first to understand these choices. Some models will need more memory in Rstudio. To increase the

memory allocation, run: memory.limit(size = 50000).

```
# # List to store outputs
outputList<-list()

# Increse memory limit (May need this for PSSnormal and PSSlogistic)
# memory.limit(size = 60000)
options(warn = 1)
for(y in c(testYears)){
  for(d in c(testDays)){

    outputList[[paste("",y,"_",d, sep = "")]]<-InSeasonProjection(model.version = model.version,
                                                                  myYear = y,
                                                                  myDay = d,
                                                                  n.chains = n.chains,
                                                                  CAN_hist = CAN_hist,
                                                                  pf_hist = pf_hist,
                                                                  PSS_hist = PSS_hist,
                                                                  PSS_sd = PSS_sd,
                                                                  n.thin = n.thin,
                                                                  n.iter = n.iter,
                                                                  GSI_by_year = GSI_by_year,
                                                                  Eagle_hist = Eagle_hist,
                                                                  normal = FALSE,
                                                                  logistic = FALSE,
                                                                  prior.df.log = logistic.all,
                                                                  prior.df.norm = normal.all,
                                                                  multiplier = 1,
                                                                  startDayPSS = 148,
                                                                  startYearPSS = 2005
                                                                  )
    print(paste("Day =",d,"Year =",y))
  } #dloop

  print(paste("Finally done with year",y))
} #yloop


```

3. Save the ouputs to the “outputs” folder in your directory.

```

# Save the resulting model output here. this will save to the output folder.
saveRDS(object = outputList, file = file.path(dir.output,
                                              "VerPSSreg 27Mar24.RDS"))

```

4. Next, use the retrospective function to generate mean absolute error (MAPE), root mean square error (RMSE), and percent error values (PE). These can be plotted further down in the script.

```
##### Calculations for retrospecitve testing ########################################

# Uses retrospective.function
#  This function calculates RMSE, MAPE, PE, and precision (not currently used)

# PSSreg
RetroList_verPSSreg <- retrospective.function(outputList = outputlist_verPSSreg, # The model results from above
                                      testYears = testYears,                  # The years tested
                                      testDays = testDays,                    # The days across the season tested
                                      CAN_hist = CAN_hist,              #
                                      pf = FALSE)

# Preseason forecast
RetroList_pf <- retrospective.function(outputList = outputlist_verPSSreg, # This can be from any model version
                                       testYears = testYears,
                                       testDays = testDays,
                                       CAN_hist = CAN_hist_new,
                                       pf_hist = pf_hist,
                                       startYearRetro = 2007,
                                       endYearRetro = 2022,
                                       pf = TRUE)
```