WDFW SOS Analysis

Thomas Buehrens

Contents

This document was generated on 11/08/2024.

Purpose

The purpose of this document is to record the steps and code necessary to reproduce the 2020 State Of the Salmon (SOS) analysis completed by WDFW. A full report with methods, results, and conclusions is found here: ***Full WDFW SOS Report Link***

Requirements

All analyses require R software **(link)** (v3.4.3) for data retrieval, data processing, and summarizing model results, and Stan software **(link)** for Hamiltonian Monte Carlo (HMC) simulation.

Functions

We also need a couple of helper functions which we will load from the functions folder, which we will load using the walk() function from the purrr package (which we will install if it is not already installed).

```
# ===========================================
# Source function files in functions folder
# ===========================================
```

```r
if (!require("purrr")) {
  install.packages("purrr")
  library("purrr")
} else {
  (library(purrr))
}
path <- "functions/"
files <- paste0(path, list.files(path))
purrr::walk(files, source)
```

Packages

In addition to purr, 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.

```r
# ===============================================
# Load packages, install and load if not already
# ===============================================
using(
  "tidyverse",
  "rstan",
  "reshape2",
  "plotly",
  "reshape2",
  "metR",
  "directlabels",
  "RColorBrewer",
  "MASS",
  "tidyr",
  "ggrepel",
  "readxl",
  "ggforce",
  "readr",
  "ggplot2",
  "gridExtra",
  "tinytex",
  "stringdist",
  "DBI"
)
```

User Inputs

We need to specify data source file names, list some manual data filter conditions, and specify which ESUs and DPSs we would like to analyze to estimate trends and smoothed abundances for:

```r
# ========================
# designate data file names
# ========================
# ESU_DPS_list file name (also lists which ESUs to use CA for)
ESU_DPS_list <- "ESU_DPS_List.csv"
# ESU_DPS_list file name
ESU_DIP_list_all <- "ESU_DIP_list_all.csv"
# Recovery_Goals file name
```

```r
Recovery_Goals <- "Recovery_Goals.csv"
Recovery_Goals_LUT_edited <- "recoverygoals_LUT_edited.csv"

# ========================================
# set date stamp for file names and  plots
# ========================================
# set data date--this will name files and
# folders and tell the analysis code which
# date's data to use
data_date <- "2024-11-05" # Sys.Date()


# ========================================
# set manual filter conditions for CA data
# ========================================
# list of pops by ESAPOPNAME that 1) do not
# have NOSAIJ or NOSAEJ data AND 2) TSAIJ
# and TSAEJ have too many hatchery fish to
# be treated as NOSA substitutes

HatchPops <- c( # total spawners includes substantial hatchery fish)
  "Green River - winter Steelhead",
  "Mid-Hood Canal - fall Chinook salmon",
  "East Hood Canal Tributaries - winter Steelhead",
  "Skokomish River - winter Steelhead",
  "West Hood Canal Tributaries - winter Steelhead"
)
# ==================================================================
# POPFIT_exceptions list of pops where POPFIT != "same" or "multiple" but is "close enough"
# that we should use anyway; listed by COMMONPOPNAME2
# ==================================================================
POPFIT_exceptions <- c( # no spawners in chinook river
  "Grays and Chinook Rivers - fall Chinook salmon",
  # ok to use only spawners above KFH
  "Kalama River - spring Chinook salmon",
  # assumes no mainstem spawners
  "Lower Cowlitz River - late Coho salmon",
  # assumes no mainstem spawners
  "Lower Cowlitz River - winter Steelhead",
  # WA only; use 1/2 recovery goal
  "Upper Gorge Tributaries - fall Chinook salmon",
  # WA only; use 1/2 recovery goal...but estimates still missing white salmon
  "Lower Gorge Tributaries - late Coho salmon",
  # assumes no fish spawn in NF and mainstem Toutle below SRS
  "Toutle River - fall Chinook salmon",
  # some years mislabeled as partial instead of multiple
  "Upper Cowlitz River - spring Chinook salmon",
  # Tucannon Steelhead
  "Tucannon River - summer Steelhead",
  # North Fork Toutle
  "North Fork Toutle River - winter Steelhead",
  # Hood canal summer chum
  "Hood Canal - summer chum salmon",
  # Elwha steelhead--recovery goal is for whole basin. abundance estimate only includes winter runs; ma
  "Elwha River - winter Steelhead"
)
# ============================================================================
# special cases to remove in order to get 1 abundance data pt per pop per year;
# these are supplied as a listed of quoted filter conditions
# ============================================================================
specialcaselist <- list(
  # 1: #below dams estimate is sketchy
  quote(!COMMONPOPNAME2 == "Lower Gorge Tributaries - fall Chinook salmon"),
  # 2: Entiat data lists multiuple  methods as best....use method 4 since full timeseries.
  quote(
    !(COMMONPOPNAME2 == "Entiat River - spring Chinook salmon" &
      METHODNUMBER %in% c(1, 2, 3))
  ), # duplicate
  quote(
    !(COMMONPOPNAME2 == "Entiat River - spring Chinook salmon" &
      METHODNUMBER == 4 & SPAWNINGYEAR == 2019)
  ),
  # 4: Use method 2-3 for Joseph Creek Steelhead; Patch Occupancy model
  quote(
    !(COMMONPOPNAME2 == "Joseph Creek - summer Steelhead" &
      OTHERDATASOURCES %in% c("Nez Perce Tribe | Confederated Tribes of the Umatilla Indian Reservation
    )
  )
)

# ============================================
# Set parameters for summarization of results
# ============================================
# Only include population in ESU summary that include recovery goals?
Withgoalsonly <- "yes"
# Number of years to calculate geomean of smoothed abundance
geomeanyears <- 5
# last year in geomean abundance calculations
lastyear <- 2023
# set number of years for forward projection of ESUs
futureyears <- 5
# exlcude populations from geomean smoothed abundance calculation
# that have no new observed data in period geomean is being calculated for?
filtergeomeansforpopswithnonewdata <- "No"
# use only data since listing
databeforelisting <- "No"
```

Data Preparation

Here we will use a data filtering algorithm described in the full report to select appropriate natural origin spawner abundance data for use in the status and trend analysis. We will then prepare analysis input files (for Stan), and we will plot the raw abundance data. An abbreviated description of the algorithm used to identify final data used for the analysis follows:

1. We downloaded the entire Coordinated Assessments database here **(link)**. The retrieval date is part of the file name (located in the "data" folder).

2. Data were limited to ESA/DPS-listed populations located partially or entirely in Washington State.
3. Data were limited to those with "POPFIT" designated as "Same" or "Multiple", indicating that the population estimate had complete spatial and temporal coverage (as opposed to commonly monitored indexes of abundance measured at the sub-population scale, for which comparison with population-level recovery goals is inappropriate). There were 12 exceptions to this, which are documented as "POPFIT_exceptions" in the user inputs.
4. Data were limited to those for which "BESTVALUE" was designated "Yes" indicating that of multiple potential abundance estimates available for that year and population, a particular estimate was the best estimate.
5. A series of population-specific manual filters (4 total) were applied to eliminate duplicate data sets (i.e., more than one data point per population per year), which are documented as "specialcaselist" in the user inputs.
6. An algorithm selected the type of data to use for the analysis, looking for data types in the following order and stopping when the first data type was found with records: Natural Origin Spawner Abundance Including Jacks, Natural Origin Spawner Abundance Excluding Jacks, Total Spawner Abundance Including Jacks, Total Spawner Abundance Excluding Jacks.
7. A final population-specific manual filter was applied to eliminate datasets for which Total Spawner Abundance was not an appropriate surrogate for Natural Origin Spawner Abundance because the population contained a non-negligible proportion of hatchery spawners. For the vast majority of populations, if any data was available, natural origin spawner data was available. For some, our algorithm only identified total spawner data (i.e., hatchery and wild). We manually inspected the list of populations for which total spawner data was selected to identify those for which total spawners potentially contained a non-negligible proportion of hatchery origin spawners. These populations (n = 5) were filtered out, leaving only the total spawner data that could be appropriately treated for analysis purposes as natural origin spawner abundance. The filter list is available in user inputs as "HatchPops". The final data type used for each population is listed in the field "final_abundance_data_type" in the "All_Data" file in the results folder here: **(link)**.
8. Data were then filtered to only include years from ESA listing (which varied by ESU/DPS) through present as our focus was on status and trend since the ESA listing.
9. Finally, we compared our final population list to the complete list of populations to determine if any populations had suitable data that was not yet available in Coordinated Assessments. Twenty-one populations were identified for which data was available either in WDFW-SCORE here **(link)**, or through co-manager agreed to datasets provided via personal communication. These data were formatted to match the Coordinated Assessments data, loaded via the "prepNONCAdata" function, and merged with the filtered Coordinated Assessments data to complete the final analysis. This supplemental data not from coordinated assessments may be found in the "Raw data for pops not in CA" file in the "data" folder here: **(link)**

Other data inputs (available in the "data" folder) included recovery goals (pulled from various NOAA-adopted recovery plans, of which we used either the minimum viability goals or low productivity goals), a list of the ESA-listed ESU/DPSs in Washington ("ESU_DPS_List") that included listing dates, and a list of all populations by listed ESU/DPS in Washington ("ESU_DIP_List_all").

```
# prep and filter SPI data
dat <- prepare_SPi_data(
  data_date = data_date,
  ESU_DPS_list = ESU_DPS_list,
  Recovery_Goals_LUT_edited = Recovery_Goals_LUT_edited,
  Recovery_Goals = Recovery_Goals,
  POPFIT_exceptions = POPFIT_exceptions,
  specialcaselist = specialcaselist,
```

```r
  databeforelisting = databeforelisting
)

#### update log (compare years,pops, data types this year vs. last)
read_csv(here::here("year_pops.csv")) %>%
  full_join(read_csv("https://raw.githubusercontent.com/tbuehrens/WDFW_SOS_Analysis/refs/heads/2022/yea
    by = join_by(ESU_DPS, COMMONPOPNAME2),
    suffix = c(".2024", ".2022")
  ) %>%
  mutate(Updates = ifelse(maxyr.2024 > maxyr.2022, "", "not updated")) %>%
  write.csv("update_log.csv", row.names = F)



# make analysis files for status and trend analysis
# (we use bind_rows to combine CA and Non-CA data)
makefiles(
  data = dat,
  data_date = data_date,
  databeforelisting = databeforelisting
)

# make plots of raw abundance data by population and ESU/DPS
plotfunc(
  data = dat,
  data_date = data_date,
  ESU_DPS_list = ESU_DPS_list,
  Recovery_Goals = Recovery_Goals,
  Withgoalsonly = Withgoalsonly
)
```

Run Status and Trend Analysis

Here we will run the status and trend analysis using Stan via rstan. We will then tidy up the results to prepare for summarization and plotting. You must have rtools installed and stan installed for this to work:

```r
# run analysis for any populations that is not yet complete
analyzedata(
  data_date = data_date,
  ESUsubset = c(
    "Lower Columbia coho",
    "Lower Columbia Chinook",
    "Lower Columbia steelhead",
    "Mid-Columbia steelhead",
    "Snake River spring and summer Chinook",
    "Snake River steelhead",
    "Upper Columbia spring Chinook",
    "Upper Columbia steelhead",
    "Snake River fall Chinook",
    "Puget Sound steelhead",
    "Puget Sound Chinook",
    "Ozette Lake sockeye",
    "Hood Canal summer chum",
    "Lower Columbia chum"
  ),
  ESU_DPS_list = ESU_DPS_list,
  cores = 4,
  chains = 4,
  iter = 2000,
  warmup = 1000,
  thin = 1,
  control = list(adapt_delta = 0.95)
)

# combine results output
combineresults(data_date = data_date, Recovery_Goals = Recovery_Goals)

# merge results with raw data
resultsdata <- makeresultsdata(
  data_date = data_date,
  data = dat,
  ESU_DPS_list = ESU_DPS_list, Recovery_Goals = Recovery_Goals,
  Smoothed_Abundance = Smoothed_Abundance
)
```

Summarize Status and Trend Results

Here we will summarize our status and trend results, first at the population level, and then at the ESU/DPS level. This code may take up to 5-15 minutes to run; adjust expectations accordingly.

```r
# make plots of raw abundance data by population and ESU/DPS
# with smoothed abundances added as lines
resultsplots <- plotfunc2(
  resultsdata = resultsdata,
  Recovery_Goals = Recovery_Goals,
  Withgoalsonly = Withgoalsonly,
  data_date = data_date
)

# make ESU and DIP Results
ESU_DIP_results <- make_ESU_DIP_results(
  resultsdata = resultsdata,
  lastyear = lastyear, data_date = data_date,
  geomeanyears = geomeanyears,
  filtergeomeansforpopswithnonewdata = filtergeomeansforpopswithnonewdata
)

# plot results
esu_results_plot(
  ESU_DIP_results = ESU_DIP_results,
  data_date = data_date,
  futureyears = futureyears,
  resultsplots = resultsplots,
  bypopsplots = "No",
  geomeanyears = geomeanyears,
  lastyear = lastyear
)
```

Reproducing this pdf or html page

In order to reproduce this pdf or html page you need to have a LaTex application installed. Running this snippet of code will automatically install tinytex on your machine so you can render pdfs and html:

```
# tinytex::install_tinytex()
```

This document was compiled at 2024-11-08 09:22:14.362534 using the following packages:

```
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United States.utf8  LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8 LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/Los_Angeles
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] DBI_1.2.3             stringdist_0.9.12     tinytex_0.53          gridExtra_2.3
##  [5] ggforce_0.4.2         readxl_1.4.3          ggrepel_0.9.6         MASS_7.3-61
##  [9] RColorBrewer_1.1-3    directlabels_2024.1.21 metR_0.15.0          plotly_4.10.4
## [13] reshape2_1.4.4        rstan_2.32.6          StanHeaders_2.32.10   lubridate_1.9.3
## [17] forcats_1.0.0         stringr_1.5.1         dplyr_1.1.4           readr_2.1.5
## [21] tidyr_1.3.1           tibble_3.2.1          ggplot2_3.5.1         tidyverse_2.0.0
## [25] purrr_1.0.2
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1   viridisLite_0.4.2  farver_2.1.2       blob_1.2.4         loo_2.8.0
##  [6] fastmap_1.2.0      lazyeval_0.2.2     tweenr_2.0.3       digest_0.6.37      timechange_0.3.0
## [11] lifecycle_1.0.4    magrittr_2.0.3     compiler_4.4.2     rlang_1.1.4        tools_4.4.2
## [16] utf8_1.2.4         yaml_2.3.10        data.table_1.16.0  knitr_1.48         labeling_0.4.3
## [21] htmlwidgets_1.6.4  bit_4.5.0          pkgbuild_1.4.4     curl_5.2.3         here_1.0.1
## [26] plyr_1.8.9         withr_3.0.1        odbc_1.5.0         grid_4.4.2         polyclip_1.10-7
## [31] stats4_4.4.2       fansi_1.0.6        colorspace_2.1-1   inline_0.3.19      scales_1.3.0
## [36] isoband_0.2.7      cli_3.6.3          crayon_1.5.3       rmarkdown_2.28     generics_0.1.3
## [41] RcppParallel_5.1.9 httr_1.4.7         tzdb_0.4.0         cachem_1.1.0       parallel_4.4.2
## [46] cellranger_1.1.0   matrixStats_1.4.1  vctrs_0.6.5        V8_5.0.1           jsonlite_1.8.9
## [51] hms_1.1.3          bit64_4.5.2        glue_1.7.0         codetools_0.2-20   stringi_1.8.4
## [56] gtable_0.3.5       QuickJSR_1.3.1     quadprog_1.5-8     munsell_0.5.1      pillar_1.9.0
## [61] htmltools_0.5.8.1  R6_2.5.1           rprojroot_2.0.4    vroom_1.6.5        evaluate_1.0.0
## [66] backports_1.5.0    memoise_2.0.1      Rcpp_1.0.13        checkmate_2.3.2    xfun_0.47
## [71] pkgconfig_2.0.3
```