Update Log - Oklahoma Fishery Analysis Application

Main App

1. 9/3/18 - Dray
   1. Problem: catch curve was incorrectly calculating mortality. I had calculated the natural log of the catch before putting data into catchCurve() function (FSA). Turns out catchCurve() calculates natural log on its own too…so essentially, it was taking a log of a log…no wonder why they were always such low mortality rates.
   2. Fixed: I reworked code to account for this. Just removed step where natural log of frequency was previously calculated and renamed a few things in both the catch curve plot function and the mortality table function.
2. 9/3/18 – Dray
   1. Problem: Age database needed updated gear codes – also went through age data validation code
   2. Fixed: compiledagedata.csv was updated with new gear codes
   3. Note: All fields in compiledagedata.csv should comply with data validation rules
      1. Exception: Day field – a few records (very small handful) had “.” for Day. I deemed this okay, but Validation App makes it a required field from now on.
3. 9/30/18 – Dray
   1. Problem: Main app wouldn’t work with imported data that had periods in the TL\_mm or Wt\_g fields (R reads this field as a factor).
   2. Fixed: Wrote functions that change any periods to NA, then changes TL\_mm and Wt\_g fields to numeric.
4. 11/15/18 – Dray
   1. Problem: Change simple CV to RSE in CPUE tables. Also change calculation method for N RSE(25) and RSE(40)
   2. Fixed: What ODWC actually needed was RSE. Still not a completely correct method, but better than nothing. Methods for calculating N RSE(25) and N RSE(40) were incorrect. Changed to method used by Dumont and Schlechte (2004)… N = (CV/RSE)2
5. 11/15/18 – Dray
   1. Problem: Reference R2 did not account for “weighted” regression. Ogle’s catchcurve function does this automatically as a TRUE/FALSE argument, so I didn’t know how to do this specifically. R2 value isn’t included in Ogle’s functions, so when calculating it, I had to do the weighted regression manually as well.
   2. Fixed: Figured out how to do this…now calculates weights in first linear model and runs a second linear model with the weights.
6. 11/15/18 – Dray
   1. Problem: Including sex-specificity for Paddlefish
   2. Fixed: modified species codes in speciesinfo.csv…320 for Paddlefish (all), 320.1 for Paddlefish (male), 320.2 for Paddlefish (female). Also modified the WSnames.csv (links ODWC species codes with species names needed for relative weight calculations in FSA functions) to account for new sex-specific codes (each sex has different relative weight calculation, including one for an overall paddlefish)…Brown and Murphy 1993. Also updated gabelhousenames.csv to include all 3 paddlefish spp codes (used in FSA functions to reference PSD groupings for species).
7. 11/15/18 – Dray
   1. Fixed: Updated Total Effort Table to use Gear.Length (minutes) for electrofishing samples (consistent with what is used for CPUE). Also fixed the error produced when uploading an independent sample (one of the functions was referencing a selectize input from the first tab.
8. 3/2/2020 - Dan
   1. Fixed: update to Shiny package broke renderDataTable function.  Research into this indicates this function is being depreciated and the same function from the DT package is to be used instead. I added DT package to the app and changed the code to specifically call DT::renderDataTable to make tables.
9. 1/17/2021 – Dan
   1. Problem: CPUE by PSD size class often add to more than the total CPUE…typically when there were sites with Species.Code=98 (no fish in sample). Upon investigating, we originally had deleted any row that did not have a Gabelhouse.Name. This means that any sample that either had no fish, or only caught fish that do not have PSD size classes defined would be removed, so when the addZeroCatch function (which is being depreciated) was run, the sample was not there to receive zeros. This was necessary as if you run psdAdd() on spp with missing species names, there was a bug that added additional rows and made it so the list of PSD names and the original file do not have the same number of rows (causes cbind or mutate to throw an error due to mismatch).
   2. Solution: I contacted Derek Ogle about the bug and he quickly rebuilt psdAdd so it no longer had this behavior. I then rebuilt the code using a dplyr approach and waited to throw out the spp that do not have PSD size classes until after using complete() (which is the preferred approach over addZeroCatch). I also added logic to find fish where no TL data was present and deleted these also so they do not appear in the table (they may have a total CPUE, but if no TL’s were taken, can’t meaningfully express this in PSD-based CPUEs). I also allowed trophy size to be in the table. The sizes sort fine using dplyr verbs instead of the old approach.
10. 1/17/2021 - Dan
    1. Made several other small changes:
       1. Started replacing plyr’s join with dplyr left\_join.  However, there are places where we intentionally made gear code or species code character rather than numeric and left\_join checks this before running and throws an error so these are left.  We also have 2 instances of join\_all that I’m not sure has a dplyr counterpart, but I’ll need to research.  I’d like to get us off of plyr entirely some day if we can as loading order is an issue that causes lots of problems (especially if we want to load dplyr in ui.r, which is not done now...would also have to load plyr and do so first to avoid plyr overwriting dplyr.
       2. Changed renderTable function for PSD size definitions (both mm and inch) so NA displays as blank space in table.
11. 10/18/2021 Dan modified lines that created “w” as length category width for lencat() function as the original code could not deal with TL/30 = exactly 20 or exactly 15 (roughly lines 1154-1162 and again on 1181-1189). Found in response to bug report by Mike Hollie.
12. 10/19/2021 Deployed new data Ashley had sent me last month…several sampling files were excluded due to not being properly formatted/validated. All age data were able to be fixed and used (but loaded some in a separate run after realizing the only problem they had was including a couple blank rows at the bottom of the file.
13. 11/28/2021-5/19/2022 (deployed to shiny server 5/19/22).
    1. Rolled out the percentile tab (after a couple months of beta testing by a group in ODWC)
       1. Included percentile data is very selective and does not include records with heavy amounts of duplication…throws away about 10% of the available data, but the duplication issue could easily double or triple CPUE values if it is true duplication, so this is better than including.
    2. Updated to Shiny v 1.7.1, which broke percentile tab’s statements that check to see if first row of output tables are all NA values (MARGIN = 1, FUN = function(x) all(is.na(x)))==F). I rewrote these so they now check if the number of rows with NA in Species Name column is < the total number of rows (i.e., they are not all NA).
    3. Developed modal dialog to allow user to pick how to use the app (start with last 5 years, last 10 year, all years, or user upload data) to speed startup time in cases where full database is not needed.
    4. Rewrote code to cross-populate selections across selectize box pairs.
       1. After trying lots of options, settled on getting rid of the cross-talking code and name boxes. Instead, I add a check box that determines if codes are used (default) or names are used (if you check the box).
       2. I then made the default search a combination of name and code (name first so sorts by names)…this way you can search on both name or code at once. Added a check box to just search on code (which sorts by code). May delete the extra code for the check box as I’m not sure anyone needs that, but will see what response I get at ODWC workshop before deciding.
          1. I’m saving the main datafiles with new fields for name\_code on lake and gear, and code\_name for spp. This is considerably faster than creating these on the fly when loading. It is noticeably slower to read the file in, but it adds about 0.6 seconds for database with all years in it whereas it is many seconds to add the columns (might be 10x slower, but I did not formally benchmark).
       3. Added code that filters remaining choices to only those that still are options in the database given what has already been selected
       4. Also added code so that if only one item is available as an option, it automatically becomes selected. This means if you upload your own data and there is just one data set in it, it auto-selects everything.
       5. Old things I tried with cross-populating name-code boxes…I’ll delete this once I deploy everything and know I do not need it.
          1. Originally had all in one observe() function. Separating each box in its own observeEvent triggered by changing the input$??? where ??? is the name of the selectize box value for the alternate pair prevented infinite loop, but it also makes it so that you cannot delete the last selected value and have that propagate to the paired selectize box (it always leaves one item in there). I finally settled on putting each pair in an observeEvent function and this seemed to solve both issues. Using observeEvent is also necessary for filtering remaining choices (see below) as this cannot be done with observer updateSelectizeInput statements as it creates infinite loops.
          2. I also added code to check new values against the previous values to stop any activity if there is a “new” value that does not actually represent a change (also helps prevent infinite looping)
             1. To make this work, I created selBoxOld$??? values to save the last state of each box…if a change is detected, but the new value matches the old value, nothing is done by the observeEvent box. This effectively kills the infinite loop problem. I then needed to use if else code in the updateSelectizeInput to set blank boxes to character(0) to get it to make the box blank
          3. Still a bit imperfect:
             1. One bug remains…for some reason, if you delete all values from one of the boxes on the right (names rather than code boxes), it does not auto-update the choices in all boxes, but rather is stuck on whatever choices were associated with the last selected value(s)
             2. Occasionally things hang a long time or simply delete a selection….cannot reproduce this reliably enough to debug and it is rare. Can be worked around by just waiting, then reselecting if the previous selection did not take.
    5. Updated read\_csv to read.fst. Started by chancing to fread from data.table package and stuck with .csv files…this approach was about 10x faster at loading large files (very noticeable improvement). Further, used fst package and save main databases as \*.fst files because this is even faster (fread on csv file takes about 1.5x longer than read.fst) and I can save the setkey values of data.tables this way…which means we save time on loading and setting keys while still getting speed of data.table-based data.
       1. Should someday finish rewriting code to keep the main data in a data.table rather than data frame. I’ve mostly done this, but did not rewrite all code on the multispp, single-spp, and percentile output tabs (but all selection code now uses data.table functions).
    6. Redesigned code for user-uploaded age data so this uses the same routine as the built-in age data
       1. Fixes bug where if user uploaded file with more than one spp, it would produce ALK using all spp combined.
       2. Added code so that if only one species, lake, and year were in uploaded data, it auto-populates the selectize boxes
       3. As above, combined spp code\_name and did away with cross-talking code-name selectize boxes.
    7. Fixed bug in CPUE table where Species.Code 98 “No fish in sample” was showing as a row in the final CPUE table.
    8. Changed equation for age-frequency figure to multiply by 100 to make percentages not proportions (Y-label said percent, but decimals were showing).
    9. Modified code to prevent as many warning/error messages
       1. Rewrote abiotic data table on Catch Analysis tab to explicitly deal with all NA data
    10. Added theoretical maximum age, observed maximum age, and two estimates of natural mortality based on theor max age or von bert parameters (includes download button for nat mort estimates)
    11. Modified user manual to detail percentile tabs and account for several of the above changes.
14. 6/8/2022 Added checkbox and code that allows ALK to extrapolate to fish smaller than those aged. Saw that ODWC frequently had a floor on fish sizes that are aged (e.g., crappie less than 100mm are always age-0 and are never aged). This produced inaccurate length at age 0 as only fish 100mm and larger were considered in the calculation given that was the smallest size aged.
    1. Default is to allow this. Only possible problem would be if only age-1+ are aged…then the check box might try to parse age-1 vs age-0 incorrectly (or might assume all smaller fish are age-1 and no age-0 exist)
15. 9/1/2022 – Modified to account for Verified.TL and Verified.Wr files in user-uploaded files rather than common Verified.TL.Wr column (was afraid users would fix TL issues but miss Wr issues that are left related to weight errors).