Automated Assessment Update Instructions

General Instructions

The symbol #[~] marks the pieces of code that need your attention. If you change anything else, make sure you save an earlier version! Searching for #[~] is a quick way to locate all the places where you should make changes.

Setting up an automated summary report for your species of choice requires a few data items. All of these should be in .csv format with column headings as listed below. If your species does not have data for all of the column headings, don't worry, just delete the columns you are not using. The order of the columns is not important, but you will want to pay attention to the column names (you will need to use them later) and try not to use spaces, or "." in them.

IMPORTANT - put these somewhere in your ***network*** homework directory

- Catch data: [Year, discards and landings (recreational and commercial, up to 8 fleets), and total catch (however you define it)]
- Model results: [Year, SSB, cv(SSB), F, cv(F), Recruits, cv(Recruits), SSB, F and recruits from the old model] NOTE: if you are using VPA - put in a cv for the terminal year only, leave the other years blank.
- Survey results: [Year, each survey index (up to 8) and its cv]

Steps for getting started:

1) Locate the R script
"\\net\home2\dhennen\EIEIO\MakeUpdateDoc_Server1.r"

2) Copy it into a directory (e.g. make a new one called "AutoAssessUpdate") in your ***network*** home folder.

3) Click on the link below ot open a browser window and type in the URL (you will have to log in with your network pw)

`http://venus.nefsc.noaa.gov`

4) Open the R script (from step 1) *e.g.* (~/swigley/AutoAssessUpdate/MakeUpdateDoc_Server1.r). from within Rstudio (control+o, or click File/Open, or click the folder icon on the upper left side of your window).

5) Change the directory path on line 15 of the R script to match the directory you set up in step 2 (but don't change anything else yet!).

This R script should now produce an example summary document in your directory when you hit the source button (upper right corner of the top left panel). It will copy some new files and create sub directories in the directory you created. ***Please don't rename the sub directories!*** Feel free to add items to them, but leave the names as they are.

6) Overwrite the example data in the folder "data" that now exists in the directory you set up in step 2. If you don't have data for some columns (e.g. cv's for biomass estimates), just delete them.

Note: You can change the names of the files, but you will need to uncomment line 22 and enter your new file names in the correct order! R needs to know what files to look for.

7) Begin looking for #[~] and modifying the example R script to fit your species! Please take the time to notice and read the explanation in the comments (look for the # and green type).

Tips

There are many items you will need to change in order to adapt the working example to your species. Not everyone will need all of the items. For example, you may not have cv around your F estimates. That is fine, you can change the value of the item to NULL or delete, or comment it out - all should work. It is possible that I didn't error trap all of these, so if you delete or comment out an item and your script chokes, but it runs when you NULL it - please let me know!

If you see a number provided in quotes in the working example, provide your number in quotes as well. This is usually done in cases where table cells might contain a number and its confidence interval, so the quotes provide a simple way to present them. Changing from a string (something in quotes) to a number (no quotes) or vice versa, may cause some problems.

It is good practice to close your data files before running the R code that reads them as your data file editor may not allow multiple programs to read them simultaneously.

Working on the server can be weird. If there is a lot of activity on Neptune, it may take a while for your .pdf document to reload. Check the time stamp on your current version before getting frustrated that your changes are not showing up!

Finally if you see a table or figure showing up that doesn't belong (*e.g.* you don't have projections, but you are seeing a projection table), it is likely that the table or figure is left over from when you compiled the example script in step 5 above. To fix this, just delete everything in your "tables" and "figures" directories and source again. All the tables and figures will be made anew and the offending leftovers will not.

R Basics

Skip this if you already use R

There are only a few things you need to know about R before using this source code.

The R assignment character is $<-$, which is directional. So if you want to assign the value "8" to a variable called x, you type "$x <- 8$" or "$8 -> x$". You can also use "=" if you like in this application.

We really only need 2 of R's many data structures for our purposes, vectors and stings. Vectors are simple 1 dimensional arrays and they are created using "c()", for example $x <- c(8, 6, 435)$ creates a vector containing the values 8,6, and 435. Strings are character variables and they are created using "". For example, $y <-$ "$BigShark$" gives y the text "Big Shark".

Most of the assignments to string variables in this code is made using the "paste" function, which simply puts several different strings together. For example $y <- paste($"Big","$Shark$"$)$ will give y the same value as the above example. The different components must be separated by commas and if you want to put literal text in, you must use quotes (*e.g* $y <- paste($"Big", $Shark)$ would give an error unless you have defined Shark as a variable). This is useful for concatenating strings and other variables as well. For example, if you have defined $x <- 8$ and then type $y <- paste(x,$"$Sharks$"$)$ you will assign y "8 Sharks". The only quirky part of this function is that you can be tripped up by spaces between the various parts. There is an argument called "sep" that can be placed at the end of the paste statement that defines the character you want to separate each of the things you are pasting together. For example, $y <- paste($"Big","$Shark$", $sep =$ " $*$ "$)$ would give y the value "Big*Shark". The "sep" argument is a space by default, but I usually turn it off (sep="") and add my own spaces.

R does not like you to have column headers (names of data vectors) in your data files that start with a number (*e.g.* 3yrMovAvg is bad, but MovAvg3yr is OK). R will typically add an "X" to your variable name if it starts with a numeric and this will mess up the script.

R is free and open source and there are a gazillion resources on the internet. If you have a question, I recommend you simply type it into a search engine. You will likely be surprised at how easy it is to find answers!

R Studio Basics

Skip this if you already use R Studio

The program we will be using to edit code is called R Studio. It will run from the server Neptune, which will have some implications discussed below.

R Studio has 4 panels. The only ones you need to look at are the "Source" and "Console". "History" and "Workspace" are less important and you can minimize them to give you a better view of the important stuff. "Source" will hold your R source code and the console will show you the result of any commands you execute. In the source window there are 2 buttons you will want to familiarize yourself with: "Run" which will execute the line your cursor is in, or the block of code you have selected. "Source" will execute the entire script. You will probably use both. You can save your changes using the usual File/Save combination or control+s. If you don't like the layout of the panels you can change it with Tools/Options/Pane Layout.

One other nice feature of Rstudio is that you can block any chunk of code between curly brackets {} and then hide that chunk by collapsing it. All you need to do is click on the little triangle that appears to the left of the left bracket and it will collapse everything to the closing bracket. To expand, click the triangle again. This is a useful way to not have to look at code you don't need to change or have finished changing.

Running R on the Server

The main reason for doing this is that you don't have to add any software to your desktop. All the packages and compilers you need are already installed and will be maintained by others. Finally, it saves me from having to write several versions of the R and LaTex code that will run on the various operating systems run in the branch.

You may have to get used to a few small differences between Linux and Windows, but they are very minor. The main one is that the paths have "/" instead of "\" separating the directories.

You may also experience slower than normal compile times if the server is busy. Not much we can do about that, but at least your local machine won't

be slowed down!

Debugging

R
Good debugging practices will serve you well as always. Try to isolate your error (if Rstudio does not helpfully provide you with a line number), by running small sections of your code sequentially. If the source of your error is a variable that does not appear in your R script, it is most likely a variable that comes from one of the functions in EIEIO_functions.r. You won't able to edit these functions because they are write protected, but you can look at them. To see a function in R you just need to type the name of the function. The source code for it will appear in your console window. I can help you figure out what is happening when errors of this type pop up and, if necessary, I will fix any bugs you happen to find.

LaTex
The R code will create a file called BigCall.tex, which is called by the LaTex document MakeUpdate2.tex. If you know LaTex and you are experiencing LaTex errors when sourcing your R script, it may be worth while to look at BigCall.tex to see if there are obvious errors. You can also look at MakeUpdate2.log. MakeUpdate2.tex is mainly a driver for items that are defined in BigCall.tex and should not be the source of any LaTex errors. Let me know if you are experiencing LaTex based errors and can't figure them out.

LaTex Variables

The following are the R variables created for the EIEIO project that operate in the LaTex world. You can use any of the following ".tx" variables in R to produce symbols and effects in LaTex. For example putting "$paste(SSBtarg.tx,$ " = 1035 (mt)", $sep =$ "")" into a figure caption variable will eventually produce "SSB_{target} = 1035 (mt)" in that caption in the LaTex summary document. R variables are as follows:

Fisheries symbols
BMSY.tx = B_{MSY}
BMSYproxy.tx = $B_{MSY}proxy$
FMSY.tx = F_{MSY}
FMSYproxy.tx = $F_{MSY}proxy$
F30.tx = $F_{30\%}$
F35.tx = $F_{35\%}$
F40.tx = $F_{40\%}$

F45.tx = $F_{45\%}$
F50.tx = $F_{50\%}$
FFull.tx = F_{Full}
Ftarg.tx = F_{Target}
Fthresh.tx = $F_{Threshold}$
F0.1.tx = $F_{0.1}$
Fbar.tx = $\bar{F}$
SSBbar.tx = $S\bar{S}B$
SSBMSY.tx = SSB_{MSY}
SSBMSYproxy.tx = $SSB_{MSY}proxy$
SSBtarg.tx = SSB_{Target}
SSBthresh.tx = $SSB_{Threshold}$
BMSY.tx = B_{MSY}
Btarg.tx = B_{Target}
Bthresh.tx = $B_{Threshold}$
half.tx = $\frac{1}{2}$
EMSY.tx = E_{MSY}
EMSYproxy.tx = $E_{MSY}proxy$
E30.tx = $E_{30\%}$
E35.tx = $E_{35\%}$
E40.tx = $E_{40\%}$
E45.tx = $E_{45\%}$
E50.tx = $E_{50\%}$
EFull.tx = E_{Full}
Etarg.tx = E_{Target}
Ethresh.tx = $E_{Threshold}$
E0.1.tx = $E_{0.1}$
Ebar.tx = $\bar{E}$
alpha.tx = α
beta.tx = β
gamma.tx = γ
delta.tx = δ
lambda.tx = λ
rho.tx = ρ
mu.tx = μ
sigma.tx = σ
ge.tx = $\geq$
le.tx = $\leq$

Formatting text
To make *italics* use \\textit{words I want italicized}
To make **bold face** use \\textbf{words I want in bold face}
To do ***both*** use \\textbf{\\textit{words I want italicized and bolded}}

Section headers and miscellaneous stuff
sosHead.tx = **State of Stock:**
ProjHead.tx = **Projections:**
SpecComHead.tx = **Special Comments:**

- item.tx makes a bullet (requires the commands either "beginitem.tx" or "SpecComHead.tx", as well as enditem.tx)

RefHead.tx = **References:**
lbreak.tx =
(a line break)

Figure references
fig1.2.ref = (Figures 1-2); fig1.ref = Figure 1; fig2.ref = Figure 2; fig3.ref = Figure 3; fig4.ref = Figure 4; fig5.ref = Figure 5

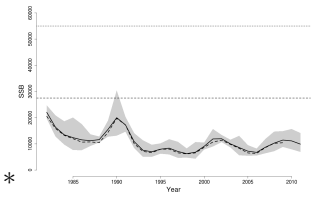

Figure 1: SSB plot

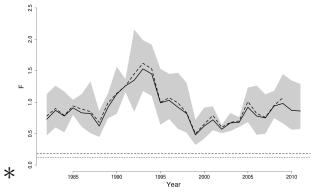

Figure 2: F plot

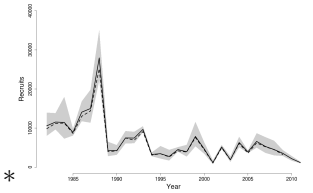

Figure 3: Recruit plot

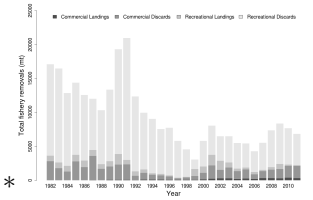

Figure 4: Removals plot

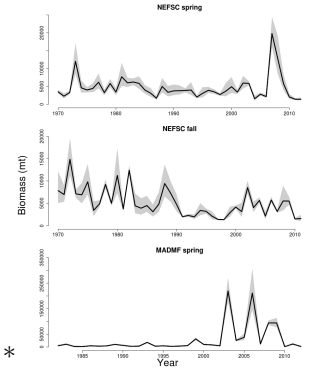

Figure 5: Survey plot