Stock Assessment Template in R Markdown and Git

Melissa Monk, NMFS SWFSC

Contents

1 Introduction

The stock assessment template described is based on the 2015 China rockfish assessment document. It is designed to aid in writing a stock assessment document, for models using Stock Synthesis and the R package r4ss. The template can currently handle up to three independent assessment models within the same assessment document, e.g., China rockfish was split into three assessments, 1) south of 40°10′ N. latitude, 2) 40°10′ N. latitude to the OR/WA border, and 3) Washington state.

The current repository contains .csv, SS, and r4ss files for an arbitrary stock assessment. Removing these files before you replace them with your own will result in runtime errors. It is recommended that you replace the files as you prepare them, knitting and committing the document each time you make a significant change.

Portions of the document need to be edited via .csv files, which will be discussed below. Please note, all of the instructions herein are geared towards PC users.

1.1 R Markdown and Git

The template was created for version control (Git) use in RStudio. R Markdown integrates plain text, LaTeX, HTML, and R code to create a reproducible document. The template is an R project (.Rproj), that keeps all of the files associated with the template organized. More on RStudio projects here. R Markdown's website is also full of resources.

Git is a distributed version control system, that houses files and repositories on a remote server. When you want to access these files/repositories, such as the assessment document project, you checkout a copy from the server, work on it locally (your computer), commit those changes, and then push your changes to the server. By committing a snapshot of a file or the repository, Git stores that version of the file in its memory. Each time you commit changes made to a file, they are stored in memory. This is a huge advantage if you then run into an error and need to revert to a previous version of your work. To revert back, click *Diff* on the Git tab (top right window of RStudio), and then *Revert*.

Git allows you to easily switch between computers and collaborate with the other assessment authors. This template is housed on GitHub, but there are dozens of options available to host your file. The Pro Git Book (online and free) is a great place to learn about Git.

1.2 Other Git notes

Collaborating with Git

There are two options for STAT teams to collaborate on the assessment document, described below. Remember, the repositories for the two options below are still public, and anyone can fork the repository to their own account. If you do not want the public to view your work, you will have to work through a private repository. If you have a university affiliation, you can request free private from GitHub. As a warning, on GitHub, all collaborators on a

private repository have owner-level permissions. Other Git hosts, such as BitBucket, offer free private repositories if this is a concern.

Personal repository

Under this model, one person will ultimately have complete access to the repository. The best recommendation is to make the STAT team lead, or whomever will be foremost responsible for document writing, the owner of the repository. This person will follow the below directions to fork the StockAssessment_template to their own account. Anyone who wishes to contribute to the repository will submit a pull request to the owner to incorporate changes. A more detailed description on contributing to open source projects can be found here.

Organization with teams repository

If you want more than one person to have read/write access to the repository, you would create an Organization with Teams. Remember, these repositories are still public, and anyone can fork the repository to their own account.

2 Getting started

Jennifer (Jenny) Bryan has a top-notch tutorial, 'Happy Git and GitHub for the useR' on the Git basics and connecting GitHub to RStudio. I HIGHLY recommend reading through her tutorials. I produced some documentation below (before discovering Jenny's work) and the combination of the two should get you ready to create R Markdown documents with GitHub and RStudio.

2.1 Necessary software

If you have not registered for free GitHub and Mendeley accounts, do so now. We'll cover more on Mendeley later on.

Make sure you have the latest versions of the following programs installed on your machine:

- Git (Contains instructions to install on Linux, Mac, and PC)
- R (latest version is usually a good idea)
- RStudio (Note: RStudio Preview will NOT work with the template until a bug is fixed)
- PC users: MikTeX or TeX Live (needed to render a PDF to LaTeX)
- Mac users: MacTeX (needed to render a PDF to LaTeX)
- Mendeley (optional, but useful for citation management)

Eric Anderson has directions for the Git and R setup for Mac users here.

R packages are automatically checked for in the `0-Run_r4ss_plots.R` script. You can add any other packages you want to load there. You may get an error and have to install `rmarkdown` and `knitr` manually.

2.2 Git, GitHub and SSH authentication

1. You need to tell git who you are. Navigate to Git Bash from your start menu, or in RStudio navigate to *Tools > Shell*.

2. In Git Bash, set your username and email (if you haven't done this previously). The username does not have to be your GitHub username, but the email address HAS to be the email you registered with GitHub, it can also be your first and last name. In the below commands, replace 'Melissa Monk' and my email with your name and email:

```
git config –global user.name 'Melissa Monk'
git config –global user.email 'melissa.monk@noaa.gov'
```

3. Because these commands return nothing, check to see that git registered your information by typing:

```
git config –global –list
```

4. If you want to avoid the prompt for your username and password every time you push and pull the repository, read the remainder of this section. Otherwise, skip it! Getting the SSH-authentication to work took me a few tries, so be patient if it doesn't work the first time. If the steps below don't work for you, check out Jenny Bryan's tutorial or this R-blogger post.

5. You can create SSH keys from either RStudio or the shell. First, check to see if you have existing keys.

   Open Git Bash and type:

```
ls -al  /.ssh
```

   If this command tells you .ssh does not exist, you don't have SSH keys. If the command returns a list of files including id_rsa.pub >and id_rsa, you have a key pair. You can skip to Step 9. Otherwise, set up the keys via Git Bash or RStudio.

2.2.1 Via RStudio (much easier)

6. a. In RStudio, in the menu navigate to Tools -> Global Options -> Git/SVN. Click *Create RSA Key*. This should generate the key. Click *View public key* and copy the contents. Go to step 9.

2.2.2 Via Git Bash

6. b. In Git Bash, type the following, replacing my email with the email associated with your GitHub account:

```
ssh-keygen -t rsa -b 4096 -C "melissa.monk@noaa.gov"
```

7. b. Accept the proposed key storage location by pressing Enter. You will be prompted to enter an option passphrase to protect the key. Either enter a passphrase, or leave this empty and press Enter (my preference).

8. b. Make sure the ssh-agent is enabled on your machine, by typing *eval "$(ssh-agent -s)"* into Git Bash. It should return and Agent pid.

Add your key to the agent:

```
ssh-add  /.ssh/id_rsa
```

If you chose to add a passphrase, you'll be prompted for it here. If you're successful, the shell will return something like:

```
Identity added: /Users/melissa.monk/.ssh/id_rsa (/Users/melissa.monk/.ssh/id_rsa)
```

Go to step 9.

---

9. Navigate to your profile on GitHub. From the View Profile screen select the *Edit Profile* button in the top right. Find *SSH keys* in the left-hand menu, and select *New SSH key*. Name the key something like 'Laptop key' (you'll need a separate key for each computer). Paste the View Public Key contents from RStudio into the *Key* box.

10. To check that the ssh-authentication works, type the following in Git Bash:

```
ssh -T git@github.com
```

If it works, you should get something like
Hi your_username! You've successfully authenticated, but GitHub does not provide shell access.

11. Remember to copy the SSH URL from the repository page to clone the repository.

Change remote.origin.url from HTTPS to HTTP (step may not be necessary)

12. If RStudio is still asking you for a username and password after you set up the ssh-authentication, type the following in Git Bash

```
git config remote.origin.url
```

```
git@github.com:your_username/your_project.git
```

2.3 Protecting confidential information

Keep in mind that if you're using a public repository, everything you push is visible to the world! Even if you realize that you've pushed confidential information and remove it with your next commit and push to GitHub, the confidential is **not actually deleted**!! The only way to remove the confidential information is to wipe the repository from GitHub, or go through a laborious process of removing your commit history.

If you have data files that you want to keep with the repository locally, you can add it to the .gitignore file. An example of this would be if you had a text file containing confidential information that you were then going to use within an R code chunk to create a non-confidential table or figure for your assessment.

If you're writing R code chunks that use ODBC connections, you can use the following script for the channel to prompt you for your password:

```
channel<-odbcConnect("database",
       uid="username",
       pwd=.rs.askForPassword("Enter password:"))
```

2.4 Update your fork from GitHub

If the StockAssessment_template repository on Melissa's GitHub is updated, you can update your fork. The directions below work, but they will leave you "1 commit head of the master."

1. Open your fork repository on GitHub.
2. Click on `Pull Requests`.
3. Click on `New Pull Request`. By default, GitHub will compare the original with your fork, and there shouldn't be nothing to compare if you didn't make any changes.
4. Change the `Base` drop down's so both point to your fork and then you'll get a prompt to `Compare across repos` (if no changes were made in the fork) or click Edit and switch the base manually. Now GitHub will compare your fork with the original, and you should see all the latest changes.
5. Click on `Create` to create a pull request for this comparison and assign a predictable name to your pull request (e.g., Update from original).
6. Click on Send pull request.
7. Scroll down and click `Merge pull request` and finally Confirm merge (If your fork didn't have any changes, you will be able to merge it automatically).

Directions I think will work better are here, but I haven't tried them yet!.

3 Using the template step-by-step

3.1 Forking the template

Once you have the necessary software installed and your machine talking to GitHub, you're ready to fork (make a copy of) the StockAssessment_template repository and begin your own assessment document.

1. Navigate to the StockAssessment_template on my GitHub page, melmonk.
2. In the top-right corner of the the StockAssessment_template repository page, click *Fork*.
    - You now have a copy of the StockAssessment_template repository in your own account.
3. Navigate back to your personal GitHub page.
    - StockAssessment_template should now appear in your list of repositories.
    - If you want to rename the repository, go to Settings (in the menus below the repository name in blue).
4. Click the link to the StockAssessment_template from your personal page.
    - On the right-hand side you'll see *HTTPS* clone URL, click the copy button, unless you're connecting to the GitHub via *SSH* (see the *Github and ssh-authentication section* and give it a quick read at this step).
    - This will allow you to pull the repository to your desktop.
5. Now open RStudio, navigate to *File -> New Project*.
6. Select *Version Control*, then *Git*.
7. Paste the URL you just copied on GitHub into the Repository URL box.
    - The *Project directory name:* will be autofilled, although you can change it.
    - You can also change the *Create project as a subdirectory of* box.
8. Now click *Create Project*.
    - Windows will pop-up asking for your GitHub username and then password (unless you have ssh-authentication).
    - The repository will now download.
9. Before we start changing files, let's make sure the template knits on your machine.
    - Navigate to ./RCode/0-Run_r4ss_plots.R and run through Section 1. You need to do this to produce the r4SS plots which I removed from the template repository to speed up download times and prevent compiling errors, e.g., the working directory is part of the r4SS code and is specific to your machine and inserting plots. Once you fork the template and have it on your machine, you can remove the '/r4SS/' from the .gitignore file.

- At the top of the RStudio scripts pane you should see a ball of yarn icon that says "knit." Click **knit to PDF** from the drop down menu.
- If there are no errors, RStudio will execute the R code chunks, and convert the document to a PDF and open a preview window with the knitted document.
- If it works, go to step 11. If it doesn't knit, we need to debug before you move on.

10. You're ready to write your assessment!

    - You can manipulate the files/folders on your desktop copy, e.g., csv files and figures, and these will show up as changed in the Git tab.
    - To keep the document looking neat, you can hide R code chunks using the arrow that appears between the R script line number and the start of the R code chunk, just click on it.

3.2 Species-specific set-up and running the template

Once you have forked the assessment template and created a new project in RStudio, you're ready to begin.

1. Navigate to the project folder on your computer and replace the contents of the SS folder with the contents of the run(s) you wish to use. If you only have one model, copy the SS files to the SS/Base_model1 folder.

2. If you don't currently have the project open in RStudio, do that now.

3. In RStudio, the project navigation pane is the bottom panel on the right, (contains tabs for Files, Plots, Packages, Help, Viewer). Navigate to and open ./Rcode/0-Run_r4ss_plots.R. You will need to change the working directory, number of models, and the file names of the control and data files. Do not source this script. Work through the end of Section 1 if you only have one model, and continue to Section 2 if you have multiple models. Section 3 is to write the r4ss output to a .csv file if you'd like. More details in SS and r4ss, section 3.4 below.

4. In RStudio, open the file: ./Rcode/Preamble.R. Change the all of the variables related to your assessment. By default, you can only work in the working directory containing the R project, so the working directly is automatically set by the session.

5. To change the document title, edit the title in the YAML (top of the .Rmd document)

6. To change the authors and their affiliations, open and edit the Titlepage.tex file

7. You are now ready to render the first version of the document. At the top of the RStudio scripts pane you should see a ball of yarn icon that says "knit." Click **knit to PDF** from the drop down menu. If there are no errors, RStudio will execute the R code chunks, and convert the document to a PDF and open a preview window with the knitted document.

8. If you receive an error, read the error message carefully and start debugging :) See a list of common errors in Common Errors, section 7.13.

9. Commit and push to GitHub often!

3.3 Saving, committing, and pushing your changes

1. Each time you make a change to a file in the repository, that file appears in the Git window

    - RStudio will automatically save the .Rmd file each time it's knitted (or save as usual; Ctrl+S)

2. To commit changes, click *Commit* in the Git window.

    - The differences between the last commit and current state of the file appear

3. Stage each file you want to commit by checking the boxes under *Stage* in the top-left window

    - Once checked, the status will change to "M," indicating the file has been modified
    - If you're deleting a file, the status will change to "D," indicating the file has been deleted
    - If you're adding a file, the status will change to "A," indicating the file has been added
    - Staged files are now ready to be committed

4. Before you can commit files, you have to write a short (at least one character) message in the top-right *Commit message* box

5. You can now click *Commit*

6. If you're finished with your session or just want to send your changes back to GitHub, now's the time to push them

7. Click the green Up arrow in the Git tab

    - This prompts you for your GitHub username and password and then proceeds to push your changes
    - See the section below on ssh-authentication if you want to avoid entering your username and password each time you push or pull the repository
    - When you're done for the day navigate to File -> Close project
    - To start working on the project again, open RStudio, navigate to File -> Open Project, find your file, open it, and pull (blue down arrow from the Git tab) any updates from the repository to your local machine.

Remember, you can always go back to an older version of the document you committed. Sometimes you'll get an error where it's easier to wipe the project from your desktop and pull a new version from GitHub than debug.

If you're working in a repository that has multiple owners, the most common error is that the other person has pushed changes to GitHub while you're still working. When you try and push your changes, you're going to get a Merge error. This can be a headache so try and avoid working simultaneously. I recommend using Google Docs to work on text, and having one person take responsibility of the document.

3.4 SS and r4ss

The StockAssessment_template repository contains both your SS and r4ss output for the model(s) to include in the document.

The SS folder contains four subfolders, one for each of up to three model outputs, and a linebreak_files folder that will contain the formatted SS input files, i.e., control, forecast, etc. When you're ready to replace the China rockfish model with your own, open the appropriate Base_model# folder and copy your SS model files here. Do this for the other Base_model# folders if you have multiple models. If you have only one model, you can empty the contents of the subsequent folders.

Note: you can rename "Base_model1" to something more intuitive. If you do, you will need to change the folder name in the "`Rcode/0-Run_r4ss_plots.R`" script.

Run the script `0-Run_r4ss_plots.R`. If you have only one model, you can skip the plot comparisons section. You can also add plot modifications in this script, e.g., extending plot margins for a plot group. The SS_output object(s) will be saved within the workspace `SS_output.RData` in the r4ss subfolder. Within the r4ss subfolder, model specific plots are saved in a `plots_mod#` folder (i.e., `plots_mod1`) and model comparison plots are saved in `plots_compre`. The r4ss subfolder also contains placeholder folders for forecast plots (`plots_forecast`), profile likelihood plots (`plots_profiles`), retrospective analysis plots (`plots_retros`), and sensitivity analysis plots (`plots_sensitivity`).

4 The document, section by section

All files used within the project need to be in the same parent folder. Be careful pasting text from Word, Google Docs, Textpad, etc. into R. Symbols such as a hyphens, quotes, apostrophes may not copy "correctly" and will either not show up when the document is knitted, or will be knitted as the incorrect symbol.

4.1 A list of folders and files in the Assessment template project

This list is ever-evolving, but will give you a sense of what's included in the repository, in order as they appear by default in the RStudio Files window (bottom right pane).

- **.gitignore** Contains the names of files and file types to ignore when pushing and pulling
- **.Rhistory** Contains the R session history, included in .gitignore
- **accessibility-meta.sty** Style to create an ADA accessible document - currently doesn't work, but someone can try and figure it out!
- **Assessment_template.* (.pdf, .Rmd, .proj, .tex)** The .Rmd file is the main document for the Assessment template. The .proj file is the RStudio file directory that houses the entire Assessment Template project, and is the file you open into RStudio. The TeX and PDF files are generated when you knit the document.
- **Assessment_template_files folder** This folder contains files automatically generated and are included in the gitignore file list
- **BibFile.bib** This is the bibliography file, which I generate in Mendeley
- **CJFAS.csl** This is the citation style file for the bibliography. This particular citation style file creates a references section following the format of the Canadian Journal of Fisheries and Aquatic Sciences
- **cover_photo.png** If you want to include a picture of your fish species on the cover, replace this picture with yours, and name it cover_photo.
- **Default_template_modified.tex** The default pandoc template had a text rendering issue for text with all capital letters when viewed in Adobe. This modified version of the template comments out the lines that caused the issue (hopefully).
- **Example_tables_figures folder** This folder contains examples of figures and tables and how to create them. Look here if there's a table or figure from the 2015 China rockfish assessment that you want to mimic, but can't figure out how.
- **Figures folder** This is where all of the user-created plots are stored (NOT r4ss generated plots).
- **header.tex** This file contains a list of the LaTeX packages to use in the document.
- **r4ss folder** This is where all of the r4ss generated plots are stored.
- **Rcode folder** This folder houses all of the larger R code scripts needed for the document.
- **ReadMe folder** The .Rmd file is the source for the ReadMe file for the Assessment Template. The TeX file is generated when the document is knit, and the PDF is for the user to read!
- **SS folder** This folder houses all of the Stock Synthesis files for each base model.
- **SS_file_appendices.Rmd** This is the child .Rmd file that creates the SS file Appendices. You shouldn't need to edit this file.

- **Test_figures_tables.Rmd** This .Rmd file is where you can test R code chunks figures and tables before adding them to the main Assessment_template .Rmd file, that you render to Test_figures_tables.pdf. It has to be in the main folder to work. This will save you a lot of time debugging!

- **Titlepage.tex** This LaTeX file generates the title page and should be edited to include the correct authors.

- **txt_files folder** This folder houses all of the text (.txt or .csv) files used to generate tables.

4.2 The .gitignore file

RStudio automatically creates the `.gitignore` file for a new project, and the Assessment_template document contains one in the root directory. This file lists all of the files/file types you want Git to ignore when you commit and push files. For instance, you don't need to push/pull the Assessment_template repository. If you add files to .gitignore, you have to delete and commit them as deleted before they will be ignored.

Below is the .gitignore file for Assessment_template. It ignores the Rproject user file, the Rhistory files, the files associated with knitting the document, as well as any files containing 'unnamed-chunk' (files created by LaTeX every time you knit). More info on ignoring files here.

```
===
.Rproj.user
===
.Rhistory
===
/Assessment_template.pdf
/Assessment_template.html
Assessment_template.tex
Test_figures_tables.tex
*.docx

**/list_of_dataframes.csv
**/mod_structure.csv
===  Latex temp files
*unnamed-chunk*
```

4.3 The YAML

The YAML contains all of the document front-matter and must be the first set of code in any .Rmd file. You cannot add comments to the YAML. Each YAML element used in the assessment template is described below. The document is currently only authored to be knit to a pdf file.

--- YAML begins with a line of three hyphens, no spaces
title: "" Provide the title of the document in quotes
author: '' Leave blank, authors are defined in Titlepage.tex
date:" Leave blank, date is defined in Titlepage.tex
output: Begin defining output variables
  pdf_document: Begin defining pdf output variables
    fig_caption: yes Should figures be captioned? yes or no
    highlight: haddock Color scheme for highlighting R code; options below
    includes: Define external documents to include
      before_body: Titlepage.tex Include Titlepage.tex (title page first)
      in_header: header.tex Include header.tex (all necessary LaTeX packages)
    keep_tex: yes Keep intermediate TeX output? yes or no
    latex_engine: xelatex Define the latex engine (sometimes matters) template: Default_template_modified.tex Template comments out lmodern package number_sections: yes Number the document sections? yes or no
    toc: yes Include a table of contents? yes or no
    toc_depth: 4 Number of subheadings to include in the table of contents
  html_document: Begin defining HTML output variables
    toc: yes Include a table of contents? yes or no
  word_document: default Begin defining Word document output variables
documentclass: article LaTeX document class
fontsize: 12pt Default font size
geometry: margin=1in Page margin size
csl: CJAFS.csl Bibliography style
bibliography: BibFile.bib Bibliography file name
link-citations: yes Adds hyperlinks to citations --- YAML ends with a line of three hyphens, no spaces

Notes:

- Keeping the TeX file can help with debugging.
  - The line number for a given error can refer to .Rmd, .tex file, or be completely random
- Options for the R script highlighting include: default, tango, pygments, kate, monochrome, espresso, zenburn, haddock and textmate. Play around with them to see which color you like best.
- The HTML and Word document settings are currently dummy settings just so you can knit to these. Future work can be done to knit the document to these formats.

- I've included the modified default pandoc template that comments out the lmodern package and other associated packages. These packages cause some strange font rendering of acronyms (or other words in all capital letters) when viewed in Adobe products.

4.4 The Meat and Bones

The Assessment Template contains most (if not all) of the headers in the Terms of Reference. I have left bits and pieces in the document that likely apply to all assessments, e.g., the citation for the Hamel prior in the Priors section. The following sections will provide details on each section of the template.

4.5 Executive Summary

The Executive Summary is basically written and calls the r4SS output and csv files you provide. You will have to edit the text (.csv) files, such as the catch histories and landings by fleet, and decision tables. For all of these, you need to edit the text file and possibly the R code (in Rcode/R_exec_summary_figs_tables.R) depending on the table/figure structures.

The following Executive Summary tables are associated with .csv files, which will need to be replaced with your data. This may also require editing the column alignment options if you have a different number of columns than in the default template. This is where you'll want to test your tables and figures in the Test_figures_tables.Rmd file, by copying the code to the main documents.

- Table a. Recent landings by fleet.
  - `Exec_catch_summary.csv`
- Table n. Recent trend in total catch... relative to management guidelines.
  - `Exec_mngmt-performance.csv`
- Tables p-r. Decision table(s)
  - `DecisionTable_mod1.csv` (and if needed `DecisionTable_mod2.csv` and `DecisionTable_mod3.csv`)
- Table s. Base case results summary. Note: This table is a mix of a .csv file and r4SS output.
  - `Exec_basemodel_summary.csv` and r4ss output. the .csv file contains the harvest guidelines. All other values are pulled from the r4ss output.

4.6 Appendices: SS input files

The script in `Run_SS_input_linebreaks.R` contains the function and then commands to edit the SS input files for printing in Appendices A-D, where Appendix A: SS data file, Appendix B: SS control file, Appendix C: SS starter file, and Appendix D: SS forecast file. The script for the appendices is contained in a child .Rmd file, `Appendices.Rmd`. The Appendices A-D are appended to the document via the following R code chunk:

```
<!-- ```{r child="Appendices.rmd"}  # '<!--' opens an HTML comment
         ``` -->                    # '-->'  ends an HTML comment
```

The SS appendix files are currently commented out in the Assessment_template.Rmd file, as well as in the above box, using HTML comment syntax. They are commented to save on runtime and reduce the document size while editing. Remove the HTML comment syntax to include the SS appendices.

## 4.7 References section

If you have a citation manager you're committed to that will create a .bib file, you can skip this section.

**Create the References section with Mendeley**

If you have not already downloaded Mendeley and created a free account, do so now.

R Markdown can read a number of bibliography styles, see Bibliographies section. I'm providing directions for creating a .bib file in Mendeley. Using Mendeley is not required, but it's free and a great citation manager. The biggest downfall with Mendeley is that you cannot include italics in title names. I provide a (somewhat clunky) work-around below. One other caveat of R Markdown is that the References section is automatically placed at the end of the document, which means after the table, figures, and appendices.

1. Download Mendeley Desktop here and also install the Web Importer (for use with Google Scholar, or a journal's website). If you don't already have a free account, create one now.
2. Create a group on Mendeley for collaboration, e.g., China Rockfish Assessment 2015, if you want to collaborate on the references section, i.e., allow your co-authors to add citations.
    - *File > New Group*
    - The setup should be self-explanatory.
3. Either input a reference manually or import it to Mendeley from Google Scholar.
    - Ensure all the pieces, e.g., page numbers, are imported. If not, enter them manually.
```

- Check this for each reference, or else you will be re-doing the search for all of your literature at the last minute.

4. Add references to the group folder in Mendeley Desktop.

    - The reference will also be added to your main library.

5. Make sure the Citation Key field is not blank and matches the key you want to reference it as in the R Markdown document.

    - The citation key is used to cite documents in the assessment.
    - Best practice: use the first author's last name and the year of publication, e.g., Monk2015.

6. To update the .bib file, Go to Documents tab in the group folder, select all, and go to *File > Export*.

    - Export the files as a .bib file.
    - Save and overwrite the BibFile.bib file in your version-controlled working folder.

7. Make sure you include the new .bib file when you push your changes to GitHub.

To add a reference to the document type [@CitationKey], which will include the reference in parentheses. To include the reference as the year only, type [-@CitationKey]. If you include a year only citation, remember to manually type in the author part of the citation.

The 2015 China rockfish assessment bibliography collection is public in Mendeley. You can find it in Mendeley by searching for the group "China Rockfish Assessment 2015." Once you join the group, you can create a new group and drag all of the references you want from the China Rockfish assessment to your new group.

Workaround for italicizing scientific names (knit to pdf) The .bib file is rendered via LaTeX, so you can enclose a scientific name with `\emph{}` in the Mendeley citation, e.g., `\emph{Sebastes nebulosus}`, to produce *Sebastes nebulosus*. Once you create the .bib file, the scientific names are converted to `$\backslash$emph\{Sebastes nebulosus\}`. You will have to manually go into the .bib file and edit the scientific names so they once again look like `\emph{Sebastes nebulosus}`, which can be sped up by by using Find and Replace (Ctrl+F). Wait to do this until you have the final version of your .bib file, or you'll have to do this every time you edit and overwrite the .bib file. You can view and edit the .bib file in RStudio.

4.8 Before you publish

There are a few LaTeX packages turned on in the default template on GitHub. These are controlled in the header.tex file. To turn a feature off, comment out the package with a %.

1. Package lineno allows for line numbers throughout the document. This was helpful for reviewers during the STAR panel and editing post-STAR panel.
2. Package showlabels prints the section, figure and table labels. This is helpful if you're trying to remember which figure/table you're cross-referencing in the text.
3. Package draftwatermark places a water mark on the pages.

5 Creating Tables

Tables are generated within R code chunks using the R package xtable. The xtable vignettes are extremely useful. I recommend starting there if you have a question.

Tables in the Assessment_template are generated from R output, including r4ss, or a .txt/.csv file (located in the `txt_files` subfolder). You'll create tables inside R code chunks using the xtable package. I highly recommend using the Test_figures_tables.Rmd document to test run any new tables you want to add to the document.

Create a table using these general steps:

1. You can either read in a .csv file or manipulate data from the r4ss output. Either way, the dataframe should resemble the same format (row and columns) that you want in the table. Start an R code chunk and read in the data.
2. Edit the column names. R will likely remove spaces if done before data manipulation is complete.
3. Create the table using the xtable command.
    - `xtable(dataframe_name, caption=c("Table caption"), label='tab:table_label)`
    - The label allows you reference the table in the document, ex. `\ref{tab:table_label}` and will automatically number each table.
    - I find it good practice to precede a table label with 'tab' so you can easily recognize the reference (e.g., tab:Exec_catch)
4. Adjust the column alignment
    - `align(table_name) = c('l','l','>{\\centering}p{1in}')`
    - You must have one dummy column alignment parameter for row.names, if you are not printing row names

- Common alignment options
  - `'l'`, `'c'`, or `'r'` for left, center or right alignment (these options do not adjust column width)
  - `'>{\\centering}p{1in}'` for center alignment where you assign the column width, 1 inch in this case
  - `'>{\\raggedright}p{1in}'` for left alignment where you assign the column width, 1 inch in this case
  - `'>{\\raggedleft}p{1in}'` for right alignment where you assign the column width, 1 inch in this case

5. Print the table

- print(table_name)
- Common print options include
  - include.rownames=FALSE (don't include rownames as a column)
  - caption.placement = "top" (place caption above the table)
  - sanitize.text.function = function(x){x} (you'll see this where I include LaTeX syntax in the table - however, including this when not needed produces an error)
  - hline.after = c(-1,0,6,12) (include extra horizonal lines in the table after the line specified)
  - scalebox = .6 (scales the table if it doesn't fit on the page, value of .6 is 60% original size)
  - floating.environment="sidewaystable" (creates a table in landscape mode, but you cannot move the page number to another side)
  - tabular.environment="longtable" (creates a table spanning multiple pages)
  - add.to.row=addtorow (include if you're adding rows to the top of the table, see Spanning multiple columns below)
  - size = "small" (or any other recognized default LaTeX font size)

To create a table that is both in landscape mode and spans multiple pages, you create the table as a longtable, and rotate the page. See the model parameters table for an example.

5.1 Including special characters in tables

Adding special characters or bold/italic font to cells in a table varies slightly depending on if the table is read in from a text file or if the table is created directly from R output.

5.1.1 Table content from a text file (.txt or .csv)

You will have to manually add in the LaTeX syntax for bold or italic font into the text file. For example, if you want the year 2005 bolded the cell will read `\\textbf{2005}`. Or for example you want to bold Total Catch OY, you type `\\textbf{Total Catch OY}` into the cell. When

you create the table with xtables, you must include the option, `sanitize.text.function = function(x){x}`. Example taken from Table n of the 2015 China rockfish assessment, created using the ./txt_files/Exec_mngmt_performance.csv file.

5.1.2 Table content from R code chunks

LaTeX syntax can be included in R code or R code chunks. You must precede each LaTeX command with two backslashes instead of one. The following will bold and italicize the text: `\\textbf{\\textit{Reference points based on SPR proxy for MSY}}`. For an example, see the Reference Points table code in ./RCode/R_exec_summary_figs_tables.R file, which is Executive Summary Table m.

If you include a percent sign % in the R code or R code chunk, it needs to be preceded by two backslashes, `\\%`. However, this is not necessary in the main text of the R Markdown document.

You can use inline math mode just as you would in the main text. For example, `$SPR_{B40\\%}$` produces $SPR_{B40\%}$ within an R code chunk. Note that if you are using a % sign in math mode in the main text, which is LaTeX, it must be preceded with one backslash.

5.2 Spanning multiple columns

Sometimes you may want a column header to to span multiple columns, equivalent to the merge columns function in Excel. This can be done within xtable and may take some playing around with. See Executive Summary Table p, the Decision Table, for an example. The "States of Nature" column header spans multiple columns. The following code is specific to this table, but I'll explain each line, numbered in the box below. For this to be included in the table, you will need to include `add.to.row=addtorow` in the print() table command.

```
1. addtorow <- list()
2. addtorow$pos <- list()
3. addtorow$pos[[1]] <- -1
   addtorow$pos[[2]] <- -1
4. addtorow$command <-
         c('\\multicolumn{3}{c}{}
          & \\multicolumn{2}{c}{}
          & \\multicolumn{2}{c}{\\textbf{States of nature}}
          & \\multicolumn{2}{c}{} \\\\\n',

          '\\multicolumn{3}{c}{}
         & \\multicolumn{2}{c}{Low M 0.05}
         & \\multicolumn{2}{c}{Base M 0.07}
         & \\multicolumn{2}{c}{High M 0.09} \\\\\n')
```

1. The `addtorow <- list()` creates the addtorow variable as a list, which should have two components, pos and command.
2. The `addtorow$pos <- list()` turns the pos component into a list which will contain the positions of the the rows you're adding
3. The `addtorow$pos[[1]] <- -1` and `addtorow$pos[[2]] <- -1` set both the rows we're adding to appear before the column names of the table.
4. The `addtorow$command` creates the list of column headers and this is where we can insert `\\multicolumn`. All of the information for a single row is in single quotes ending with `\\\\\n`. We're adding two rows to the dataset, so we have two sets of row commands. There are five backslashes preceding the 'n' because backslashes get lost in translation, just like we're using two backslashes again to call multicolumn.

The first thing to note is the decision table has nine columns. The value in the first set of curly brackets after multicolumn (3 in this first row, `\\multicolumn{3}`) gives the number of columns to span and must add up to the number of columns in the table. So here, 3+2+2+2 = 9 (you don't need to worry about an extra column for row names). The second set of curly brackets gives the centering for the text, here all of which are 'c' for center. The text for the multicolumn goes in the third set of curly brackets, and can also be left blank.

5.3 Horizontal and vertical lines

You can include extra horizontal lines between any two rows of a table by including hline.after=c(), wherein you list the table rows after which you want to insert a horizontal line. Row number -1 will place a horizontal line above the header, 0 will print a line below the header and any other number, say 10, will print a horizontal line after row 10. See Executive Summary Table p, the Decision Table, for an example.

Vertical lines are added in the alignment command for xtable. Vertical lines are inserted as vertical bars, |, and in the example below, to the right of a column. This example is also taken from the Executive Summary Decision table, Table p.

```
align(decision_mod2.table) = c('l','l|','c','c|',
                               '>{\\centering}p{.7in}', 'c|',
                               '>{\\centering}p{.7in}','c|',
                               '>{\\centering}p{.7in}','c')
```

5.4 Shading table cells

TOADS

6 Creating/Inserting Figures

Figure files (pre-existing image files) are incorporated via R Markdown syntax (e.g., r4ss figures) or R code chunks (e.g., create a figure within the template using your favorite graphics package). You may not introduce linebreaks in the R Markdown figure syntax.

The following code will add figure.png from the plots folder to the document.

```
![Figure caption. \label{fig:figure_label}](plots/figure.png)
```

For the standard r4ss plots, remember to change the plot directory, e.g., 'r4ss/plots_mod3' to access plots for model #3. I like to precede the figure label with 'fig:' in case there are figures and plots referencing the same data. I also find it helpful to label r4ss figures with the model number, e.g., Mod3_, followed by the figure name (see below for an example).

```
![Figure caption. \label{fig:Mod3_comp_lendat_flt5mkt2}](r4ss/plots_mod3/
comp_lendat_flt5mkt2.png)
```

To create a figure within R code chunks, I find it helpful to actually use two R code chunks, one for data manipulation and a second to plot the figure. This is because you want to set include=FALSE for the chunk manipulating the data, and include=TRUE (default value) for the chunk plotting the figure, and also so we can include the figure caption. An example is below. Notice too, you need to only change the R code chunk options if they differ from the document's global options (described in the R Code Chunks section).

````
```{r, include=FALSE}    #include is the only option we need to change
     CA_rec_remov = CA_rec_removal[,1:5]
     colnames(CA_rec_remov) = c('Year','South PC','South PR',
                                'North PC','North PR')
     CA_rec_remov1 = melt(CA_rec_remov,id='Year')
     colnames(CA_rec_remov1) = c('Year','Fleet','Removals')
```
```{r,fig.cap="Removals (mt) from the California recreational party/charter
   and private sectors, north and south of $40^\\circ 10^\\prime$.
\\label{fig:CA_rec_removal}"}
      ggplot(CA_rec_remov1, aes(x=Year, y=Removals,fill=Fleet)) +
      geom_area(position='stack') +
      scale_fill_manual(values =
          c( 'lightsteelblue3','coral',"aquamarine2","mediumpurple")) +
      scale_x_continuous(breaks=seq(1928,2014,10)) +
      ylab("Removals (mt)")
```
````

7 General topics

7.1 Syntax (R Markdown and LaTeX)

The syntax used throughout will depend on the output file type, PDF or HTML. If you only want to knit to a pdf, you can use either the R Markdown or LaTeX syntax in the examples below. However, if you want to be able to knit to HTML you're better off using the R Markdown syntax, since HTML will not render LaTeX. See the section on pandoc markdown for more on syntax.

Knitting to a Word document is unstable and will result in strange/missing output.

7.2 Paragraphs

To separate paragraphs, you must leave a blank line between paragraphs. You can create hard line breaks (without a blank line between paragraphs) by either leaving two or more spaces at the end of the last paragraph or a backslash between the last paragraph and the new line.

The width of the text will depend on how wide your RStudio window is, unless you choose to use carriage returns to keep the text as a maximum width in the viewing window. R Markdown ignores carriage returns, and will not start a new paragraph without a blank line.

7.3 Spell checking

Do not completely rely on RStudio's spell checker, which doesn't even include RStudio as a word. Use the spell checker, but edit the document yourself as well. There is not autocorrect function, as in Word.

7.4 LaTeX

If you're looking for a specific LaTeX symbol or command, it's easiest just to Google it.

You can use LaTeX commands throughout the document, such as '\newpage' and '\FloatBarrier,' which are helpful for inserting a page break and keeping figures from being rearranged. I've also found that it's good to insert the float barrier command after every three figures or so, to prevent a runtime error.

7.5 Fonts and font size

You cannot control both the font and the font size in R Markdown. The font size specified in the YAML (see YAML section) is set at 12pt.

You can specify italics and bold fonts using either LaTeX or R Markdown syntax. If you only want to knit to a PDF, my preference is to stick to the LaTeX syntax, as you'll see throughout the document, but that's personal opinion.

```
*Italics*
R Markdown: *word* or  _word_
LaTeX: \emph{word}


**Bold**
R Markdown: **word** or __word__
LaTeX:  \textbf{word}
```

7.6 Section headers

Numbered headers in R Markdown are as follows:

```
#Header 1
##Header 2
###Header 3
```

To create a header without a number, e.g., for the Executive Summary sections, follow the header with {-}, ex. `#Header{-}`.

Un-numbered subsection headers that you don't want to appear in the table of contents can be created by starting a section with a bold or italics header.

7.7 Numbering (pages, tables, figures)

The table of contents will automatically be numbered using lower case roman numerals. Arabic numbering of pages begins with the Executive Summary. Tables and figures in the Executive summary are lowercase alphabetic. This is defined using the following LaTeX script:

`\pagenumbering{arabic}` Defines page numbering as Arabic numbers
`\setcounter{page}{1}` Sets the first page number to 1
`\renewcommand{\thefigure}{\alph{figure}}` Defines figure labels as alphabetic
`\renewcommand{\thetable}{\alph{table}}` Defines table labels as alphabetic

Immediately preceding the Introduction section, the labels for figures and tables are reset:

```
\renewcommand{\thefigure}{\arabic{figure}} Defines figure labels as Arabic numbers
\renewcommand{\thetable}{\arabic{table}} Defines table labels as Arabic numbers
\setcounter{figure}{0} Set figure number to 0; first figure will be Figure 1
\setcounter{table}{0} Set figure number to 0; first table will be Table 1
```

We do not need to reset the page numbers because they continue from the Executive Summary. The other place you need to reset page and figure numbers is with each appendix:

```
#Appendix A. Appendix Title{-} Appendix header without a number
\label{sec:AppendixA} Creates a section label to reference it throughout the document
\renewcommand{\thepage}{A-\arabic{page}} Precede page numbers with A-, e.g., A-1
\renewcommand{\thefigure}{A\arabic{figure}} Add A to figures, e.g., Figure A1
\renewcommand{\thetable}{A\arabic{table}} Add A to tables, e.g., Table A1
\setcounter{page}{1} Set the first page number to 1
\setcounter{figure}{0} Set the figure number to 0; first figure will be Figure A1
\setcounter{table}{0} Set the figure number to 0; first table will be Table A1
```

The appendices for the SS code do not include commands to reset figure and table numbering, as there are no figures or tables in these sections.

7.8 Lists

I prefer using LaTeX for lists, but R Markdown also has its own syntax for lists. A LaTeX example is below:

```
Ordered lists:
\begin{enumerate}
  \item List item No. 1 in the list
  \item List item No. 2 in the list, etc.
\end{enumerate}

Unordered lists:
\begin{itemize}
  \item First item
  \item Next item
\end{itemize}
```

7.9 Equations and math mode

Wikipedia and the internet can help you with mathematical symbols as needed.

To get subscripts, superscripts and degree symbols for latitude/longitude, the easiest way is through math mode. In LaTeX, dollar signs indicate inline math mode, the underscore

produces a subscript, and a caret produces a superscript. You'll see throughout the document that biological reference points are typed in math mode, `$SPR_{50\%}$` is $SPR_{50\%}$. As noted earlier in the document, you must precede a percent sign with a backslash when typing in math mode.

For latitude or longitude, follow the format `$40^\circ 10^\prime$`, which produces $40^\circ 10^\prime$. As discussed above in the Tables section, when you use math mode within R code to create a table, two backslashes are necessary, and the latitude or longitude is written as `$40^\\circ 10^\\prime$`.

7.10 R code chunks

R code is written in the .Rmd file as R code chunks. R code can be rendered or displayed for illustration and the R Markdown Reference Card contains almost everything you need. Here is the simplest R code chunk, which also prints the results:

````
```r
    1+1
```

```
## [1] 2
```
````

The preamble of Assessment_template.Rmd sets the global options for R code chunks

````
<!-- #commented out these lines for presentation puposes
` ```{r global_options, include=FALSE} #sets global options
  knitr::opts_chunk$set(echo=FALSE, warning=FALSE, message=FALSE) #options
```  -->
````

These options tell R Markdown not to print the R code (echo=FALSE), and to ignore warnings and messages from R (message=FALSE and warning=FALSE). The additional code chunk option of results='asis' appears in R code chunks for tables to prevent unwanted reformatting. See the R Markdown Reference Card for more details.
````

7.11 Commenting

Comments within the main R Markdown document body are included via HTML syntax, <! − −Add your comment here − −! >

Inline R code within an HTML comment is still rendered. However, you can comment out R code chunks.

```
If you type:
<!--R code within an HTML comment that is NOT commented will
be evaluated, 'r 1+1'. -->

This is the result:
<!-- R code within an HTML comment that is NOT commented will
be evaluated, 2. -->

If you type:
<!-- R code within an HTML comment that IS commented out
produces a blank space, 'r #1+1'. -->

This is the result:
<!-- R code within an HTML comment that IS commented out
produces a blank space, . -->
```

7.12 ADA compliance

In 1998 Congress passed the Section 508 Amendment to the Rehabilitation Act of 1973, requiring that all federally-funded documents are accessible to those with disabilities. Andy Clifton has the AccessibleMetaClass on his GitHub page. The accessibility-meta.sty is part of the Assessment_template.

Alternative text is a description of an equation, link, or figure. These are pop-ups in a PDF viewer, i.e., hover your mouse over the picture and the pop-up will appear. These can be added to the source document using **pdftooptip** from the `pdfcomment` package. For example, the cover photo is now included using `\pdftooltip{\includegraphics{cover_photo}}{This is a fish.}`.

To create the ADA compliant PDF, you'll need to take the final TeX document and compile it outside of R Markdown. In the TeX preamble add the following package.`\RequirePackage[l2tabu, orthodox]{nag}`. currently, there is an error that will require a lot of time to debug (!TeX capacity exceed, sorry...). However, the document can be tagged in Adobe Pro in a matter of seconds.

7.13 Common Errors

- You have a version of the PDF assessment document open (not just the preview version)
- You forgot a backslash before an underscore, or another reserved LaTeX character
- Your table alignment is off, or xtable just doesn't like your alignment parameters. Simplify the alignment and add in alignment justifications incrementally
- You have too many tables or figures in a row without a float barrier. You need to insert '\Floatbarrier' between table or figure, Error:"LaTeX Error: too many unprocessed floats."
- You have a space in a file name somewhere. LaTeX does not like to read file names with spaces.
- You have an illegal space or carriage return, most likely in the R Markdown image syntax
- You don't have a blank line after an R code chunk. *Sometimes*, you'll get an error if you end an R code chunk and on the next line include a float barrier or other LaTeX syntax.