

Commission for the Conservation of
Southern Bluefin Tuna



みなみまぐろ保存委員会

Report of the Operating Model Specification and Software Upgrade Workshop

**20 – 22 November 2023
Tokyo, Japan**

Report of the Operating Model Specification and Software Upgrade Workshop

20 – 21 November 2023

Tokyo, Japan

Opening

1. The Chair of the Operating Model Specification and Software Upgrade Workshop (OM Coding WS), Dr. Ana Parma, opened the meeting and welcomed participants (**Attachment 1**).
2. The draft agenda was discussed and amended, and the adopted agenda is shown in **Attachment 2**.
3. The list of documents for the meeting is shown in **Attachment 3**.

Agenda Item 1. Introduction

4. The consultant for the OM coding project, Dr. Darcy Webber, reviewed the goals of the project and presented a brief progress report. The introduction, as well as the different agenda items were covered using five presentations which are included as attachments 4 to 8.

Agenda Item 2. Overview of model code

5. Dr. Webber presented an overview of the model code structure (see Attachment 5) including the series of R functions and the C++ code that evaluates the objective function. In the latter, he explicitly retained ADMB code alongside the new TMB code and this helped show the differences between the two languages. He guided participants through the steps involved in connecting to the SBT Github repositories where the new software is hosted.
6. At present, the `sbt` package is hosted in Dr. Webber private Github repository because the assessment input data files are confidential, meaning that the repository cannot be made public, but maintaining a private institutional Github repository that uses Github actions (for building the website and running model checks) is costly. Alternatively, the input data could be stored separately (in the CCSBT private site) and the package could be made public.
7. Participants installed the new `sbt` R package developed under the project as well as an `sbt_models` suite of helper functions for running examples, plotting, and analysis of model outputs. All participants were able to download, build, and execute a stock assessment using the new `sbt` package and generate plots using `sbt_models`.
8. In general, the new TMB version of the assessment (i.e., `sbt`) code matched the current ADMB version code except for obvious differences in syntax between the programming languages. Certain computations related to fixed data and functions were also moved outside the main C++ code into simpler R scripts for improved efficiency and transparency. This included preprocessing of the data (i.e., as done in

the PRELIMINARY_CALCS section of the ADMB code) and streamlining all the data into simple ASCII files.

9. Dr. Webber presented comparisons between the ADMB and TMB implementations of the `sbt` assessment model when each was provided the same input parameters and data. Model predictions agreed to within acceptable levels of machine precision (e.g., $1.e-8$) for all outputs demonstrating that the population dynamics, likelihoods, and inferred quantities were nearly exactly replicated in the new TMB software.

Agenda Item 3. Running models

10. Following the instructions provided in Attachment 6, participants installed all the current software and were able to build the package ("[sbt](#)") and run the current version of the model and data using scripts provided in "[sbt_models](#)". This meant that all the software worked seamlessly on Linux, MacOS, and Windows operating systems.
11. Dr. Webber showed that the new model matched the existing model precisely. He demonstrated a variety of features that simplified the data input and plotting and diagnostic outputs.
12. He first ran a single operating model (OM) grid cell to demonstrate the workflow used to generate an assessment result from raw data input (i.e., in .csv format), converting to formats required by the `sbt` model, compiling the TMB code, executing an assessment model fit, and plotting the resulting estimates.
13. Code examples were also provided to complete the above steps for the entire OM grid (Attachment 6). This includes creating a grid, extracting outputs, plotting inputs/outputs, and comparisons with the ADMB model.

Agenda Item 4. Bayesian inference

14. The new `sbt` package has substantially greater power to assess uncertainty in the stock assessment via Bayesian methods. Dr. Webber presented an example Bayesian implementation via the R `tmbstan` package applied to a single OM grid cell as well as another example involving four OM grid cells (see Attachment 7).
15. This includes extracting and plotting MCMC outputs, MCMC diagnostics, reduced grid structure for MCMC runs, comparison of OM uncertainty obtained via MCMC (including within-cell uncertainty) vs standard grid, and model comparison using the leave-one-out information criterion.
16. Dr. Webber also demonstrated how `sbt` implements the leave-one-out information criterion (LOO IC) for assessing the relative credibility and possibly future weighting of alternative operating models.
17. Overall, the group was enthusiastic about the potentially large improvements to future `sbt` assessments made possible by the new `sbt` package and especially how much progress was made in such short time. The group looks forward to seeing the next phase of the `sbt` package in the June 2024 OMMP meeting.

Agenda Item 5. Next steps

18. The group discussed modifications (described below) to the assessment modelling approach that could improve several aspects including computational efficiency, bias reduction, and representation of uncertainty. Some of the proposed changes would be developed over the next few years but would not be incorporated into the formal stock assessment and operating model conditioning until after the next stock assessment to be conducted in 2026.
19. ***Fisheries and selectivity***—The group discussed alternative methods for incorporating LL3 and LL4 fisheries into the assessment model. These fisheries each had short periods of substantial catches in the distant past and several years of very small catches and low-quality length-composition data more recently. Fitting the length composition data for these fisheries requires estimating time-varying selectivity involving hundreds of extra parameters in the assessment model even though there is little information gained. In addition, these fisheries are not carried into the projection code and so there is no need to estimate a selectivity for them. The proposal for future model changes involved (i) converting the length compositions to age composition via cohort slicing; (ii) subtracting the calculated catches-at-age from the SBT abundance-at-age in each year; and (iii) removing the length compositions from the model likelihood (fitting procedure). This approach would adequately account for these fisheries' removals while simplifying the model parameter estimation without loss of information about SBT abundance.
20. ***Time-varying selectivity for remaining fisheries***—the group discussed a new Gaussian Markov Random Field (GMRF) approach to modelling time-varying selectivity for LL1, LL2, Indonesian, and Australian fisheries. The GMRF approach would provide a more formal statistical implementation of the current piece-wise time-varying approach to fishery selectivity.
21. ***Length-/age-composition likelihoods***—the group discussed alternative likelihood formulations for composition data including the Dirichlet-multinomial to provide a more objective self-weighting approach to fitting this data.
22. Australia presented paper OM Coding WS/2311/04, which described a possible length-/age-structured approach to the next generation SBT assessment. As noted above, such changes would be developed over several years and would not be implemented in the next stock assessment cycle but could be available for the following one. Even though a length-/age-based model would require adding a new length dimension to the model, several computational processes within the model would become more efficient, simpler, and probably more accurate since some processes in the population and fishery dynamics are length-based. For example, juvenile mortality, selectivity, reproductive output, and recapture of tagged individuals are influenced by individual body length, but the current approach treats these mainly as age-based or as functions of an overall mean length.
23. A joint length-/age-structured approach would improve the representation of parent ages in the CKMR POP and HSP likelihoods. In particular, the current model assumes that the age of a parent is known exactly based on its length even though an

SBT of a given length could be one of many possible ages. A length dimension would accommodate this uncertainty as well as possibly reduce associated bias.

24. For future structural model changes that track length and age more completely, it will be important to evaluate details on things like the number and size of length bins for both computation and population-level applicability.
25. The group noted that a good practice to do a release as each assessment is completed and adopt semantic version numbering system (e.g., <https://semver.org>). This should also be used to lock the model and data for that assessment so it can be recovered in the future. This may also include saving the computer environment that was used at the time (e.g., using Dockers).
26. For future model configurations, the group discussed how the CPUE predictions from the model should be treated based on length compositions that are weighted by density rather than catch (as presently done). Furthermore, the size composition data for the LL1 fleet includes size composition data corresponding to the catch from other members (e.g., Korea, New Zealand, and Australia) which should be excluded for CPUE predictions. This is because the CPUE data are based solely on the Japanese fleet at present.
27. A number of issues encountered during the model recoding were considered by the workshop and either resolved or assigned to participants for implementation of needed code changes. The group discussed next steps and timeline proposals.

Ideally the following changes would be implemented before the June 2024 meeting so that they can be evaluated by the OMMP working group:

- a) Lump the LL3 and LL4 fisheries and cohort slice and treat as removals
- b) Specify the LL1, LL2, Australian and, Indonesian selectivity using GMRF
- c) Review this years sensitivities and robustness tests and make sure all the code to do these is available
- d) Can filter out some of the POPs in `get_data` that result in likelihood values that are not used in the estimation
- e) Name the grid runs in `run_grid`
- f) Implement grid sampling in the R code
- g) Re-code tag likelihoods to remove the H^* parameters (harvest rate for mixing periods) and add the output for the PSIS-LOO diagnostic
- h) Implement the Dirichlet-multinomial likelihood for composition data
- i) Code prior distributions in short-hand (following R format; e.g., `dnorm()`)
- j) Incorporate the age-uncertainty for the adult part of the POP calculations (the possible ages given length)
- k) Update website to improve documentation (e.g., add vignette on “how to run the grid”).
- l) Evaluate if other “Stan” R packages (e.g., `adnuts`) can be used to help evaluate model runs.

Other tasks that could be completed at the June 2024 meeting include:

- m) Review harvest rate function and determine if a penalty is required to keep it below 0.9 (currently there is no penalty in the sbt model)
- n) Categorize what we want to add to REPORT and ADREPORT in the TMB code

- o) Implement “one-step ahead residuals” diagnostics for judging fits to composition data
- p) Evaluate how the grid should be modified in light of new MCMC capabilities

Tasks that could be done after the June 2024 meeting include:

- q) Projection model developments: two options were discussed, an interim option that requires the TMB code to output the same variables that the ADMB conditioning code passes to the projection code, so that the old projection code can be run (with inputs in the same format) or a final option where projections are implemented within the “simulate” blocks of the TMB code.
- r) Add in the supplemental optimization code to compute MSY quantities by year using year-specific parameters and catch allocations between fleets.

Work plan

28. In addition to the June 2024 meeting, the group discussed online workshops noting that two 2-hour online meetings each year are part of the project. The group noted that one webinar to be conducted during December this year would provide the opportunity for OMMP members that did not attend to workshop to review this report and provide feedback on direction and project status. A second meeting would be prior to the June meeting and may be more technical in nature.

List of Attachments

Attachment

1. **List of Participants**
2. **Agenda**
3. **List of Documents**
4. **Presentation - Project Introduction**
5. **Presentation - The Model Code**
6. **Presentation – Running Models**
7. **Presentation - Bayesian Inference and sbt**
8. **Presentation – Next Steps**

List of Participants
Operating Model Specification and Software Upgrade Workshop

First name	Last name	Title	Organisation	Email
CHAIR				
Ana	PARMA	Dr	Centro Nacional Patagonico	anaparma@gmail.com
SCIENTIFIC COMMITTEE CHAIR				
Kevin	STOKES	Dr	ESC Chair	kevin@stokes.net.nz
SCIENTIFIC ADVISORY PANEL				
James	IANELLI	Dr	University of Washington	jim.ianelli@gmail.com
CONSULTANT				
Darcy	WEBBER	Dr	Quantifish	darcy@quantifish.co.nz
MEMBERS				
AUSTRALIA				
Rich	HILLARY	Dr	CSIRO Environment	Rich.Hillary@csiro.au
Paige	EVESON	Ms	CSIRO Environment	Paige.eveson@csiro.au
INDONESIA				
Fayakun	SATRIA	Dr	Research Center for Fishery, National Research and Innovation Agency, Indonesia	fsatria70@gmail.com
Lilis	SADIYAH	Dr	Research Center for Fishery, National Research and Innovation Agency, Indonesia	sadiyah.lilis2@gmail.com
JAPAN				
Tomoyuki	ITOH	Dr	Fisheries Resources Institute, Japan Fisheries Research and Education Agency	ito_tomoyuki81@fra.go.jp
FISHING ENTITY OF TAIWAN				
Ching-Ping	LU	Dr	National Taiwan Ocean University	michellecplu@gmail.com

Agenda
Operating Model Specification and Software Upgrade Workshop
(OM Coding WS)
20 – 22 November 2023
Tokyo, Japan

1. Introduction

- 1.1 Helping with any installation issues, compiler issues, and using GitHub

2. Overview of the model code

3. Running models

- 3.1 Running some models together
- 3.2 Running a grid together

4. Bayesian inference

5. Next steps

- 5.1 Tackling some issues
- 5.2 Implementing some model changes

List of Documents

1. Provisional Agenda
2. List of Participants
3. (Australia) Hillary, R. and P. Evenson. Population model and likelihood ideas for next CCSBT OM.

Operating Model Specification and Software Upgrade Workshop

Introduction

Darcy Webber

20-22 November 2023
Tokyo, Japan



Contents

1. Project specifications
2. General introduction
3. Software requirements
4. GitHub
5. R package
6. Installation
7. Interactive session



Project specifications

- A (Start year): 2023
- B (Duration): 3 years
- C (General category): OM
- D (Sub category): Asses
- E (Project title): Operating model specification and software upgrade
- :
- I (Impact Scale): High
- J (Impact timing): Med
- K (Priority): to be completed at ESC meetings.
- L (Rank): to be completed at ESC meetings.
- ** (budget source): CCSBT

3



Project specifications: problem definition

F (Problem): The current operating model (OM) specifications, code, and software present challenges for

1. communicating the population dynamics and statistical assumptions underpinning the SBT model;
2. addressing uncertainty within the OM grid; and
3. revising and implementing alternative hypotheses in stocks assessments and future MP evaluations.

4



Project specifications: objectives

G (Objectives):

1. Update and revise OM documentation to match the OM code;
2. Develop new OM implementations in either Stan or Template Model Builder (TMB) software;
3. Code modifications to the OM to be decided by the OMMP Working Group to improve estimation efficiency and allow future flexibility in adding/removing complexity and features as needed;
4. Complete validation test comparing estimates from new implementation with current ADMB version.

5



Project specifications: rationale

H (Rationale): Upgrading to modern software will improve the flexibility, utility, and understanding of the SBT operating and assessment models for all CCSBT participants. Improvements to model structural and statistical procedures will potentially result in better presentation and understanding of historical, current and future SBT stock status, its associated uncertainty, and MP performance.

6

Project specifications: resources



2023	2024	2025
25d Consultant	20d Consultant	20d Consultant
2d MP Coordinator	2d MP Coordinator	2d MP Coordinator
—	1d extra at ESC meeting (VEH, Cat, 3P, 1C, 1Ch, Sec)	—
1d extra at Seattle OMMP meeting (Cat, 3P, 1C, 1Ch)	—	—
3d dedicated inf. OMMP meeting (Tokyo: FreeV, Cat, 3P,1C, 1Ch)	5d dedicated OMMP meeting (Seattle: FreeV, Cat, 3P, 1C, 1Ch)	—
2*2hr online meetings (3P,1C, 1Ch, Sec)	2*2hr online meetings (3P,1C, 1Ch, Sec)	2*2hr online meetings (3P,1C, 1Ch, Sec)

The abbreviations used in are:
 Sec=Secretariat Staff,
 Ch=Independent ESC Chair,
 P=Independent Advisory Panel,
 C=Consultant,
 Cat=Catering only, VEH=venue & equipment hire etc.,
 FreeV=Venue & some equipment at no cost.

7

Project specifications: work plan 2023



- ✓ Cleaning of old code and documentation.
- ✓ Darcy works on new conditioning code to match old code.
- ❑ One or more informal short (1-2 hour) online meetings.
- ✓ One extra day added to the scheduled in-person OMMP meeting to discuss progress.
- ❑ 3-day in-person meeting in November focused on the transition to the new code:
 - ❑ compare conditioning results obtained with old and new code;
 - ❑ show structure and receive feedback;
 - ❑ discuss projection code (could run old projection code with outputs from new code as an intermediate step);
 - ❑ prioritise work (changes to the code) for 2024
 - ❑ Provide training/tutorial

8

Project specifications: work plan 2024



- ❑ 5-day OMMP in person meeting in June to discuss/implement/evaluate changes to the OM (conditioning and projections), and provide training/tutorial
- ❑ One extra day at ESC to discuss progress.

9

General introduction



- The operating model (OM) has been coded in template model builder (TMB)
- I have integrated the model into an R package that I have named *sbt*
- The R package is available online on GitHub
- The R package / GitHub format has many symbiotic benefits including
 - Readily available documentation
 - Continuous integration (CI)
 - Issue/bug tracking
 - Ease of use

10



Software requirements

- The statistical programming language R
 - Recommend an up-to-date version of R (I am using version 4.1.2)
 - Recommend a graphical user interface (GUI) for R (e.g., RStudio)
- R packages
 - *TMB, RcppEigen, tmbstan, rlang*
 - *tidyverse, reshape2, scales, ggridges*
 - *rstan, loo*
 - *parallel, doParallel, foreach, MASS*
 - *knitr, kableExtra, rmarkdown, testthat*
- GitHub
 - You will need a GitHub account to access *sbt* as it is a private repository
 - RStudio has GitHub access built in (see the “Git” tab)

11



GitHub

The *sbt* R package is stored on GitHub: <https://github.com/quantifish/sbt>

This is a private repository so you will need a GitHub account to be able to access the repository. If you do not have access please let me know your username so I can allow you access to the repository.

If you want to clone the repository to your computer then you can use something like:

```
git clone https://github.com/quantifish/sbt
```

12

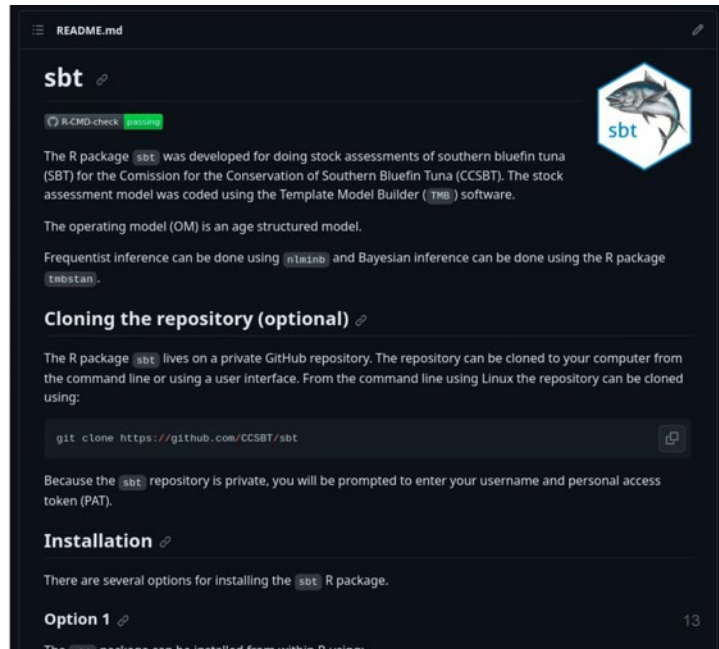
GitHub: README

The README.md file is generated using README.Rmd.

It automatically displays when you navigate to the GitHub repository.

It contains some installation instructions.

Also note the R-CMD-check **passing** badge (more later).



sbt

R-CMD-check **passing**

The R package **sbt** was developed for doing stock assessments of southern bluefin tuna (SBT) for the Comission for the Conservation of Southern Bluefin Tuna (CCSBT). The stock assessment model was coded using the Template Model Builder (**TMB**) software.

The operating model (OM) is an age structured model.

Frequentist inference can be done using **nlminb** and Bayesian inference can be done using the R package **tmstan**.

Cloning the repository (optional)

The R package **sbt** lives on a private GitHub repository. The repository can be cloned to your computer from the command line or using a user interface. From the command line using Linux the repository can be cloned using:

```
git clone https://github.com/CCSBT/sbt
```

Because the **sbt** repository is private, you will be prompted to enter your username and personal access token (PAT).

Installation

There are several options for installing the **sbt** R package.

Option 1

The **sbt** package can be installed from within R using:

R package

The components of the R package include:

- data
- data-raw
- docs
- man
- pkgdown
- R
- src
- tests
- vignettes
- DESCRIPTION
- LICENSE
- NAMESPACE
- NEWS.md
- _pkgdown.yml
- README.md
- README.Rmd
- .gitignore
- .Rbuildignore





R package: data and data-raw

The **sbt/data-raw** directory contains:

- Two directories containing all of the input/output files for two model runs (base22 and base22sqrt_trollpe)
- The **sbt/data-raw/csv** directory which contains “.csv” files with all of the input data (e.g., aerial_survey.csv, catch.csv, cpue.csv, mean_length.csv, tag_recaptures.csv)
- An R script for processing the raw data and saving to the **sbt/data** directory in the compressed “.rda” format:
 - data_csv1.rds, data_labrep1.rda, and data_par1.rda
 - tag_recaptures.rda, tag_releases.rda, and tag_reporting.rda

15



R package: R

The R directory contains all of the R functions that are included with the package. Each function is documented using Roxygen so that when the package is built, the help files (stored in the **sbt/man** directory) are updated automatically. For example:

```

#' Plot CPUE
#'
#' Plot CPUE fit.
#'
#' @param data a \code{list} containing the data that was passed to \code{MakeADFun}.
#' @param object a \code{list} specifying the AD object created using \code{MakeADFun}.
#' @param posterior an \code{rstan} object created using the \code{rstan} function.
#' @param probs a numeric vector of probabilities with values in \code{[0,1]} for plotting quantiles of the posterior distribution.
#' @return a \code{ggplot2} object.
#' @import ggplot2
#' @importFrom rlang .data
#' @importFrom scales pretty_breaks
#' @export
#'
plot_cpue <- function(data, object, posterior = NULL, probs = c(0.05, 0.95)) {
  df <- data.frame(year = data$cpue_year + data$first_yr,
                  obs = data$cpue_obs,
                  pred = object$report()$cpue_pred)

  ggplot(data = df, aes(x = .data$year, y = .data$obs)) +
    geom_point() +
    geom_line(aes(y = .data$pred)) +
    labs(x = "Year", y = "CPUE") +
    scale_x_continuous(breaks = pretty_breaks()) +
    scale_y_continuous(limits = c(0, NA), expand = expansion(mult = c(0, 0.05)))
}

```

Doxygen help

R code / function

16



R package: src

The **sbt/src** directory contains the TMB model itself. The model code is split into two files:

- **sbt/src/sbt_v100.cpp**
- **sbt/src/functions.hpp**

The **sbt/src** directory also contains makefiles for both Linux and Windows machines:

- **sbt/src/Makefile**
- **sbt/src/Makefile.win**

When the package is built/installed, the model is compiled and then added to the R package.

17



R package: tests

The **sbt/tests** directory contains a suite of tests that are run using the R package *testthat*. These tests include comparing the values of likelihood components and derived values from the TMB model and ensuring that they adequately match the ADMB model (many of these tests are also illustrated in the Comparison with the ADMB model vignette).

These tests will run when:

- The “Test” button in RStudio is pushed or using `devtools::test()`
- The “Check” button in RStudio is pushed or using `devtools::check()`
- A push is made to the GitHub repository (i.e., the tests are run automatically using GitHub actions and an email is sent out if any tests fail)

18

R package: vignettes



The **sbt/vignettes** directory contains all of the package vignettes which are articles/pages that can include helpful information and examples. So far, the vignettes for *sbt* include:

- **sbt** - also called “Get started” on the website. This contains an example of how to run a single grid cell.
- **sbt_dev** - a guide for model developers.
- **sbt_equations** - not finished.
- **sbt_mcmc** - Bayesian inference including an example of how to run an MCMC.
- **sbt_vs_admb** - comparison between *sbt* and the ADMB model including a fixed parameter run and a model fit.

19

R package: logo

- The logo is generated using **sbt/man/figures/logo.R**
- The SBT picture was painted by Joanne Webber (my mum) specially for the *sbt* package



20



R package: website

The website is stored in **sbt/docs** and is built using the R package `pkgdown` and can be built locally using:

```
pkgdown::build_site()
```

If the website is built (as above) and then pushed to the repository, it will become publicly available here:

<https://quantifish.co.nz/sbt/>

Alternatively, the website can be built automatically by GitHub using actions (i.e., continuous integration) and will update whenever a commit is pushed to GitHub.

21



R package: website

Pages on the website include:

- **sbt - `sbt/README.Rmd`**
- **Get started - `sbt/vignettes/sbt.Rmd`**
- **Articles**
 - Model development - `sbt/vignettes/sbt_dev.Rmd`
 - Model equations - `sbt/vignettes/sbt_equations.Rmd`
 - Bayesian inference - `sbt/vignettes/sbt_mcmc.Rmd`
 - Comparison with the ADMB model - `sbt/vignettes/sbt_vs_admb.Rmd`
- **Reference**
 - Helper functions
 - Plots
 - Data
- **Changelog - `sbt/NEWS.md`**

22

R package: website



Navigation tabs

Article title

Article source

Navigation tabs

Code chunk

23

The screenshot shows the website for the 'sbt' R package. At the top, there is a navigation bar with tabs for 'Get started', 'Articles', 'Reference', and 'Changelog'. The 'Articles' tab is selected. Below the navigation bar, the article title 'Bayesian inference' is displayed, along with the source 'yjoettes/sbt_mcmc_blog'. The article content includes an 'Introduction' section, a 'Load inputs' section with a code chunk, and a 'Model setup' section. The code chunk contains R code for loading libraries and setting options. The 'Model setup' section provides an example of how to use the package. The page number '23' is visible in the bottom right corner.

```
library(tidyverse)
library(sbt)
library(tmbstan)
library(kableExtra)
library(bayesplot)

theme_set(theme_bw())

# Specify the number of cores for MCMC
options(mc.cores = parallel::detectCores())
```

R package: continuous integration (CI)



GitHub actions is used to automatically run R CMD CHECK whenever a commit is pushed to the GitHub repository:

- Controlled using the `.github/workflows/R-CMD-check.yaml` file
- Compiles the model / package on Linux, Mac, and Windows machines
- Checks for errors in the code
- Runs tests using the R package `testthat`

Installation



A list of different ways to install *sbt* can be found in the **README** document.
For this workshop it would be best if you could:

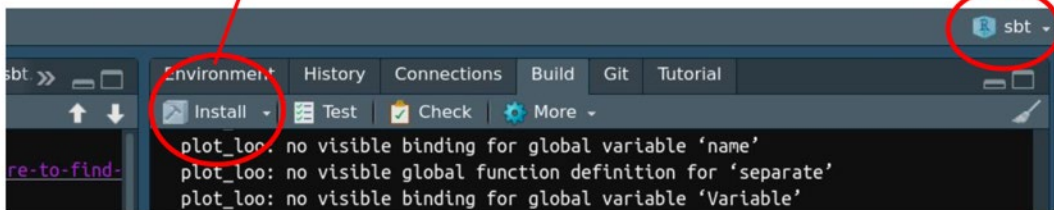
1. Install GitHub, RStudio, and Rtools (if on Windows)
2. Clone the *sbt* repo: `git clone https://github.com/quantifish/sbt`
3. Clone the working repo: `git clone https://github.com/CCSBT-Dev/sbt_models`

25

Installation



1. Open RStudio
2. Recommend creating a project using New Project->Existing Directory->Navigate to sbt->Create Project
3. Install *sbt* using: `devtools::install("sbt")` or pushing the "Install" button



26

Interactive session



Helping with anything:

- installation issues
- compiler issues
- using GitHub

Operating Model Specification and Software Upgrade Workshop

The Model Code

Darcy Webber

20-22 November 2023
Tokyo, Japan



1

Contents

- Template model builder (TMB)
- Where to find the *sbt* model/code
- Differences between the TMB and ADMB languages
- Comparisons between the TMB and ADMB models
- Adding new code

2

Template model builder (TMB)

Template Model Builder (TMB) is an R package for fitting statistical latent variable models to data. It is inspired by ADMB. Unlike most other R packages the model is formulated in C++. This provides great flexibility, but requires some familiarity with the C/C++ programming language.

A TMB project consists of an R file (*.R) and a C++ file (*.cpp). The R file does pre- and post processing of data in addition to maximising the log-likelihood contained in *.cpp.

The purpose of the C++ program is to evaluate the objective function, i.e. the negative log-likelihood of the model. The program is compiled and called from R, where it can be fed to a function minimizer like [nlminb](#).

3

Where to find the sbt model/code

The *sbt* model/code can be found in the *sbt* repository under the source (src) directory and consists of two files:

- `sbt/src/sbt_v100.cpp`
- `sbt/src/functions.hpp`

4

Functions in *sbt*

- `get_phi`
- `get_M`
- `get_rho`
- `get_recruitment`
- `get_selectivity`
- `get_initial_numbers`
- `get_harvest_rate`
- `first_difference`
- `third_difference`
- `norm2`
- `square`
- `get_length_like`
- `get_age_like`
- `get_cpue_like`
- `get_aerial_survey_like`
- `get_troll_like`
- `get_tag_like`
- `get_POP_like`
- `get_HSP_like`
- `get_GT_like`
- `get_sel_like`
- `get_recruitment_prior`

5

Functions in *sbt*

```
template <class Type>
vector<Type> get_initial_numbers(Type B0, Type steep, Type *R0, Type *alpha, Type *beta,
vector<Type> M_a, array<vector<Type>> phi_ya) {

    int n_age = M_a.size();
    vector<Type> rel_N(n_age);

    rel_N(0) = Type(1.0);
    for (int ia = 1; ia < n_age; ia++) {
        rel_N(ia) = rel_N(ia - 1) * exp(-M_a(ia - 1));
    }
    rel_N(n_age - 1) /= Type(1.0) - exp(-M_a(n_age - 1));

    *R0 = B0 / (phi_ya(0) * rel_N).sum(); // R0(ixx) = B0(ixx)/(column(phi,first_yr)*relN);
    *alpha = (Type(4.0) * steep * *R0) / ((Type(5.0) * steep) - Type(1.0));
    *beta = (B0 * (Type(1.0) - steep)) / ((Type(5.0) * steep) - Type(1.0));

    return *R0 * rel_N; // N(1,first_yr) = R0(1)*relN;
}
```

```
number ysa(0, 0) = get_initial_numbers(par_B0, par_h, &R0, &alpha, &beta, M_a, phi_ya);
```

Inputs

Pointer

Functions can only return one container (e.g., a vector or a matrix).

Variables declared in a function are not available globally.

A pointer is an address to a location in memory that can be passed to a function.

When a value is assigned to the pointer, it saves the value to the original location of the defined variable. Thus, we can return many things from a function.

6

Differences between the TMB and ADMB languages

Type definitions for variables

```
// Declare variables
Type R0, alpha, beta;
vector<Type> S_a(n_age);
array<vector<Type>> number_ysa(n_year + 1, n_season);
array<vector<Type>> hrate_ysa(n_year + 1, n_season);
vector<Type> urate_f(n_fishery);
array<vector<Type>> catch_pred_fya(n_fishery, n_year + 1);
array<Type> catch_pred_ysf(n_year + 1, n_season, n_fishery);
vector<Type> spawning_biomass_y(n_year + 1);
```

```
int nC, nK, cmin, cmax, cdif;
Type xtmp, cumS, pp;
matrix<Type> gamx_ya(n_year, n_age);
vector<Type> phi_a(n_age);
vector<Type> phsp_i(n_hsp);
vector<Type> lp(n_hsp);
```

```
FUNCTION void get_HSP_like(int ix) // RH 2017: HSP data likelihood
  int c1,c2,cmin,cmax,nC,nK,diffc;
  dvariable cumS,xtmp;
  qhsp = mfxp(lnqhsp);
```

```
FUNCTION void get_indices_like(int ix)
  int ii, iy;
  dvariable tmpN, like_aerial;
  dvector weights(2,4);
  ln_like(ix) = 0.;
```

7

Differences between the TMB and ADMB languages

Zero indexing

```
for (int i = 0; i < n_af; i++) {
  f = af_fishery(i) - 1;
  y = af_year(i);
  amin = af_min_age(i);
  amax = af_max_age(i);
  n_a = amax - amin + 1;
```

8

Differences between the TMB and ADMB languages

If statements that depend on a parameter or quantity derived from a parameter are NOT allowed.

Things you should NOT do in TMB

Josh Pritsker edited this page on Jul 31, 2020 · 7 revisions

- Do not use `if(x)` if `x` is a `PARAMETER`, or is derived from a parameter. TMB will remove the whole if-statement and hence produce surprising results. As an alternative, you can use `CondExp` statements. However, this can easily create a non-differentiable likelihood, so make sure you know what you are doing (i.e., check the differentiability of ahead of time). (See: [this thread](#) and [this comment](#))
- Do not attempt to weight the loglikelihood within the C++ code (See: [here](#)). This results in a non-normal latent distribution, decreasing the accuracy of the Laplace approximation.

9

Differences between the TMB and ADMB languages

If statements that depend on a parameter or quantity derived from a parameter are NOT allowed.

AD Conditional Expressions

Syntax

```
result = CondExpRel(left, right, if_true, if_false)
```

Purpose

Record, as part of an AD of `Base` operation sequence, the conditional result

```
if( left Cop right )
  result = if_true
else
  result = if_false
```

The relational `Rel` and comparison operator `Cop` above have the following correspondence:

```
Rel  Lt  Le  Eq  Ge  Gt
Cop  <  <= == >= >
```

If `f` is the `ADFun` object corresponding to the AD operation sequence, the assignment choice for `result` in an AD conditional expression is made each time `f.Forward` is used. `AD comparison operators` which are boolean valued and not included in the AD operation sequence.

10

Differences between the TMB and ADMB languages

If statements that depend on a parameter or quantity derived from a parameter are NOT allowed.

```
if (phsp_i(i) > 0.0) {  
  if (nK > 0) lp(i) = nK * log(phsp_i(i)) + (nC - nK) * log(Type(1.0) - phsp_i(i));  
  if (nK == 0) lp(i) = nC * log(Type(1.0) - phsp_i(i));  
}
```



```
lp(i) = CppAD::CondExpGt(phsp_i(i), Type(0.0),  
  nK * log(phsp_i(i)) + (nC - nK) * log(Type(1.0) - phsp_i(i)),  
  Type(0.0));
```

From the HSP likelihood function.

Also in the POP likelihood and harvest rate calculations (we should review the harvest rate function). 11

Differences between the TMB and ADMB languages

Random effects - for example, you could estimate sigmaR

```
# Specify the random effects  
# Random <- c("par_rdev_y", "par_sels_change_i")  
Random <- c()  
  
# Create the AD object ----  
  
obj <- MakeADFun(data = Data, parameters = Params, map = Map, random = Random,  
  hessian = TRUE, inner.control = list(maxit = 1000), DLL = "sbt")
```

12

Comparisons between the TMB and ADMB models

See the **Comparison with the ADMB model** vignette - go to website.

13

Adding new code

See the **Model development** vignette - go to website.

14

Operating Model Specification and Software Upgrade Workshop

Running Models

Darcy Webber

20-22 November 2023
Tokyo, Japan



1

Contents

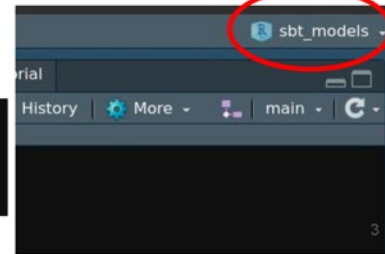
- Running a model (single grid cell)
- Generating a grid of model runs
- Extracting outputs
- Generating figures and tables

2

Running a model (single grid cell)

1. Open RStudio
2. Recommend creating a project using New Project->Existing Directory->Navigate to sbt_models->Create Project
3. Open the file `example_run.R`
4. Step through this code - this will run a single grid cell!

```
# Optimize ----  
opt <- nlminb(start = obj$par, objective = obj$fn, gradient = obj$gr,  
             lower = bnd$lower, upper = bnd$upper)
```



Running a model (single grid cell)

Alternatively, see the **Get started** tab on the website which steps through a model run.

The screenshot shows the 'sbt model' website. At the top, there are navigation tabs: 'Get started', 'Articles', 'Reference', and 'Changelog'. A search bar is on the right. The main heading is 'sbt model' with a source link 'vignettes/sbt.Rmd'. Below this is the 'Load inputs' section, which explains that the required R libraries are loaded and the 'sbt' TMB model is dynamically loaded. It provides R code for loading libraries and setting a theme. Below the code, it mentions that example inputs are loaded from the 'sbt' package. A sidebar on the right titled 'On this page' lists: 'Load inputs', 'Input tables', 'Input plots', 'Model setup', 'Optimisation', and 'Plot outputs'. The page number '4' is visible in the bottom right corner.

The screenshot shows an RStudio window with a tutorial titled 'Running a model (single grid cell)'. The tutorial includes a list of steps: 1. Open RStudio, 2. Recommend creating a project using New Project->Existing Directory->Navigate to sbt_models->Create Project, 3. Open the file example_run.R, 4. Step through this code - this will run a single grid cell!. Below the steps is a code block for the 'optimize' function:

```
opt <- nlm(bstart = objspar, objective = objfn, gradient = objgr, lower = bndlower, upper = bndupper)
```

. To the right of the code is a small screenshot of the RStudio interface with a red circle around the 'sbt_models' folder in the file browser. Below the code block is another heading 'Running a model (single grid cell)' followed by the text 'Alternatively, see the **Get started** tab on the website which steps through a model run.' At the bottom of the RStudio window, there is a smaller version of the 'sbt model' website screenshot seen in the previous block. The page number '4' is visible in the bottom right corner.

Extracting outputs

See the **Reference** section of the website under the heading **Helper functions** for a list of functions, some of which can be used to extract outputs.

Otherwise, take a look at the code that produces the plots (next slide).

Helper functions

Build a data set or extract something.

`get_array()`
Convert a 2D array of vectors to an array

`get_bounds()`
Get default parameter bounds

`get_data()`
Set up the data input file

`get_dl()`
Obtain dl

`get_grid()`
Set up a grid

`get_length_at_age()`
Obtain the length at age array

`get_loo()`
Obtain the leave-one-out information criterion (LOO IC)

`get_map()`
Get default parameter mapping

`get_parameters()`
Get default initial parameter values

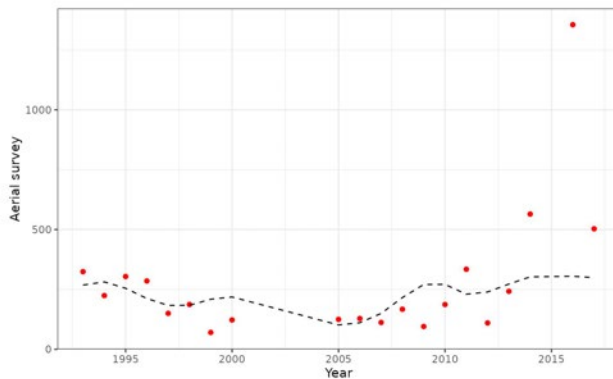
`get_posterior()`
Obtain samples from the posterior distribution of reported quantities

`get_sel_list()`
Get list of selectivity parameters

6

Generating figures and tables

See the **Reference** section of the website under the heading **Plots** for a list of plots that can be produced.



Plots

Plot an input or output.

`plot_aerial_survey()`
Plot aerial survey

`plot_biomass_spawning()`
Plot spawning biomass

`plot_catch()`
Plot catch

`plot_cpue()`
Plot CPUE

`plot_hrate()`
Plot harvest rate

`plot_initial_numbers()`
Plot initial numbers

`plot_length_at_age()`
Plot length

`plot_loo()`
Plot PSIS LOO

`plot_natural_mortality()`
Plot natural mortality

7

Operating Model Specification and Software Upgrade Workshop

Bayesian Inference

Darcy Webber

20-22 November 2023
Tokyo, Japan



1

Contents

- Introduction
- Running an MCMC
- Extracting samples from the posterior
- Plotting MCMC outputs
- MCMC diagnostics
- Reduced Bayesian grid structure
- Comparison of grid and MCMC uncertainty
- Model comparison
- Model averaging/stacking

2

Introduction

Bayesian inference is a method of statistical inference in which Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available.

Markov chain Monte Carlo (MCMC) methods comprise a class of algorithms for sampling from a probability distribution. By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by recording states from the chain.

3

Running and MCMC

1. Open RStudio
2. Open the file `example_mcmc.R`
3. Step through this code - this will run an MCMC for a single grid cell!
4. MCMC takes about 3 hours to complete (but don't worry, I prepared one earlier!)

```
# Run MCMC ----  
  
if (FALSE) {  
  mcmc1 <- tmbstan(obj = obj, lower = bnd$lower, upper = bnd$upper,  
                 init = rep(list(Params), 2), chains = 2,  
                 control = list(max_treedepth = 12, adapt_delta = 0.9))  
  save(mcmc1, file = "mcmc1.rda")  
} else {  
  load("mcmc1.rda")  
}
```

4

Extracting samples from the posterior

Extracting samples from the posterior can be done using the `get_posterior` function.

Unfortunately, this function is very slow and (currently) only extracts one output at a time.

Profiling revealed that the slowest part of this function is the `object$report(par)` call making it difficult to speed up.

Obtain samples from the posterior distribution of reported quantities



Source: [R/get_posterior.R](#)

When samples from the posterior distribution are obtained using `tmbstan`, only the model parameters can be accessed directly from the `stanfit` object (i.e., derived quantities such as `par_B0` or `M_a` cannot be accessed directly). Accessing derived quantities must be done using the this function.

Usage

```
get_posterior(object, posterior, pars = "par_B0", option = 1)
```

Arguments

object

A `list` specifying the AD object created using `MakeADFun`.

posterior

An `rstan` object created using the `tmbstan` function.

pars

The parameter(s) to be extracted.

option

The parallel option to use.

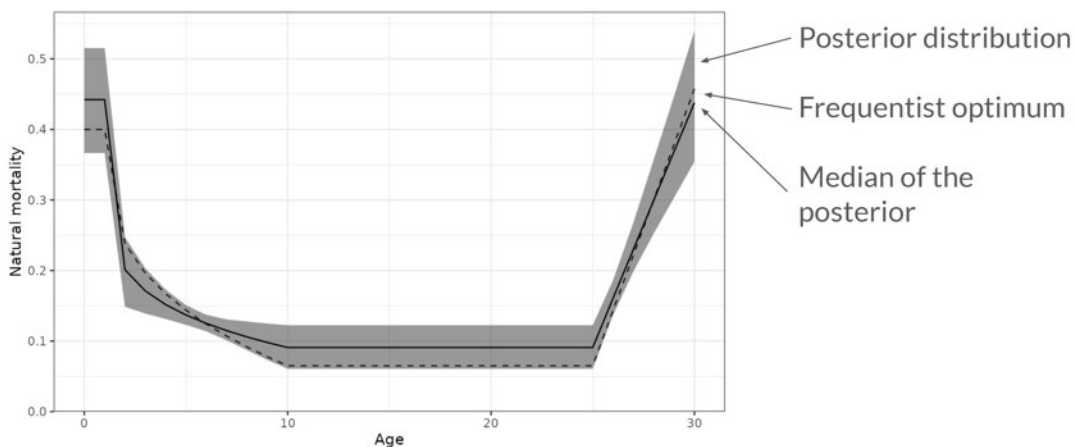
Value

A `data.frame`.

5

Plotting MCMC outputs

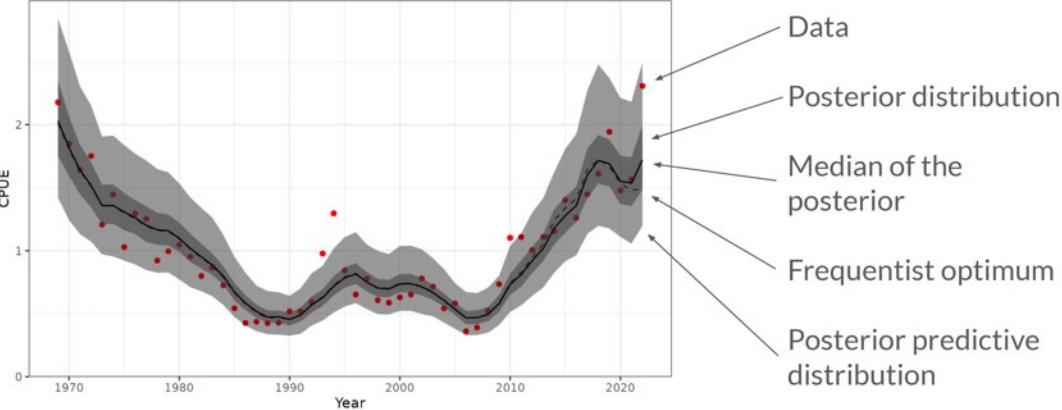
Function: `plot_natural_mortality`



6

Plotting MCMC outputs

Function: `plot_cpue`

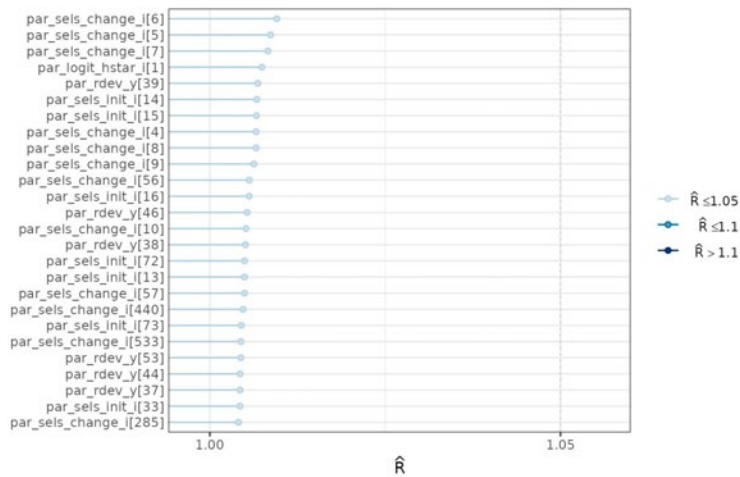


MCMC diagnostics

- There should be no divergent transitions
- The effective sample size of all parameters should be at least 400
- The Rhat statistic for all parameters should be less than 1.01
- MCMC trace plots should look good
- Should check for parameters up against bounds

8

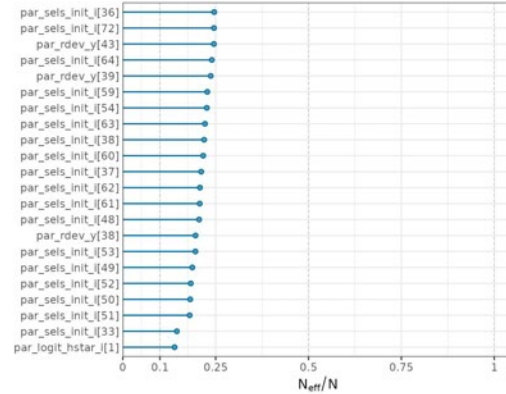
MCMC diagnostics: Rhat



9

MCMC diagnostics: effective sample size

Parameter	Mean	2.5%	Median	97.5%	Eff. N	Rhat
lp__	-6383.375	-6436.974	-6383.432	-6332.278	693.891	1.001
par_log_B0	15.981	15.651	15.982	16.311	520.853	1.003
par_log_m0	-0.819	-1.004	-0.816	-0.663	2097.951	1.001
par_log_m4	-1.889	-2.030	-1.886	-1.755	1089.033	1.001
par_log_m10	-2.410	-2.803	-2.398	-2.100	670.913	1.001
par_log_m30	-0.825	-1.038	-0.826	-0.617	1937.215	1.000
par_log_cpue_q	-0.017	-0.075	-0.016	0.040	2839.957	1.000
par_logit_hstar_i[1]	-4.318	-8.184	-3.948	-3.174	278.747	1.007
par_logit_hstar_i[2]	-4.663	-9.265	-4.004	-2.640	802.172	1.000

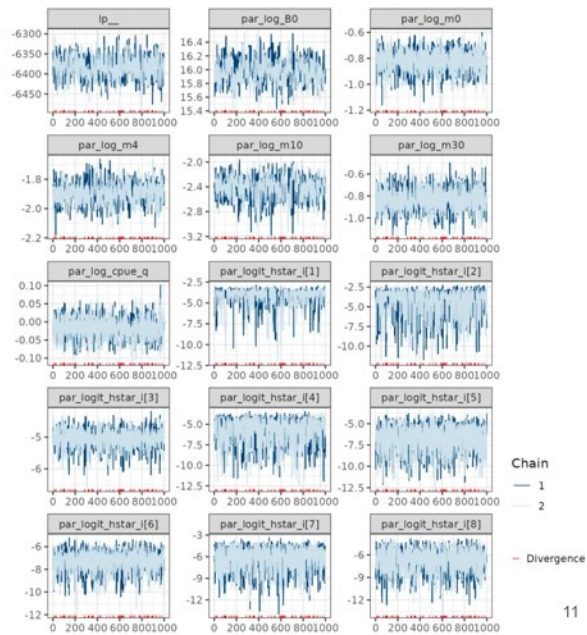


10

MCMC diagnostics: trace plots

Signs of poor mixing for B0 (also has relatively low effective sample size rate), m10, and possibly m4. These issues are all easily fixed with more iterations, longer warm up, tweaking MCMC settings, and/or thinning.

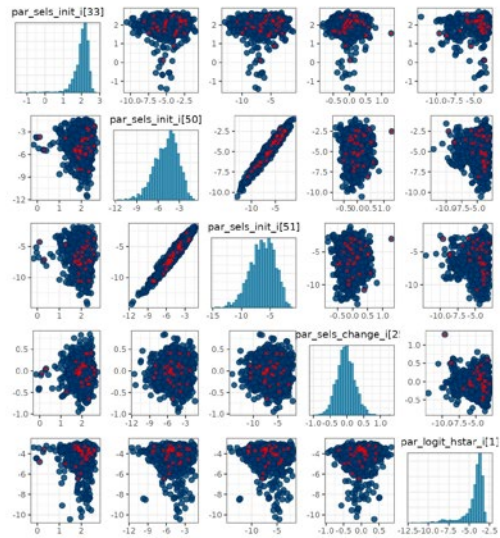
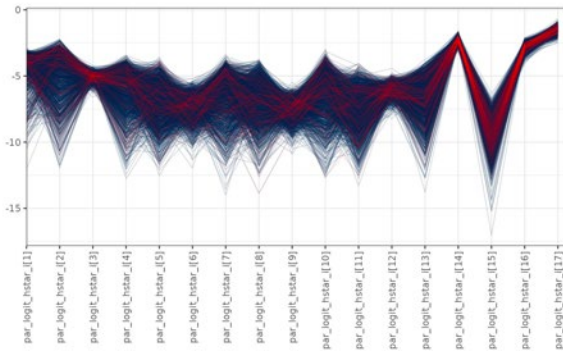
hstar[1] is having a very hard time (it has the worst effective sample size rate, see previous slide).



11

MCMC diagnostics: pairs and parallel coordinates plots

Divergent transitions shown in red.



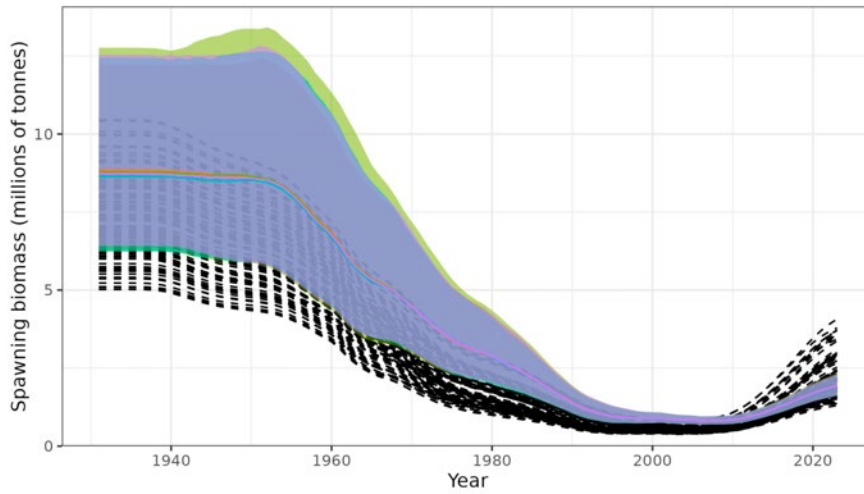
12

Reduced Bayesian grid structure

- Run a grid using the `run_grid` function (108 runs)
 - $m_0 = \{0.4, 0.45, 0.5\}$
 - $m_{10} = \{0.065, 0.085, 0.105\}$
 - $h = \{0.55, 0.63, 0.72, 0.8\}$
 - $\psi = \{1.5, 1.75, 2\}$
- Propose to run a reduced grid of MCMCs (12 MCMCs). No function to do this yet. Could reduce this grid even further?
 - Estimate m_0
 - Estimate m_{10}
 - $h = \{0.55, 0.63, 0.72, 0.8\}$
 - $\psi = \{1.5, 1.75, 2\}$

13

Comparison of grid and MCMC uncertainty

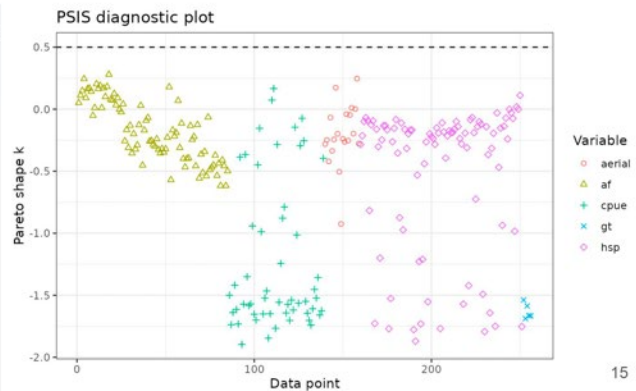


14

Model comparison

The leave-one-out information criterion (LOO IC) can be used for model diagnosis (e.g., identifying specific problematic data points), model selection, and model averaging.

```
if (FALSE) {  
  loo1 <- get_loo(data = Data, object = obj, posterior = mcmc1)  
  save(loo1, file = "loo1.rda")  
} else {  
  load("loo1.rda")  
}  
  
print(loo1)  
#>  
#> Computed from 2000 by 4110 log-likelihood matrix  
#>  
#>      Estimate      SE  
#> elpd_loo  5562.6  767.6  
#> p_loo     135.2   16.0  
#> looic    -11125.2 1535.1  
#> -----  
#> Monte Carlo SE of elpd_loo is NaN.  
#>  
#> All Pareto k estimates are good (k < 0.5).  
#> See help('pareto-k-diagnostic') for details.
```



15

Model averaging/stacking

Model averaging or stacking can also be done using the LOO package. First the model weights can be extracted using the `loo_model_weights` function. Then samples from each model can be extracted from each of the models according to their model weights (or something along these lines).

```
> loo::loo_model_weights(x = list(loo_grid1, loo_grid2, loo_grid3, loo_grid4),
+                          method = "pseudobma")
Method: pseudo-BMA+ with Bayesian bootstrap
-----
      weight
model1 0.362
model2 0.131
model3 0.246
model4 0.261
```

Operating Model Specification and Software Upgrade Workshop

Next Steps

Darcy Webber

20-22 November 2023
Tokyo, Japan



1

Contents

- Outstanding issues
- Model changes
- Projections code
- Management procedure evaluation (MPE) code
- Tackling some issues
- Implementing some model changes

2

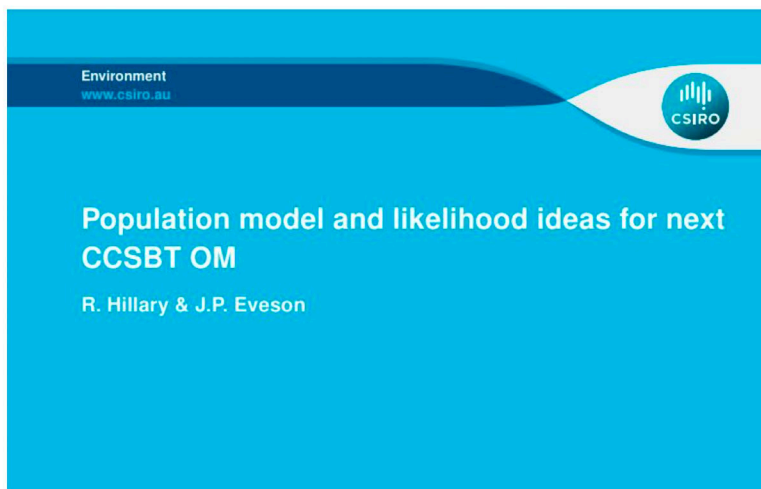
Outstanding issues

All outstanding issues can be found here:

<https://github.com/quantifish/sbt/issues>

3

Model changes



4

Model changes

- Change to age-length model (big job)
- Selectivity
- What else?

Projections code

Nothing done yet. Can be done using simulation block in TMB. For example:

```
SIMULATE {  
  y = rnorm(mu, sd); // Simulate response  
  REPORT(y); // Report the simulation  
}
```

And this can be run using:

```
obj$simulate()
```

<https://kaskr.github.io/adcomp/Simulation.html>

6

Tackling some issues

- Check age likelihood again (small difference in likelihood) Apply maximum harvest rate in `get_harvest_rate` function
- Revise the `get_tag_like` function and output likelihood for each data point so that the LOO IC can be calculated
- What should we do with the `get_posterior` function? It is slow. Could output posterior for all quantities in `REPORT`, which would be even slower, but would only need to be done once.
- What should we do with selectivity?
- Review all priors and make them all proper.
- Review all likelihoods and change to R style likelihoods where applicable.

7

Implementing some model changes

- Change to age-length model (big job)
- Selectivity
- What else?

