PEER REVIEW OF THE NORTH ATLANTIC SWORDFISH MANAGEMENT STRATEGY EVALUATION (MSE) CODE AND ALGORITHMS

Anonymous[1]

SUMMARY

This paper presents a draft peer review of the source code in the SWOMSE R package and its dependencies in the openMSE package. A static code analysis for programming style and package structure was completed and followed by a line-by-line review of all R, TMB, and C++ functions for mathematical accuracy and potential performance bottlenecks, and finally, simulation tests on the package, including estimating model equilibria via simulation, and performing unit-tests on core functions. Particular attention was paid to the translation from a sex-structured, multi-fleet Stock Synthesis 3 assessment model to a single-sex, single-fleet operating model, and its effect on model equilibria. Overall, there were relatively few errors and omissions in the source code and documentation, which is encouraging given the nature and scope of the package; however, the population dynamics code implementing the plus group was incorrect, leading to positively biased variability in population biomass and fishing mortality. We demonstrate this error, as well as its correction, via simulation. Otherwise, we noted missing documentation for several key calculations, importantly, the specifics of how SS3 assessments were used to condition the SWOMSE operating model, and a tendency for the TAC to not be completely caught in low fixed TAC simulations.

RÉSUMÉ

Ce document présente un examen par des pairs provisoire du code source du progiciel SWOMSE R et de ses dépendances dans le progiciel openMSE. Une analyse du code statique pour le style de programmation et la structure du progiciel a été réalisée et suivie d'un examen ligne par ligne de toutes les fonctions R, TMB et C++ à des fins de précision mathématique et d'éventuels goulots d'étranglement des performances, et finalement de tests de simulation sur le progiciel, incluant l'estimation des conditions d'équilibre du modèle par simulation ainsi que la réalisation de tests unitaires sur les fonctions de base. Une attention particulière a été accordée à la traduction d'un modèle d'évaluation Stock Synthesis 3 structuré par sexe, pluri-flottilles en un modèle opérationnel à un seul sexe et une seule flottille, et son impact sur les conditions d'équilibre du modèle. Dans l'ensemble, il y avait relativement peu d'erreurs et d'omissions dans le code source et la documentation, ce qui est encourageant au vu de la nature et de la portée du progiciel ; toutefois, le code de la dynamique des populations appliquant le groupe plus était incorrect, donnant lieu à une variabilité positivement biaisée de la biomasse de la population et de la mortalité par pêche. Nous démontrons cette erreur, et sa correction, par simulation. Nous avons noté autrement une documentation manquante pour plusieurs calculs clés, et surtout les détails de la façon dont les évaluations SS3 avaient été utilisées pour conditionner le modèle opérationnel SWOMSE et une tendance du TAC à ne pas être complètement capturé dans les simulations ayant un faible TAC établi

RESUMEN

Este documento presenta un proyecto de revisión por pares del código fuente del paquete R SWOMSE y sus dependencias en el paquete openMSE. Se finalizó un análisis del código estático para el estilo de programación y la estructura del paquete seguido de una revisión línea por línea de todas las funciones de R, TMB y C++ para ver su precisión matemática y posibles cuellos de botella del desempeño. Finalmente, se realizaron pruebas de simulación del paquete, lo que incluye estimar el equilibrio del modelo por medio de simulación y realizar pruebas unitarias de las funciones principales. Se prestó particular atención a la traducción desde un modelo de evaluación Stock Synthesis 3 multiflota y estructurado por sexo a un modelo operativo de un solo sexo y una sola flota

[1] Landmark Fisheries Research, 213-2414 St Johns St, Port Moody, BC, V3H 2B1, Canada

y a su efecto en el equilibrio del modelo. En general, hubo relativamente pocos errores y omisiones en el código fuente y la documentación, lo que es alentador dada la naturaleza y el alcance del paquete, sin embargo, el código de la dinámica de la población que implementa el grupo plus era incorrecto, por lo que conducía a una variabilidad sesgada positivamente en la biomasa de la población y la mortalidad por pesca. Demostramos este error, así como su corrección por medio de simulación. Por otra parte, hemos advertido que falta documentación para varios cálculos clave, siendo los más importantes los detalles de cómo se utilizaron las evaluaciones de SS3 para condicionar el modelo operativo de SWOMSE y una tendencia a que el TAC no sea capturado completamente en las simulaciones de un TAC bajo fijado.

1. Introduction

The International Commission for the Conservation of Atlantic Tunas (ICCAT) has initiated a Management Strategy Evaluation (MSE) for the North Atlantic Swordfish (SWO) transboundary fishery. A key part of any MSE process is the closed-loop feedback simulation framework, which is a linked set of models representing the biological population, fishery independent and dependent observations and removals, and harvest decision making components of the fishery system. The R package SWOMSE, developed by Blue Matter Science, implements the closed-loop feedback simulation framework used for the SWO MSE, providing functions that together simulation test candidate management procedures (CMPs) against (yet to be defined) quantitative fishery management objectives.

This document, submitted to the Standing Committee on Research and Statistics (SCRS), reports the findings of an in-depth code review, conducted by Landmark Fisheries Research, of the SWOMSE R package and its dependencies in the openMSE package. The package version reviewed below is current as of May 12, 2021, and any updates to any of the package code since that time are not reviewed. However, the reconditioned SS3 assessments for Operating Model 1, dated May 19 2021, were used for simulation tests.

The structure of this review includes (i) review and recommendations for the documentation, (ii) a set of recommendations for the package code, and (iii) some simulation and unit tests of the package behaviour under scenarios testing model equilibria and model response to extreme catches. Within the code reviews, we identify Major or Minor Recommendations given their overall impact on expected results (e.g., Minor Recommendations are usually optional style and coding suggestions). Supporting details of the static code analyses and line-by-line review can be found in Appendices A and B, respectively.

2. Review of documentation, code, and simulation testing

2.1 Documentation

The Trial Specifications Document (TSD) is clearly written and informative; however, a precise mathematical description of the translation simplifying the SS3 operating model from a sex-structured, multi-fleet model to a combined-sex, single fleet operating model was not included. A webpage comparing SS3 and openMSE models for Operating Model 1 in the SWOMSE package showed that the spawning biomass is much larger for the SWOMSE OM than the SS3 model, due to the inclusion of males in the spawning biomass, but, otherwise, much closer on the biomass depletion scale[2]. Catches and fishing mortality rates in the MSEtool reconstruction appear, on average, positively biased compared to their corresponding SS3 values.

There will be some level of irreducible bias in this simplification, because fishing mortality is a non-linear effect, modulated by size selectivity, which cannot simply be averaged across the two sexes similar to the approach used for weight-at-age.

Documentation Recommendation 1: Include a precise mathematical description of the method used to translate the sex-structured, multi-fleet Stock Synthesis 3 (SS3) model outputs to the combined-sex, single-fleet MSEtool operating model.

[2] https://iccat.github.io/nswo-mse/OM_Compare/OM_1.html

Documentation Recommendation 2: Remove the following line from the TSD:

"Provided that the candidate management procedures use information on the relative trends in fishery data (e.g., indices of abundance and historical landings) rather than the absolute level, the disparity in absolute magnitude between the two OM frameworks is unlikely to impact the relative performance of the CMPs."

This statement about expected MP performance based on relative trends is not guaranteed and seems to arm-wave the issue away. It may also be impractical for MP design, as presumably the chosen MP would need to be able to set an absolute TAC for the real SWO system. Further, it gives a misleading impression that the conditioning is more different to the SS3 model than shown in the example comparison. In fact, relative errors in absolute catches and depletion are very small, and the key trends in biomass and catch are all present. Contractors should instead focus on the close match between the SS3 assessment and MSEtool OM biomass and catch series[1] for why the discrepancies are not an issue.

Documentation Recommendation 3: Ensure that OM reports and example comparison plots match the current conditioning of the operating model.

We found in the course of our review that some plots and reported reference points on the SWOMSE website/TSD did not match the quantities in the SWOMSE package. For example, aggregated fleet selectivity in the example OM/SS3 comparison has a peak at age 3 for all years, but the reconditioned OMs appear to have a peak at age 1 (see *Major Code Recommendation 4* below). Similarly, the OM reports have different reference points reported (see section 2.3).

2.2 Code analysis

In what follows, square bracket notation (e.g., [35], [122 – 137]) indicates line numbers in the source code. Source code filenames are given at the beginning of each recommendation.

2.2.1 Major Recommendations

Major Code Recommendation 1: Correct the numbers-at-age calculations in the popdynOneTSCPP() function from the MSEtool package file src/popdynCPP.cpp [35]. This formulation is incorrect and produces overly variable plus group dynamics compared to the correct formulation (**Figure 1**). Apart from being a necessary correction from a model accuracy point of view, this change may also reduce the bias in the conditioning and simplification from the SS3 model mentioned above.

The popdynOneTScpp() function advances the population's numbers-at-age and mortality forward by a single time-step. This function incorrectly models the population dynamics of the plus group [35] using an equilibrium formula. The mathematical form of the equation (as coded) is

$$N_{A+,t} = N_{A-1,t-1}e^{-Z_{A-1,t-1}} + N_{A-1,t-1}e^{-Z_{A-1,t-1}}\frac{e^{-Z_{A,t-1}}}{1-e^{-Z_{A,t-1}}}.$$

where $A+$ is the plus group and t is the time step. This formulation ignores the accumulation of age A+ fish in the plus group, and instead replaces them with recruitment from age A-1, scaled by the limit of a geometric series in $e^{-Z_{A,t-1}}$. The apparent intent is to match the dynamics from the equilibrium survival. The reason the sum of a geometric series can be used for equilibrium survival is that the coefficient of the term $e^{-\sum_a Z_a}$ is the same for all ages (i.e., 1), so the series can be collapsed to its limit. For non-equilibrium population dynamics, that condition is violated by recruitment process error, but the algebra is actually simpler. Starting from

$$N_{A+,t} = \sum_{a=A-1}^{\infty} N_{a,t-1}e^{-Z_{a,t-1}}$$

we can separate the first term to get

$$N_{A+,t} = N_{A-1,t-1}e^{-Z_{A-1,t-1}} + \sum_{a=A}^{\infty} N_{a,t-1}e^{-Z_{a,t-1}}.$$

Next, we apply the assumption that $Z_{a,t-1} = Z_{A,t-1}$ for all $a \geq A$, which gives

$$N_{A+,t} = N_{A-1,t-1}e^{-Z_{A-1,t-1}} + e^{-Z_{A,t-1}} \sum_{a=A}^{\infty} N_{a,t-1}.$$

Finally, substituting $N_{A+,t-1} = \sum_{a=A}^{\infty} N_{a,t-1}$ to reflect that the plus-group accumulates all ages $a \geq A$ gives

$$N_{A+,t} = N_{A-1,t-1}e^{-Z_{A-1,t-1}} + N_{A+,t-1}e^{-Z_{A,t-1}}.$$

The application of the geometric sum here creates a large relative difference between the two specifications, which makes sense as some of the variability in the plus group each year is lost when any age A+ fish are assumed to die (or emigrate). The plus group under the MSEtool specification appears to be on average unbiased from the plus group using correct dynamics, but the MSEtool specification is relatively quite variable (**Figure 1**), which matches expectations since we simulated an average recruitment model. Under the MSEtool specification, the plus group ranges from one-half to three times the correct plus group size under the corrected version (**Figure 1**, right hand column). The code for the comparison is included in Appendix C.

The code can be corrected to equation (1) by changing [35] to the following

```
Nnext(maxage, A) += Ncurr(maxage, A) * exp(-Zcurr(maxage, A));
```

which corrects the term for fish that remain in the plus group. The term for recruitment from age A-1 was already correct, and added in the preceding loop over age classes [30-32].

Major Code Recommendation 2: Analyse sensitivity of the fleet selectivity aggregation to alternative weights (SWOMSE package, R/SWO_SS2OM.R [33]).

We recommend a sensitivity analysis of the weightings used to average selectivity. Given that weights are based on exploitation rate as a proportion of vulnerable biomass, there may be some implicit over-weighting of some fleets, especially since all fleets do not have the same selectivity shape. We recommend additional tests of catch-weighting as well as weighting according to exploitation rate as a proportion of some total biomass that is independent of selectivity (e.g., age-3+).

The aggregated fleet model calculates annual aggregated fleet selectivity and retention probability from an exploitation-rate weighted average of individual fleet selectivity-at-age in each year [539 – 638]. This approach results in time-varying fleet-averaged selectivity and retention curves reflecting the relative fishing pressure applied by each fleet.

It is difficult to follow some parts of this calculation without clear and concise documentation of the simplification process, as recommended above. As noted above, the simplification method does produce some bias in the catch series. However, while there are other methods to achieve the similar results (e.g., Pope's approximation or solving the Baranov equation), there will always be some level of irreducible bias associated with all methods given the averaging of sexual dimorphism, the higher complexity biomass dynamics, and time-varying allocation of catch to each fleet in the history. For example, solving the Baranov equation would make the catches match exactly, but might produce more bias in biomass depletion or numbers-at-age, as it will likely produce different fishing mortality rates.

Major Code Recommendation 3: Convert the convergence criterion in the Newton-Raphson F estimation function calcF() to a relative error.

A unit test of the calcF() function was conducted by deriving catch from a given input fishing mortality rate and model state, and then using that catch to estimate the F from the calcF() function. A grid of Fs was tested, with values ranging from 0.0001 to 10, roughly evenly spaced on the $\log_{10}$ scale (code supplied in Appendix C).

Results show that the maximum number of 50 Newton-Raphson iterations and the convergence threshold for a difference of 10^{-6} t in catch are incompatible (**Table 1**). For all cases where the input $F > 10^{-3}$, the optimization reaches the maximum number of iterations and the difference in catch is always at least 10^{-2} t. Despite never satisfying the convergence criterion before reaching the maximum iterations, the performance of the iterative calcF() estimator is still acceptable, with the relative error in the F estimates on the order of 10^{-5}, with larger errors for the highest tested input F = 10, which is never allowed in simulations given that catches are scaled to 99% of vulnerable biomass.

The convergence criterion of 10^{-6} t is likely too strict for the SWO stock. SWO biomass units are in metric tonnes, but spawning biomass and catch series often have 5 or 4 significant digits, respectively. This means that the convergence criterion is about 10 orders of magnitude smaller than the catches being simulated in the projection period, which is overly conservative. We recommend changing the convergence criterion to a relative error, which would likely lead to performance gains by reducing iterations.

Major Code Recommendation 4: Consider adding alternative projection scenarios testing for robustness to future selectivity patterns.

The current package uses a fixed selectivity pattern in the projection period, which is the exploitation rate weighted average of fleet selectivity in 2017. While there are both asymptotic and dome-shaped selectivity curves in the full multi-fleet model, the fleet aggregated selectivity pattern is heavily domed[2], and does not reach a maximum of 1.0, meaning that apical F values are over-estimated. From the most recent operating model reconditioning, the highest selectivity in projection years is for 1-year old fish (**Figure 2**), meaning that fish feel maximum fishing mortality 3-4 years before they mature. While historical weighted-aggregate selectivity curves all show the peak of the dome at 3 years old[3], there is evidence of time-varying selectivity, especially in for age 2 - 4 fish. The effects of those dynamics may need to be explored, given the fast growth of swordfish at ages 2-4 (**Figure 2**), the lack of catch allocation control in the real fishery, and the implications of these two factors for recruitment overfishing. Further, the exploitation rates used to weight the selectivity in 2017 are highly uncertain, given the high uncertainty associated with biomass and recruitment in the final years of the stock assessments, and the high left skew of the selectivity pattern.

Major Code Recommendation 5: Remove commented out code and deprecated calculations/objects/functions from the packages.

One style issue found by static code analysis was the presence of commented out code (Appendix A). Commented out code is common during development phases of software, often related to debugging or deprecated functions that are no longer actively used by the software (Pham and Yang 2020). The presence of commented out code is controversial, but largely benign, primarily affecting readability of code. However, there is a risk of errors if collaborators or future developers uncomment the code without understanding why it was originally commented. Given that the openMSE package is version controlled on github, there is no reason for the commented out code to remain in the R scripts for production versions of the openMSE package.

Major Code Recommendation 6: Compare simulated length-composition data from the aggregated fleet in the historical period to the true data, aggregated using the same weights as the selectivity averaging.

It is hard to diagnose issues with the simulated length compositions used for setting TACs, as the length compositions are generated by a very different process between the OMs and the SS3 assessments. It would be informative to see how distributions of simulated historical length compositions for each year compare to weighted averages of the true data. This could be combined with the fleet aggregation sensitivity analysis, using the same weights for selectivity aggregation and observation aggregation.

Major Code Recommendation 7: Explain why fixed TAC CMPs do not catch the full TAC by between 7% and 15%.

In the constant TAC CMPs tested in Section 2.3.2, the STC() and LTC() functions showed that average catch was below the fixed TAC for the two lower TAC CMPs. Such behaviour was unexpected as there is no implementation error for the SWO stock, and the TACs were low enough that the stock was able to equilibrate at a high biomass.

[3] https://iccat.github.io/nswo-mse/OM_Compare/OM_1.html

Further, the CalcMPDynamics() function does not appear to count discarding against the TAC, which would be one explanation for this behaviour.

2.2.2 Minor Code Recommendations

Minor Code Recommendation 1: Remove deprecated fleet aggregation code from SWO_SS2OM()

There are several lines of code that appear to calculate vulnerability V and apical fishing mortality F from a difference of total mortality Z and natural mortality M [482 – 525], similar to using Pope's approximation (Pope 1972); however, it appears that aside from the Find array, these aren't used in the operating model.

Minor Code Recommendation 2: Simplify equilibrium survival calculations for the plus-group in the Simulate() and MSYCalcs() functions (See **Table B2** for line numbers).

The equilibrium survival model is oddly specified but mathematically accurate. The plus group age survival dynamics are coded as

$$l_{A+} = l_{A-1}e^{-M_{A-1}} + l_{A-1}e^{-M_{A-1}}e^{-M_A}/(1 - e^{-M_A}).$$

where A+ is the plus group age. As commented in the code, the second term is capturing the sum of a geometric series in e^{-M_A}, i.e. $\sum_{i=1}^{\infty} e^{-iM_A} = \frac{e^{-M_A}}{1-e^{-M_A}}$. It is more common (and slightly simpler) to use

$$l_{A+} = l_{A-1}e^{-M_{A-1}}/(1 - e^{-M_A})$$

which can also be derived from the sum of a geometric series, or equivalently by solving $l_{A+} = l_{A-1}e^{-M_{A-1}} + l_{A+}e^{-M_A}$ for l_A. Both specifications give the same equilibrium survival for an unfished and fished population (**Figure 1**, left column). The latter simpler specification will produce marginally faster run-times as there will be fewer mathematical operations, but otherwise there is no difference.

Minor Code Recommendation 3: Scale step sizes in calcF() function by half when iteratively applying the Jacobian to avoid non-convergence in edge cases.

The mathematics of the calcF() function are correct, but the formulation with no scaling of steps between iterations could lead to non-convergence (e.g, an infinite loop or overshoot) at very high or very low catches. This is a common problem with Newton-Raphson optimization in general, which can be addressed by scaling step-sizes by a fixed fraction, or in some cases by an adaptive step size algorithm.

2.3 Simulation tests

2.3.1 Deriving yield curves by simulation

MSY reference points were estimated for Operating Model 1 by simulation. A grid of CMPs was defined over a fixed harvest rate applied to the SWO spawning stock biomass, which was known exactly at a 1-year lag. We reconditioned OM 1 from the recently completed SS3 reconditioning, modified to have a 200 year projection and deterministic recruitment so that long-run equilibria could be reached. The code used to define the CMPs and modify the OMs is provided in Appendix C.

The CMP grid generates TACs from harvest rates applied to perfectly known spawning biomass, stepping from 0% to 50% over a grid of 100 harvest rates (using approximately 0.5% jumps). For each harvest rate, approximate equilibrium yield and biomass curves are taken as the median over all simulation replicates over the last 2 years of the projection period. While the median over replicates was probably unnecessary given the deterministic recruitment, there were some minor numerical differences between replicates that we wished to integrate over. Approximate optimal fishing mortality (F^*_{MSY}), and the associated yield (*MSY**) and spawning biomass (B^*_{MSY}) were then found by fitting a cubic spline to the catch as a function of fishing mortality and solving for the stationary point (i.e., F^*_{MSY}), which was then substituted into the catch spline and a biomass spline to estimate *MSY** and B^*_{MSY}. For comparison, operating model equilibrium yield and biomass curves were also calculated using the yield-per-recruit approach in the MSYCalcs()

function. MSYCalcs() was applied to a grid of fishing mortality rates from 0 - 3, with the life history and selectivity parameters taken from the MSE@Hist object for OM 1. YPR equilibria were calculated for both 1950 and 2018 to account for time-varying selectivity and weight-at-age from fleet aggregation in the historical period. The reference points reported in the @RefPoints slot of the MSE object were averages of the optima for every year in the historical period.

Simulated equilibria and YPR reference points did not match, likely due to the incorrect plus-group population dynamics. The 1-year lag in biomass information may have an effect on short term dynamics, but not at the end of 200 years of deterministic recruitments. The approximated F_{MSY}^{*} value is very close to the YPR derived reference value for 2018, and the averaged historical F_{MSY} value, but the simulated MSY^{*} value was about 500t lower than the YPR value for 2018 (**Figure 3**). As expected, given the weight-at-age difference in 1950, the OM unfished spawning biomass from 1950 did not match the biomass that the operating model approaches with no fishing over the full 50 year projection (**Figure 3**). When compared to the reference points calculated based on 2018 life history parameters, the SSB0 values are closer, reflecting that the same weight-at-age is used for the entire projection time period. However, there is still a bias, that may be an effect of the incorrect plus-group population dynamics model noted in the recommendations above. At low fishing mortality rates the trajectory of the spawning stock biomass starts to smoothly decrease its growth rate after around 25 years into the projection, but at a 0% harvest rate the biomass climbs past the 2018 SSB0 value and clearly equilibrates above the expected equilibrium (**Figure 4**). At higher levels of fishing mortality, the biomass stops climbing and equilibrates at lower biomasses, as expected, but the biomass is still positively biased, as based on the YPR optimal harvest rates we expect the biomass to equilibrate to B_{MSY} for a harvest rate around 4.5% (**Figure 4**).

Fishing mortality rates associated with higher harvest rates in OM 1 peak at about 0.8, which is at a constant target harvest rate of around 40%, after a notch showing a sharp decline at harvest rates of around 20% (**Figure 5**). Presumably, both the notch and plateau behaviour is caused by scaling catches in the applyMP() function so that they are less than 99% of available biomass. Catch scaling is a standard approach to robustifying the OM against edge-cases that may cause aborted simulations or biased performance metrics, which are explored further in the next section of this review. Such catch scaling explains the behaviour of the right-hand end of the simulated yield curve, where instead of coming down sharply to the x-axis, as is shown for YPR derived yield curves, the simulated yield curve smoothly approaches zero as fishing mortality increases (**Figure 3**). Catch scaling also explains the divergence between the target harvest rate applied to set TACs, and the realised harvest rates defined as catch over spawning biomass (**Figure 6**).

The need for catch scaling at relatively low target harvest rates (as applied to spawning biomass) is related to the large difference between the harvest rates, measured as catch over spawning biomass, and the apical F values applied to the vulnerable biomass. For example, Fmsy = 0.15 under OM 1, but this translates to a Umsy of 4.3 % (**Figure 3**). This stems from the difference between the shape of the aggregated selectivity curve, with a dome that peaks at age-1, and the logistic shape of the maturity ogive (**Figure 2**). Given the lower weight-at-age of fully selected fish, a high apical F produces a much lower harvest rate as a proportion of spawning biomass. While these dynamics are not incorrect, they are important to keep in mind when designing management procedures and choosing target harvest rates, especially if MPs will set TACs based on surveys with asymptotic selectivity or estimates of spawning biomass.

The reference points found by simulation and the YPR derivation do not match the reference points reported on the SWOMSE website[4]. The report for OM 1 on the website show $F_{MSY} = 0.039$, while the YPR derivation gives $F_{MSY} = 0.15$. Spawning biomass is also incorrect, and apparently too low by a factor of 2, which may indicate that female spawning biomass is being reported on the SWOMSE website.

2.3.2 Testing OMs under constant catch

The choice to scale catches to be less than the vulnerable biomass is a standard method of robustifying the simulations against crashes caused by overly aggressive CMPs and/or assessment errors. These ‘edge-cases’ that cause stock biomass crashes can lead to aborted simulation runs, costing precious development and simulation time, or defects that produce biased performance metrics and, possibly, the wrong choice of MP. Therefore, the effects of catch scaling to avoid failure under edge cases should be quantified and well-understood.

[4] https://iccat.github.io/nswo-mse/Reports/OM_Diagnostics/OM-1-Report.html#7_Reference_Points

Five constant catch MPs were simulated for OM 1 with a 200 year projection, with catch set to multiples of MSY: 2.52 kt, 5.04 kt, 10.08 kt (MSY*), 15.12 kt, and 20.16 kt. Under each fixed catch level, projected spawning biomass and realised catch trajectories for each of the 48 replicates with process error were reviewed, with performance metrics calculated using SWOMSE functions.

Constant catch MPs with catch at or below OM maximum sustainable yield performed as expected. Median spawning biomass approached the equilibrium biomass level associated with the constant removals, while the distribution of biomass showed reasonable variation (**Figure 9**). For the constant TAC = 10.08 kt (MSY), the lower percentiles of biomass appeared to approach zero, which is to be expected in replicates where a string of negative log-normal process errors occur, especially when the auto-correlation is high as in OM 1 ($AC \approx 0.8$). All constant catch MPs with TAC > MSY eventually crashed with 100% probability, as expected, with the TAC = 15.12 kt MP taking longer to crash than the TAC = 20.06 kt MP. Spawning biomass under all TAC ≥ MSY MPs smoothly approached 0 instead of crashing hard (**Figure 9**), similar to the simulated equilibria (**Figure 3**). The smooth decline was a function of the catch scaling to 99% of vulnerable biomass, creating an optimistically biased (i.e., longer) "time-to-fail".

According to average catch performance metrics STC() and LTC() (see **Table B1**), realised catches appear to be negatively biased by about 7% - 15% compared to the TAC (**Table 2**). This is expected for the TACs that are greater than or equal to MSY, as the stock crashes, but was not expected for the TACs below MSY, given that there was no implementation error. Based on the CalcMPDynamics() function, discarding is not being counted against the TAC, so it is unclear why this negative bias would persist.

There does not appear to be a significant issue from the edge cases where catch scaling needs to be applied. While scaling catches above 99% of vulnerable biomass produces more optimistic outcomes than allowing MPs to take all the available biomass, the effect is only applied when MPs are already fishing very hard (i.e., at target harvest rates above 99%). Any MP that triggers the catch scaling in almost every year and replicate (e.g., the TAC > MSY MPs above) would presumably be rejected by fishery managers based on a range of metrics. However, there may be some middle ground that extirpates the stock at a more realistic rate while avoiding exceptions that create model defects, such as biased performance metrics, or programmatic failures such as crashed software. After all, some stocks (e.g. Northern Cod in Canada) crash pretty quickly

3. Conclusion

In general, the SWOMSE and related packages are good examples of applied scientific computing software developed for testing management procedures. There are some style issues noted by the static code analysis, which the developers can choose to address or not. We are confident that after the Major Recommendations and code corrections are implemented, the SWOMSE and MSEtool packages will help identify precautionary management procedures suitable for application to the North Atlantic transboundary Swordfish fishery.

4. Literature Cited

Pham, T. M. T. and Yang, J. (2020). The secret life of commented-out source code. In Proceedings of the 28th International Conference on Program Comprehension, pages 308–318.

Pope, J. (1972). An investigation of the accuracy of virtual population analysis using cohort analysis. ICNAF Research Bulletin, 9(10):65–74.

Seber, G. (1997). Estimation of Animal Abundance. Oxford University Press, 2nd edition.

Acknowledgements

This work was carried out under the provision of the ICCAT Science Envelope and the ICCAT – European Union Grant Agreement No. SI2.819116 - Strengthening the scientific basis for decision-making in ICCAT, and the ICCAT-US Data Fund.

Table 1. Results of the unit test of the calcF() function.

True F	Catch	NR Iterations	Estimated F	Difference in catch	Relative error in F
1.00E-04	1.51E+01	0	1.00E-15	15.071102	1
0.001	1.51E+02	0	1.00E-15	150.696016	1
0.01	1.51E+03	50	0.00999992	0.01564414	8.34E-06
0.1	1.49E+04	50	0.09999913	0.15999214	8.71E-06
0.5	7.14E+04	50	0.49999476	0.87957212	1.05E-05
1	1.35E+05	50	0.99998696	1.96016628	1.30E-05
2	2.45E+05	50	1.99996105	4.72247575	1.95E-05
10	6.44E+05	50	9.99828557	45.7675651	0.00017144

Table 2. Realised short-term average catch (2018 – 2027) and long-term average catch (2208 – 2217) as calculated by the STC() and LTC() metrics, under the five fixed TAC CMPs. All quantities are in kilotonnes, short-term and long-term columns show the mean values and range across 48 simulation replicates.

Fixed TAC	Short-term average catch	Long-term average catch
2.52	2.35 (2.27 – 2.41)	2.41 (2.36 – 2.45)
5.04	4.68 (4.54 – 4.81)	4.81 (4.70 – 4.88)
10.08	9.28 (9.01 – 9.52)	0.79 (0.00 – 0.96)
15.12	13.68 (8.37 – 14.13)	0.00
20.16	14.42 (5.08 – 18.85)	0.00

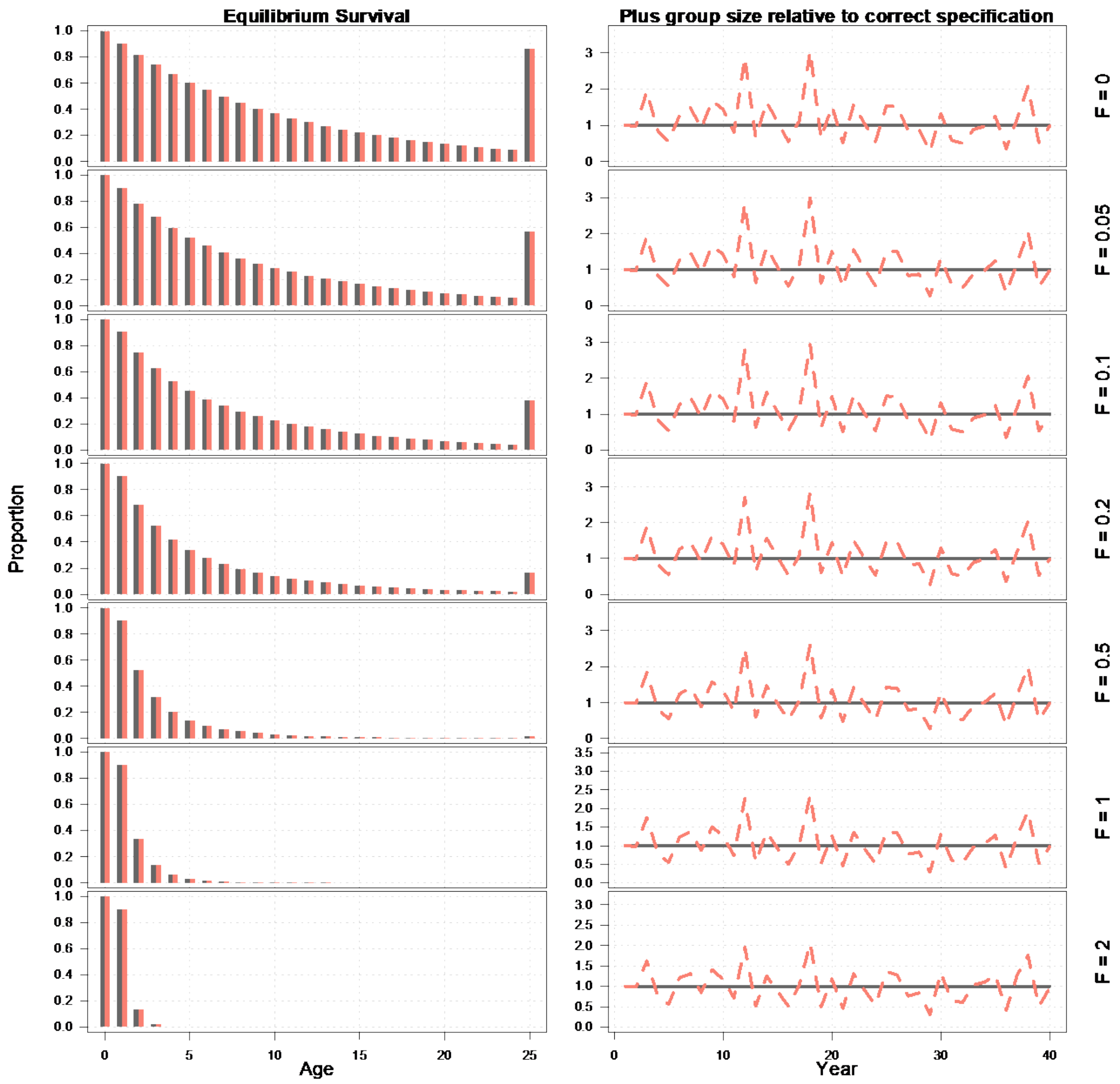


Figure 1. A comparison of equilibrium survival (left) and plus group population dynamics (right) specifications for 7 different levels of constant fishing mortality under the MSEtool coded specification (pink) and suggested specification (grey). The right hand column is a relative difference of the two specifications under a simulated time-varying recruitment, with the MSEtool plus group numbers (pink) divided by the corrected plus group numbers (grey).

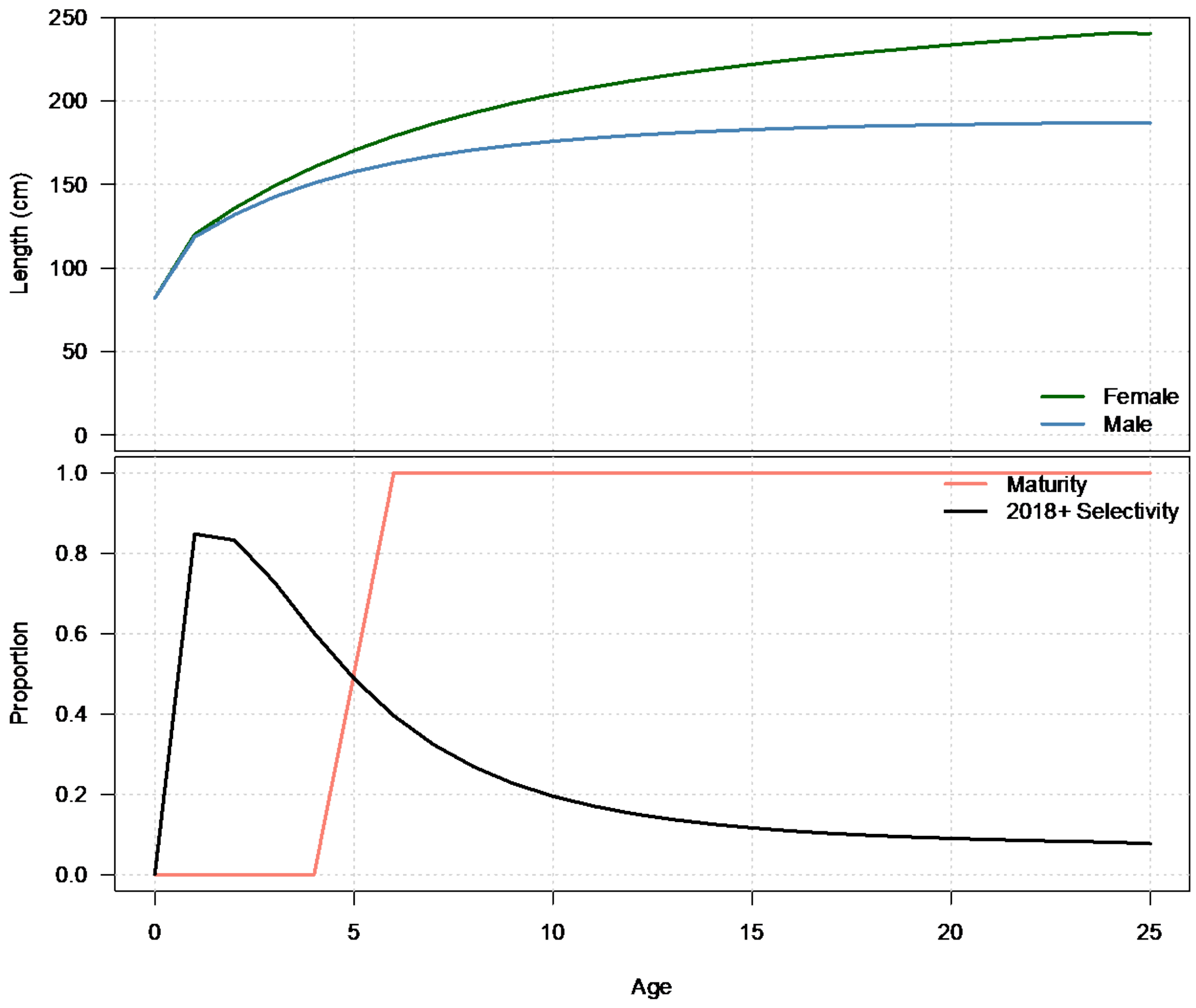


Figure 2. Length-at-age (top) for males and females in the stock-synthesis model conditioning Operating Model 1, and projection period fleet-aggregated selectivity- and maturity-at-age (bottom) for the conditioned MSEtool OM 1.

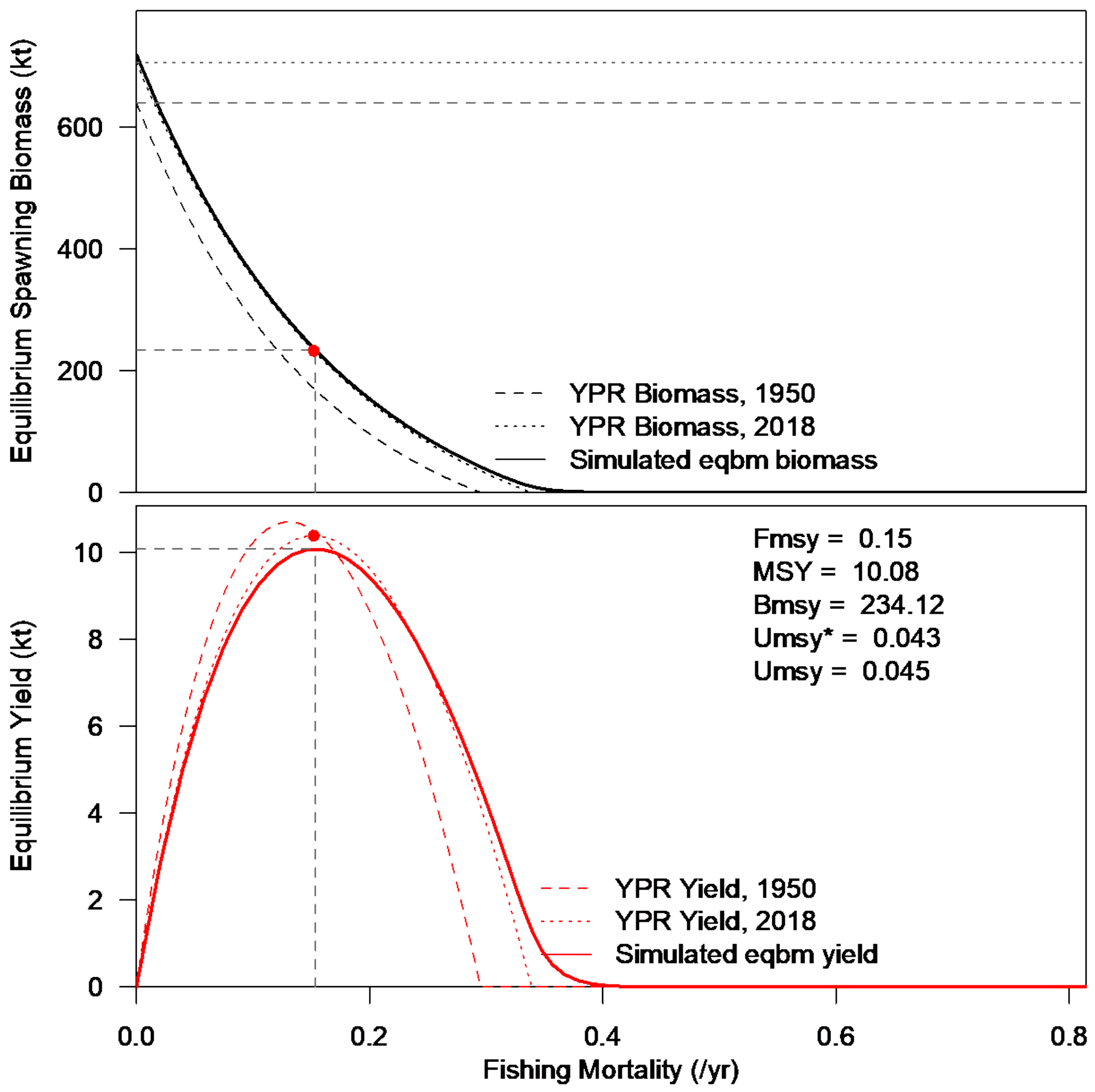


Figure 3. Equilibrium biomass (top) and yield (bottom) curves for OM 1 found via YPR analysis assuming model pars in 1950 and 2018, and via simulation of a grid of constant harvest rate MPs. Red points in both panels show the reference points reported in the MSE object, while dashed line segments joining the axes to the curves show the estimated reference points from the simulated curves. The horizontal dashed line at the top of the biomass panel shows OM SSB0. Umsy was calculated as operating model MSY/Bmsy, and the Umsy* was calculated as simulation MSY/Bmsy.

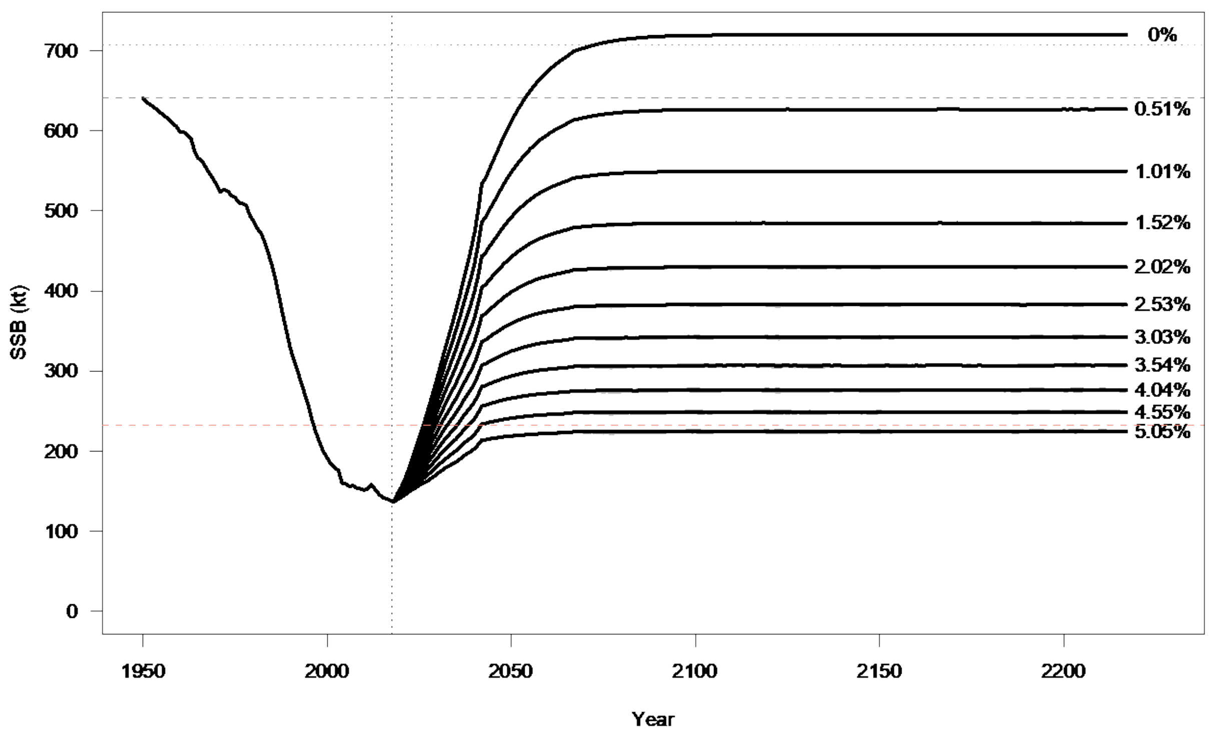


Figure 4. Spawning stock biomass trajectories for OM 1 with deterministic recruitment under constant harvest rates between 0% and 5%. The vertical dotted line shows the beginning of the projection period, and the horizontal dashed lines show unfished spawning stock biomass (grey) and B_{MSY} (pink).

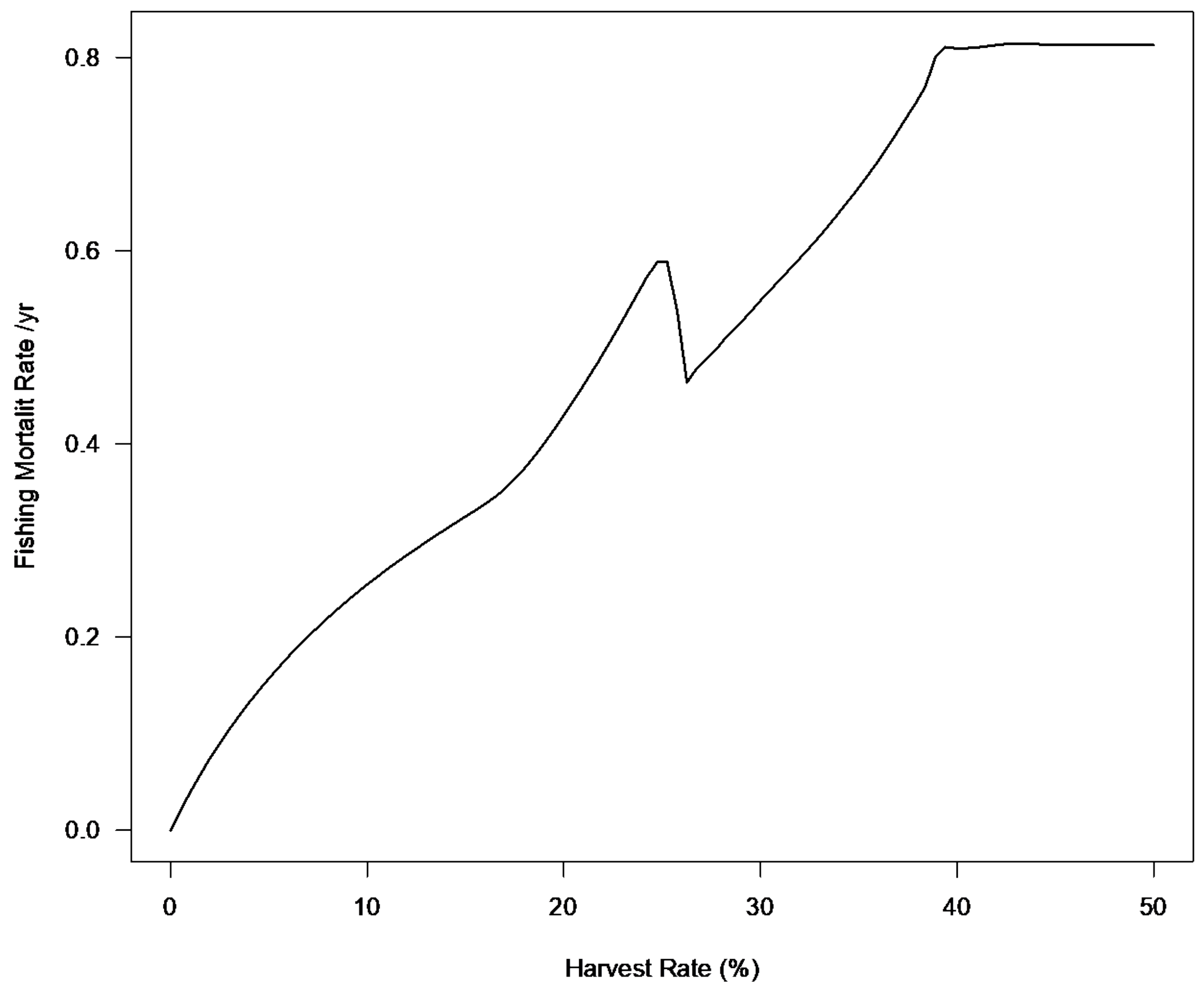


Figure 5. Median realised fishing mortality rates in the last 2 years of the 200 year projection (vertical axis) resulting from applying fixed target harvest rates as a percentage of spawning biomass (horizontal axis) in OM 1 with deterministic recruitment.

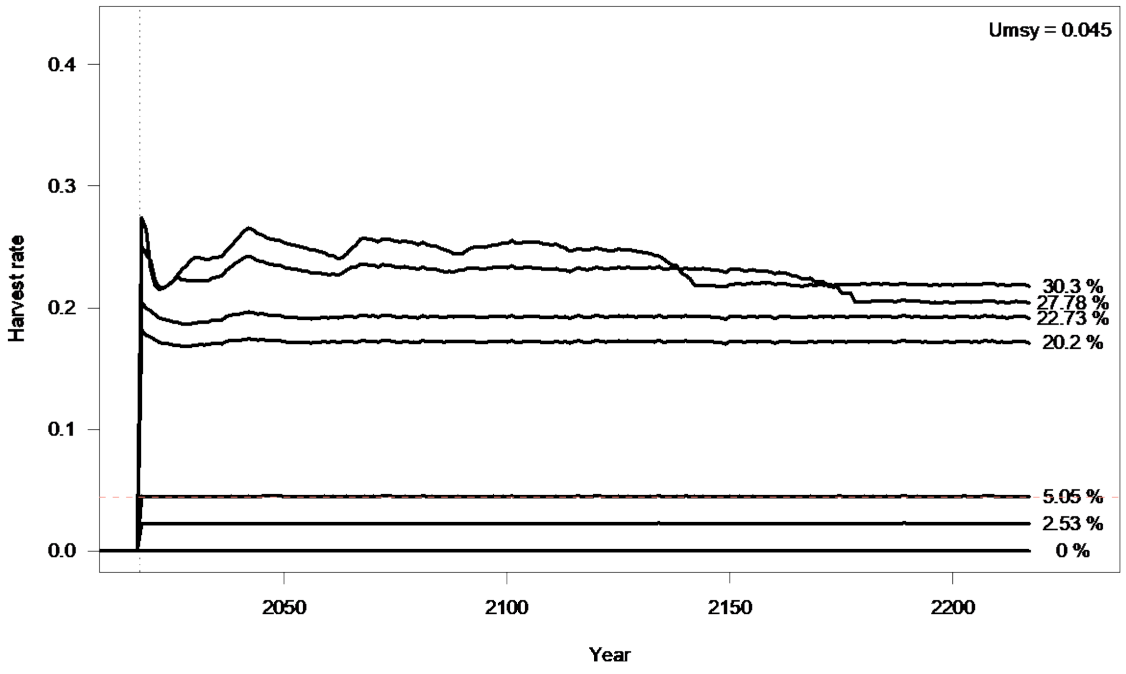


Figure 6. Realised harvest rates in the projection period when applying a perfect information, fixed harvest rate MP to OM 1. Each line represents the median realised harvest rate under a single fixed harvest rate applied to the spawning stock biomass known with perfect information, which is indicated by the line labels on the right. The optimal harvest rate, as a proportion of spawning biomass, based on 2018 weigh-at-age and selectivity is shown as a pink dashed line at 4.5%.

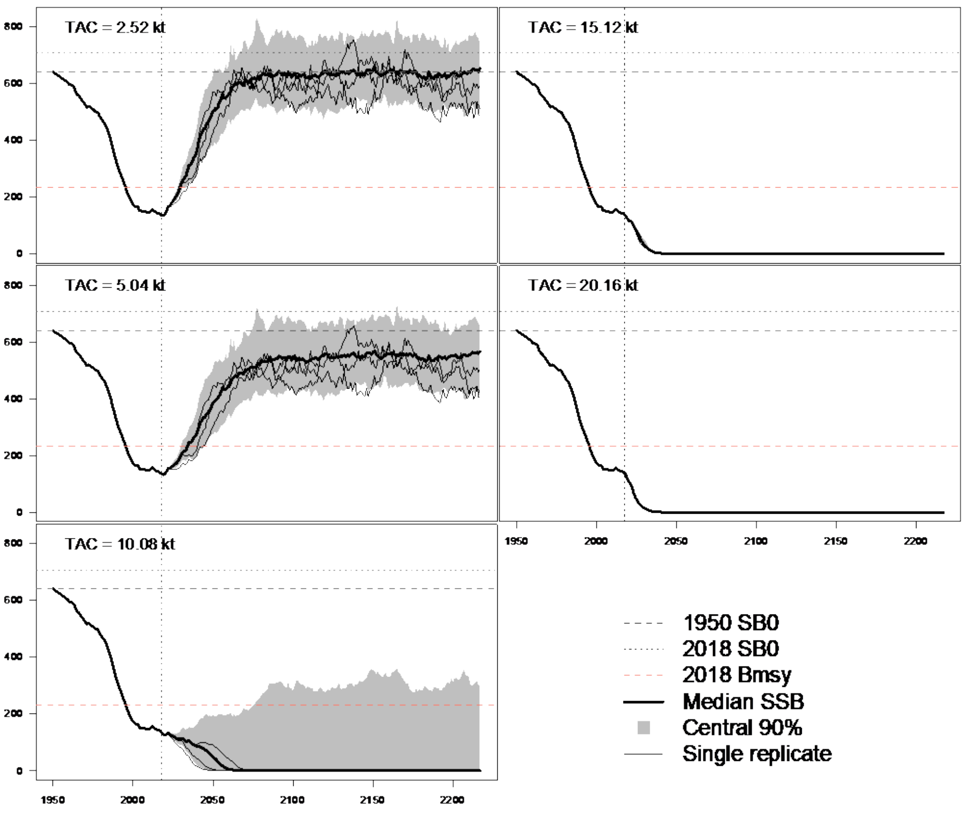


Figure 7. Spawning stock biomass simulation envelopes under 5 fixed TAC MPs.

Appendix A

Static Code Analyses

Static code analysis involves applying automated tools that analyse code for common issues. The most common static code analyses are "linters", which are used for making sure code adheres to a common programming style, but other applications include logic checking, and code dependency mapping.

A.1 Inter-package and intra-package function dependencies

We applied the DependenciesGraph R package[5] to determine the dependencies of the SWOMSE package (Robert 2016), such as the openMSE package. As advertised, SWOMSE depends on the openMSE package (Figure A1), which is an umbrella/meta-package containing the MSEtool, DLMtool, and SAMtool packages developed by Blue Matter Science (Figure A2).

For function dependencies within each package, the SWOMSE package acts as a wrapper for openMSE (Figure A3), with the main purpose of providing functions for SWO specific tasks that do not fit within the general MSEtool/openMSE framework. This is a completely acceptable way of meeting unique requirements for SWO that are too path-specific for the general openMSE suite of packages.

The main workhorse function of the SWOMSE package is the SWO_SS2OM() function that conditions the openMSE operating model history from the SWO Stock Synthesis 3 (SS3) assessment models. The SWOMSE package contains other functions (Figure A3), such as example CMPs and performance metrics, but the conditioning function forms the foundation of the population dynamics for the SWO MSE, and therefore is the most important function reviewed. Further, the conditioning process simplifies the structure of the SWO SS3 model from sex-structured and multi-fleet to the openMSE combined sex and single-fleet operating model. Assumptions for this reduction in dimensionality require careful consideration and should be clearly stated, since they have potential to affect MP performance.

Review of the openMSE package is restricted to selected functions from the MSEtool package. The DLMtool and SAMtool packages provide data-limited and data-moderate/rich stock assessment model functions that are not utilized at this stage for SWO MSE, and therefore were not reviewed. The SAMtool and DLMtool may be used as examples for CMP development, but are unlikely to be applied for the management of the SWO stock. The SWO related functions from the MSEtool package were determined by static code analysis (Figure A4), and 'following our nose' through from the initial call of the runMSE() function that starts a closed loop feedback simulation.

A.2 Flowcharts of package functions

We applied the flow R package[6] to generate flowchart diagrams for all functions in the SWOMSE and MSEtool pacakages (Fabri 2021). We did not find any logical errors or inconsistencies in the flow-charts, but they did assist in providing improved readability of the code-base. Flow charts were quite large, so we do not include all of them here, but an example is shown in Figure A5. From the flow charts, we identified the functions that are called when a user initiates SWO MSE simulations using the runMSE() function, which were reviewed line-by-line (**Table B1** and **B2**)

A.3 Linting for programming style

We applied the lintr package to the SWOMSE and MSEtool packages (Hester *et al.* 2020). We have attached a summary of the linting output for SWOMSE (**Table A.1**). The linting output for MSEtool is much longer, given the higher number of functions, so an excerpt is included (**Table A.2**) and some raw data is supplied in Appendix C.

There were about 1400 exceptions in the SWOMSE package, and about 32000 exceptions in the MSEtool package. Most exceptions are simple style issues that can likely be ignored, such as line-lengths and spacing, but would have readability benefits. However, there was a non-trivial amount of commented out code. Commented out code is common during development phases of software, often related to debugging or deprecated functions that are no longer actively used by the software (Pham and Yang 2020). The presence of commented out code is controversial, but largely benign, and also primarily affects readability of code. However, there is a risk of errors if collaborators or future developers

[5] https://github.com/datastorm-open/DependenciesGraphs
[6] https://github.com/moodymudskipper/flow

uncomment the code without understanding why it was originally commented. Given that the openMSE package is version controlled on github, there is no reason for commented out code to remain in the R scripts for production versions of the openMSE package.

A.4 Literature cited

Fabri, A. (2021). flow: View and Browse Code Using Flow Diagrams. R package version 0.0.2.

Hester, J., Angly, F., and Hyde, R. (2020). lintr: A ’Linter’ for R Code. R package version 2.0.1.

Pham, T. M. T. and Yang, J. (2020). The secret life of commented-out source code. In Proceedings of the 28th International Conference on Program Comprehension, pages 308–318.

Robert, T. (2016). DependenciesGraphs: Dependencies visualization between functions and environments. R package version 0.3.

Table A1. Summary of the lintr package output when applied to the functions in the SWOMSE package.

Filename	Linter	No of issues
R/functions.R	function_left_parentheses_linter	1
R/functions.R	infix_spaces_linter	6
R/functions.R	line_length_linter	1
R/functions.R	object_name_linter	9
R/functions.R	object_usage_linter	5
R/functions.R	single_quotes_linter	1
R/MPs.R	commas_linter	2
R/MPs.R	infix_spaces_linter	9
R/MPs.R	object_name_linter	33
R/MPs.R	single_quotes_linter	3
R/MPs.R	spaces_left_parentheses_linter	1
R/MPs.R	trailing_blank_lines_linter	2
R/OMfit_diagnostics.R	assignment_linter	3
R/OMfit_diagnostics.R	commas_linter	13
R/OMfit_diagnostics.R	infix_spaces_linter	86
R/OMfit_diagnostics.R	object_name_linter	7
R/OMfit_diagnostics.R	object_usage_linter	12
R/OMfit_diagnostics.R	paren_brace_linter	5
R/OMfit_diagnostics.R	pipe_continuation_linter	1
R/OMfit_diagnostics.R	seq_linter	1
R/OMfit_diagnostics.R	single_quotes_linter	10
R/OMfit_diagnostics.R	spaces_left_parentheses_linter	6
R/PMs.R	commas_linter	2
R/PMs.R	object_name_linter	12
R/PMs.R	object_usage_linter	6
R/Roxygen_OMs.r	single_quotes_linter	289
R/Roxygen_OMs.r	trailing_blank_lines_linter	2
R/Roxygen_OMs.r	trailing_whitespace_linter	316
R/Shiny.R	commas_linter	1
R/Shiny.R	object_name_linter	2
R/Shiny.R	single_quotes_linter	1
R/SWO_SS2OM.R	assignment_linter	1
R/SWO_SS2OM.R	commas_linter	65
R/SWO_SS2OM.R	cyclocomp_linter	1
R/SWO_SS2OM.R	function_left_parentheses_linter	1
R/SWO_SS2OM.R	infix_spaces_linter	190
R/SWO_SS2OM.R	line_length_linter	11
R/SWO_SS2OM.R	object_name_linter	178

R/SWO_SS2OM.R	object_usage_linter	55
R/SWO_SS2OM.R	open_curly_linter	2
R/SWO_SS2OM.R	paren_brace_linter	6
R/SWO_SS2OM.R	pipe_continuation_linter	9
R/SWO_SS2OM.R	seq_linter	6
R/SWO_SS2OM.R	single_quotes_linter	24
R/SWO_SS2OM.R	spaces_inside_linter	4
R/SWO_SS2OM.R	spaces_left_parentheses_linter	64
R/SWO_SS2OM.R	trailing_blank_lines_linter	2

Table A2. An excerpted summary of the lintr package output when applied to the functions in the popdyn.R and runMSE.R scripts in the MSEtool package.

File Path	**Linter name**	**No. of Issues**
R/popdyn.R	commas_linter	205
R/popdyn.R	commented_code_linter	9
R/popdyn.R	cyclocomp_linter	2
R/popdyn.R	infix_spaces_linter	396
R/popdyn.R	line_length_linter	182
R/popdyn.R	object_name_linter	296
R/popdyn.R	object_usage_linter	11
R/popdyn.R	single_quotes_linter	2
R/popdyn.R	spaces_inside_linter	2
R/popdyn.R	spaces_left_parentheses_linter	52
R/runMSE.R	assignment_linter	2
R/runMSE.R	commas_linter	405
R/runMSE.R	commented_code_linter	17
R/runMSE.R	cyclocomp_linter	2
R/runMSE.R	function_left_parentheses_linter	1
R/runMSE.R	infix_spaces_linter	505
R/runMSE.R	line_length_linter	155
R/runMSE.R	object_name_linter	464
R/runMSE.R	object_usage_linter	29
R/runMSE.R	open_curly_linter	1
R/runMSE.R	seq_linter	1
R/runMSE.R	single_quotes_linter	35
R/runMSE.R	spaces_inside_linter	1
R/runMSE.R	spaces_left_parentheses_linter	35
R/runMSE.R	trailing_blank_lines_linter	20

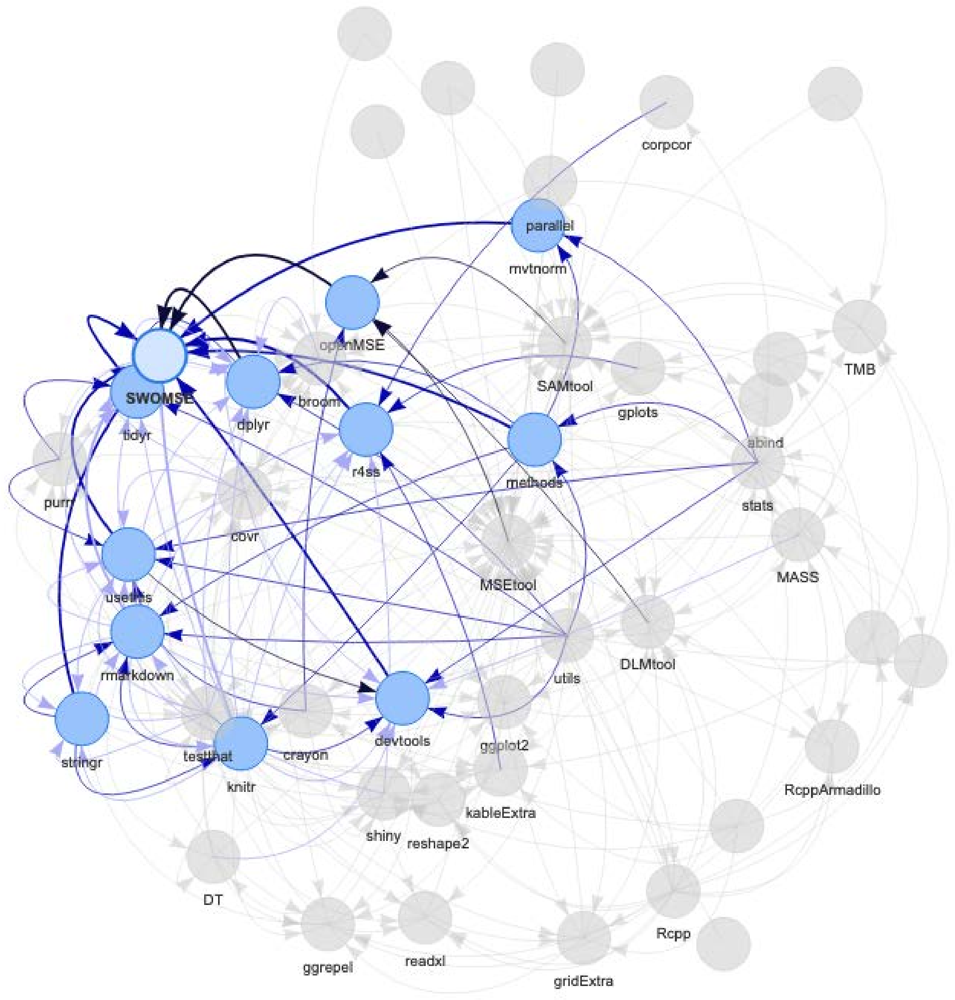


Figure A1. Package dependency directed graph for the SWOMSE package (indicated by the thicker border on blue circle). Arrow direction indicates contribution (i.e., an arrow from r4ss to SWOMSE indicates that SWOMSE depends on r4ss). [SAM – read package manual for description of the legend here)

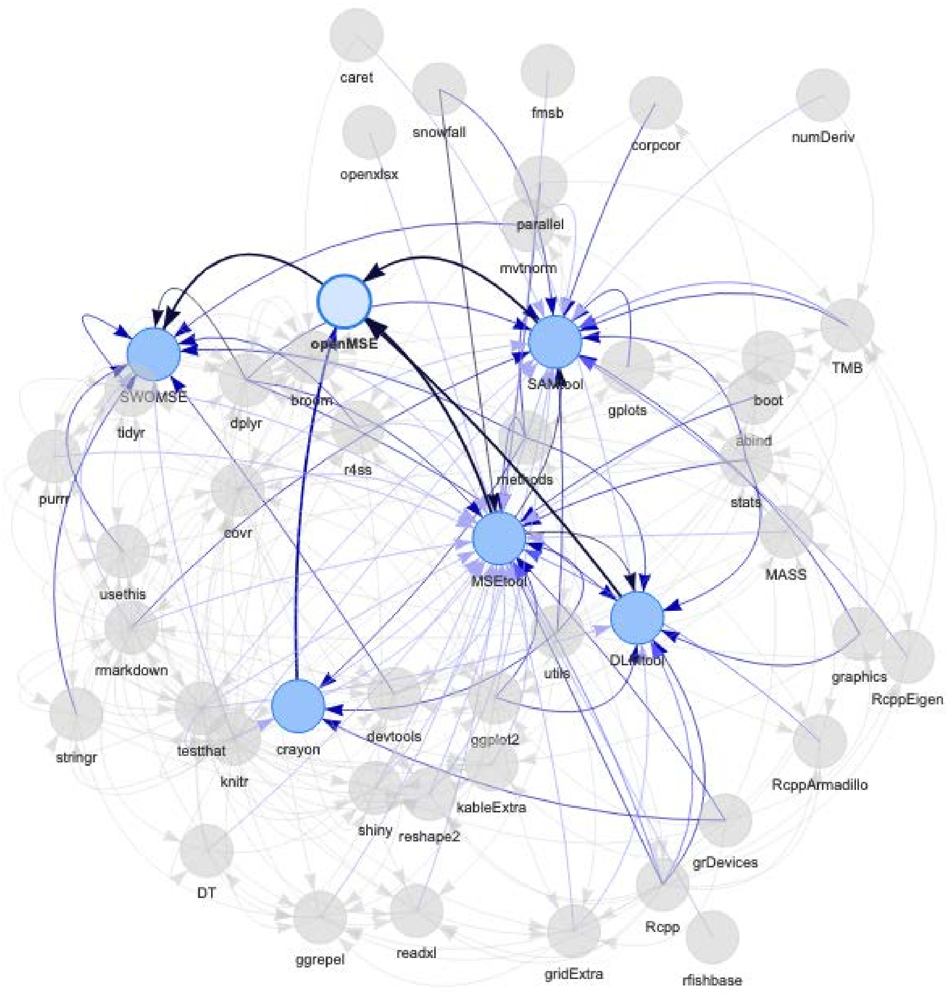


Figure A2. Package dependency directed graph for the openMSE package (indicated by the thicker border on blue circle). This is the same graph as in Figure 1, but with the openMSE package as the focus. Arrow direction indicates contribution (i.e., an arrow from openMSE to SWOMSE indicates that SWOMSE depends on openMSE). [SAM – read package manual for description of the legend here]

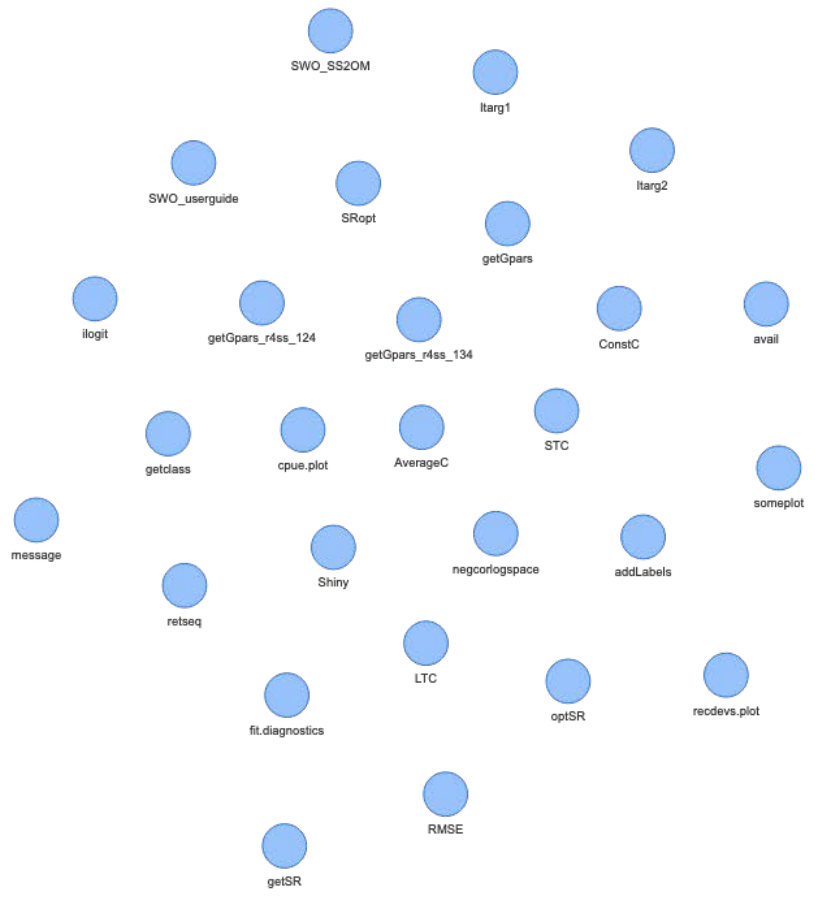


Figure A3. Function dependency graph for the SWOMSE package. There are no arrows here because the functions in the SWOMSE package do not depend on the other functions, as SWOMSE provides NSWO MSE specific functionality for the openMSE package.

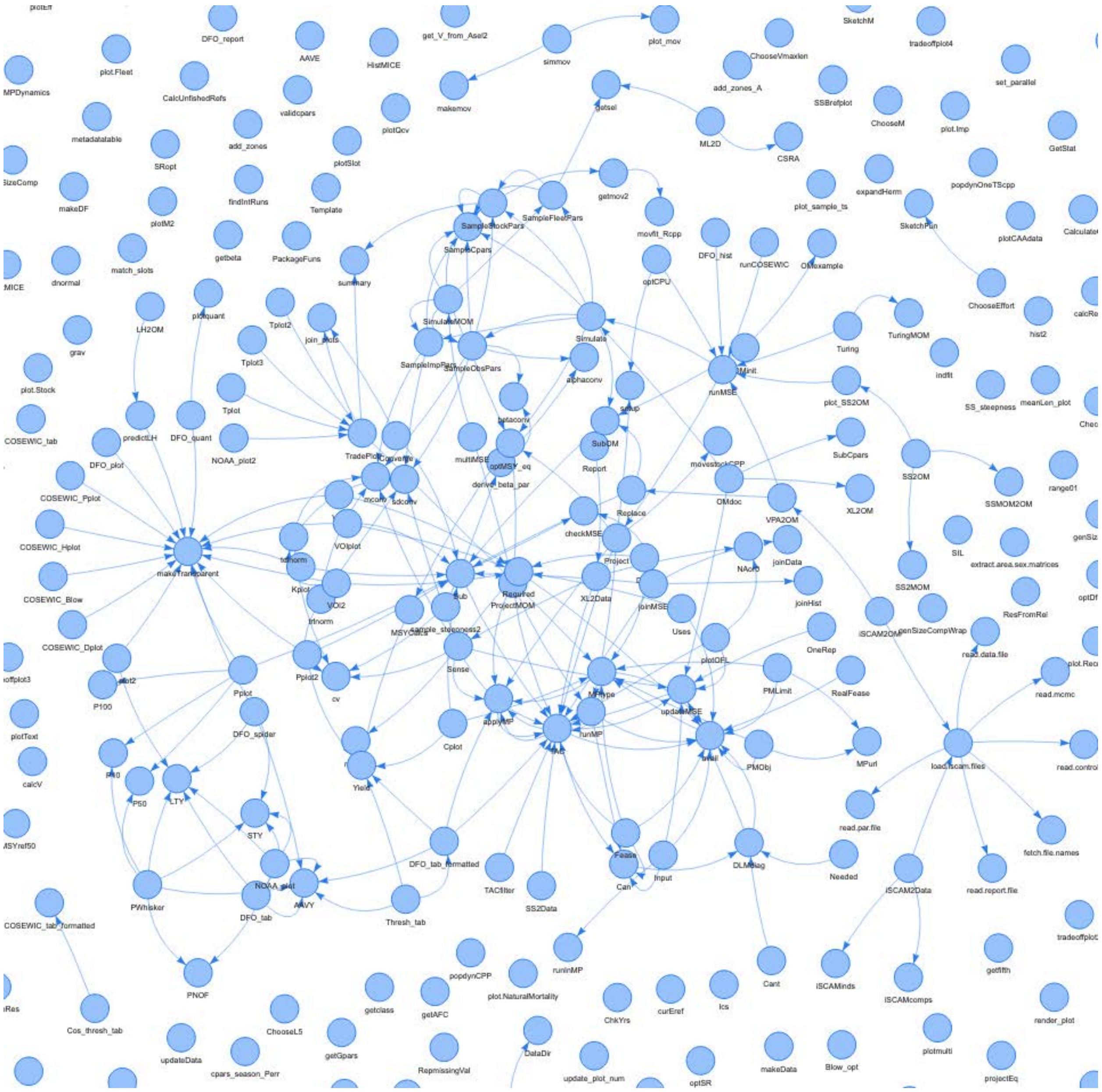
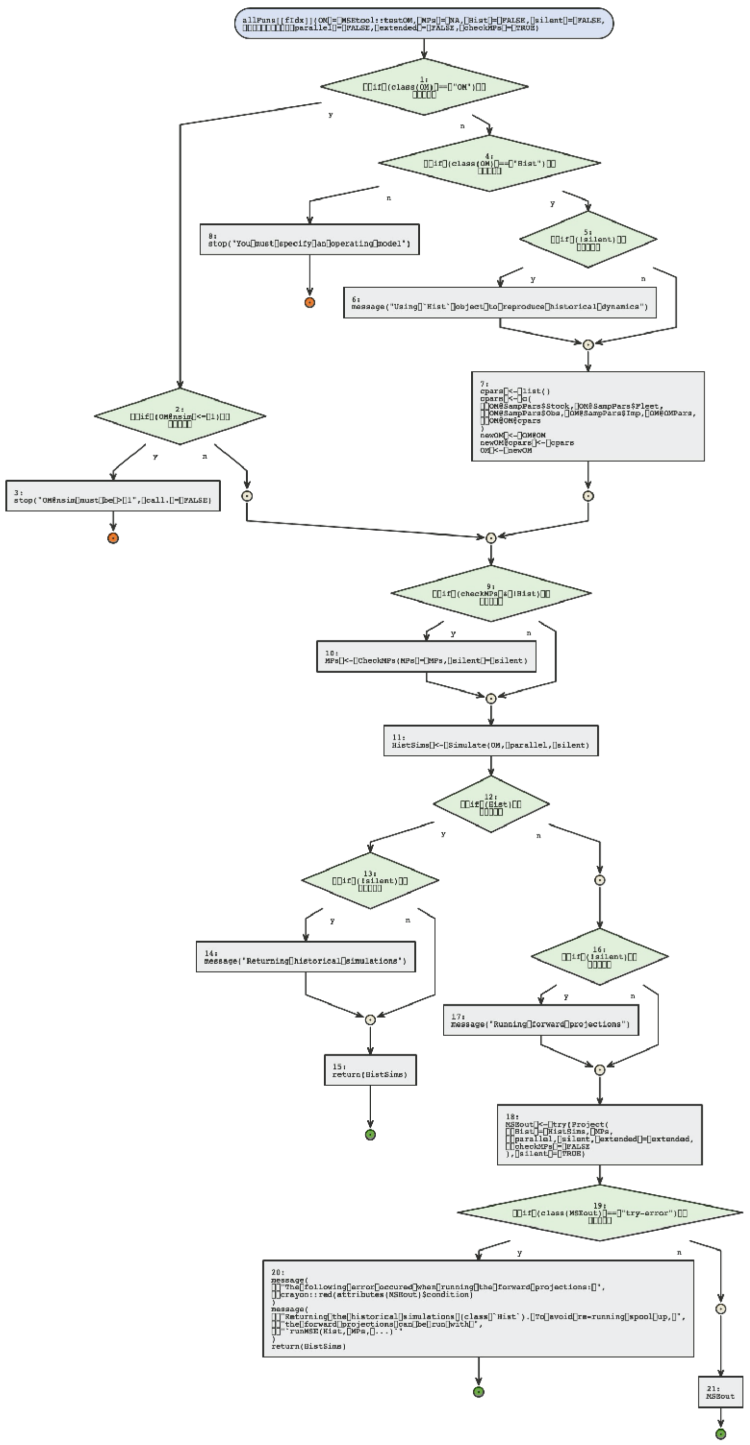


Figure A4. A cropped image of the function dependency graph for the openMSE package. Arrows indicate dependency (e.g., an arrow from the circle labeled runMSE to the circle labeled Simulate indicates that the runMSE() function calls the Simulate() function.



```
allFuns[[fIdx]](OM□=□MSEtool::testOM,□MPs□=□NA,□Hist□=□FALSE,□silent□=□FALSE,
□□□□□□□□□□□□parallel□=□FALSE,□extended□=□FALSE,□checkMPs□=□TRUE)

1:
□□if□(class(OM)□==□"OM")□□
□□□□□

2:
□□if□(OM@nsim□<=□1)□□
□□□□□

3:
stop("OM@nsim□must□be□>□1",□call.□=□FALSE)

4:
□□if□(class(OM)□==□"Hist")□□
□□□□□

5:
□□if□(!silent)□□
□□□□□

6:
message("Using□`Hist`□object□to□reproduce□historical□dynamics")

7:
cpars□<-□list()
cpars□<-□c(
□□OM@SampPars$Stock,□OM@SampPars$Fleet,
□□OM@SampPars$Obs,□OM@SampPars$Imp,□OM@OMPars,
□□OM@OM@cpars
)
newOM□<-□OM@OM
newOM@cpars□<-□cpars
OM□<-□newOM

8:
stop("You□must□specify□an□operating□model")

9:
□□if□(checkMPs□&□!Hist)□□
□□□□□

10:
MPs□<-□CheckMPs(MPs□=□MPs,□silent□=□silent)

11:
HistSims□<-□Simulate(OM,□parallel,□silent)

12:
□□if□(Hist)□□
□□□□□

13:
□□if□(!silent)□□
□□□□□

14:
message("Returning□historical□simulations")

15:
return(HistSims)

16:
□□if□(!silent)□□
□□□□□

17:
message("Running□forward□projections")

18:
MSEout□<-□try(Project(
□□Hist□=□HistSims,□MPs,
□□parallel,□silent,□extended□=□extended,
□□checkMPs□=□FALSE
),□silent□=□TRUE)

19:
□□if□(class(MSEout)□==□"try-error")□□
□□□□□

20:
message(
□□"The□following□error□occured□when□running□the□forward□projections:□",
□□crayon::red(attributes(MSEout)$condition)
)
message(
□□"Returning□the□historical□simulations□(class□`Hist`).□To□avoid□re-running□spool□up,□",
□□"the□forward□projections□can□be□run□with□",
□□"`runMSE(Hist,□MPs,□...)`"
)
return(HistSims)

21:
MSEout
```

Figure A5. An example logical flow-chart generated by the flow package for the runMSE() function in the MSEtool package. Square boxes are an inserted space character meant to preserve indenting, and are unable to be removed without affecting formatting.

Line-by-line review

Table B3. Selected SWOMSE package functions that were reviewed line-by-line. Function names are appended with parentheses (e.g., foo()), and other R variable are in italics. Line numbers are shown in square brackets in the comments and recommendations.

File Path	Function name(s)	Line No	Comments	Recommendations
R/SWO_SS2OM.R	SWO_SS2OM()	33	The SWO_SS2OM() function conditions the openMSE operating model from the sex-structured, multi-fleet SS3 assessment models. The conditioning process simplifies the model structure, collapsing the sex-structured and multi-fleet SS3 model into a combined sex and single-fleet operating model.	Improve documentation of simplification in the TSD. Remove deprecated fleet aggregation [482 – 525], and sensitivity test the exploitation rate weightings.
R/PMs.R	STC(), LTC()		Both the STC() and LTC() functions are example performance metrics that calculate the short term and long term average catch, respectively. They call the ChkYrs(), calcProb(), and calcMean() functions from the MSEtool package, which are reviewed below. STC() and LTC() are tested in the evaluation of the constant catch MPs in this review.	

Table B4. Selected MSEtool package functions reviewed line-by-line. Function names are appended with parentheses (e.g., foo()), and other R variable are in italics. Line numbers are shown in square brackets in the comments and recommendations.

File Path	**Function name**	**Line No**	**Comments**	**Recommendation**
R/runMSE.R	Simulate()	4	The Simulate() function sets up the historical operating model population dynamics, using outputs from the SWO_SS2OM() function, and simulates the historical period of the simulations, which are passed to the Project() function for CMP testing. As per functional programming principles, most mathematical expressions are included in functions that are called by Simulate(), so detailed review is saved for those functions. The parameter sampling functions are not reviewed in detail, as the SWO stock has a single history across simulation replicates with fixed parameter values [67 - 123]. While the simulated combined index of abundance will have sampled observation model parameters, these are sampled at another location after post-fitting the parameters to the OM.	Simplify equilibrium survival calculations [145 & 175]. They are mathematically accurate, but perform extraneous calculations that could be removed to improve code execution speed.
	Project()	910	The Project() function takes a *Hist* object generated by Simulate(), and projects the closed loop feedback simulations into the future one time-step at a time. Similar to Simulate(), most mathematical expressions are contained in other called functions.	
	runMSE()	1556	The runMSE() function is the external package function that runs an MSE given an OM, a set of MPs, and other user-defined MSE settings (e.g., observation and implementation models). This function wraps the CheckMPs(), Simulate() and Project() functions.	
src/popdynCPP.cpp	popdynOneTScpp()	19	The popdynOneTScpp() function advances the population's numbers-at-age and mortality forward by a single time-step.	Correct plus-group dynamics [35] (derivation given in main text)

File Path	**Function name**	**Line No**	**Comments**	**Recommendation**
	movestockCPP()	57	The movestockCPP() function applies the age-dependent movement model. At each time-step, movestockCPP() takes the arrays for the numbers-at-age and the Markov movement matrices for each age class, and applies the movement matrix.	There may be marginal performance gains by using a matrix multiplication library such as Eigen for applying Markov movement models, rather than nested loops [65-71]. Gains would likely be small, given the small number of spatial strata used for SWO.
	popdynCPP()	123	The popdynCPP() function is used in the historical period to advance the population dynamics forward in multiple time-steps when closed loop feedback simulation is unnecessary (i.e., the fishing mortality rates are fixed inputs). The popdynCPP() function uses the popdynOneTSCPP() function for annual population dynamics calculations, the moveStockCPP() function for applying movement, and calculates recruitment for the specified stock-recruitment model.	
R/Data_Functions.R	applyMP()	2655	The applyMP() function organises list/array objects for the application of an MP, and then applies the MP.	
	runMP()	744	The runMP() function acts as a wrapper for the applyMP() function for use in MP testing. This function is only used for testing CMPs outside of closed loop simulations.	
R/popdyn.R	optMSY_eq()	129	A wrapper for the MSYcalcs() function to optimise yield as a function of fishing mortality.	
	MSYcalcs()	176	Calculates equilibrium yield, given an input *logF* value and the stock life history parameters. The	Same recommendation as the Simulate() function for the survival calculations. They

File Path	Function name	Line No	Comments	Recommendation
			MSYcalcs() function was tested in the equilibrium simulations in this review.	are correct but involve more mathematical operations than necessary.
	CalcMPDynamics()	407	The calcMPDynamics() function enables progressing population dynamics forward for a single time step, based on the TAC recommendation from a management procedure. Its primary purpose is to convert a TAC into a fishing mortality rate by solving the Baranov catch equation using the calcF() function.	
	calcF()	866	The calcF() function solves the Baranov catch equation for fishing mortality rates using a Newton-Raphson method. This is a standard approach used for both stock assessments conditioned on catch, or generating Fs from TACs in simulation. The mathematics of the calcF() function are correct, but the formulation with no scaling of steps between iterations could lead to non-convergence (e.g, an infinite loop or overshoot) at very high or very low catches. This is a common problem with Newton-Raphson optimization in general, and can be addressed by scaling step-sizes by a fixed fraction, or in some cases by an automated optimization of step sizes. We explore whether this behaviour exists for calcF() in a unit test.	Scale step sizes when iteratively applying Jacobian by a half to avoid non-convergence.
R/Data_make_update.R	makeData()	3	This function sets up the Data object that is provided to MP functions to generate TACs at each projection time step. The hyperstability and hyperdepletion equations [32, 41, 50] are accurate.	
	updateData()	202	Updates the Data object in each projection year. This mostly involves re-arranging data structures. There are some observation index calculations for biomass indices with possible	

File Path	**Function name**	**Line No**	**Comments**	**Recommendation**
			hyperstability/hyperdepletion [248, 282, 312], all of which are accurate.	
	simCAL()	202	Generates catch-at-length data and summary statistics for each time step. Wraps the genSizeCompWrap() function. Summary statistics are length-at-first-capture [619], mean length [626], modal length [630] and the mean length above the modal length [633]. This function was unit tested in this review.	
	genSizeCompWrap()	677	Generates catch-at-length data. Wraps genSizeComp2(), which is a C++ function. The inner function is reviewed and unit tested.	There is some commented out code for another method to generate size composition data, which we recommend removing if it is not used.
	AddRealData()	750	The AddRealData() function overwrites the historical period data with real data from the conditioning assessment. There is nothing mathematical to review here.	We recommend visual checks of this process by plotting the data in the Data object against the real data.
src/genLenComp.cpp	genSizeComp2()	176	Generates size composition from numbers at age, selectivity, and a catch-at-length effective sample size. Standard approach, where the normal distribution of length-at-age [193] is modulated by size-selectivity [196], and used to sample the catch-at-age [199] via a multinomial distribution. Samples by age and length are combined and scaled to the total sample size [205-208], and returned as a matrix [215].	Compare simulated historical length comps against real data, aggregated using the same rules as the selectivity aggregation.
R/Misc_Exported.R	Required()	395	The Required() function checks that all data required by an MP are present in the Data object. This is a good robustness function for CMP developers who may be less familiar with the MSEtool package.	

File Path	Function name	Line No	Comments	Recommendation
	Setup()	544	This function sets up parallel processing and exports functions to cluster nodes. If this function had an error, then the model would fail to run in parallel. The rest of the functionality is in exported objects reviewed above and below.	
	updateMSE()	605	Allows backwards compatibility with older DLMtool and MSEtool outputs.	
R/Misc_Internal.R	calcRecruitment()	290	Calculates recruitment in the projection period. The calcRecruitment() function distributes recruitment spatially based on an R0a parameter, which is the R0 value scaled to the fraction of unfished biomass in each area.	The recruitment calculations in the projection period are central to population dynamics and should be better documented.
	applyAC()	396	Standard application of auto-correlation to a sequence of random errors. It is correctly implemented.	

Appendix C

Code for simulation and unit tests

Code for the SWOMSE and MSEtool code review functions and plots can be found at the public bitbucket repository

https://bitbucket.org/lfr_code/swo_codereview/