Fisheries example integrating FLR

GMSE: an R package for generalised management strategy evaluation (Supporting Information 5)

A. Bradley Duthie[1,3], Jeremy J. Cusack[1], Isabel L. Jones[1], Jeroen Minderman[1], Erlend B. Nilsen[2], Rocío A. Pozo[1], O. Sarobidy Rakotonarivo[1], Bram Van Moorter[2], and Nils Bunnefeld[1]

[1] Biological and Environmental Sciences, University of Stirling, Stirling, UK [2] Norwegian Institute for Nature Research, Trondheim, Norway [3] alexander.duthie@stir.ac.uk (mailto:alexander.duthie@stir.ac.uk)

Integration and simulation with fisheries

Early development of management strategy evaluation (MSE) models originated in fisheries (Polacheck et al. 1999; Smith, Sainsbury, and Stevens 1999; Sainsbury, Punt, and Smith 2000). Consequently, fisheries-focused software for MSE has been extensively developed, including R libraries that focus on the management of species of exceptional interest, such as the Atlantic Bluefin Tuna (*Thunnus thynnus*) (ABFTMSE (https://github.com/ICCAT/abft-mse/tree/master/R_package/ABTMSE); Carruthers and Butterworth 2018b; Carruthers and Butterworth 2018a), and Indian Ocean Bigeye (*T. obesus*) and Yellowfin (*T. albacares*) Tuna (MSE-IO-BET-YFT (https://github.com/pjumppanen/MSE-IO-BET-YFT); Kolody and Jumppanen 2016). The largest of all such libraries is the Fisheries Library in R (FLR (http://www.flr-project.org/)), which includes an extensive collection of tools targeted for fisheries science. The FLR library has been used in over a hundred publications (http://www.flr-project.org/#publications) (recent publications include Jardim et al. 2018; Mackinson et al. 2018; Utizi et al. 2018), and includes an MSE framework (http://www.flr-project.org/doc/An_introduction_to_MSE_using_FLR.html) for evaluating different harvest control rules.

<<<<<<< HEAD As part of the ConFooBio (https://sti-cs.org/confoobio/) project, a central focus of GMSE is on simulating the management of animal populations of conservation interest, with a particular emphasis on understanding conservation conflict; further development of GMSE is expected to continue with this as a priority, further building upon the decision-making algorithms of managers and users to better understand how conflict arises and can be managed and mitigated. Hence, GMSE is not intended as a substitute for packages such as FLR (http://www.flr-project.org/), but the integration of these packages with GMSE could make use of GSME's current and future simulation capabilities, and particularly the genetic algorithm (). Such integration might be possible using the `gmse_apply` function, which allows for custom defined submodels () to be used within the GMSE framework, and with default GMSE submodels. Hence, GMSE might be especially useful for modelling the management of fisheries under conditions of increasing competing staleholder demands and conflicts. We do not attempt such an ambitious project here, but instead show how such a project could be developed through integration of FLR (http://www.flr-project.org/) and `gmse_apply`. ======= As part of the ConFooBio (https://sti-cs.org/confoobio/) project, a central focus of GMSE is on simulating the management of populations of conservation interest, with a particular emphasis on understanding conservation conflict; further development of GMSE is expected to continue with this as a priority, further building upon the decision-making algorithms of managers and users to better understand how conflict arises and can potentially be resolved. Hence, GMSE is not intended as a substitute for packages such as FLR (http://www.flr-project.org/), but the integration of these packages with GMSE could make use of GSME's current and future simulation capabilities, and particularly the genetic algorithm (). Such integration might be possible using the `gmse_apply` function, which allows for custom defined submodels ()

to be used within the GMSE framework, and with default GMSE submodels. Hence, GMSE might be especially useful for modelling the management of fisheries under conditions of increasing harvesting demands and stakeholder conflict. We do not attempt such an ambitious project here, but instead show how such a project could be developed through integration of FLR (http://www.flr-project.org/) and `gmse_apply` . >>>>>>> 366b9295e25f74c310c7185eab57ad3e19bfa83c

Here we follow a Modelling Stock-Recruitment with FLSR (http://www.flr-project.org/doc/Modelling_stock_recruitment_with_FLSR.html) example, then integrate this example with `gmse_apply` to explore the behaviour of a number of simulated fishers who are goal-driven to maximise their own harvest and a manager that aims to keep the fish stocks at a predefined target level. The core concept in GMSE is that manager can only incentivise fishers to harvest less or more by varying the cost of fishing (through e.g. taxes) given a set manager budget; please note that the manager cannot force the fisher to follow any policy. Based on the cost of fishing, the fisher can then given their own budget decide whether to invest in fishing or keep the budget. This concepts represents a nartural resource managmeent and conservation conflict, where one party aims to maximise their livelihood (fisher) and the other aims to keep a population at a sustainable level and prevent it from going extinct. Importantly, the manager does not have full control over fishers but can set policies to incentivise sustainable behaviour. We emphasise that this example is provided only as demonstration of how GMSE can potentially be integrated with already developed fisheries models, and is not intended to make recommendations for management in any population.

Integrating with the Fisheries Library in R (FLR (http://www.flr-project.org/))

The FLR toolset includes a series of pacakges (http://www.flr-project.org/#packages), with several tutorials (http://www.flr-project.org/doc/index.html) for using them. For simplicity, we focus here on a model of stock recruitment (http://www.flr-project.org/doc/Modelling_stock_recruitment_with_FLSR.html) to be used as the population model in `gmse_apply` . This population model will use sample data and one of the many available stock-recruitment models available in FLR, and a custom function will be written to return a single value for stock recruitment. Currently, `gmse_apply` requires that submodels return subfunction results either as scalar values or data frames that are structured in the same way as GMSE submodels. But interpretation of scalar values is left up to the user (e.g., population model results could be interpreted as abundance or biomass; manager policy could be interpreted as cost of harvesting or as total allowable catch). For simplicity, the observation (i.e., estimation) model will simply be the stock reported from the population model with error, and the manager model will be a total allowable catch calculated from the stock-recruitment relationship that accounts for the number of fishers in the system. The user model, however, will employ the full power of the default GMSE user function to simulate user actions. We first show how a custom function can be made that applies the FLR toolset to a population model.

Modelling stock-recruitment for the population model

Here we closely follow a tutorial from the FLR project (http://www.flr-project.org/doc/Modelling_stock_recruitment_with_FLSR.html). To build the stock-recruitment model, the `FLCore` package is needed (Kell et al. 2007).

```
install.packages(c("FLCore"), repos="http://flr-project.org/R");
```

To start, we need to read in the `FLCore` and GMSE libraries.

```
library(FLCore);
```

```
## Loading required package: lattice
```

```
## FLCore (Version 2.6.7, packaged: 2018-04-17 09:12:42 UTC)
```

```
library(GMSE);
```

For a simplified example in GMSE, we will simulate the process of stock recruitment over multiple time steps using an example stock-recruitment model. The stock-recruitment model describes the relationship between stock-recruitment and spawning stock biomass. The sample that we will work from is a recreation of the North Sea Herring (`nsher`) dataset available in the `FLCore` package (Kell et al. 2007). This data set includes recruitment and spawning stock biomass data between 1960 and 2004. First, we initialise an empty `FLSR` object and read in the recreated herring data files from GMSE, which contains recruitment (`rec.n`) and spawning stock biomass (`ssb.n`)

```
newFL   <- FLSR(); # Initialises the empty FLSR object
data(nsher_data); #called from GMSE library rather than nsher from FLCore library
```

The recruitment (`rec.n`) and spawning stock biomass (`ssb.n`) data need to be in the form of a vector, array, matrix to use them with `FLQuant` . We will convert `rec.n` and `ssb.n` into matrices.

```
rec.m        <- as.matrix(rec.n);
ssb.m        <- as.matrix(ssb.n);
```

We can then construct two `FLQuant` objects, specifying the relevant years and units.

```
Frec.m       <- FLQuant(rec.m, dimnames=list(age=1, year = 1960:2004));
Fssb.m       <- FLQuant(ssb.m, dimnames=list(age=1, year = 1960:2004));
Frec.m@units <- "10^3";
Fssb.m@units <- "t*10^3";
```

We then place the recruitment and spawning stock biomass data into the `FLSR` object that we created.

```
rec(newFL)    <- Frec.m;
ssb(newFL)    <- Fssb.m;
range(newFL)  <- c(0, 1960, 0, 2004);
```

The `FLCore` package offers several stock-recruitment models. Here we use a Ricker model of stock recruitment (Ricker 1954), and insert this model into the `FLSR` object below.

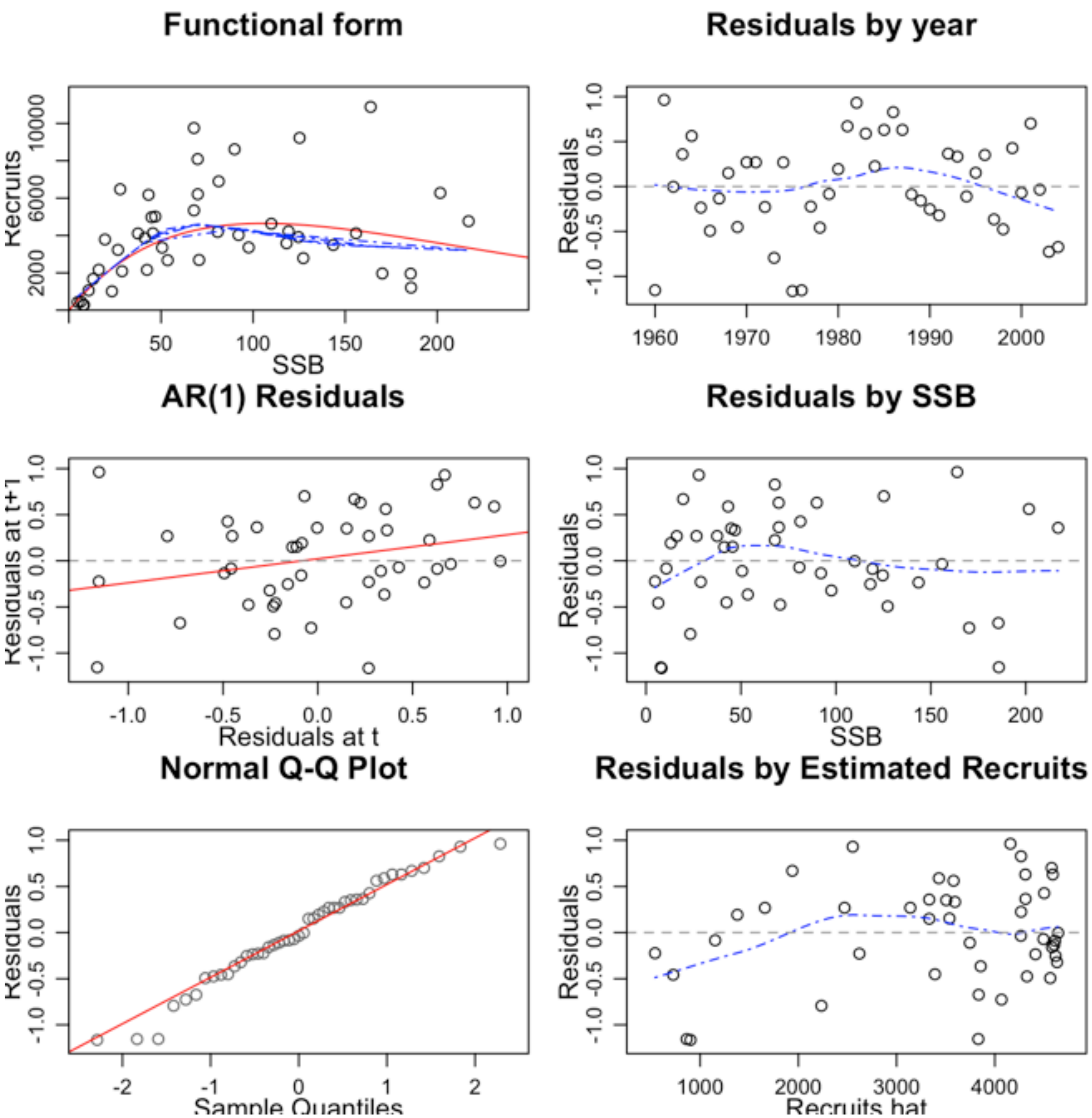
```
model(newFL) <- ricker();
```

Parameters for the Ricker stock-recruitment model can be estimated with maximum likelihood.

```
newFL <- fmle(newFL);
```

Diagnostic plots, identical to those of the modelling stock-recruitment tutorial (http://www.flr-project.org/doc/Modelling_stock_recruitment_with_FLSR.html) for the `nsher_ri` example, are shown below.

```
plot(newFL);
```



We now have a working example of a stock-recruitment model, but for our integration with `gmse_apply`, we will want a function that automates the above to simulate the process of updating the stock-recruitment model. We do this using the custom function created below.

```
update_SR_model <- function(rec_m, ssb_m, years){
    Frec_m       <- FLQuant(rec_m, dimnames=list(age = 1, year = years));
    Fssb_m       <- FLQuant(ssb_m, dimnames=list(age = 1, year = years));
    Frec_m@units <- "10^3";
    Fssb_m@units <- "t*10^3";
    rec(newFL)   <- Frec.m;
    ssb(newFL)   <- Fssb.m;
    range(newFL) <- c(0, years[1], 0, years[length(years)]);
    model(newFL) <- ricker();
    newFL        <- fmle(newFL);
    return(newFL);
}
```

The above function will be used within another custom function to predict the next time step of recruitment.

```
predict_recruitment <- function(rec_m, ssb_m, years, new_ssb){
    newFL <- update_SR_model(rec_m, ssb_m, years);
    a     <- params(newFL)[[1]] # Extract 'a' parameter of the Ricker model
    b     <- params(newFL)[[2]] # Extract 'b' parameter of the Ricker model
    rec   <- a * new_ssb * exp(-b * new_ssb); # Predict the new recruitment
    return(rec)
}
```

In `gmse_apply`, we will use the `predict_recruitment` function above as the resource (i.e., operational) model. The `new_ssb` reads in the new spawning stock biomass, which will be calculated from the built-in GMSE `user` model.

Integrating `predict_recruitment` with `gmse_apply`

The FLR project (http://www.flr-project.org/) includes libraries that can be used to perform a management strategy evaluation (http://www.flr-project.org/doc/An_introduction_to_MSE_using_FLR.html) (MSE) under fisheries-focused observation, manager, and user models. We will not recreate this approach (http://www.flr-project.org/doc/An_introduction_to_MSE_using_FLR.html), or integrate any other submodels into GMSE as was done for the population model above, although such integration of submodels should be possible using similar techniques. Our goal here is to instead show how the `predict_recruitment` model created above can be integrated with `gmse_apply`, which can then make use of the genetic algorithm to predict the behaviour fishers.

We will use a custom observation model, which will simply estimate recruitment with some fixed error.

```
obs_ssb <- function(resource_vector){
    obs_err <- rnorm(n = 1, mean = 0, sd = 100);
    the_obs <- resource_vector + obs_err;
    return(the_obs);
}
```

Hence, we can now feed the data from `rec.m` and `ssb.m` through `predict_recruitment`, which will return a value for new recruitment, and this new value can in turn be fed into `obs_ssb` to predict recruitment with some error. We also need a new spawning stock biomass `new_ssb`, which we can just initialise with the biomass from the last year in `ssb.m`

```
ssb_ini <- ssb.m[length(ssb.m)];
new_rec <- predict_recruitment(rec_m = rec.m, ssb_m = ssb.m, years = 1960:2004,
                               new_ssb = ssb_ini);
obs_rec <- obs_ssb(new_rec);
```

An initial run of these models gives values of 3835.21 for `new_rec` and 3732.61 for `obs_rec`. We are now ready to use the built-in manager and user submodels in `gmse_apply`. We will assume that managers attempt to keep a recruitment of 5000, and that there are 4 independent fishers [stakehodlers in gmse_apply says 10, is that a difference?] who attempt to maximise their catch. We assign a user budget of `manager_budget = 10000`, and all other values are set to GMSE defaults. In the built-in GMSE functions, the manager will use the estimate of recruitment based on `obs_rec` and use it to set the cost of harvesting (`culling` in GMSE).

```
yrspan        <- 1960:2004;
rec.m         <- as.matrix(rec.n);
ssb.m         <- as.matrix(ssb.n);

sim <- gmse_apply(res_mod = predict_recruitment, obs_mod = obs_ssb,
                  rec_m = rec.m, ssb_m = ssb.m, years = yrspan,
                  new_ssb = ssb_ini, manage_target = 5000, stakeholders = 10,
                  manager_budget = 10000);
print(sim);
```

```
## $resource_results
## [1] 3835
##
## $observation_results
## [1] 4106.386
##
## $manager_results
##          resource_type scaring culling castration feeding help_offspring
## policy_1             1      NA     442         NA      NA             NA
##
## $user_results
##         resource_type scaring culling castration feeding help_offspring
## Manager             1      NA       0         NA      NA             NA
## user_1              1      NA       2         NA      NA             NA
## user_2              1      NA       2         NA      NA             NA
## user_3              1      NA       2         NA      NA             NA
## user_4              1      NA       2         NA      NA             NA
## user_5              1      NA       2         NA      NA             NA
## user_6              1      NA       2         NA      NA             NA
## user_7              1      NA       2         NA      NA             NA
## user_8              1      NA       2         NA      NA             NA
## user_9              1      NA       2         NA      NA             NA
## user_10             1      NA       2         NA      NA             NA
##         tend_crops kill_crops
## Manager         NA         NA
## user_1          NA         NA
## user_2          NA         NA
## user_3          NA         NA
## user_4          NA         NA
## user_5          NA         NA
## user_6          NA         NA
## user_7          NA         NA
## user_8          NA         NA
## user_9          NA         NA
## user_10         NA         NA
```

The resource and observation results above are interpreted in terms of recruitment, while the manager results are interpreted in terms of the cost of harvesting a unit of spawning stock biomass and the user results are interpreted in terms of how much biomass was harvested. Note in the run of `gmse_apply` that the arguments for our custom resource and observation models (`predict_recruitment` and `obs_ssb` , respectively) are read directly in as arguments of `gmse_apply` itself. The `gmse_apply` function will figure out which subfunctions custom arguments should go to, then update these arguments as needed over the course of a single run of `gmse_apply` .

Simulation with `gmse_apply` over multiple time steps

We are now ready to loop the `gmse_apply` function over multiple time steps. To do this, we will update the `rec.m` and `ssb.m` matrices after each time step, simulating 20 years into the future. The population model `predict_recruitment` will use these data to dynamically update parameters of the Ricker model, as might occur in an empirical fishery that is being monitored. We will use the results from the observation model to update recruiment for the new year in `rec.m` . For simplicity, spawning stock biomass prior to harvest will be randomly sampled from a value in last 10 years (i.e., from `ssb.m` between 1994 and 2004), but more realistic models could relate this spawning stock biomass to recruitment and environmental variables from a prevoius year; spawning stock biomass will be decreased after harvest based on user actions. The GMSE initialisation and simulation is below.

```
# This code initialises the simulation -----------------------------------------
yrspan       <- 1960:2004;
rec.m        <- as.matrix(rec.n);
ssb.m        <- as.matrix(ssb.n);
ssb_ini      <- ssb.m[length(ssb.m)];
sim_old      <- gmse_apply(res_mod = predict_recruitment, obs_mod = obs_ssb,
                           rec_m = rec.m, ssb_m = ssb.m, years = yrspan,
                           new_ssb = ssb_ini, manage_target = 3500,
                           stakeholders = 10, manager_budget = 10000,
                           get_res = "Full");
# The code below simulates 20 time steps ---------------------------------------
sim_sum  <- matrix(data = NA, nrow = 20, ncol = 6); # Hold results here
for(time_step in 1:20){
    # Update the relevant parameter values as necessary -----------------------
    rand_ssb        <- sample(x = ssb.m[35:45], size = 1);
    harvest         <- sum(sim_old$basic_output$user_results[,3]);
    new_rec_m       <- c(sim_old$rec_m, sim_old$observation_vector);
    new_ssb_m       <- c(sim_old$ssb_m, rand_ssb - harvest);
    sim_old$rec_m   <- matrix(data = new_rec_m, nrow = 1);
    sim_old$ssb_m   <- matrix(data = new_ssb_m, nrow = 1);
    sim_old$years   <- c(sim_old$years, time_step + 2004);
    sim_old$new_ssb <- sim_old$ssb_m[length(sim_old$ssb_m)];
    # Run a new simulation in the loop: custom functions are always specified -
    sim_new  <- gmse_apply(get_res = "Full", old_list = sim_old,
                           res_mod = predict_recruitment, obs_mod = obs_ssb);
    # Record the results in sim_sum ---------------------------------------------
    sim_sum[time_step, 1] <- time_step + 2004;
    sim_sum[time_step, 2] <- sim_new$basic_output$resource_results[1];
    sim_sum[time_step, 3] <- sim_new$basic_output$observation_results[1];
    sim_sum[time_step, 4] <- sim_new$basic_output$manager_results[3];
    sim_sum[time_step, 5] <- harvest;
    sim_sum[time_step, 6] <- sim_new$new_ssb;
    # Redefine the old list ------------------------------------------------------
    sim_old               <- sim_new;
}
colnames(sim_sum) <- c("Year", "Recruitment", "Recruit_estim", "Harvest_cost",
                         "Harvested", "SSB");
print(sim_sum);
```

```
##       Year Recruitment Recruit_estim Harvest_cost Harvested      SSB
##  [1,] 2005        2347      2338.460          481        20  24.8673
##  [2,] 2006        3747      3732.141          481        20  50.6639
##  [3,] 2007        2736      2885.065          509        20  30.6133
##  [4,] 2008        3988      3951.546          481        10 175.5799
##  [5,] 2009        4647      4698.300          538        20 105.2627
##  [6,] 2010        4328      4282.308          491        10  70.7603
##  [7,] 2011        3747      3760.657          533        20  50.6639
##  [8,] 2012        4328      4464.711          591        10  70.7603
##  [9,] 2013        4387      4227.396          552        10 145.9025
## [10,] 2014        3303      3204.377          580        10  40.6133
## [11,] 2015        4629      4696.307          585        10 115.2627
## [12,] 2016        4208      4411.092          517        10 160.1926
## [13,] 2017        4082      3932.514          535        10  60.6639
## [14,] 2018        3303      3159.562          542        10  40.6133
## [15,] 2019        4387      4327.585          610        10 145.9025
## [16,] 2020        3303      3196.303          459        10  40.6133
## [17,] 2021        2919      2800.342          575        20  33.5966
## [18,] 2022        4387      4307.988          478        10 145.9025
## [19,] 2023        2736      2537.123          485        20  30.6133
## [20,] 2024        2347      2402.643          499        20  24.8673
```

The above output from `sim_sum` reports the recruitment (resource or operational model), recruitment estimate (observation error model), management set harvest cost (harvest control model), user harvested numbers (implementation model) and spawning stock biomass (SSB) simulation results. This example simulation demonstrates the ability of GMSE to integrate with fisheries libraries such as FLR (http://www.flr-project.org/) through `gmse_apply` . In addition to being a useful wrapping function for MSE submodels, `gmse_apply` can therefore be used to take advantage of the genetic algorithm in the GMSE default manager and user models. This flexibility will be retained in future versions of `gmse_apply` , allowing custom resource and observation models that are built for specific systems to be integrated with an increasingly complex genetic algorithm simulating various aspects of human decision-making.

Conclusions

GMSE is a general, flexible, tool for simulating the management of resources under situations of uncertainty and conflict. Management Strategy Evaluation (Bunnefeld, Hoshino, and Milner-Gulland 2011; Punt et al. 2016), the framework upon which GMSE is based, had its origin in fisheries management (Polacheck et al. 1999; Smith, Sainsbury, and Stevens 1999; Sainsbury, Punt, and Smith 2000), and here we showed one example of how GMSE could be integrated with the core package of the Fisheries Library in R (http://www.flr-project.org/).

Future versions of GMSE will continue to be open-source and developed to avoid unecessary dependencies (GMSE v.0.4.0.3 requires only base R). Key goals including (1) providing highly general and useful default `resource` , `observation` , `manager` , and `user` submodels for a variety of MSE modelling tasks, (2) keeping these submodels highly modular so that they can be developed in isolation given standardised data structures, and (3) allowing these modular submodels to be integrated with custom defined submodels as flexibly as possible using `gmse_apply` . Contributions in line with these goals, and suggestions for new features, can be made on GitHub (https://github.com/bradduthie/gmse).

Bunnefeld, Nils, Eriko Hoshino, and Eleanor J Milner-Gulland. 2011. “Management strategy evaluation: A powerful tool for conservation?” *Trends in Ecology and Evolution* 26 (9): 441–47. doi:10.1016/j.tree.2011.05.003 (https://doi.org/10.1016/j.tree.2011.05.003).

Carruthers, T, and D Butterworth. 2018a. “ABT-MSE : An R package for atlantic bluefin tuna management strategy evaluation.” *Collective Volume of Scientific Papers ICCAT* 74 (6): 3553–9.

———. 2018b. “Performance of example management procedures for atlantic bluefin tuna.” *Collective Volume of Scientific Papers ICCAT* 73 (6): 3542–52.

Jardim, Ernesto, Margit Eero, Alexandra Silva, Clara Ulrich, Lionel Pawlowski, Isabel Riveiro, Steven J Holmes, et al. 2018. “Testing spatial heterogeneity with stock assessment models.” *PLoS One* 13: e0190891. doi:10.1371/journal.pone.0190791 (https://doi.org/10.1371/journal.pone.0190791).

Kell, L T, I Mosqueira, P Grosjean, Jean-Marc Fromentin, D Garcia, R Hillary, E Jardim, et al. 2007. “FLR: an open-source framework for the evaluation and development of management strategies.” *ICES Journal of Marine Science* 64 (4): 640–46. https://mail.google.com/mail/u/0/?ui=2{\&}pli=1{\%}5Cnpapers2://publication/doi/10.1093/icesjms/fsm012 (https://mail.google.com/mail/u/0/?ui=2{\&}pli=1{\%}5Cnpapers2://publication/doi/10.1093/icesjms/fsm012).

Kolody, Dale, and Paavo Jumppanen. 2016. “IOTC Yellowfin and Bigeye Tuna Management Strategy Evaluation: Phase 1 Technical Support Project Final Report.” June. CSIRO: Oceans & Atmosphere.

Mackinson, Steven, Mark Platts, Clement Garcia, and Christopher Lynam. 2018. “Evaluating the fishery and ecological consequences of the proposed North Sea multi-annual plan.” *PLoS One* 13: e0190015. doi:10.1371/journal.pone.0190015 (https://doi.org/10.1371/journal.pone.0190015).

Polacheck, T, N L Klaer, C Millar, and A L Preece. 1999. “An initial evalutaion of management strategies for the southern bluefin tuna fishery.” *ICES Journal of Marine Science* 56 (6): 811–26. doi:10.1006/jmsc.1999.0554 (https://doi.org/10.1006/jmsc.1999.0554).

Punt, André E, Doug S Butterworth, Carryn L de Moor, José A A De Oliveira, and Malcolm Haddon. 2016. “Management strategy evaluation: Best practices.” *Fish and Fisheries* 17 (2): 303–34. doi:10.1111/faf.12104 (https://doi.org/10.1111/faf.12104).

Ricker, W E. 1954. “Stock and recruitment.” *Journal of the Fisheries Board of Canada* 11 (5): 559–623.

Sainsbury, Keith J, André E Punt, and Anthony DM Smith. 2000. “Design of operational management strategies for achieving fishery ecosystem objectives.” *ICES Journal of Marine Science* 57 (3): 731–41. doi:10.1006/jmsc.2000.0737 (https://doi.org/10.1006/jmsc.2000.0737).

Smith, A D M, K J Sainsbury, and R A Stevens. 1999. “Implementing effective fisheries-management systems – management strategy evaluation and the Australian partnership approach.” *ICES Journal of Marine Science* 56 (6): 967–79. doi:10.1006/jmsc.1999.0540 (https://doi.org/10.1006/jmsc.1999.0540).

Utizi, Kizzi, Emilio Notti, Antonello Sala, Alessandro Buzzi, Ilaria Rodella, Umberto Simeoni, and Corinne Corbau. 2018. “Impact assessment of EMFF measures on Good Environmental Status (GES) as defined by Italy.” *Marine Policy* 88. Elsevier Ltd: 248–60. doi:10.1016/j.marpol.2017.12.003 (https://doi.org/10.1016/j.marpol.2017.12.003).