DLMtool: Data-Limited Methods Toolkit (v2.1.1)

Tom Carruthers*and Adrian Hordyk

November 2015

Contents

*t.carruthers@fisheries.ubc.ca

1 Introduction

As many as 90 per cent of the world's fish populations have insufficient data to conduct a conventional stock assessment (Costello et al. 2012). Although a wide range of data-limited management procedures (MPs; stock assessments, harvest control rules) have been described in the primary and gray literature (Newman et al. 2014), these are not readily available, easily tested or compared. Critically, the path forward is unclear. How do these MPs perform comparatively? What are the performance trade-offs? What MPs are inappropriate for given stock/fishery/data quality? What is the value of collecting additional data? What is an appropriate stop-gap management approach?

DLMtool is a collaboration between the University of British Columbia and the Natural Resources Defense Council aimed at addressing these questions by offering a powerful, transparent approach to selecting and applying various data-limited MPs. DLMtool uses Management Strategy Evaluation (MSE, closed-loop simulation) and parallel computing to make powerful diagnostics accessible. A streamlined command structure and operating model builder allow for rapid simulation testing and graphing of results. The package is relatively easy to use for those inexperienced in R, however complete access and control is available to more experienced users.

While DLMtool includes over 55 MPs (e.g. DCAC, DBSRA), it is also designed to be extensible in order to encourage the development and testing of new MPs for informing management of data-limited fish stocks. The package is structured such that the same MP functions that are tested by MSE can be applied to provide management recommendations from real data. Easy incorporation of real data is central advantage of the software and a set of related functions automatically detect what MP can be applied given the available data and what additional data are required to get other MPs working.

DLMtool has been used in setting catch-limits at the Mid-Atlantic Fishery Management Council (US) and is being used to test management procedures in California state fisheries (California Deparment of Fish and Wildlife), the Caribbean (NOAA), and for seafood certification purposes (MSC).

2 A note on version 2.1.1

A bug was found in which length at first capture was being sampled from a uniform distribution U(LB,UB*2) rather than U(LB,UB). This has been fixed.

3 A note on version 2.1

The package has overgone a substantial overhaul. In response to popular demand, simulation and data are entirely length-based now. It follows that many objects that worked with 2.0 will no longer be compatible. In most cases it is very quick to make files/objects compatible with version 2.1 but nonetheless we apologise if this is frustrating!

The package is subject to ongoing testing. Once again, if you find a bug or a problem please send a report to t.carruthers@fisheries.ubc.ca so that I can fix it!

Fundamentally the package is stochastic so if you run into problems with the code, please report it (along with a random seed) and in the mean time simply try running it again: the problem may be attributable to a rare combination of sampled parameters.

Be warned that if you abort a parallel process (e.g. runMSE()) half-way through you are in the lap of the Gods! It will often be necessary to restart the cluster sfInit() or even restart R.

Its probably best not to try and use the package for very short lived stocks (that live for less than 5 years) due to the problems with approximating fine-scale temporal dynamics with an annual model. Technically you could just divide all your parameters by a subyear resolution but the TAC would be set by sub year and the data would also be available at this fine-scale which is highly unlikely in a data-limited setting.

—- New to version 2.1 —-

(1) The whole toolkit has moved to a length-based simulator (maturity, fisheries selectivity by length)

(2) We've dropped spatial targetting for the moment as it was a flawed implementation and could not distribute fishing correctly with respect to both density and amount of resource among the two areas.

(3) Tplot2 adds a different set of tradeoffs including long term and short term probability of achieving 50 per cent of FMSY yield and average annual variability in yields

(4) Version 2.0 was bugged and did not include observation error in estimates of current stock abundance and depletion (only biases were simulated). Many thanks to Helena Geromont for spotting this. This has now been corrected

(5) DLM_data objects now have a slot LHYear which is a numeric value corresponding with the last historical year. This is needed for some MPs that want to run off only the past data rather than the updated (projected, closed-loop simulation) data.

(6) Post-MSE you can now run a Convergence function CheckConverg() to see if performance metrics are stable.

(7) The package now contains CSRA a tool for calculating very rough estimates of current depletion and fishing mortality rate from mean catch data.

(8) getAFC is also available that can be used for converting length estimates to age estimates through a stochastic growth model.

(9) The value of information function (VOI) was bugged in version 2.0. This has now been fixed.

(10) Users can now send their own parameter values to the runMSE function allowing outputs from stock assessments or correlated parameters (e.g. K and age at maturity) values.

(11) After deliberation, Pope's approximation has been used to account for intra-year mortality (ie TACs are taken from biomass at the start of the year subject to half of natural mortality rate). This is probably a reasonable approximation in a data-limited setting: alternative structural assumptions for M are eclipsed by

uncertainty in M itself and other operating model parameters such as selectivity and bias in observation of data such as annual catches.

(12) The simulation of length composition data was bugged in version 2.0. The variability in length at age was taken from the observation model. Using the perfect information observation model therefore led to no variabiltiy in length at age and hence very odd length composition data. This has been solved and for now a fixed 10 per cent CV in length at age is assumed (normally distributed).

(13) A bug with delay-difference MPs has been fixed (DD and DD4010) in which stochastic TACs were sampled when reps =1. This should just be the mean estimate. The result is that DD is much less variable between years but comes with less contrast in the data. In addition to the much less variable catch recommendations, long term mean performance of the MP is reduced while medium term peformance has been improved.

(14) In the move to length-based inputs it is possible to prescribe wild biases for maximum length and length at maturity. In this version these sampled biases are not correlated so it is possible to create simulated data sets where maximum length is lower than length at 95 per cent maturity and length at 50 per cent maturity. We put a hard ceiling on this such that length at 95 per cent maturity must be below 90 percent of maximum length and length at 50 percent maturity must be below 90 percent of length at 95 percent maturity. This isn't great and this will be improved for v2.11

(15) The package now works without initiating a cluster sfInit().

(16) A simple modification to DCAC has been added EDCAC (Harford and Carruthers 2015) that better accounts for absolute stock depletion.

—- New to version 2.0 —-

(1) Much has changed in package terminology to make the package more generally applicable. For example, OFL (overfishing limits, FMSY x current biomass), now belongs to a larger class of TACs (Total Allowable Catches).

(2) There are now just two classes of DLM MPs, DLM_output (MPs linked to output controls e.g. TACs) and DLM_input (MPs linked to input controls such as time-area closures, age selectivity and effort).

The new DLM_input function classes have four components, fractional reallocation of spatial effort, fraction of effort in final historical year prescribed in the current year, spatial limits on fishing mortality and a user-defined age-selectivity curve. For example, given an hypothetical stock with 8 age classes a DLM_input method might return a vector c(0.5, 0.8, 0,1, 0,0,0,0,1,1,1,1). This is interpreted as a 50 percent reallocation (Allocation = 0.5) of spatial effort, with a total effort that is 80 percent of historical levels (Effort = 0.8) with a closure in area 1 and full fishing in area 2 (Spatial = c(0,1)) and knife-edge selectivity at age class 5 (Selectivity = c(0,0,0,0,1,1,1,1)).

To demonstrate this new feature there are four new input controls, current effort (curE), 75 percent of current effort (curE75), age selectivity that matches the maturity ogive (matagelim) and a marine reserve in area 1 (area1MR).

(3) A 'dumb' MP has been added: Mean Catch Depletion (MCD) that simply calculates a TAC based on mean catches and depletion ie depletion x 2 x mean catch. This is to demonstrate the (theoretically) very high information content of a reliable estimate of current stock depletion.

(4) A better length composition simulator has been added. Note that this still renews the normal length structure between ages and does not properly simulate the higher mortality rate of larger, faster growing fish (a growth type group simulator is on its way).

(5) Help documentation has been much improved including complete guides for Fleet, Stock, Observation and MSE objects. Eg class?MSE

(6) Minor bugs have been found with the help of Helena Geromont including a problem with update intervals of 1 and low simulated steepness values.

(7) Reliability is much improved following a full combinatorial test of all Fleet, Stock, Observation objects against all MPs.

(8) A dedicated Value of information function is now available for MSE objects: VOI(MSEobject) which is smarter than the former version which was included in plot(MSE object class).

(9) Plotting functions have been improved, particularly Tplot, Kplot, Pplot and plot(DLM_data object class)

(10) SPmod has been robustified to stop strongly negative surplus production estimates from leading to erratic behavior.

(11) The butterfish stock type now has less variable recruitment and slightly lower natural mortality rate as previous values were rather extreme and lead to data generation errors (with natural mortality rate as high as 0.9, butterfish is right at the limit of what can be simulated reasonably with an annual age-structured operating model)

—- Coming soon to v2.11 —-

Spawning Potential Ratio (SPR) based MPs (e.g. Prince et al. 2014).

Input control versions of existing DLM_output MPs.

Growth type group simulators of length composition data.

Better simulation of length at maturity and maximum length.

An implementation model that allows for simple bioeconomics and deliberate adaptive management practices.

4 Prerequisites

At the start of every session there are a few things to do: load the DLMtool library, make data available and set up parallel computing.

4.1 Loading the library

```
library(DLMtool)

## Loading required package:  snowfall
## Loading required package:  snow
## Loading required package:  boot
## Loading required package:  MASS
## Loading required package:  parallel
##
## Attaching package:  'parallel'
##
## The following objects are masked from 'package:snow':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##    clusterExport, clusterMap, clusterSplit, makeCluster,
##    parApply, parCapply, parLapply, parRapply, parSapply,
##    splitIndices, stopCluster
```

4.2 Unpacking data

A list object DLMdat is unpacked which puts all objects and data in the current workspace.

```
for(i in 1:length(DLMdat))assign(DLMdat[[i]]@Name,DLMdat[[i]])
```

4.3 Initiating the cluster

Note that most computers make use of hyperthreading technology so a quad-core PC has 8 threads, this is set to 2 here to meet CRAN-R package submission requirements. You can automatically detect the number of threads using detectCores(). Ie type sfInit(parallel=T,detectCores()). Currently I cannot get a vignette to build with a cluster so these lines are not included anymore. In practice you should use cluster computing however.

```
#sfInit(parallel=T,cpus=2)
```

4.4 Exporting all data and objects to the cluster

In order to make all DLMtool functions and objects available for parallel processing we export them to the cluster. As with sfInit() I couldnt' include cluster computing for vignette building so this line is disabled - you should however run this after sfInit().

```
#sfExportAll()
```

4.5 Set a random seed

In order to make results presented here reproducible, we set a random seed for this R session.

```
set.seed(1)
```

5 Quick start

Here is a quick demonstration of core DLMtool functionality.

5.1 Define an operating model

The operating model is the 'simulated reality': a series of known simulations for testing various data-limited MPs. Operating models can either be specified in detail according to each variable (e.g. sample natural mortality rate between 0.2 and 0.3, trajectory in fishing effort of between 0.5 and 1 per cent per time step) or alternatively the user can rapidly construct an operating model based on a set of predefined Stock, Fleet and Observation models. In this case we take the latter approach and pick the Blue_shark stock type, a Generic fleet type and an observation model that generates data that can be both imprecise and biased.

```
OM<-new('OM',
        Blue_shark,
        Generic_fleet,
        Imprecise_Biased)
```

The operating model class 'OM' has many different slots which control the ranges of population and fleet parameters that may be sampled in addition to parameters that control the quality of the data simulated. You can list these using slotNames() or can look up the help file entry:

```
slotNames(OM)
```

```
##  [1] "Name"          "nyears"        "maxage"        "R0"
##  [5] "M"             "Msd"           "Mgrad"         "h"
##  [9] "SRrel"         "Linf"          "K"             "t0"
## [13] "Ksd"           "Kgrad"         "Linfsd"        "Linfgrad"
## [17] "recgrad"       "a"             "b"             "D"
## [21] "Size_area_1"   "Frac_area_1"   "Prob_staying"  "Source"
## [25] "L50"           "L50_95"        "L5"            "LFS"
## [29] "Vmaxlen"       "beta"          "Spat_targ"     "Fsd"
## [33] "Fgrad"         "qinc"          "qcv"           "AC"
## [37] "Cobs"          "Cbiascv"       "CAA_nsamp"     "CAA_ESS"
## [41] "CAL_nsamp"     "CAL_ESS"       "CALcv"         "Iobs"
## [45] "Perr"          "Mcv"           "Kcv"           "t0cv"
## [49] "Linfcv"        "LFCcv"         "LFScv"         "B0cv"
## [53] "FMSYcv"        "FMSY_Mcv"      "BMSY_B0cv"     "LenMcv"
## [57] "rcv"           "Dbiascv"       "Dcv"           "Btbias"
## [61] "Btcv"          "Fcurbiascv"    "Fcurcv"        "hcv"
## [65] "Icv"           "maxagecv"      "Reccv"         "Irefcv"
## [69] "Crefcv"        "Brefcv"

class?OM
```

5.2 Define a subset of data-limited MPs

There are three different types of MP currently included in DLMtool: DLM_output (output controls, e.g. a TAC), DLM_input (size/age/spatial controls). In this example we use a generic class finder 'avail' to list all available methods of class 'DLM_output' and select some for simulation testing.

```
avail('DLM_output')

##  [1] "BK"          "BK_CC"       "BK_ML"       "CC1"         "CC4"
##  [6] "CompSRA"     "CompSRA4010" "DBSRA"       "DBSRA4010"   "DBSRA_40"
## [11] "DBSRA_ML"    "DCAC"        "DCAC4010"    "DCAC_40"     "DCAC_ML"
## [16] "DD"          "DD4010"      "DepF"        "DynF"        "EDCAC"
## [21] "FMSYref"     "FMSYref50"   "FMSYref75"   "Fadapt"      "Fdem"
## [26] "Fdem_CC"     "Fdem_ML"     "Fratio"      "Fratio4010"  "Fratio_CC"
## [31] "Fratio_ML"   "GB_CC"       "GB_slope"    "GB_target"   "Gcontrol"
## [36] "Islope1"     "Islope4"     "Itarget1"    "Itarget4"    "LstepCC1"
## [41] "LstepCC4"    "Ltarget1"    "Ltarget4"    "MCD"         "MCD4010"
## [46] "Rcontrol"    "Rcontrol2"   "SBT1"        "SBT2"        "SPMSY"
## [51] "SPSRA"       "SPSRA_ML"    "SPmod"       "SPslope"     "YPR"
## [56] "YPR_CC"      "YPR_ML"

MPs<-c("Fratio",
           "DCAC",
           "Fdem",
           "DD")
```

To find out more about these MPs you can use the built-in R help functions. E.g:

```
?Fratio
?DBSRA
```

Or simply view the code. E.g:

```
Fratio
```

```
## function (x, DLM_data, reps = 100)
## {
##     depends = "DLM_data@Abun,DLM_data@CV_Abun,DLM_data@FMSY_M,\n          DLM_data@CV_FMSY_M,DLM_data
##     Ac <- trlnorm(reps, DLM_data@Abun[x], DLM_data@CV_Abun[x])
##     TACfilter(Ac * trlnorm(reps, DLM_data@Mort[x], DLM_data@CV_Mort[x]) *
##         trlnorm(reps, DLM_data@FMSY_M[x], DLM_data@CV_FMSY_M[x]))
## }
## <environment: namespace:DLMtool>
## attr(,"class")
## [1] "DLM_output"
```

5.3 Run an MSE and plot results

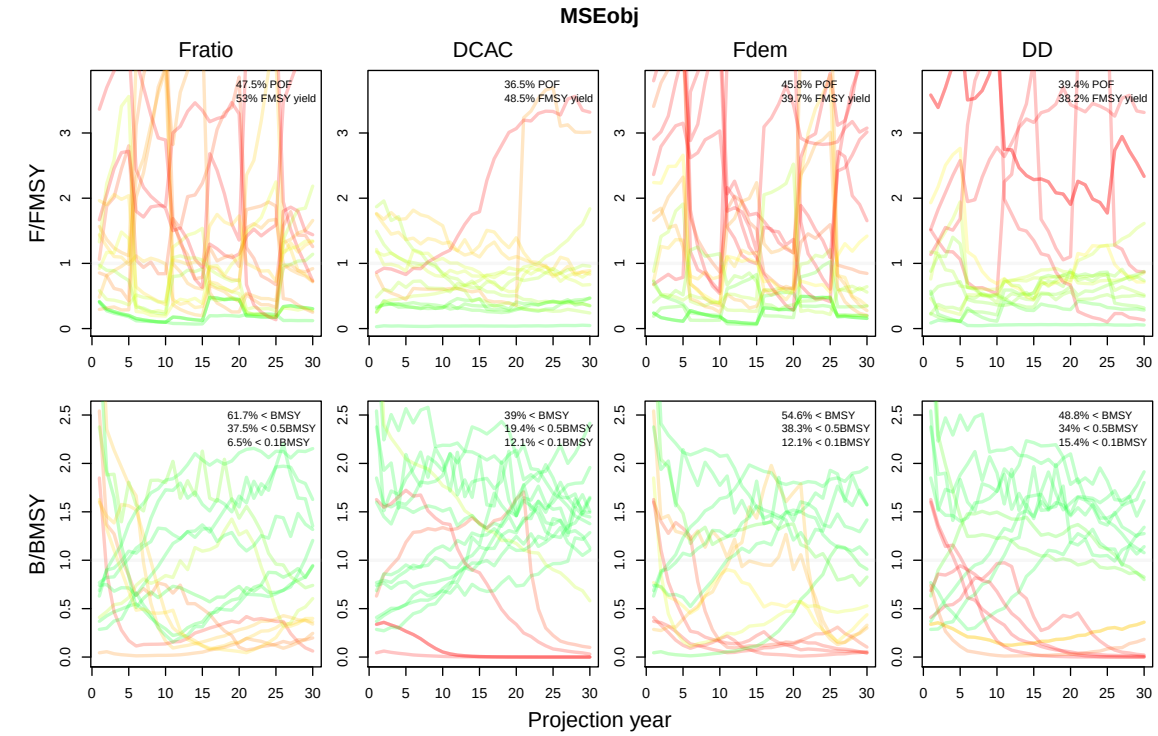
The MPs can now be tested using the operating model. NOTE that this is just a demonstration, in a real MSE you should use many more simulations (nsim more than 200), reps (samples per method more than 100) and perhaps a more frequent assessment interval (interval of 2 or 3 years). Note that when reps is set to 1, all stochastic MPs use the mean value of an input and do not sample from the distribution according to the specified CV (the DLM_output MPs become deterministic and no longer produce samples of the TAC recommendation).

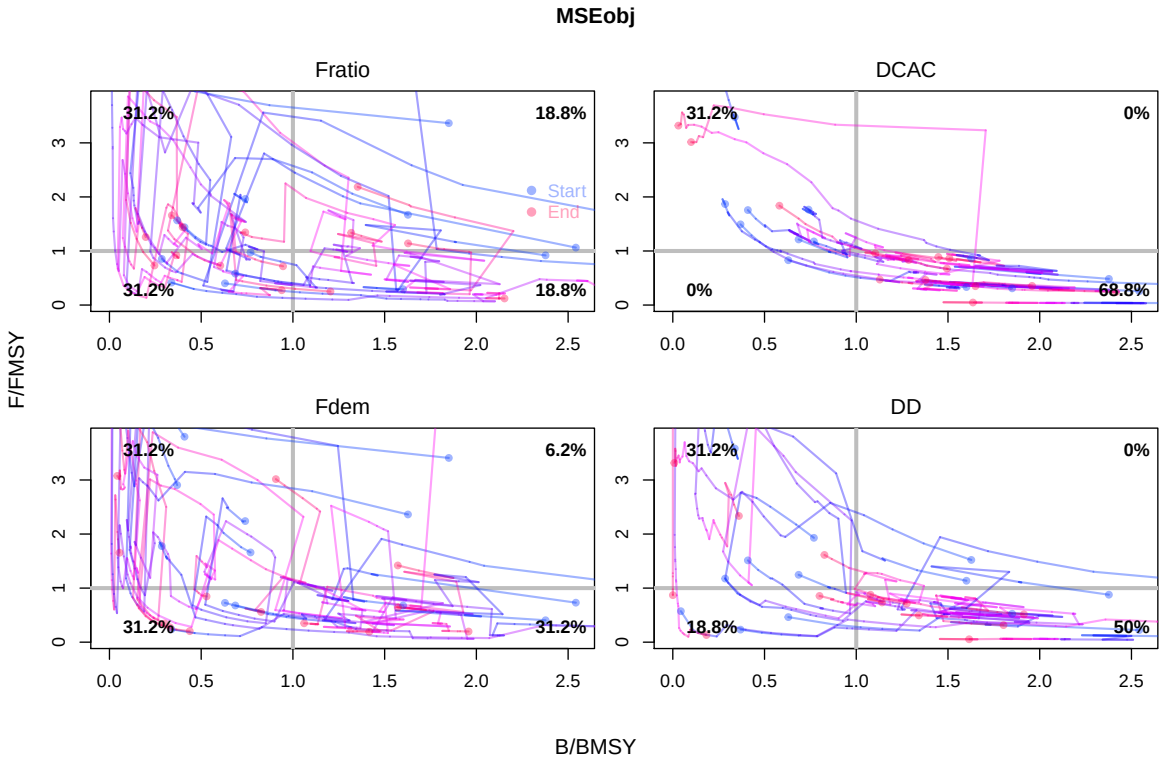
```
SnapMSE<-runMSE(OM,MPs,nsim=16,reps=1,proyears=30,interval=5)
```

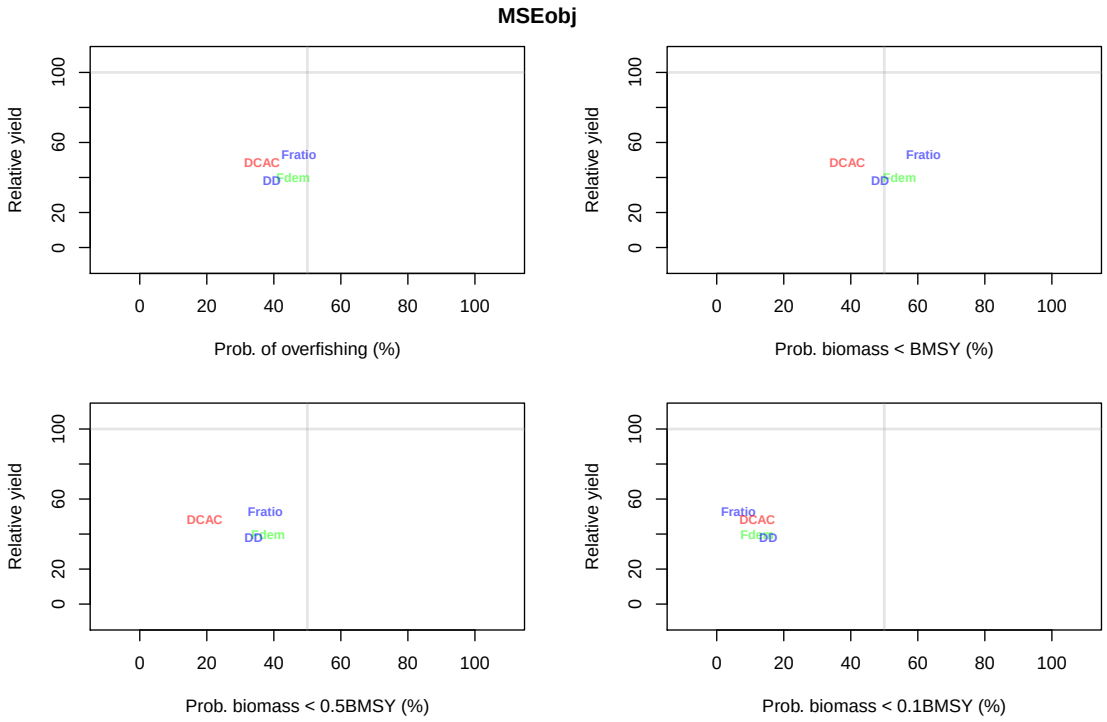
```
## [1] "Loading operating model"
## [1] "Optimizing for user-specified movement"
## [1] "Optimizing for user-specified depletion"
## [1] "Calculating historical stock and fishing dynamics"
## [1] "Calculating MSY reference points"
## [1] "Calculating reference yield - best fixed F strategy"
## [1] "Determining available methods"
## [1] "1/4 Running MSE for Fratio"
## ............................
## [1] "2/4 Running MSE for DCAC"
## ............................
## [1] "3/4 Running MSE for Fdem"
## ............................
## [1] "4/4 Running MSE for DD"
## ............................
```

The generic plot method provides (1) trade-off plots of expected (mean) performance of the MPs in terms of stock status, overfishing and yield, Kobe plots and overfishing trajectories.

```
plot(SnapMSE)
```


MSEobj
Fratio
DCAC
Fdem
DD
47.5% POF
53% FMSY yield
36.5% POF
48.5% FMSY yield
45.8% POF
39.7% FMSY yield
39.4% POF
38.2% FMSY yield
F/FMSY
61.7% < BMSY
37.5% < 0.5BMSY
6.5% < 0.1BMSY
39% < BMSY
19.4% < 0.5BMSY
12.1% < 0.1BMSY
54.6% < BMSY
38.3% < 0.5BMSY
12.1% < 0.1BMSY
48.8% < BMSY
34% < 0.5BMSY
15.4% < 0.1BMSY
B/BMSY
Projection year


MSEobj
Fratio
DCAC
31.2%
18.8%
Start
End
31.2%
18.8%
31.2%
0%
0%
68.8%
F/FMSY
Fdem
DD
31.2%
6.2%
31.2%
31.2%
31.2%
0%
18.8%
50%
B/BMSY



You can access these plots individually: trade-offs - Tplot(), overfishing trajectories - Pplot() and Kobe plots - Kplot()

5.4 Applying MPs to real data

A number of real DLM data-objects (class DLM_data) were loaded into the workspace at the start of this session. In this section we examine a real data object and apply data-limited MPs to these data. Just like the operating model we can find all the objects of real data class 'DLM_data', we can list the slots of a DLM_data object and also look up this class in the help file:

```
avail('DLM_data')
```

```
## [1] "Atlantic_mackerel"  "China_rockfish"     "Cobia"
## [4] "Example_datafile"   "Gulf_blue_tilefish" "Red_snapper"
## [7] "Simulation_1"       "ourReefFish"
```

```
slotNames(China_rockfish)
```

```
##  [1] "Name"       "Year"       "Cat"        "Ind"        "Rec"
##  [6] "t"          "AvC"        "Dt"         "Mort"       "FMSY_M"
## [11] "BMSY_B0"    "Cref"       "Bref"       "Iref"       "L50"
## [16] "L95"        "LFC"        "LFS"        "CAA"        "Dep"
## [21] "Abun"       "vbK"        "vbLinf"     "vbt0"       "wla"
## [26] "wlb"        "steep"      "CV_Cat"     "CV_Dt"      "CV_AvC"
## [31] "CV_Ind"     "CV_Mort"    "CV_FMSY_M"  "CV_BMSY_B0" "CV_Cref"
## [36] "CV_Bref"    "CV_Iref"    "CV_Rec"     "CV_Dep"     "CV_Abun"
## [41] "CV_vbK"     "CV_vbLinf"  "CV_vbt0"    "CV_L50"     "CV_LFC"
## [46] "CV_LFS"     "CV_wla"     "CV_wlb"     "CV_steep"   "sigmaL"
## [51] "MaxAge"     "Units"      "Ref"        "Ref_type"   "Log"
```

```
## [56] "params"     "PosMPs"     "MPs"        "OM"         "Obs"
## [61] "TAC"        "TACbias"    "Sense"      "CAL_bins"   "CAL"
## [66] "MPrec"      "LHYear"

class?DLM_data
```

DLMtool includes functions to interrogate a real data object to see what methods can be applied, those that cannot and also what data are needed to get those methods working:

```
Can(China_rockfish)

## [1] "CC1"      "CC4"      "DCAC"     "DCAC4010" "DCAC_40"  "EDCAC"
## [7] "MCD"      "MCD4010"

Cant(China_rockfish)

##       [,1]           [,2]
##  [1,] "BK"           "Produced all NA scores"
##  [2,] "BK_CC"        "Insufficient data"
##  [3,] "BK_ML"        "Insufficient data"
##  [4,] "CompSRA"      "Insufficient data"
##  [5,] "CompSRA4010" "Insufficient data"
##  [6,] "DBSRA"        "Insufficient data"
##  [7,] "DBSRA4010"    "Insufficient data"
##  [8,] "DBSRA_40"     "Insufficient data"
##  [9,] "DBSRA_ML"     "Insufficient data"
## [10,] "DCAC_ML"      "Insufficient data"
## [11,] "DD"           "Insufficient data"
## [12,] "DD4010"       "Insufficient data"
## [13,] "DepF"         "Produced all NA scores"
## [14,] "DynF"         "Insufficient data"
## [15,] "FMSYref"      "Insufficient data"
## [16,] "FMSYref50"    "Insufficient data"
## [17,] "FMSYref75"    "Insufficient data"
## [18,] "Fadapt"       "Insufficient data"
## [19,] "Fdem"         "Insufficient data"
## [20,] "Fdem_CC"      "Insufficient data"
## [21,] "Fdem_ML"      "Insufficient data"
## [22,] "Fratio"       "Produced all NA scores"
## [23,] "Fratio4010"   "Produced all NA scores"
## [24,] "Fratio_CC"    "Insufficient data"
## [25,] "Fratio_ML"    "Insufficient data"
## [26,] "GB_CC"        "Produced all NA scores"
## [27,] "GB_slope"     "Insufficient data"
## [28,] "GB_target"    "Insufficient data"
## [29,] "Gcontrol"     "Insufficient data"
## [30,] "Islope1"      "Insufficient data"
## [31,] "Islope4"      "Insufficient data"
## [32,] "Itarget1"     "Insufficient data"
## [33,] "Itarget4"     "Insufficient data"
## [34,] "LstepCC1"     "Insufficient data"
## [35,] "LstepCC4"     "Insufficient data"
## [36,] "Ltarget1"     "Insufficient data"
## [37,] "Ltarget4"     "Insufficient data"
```

```
## [38,] "Rcontrol"    "Insufficient data"
## [39,] "Rcontrol2"   "Insufficient data"
## [40,] "SBT1"        "Insufficient data"
## [41,] "SBT2"        "Produced all NA scores"
## [42,] "SPMSY"       "Insufficient data"
## [43,] "SPSRA"       "Insufficient data"
## [44,] "SPSRA_ML"    "Insufficient data"
## [45,] "SPmod"       "Insufficient data"
## [46,] "SPslope"     "Insufficient data"
## [47,] "YPR"         "Insufficient data"
## [48,] "YPR_CC"      "Insufficient data"
## [49,] "YPR_ML"      "Insufficient data"
## [50,] "MRnoreal"    "Insufficient data"
## [51,] "MRreal"      "Insufficient data"
## [52,] "curE"        "Insufficient data"
## [53,] "curE75"      "Insufficient data"
## [54,] "matagelim"   "Insufficient data"
```

```
Needed(China_rockfish)
```

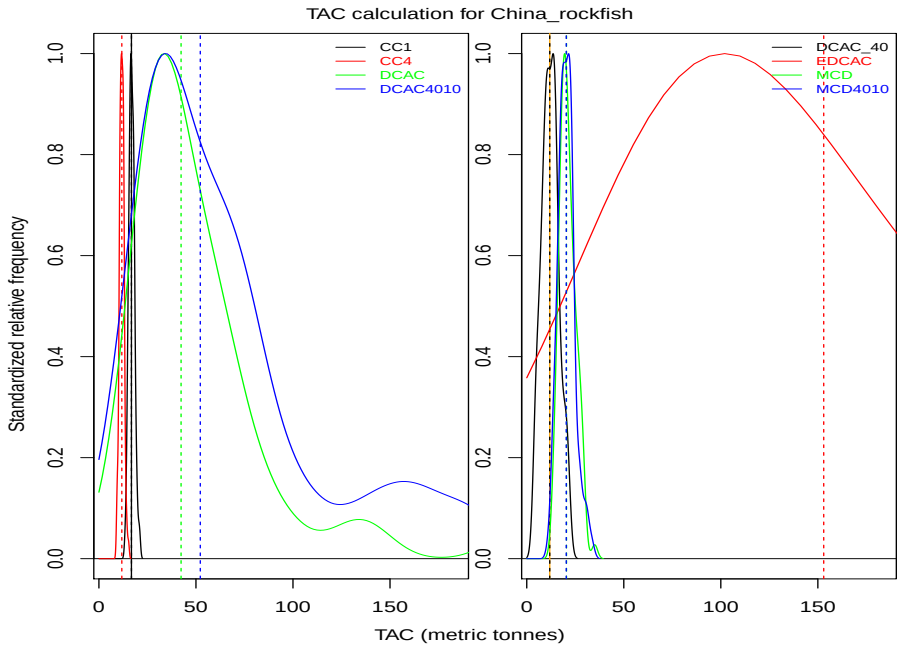
```
##  [1] "BK: LFC, Abun, vbK, vbLinf"
##  [2] "BK_CC: LFC, CAA, vbK, vbLinf"
##  [3] "BK_ML: LFC, vbK, vbLinf, CAL"
##  [4] "CompSRA: L50, LFC, LFS, CAA, vbK, vbLinf, vbt0, wla, wlb, steep, MaxAge"
##  [5] "CompSRA4010: L50, LFC, LFS, CAA, vbK, vbLinf, vbt0, wla, wlb, steep, MaxAge"
##  [6] "DBSRA: L50, Dep, vbK, vbLinf, vbt0"
##  [7] "DBSRA4010: L50, Dep, vbK, vbLinf, vbt0"
##  [8] "DBSRA_40: L50, vbK, vbLinf, vbt0"
##  [9] "DBSRA_ML: L50, vbK, vbLinf, vbt0, CAL"
## [10] "DCAC_ML: vbK, vbLinf, CAL"
## [11] "DD: Ind, L50, vbK, vbLinf, vbt0, wla, wlb, MaxAge"
## [12] "DD4010: Ind, L50, vbK, vbLinf, vbt0, wla, wlb, MaxAge"
## [13] "DepF: Dep, Abun"
## [14] "DynF: Ind, Abun"
## [15] "FMSYref: OM"
## [16] "FMSYref50: OM"
## [17] "FMSYref75: OM"
## [18] "Fadapt: Ind, Abun"
## [19] "Fdem: Abun, vbK, vbLinf, vbt0, wla, wlb, steep, MaxAge"
## [20] "Fdem_CC: CAA, vbK, vbLinf, vbt0, wla, wlb, steep, MaxAge"
## [21] "Fdem_ML: vbK, vbLinf, vbt0, wla, wlb, steep, MaxAge, CAL"
## [22] "Fratio: Abun"
## [23] "Fratio4010: Dep, Abun"
## [24] "Fratio_CC: CAA"
## [25] "Fratio_ML: vbK, vbLinf, CAL"
## [26] "GB_CC: Cref"
## [27] "GB_slope: Ind"
## [28] "GB_target: Ind, Cref, Iref"
## [29] "Gcontrol: Ind, Abun"
## [30] "Islope1: Ind, MPrec"
## [31] "Islope4: Ind, MPrec"
## [32] "Itarget1: Ind, CAL"
## [33] "Itarget4: Ind, CAL"
## [34] "LstepCC1: CAL, MPrec"
## [35] "LstepCC4: CAL, MPrec"
```

```
## [36] "Ltarget1: CAL"
## [37] "Ltarget4: CAL"
## [38] "Rcontrol: Ind, Dep, Abun, vbK, vbLinf, vbt0, steep, MaxAge"
## [39] "Rcontrol2: Ind, Dep, Abun, vbK, vbLinf, vbt0, steep, MaxAge"
## [40] "SBT1: Ind"
## [41] "SBT2: Rec, Cref"
## [42] "SPMSY: L50, vbK, vbLinf, vbt0, MaxAge"
## [43] "SPSRA: Dep, vbK, vbLinf, vbt0, steep, MaxAge"
## [44] "SPSRA_ML: vbK, vbLinf, vbt0, steep, MaxAge, CAL"
## [45] "SPmod: Ind, Abun"
## [46] "SPslope: Ind, Abun"
## [47] "YPR: LFS, Abun, vbK, vbLinf, vbt0, wla, wlb, MaxAge"
## [48] "YPR_CC: LFS, CAA, vbK, vbLinf, vbt0, wla, wlb, MaxAge"
## [49] "YPR_ML: LFS, vbK, vbLinf, vbt0, wla, wlb, MaxAge, CAL"
## [50] "MRnoreal: MaxAge"
## [51] "MRreal: MaxAge"
## [52] "curE: MaxAge"
## [53] "curE75: MaxAge"
## [54] "matagelim: MaxAge"
```

The function TAC() automatically detects which MPs can be applied and calculates a TAC distribution for each MP, which can then be plotted.

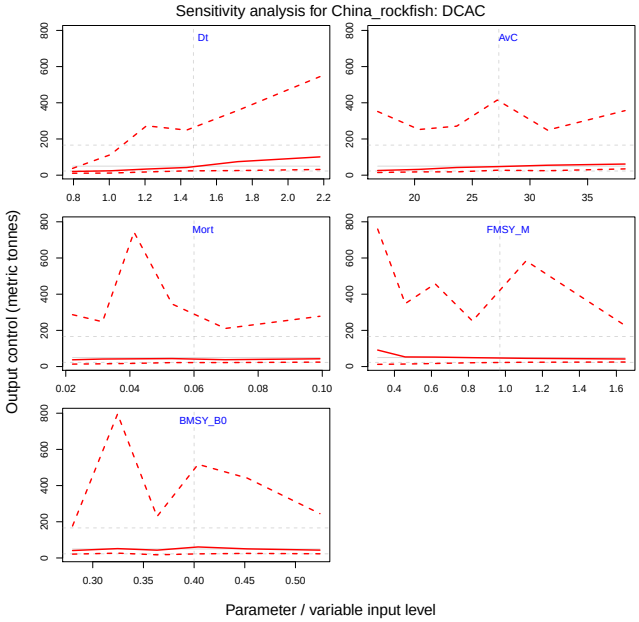
```
RockReal<-TAC(China_rockfish)


## [1] "Method DCAC produced greater than 50% NA values"
## [1] "Method DCAC4010 produced greater than 50% NA values"
```

```
plot(RockReal)
```



5.5 Conduct a sensitivity analysis

```
RockReal<-Sense(RockReal,"DCAC")
```



The sensitivity plot reveals which inputs to an MP most strongly affect the TAC recommendation. In principle this may help to focus data discussion on the most critical inputs and their credibility.

6 From MSE to management recommendations

In this section we take a more thorough, systematic approach to MSE and data implementation. This is an example of how DLMtool may be used to select MPs and then apply them to real data. This is intended to be a straw-man demonstration and in no way should this be interpreted as a recommendation about appropriate management objectives!

In this example our real-life stock is a moderately long-lived reef fish species of moderately high recruitment compensation that has been subject to fairly consistent fishing pressure over recent years. We suspect that fishing activities do not effectively operate on older age classes since the fish exhibit ontogenetic movements offshore where there is less fishing. In general the stock is thought to be a relatively low stock levels going by catch rate observations but frankly, we don't have a precise handle on stock depletion. Since fishing activities have changed spatial distribution and the stock is targetted there is the potential for hyperstability in our observations of catch rates over time.

This section assumes that you have completed the prerequisites (Section 3): .

6.1 Building an appropriate operating model

We start by specifying an operating model. Looking at the prebuilt stock objects we decide that the 'Snapper' stock object is the closest fit.

```
avail('Stock')

##  [1] "Albacore"     "Blue_shark"   "Bluefin_tuna" "Butterfish"
##  [5] "Herring"      "Mackerel"     "Porgy"        "Rockfish"
##  [9] "Snapper"      "Sole"         "Toothfish"

ourstock<-Snapper
```

We make some modifications to better suit our particular case study such as higher natural mortality rate, stock depletion (D) between 5 and 30 per cent of unfished levels and a candidate MPA (between 5 - 15 percent of unfished biomass) with retention (probability of staying in the MPA) of 80 - 99 percent. Remember to get help on the OM objects and their slots type class?OM (or the components of the OM: class?Stock, class?Fleet, class?Observation) at the command line.

```
ourstock@M<-c(0.2,0.25)
ourstock@maxage<-18
ourstock@D<-c(0.05,0.3)
ourstock@Frac_area_1<-c(0.05,0.15)
ourstock@Prob_staying<-c(0.4,0.99)
```

We now choose a fleet type for our operating model and choose to modify a generic fleet of flat recent effort, adding dome-shaped vulnerability as a possibility for older age classes:

```
ourfleet<-Generic_FlatE
ourfleet@Vmaxlen<-c(0.5,1)
```

Finally, Using our fleet and stock objects we construct an operating model object assuming that the data we have are likely to be imprecise and potentially biased. Type avail('Observation') at the command line to see the various pre-defined observation model objects.

```
ourOM<-new('OM',ourstock,ourfleet,Imprecise_Biased)
```

6.2 MSE evaluation of methods

Now that we have our operating model we run a trial MSE. In this case we use a very small number of simulations (20, which is low to meet CRAN-R package building requirements) but change the length of the projection and the length of the interval between updates to reflect our stock and management system.

Since we do not specify a vector of particular methods, the MSE will run for all possible MPs. Note that this could take a few minutes depending on how monstrous you computer is. Note that in a real setting it might be advisable to increase the number of simulations to at least 192 and, if stochastic MPs are to be used, increase the samples per method (reps) to at least 100 for this first stage to obtain stable aggregate results.

```
ourMSE<-runMSE(ourOM,proyears=20,interval=5,nsim=16,reps=1)

## [1] "Loading operating model"
## [1] "Optimizing for user-specified movement"
## [1] "Optimizing for user-specified depletion"
## [1] "Calculating historical stock and fishing dynamics"
## [1] "Calculating MSY reference points"
## [1] "Calculating reference yield - best fixed F strategy"
## [1] "Determining available methods"
## [1] "No MPs specified: running all available"
##  [1] "BK"          "BK_CC"       "CC1"         "CC4"         "CompSRA"
##  [6] "CompSRA4010" "DBSRA"       "DBSRA4010"   "DBSRA_40"    "DCAC"
## [11] "DCAC4010"    "DCAC_40"     "DD"          "DD4010"      "DepF"
## [16] "DynF"        "EDCAC"       "FMSYref"     "FMSYref50"   "FMSYref75"
## [21] "Fadapt"      "Fdem"        "Fdem_CC"     "Fratio"      "Fratio4010"
## [26] "Fratio_CC"   "GB_CC"       "GB_slope"    "GB_target"   "Gcontrol"
## [31] "Islope1"     "Islope4"     "Itarget1"    "Itarget4"    "LstepCC1"
## [36] "LstepCC4"    "Ltarget1"    "Ltarget4"    "MCD"         "MCD4010"
```

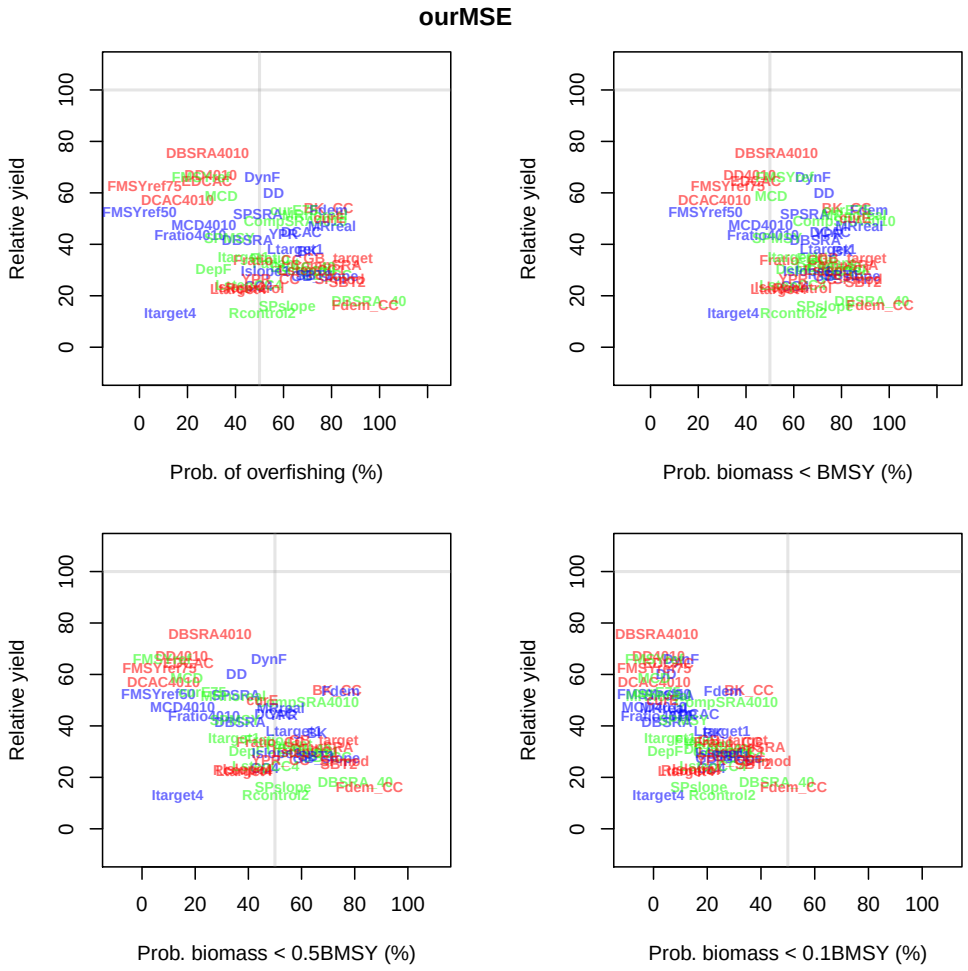
```
## [41] "Rcontrol"    "Rcontrol2"   "SBT1"        "SBT2"        "SPMSY"
## [46] "SPSRA"       "SPmod"       "SPslope"     "YPR"         "YPR_CC"
## [51] "MRnoreal"    "MRreal"      "curE"        "curE75"
## [1] "1/54 Running MSE for BK"
## .....................
## [1] "2/54 Running MSE for BK_CC"
## .....................
## [1] "3/54 Running MSE for CC1"
## .....................
## [1] "4/54 Running MSE for CC4"
## .....................
## [1] "5/54 Running MSE for CompSRA"
## .....................
## [1] "6/54 Running MSE for CompSRA4010"
## .....................
## [1] "7/54 Running MSE for DBSRA"
## .....................
## [1] "8/54 Running MSE for DBSRA4010"
## .....................
## [1] "9/54 Running MSE for DBSRA_40"
## .....................
## [1] "10/54 Running MSE for DCAC"
## .....................
## [1] "11/54 Running MSE for DCAC4010"
## .....................
## [1] "12/54 Running MSE for DCAC_40"
## .....................
## [1] "13/54 Running MSE for DD"
## .....................
## [1] "14/54 Running MSE for DD4010"
## .....................
## [1] "15/54 Running MSE for DepF"
## .....................
## [1] "16/54 Running MSE for DynF"
## .....................
## [1] "17/54 Running MSE for EDCAC"
## .....................
## [1] "18/54 Running MSE for FMSYref"
## .....................
## [1] "19/54 Running MSE for FMSYref50"
## .....................
## [1] "20/54 Running MSE for FMSYref75"
## .....................
## [1] "21/54 Running MSE for Fadapt"
## .....................
## [1] "22/54 Running MSE for Fdem"
## .....................
## [1] "23/54 Running MSE for Fdem_CC"
## .....................
## [1] "24/54 Running MSE for Fratio"
## .....................
## [1] "25/54 Running MSE for Fratio4010"
## .....................
## [1] "26/54 Running MSE for Fratio_CC"
## .....................
```

```
## [1] "27/54 Running MSE for GB_CC"
## .....................
## [1] "28/54 Running MSE for GB_slope"
## .....................
## [1] "29/54 Running MSE for GB_target"
## .....................
## [1] "30/54 Running MSE for Gcontrol"
## .....................
## [1] "31/54 Running MSE for Islope1"
## .....................
## [1] "32/54 Running MSE for Islope4"
## .....................
## [1] "33/54 Running MSE for Itarget1"
## .....................
## [1] "34/54 Running MSE for Itarget4"
## .....................
## [1] "35/54 Running MSE for LstepCC1"
## .....................
## [1] "36/54 Running MSE for LstepCC4"
## .....................
## [1] "37/54 Running MSE for Ltarget1"
## .....................
## [1] "38/54 Running MSE for Ltarget4"
## .....................
## [1] "39/54 Running MSE for MCD"
## .....................
## [1] "40/54 Running MSE for MCD4010"
## .....................
## [1] "41/54 Running MSE for Rcontrol"
## .....................
## [1] "42/54 Running MSE for Rcontrol2"
## .....................
## [1] "43/54 Running MSE for SBT1"
## .....................
## [1] "44/54 Running MSE for SBT2"
## .....................
## [1] "45/54 Running MSE for SPMSY"
## .....................
## [1] "46/54 Running MSE for SPSRA"
## .....................
## [1] "47/54 Running MSE for SPmod"
## .....................
## [1] "48/54 Running MSE for SPslope"
## .....................
## [1] "49/54 Running MSE for YPR"
## .....................
## [1] "50/54 Running MSE for YPR_CC"
## .....................
## [1] "51/54 Running MSE for MRnoreal"
## .....................
## [1] "52/54 Running MSE for MRreal"
## .....................
## [1] "53/54 Running MSE for curE"
## .....................
## [1] "54/54 Running MSE for curE75"
```

```
## ..................
```

A summary trade-off plot reveals a wide range of performance:

```
Tplot(ourMSE)
```



In this example process, we decide that we would like to select a targetted subset of these MPs that have greater than 50 percent of long-term best yield (given ideal fixed fishing mortality rate), less than a 50 percent probability of overfishing and less than a 20 percent probability of dropping below a low stock level, in this case 50 percent of BMSY. To do this we calculate the summary table and subset it:

```
Results<-summary(ourMSE)
Results

##             MP Yield stdev   POF stdev    P10 stdev   P50 stdev  P100
## 1           BK 37.26 26.59 70.00  32.61 22.19 30.44 65.94 35.88 80.62
## 2        BK_CC 53.92 81.84 78.75  35.47 35.62 27.92 73.44 37.93 82.19
## 3          CC1 29.74 29.70 73.12  41.83 31.88 28.34 65.62 40.24 79.06
## 4          CC4 23.82 20.47 49.69  49.18 21.56 27.49 46.25 43.07 60.62
## 5      CompSRA 31.49 34.29 77.81  26.52 35.94 36.80 66.25 37.75 80.62
## 6  CompSRA4010 48.83 58.80 65.00  25.50 27.19 33.11 62.19 32.30 81.25
```

```
## 7        DBSRA 41.66 41.20 45.00  29.61  5.00 14.26 36.88 24.21 69.06
## 8    DBSRA4010 75.67 75.54 28.44  21.73  1.25  2.89 25.62 21.67 52.81
## 9     DBSRA_40 17.91 10.53 95.62  17.50 45.94 27.52 80.31 26.23 92.81
## 10        DCAC 44.64 34.05 67.50  40.99 16.88 26.26 50.00 42.19 75.31
## 11    DCAC4010 57.34 45.20 15.94  25.57  0.31  1.25  8.12 11.81 26.88
## 12     DCAC_40 30.89 19.90 68.12  43.12 23.44 28.15 59.06 40.67 74.69
## 13          DD 60.11 35.33 55.94  37.69  4.69 16.17 35.62 35.68 72.81
## 14      DD4010 67.11 42.46 29.69  20.85  1.88  6.29 15.00 17.80 41.56
## 15        DepF 29.79 30.31 30.94  30.29  4.38  7.72 39.38 36.65 60.00
## 16        DynF 65.63 49.72 51.25  31.86 10.31 16.98 47.81 36.88 68.12
## 17       EDCAC 64.63 59.86 28.12  30.54  5.62 14.71 17.50 24.90 44.06
## 18     FMSYref 66.19 17.73 25.62  14.82  0.31  1.25  7.50  9.49 56.25
## 19   FMSYref50 52.56 13.36  0.00   0.00  0.31  1.25  6.25  7.42 23.12
## 20   FMSYref75 62.56 15.63  2.19   3.64  0.31  1.25  6.56  8.11 32.50
## 21      Fadapt 32.37 27.94 56.88  38.72 20.62 32.35 55.31 38.96 75.94
## 22        Fdem 53.46 45.60 78.75  26.17 25.94 32.21 74.69 31.17 91.56
## 23     Fdem_CC 16.01  9.61 94.06  14.05 52.19 26.83 85.62 20.16 96.25
## 24      Fratio 34.71 24.37 54.38  43.32 15.31 25.00 45.94 40.79 69.69
## 25  Fratio4010 43.67 52.91 21.25  20.45  1.25  2.89 23.44 21.35 47.19
## 26   Fratio_CC 33.62 54.90 53.44  45.16 27.81 35.87 48.44 42.53 60.00
## 27       GB_CC 27.78 23.31 78.12  40.33 32.50 30.11 69.38 42.07 81.56
## 28    GB_slope 27.22 17.37 77.19  38.90 28.12 31.14 69.38 40.16 82.19
## 29   GB_target 34.21 30.26 82.81  36.70 29.69 30.30 68.44 39.78 84.38
## 30    Gcontrol 29.85 30.70 69.38  41.23 25.94 22.60 62.19 39.75 77.81
## 31     Islope1 29.03 22.06 52.50  49.36 24.69 29.80 50.62 44.30 65.94
## 32     Islope4 22.91 15.90 41.56  48.05 16.25 24.60 39.69 42.76 54.06
## 33    Itarget1 34.74 24.31 43.44  46.32  6.25 12.18 34.69 37.70 60.00
## 34    Itarget4 12.74 18.18 12.81  29.78  1.88  6.29 13.44 24.61 34.69
## 35    LstepCC1 29.47 23.88 71.88  40.45 29.06 31.05 61.56 41.86 78.44
## 36    LstepCC4 23.81 16.45 45.62  49.76 22.19 27.99 46.56 44.19 59.69
## 37    Ltarget1 37.49 34.66 65.00  43.01 25.62 30.10 57.19 43.36 74.38
## 38    Ltarget4 22.10 18.78 41.25  48.32 12.19 24.08 38.44 42.92 53.75
## 39         MCD 58.74 47.37 34.06  31.48  0.31  1.25 16.88 16.42 50.62
## 40     MCD4010 47.48 54.65 26.88  29.66  0.31  1.25 15.31 16.78 46.25
## 41    Rcontrol 22.92 15.88 48.12  48.71 13.75 18.30 37.81 39.50 63.75
## 42   Rcontrol2 13.24  8.87 51.25  50.41 25.62 30.98 50.31 46.89 60.00
## 43        SBT1 28.31 19.24 72.81  40.04 29.06 31.00 65.62 40.45 80.00
## 44        SBT2 25.24 20.73 86.56  33.90 37.19 31.99 74.06 39.72 89.06
## 45       SPMSY 42.50 40.71 37.50  39.92 11.25 24.53 35.00 37.77 53.12
## 46       SPSRA 51.96 33.44 49.38  34.59  5.94 16.25 35.31 33.79 64.69
## 47       SPmod 26.46 29.06 83.44  31.40 42.19 33.32 76.25 37.79 86.88
## 48     SPslope 15.61 17.71 61.56  38.33 16.88 23.73 53.12 34.83 73.12
## 49         YPR 44.22 29.91 60.00  40.17 10.94 24.03 53.12 40.41 74.69
## 50      YPR_CC 26.37 29.33 54.69  43.65 26.88 27.26 52.19 41.23 65.94
## 51    MRnoreal 51.51 14.96 73.12  32.96  1.88  6.29 34.38 33.11 85.31
## 52      MRreal 46.98 14.03 80.31  32.89  3.75 13.72 52.19 33.81 87.50
## 53        curE 50.23 15.26 79.06  33.77  3.44 12.48 45.31 33.54 85.94
## 54      curE75 53.19 17.25 64.38  35.91  1.25  3.87 22.81 31.04 82.19
##    stdev  LTY  STY    VY
## 1  32.04 25.0 35.0  12.5
## 2  30.49 19.4 24.4  18.8
## 3  32.10 22.5 34.4  37.5
## 4  38.90 12.5 29.4  43.8
## 5  30.87 30.0 31.9  18.8
## 6  27.23 21.9 38.8   0.0
```

```
## 7  25.70 36.9 35.6  18.8
## 8  29.94 34.4 23.1   NaN
## 9  19.06 10.6 48.1  31.2
## 10 33.34 42.5 50.6  68.8
## 11 20.81 50.0 18.8  25.0
## 12 35.42 34.4 53.8  68.8
## 13 33.37 63.7 40.0  56.2
## 14 21.81 62.5 36.9  25.0
## 15 39.33 28.7 20.0  18.8
## 16 34.20 37.5 25.6   0.0
## 17 28.71 39.4 16.9  12.5
## 18 26.11 81.2 40.6 100.0
## 19 15.26 50.0 18.8 100.0
## 20 20.00 78.1 26.2 100.0
## 21 35.97 28.1 20.0  12.5
## 22 22.11 28.7 31.2   0.0
## 23 10.08  8.8 35.0   6.2
## 24 37.79 36.2 20.6  31.2
## 25 30.44 31.2 23.1   NaN
## 26 36.38 15.6 24.4  25.0
## 27 34.48 16.2 32.5  18.8
## 28 29.94 19.4 35.0  31.2
## 29 32.91 22.5 35.6  25.0
## 30 29.44 12.5 34.4  12.5
## 31 38.65 25.6 37.5  56.2
## 32 39.25  6.2 23.1  68.8
## 33 37.99 32.5 21.9  56.2
## 34 26.74  6.2  0.0  75.0
## 35 34.09 18.8 33.1  50.0
## 36 39.22  6.9 28.7  56.2
## 37 37.19 24.4 33.8  43.8
## 38 39.18  9.4 27.5  62.5
## 39 30.76 43.1 36.2  37.5
## 40 27.78 35.6 37.5   NaN
## 41 36.03 16.2 36.9  50.0
## 42 41.87  3.1 21.2  37.5
## 43 31.09 17.5 33.1  37.5
## 44 29.45 19.4 46.9  31.2
## 45 35.44 26.9 28.1  18.8
## 46 35.28 36.2 35.0  12.5
## 47 23.73 13.8 37.5  12.5
## 48 31.08 16.9 29.4   NaN
## 49 35.89 33.8 29.4  12.5
## 50 38.13 16.2 29.4  31.2
## 51 30.90 44.4 50.6   6.2
## 52 30.06 40.6 51.9   6.2
## 53 31.10 45.0 51.9   6.2
## 54 33.32 46.2 48.1   0.0

Targetted<-subset(Results, Results$Yield>50 & Results$POF<50 & Results$P50<20)
Targetted

##           MP Yield stdev   POF stdev   P10 stdev.1   P50 stdev.2  P100
## 11  DCAC4010 57.34 45.20 15.94  25.57 0.31    1.25  8.12   11.81 26.88
## 14    DD4010 67.11 42.46 29.69  20.85 1.88    6.29 15.00   17.80 41.56
```

```
## 17     EDCAC 64.63 59.86 28.12  30.54 5.62   14.71 17.50   24.90 44.06
## 18   FMSYref 66.19 17.73 25.62  14.82 0.31    1.25  7.50    9.49 56.25
## 19 FMSYref50 52.56 13.36  0.00   0.00 0.31    1.25  6.25    7.42 23.12
## 20 FMSYref75 62.56 15.63  2.19   3.64 0.31    1.25  6.56    8.11 32.50
## 39       MCD 58.74 47.37 34.06  31.48 0.31    1.25 16.88   16.42 50.62
##    stdev.3  LTY  STY    VY
## 11   20.81 50.0 18.8  25.0
## 14   21.81 62.5 36.9  25.0
## 17   28.71 39.4 16.9  12.5
## 18   26.11 81.2 40.6 100.0
## 19   15.26 50.0 18.8 100.0
## 20   20.00 78.1 26.2 100.0
## 39   30.76 43.1 36.2  37.5
```

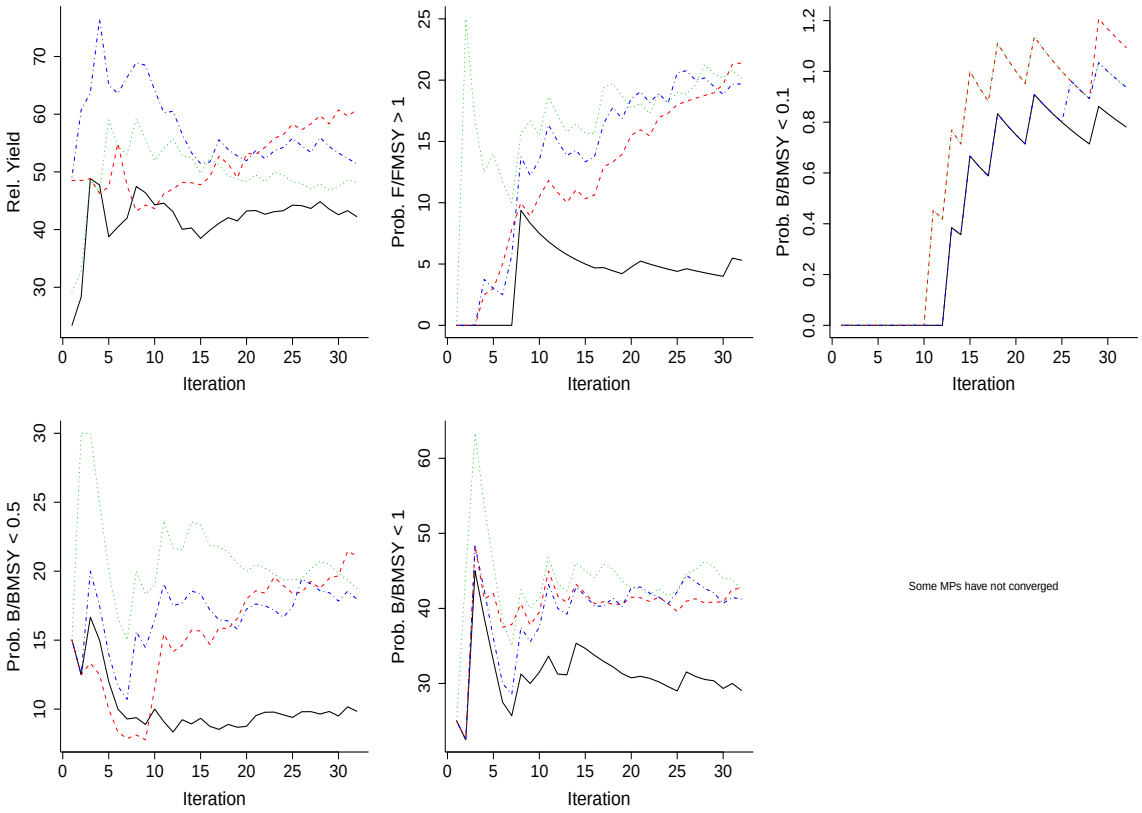
Our new subsetted methods can be used to run a more focused MSE that includes a greater number of simulations for a detailed assessment of performance. Again note that in a real setting it would be advisable to increase the number of simulations further to at least 400.You might also want to increase the number of stochastic samples per method (reps) to 200 or more.

```
TargMP<-Targetted$MP[grep("FMSYref",Targetted$MP,invert=T)]
ourMSE2<-runMSE(ourOM,TargMP,proyears=20,interval=5,nsim=32,reps=1)
```

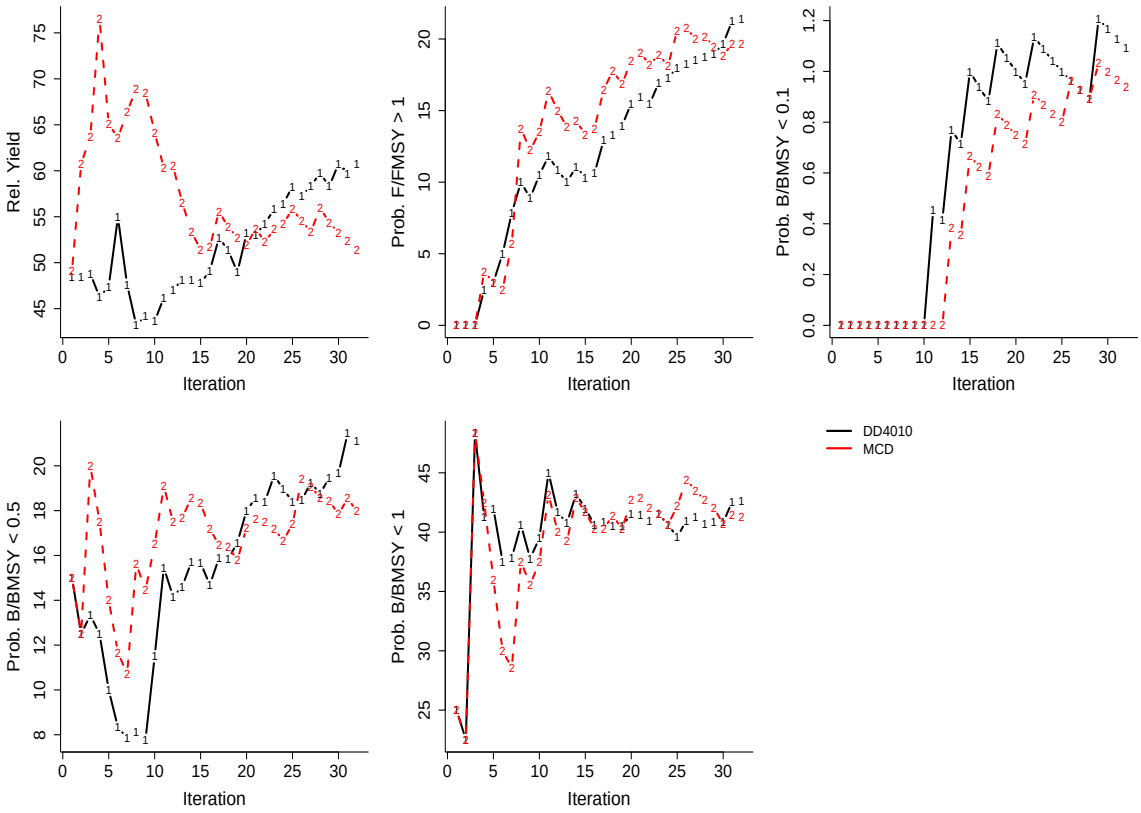
```
## [1] "Loading operating model"
## [1] "Optimizing for user-specified movement"
## [1] "Optimizing for user-specified depletion"
## [1] "Calculating historical stock and fishing dynamics"
## [1] "Calculating MSY reference points"
## [1] "Calculating reference yield - best fixed F strategy"
## [1] "Determining available methods"
## [1] "1/4 Running MSE for DCAC4010"
## ..................
## [1] "2/4 Running MSE for DD4010"
## ..................
## [1] "3/4 Running MSE for EDCAC"
## ..................
## [1] "4/4 Running MSE for MCD"
## ..................
```

Lets check convergence in some performance metrics as simulations are added (the first plot is for all MPs, the second shows only those that did not converge):

```
CheckConverg(ourMSE2)
```

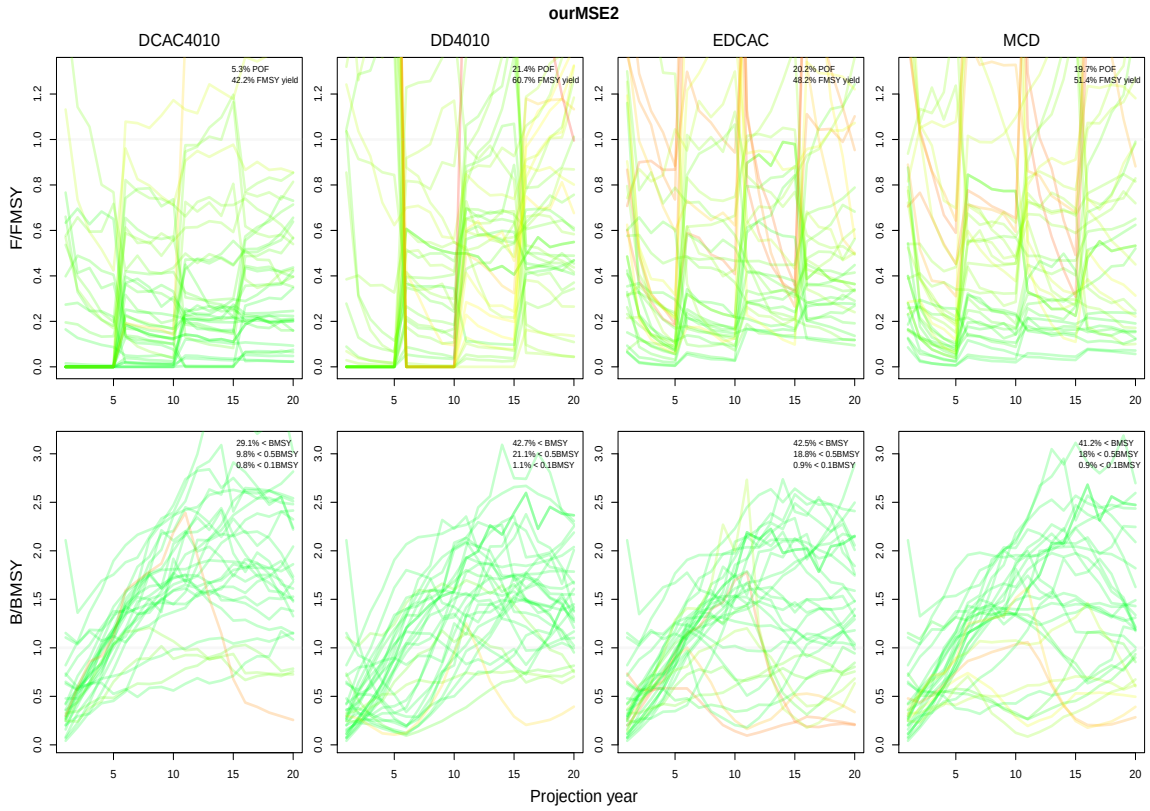



```
## Some MPs may not have converged in 32 iterations (threshold = 2%)
## MPs are:  DD4010  MCD
## MPs #:  2  4

##   Num     MP
## 1   2 DD4010
## 2   4    MCD
```

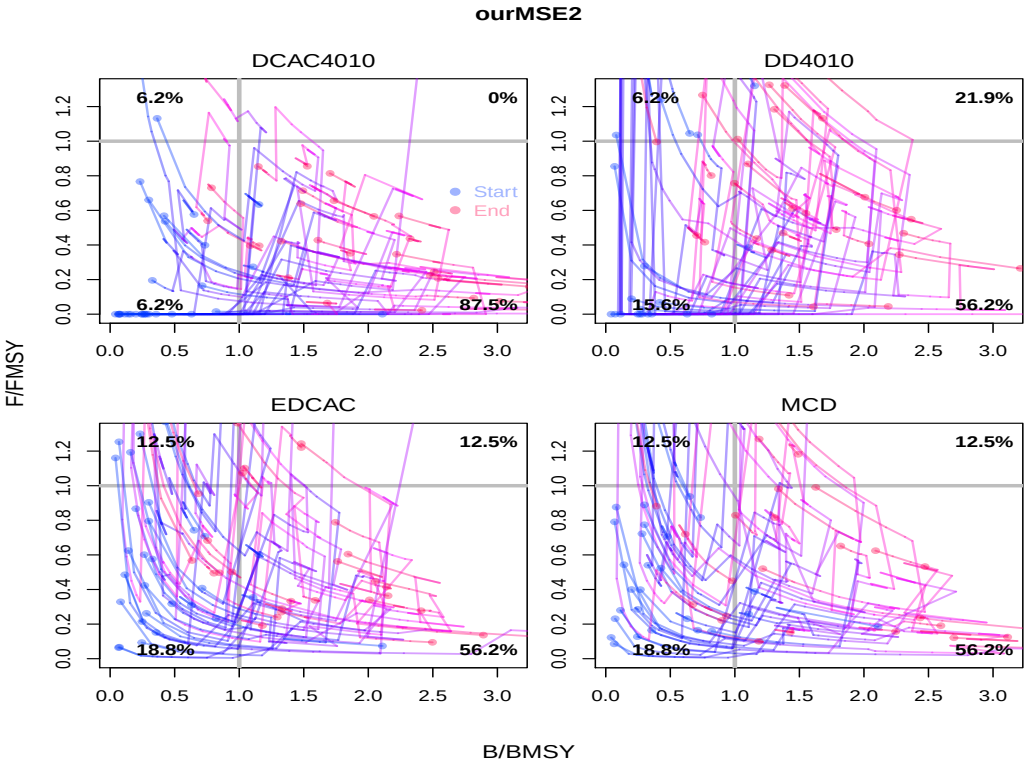


Several detailed plots can provide greater information about exactly how each MP performed over the projected time period including Projection and Kobe plots:

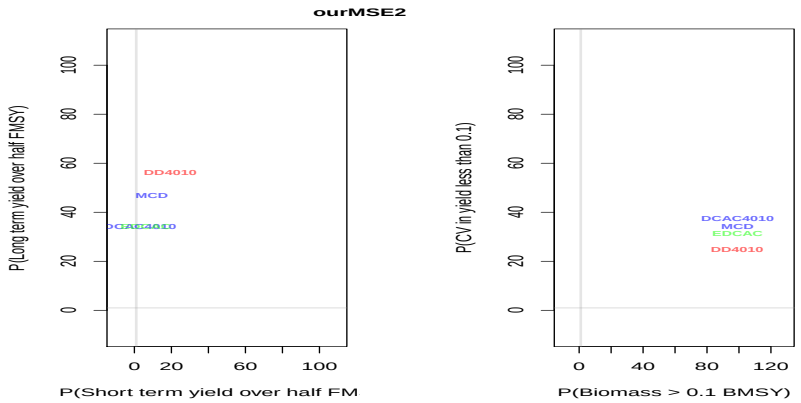
```
Pplot(ourMSE2)
```




```
Kplot(ourMSE2)
```



```
Tplot2(ourMSE2)
```



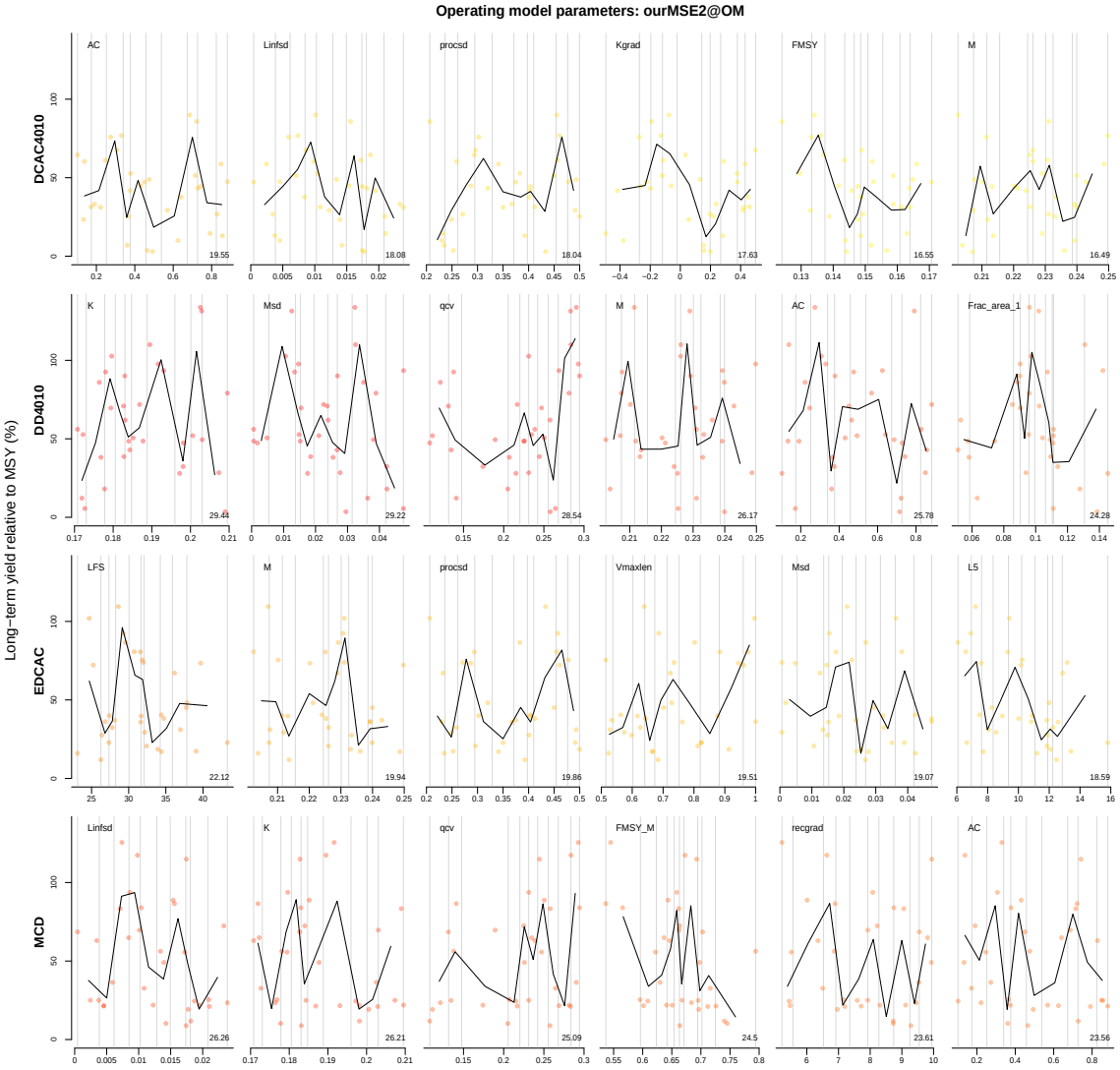
These plots indicate that the depletion based F MPs based on a fixed ratio of FMSY to M (DepF) and the simple delay-difference model (DD) linked ot the 40-10 harvest control rule (DD4010) are at the upper limit of the trade-off space (ie all other MPs provide worse performance in one or more dimensions). In this case the MPs offer a contrasting trade-off between probability of overfishing and long-term yield. It remains to be seen whether these approaches can be applied to the real data for our reef fish

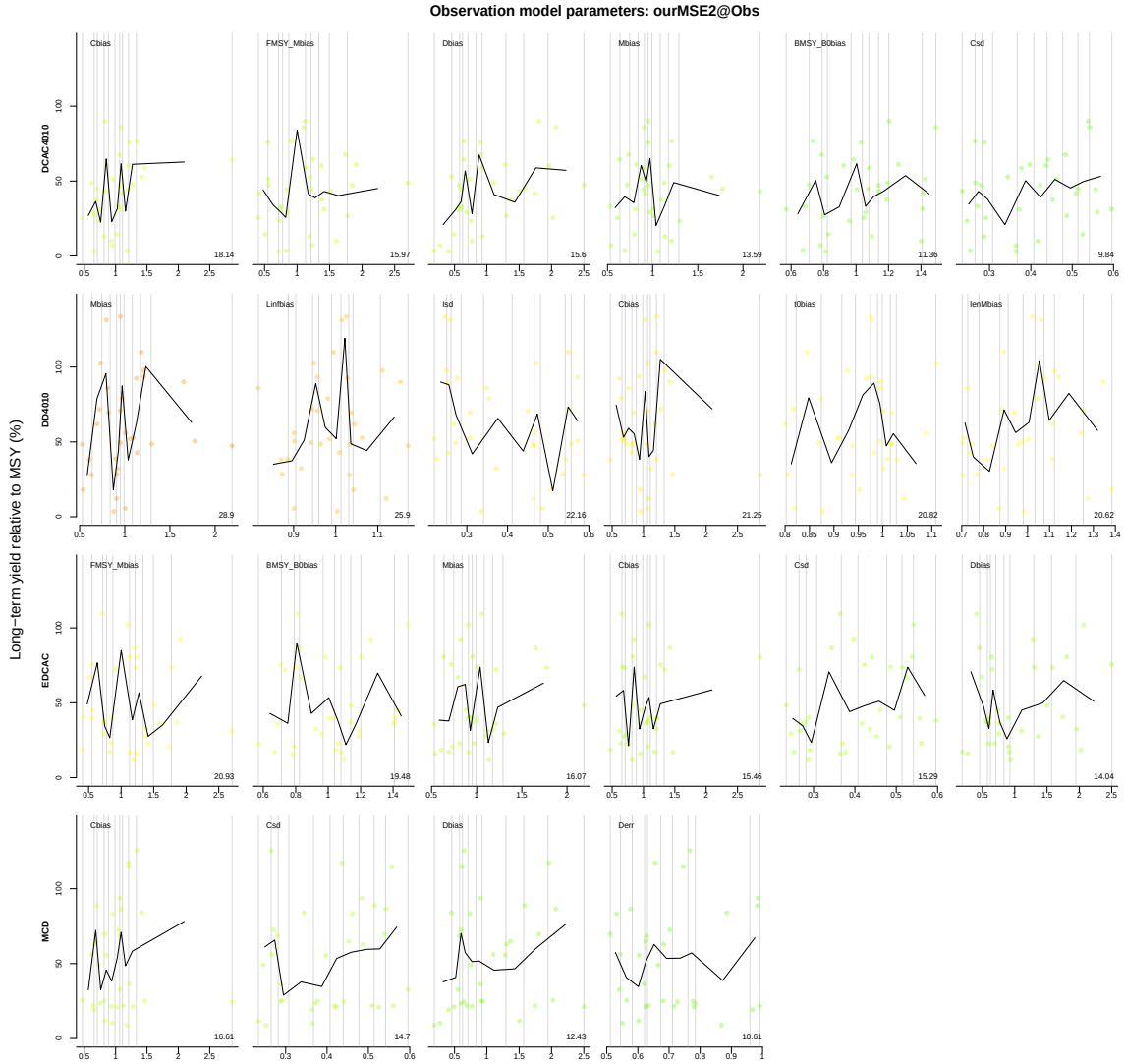
6.3 Value of information analysis

A value of information function is available that allows users to establish which of the sampled parameters of the operating model or the observation model are most correlated with performance. This helps to guide future data collection effort to target those inputs that are most critical for performance. The VOI function also provides a metric of the robustness of MPs: while an MP's aggregate mean performance may be quite good it might be concerning if performance was strongly compromised given alternative plausible scenarios.

The inputs are organized in order of most correlated to least correlated from left to right. The label of the plots indicates a sampled parameter in either ourMSE2@OM (operating model parameters) or ourMSE2@Obs (observation model parameters). The help file for these slots provides details on how to interpret these labels. For example Mbias is bias in the observed value of natural mortality rate, Dbias is bias in the observed value of stock depletion. Note that in a real analysis many more simulations should be undertaken to provide a reliable performance pattern. Ie nsim should be higher when using runMSE().

```
VOI(ourMSE2)
```


Operating model parameters: ourMSE2@OM
Long−term yield relative to MSY (%)
DCAC4010
DD4010
EDCAC
MCD
AC
Linfsd
procsd
Kgrad
FMSY
M
K
Msd
qcv
M
AC
Frac_area_1
LFS
M
procsd
Vmaxlen
Msd
L5
Linfsd
K
qcv
FMSY_M
recgrad
AC



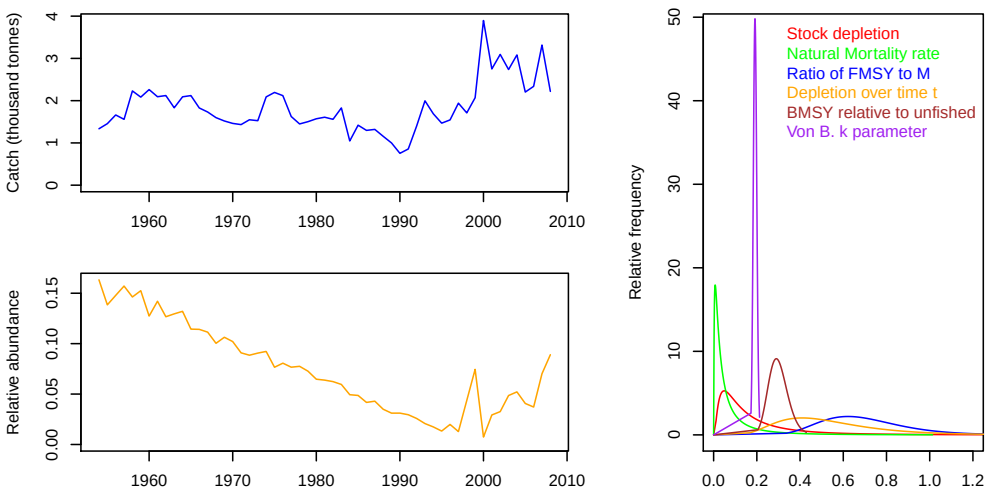

```
## [[1]]
##         MP      1      2      3       4       5          6
## 1 DCAC4010     AC Linfsd procsd   Kgrad    FMSY          M
## 2           19.55  18.08  18.04   17.63   16.55      16.49
## 3   DD4010      K    Msd    qcv       M      AC Frac_area_1
## 4           29.44  29.22  28.54   26.17   25.78      24.28
## 5    EDCAC    LFS      M procsd Vmaxlen     Msd         L5
## 6           22.12  19.94  19.86   19.51   19.07      18.59
## 7      MCD Linfsd      K    qcv  FMSY_M recgrad         AC
## 8           26.26  26.21  25.09    24.5   23.61      23.56
##
## [[2]]
##         MP          1           2     3     4          5        6
## 1 DCAC4010      Cbias  FMSY_Mbias Dbias Mbias BMSY_B0bias      Csd
## 2              18.14       15.97  15.6 13.59       11.36     9.84
## 3   DD4010      Mbias    Linfbias   Isd Cbias      t0bias lenMbias
## 4               28.9        25.9 22.16 21.25       20.82    20.62
## 5    EDCAC FMSY_Mbias BMSY_B0bias Mbias Cbias         Csd    Dbias
## 6              20.93       19.48 16.07 15.46       15.29    14.04
## 7      MCD      Cbias         Csd Dbias  Derr
```

```
## 8                16.61         14.7 12.43 10.61
```

6.4 Applying MPs to our real data

A real DLM_data object 'ourReefFish', was loaded into the current R session when we loaded up the data in the Prerequisites section above. We can summarise some of the data in this DLM_data object using the generic function summary():

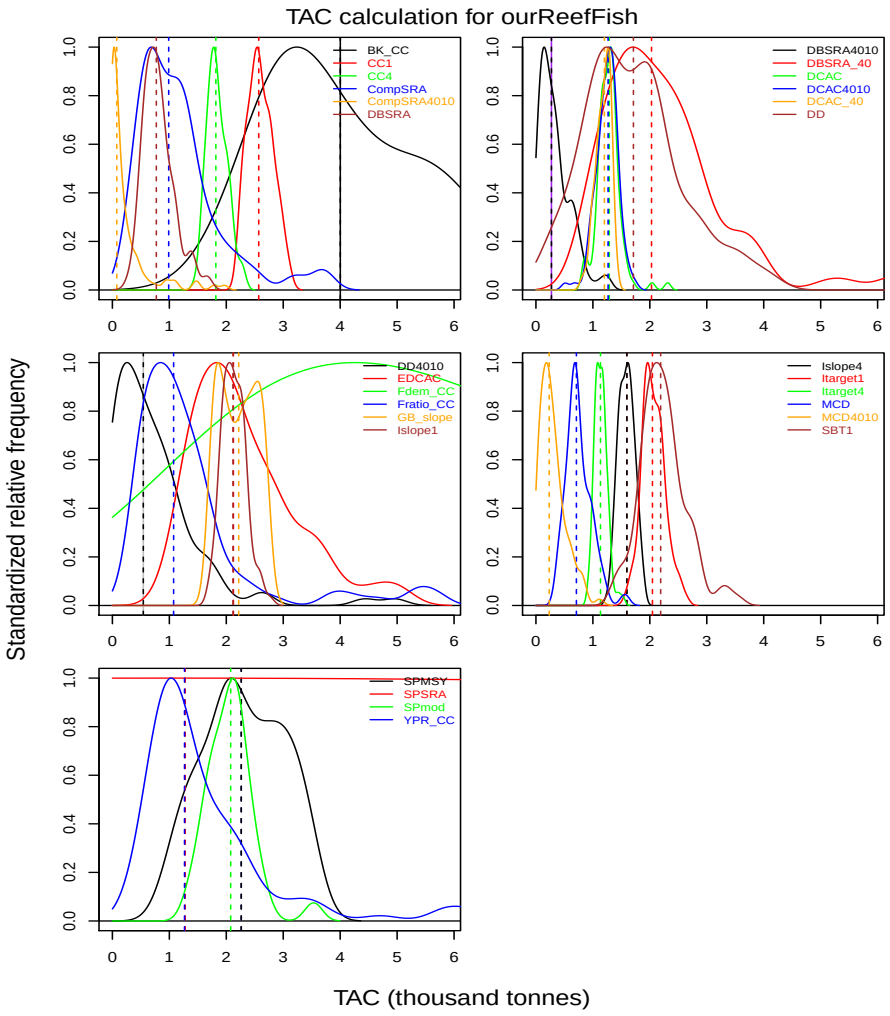
```
summary(ourReefFish)
```



The Can() function reveals that a range of MPs are available but not the three best performing MPs identified by the MSE (DynF, Fratio, YPR). Nonetheless we can calculate and plot the TACs for the available methods:

```
ourReefFish<-TAC(ourReefFish)

## Warning in rlnorm(reps, mconv(mu, mu * cv), sdconv(mu, mu * cv)):  NAs produced

## Warning in rlnorm(reps, mconv(mu, mu * cv), sdconv(mu, mu * cv)):  NAs produced

## [1] "Method SPmod produced greater than 50% NA values"
```

```
plot(ourReefFish)
```

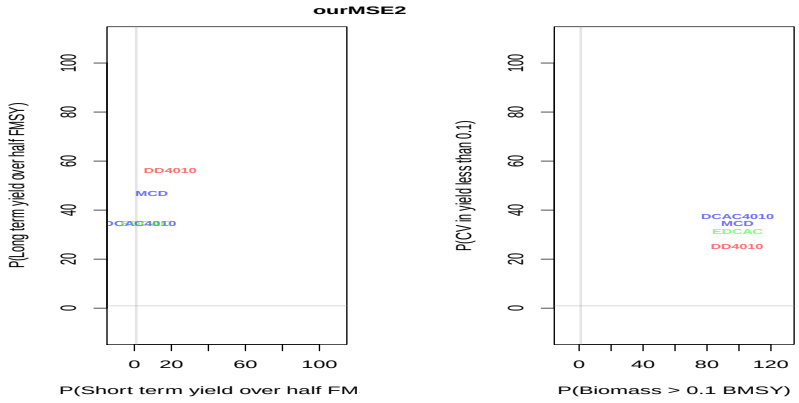


The Needed() function shows that there is only one data requirement to make each of these remaining MPs work, a current estimate of abundance (Abun, a slot in the DLM_data object. See class?DLM). While this may seem like rather a demanding data requirement it is worth remembering that DynF, Fratio and YPR outperformed the remaining methods on average despite fairly poor current abundance information that could have observation error with a CV of up to 100 per cent and a bias sampled from a uniform-on-log distribution distribution between 1/5 and 5:

```
ourOM@Btcv

## [1] 0.5 1.0

ourOM@Btbias

## [1] 0.2 5.0
```

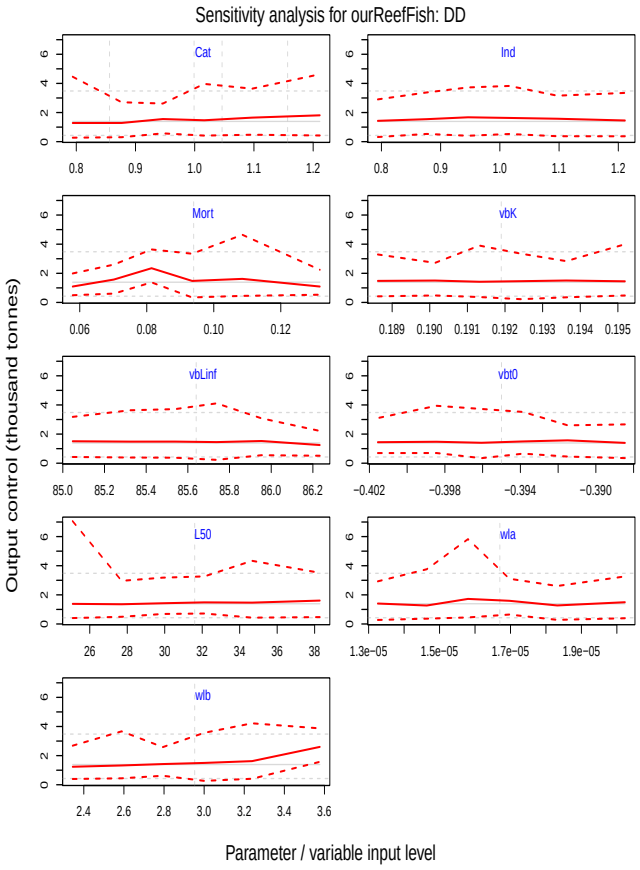
In other words our MSE generated observations of current biomass that could easily be 1/5 or five times the true level across the whole time series and were very noisy. The question now is whether gathering such data would be worthwhile (e.g. a systematic fishery independent survey). To refresh our memory we can re-plot he tradeoffs of the targetted MSE:

```
Tplot2(ourMSE2)
```



If we focus on output controls there are a cluster of MPs that offer comparable performance that are available for our real data such as the delay-difference stock assessment (DD). We can use sensitivity testing to better understand how fragile TAC recommendations are to changes in our data inputs:

```
ourReefFish<-Sense(ourReefFish,'DD')
```



In this case it is a toss-up between them. Both show primary sensitivity to bias in reported catches (fairly obviously) and little sensitivity to the remaining inputs. In this case it might be necessary to try an alternative run of the operating model given other credible parameters to see if we can distinguish between these MPs.

6.5 What have we learned?

In this simple walkthrough we have established what MPs work best for our stock, fishery and observation type. It was possible to establish the frailties of these MPs by examining what simulated parameters drive yield and probability of overfishing (using the VOI() function). Our application to real data produced actual TAC recommendations for MPs that were available. At least three MPs could not be applied that the MSE indicated could provide benefits in terms of both yield and limiting overfishing. We know what data are necessary to make these work but have yet to decide whether collecting these data is worthwhile. Above all, the approach is transparent and reproducible.

Depending on how utility is characterised, it may be possible to establish the cost-efficacy of future data-collection based on the long-term yield differential of the methods that are available and those that need additional data.

7 Designing new methods

DLMtool was designed to be extensible in order to promote the development of new MPs. In this section we design a series of new MPs that include spatial controls and input controls in the form of age-restrictions. The central requirement of any MP is that it can be applied to a DLM_data object using the function sapply (sfSapply() in parallel processing).

DLM_data objects have a single position x for each data entry, e.g. one value for natural mortality rate, a single vector of historical catches etc. In the MSE analysis this is extended to nsim positions. It follows that any MP arranged to function sapply(x,MP,DLM_data) will work. For example we can get 5 stochastic samples of the TAC for the demographic FMSY MP paired to catch-curve analysis FdemCC applied to a real data-limited data object for red snapper using:

```
sapply(1,Fdem_CC,Red_snapper,reps=5)
```

```
##           [,1]
## [1,]  7.045566
## [2,]  7.282668
## [3,] 11.477060
## [4,] 27.047357
## [5,] 11.322460
```

The MSE just populates a DLM_data object with many simulations and uses sfSapply() (snowfall cluster computing equivalent) to calculate an management recommendation for each simulation. By making methods compatible with this standard the very same equations are used in both the MSE and the real management advice.

The following new MPs illustrate this.

7.1 Average historical catch MP

The average historical catch has been suggested as a starting point for setting TACs in the most data-limited situations (following Restrepo et al. 1998). Here we design such an MP:

```
AvC<-function(x,DLM_data,reps)rlnorm(reps,log(mean(DLM_data@Cat[x,],na.rm=T)),0.1)
```

Note that all MPs have to be stochastic in this framework which is why we sample from a log-normal distribution with a CV of roughly 10 per cent.

Before the MP can be 'seen' by the rest of the DLM package we have to do three more things. The MP must be assigned a class based on what outputs it provides. Since this is an output control (TAC) based MP we assign

it class 'DLM_output'. The MP must also be assigned to the DLMtool namespace and -if we are using parallel computing- exported to the cluster:

```
class(AvC)<-"DLM_output"
environment(AvC) <- asNamespace('DLMtool')
#sfExport("AvC")
```

7.2 Third-highest catch

In some data-limited settings third highest historical catch has been suggested as a possible catch-limit. Here we use a similar approach to the average catch MP above (AvC) and take draws from a log-normal distribution with CV of 10 per cent:

```
THC<-function(x,DLM_data,reps){
  rlnorm(reps,log(DLM_data@Cat[x,order(DLM_data@Cat[x,],decreasing=T)[3]]),0.1)
}
class(THC)<-"DLM_output"
environment(THC) <- asNamespace('DLMtool')
#sfExport("THC")
```

7.3 Fishing starting at age 5

To simulate input controls that aim to alter the age-vulnerability to fishing it is possible to design an MP of class 'DLM_input'. These simply describe a vector of fractional vulnerability (0-1) across ages. In this example we freeze fishing on age-classes 1-4 and maintain fishing for ages 5+:

```
agelim5<-function(x,DLM_data){
  Allocate<-1 # Fraction of effort reallocated to open area
  Effort<-1  # Fraction of effort in last historical year
  Spatial<-c(1,1) # Fraction of effort found in each area
  Vuln<-c(rep(0,4),rep(1,DLM_data@MaxAge-4)) # Age vulnerability
  c(Allocate, Effort, Spatial, Vuln) # Input controls stitched togther
}
class(agelim5)<-"DLM_input"
environment(agelim5) <- asNamespace('DLMtool')
#sfExport("agelim5")
```

Note that for compatibility, these approaches still require an 'x' argument even if they don't make use of it (ie they are the same regardless of the data or simulated data).

7.4 Reducing fishing rate in area 1 by 50 per cent

Spatial controls operate similarly to the age/size based controls: a vector of length 2 (the spatial simulator is a 2-box model) that indicates the fraction of current spatial catches. In this example we reduce catches in area 1 by 50 percent and assign the MP class 'DLM space'.

```
area1_50<-function(x,DLM_data){
  Allocate<-0 # Fraction of effort reallocated to open area
  Effort<-1  # Fraction of effort in last historical year
  Spatial<-c(0.5,1) # Fraction of effort found in each area
```

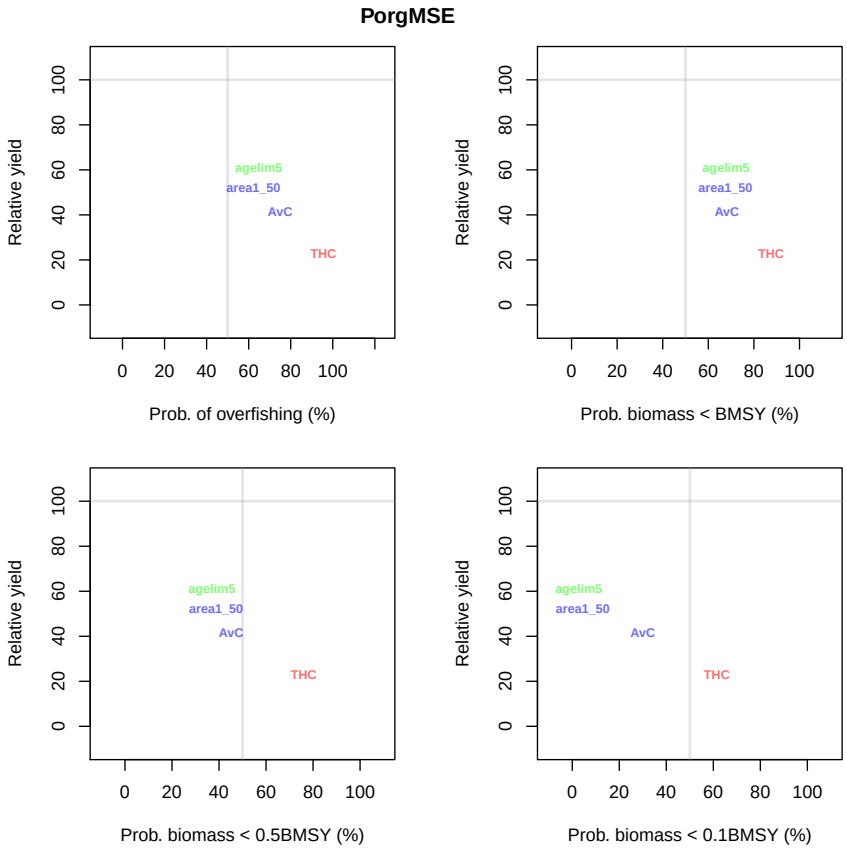
```
  Vuln<-rep(NA,DLM_data@MaxAge) # Age vulnerability is not specified
  c(Allocate, Effort, Spatial, Vuln) # Input controls stitched togther
}
class(area1_50)<-"DLM_input"
environment(area1_50) <- asNamespace('DLMtool')
#sfExport("area1_50")
```

7.5 Applying the new MPs

Our MPs are now compatible with all of the DLMtool functionality. Lets run a quick MSE and see how they fare:

```
new_MPs<-c("AvC","THC","agelim5","area1_50")
OM<-new('OM',Porgy, Generic_IncE, Imprecise_Unbiased)
PorgMSE<-runMSE(OM,new_MPs,maxF=1,nsim=20,reps=1,proyears=20,interval=5)

## [1] "Loading operating model"
## [1] "Optimizing for user-specified movement"
## [1] "Optimizing for user-specified depletion"
## [1] "Calculating historical stock and fishing dynamics"
## [1] "Calculating MSY reference points"
## [1] "Calculating reference yield - best fixed F strategy"
## [1] "Determining available methods"
## [1] "1/4 Running MSE for AvC"
## .....................
## [1] "2/4 Running MSE for THC"
## .....................
## [1] "3/4 Running MSE for agelim5"
## .....................
## [1] "4/4 Running MSE for area1_50"
## .....................
```

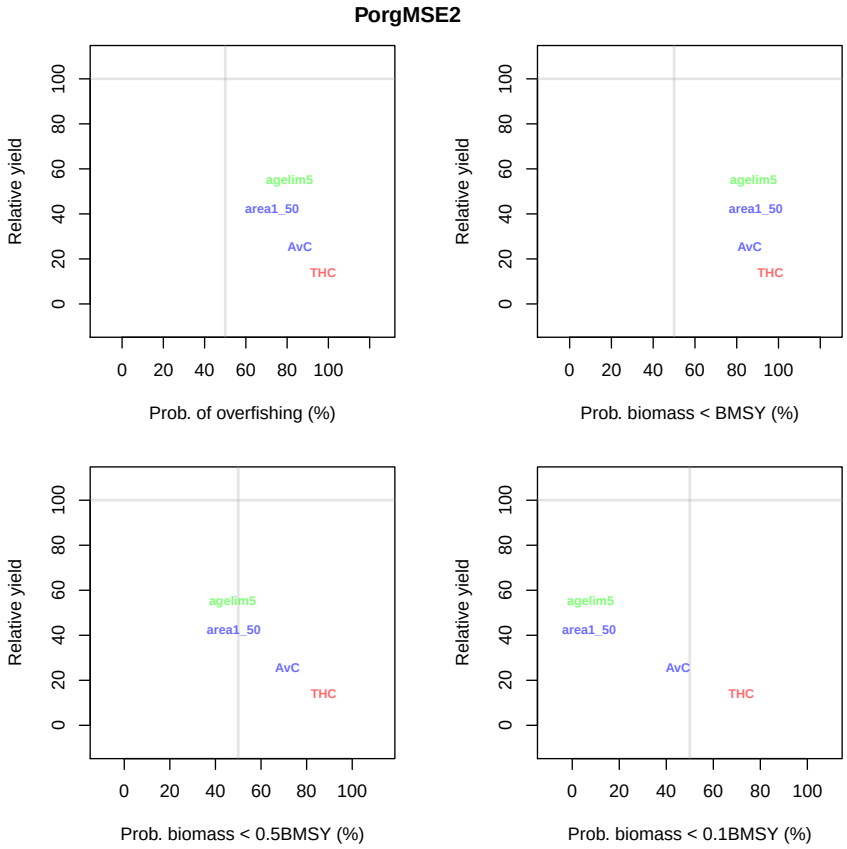
```
Tplot(PorgMSE)
```



What if starting depletion were different, e.g. likely to be under BMSY?

```
OM@D

## [1] 0.05 0.60

OM@D<-c(0.05,0.3)
PorgMSE2<-runMSE(OM,new_MPs,maxF=1,nsim=16,reps=1,proyears=20,interval=5)

## [1] "Loading operating model"
## [1] "Optimizing for user-specified movement"
## [1] "Optimizing for user-specified depletion"
## [1] "Calculating historical stock and fishing dynamics"
## [1] "Calculating MSY reference points"
## [1] "Calculating reference yield - best fixed F strategy"
## [1] "Determining available methods"
## [1] "1/4 Running MSE for AvC"
## .....................
## [1] "2/4 Running MSE for THC"
## .....................
## [1] "3/4 Running MSE for agelim5"
## .....................
## [1] "4/4 Running MSE for area1_50"
## .....................
```

```
Tplot(PorgMSE2)
```



Conveniently putting aside the likelihood of implementing a perfect knife-edge vulnerability to fishing at age 5, it appears that we have a winner in agelim5 even under different starting depletion levels. Third highest catch on the other hand appears risky to say the least. You could try some other starting depletion levels to see under what circumstances the trade-off space changes dramatically.

8 Managing real data

DLMtool has a series of functions to make importing data and applying data-limited MPs relatively straightforward. There are two approaches: (1) fill out a .csv data file in excel or a text editor and use a DLMtool function to create a properly formatted DLM_data object (class DLM_data) or (2) create a blank DLM_data object in R and populate it in R.

8.1 Importing data

Probably the easiest way to get your data into the DLMtool is to populate a .csv datafile. These files have a line for each slot of the DLM_data object e.g:

```
slotNames('DLM_data')

##  [1] "Name"       "Year"       "Cat"        "Ind"        "Rec"
##  [6] "t"          "AvC"        "Dt"         "Mort"       "FMSY_M"
```

```
## [11] "BMSY_B0"    "Cref"       "Bref"       "Iref"       "L50"
## [16] "L95"        "LFC"        "LFS"        "CAA"        "Dep"
## [21] "Abun"       "vbK"        "vbLinf"     "vbt0"       "wla"
## [26] "wlb"        "steep"      "CV_Cat"     "CV_Dt"      "CV_AvC"
## [31] "CV_Ind"     "CV_Mort"    "CV_FMSY_M"  "CV_BMSY_B0" "CV_Cref"
## [36] "CV_Bref"    "CV_Iref"    "CV_Rec"     "CV_Dep"     "CV_Abun"
## [41] "CV_vbK"     "CV_vbLinf"  "CV_vbt0"    "CV_L50"     "CV_LFC"
## [46] "CV_LFS"     "CV_wla"     "CV_wlb"     "CV_steep"   "sigmaL"
## [51] "MaxAge"     "Units"      "Ref"        "Ref_type"   "Log"
## [56] "params"     "PosMPs"     "MPs"        "OM"         "Obs"
## [61] "TAC"        "TACbias"    "Sense"      "CAL_bins"   "CAL"
## [66] "MPrec"      "LHYear"
```

You do not have to enter data for every line of the data file, if data are not available simply put an 'NA' next to any given field.

A number of example .csv files can be found in the directory where the DLMtool package was installed:

```
DLMDataDir()


## [1] "C:/Users/Tom Carruthers/AppData/Local/Temp/RtmpkdAE8j/Rinstd5c14401678/DLMtool/"
```

To get data from a .csv file you need only specify its location e.g new('DLM_data',"I:/Mackerel.csv"):

8.2 Populating a DLM_data object in R

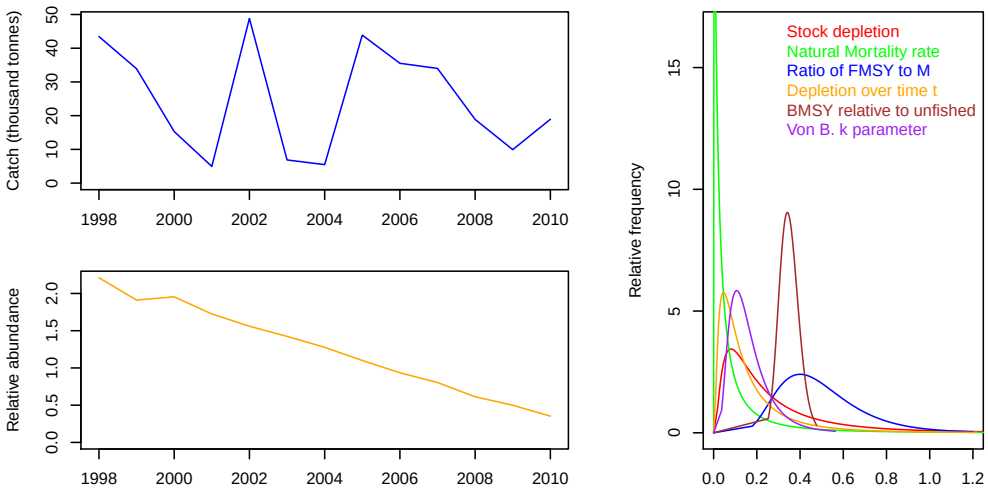
Alternatively you can create a blank DLM_data object and fill the slots directly in R. E.g:

```
Madeup<-new('DLM_data')                                  #  Create a blank DLM object


## [1] "Couldn't find specified csv file, blank DLM object created"


Madeup@Name<-'Test'                                      #  Name it
Madeup@Cat<-matrix(20:11*rlnorm(10,0,0.2),nrow=1)        #  Generate fake catch data
Madeup@Units<-"Million metric tonnes"                    #  State units of catch
Madeup@AvC<-mean(Madeup@Cat)                             #  Average catches for time t (DCAC)
Madeup@t<-ncol(Madeup@Cat)                               #  No. yrs for Av. catch (DCAC)
Madeup@Dt<-0.5                                           #  Depletion over time t (DCAC)
Madeup@Dep<-0.5                                          #  Depletion relative to unfished
Madeup@vbK<-0.2                                          #  VB maximum growth rate
Madeup@vbt0<-(-0.5)                                      #  VB theoretical age at zero length
Madeup@vbLinf<-200                                       #  VB maximum length
Madeup@Mort<-0.1                                         #  Natural mortality rate
Madeup@Abun<-200                                         #  Current abundance
Madeup@FMSY_M<-0.75                                      #  Ratio of FMSY/M
Madeup@L50<-100                                          #  Length at 50% maturity
Madeup@L95<-120                                          #  Length at 95% maturity
Madeup@BMSY_B0<-0.35                                     #  BMSY relative to unfished
```

8.3 Working with DLM_data objects

A generic summary function is available to visualize the data in a DLM_data object:

```
summary(Atlantic_mackerel)
```



You can see what MPs can and can't be applied given your data and also what data are needed to get MPs working:

```
Can(Atlantic_mackerel)
```

```
##  [1] "BK"          "CC1"         "CC4"         "DBSRA"       "DBSRA4010"
##  [6] "DBSRA_40"    "DCAC"        "DCAC4010"    "DCAC_40"     "DD"
## [11] "DD4010"      "DepF"        "DynF"        "EDCAC"       "Fadapt"
## [16] "Fdem"        "Fratio"      "Fratio4010" "GB_slope"    "Gcontrol"
## [21] "Islope1"     "Islope4"     "Itarget1"    "Itarget4"    "MCD"
## [26] "MCD4010"     "Rcontrol"    "Rcontrol2"   "SBT1"        "SPMSY"
## [31] "SPSRA"       "SPmod"       "SPslope"     "YPR"         "AvC"
## [36] "THC"         "MRnoreal"    "MRreal"      "curE"        "curE75"
## [41] "agelim5"     "area1_50"
```

```
Cant(Atlantic_mackerel)
```

```
##       [,1]           [,2]
##  [1,] "BK_CC"        "Insufficient data"
##  [2,] "BK_ML"        "Insufficient data"
##  [3,] "CompSRA"      "Insufficient data"
##  [4,] "CompSRA4010" "Insufficient data"
##  [5,] "DBSRA_ML"     "Insufficient data"
##  [6,] "DCAC_ML"      "Insufficient data"
##  [7,] "FMSYref"      "Insufficient data"
##  [8,] "FMSYref50"    "Insufficient data"
##  [9,] "FMSYref75"    "Insufficient data"
## [10,] "Fdem_CC"      "Insufficient data"
## [11,] "Fdem_ML"      "Insufficient data"
## [12,] "Fratio_CC"    "Insufficient data"
## [13,] "Fratio_ML"    "Insufficient data"
## [14,] "GB_CC"        "Produced all NA scores"
## [15,] "GB_target"    "Produced all NA scores"
## [16,] "LstepCC1"     "Insufficient data"
## [17,] "LstepCC4"     "Insufficient data"
```

```
## [18,] "Ltarget1"    "Insufficient data"
## [19,] "Ltarget4"    "Insufficient data"
## [20,] "SBT2"        "Produced all NA scores"
## [21,] "SPSRA_ML"    "Insufficient data"
## [22,] "YPR_CC"      "Insufficient data"
## [23,] "YPR_ML"      "Insufficient data"
## [24,] "matagelim"   "Insufficient data"
```

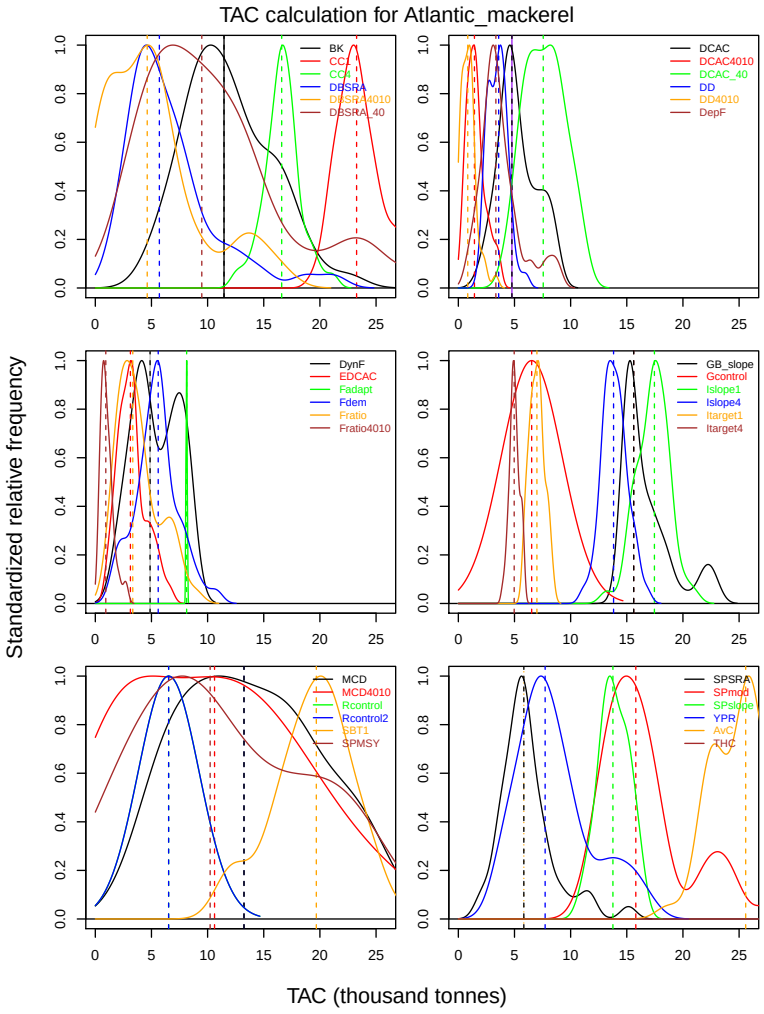
```
Needed(Atlantic_mackerel)
```

```
##  [1] "BK_CC: CAA"            "BK_ML: CAL"
##  [3] "CompSRA: CAA"          "CompSRA4010: CAA"
##  [5] "DBSRA_ML: CAL"         "DCAC_ML: CAL"
##  [7] "FMSYref: OM"           "FMSYref50: OM"
##  [9] "FMSYref75: OM"         "Fdem_CC: CAA"
## [11] "Fdem_ML: CAL"          "Fratio_CC: CAA"
## [13] "Fratio_ML: CAL"        "GB_CC: Cref"
## [15] "GB_target: Cref, Iref" "LstepCC1: CAL, MPrec"
## [17] "LstepCC4: CAL, MPrec"  "Ltarget1: CAL"
## [19] "Ltarget4: CAL"         "SBT2: Rec, Cref"
## [21] "SPSRA_ML: CAL"         "YPR_CC: CAA"
## [23] "YPR_ML: CAL"           "matagelim: NA, "
```

Spatial MPs and age-vulnerability MPs (class DLM_input) can be MSE tested but are a management recommendation in themselves. DLM_output MPs however can be calculated and plotted using getTAC() function:

```
Atlantic_mackerel<-TAC(Atlantic_mackerel,reps=48)
```

```
plot(Atlantic_mackerel)
```



9 Limitations

9.1 Idealised observation models for catch composition data

Currently, DLMtool simulates catch-composition data from the true simulated catch composition data via a multinomial distribution and some effective sample size. This observation model may be unrealistically well-behaved and favour those approaches that use these data. We (and by that I mean Adrian) is adding a growth-type-group model to improve the realism of simulated length composition data.

9.2 Harvest control rules must be integrated into data-limited MPs

In this version of DLMtool, harvest control rules (e.g. the 40-10 rule) must be written into a data-limited MP. There is currently no ability to do a factorial comparison of say 4 harvest controls rules against 3 MPs (the user must describe all 12 combinations). The reason for this is that it would require further subclasses. For example the 40-10 rule may be appropriate for the output of DBSRA but it would not be appropriate for some of the simple management procedures such as DynF that already incorporate throttling of TAC recommendations according to stock depletion.

9.3 Natural mortality rate at age

The current simulation assumes constant M with age. Age-specific M will be added soon.

9.4 Ontogenetic habitat shifts

Since the operating model simulated two areas, it is possible to prescribe a log-linear model that moves fish from one area to the other as they grow older. This could be used to simulate the ontogenetic shift of groupers from near shore waters to offshore reefs. Currently this feature is in development.

9.5 Implementation error

In this edition of DLMtool there is no implementation error. The only imperfection between a management recommendation and the simulated TAC comes in the form of the MaxF argument that limits the maximum fishing mortality rate on any given age-class in the operating model. The default is 0.8 which is high for all but the shortest living fish species.

10 References

Carruthers, T.R., Punt, A.E., Walters, C.J., MacCall, A., McAllister, M.K., Dick, E.J., Cope, J. 2014. Evaluating methods for setting catch-limits in data-limited fisheries. Fisheries Research. 153, 48-68.

Carruthers, T.R., Kell, L., Butterworth, D., Maunder, M., Geromont, H., Walters, C., McAllister, M., Hillary, R., Kitakado, T., Davies, C. 2015. Performance review of simple management procedures. ICES journal, in press.

Costello, C., Ovando, D., Hilborn, R., Gains, S.D., Deschenes, O., Lester, S.E., 2012. Status and solutions for the world???s unassessed fisheries. Science. 338, 517-520. Deriso, R. B., 1980. Harvesting Strategies and Parameter Estimation for an Age-Structured Model. Can. J. Fish. Aquat. Sci. 37, 268-282.

Dick, E.J., MacCall, A.D., 2011. Depletion-Based Stock Reduction Analysis: A catch-based method for determining sustainable yields for data-poor fish stocks. Fish. Res. 110, 331-341.

Geromont, H.F. and Butterworth, D.S. 2014. Complex assessment or simple management procedures for efficient fisheries management: a comparative study. ICES J. Mar. Sci. doi:10.1093/icesjms/fsu017

MacCall, A.D., 2009. Depletion-corrected average catch: a simple formula for estimating sustainable yields in data-poor situations. ICES J. Mar. Sci. 66, 2267-2271.

Newman, D., Berkson, J., Suatoni, L. 2014. Current methods for setting catch limits for data-limited fish stocks in the United States. Fish. Res. 164, 86-93.

Restrepo, V.R., Thompson, G.G., Mace, P.M., Gabriel, W.L., Low, L.L., MacCall, A.D., Methot, R.D., Powers, J.E., Taylor, B.L., Wade, P.R., Witzig, J.F.,1998. Technical Guidance On the Use of Precautionary Approaches to Implementing National Standard 1 of the Magnuson-Stevens Fishery Conservation and Management Act. NOAA Technical Memorandum NMFS-F/SPO-31. 54 pp.