12. Appendices

Table of Contents

Introduction

The vignettes from the ***cede*** package plus the **catch-MSY** method, the ***spm*** method, and the ***aspm*** method are listed in sequence. The only changes here relative to the originals in the R packages is that figure numbers have been made sequential. In addition, the order of the text, figures, and code, may have been altered slightly to avoid the occurrence of large gaps in the text.

The table of contents has been combined at the front, and the references combined at the rear. Tables to do not generally have legends, and some figures form part of the flow of the discussion and do not always have legends.

The aims of the vignettes are to introduce the methods and illustrate how they can be used.

cede - Catch Effort and Data Exploration

When working with relatively minor commercial species the data available is typically less comprehensive than for species that might be considered to be the economic drivers of a fishery. Nevertheless, data exploration, perhaps through plotting up different variables and how they might change through the years, can often be informative about changes in any fishery for a particular species. The **cede** R package (Catch Effort and Data Exploration) includes an array of functions that should assist with such data exploration. If a species' fishery includes CPUE data then plots of the distribution of catches, effort, and CPUE (perhaps as Log(CPUE)) can be helpful in the interpretation of such CPUE, especially if there is sufficient data to allow for CPUE standardization. **cede** now includes various functions that can also assist with CPUE standardization. All these functions are described below with examples of their use.

There should be no expectation that the functions to be used in the standardization of CPUE constitute anything like a complete treatment. This vignette only provide a very brief introduction or pointer to get people started. There are many aspects not considered (e.g. how or whether to treat zeros). This vignette remains a draft and if you find errors, omissions, or obscurities do please let me know (see DESCRIPTION for email address). In addition if you wish to reference this package when writing your SAFS assessment you can obtain one by typing `citation("cede")` into the console, which will give you the latest version.

Data Exploration

The main data set included with **cede** is called *sps* and contains typical fisheries data from a scalefish fishery. It is there mainly to assist with learning the operation of the different functions. Generally it would be better to use your own data but if you consider the *sps* data set you will gain an understanding of a typical format.

```
data(sps)
kable(sps[1:6,],digits=c(0,0,0,0,3,3,0,0,2,0))
```

Year	Month	Vessel	catch_kg	Long	Lat	Depth	DayNight	Effort	Zone
2004	4	1	220	145.117	-43.067	125	N	4.00	1
2004	4	1	280	145.250	-43.233	130	M	3.66	1
2004	4	1	180	145.150	-43.083	115	D	3.50	1
2004	4	1	70	145.233	-43.217	120	N	4.75	1
2004	4	1	200	145.100	-43.033	120	M	4.75	1
2004	4	1	100	145.767	-43.683	130	M	2.01	1

```
properties(sps)

##          Index isNA Unique     Class        Min     Max   Example
## Year         1    0     12   numeric 2003.00000 2014.00      2004
## Month        2    0     12   numeric    1.00000   12.00         4
## Vessel       3    0     23   numeric    1.00000   27.00         1
## catch_kg     4    0    442   numeric    1.00000 4500.00       220
## Long         5    0    447   numeric  144.11667  146.30  145.1167
## Lat          6    0    512   numeric  -45.83333  -40.75 -43.06667
## Depth        7    0    191   numeric    2.00000  366.00       125
## DayNight     8    0      3 character    0.00000    0.00         N
## Effort       9    0    377   numeric    0.16000    9.66         4
## Zone        10    0      3   numeric    1.00000    3.00         1
```

The *properties* function categorizes the contents of a data.frame, counting the number of NAs in each variable, if any, listing their class, their minimum and maximum (if applicable) and finally printing an example of the contents. I find this function quite useful when beginning to use a different data.frame. Generally I refer to variables within a data.frame by their names so it is important to know if they are capitalized or not as well as knowing exactly which variables are present.

Once we have our data available for analysis it is often a good idea to find ways to summarize how they vary relative to one another. With fisheries data it is common to want to know how different factors influence the total catch and whether these vary by year. Typically one might use the R function *tapply* to conduct such examinations. To simplify this use one can use the *tapsum* function from within **cede**.

The seasonality of catches can be indicative of the typical behaviour of the fishery within a year.

```
kable(tapsum(sps,"catch_kg","Year","Month"),digits=c(1,1,1,1,1,1,1,1,1
,1,1,1))
```

	1	2	3	4	5	6	7	8	9	10	11	12
2003	33.6	26.0	37.3	30.4	14.7	3.7	4.8	11.6	14.5	5.1	6.4	33.8
2004	73.7	66.2	52.7	100.8	55.9	18.3	12.7	22.8	8.4	9.4	30.1	21.6
2005	114.9	83.9	35.0	37.4	7.3	15.1	11.8	6.1	4.1	13.3	13.9	36.0
2006	79.8	53.1	45.8	27.4	0.3	1.8	2.8	3.1	0.4	5.1	9.2	55.7
2007	31.8	60.1	27.3	1.5	13.6	4.6	2.5	0.8	0.3	0.2	7.0	20.6
2008	76.3	21.6	33.0	5.5	2.1	0.7	1.3	0.5	0.2	3.2	6.4	14.1
2009	16.7	25.4	9.5	2.5	2.4	0.7	0.6	2.0	0.7	6.7	18.2	11.2
2010	40.9	22.5	11.4	2.0	0.3	0.5	1.8	2.3	1.4	1.6	0.7	4.4
2011	25.0	38.6	10.6	6.3	2.7	3.2	1.5	2.7	2.1	2.2	5.1	23.5
2012	35.3	49.4	24.9	6.4	2.9	2.6	5.1	1.6	1.6	3.4	4.4	13.1
2013	47.3	48.8	41.0	11.0	17.1	0.3	2.3	0.5	1.3	6.6	6.3	6.4
2014	11.0	10.3	21.5	12.1	6.4	11.0	8.1	15.5	3.9	3.8	26.6	49.4

Here we have examined the catch by zone where the zones are in sequence along the coast (or they would be if this was a real fisheries data).

```
tapsum(sps,"catch_kg","Year","Zone")

##             1         2      3
## 2003  94.6190  98.06400 29.197
## 2004 215.2230 210.47900 46.804
## 2005 112.7670 216.02300 50.079
## 2006  82.4370 120.29100 81.663
## 2007  42.7560  91.46240 36.161
## 2008  51.9840  93.81300 19.020
## 2009  33.9920  33.62310 28.931
## 2010  11.8070  18.71400 59.165
## 2011  37.1840  79.41725  6.892
## 2012  55.2330  65.35600 30.263
## 2013  50.3015  83.81800 54.848
## 2014  46.6240  81.44250 51.455
```

We are not limited to summarizing catch but, for example could also look at the distribution of effort as total number of hours (note the change to the default value of div so that the total number of hours is not divided by 1000). By pointing the function call to a new object one can then plot the results.

```
effbyyr <- tapsum(sps,"Effort","Year","Zone",div=1.0)
effbyyr

##            1       2      3
## 2003 2473.36 1998.01 724.13
## 2004 3558.32 2541.13 709.58
## 2005 2095.92 2750.78 639.01
## 2006 2001.37 2055.52 941.46
## 2007 1192.94 1279.45 481.96
## 2008 1426.79 1072.82 495.61
## 2009  877.81  739.13 488.86
## 2010  471.06  493.39 691.16
## 2011  855.54 1185.06 293.93
## 2012 1278.07  981.93 508.41
## 2013 1323.23  960.89 816.47
## 2014 1036.63 1222.02 681.42

# plotprep(width=7,height=4.5)
ymax <- max(effbyyr,na.rm=TRUE)
label <- colnames(effbyyr)
yrs <- as.numeric(rownames(effbyyr))
par(mfrow=c(1,1),mai=c(0.45,0.45,0.05,0.05))
par(cex=0.85, mgp=c(1.35,0.35,0), font.axis=7,font=7,font.lab=7)
plot(yrs,effbyyr[,label[1]],type="l",lwd=2,col=1,ylim=c(0,ymax),
     ylab="Total Effort (Hours) by Zone per Year",xlab="",
     panel.first=grid())
lines(yrs,effbyyr[,label[2]],lwd=2,col=2)
lines(yrs,effbyyr[,label[3]],lwd=2,col=3)
legend("topright",label,col=c(1,2,3),lwd=3,bty="n",cex=1.25)
```

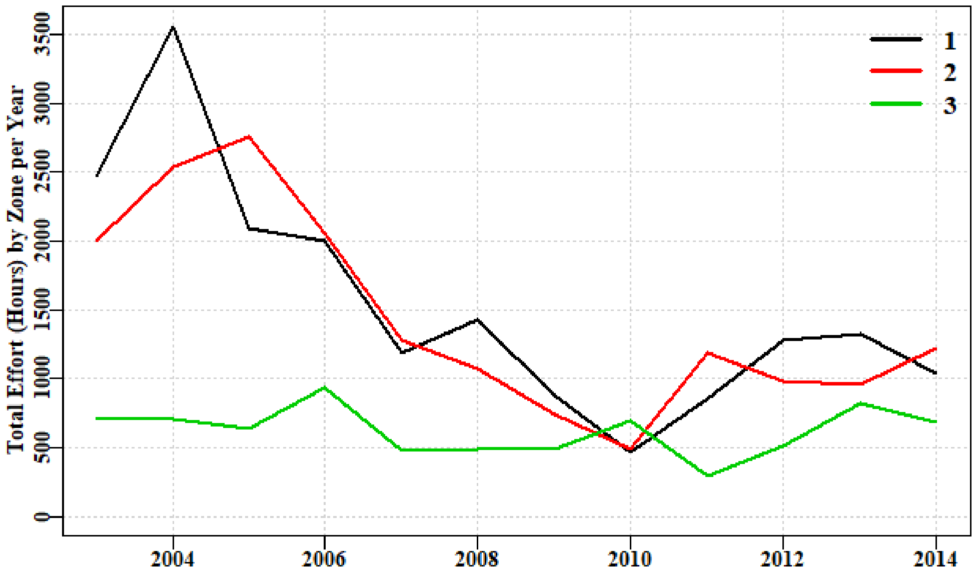


Figure 1. A plot of total effort by zone, showing that a visual illustration can often more easily highlight changes in a fishery's dynamics.

DayNight is another factor that can have large consequences for catches and catch rates. Check the description of the *sps* data set using `?sps

```
tapsum(sps,"catch_kg","Year","DayNight")
```

```
##              D        M       N
## 2003  80.54300  81.3930 59.9440
## 2004 226.67300 153.7910 92.0420
## 2005 157.21800 133.5640 88.0870
## 2006 127.24900 104.6120 52.5300
## 2007  72.13700  61.5024 36.7400
## 2008  75.67900  56.9030 32.2350
## 2009  35.10710  34.7680 26.6710
## 2010  39.00500  25.8060 24.8750
## 2011  46.14625  44.6535 32.6935
## 2012  52.92000  59.4950 38.4370
## 2013  72.16750  66.8170 49.9830
## 2014  52.40750  64.0420 63.0720
```

One of the most influential factors within each fishery is the vessel doing the catching. Often this is also a reflection of the skipper of the vessel as well as the relative performance of the boat itself. Nevertheless, it is often the case the vessel name is the only information available about the vessel's relative fishing power. It is possible to pay special attention to catch-per-vessel, although the following analysis is more general than that and can be applied to, for example, catch-by-month relative to Depth Category.

```
cbv <- tapsum(sps,"catch_kg","Vessel","Year") # often more vessels
than years
total <- rowSums(cbv,na.rm=TRUE)
cbv1 <- cbv[order(total,decreasing = TRUE),]   # sort by total catch
kable(cbv1[1:20,],digits=c(1,1,1,1,1,1,1,1,1,1,1,1))
```

	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014
2	52.5	154.8	97.3	86.0	63.0	85.4	50.5	34.6	59.5	57.5	70.5	38.4
1	1.7	107.3	73.1	32.6	23.3	25.1	18.3	17.1	30.7	42.9	31.4	80.2
7	45.3	37.8	61.5	49.7	14.1	20.7	11.5	6.2	6.3	16.2	23.0	3.2
3	31.6	15.7	39.9	32.8	21.7	16.3	12.3	25.8	20.0	21.4	41.7	15.0
13	8.6	8.2	10.1	4.3	26.7	9.5	1.1	4.0	0.3	9.3	8.9	17.1
5	0.1	27.7	40.8	29.5								9.2
8	38.6	32.1	17.2									
6	19.4	41.7	3.7	13.8								
9	1.2	9.3	1.4	6.1	15.0	6.4	2.1	1.9	5.8	3.2	12.5	13.2
10	9.8	18.0	22.8	22.0								
14	6.2	0.3	0.6	4.6	6.1	1.2	0.5				0.0	0.1
12	3.0	4.9	1.2	3.0			0.1					
4	0.3	11.0										
11	0.0	3.7	7.5									
25								0.1	0.1		0.7	2.8
17	2.5											
19	0.0						0.1	0.0	0.8	0.4	0.3	0.2
23			1.2									
24			0.2		0.6							
20	0.8											

Obviously some vessels will be much more influential than others simply because they catch a great deal more than others and hence introduce many more records into the database.

```
# plotprep(width=8,height=6) # not needed in the vignette
to <- turnover(cbv1)
yearBubble(cbv1,ylabel="sqrt(catch-per-vessel)",
           diam=0.125,txt=c(2,3,4,5),hline=TRUE)
```

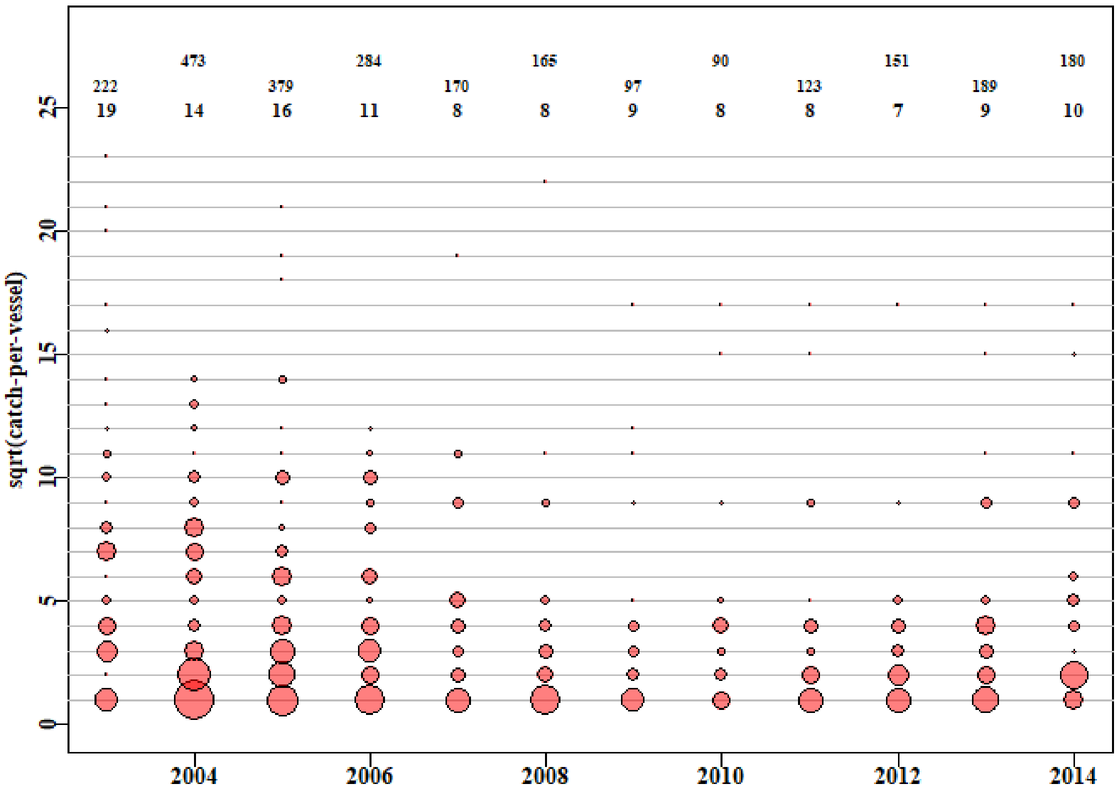


Figure 2. This hypothetical fishery is clearly dominated by four or five vessels with numerous minor players. Additionally, before 2007 there were a few more productive fishers present (this reflects the structural adjustment in the Commonwealth from which this simulated data derives). The optional horizontal lines merely delineate the individual vessels. The top two rows of numbers is the total catch per year and the bottom row of numbers is the number of vessels reporting in each year.

It is likely that if the data from the bottom nine vessels were omitted there would be no effect on any results as their catches are so minor in a relative sense. It is clear those vessels are merely casual occurrences within the fishery.

While the main vessels were reasonably consistent in terms of reporting from this fishery other vessels came and went. To summarize such activity one can use the *turnover* function which sumarizes the year-to-year changes in which vessels report being active.

```
print(to)

##      Continue Leave Start Total
## 2003       19     0     0    19
## 2004       14     5     0    14
## 2005       13     1     3    16
## 2006       11     5     0    11
## 2007         7     4     1     8
## 2008         7     1     1     8
## 2009         7     1     2     9
## 2010         7     2     1     8
## 2011         8     0     0     8
## 2012         7     1     0     7
## 2013         7     0     2     9
## 2014         9     0     1    10
```

The Continue column lists how many continued from the preceding year, the Leave column designates how many left relative to the previous year, while the Start column is literally how many started reporting in that year. The Total is the total reporting in each year. No attempt is made to follow individual vessels.

The Addition of CPUE data

You will have noticed that the data came with catch and effort but not CPUE, so we need to calculate that. In the following I test for the presence of zeros in the catch and effort to avoid generating errors of division (divide by zero errors will stop the analysis) and when taking logs. In fact, as the *properties* call showed there were no *NA* values, but is remains worth checking. While we are adding CPUE we can also group the depth data into depth classes to provide that option when standardizing the CPUE data.

```
sps$CE <- NA     # make space in the data.frame
sps$LnCE <- NA
pick <- which((sps$catch_kg > 0) & (sps$Effort > 0))
sps$CE[pick] <- sps$catch_kg[pick]/sps$Effort[pick]
sps$LnCE[pick] <- log(sps$CE)   # natural log-transformation
# categorize Depth
range(sps$Depth,na.tm=TRUE)     # to aid selection of depth class
width

## [1]   1 366

sps$DepCat <- NA
sps$DepCat <- trunc(sps$Depth/25) * 25
table(sps$DepCat)

##
##    0   25   50   75  100  125  150  175  200  225  250  275  300
325  350
##    6   19  224 1569 4583 3593 1393   74   66   21   15   21    7
5    7
```

It is clear from the summary of records by depth that most of the fishing occurs in waters of 150 meters or less.

Tables of numbers are very informative but sometimes it is much easier to gain a visual impression of patterns in one's data by plotting them. Typically, with fisheries data, one might plot each variable, such as catch, effort, log(CPUE), depth, etc, by year to see whether changes have occurred through time. Such changes might adversely affect any analysis applied so it is always a good idea to examine (explore) one's data before using it. **cede** provides a function *histyear* that an plot a histogram of a selected variable by year.

```
outH <- histyear(sps,Lbound=-1.75,Rbound=8.5,inc=0.25,pickvar="LnCE",
                 years="Year",varlabel="log(CPUE)",plots=c(4,3))
```

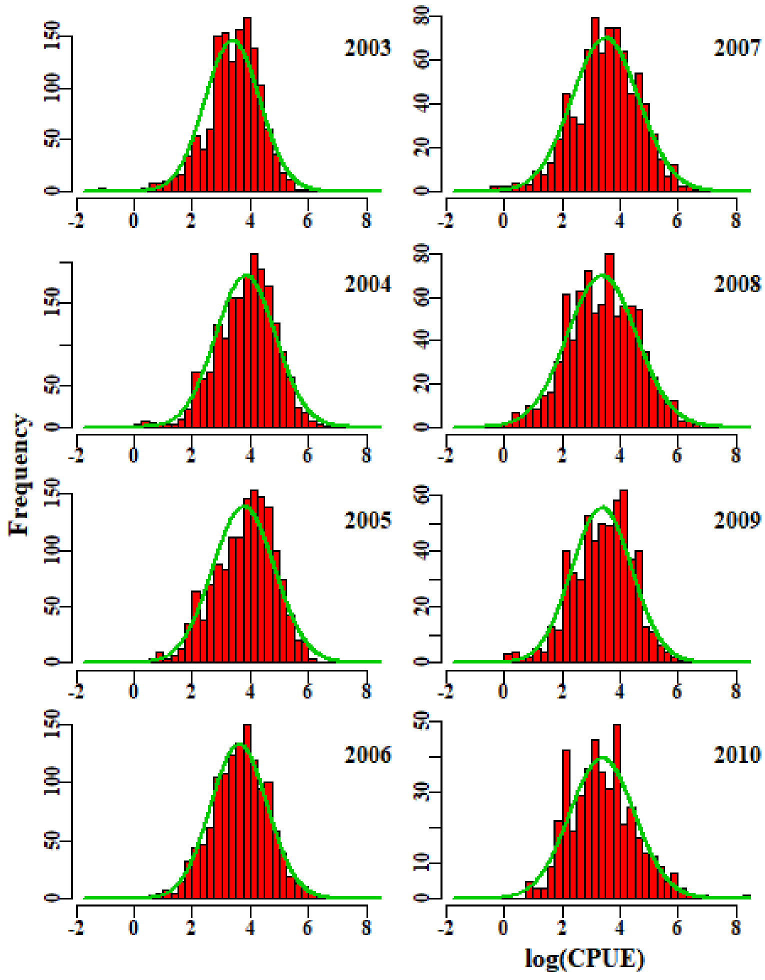


Figure 3. The distribution of the log(CPUE) each year for which data is available. The green lines are fitted normal distributions there for reference (log-transformation should normalize log-normal data).

```
outH <- histyear(sps,Lbound=0,Rbound=375,inc=12.5,pickvar="Depth",
                 years="Year",varlabel="Depth
(m)",plots=c(4,3),vline=120)
```

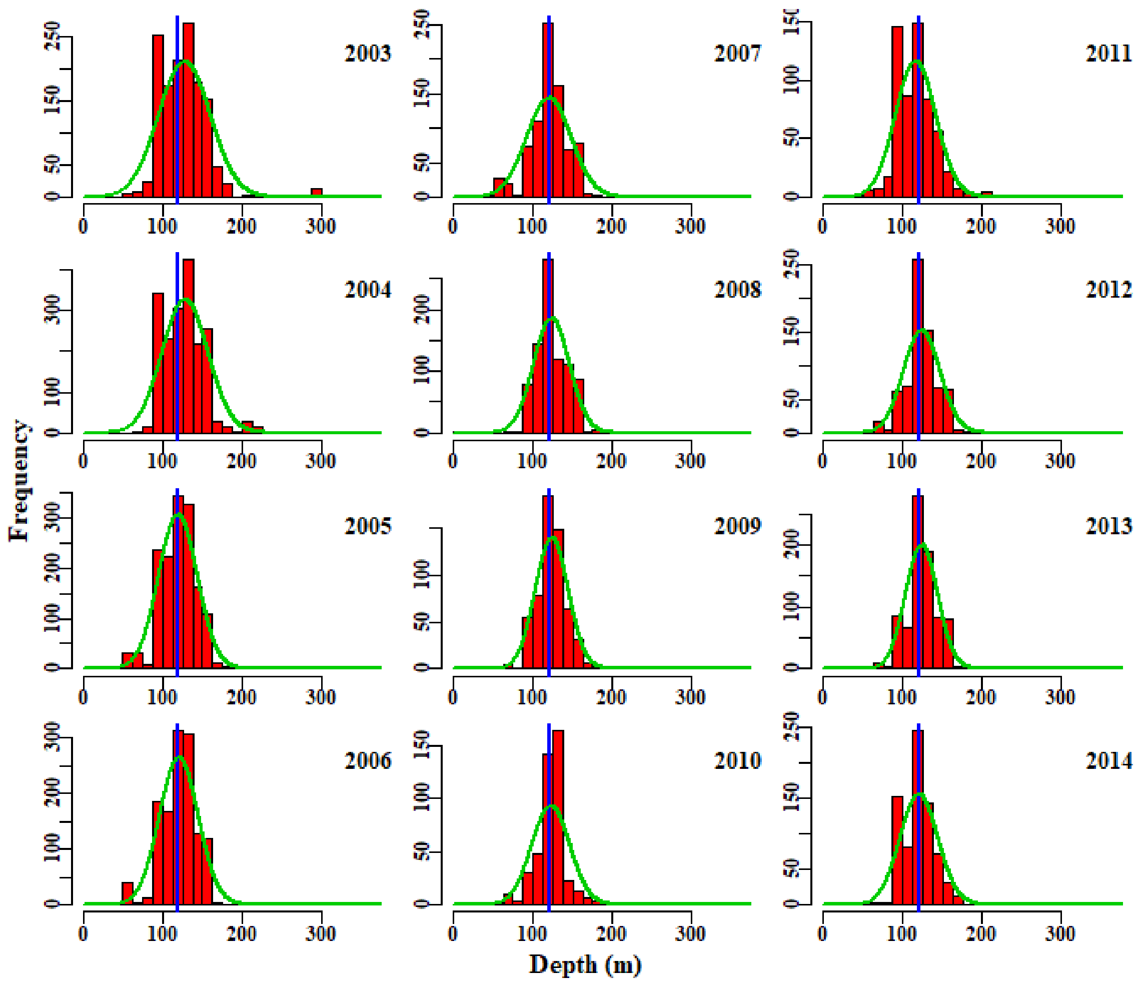


Figure 4. The distribution of reported mean depth of fishing each year. The green lines are fitted normal distributions there for reference, the blue lines are merely reference lines to ease comparisons between years.

```
outH <- histyear(sps,Lbound=0,Rbound=10,inc=0.25,pickvar="Effort",
                 years="Year",varlabel="Effort
(Hrs)",plots=c(4,3),vline=NA)
```

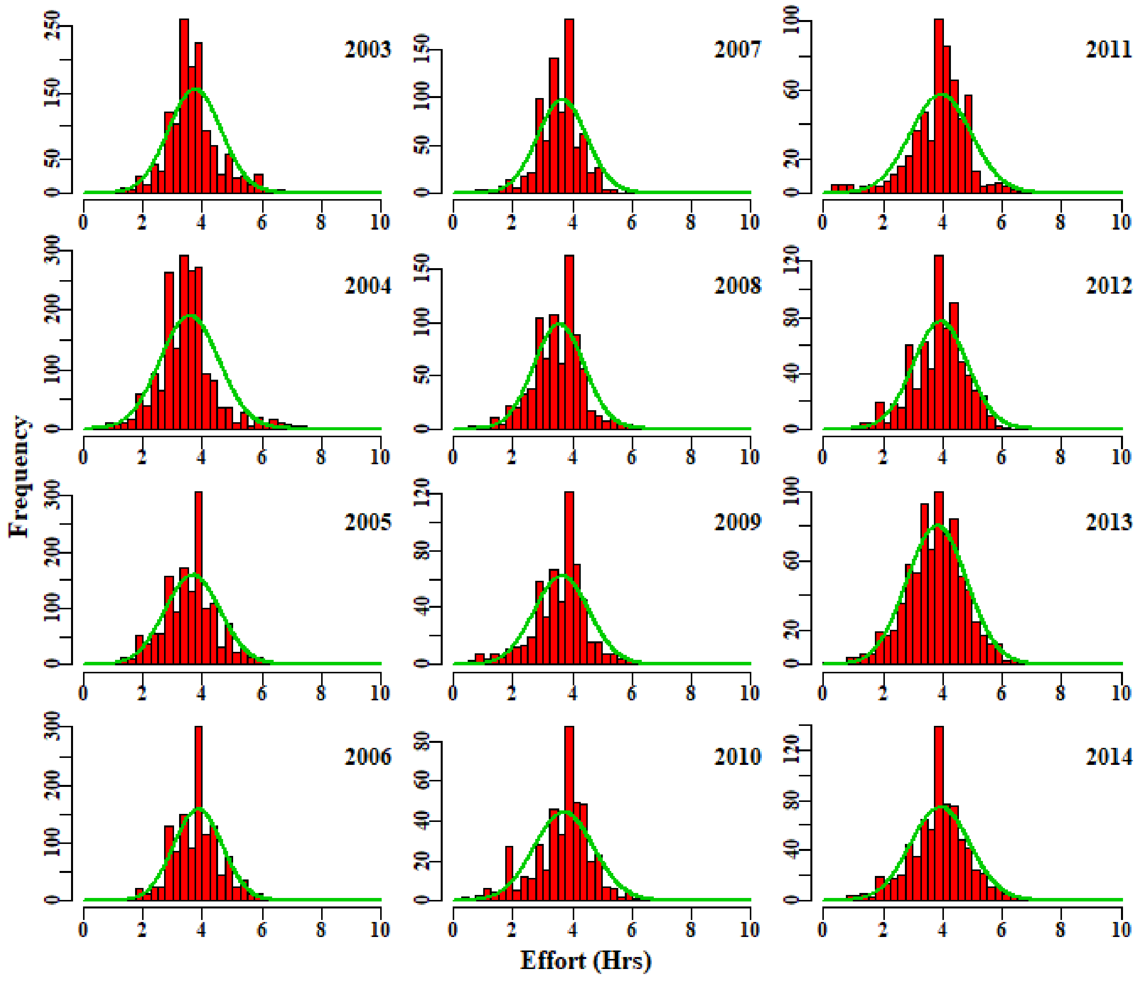


Figure 5. The distribution of reported Effort each year. The green lines are fitted normal distributions there for reference. Note the spikes of reporting four hours.

Spikes can be seen in each of the graphs and the question needs to arise whether this is due to rounding by the fishers or is a real phenomenon. In fact, unless dealing with counts of fish caught (quite possible in some fisheries) then rounding invariably occurs when estimating catches but also in effort.

```
par(mfrow=c(1,1),mai=c(0.45,0.45,0.05,0.05))
par(cex=0.85, mgp=c(1.35,0.35,0), font.axis=7,font=7,font.lab=7)
plot(sps$Effort,sps$catch_kg,type="p",pch=16,col=rgb(1,0,0,1/5),
     ylim=c(0,500),xlab="Effort (Hrs)",ylab="Catch (Kg)")
abline(h=0.0,col="grey")
```

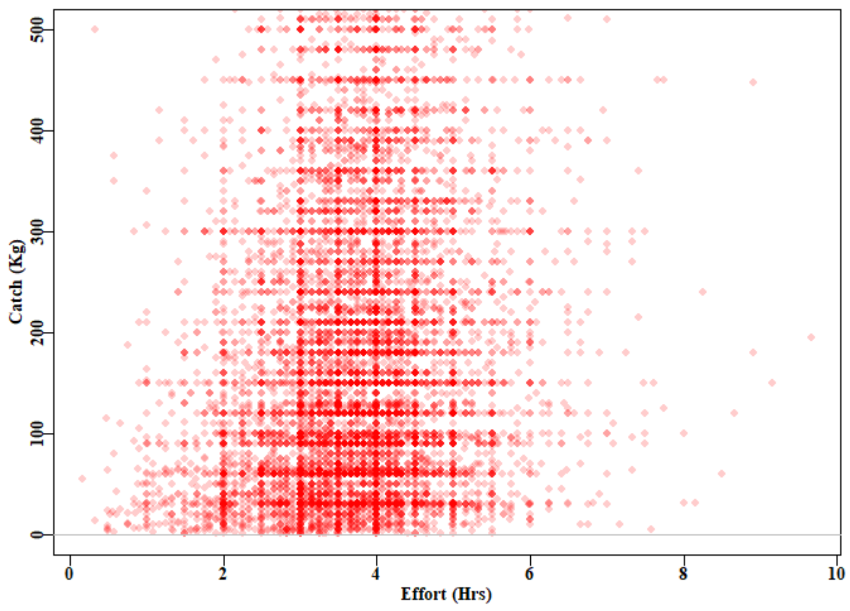


Figure 6. A plot of catch against effort for each record in the *sps* data.frame. The catch axis has been truncated at 500 kg so as to allow the rounding of catches to be less compressed and more visually obvious. It should be clear there is rounding at every half hour between 2 - 6 hours. In addition, there is rounding at about 30 kg steps from 30 - 300 kg, with other categories above that. The 30-33kg rounding reflects a belief that a standard fish bin contains about 30-33Kg of fish.

The uneven grid like nature of the catch and effort data is reflected in the CPUE data, which might make one skeptical about the notion of a predictive model attempting to predict such values. While the residuals that are the basis of th statistical model fitting might be smoother in their distribution they do derive from a comparison of smooth predicted values with the grouped observed values, so any results are likely to be uncertain and to under-estimate any inherent variation.

Despite such problems it is possible to derive useful information from fisheries data. It is generally recognized that fisheries data in general is noisy and potentially contains many errors, especially when considering the less important species that fall into the data-poor category. Neverntheless, the challenge remains of attempting to obtain useful and useable information from analysing such data.

Plotting Sketch Maps of Lat-Long data

Since the advent of GPS and GPS plotters very many fishers use there equipment and fisheries departments have started to ask for precise location data accordingly. If such latitude and longitude data are available it is often informative to plot such data as a literal map to illustrate the focus and range of a fishery. **cede** also provides the capacity to generate such sketch maps instead of using a full GIS. The idea here is not to conduct detailed spatial analyses, for which a GIS is better suited. Instead the idea is simply to gain a rapid impression of the operation of a fishery. Of course, care needs to be taken with such plots as they very obviously contain confidential information (such as exactly where fishers have been operating). This is especially important when there are very few fishers involved in a fishery. So while such images may not be able to be displayed in meetings they remain useful for data exploration purposes.

```
leftlong <- 143.0;  rightlong <- 150.0
uplat <- -40.0;  downlat <- -44.6
```

```
plotaus(leftlong,rightlong,uplat,downlat,gridon=1.0)
addpoints(sps,intitle="Location of Catches")

## 11603 1 4500

plotLand(incol="blue")
```

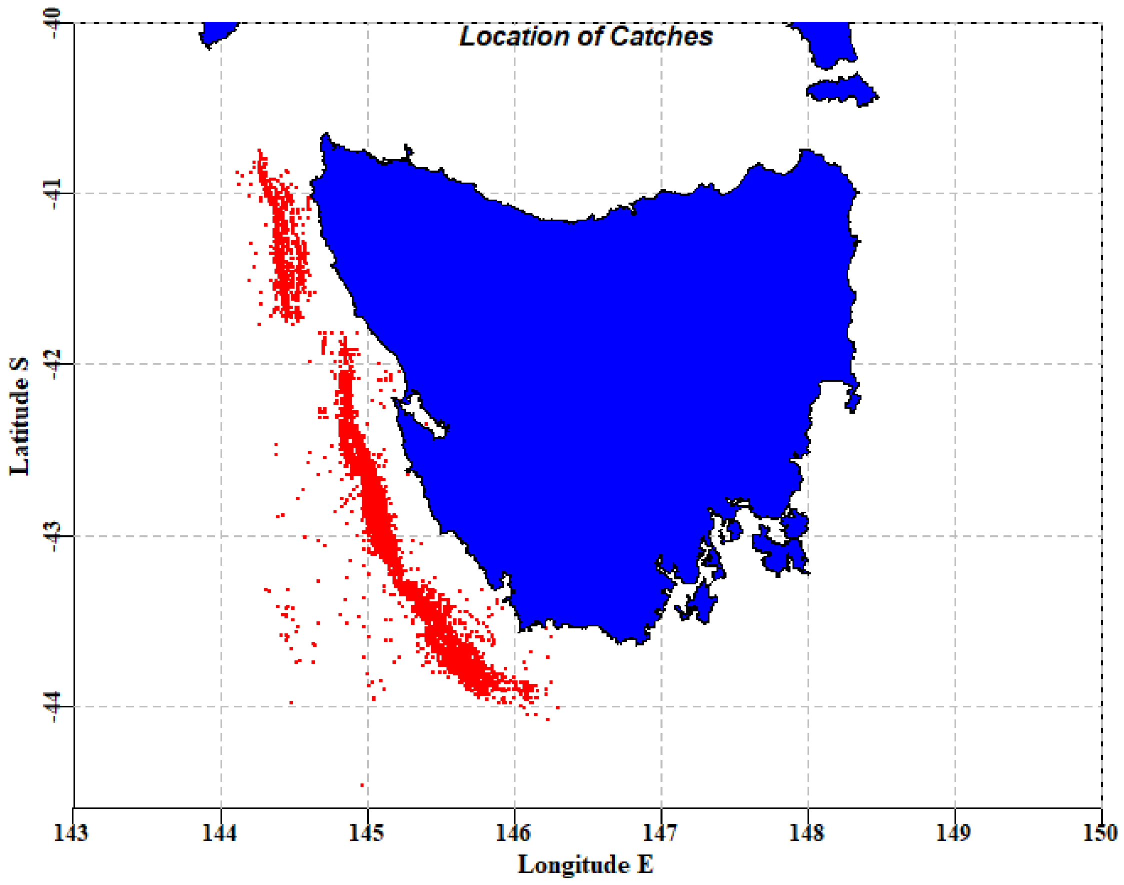


Figure 7. A sketch map of the the Lat Long data within the *sps* data set. There are clearly a number of points reported to be out oer the abyssal plain, but the majority of points define the range of the fishery.

Rather than show individual points it is also possible, by using the function *plotpolys*, to aggragate catches into different geographical sub-divisions (e.g. 0.25 or 0.5 degree squares, definable with the *gridon* parameter). If these are coloured relative to the density of total catches the locations where most of the yield of a fishery derives from becomes apparent. The output, from the function includes the plotting but also the sub-divisions used and the counts of each of those sub-divisions. The final plotting of the land is merely to provide a tidy plot.

```
leftlong <- 143.0;  rightlong <- 150.0
uplat <- -40.0;  downlat <- -44.6
plotaus(leftlong,rightlong,uplat,downlat,gridon=1.0)
plotpolys(sps,leftlong,rightlong,uplat,downlat,gridon=0.2,leg="left",
          intitle="0.2 degree squares",mincount=2)

## subdiv     87.057 8.7057 0.87057 0.087057
## counthot   0 0 0 0
## 494.8624     500 250 100 50 10 5 1 0.001
## countpoly  0 2 6 6 11 4 16 31
```

```
plotLand(incol="pink")
```

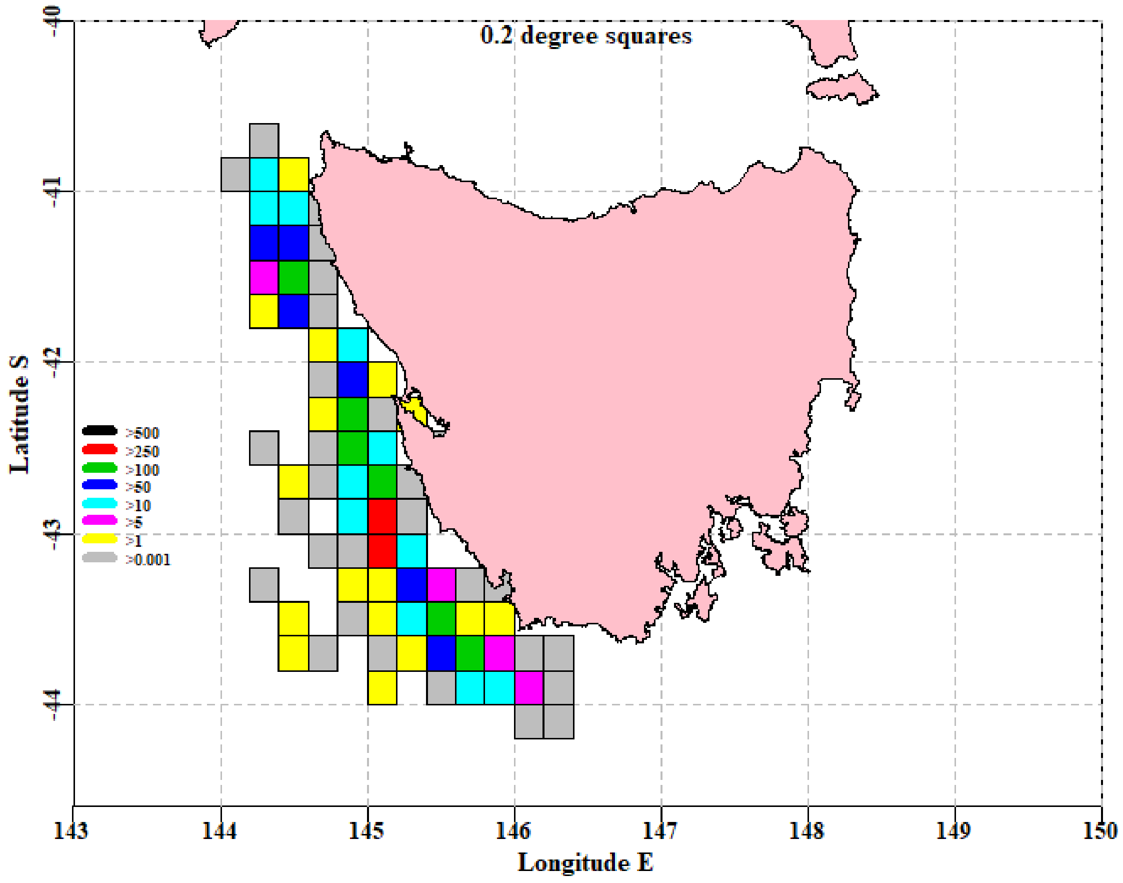


Figure 8. A sketch map of the the Lat Long data within the *sps* data set with catches aggregated into 0.2 degree squares. By requiring at least 2 records in each square before inclusion some of the deeper water extraneous records have been eliminated (although not all). The red, green, and roayl blue squares denote the areas generating the greatest yields.

Such sketch maps can be helpful, especially when plotting single year's of data to illustrate how the extent of a fishery varies through time. There are obvious limitations. There is no formal map projection, one merely alters the width and height of the plot until the visual representation of the land looks acceptable. In addition there are islands missing so as to limit the size of the underlying coastal definition data set (to see this try entering *head(cede:::aus,30)* into the console).

CPUE Standardization

Introduction

If one were to search online for CPUE standardization it would quickly become apparent that this is a very large subject with many alternative approaches and strategies. Here I will introduce two approaches that use General and Generalized Linear Models (LMs and GLMs) and that use Generalized Additive Models (GAMS). This will only be a brief introduction to the subject but the hope is that such an introduction would enable users to explore further and develop approaches best suited to their own fisheries.

Commercial catch and effort (CPUE) data are used in very many fishery stock assessments in Australia as an index of relative abundance. Using CPUE in this way assumes there is a direct relationship between catch rates and exploitable biomass. However, many other factors can influence catch rates, including vessel, gear(fishing method), depth, season, area, and time of fishing (e.g. day or night). The use of CPUE as an index of relative abundance requires the removal of the effects of variation due to changes in factors other than stock biomass, on the assumption that what remains will provide a better estimate of the underlying biomass dynamics. This process of adjusting the time series for the effects of other factors is known as standardization and the accepted way of doing this is to use some statistical modelling procedure that focuses attention onto the annual average catch rates adjusted for the variation in the averages brought about by all the other factors identified. Idiosyncrasies between species and methods across Australia means that each fishery/stock for which standardized catch rates are required entails its own set of conditions and selection of data.

The Limits of Standardization

The use of commercial CPUE as an index of the relative abundance of exploitable biomass can be misleading when there are factors that significantly influence CPUE but cannot be accounted for in a statistical standardization analysis. Over the last few decades the management of many Australian fisheries have undergone significant changes. For example, in the Commonwealth fisheries there was the introduction of the quota management system into the SESSF in 1992, and the introduction of the Harvest Strategy Policy (HSP) and associated structural adjustment in 2005 - 2007. The combination of limited quotas and the HSP is now controlling catches in such a way that many fishers have been altering their fishing behaviour to take into account the availability of quota and their own access to quota needed to land the species taken in the mixed species SESSF.

Methods

Initial Data Selection

Fisheries data is often noisy and can contain obvious errors and mistakes (e.g. an inshore species repotedly being caught in 6000 m of water). The data exploration mentioned earlier should allow one to defensibly select data for further analysis. Often such data selection is aimed at identifying records that represent typical activities in each fishery concerned. In particular some selection criteria are aimed at focussing on records where the species is being targeted. For example, most species have a depth range within which they are typically caught. Ideally, an agreed depth range should be used so that it becomes standard to select data records between some minimum and maximum dapth range. A second example relates to one vessel in the SESSF catching a particular species by a particular gear having catch rates 10 - 20 times those of other vessels fishing in the same places at the same time. Further exploration indicated that the vessel concerned had misunderstood how to fill in the log book so their data was removed from subsequent analysis. Whatever decisions are made about the selection of data, each choice should be defensible and it should be possible to present the evidence for the selection made (e.g. illustrate extreme values, typical depth ranges, unusual vessels).

Once a defensible set of data records have been selected there are other modifications needed. At its most besic a linear model is very similar to a regression analysis. If you imagine conducting a regression of Log(CPUE) against Year so as to evaluate how those catch rates have changed through time then all that would come out would be a single line having two parameters, an intercept and gradient. There are only two parameters because it would treat the factor ‘Year’ as a continuous variable. What we actually want is a separate index for each year, we need to treat the ‘Year’ factor as a categorical factor rather than as a continuous variable. Below we will illustrate the use of using all categorical factors and then a different illustration showing how to include a continuous variable such as depth, into a standardization.

Standardization

The use of *properties* indicates that in the *sps* data set contains six clear factors: Year, Month, Vessel, Depth or DepCat, DayNight, and Zone. The Zone factor is a subdivision of mainly the Latitude factor although longitude is also in there to a lesser extent.

First we need to convert some of the factors into categorical factors. For the *sps* data set there are six factor. It is good practice not to over-write your original data.frame so here the *sps* name is slightly modified to *sps1*.

```
kable(properties(sps),digits=c(0,0,0,0,6,6,6))
```

	Index	isNA	Unique	Class	Min	Max	Example
Year	1	0	12	numeric	2003.000000	2014.000000	2004
Month	2	0	12	numeric	1.000000	12.000000	4
Vessel	3	0	23	numeric	1.000000	27.000000	1
catch_kg	4	0	442	numeric	1.000000	4500.000000	220
Long	5	0	447	numeric	144.116670	146.300000	145.1167
Lat	6	0	512	numeric	-45.833330	-40.750000	-43.06667
Depth	7	0	191	numeric	2.000000	366.000000	125
DayNight	8	0	3	character	0.000000	0.000000	N
Effort	9	0	377	numeric	0.160000	9.660000	4
Zone	10	0	3	numeric	1.000000	3.000000	1
CE	11	0	3624	numeric	0.222222	4140.000000	55
LnCE	12	0	3596	numeric	-1.504077	8.328451	4.007333
DepCat	13	0	15	numeric	0.000000	350.000000	125

```
labelM <- c("Year","Zone","Vessel","Month","DayNight","DepCat")
sps1 <- makecategorical(labelM,sps)
kable(properties(sps1),digits=c(0,0,0,0,6,6,6))
```

	Index	isNA	Unique	Class	Min	Max	Example
Year	1	0	12	factor	0.000000	0.000000	2004
Month	2	0	12	factor	0.000000	0.000000	4
Vessel	3	0	23	factor	0.000000	0.000000	1
catch_kg	4	0	442	numeric	1.000000	4500.000000	220
Long	5	0	447	numeric	144.116670	146.300000	145.1167
Lat	6	0	512	numeric	-45.833330	-40.750000	-43.06667
Depth	7	0	191	numeric	2.000000	366.000000	125
DayNight	8	0	3	factor	0.000000	0.000000	N
Effort	9	0	377	numeric	0.160000	9.660000	4
Zone	10	0	3	factor	0.000000	0.000000	1
CE	11	0	3624	numeric	0.222222	4140.000000	55
LnCE	12	0	3596	numeric	-1.504077	8.328451	4.007333
DepCat	13	0	15	factor	0.000000	0.000000	125

Note that after using *makecategorical*, the factors of interest within *sps1* are now listed as factors rather than numeric, and that is enough to alter the analysis to something more like an anova than a regression analysis so that we obtain a parameter for each level of the factors used.

```
labelM <- c("Year","Zone","Vessel","Month","DayNight","DepCat")
sps1 <- makecategorical(labelM,sps)
mod <- makeonemodel(labelM)
mod

## LnCE ~ Year + Zone + Vessel + Month + DayNight + DepCat
## <environment: 0x0000000010ebab78>

class(mod)

## [1] "formula"
```

Each of the standardization methods we will use require that each statistical model to be examined needs to be a **formula**. If you enter *makeonemodel*, without brackets, into the R console you will see the final `form <- as.formula(form)`, which achieves this requirement.

If we are going to use a simple linear model then we can proceed using the function *dosingle* (try ?dosingle or just dosingle). We point the output of this function to the *out* object because there is an enormous amount of information generated, you can see this by using just *str(out)*.

```
labelM <- c("Year","Zone","Vessel","Month","DayNight","DepCat")
sps1 <- makecategorical(labelM,sps)
mod <- makeonemodel(labelM)
out <- dosingle(mod,sps1)
str(out,max.level=1)

## List of 7
##  $ Results  : num [1:12, 1:2] 0.855 1.351 1.26 1.077 0.949 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ StErr    : num [1:12, 1:2] 0 0.0377 0.0399 0.0413 0.0473 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ Optimum  : num 2
##  $ modelcoef: num [1:63, 1:4] 3.8949 0.3697 0.1962 -0.0137 -0.2159
...
##   ..- attr(*, "dimnames")=List of 2
##  $ optModel :List of 13
##   ..- attr(*, "class")= chr "lm"
##  $ modelG   :List of 13
##   ..- attr(*, "class")= chr "lm"
##  $ years    : Factor w/ 12 levels "2003","2004",..: 1 2 3 4 5 6 7 8
9 10 ...
```

One of the components of the *out* object is the *optModel*, which, not surprisingly, represents the optimum model. It is possible to run the generic functions *summary* and *anova*. The *summary* function (`summary(out)`) will generate the parameters (on the log-scale) and a few other details. the *anova* function determines the significance of each factor.

```
anova(out$optModel)

## Analysis of Variance Table
##
## Response: LnCE
##              Df  Sum Sq Mean Sq F value    Pr(>F)
## Year         11   371.6   33.78  35.072 < 2.2e-16
## Zone          2   809.2  404.58 420.019 < 2.2e-16
## Vessel       22   374.6   17.03  17.675 < 2.2e-16
## Month        11   281.3   25.57  26.546 < 2.2e-16
## DayNight      2   223.4  111.69 115.950 < 2.2e-16
## DepCat       14   432.7   30.91  32.087 < 2.2e-16
## Residuals 11540 11115.7    0.96
```

The Mean Year Estimates

For the lognormal model the expected back-transformed year effect involves a bias-correction to account for the log-normality; this then focuses on the mean of the distribution rather than the median:

$$CPUE_t = e^{(\gamma_t + \sigma_t^2/2)}$$

where γ_t is the Year coefficient for year t and σ_t is the standard deviation of the log transformed data (obtained from the analysis). The year coefficients were all divided by the average of all the Year coefficients to simplify the visual comparison of catch rate changes.

$$CE_t = \frac{CPUE_t}{(\sum CPUE_t)/n}$$

where $CPUE_t$ is the yearly coefficients from the standardization, $(CPUE_t)/n$ is the arithmetic average of the yearly coefficients, n is the number of years of observations, and CE_t is the final time series of yearly index of relative abundance.

All of this can be obtained in two ways. Within the *out* object there is the *Results* matrix which contains both the geometric mean estimates along with the optimum statistical model. *StErr* within *out* contains the standard error estimates for each of those.

```
cbind(out$Results,out$StErr)

##           Year   optimum       Year    optimum
## 2003 0.8551563 1.0803147 0.00000000 0.00000000
## 2004 1.3506682 1.5644932 0.03774809 0.03629678
## 2005 1.2600506 1.3154374 0.03988160 0.03897411
## 2006 1.0769724 1.0665040 0.04131560 0.04084128
## 2007 0.9487208 0.8715249 0.04731264 0.04691162
## 2008 0.8429911 0.8105254 0.04670561 0.04604055
## 2009 0.8422759 0.8031273 0.05292345 0.05183217
## 2010 0.8511003 0.8000413 0.05818629 0.05697959
## 2011 0.8379600 0.7449298 0.05251125 0.05168512
## 2012 1.0175412 0.9594120 0.04949327 0.04872046
## 2013 0.9505466 0.9162404 0.04723900 0.04669282
## 2014 1.1660166 1.0674496 0.04849594 0.04877541
```

Alternatively, if all the details are wanted there is another function *getfact*, which provides those extra details.

```
kable(getfact(out$optModel,"Year"),digits=c(4,4,4,4,4,4))
```

	Coeff	SE	LogCE	Scaled	t value	Prob
Year	1.0000	0.0000	0.0000	1.0803		
Year2004	1.4482	0.0363	0.3697	1.5645	10.1841	0.0000
Year2005	1.2176	0.0390	0.1962	1.3154	5.0330	0.0000
Year2006	0.9872	0.0408	-0.0137	1.0665	-0.3355	0.7373
Year2007	0.8067	0.0469	-0.2159	0.8715	-4.6015	0.0000
Year2008	0.7503	0.0460	-0.2884	0.8105	-6.2637	0.0000
Year2009	0.7434	0.0518	-0.2978	0.8031	-5.7462	0.0000
Year2010	0.7406	0.0570	-0.3020	0.8000	-5.2996	0.0000
Year2011	0.6895	0.0517	-0.3731	0.7449	-7.2178	0.0000
Year2012	0.8881	0.0487	-0.1199	0.9594	-2.4604	0.0139
Year2013	0.8481	0.0467	-0.1658	0.9162	-3.5513	0.0004
Year2014	0.9881	0.0488	-0.0132	1.0674	-0.2700	0.7872

The standardizations provide parameters for each level of each factor except the first level in each case. These are all assumed to have a log-trasformed value of 0.0 (= 1.0 on the nominal scale). All the other parameters (when log-normal errors are used, are proportional to the first level). Thus the LogCE column is the output from the standardization. The bias-adjusted transformation back to the nominal scale is described in the equations above. The 'Scaled' column is the same as the 'Coeff' column except it is has been divided through by the mean of the series. This sets the average value to 1.0, which permits simple visual comparison with other time-series. The 'SE' column provides the basis for generating the log-normally distributed confidence intervals.

```
# plotprep(width=7,height=4.5)
plotstand(out,bars=TRUE)
```

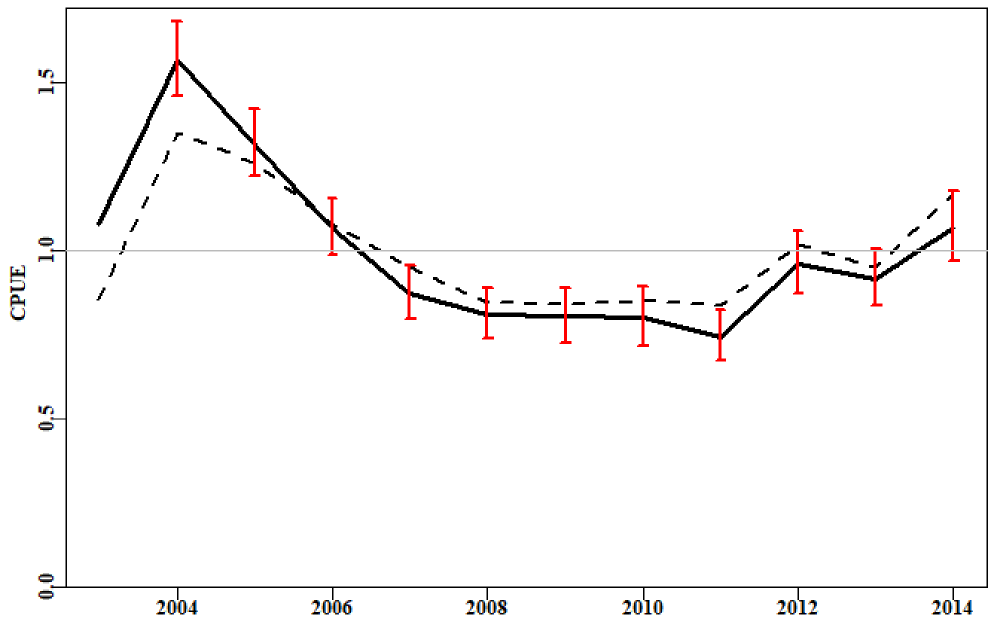


Figure 9. The standardization of the cpue data within the *sps* data set. The dashed line is the geometric mean CPUE while the solid line with 95% confidence intervals is the standardized CPUE. In places the difference between the standardized CPUE and the geometric mean CPUE is greater than the 95% log-normal confidence intervals.

One issue with this plot is that the scale makes little sense to Industry members who are more used to the nominal scale at which they operate personally. Given that the average of both the geometric mean and the optimum model is 1.0, both can be multiplied by a constant to rescale the plots. If we calculate the geoemtric mean CPUE for the whole fishery we can use that as a multiplier and that will place each time-series on a recognizable nominal scale. This can be done using the function *geomean* and include the *geo* option of *plotstand*.

```
# plotprep(width=7,height=4.5)
geom <-geomean(sps$CE)
plotstand(out,bars=TRUE,geo=geom)
```

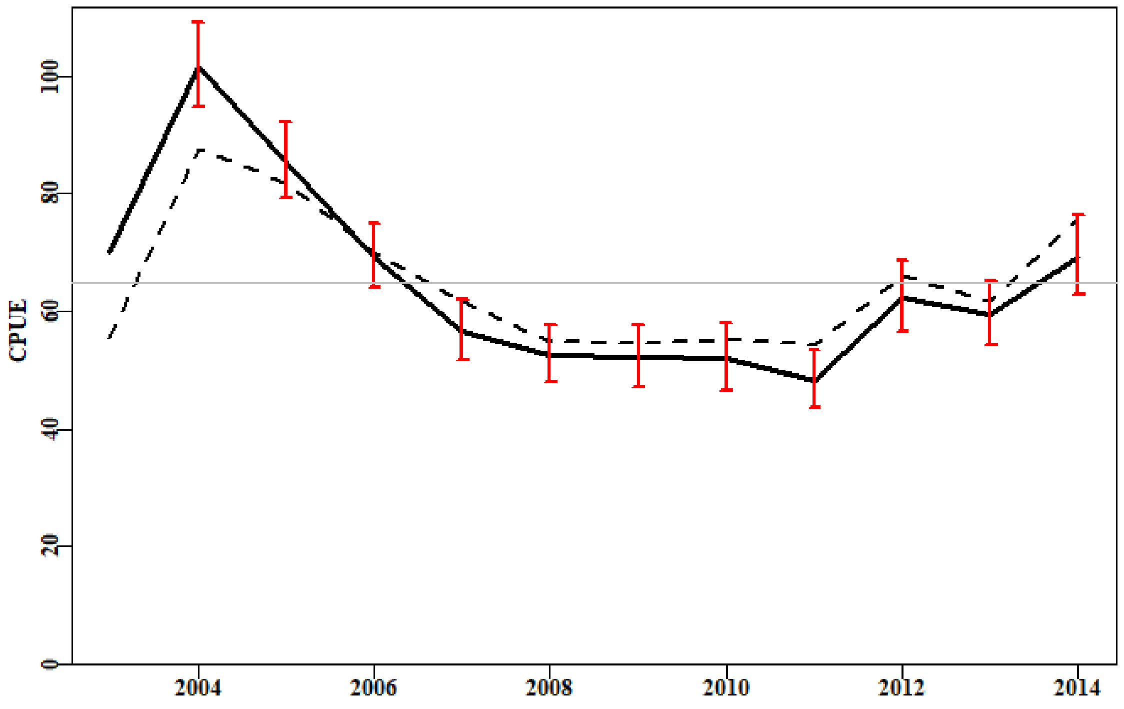


Figure 10. The standardization of the cpue data within the *sps* data set. The dashed line is the geometric mean CPUE while the solid line with 95%. These trajectories both have an average of the overall geometric mean CPUE.

It is often helpful to examine the standardizations as they increase in complexity so that the relative influence of each factor becomes more clear. However, to do this would require a little more R code.

```
# first make a matrix to hold the results
labelM <- c("Year","Zone","Vessel","Month","DayNight","DepCat")
columns <- c("adjR2","incR2","RSS","MSS","Npar","nobs","AIC")
nummod <- length(labelM)
results <- as.data.frame(matrix(0,nrow=nummod,ncol=length(columns),
                          dimnames=list(labelM,columns)))
for (i in 1:nummod) {  # sequentially build the models
   mod <- makeonemodel(labelM[1:i])  # When i = 1 LnCE ~ Year
   out <- dosingle(mod,sps1)
   outsum <- summary(out$optModel)
   aov <- anova(out$optModel)    #  Extract a range of results
   RSS <- tail(aov$"Sum Sq",1)
   df <- aov$Df
   nobs <- sum(df) + 1
   numfact <- length(df) - 1
   npars <- sum(df[1:numfact]) + 1
   AIC <- nobs * log(RSS/nobs) + (2 * npars)
   results[i,] <- c(outsum$adj.r.squared,NA,RSS,sum(aov$"Sum Sq") -
RSS,npars,nobs,AIC)
}
results[2:nummod,"incR2"] <- results[2:nummod,"adjR2"]-
results[1:(nummod-1),"adjR2"]
round(results,4)
```

```
##             adjR2  incR2      RSS       MSS Npar  nobs       AIC
## Year       0.0264     NA 13236.77  371.6062   12 11603 1552.5217
## Zone       0.0857 0.0594 12427.62 1180.7589   14 11603  824.6352
## Vessel     0.1116 0.0259 12053.06 1555.3196   36 11603  513.5497
## Month      0.1315 0.0199 11771.79 1836.5924   47 11603  261.5701
## DayNight 0.1478 0.0163 11548.41 2059.9662   49 11603   43.2834
## DepCat     0.1788 0.0309 11115.70 2492.6736   63 11603 -371.8236
```

By looking at the increments to the adjusted-R2 it is clear that the factor *DepCat* has a larger impact on the variation accounted for than even *Vessel*, so strictly the analysis should be repeated after re-ordering the different factors within labelM. The *AIC* column identifies the optimum combination of factors with the smallest value indicating the optimum. It would be worthwhile repeating the analysis with the re-ordering. Typically, if one plots each standardization on the same plot, typically, while the later factors can be statistically significant, their effect upon the trajectory of the standardized CPUE can be minimal or appear to contribute mainly noise. If the standardization is to be used within an assessment it is the trend that matters so those final few factors may only have a minor effect.

Alternative Standardization Strategies

So far we have only considered General Linear Models (which with log-normal errors give the same results as simple linear models). If we wish to use alternative residual error structures then it would be necessary to use true GLMs (as in Generalized Linear Models). These would be necessary if, for example, there was a wish to attempt using perhaps a Gamma distribution instead of log-normal, then would need to use different syntax. The standard approach when using the Gamma distribution would be to use a log-link in the GLM. In such cases then the dependent variable would then be *CE* rather than *LnCE*. The functions described so far are designed for use with log-normal residual errors that need a bias-correction Gamma residual erros do not require such a bias-correction so we will need to work directly with the estimated coefficients.

```
labelM <-
c("Year","Zone","Vessel","Month","DayNight","DepCat","Month:Zone")
sps1 <- makecategorical(labelM,sps)
mod <- makeonemodel(labelM,dependent="CE")

model4 <- glm(mod,family=Gamma(link="log"),data=sps1)
m4 <- summary(model4)$coefficients  # combine these with empty first
year
yrval <- rbind(c(0,0,0,0),m4[grep("Year",rownames(m4)),])
gamres <- cbind(yrval,exp(yrval[,"Estimate"]))
rownames(gamres) <- 2003:2014
gamres
```

```
##            Estimate Std. Error     t value     Pr(>|t|)
## 2003  0.000000000 0.00000000  0.00000000 0.000000e+00 1.0000000
## 2004  0.333489137 0.04141384  8.05260025 8.901627e-16 1.3958299
## 2005  0.232479826 0.04412451  5.26872275 1.398554e-07 1.2617250
## 2006  0.003738450 0.04620812  0.08090461 9.355192e-01 1.0037454
## 2007 -0.123979553 0.05282090 -2.34716857 1.893353e-02 0.8833979
## 2008 -0.139742192 0.05204407 -2.68507408 7.261757e-03 0.8695824
## 2009 -0.226946891 0.05850235 -3.87927840 1.053499e-04 0.7969631
## 2010 -0.079275668 0.06427291 -1.23342264 2.174433e-01 0.9237852
## 2011 -0.147106194 0.05832505 -2.52217851 1.167642e-02 0.8632023
## 2012 -0.080927349 0.05495370 -1.47264600 1.408738e-01 0.9222607
## 2013 -0.002500419 0.05314454 -0.04704940 9.624747e-01 0.9975027
## 2014  0.023195520 0.05528536  0.41955987 6.748148e-01 1.0234666

plotstand(out,bars=TRUE)
lines(2003:2014,exp(yrval[,"Estimate"]),lwd=2,col=4)
```

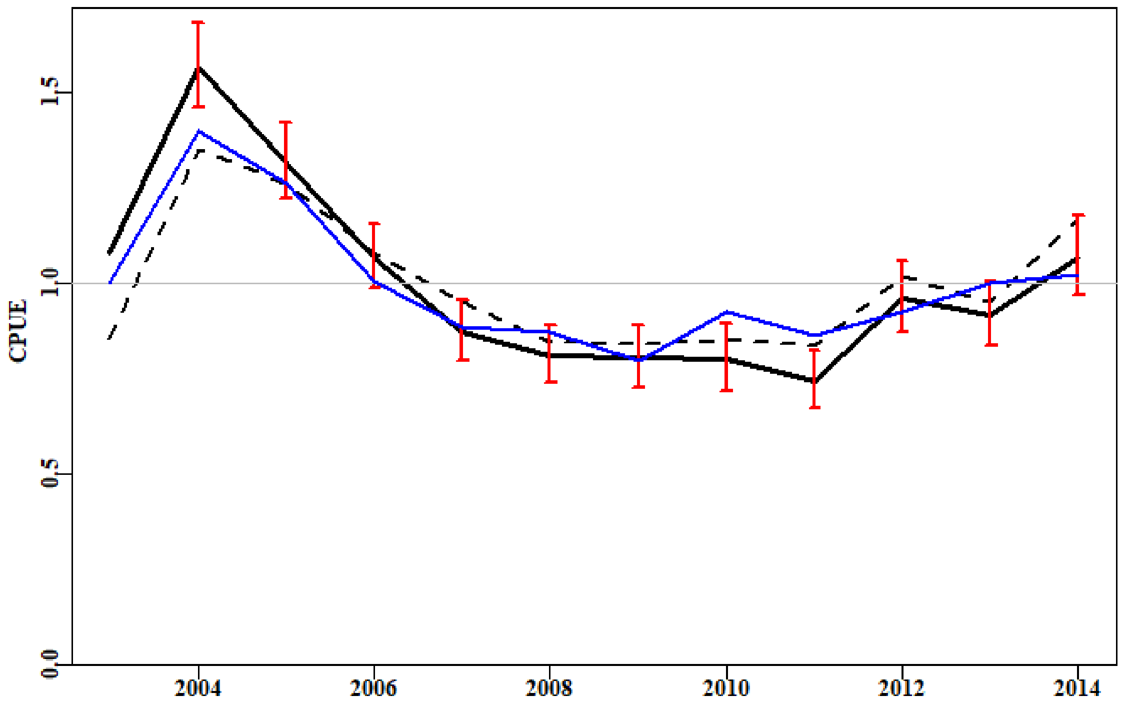


Figure 11. The standardization of the cpue data within the *sps* data set comparing a LM using log-normal with a GLM using Gamma residual errors. The dashed line is the geometric mean CPUE while the solid black line with 95% is the log-normal error standardization. Finally, the blue line is the Gamma error standaridzation.

The Use of GAMs

Generalized Additive Models are an extension of GLMs in which at least some of the factors are replaced by fitting smooth surfaces to some of the factors that are considered to have a non-linear relationship with catch rates.

In order to run them, however, it is necessary to install a number of additional R packages. As an example, we could use a GAM to add a smoother to the Lat - Long data in the sps data set. We would actually use the sps1 data set as the remaining categorical factors are also included in the analysis. A possible workflow might involve the following code.

```
# install and call these R packages and their dependencies
library(nlme)
library(mgcv)
library(gamm4)
# note the use of gam rather than lm or glm (see the examples in ?gam
for more
# details.
modelGam <- gam(LnCE ~ s(Long,Lat) + Year + Zone + Vessel + Month +
                   DayNight + DepCat, data = sps1)
anova(modelGam)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## LnCE ~ s(Long, Lat) + Year + Zone + Vessel + Month + DayNight +
##     DepCat
##
## Parametric Terms:
##          df       F p-value
## Year     11  54.145  <2e-16
## Zone      2   0.778   0.459
## Vessel   22  16.549  <2e-16
## Month    11  26.175  <2e-16
## DayNight  2 112.842  <2e-16
## DepCat   14   9.269  <2e-16
##
## Approximate significance of smooth terms:
##                edf Ref.df     F p-value
## s(Long,Lat) 26.96  28.69 19.14  <2e-16
```

We should not be surprised that the *Zone* factor is no longer significant. By including the Lat - Long surface including the *Zone* factor becomes redundant so we should really repeat the analysis without *Zone* included.

```
modelGam <- gam(LnCE ~ s(Long,Lat) + Year + Vessel + Month +
                   DayNight + DepCat, data = sps1)
anova(modelGam)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## LnCE ~ s(Long, Lat) + Year + Vessel + Month + DayNight + DepCat
##
## Parametric Terms:
##          df       F p-value
## Year     11  54.039  <2e-16
## Vessel   22  16.635  <2e-16
## Month    11  26.237  <2e-16
## DayNight  2 112.859  <2e-16
## DepCat   14   9.297  <2e-16
##
## Approximate significance of smooth terms:
```

```
##                edf Ref.df     F p-value
## s(Long,Lat) 27.14  28.74 26.65  <2e-16
```

We can use the *getfact* function to extract the results we need. The Coeff column contains the LogCE transformed back to the linear scale and Scaled is the Coeff re-scaled to a mean of 1.0. Once again if it is desired to scale this to the nominal CPUE from the fishery so as to improve communication with Industry and managers then we can use *geomean* to estimate the overal geometric mean to re-scale the 'Scaled' column to something more meaningful to industry members.

```
answer <- getfact(modelGam,"Year")
opti <- answer[,"Scaled"]
round(answer,5)

##              Coeff      SE    LogCE  Scaled  t value    Prob
## Year      1.00000 0.00000  0.00000 1.11779       NA      NA
## Year2004 1.43698 0.03557  0.36191 1.60624 10.17322 0.00000
## Year2005 1.18326 0.03826  0.16754 1.32263  4.37886 0.00001
## Year2006 0.94500 0.04009 -0.05737 1.05631 -1.43110 0.15243
## Year2007 0.76003 0.04616 -0.27546 0.84955 -5.96784 0.00000
## Year2008 0.72439 0.04523 -0.32344 0.80972 -7.15074 0.00000
## Year2009 0.70430 0.05095 -0.35185 0.78725 -6.90523 0.00000
## Year2010 0.69154 0.05594 -0.37039 0.77300 -6.62073 0.00000
## Year2011 0.66716 0.05075 -0.40601 0.74575 -7.99956 0.00000
## Year2012 0.85665 0.04794 -0.15587 0.95755 -3.25124 0.00115
## Year2013 0.82559 0.04588 -0.19271 0.92283 -4.19995 0.00003
## Year2014 0.94059 0.04789 -0.06240 1.05138 -1.30282 0.19266
```

We can gain an impression of the surface fitted to the Lat - Long data using the *plot* function, which recognizes the output from a gam and can react accordingly.

```
#plotprep(width=4.5,height=7)
 plot(modelGam,ylim=c(-44.5,-
40),xlim=c(143.5,146.5),se=FALSE,xlab="",ylab="")
 title(ylab=list("Latitude", cex=1.0, font=7),
       xlab=list("Longitude", cex=1.0, font=7))
 plotLand("pink")
```

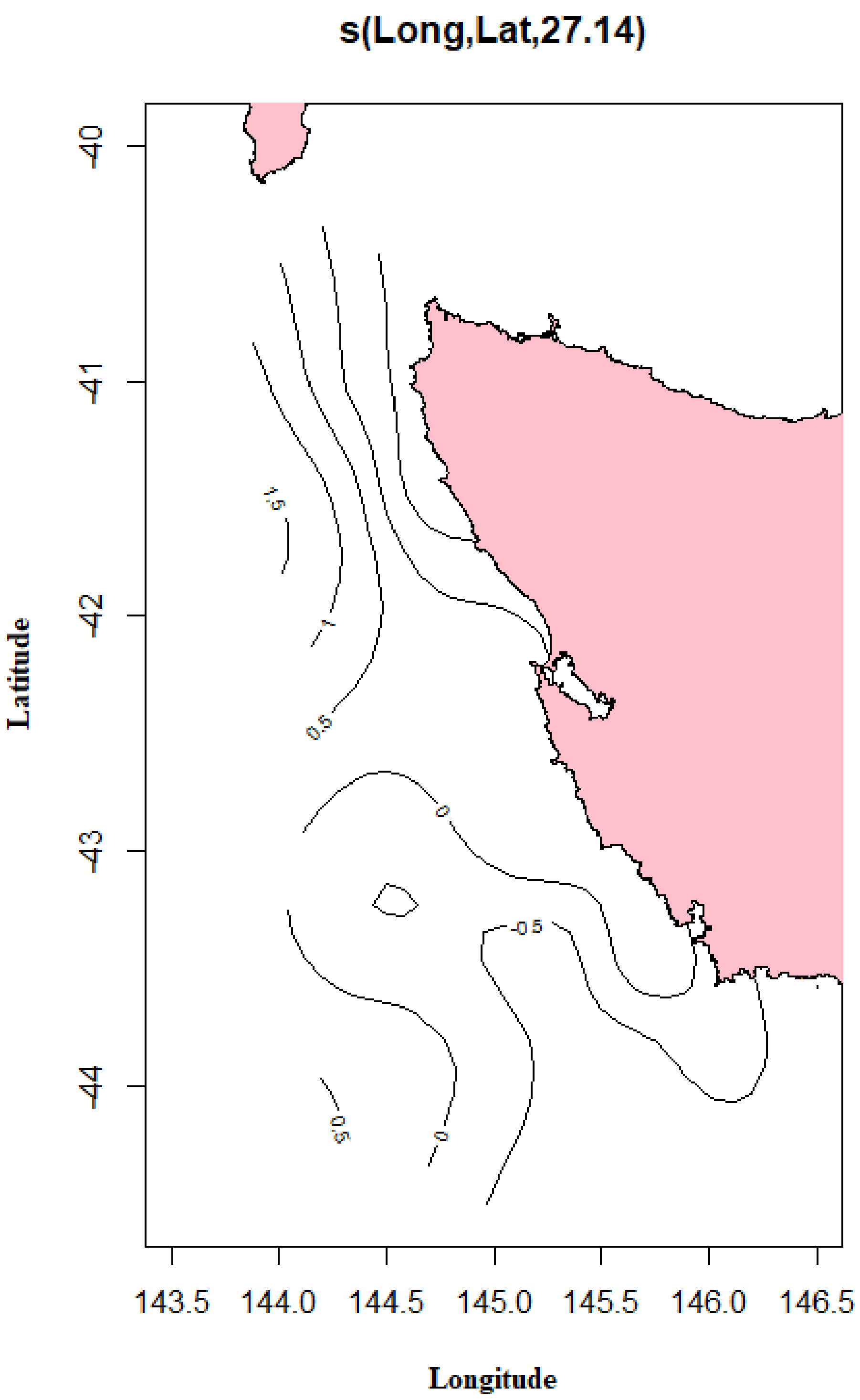


Figure 12. A plot of the surface fitted to the output from the gam function.

The effect on the year parameters is what we are really interested in for the purposes of stock assessment and we can compare the outcome of the GAM with the previous GLM.

```
#plotprep(width=7,height=4.5)
plotstand(out,bars=TRUE)
lines(facttonum(out$years),opti,col=4,lwd=2)
legend("bottomleft",c("GLM","GAM"),col=c(1,4),lwd=3,bty="n",cex=1.2)
```

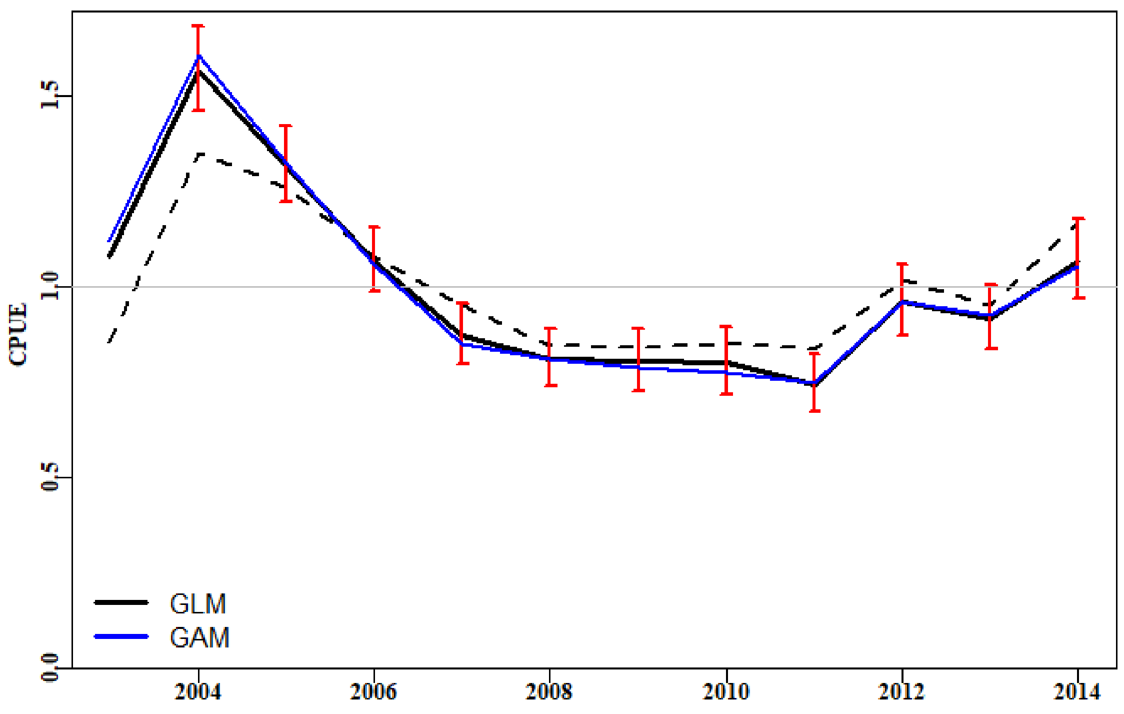


Figure 13. The standardization of the cpue data within the *sps* data set comparing an LM using log-normal residual errors with a GAM having a surface fitted to the Lat/Long data. The dashed line is the geometric mean CPUE while the solid black line with 95% is the log-normal error standardization. Finally, the blue line is the Generalized Additive Model.

catch-MSY

Introduction

Which Stock Assessment?

Which stock assessment method to apply to fisheries for data-poor to data-moderate species will depend upon what fisheries and biological data are available but also, importantly, on what management objectives need to be met within the jurisdiction in question. It may be the case that the fishery for a particular species is of sufficient size and value to warrant on-going monitoring and management towards some defined goal for the stock. In such a case the assessment used should obviously be capable of generating some notion of the current state of the fishery and indicate what management actions may be required to eventually achieve the agreed management goals. But some fisheries may be so minor that trying to actively manage them would be inefficient both practically and economically. Nevertheless, to meet the requirements of the Status of key Australian Fish Stocks (SAFS) one still requires some form of defensible stock assessment capable of determining whether the current level of fishing is sustainable.

Modified Catch-MSY

The Catch-MSY method (Martell and Froese, 2013) could be termed a 'model-assisted' stock assessment method. It only requires a time-series of catches and a set of strong assumptions to conduct a stock assessment. As only a brief description of how it is considered to work is given here, it is recommended that users read the original paper to gain an understanding of what the method does and how it does it.

The underlying stock dynamics are described by the simple model used, which in the case implemented here is a Schaefer surplus production model with parameters *r*, the population growth rate, and *K*, the population carrying capacity or unfished biomass. The model uses ratios of the initial and final catches relative to the maximum catch to set up arrays of potential values for the initial and final depletion levels as well as for the potential range of *r* and *K* values (all of which are now modifiable by the user). The method sequentially steps through the years of the fishery by randomly selects pairs of *r*-*K* values from the wide initial ranges, which defines the initial biomass, subtracting the catches, and moving the population dynamics forward each year using the predictions from the simple model. Essentially this is a stock reduction that removes catches from a known set of dynamics. However, the very many *r*-*K* pairs used (at least 20000) are combined with a fixed set of initial depletion levels (about 20 steps between the minimum and maximum initial depletion set) to generate often 100,000s of possible stock reduction trajectories. Criteria are included (e.g. no trajectory is kept if it predicted zero biomass or biomass above *K*) that lead to numerous potential trajectories being rejected. Those that are left after all criteria for acceptance have been completed constitute the set of trajectories deemed to be consistent with the known catches. The implications of these successful trajectories are used to produce an assessment of the possible status of the stock.

In this section we will describe how to conduct a catch-MSY analysis, how to extract the results from that analysis, as well as plot out illustrations of those results. In addition, we will examine how to project the successful trajectories under constant catch scenarios to determine what level of catches should lead to the majority of trajectories moving in a desired direction (rebuilding to a target, staying stable, or declining to a target depletion).

A standard workflow might consist of:

1. read in data and use **checkdata** to determine which analyses are possible
2. set the run-time parameters to conduct the desired catc-MSY analysis (see later)
3. use **run_cMSY** to conduct a catch-MSY analysis. One sensitivity you should run is to set sigpR (the proxy for process error) to 1e-10 (a very small number) to see deterministic trajectories. Do other sensitivities.
4. use **cMSYphaseplot** to determine status for SAFS, if more is wanted:
5. use **summarycMSY** to generate a summary object of the answer.
6. use **plottrajectory** to illustrate either all successful trajectories along with the successful harvest rates (use option *oneplot = TRUE*), or just a sample of 7 or 15 biomass trajectories.
7. use **plotcMSY6** to plot up the overview of successful and failed r - K combinations.
8. use **pulloutStats** to obtain summary statistics regarding MSY and depletion
9. use **plotconstC** with $deplet from the object output from *pulloutstats* to plot up the successful trajectories depicting stock depletion.
10. use **doconstC** to conduct constant catch projections to find a level of catch that lead, on average, to the population decreasing, staying stable, or increasing (possibly to some selected target within a particular time).
11. optionally, use **trendMSY** to illustrate the relationship between the estimated MSY and the average MSY of the successful trajectories.

Some Formal Details

The Catch-MSY method described here can be regarded as a model-assisted data-poor method. It uses a form of stock reduction analysis where the productivity of a given stock (its unfished biomass and its reproductive rate) is characterized within the parameters of a simple mathematical model, and how that modelled stock responds to the history of known catches (a stock reduction analysis) forms the basis of the alternative methods used to characterize productivity in management useable terms.

The Catch-MSY method (Martell and Froese, 2013) uses the relatively simple Schaefer surplus production model as the basis for describing the dynamics of the stock being described.

$$B_{t+1} = B_t + rB_t\left(1 - \frac{B_t}{K}\right) - C_t$$

where B_t represents the stock biomass in year *t*, *r* represents a population growth rate that includes the balance between recruitment and natural mortality, *K* is the maximum population size (the carrying capacity), and C_t being the catch in year *t*. The $\left(1 - \frac{B_t}{K}\right)$ represents a density dependent term that trends linearly to zero as B_t tends towards *K*.

Importantly, for our purposes, one of the properties of the discrete Schaefer surplus production model is that MSY can be estimated very simply from the parameter estimates:

$$MSY = \frac{rK}{4}$$

which reflects the symmetric production function implied by the model dynamics. A relatively simple future possible development would be to include the option of using Fox model dynamics instead of the Schaefer.

Such surplus-production (or biomass dynamic) models usually require both a time-series of total catches (landings plus discards) and a time-series of an index of relative abundance (Haddon, 2011). In Australia the index of relative abundance is most often a time-series of CPUE (ideally standardized CPUE).

Empirical Harvest Strategies

In the Southern and Eastern Scalefish and Shark Fishery (SESSF), rather than using surplus production models or other simple approaches that attempt to model the underlying population dynamics of a stock, empirical harvest strategies have been developed that use such time-series in empirical relationships that give rise directly to management related advice on catch levels (Little et al., 2011; Haddon, 2014). Such empirical harvest strategies can provide the needed management advice but do not determine stock status unless the reference period, often used in such approaches, is assumed to be a proxy for the target reference point (and associated limit reference point) for sustainability. A weight-of-evidence argument would need to be made to support the use of such a proxy. In the SESSF, this so-called Tier 4 harvest strategy is used to determine whether a stock is over-fished or not but currently cannot be used to determine whether over-fishing is occurring. In addition, there is the strong assumption made that the commercial catch rate are a direct reflection of the stock biomass. There are, however, some species, for example mirror dory (*Zenopsis nebulosa*) where catch rates increase when catches increase and then decline once catches begin to decline. They appear to be fisheries based on availability rather than the fishery being the major influence on the stock biomass and other aspects of the environment of the species appear to be driving its dynamics. The use of CPUE may this be misleading in such cases or at best lead to simply reactive management decisions (cpue goes up so can catches, cpue goes down so must catches).

More widely than the SESSF there are many fisheries within Australia that may only have a time-series of catches with only limited information related to a useable index of relative abundance. In addition, such catch time-series may not be available from the beginning of the fishery, which means that methods such as Depletion-Based Stock Reduction Analysis (Dick and MacCall, 2011) cannot be validly applied (although, as shown in Haddon et al, 2015, if sufficient years of catches are present (perhaps >25) then the method can still provide approximate estimates of management related parameters). Under such data-limited situations other catch-only based assessment methods can provide the required estimates of management interest.

Stock Reduction Analyses

As with many of the more capable catch-only data-poor approaches the Catch-MSY method evolved from the stock reduction analyses of Kimura and Tagart (1982), Kimura et al. (1984), and eventually Walters et al. (2006). It uses a discrete version of the Schaefer surplus production model (Schaefer, 1954, 1957; Haddon, 2011) to describe the stock dynamics in each case. The Catch-MSY requires a time-series of total removals, prior ranges for the r and K parameters of the Schaefer model, and possible ranges of the relative stock size (depletion levels) in the first and last years of the time-series. As described by Martell and Froese (2013) the range of initial depletion levels can be divided into a set of initial values, and a stock reduction using the known total removals, applied to each of these multiple initial depletion levels combined with pairs of *r-K* parameters randomly drawn from uniform distributions across the prior ranges of those parameters. Each of these parameter pairs plus each of the initial depletion levels are projected using the total catch trajectory leading to a stock biomass trajectory which is either accepted or rejected depending on whether the stock collapses or exceeds the carrying capacity, and whether the final depletion level falls within the assumed final range.

The initial and final depletion ranges can be relatively broad. Other criteria can be included to further constrain the biomass trajectories if extra evidence is available. Such additional constraints are still under development. For example, in some of the examples you will notice that the annual harvest rates for some accepted trajectories can be very high (> 0.5), which for many (though not all) Australian species can be considered to be implausible. Now it is possible to conduct a sensitivity analysis where trajectories implying some pre-defined harvest rate will also be rejected. These high fishing mortality trajectories are only possible for the more productive parameter combinations so removing such trajectories will likely reduce the predicted MSY (maximum productivity).

We can use the **invert** data set of catches to exemplify the process of applying the Catch-MSY method.

The data-file format

As can be seen in the R-code above, the data for the catch_MSY analysis is read into an R object containing fish, glb, and others if present. Using the *data* command places the data object, in the example it is called *invert*, into the global environment. Using the *readdata* command with a "named.csv" file also produces one large object. The format of the standard data files is given in three example appendices at the end of this document. The actual data requirements of the different methods available differs greatly. The catch-MSY method really only requires two columns in *fish* one being "year" the other being "catch". The surplus production modelling also requires a "cpue" column, and the age-structured surplus production model also requires various biological properties relating to weight-at-age, maturity-at-age, and selectivity-at-age. These are tested by the *checkdata* function.

Table 1. The format of the fish object. Not all fields are required for all analyses.

year	catch	cpue	se	geom	vessel	records
1986	283.434	0.7005	0.000	71.7	47	1592
1987	185.612	0.8842	0.038	93.0	47	1764
1988	285.942	0.9786	0.041	124.6	41	1395
1989	234.523	0.8371	0.043	139.3	39	1143
1990	329.837	1.5811	0.049	174.5	25	727
1991	461.521	1.3874	0.050	182.9	29	734
1992	668.453	1.0286	0.058	166.3	19	434
1993	574.393	1.2245	0.050	172.4	21	673
1994	482.073	1.1569	0.050	170.3	26	661
1995	529.336	0.9135	0.044	105.0	25	1070
1996	424.963	0.8085	0.042	95.4	25	1216
1997	473.406	0.7581	0.047	86.8	21	855

The use of *checkdata* indicates that both *catch-MSY* and *spm* analyses are possible with this data. This does not mean such analyses will always be valid with the given data only that the required data to conduct these analyses are present.

The *answer* object contains all the results from the catch-MSY analysis and is used by other R functions to generate summaries and plots of those results.

```
#library(simpleSA)

glb <- invert$glb     # contains available biological data
checkdata(invert)

##                    Method Possible
## catch-MSY            TRUE     TRUE
## spm                  TRUE     TRUE
## aspm                 aspm    FALSE
## catch-curves catch-curves    FALSE

# normally one would run at least 20000 iterations, preferably more
reps <- 5000
# read the help for run_cMSY to understand each input parameter
answer <- run_cMSY(fish,glb,n=reps,sigpR=0.025,maximumH=1.0)
str(answer,max.level=1)

## List of 13
##  $ R1               :List of 3
##  $ ell              : num [1:1011, 1:23] 0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ rK               : num [1:1011, 1:5] 0.345 0.168 0.309 0.219 0.22
...
##   ..- attr(*, "dimnames")=List of 2
##  $ parbound         :List of 5
##  $ Statistics       :List of 3
##  $ Rfirst           :List of 3
##  $ firstparbound    :List of 5
##  $ startbd          : num [1:23] 0.15 0.175 0.2 0.225 0.25 0.275 0.3
0.325 0.35 0.375 ...
##  $ outtab           :'data.frame':    11 obs. of  2 variables:
##  $ initialDepletion: num [1:5000, 1:23] 0.15 0.175 0.2 0.225 0.25
0.275 0.3 0.325 0.35 0.375 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ finaldepletion  : num [1:5000, 1:23] 0 0 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ B0               : num [1:5000, 1:23] 6311 4751 2683 2462 2275
...
##   ..- attr(*, "dimnames")=List of 2
##  $ MaximumH         : num 1
```

SAFS Status

A summary and illustration of the stock status can be obtained by extracting the average stock biomass trend, along with the average fishery harvest rate trend from the successful trajectories.

Using the average estimate of the r - K pairs it is also possible to generate an estimate of the production curve, from which it is possible to derive estimates of B_{msy}, *0.2K* or *0.2B0*, and H_{targ} and H_{lim}, with which the phase plot can be subdivided to provide a visual representation of the how the history of catches from the fishery are reflected in predicted changes in the biomass and harvest rate. Whether over-fishing is occurring (leading to a status of 'depleting') is determined by whether the current point lies above the H_{targ} or above the H_{lim}, which in turn is decided by the management objectives adopted in each jurisdiction. A status of 'depleted' or sustainable currently corresponds to whether the current year's point is to the left or right of the $0.2B_0$ line.

```
out <- cMSYphaseplot(answer,fish)
```

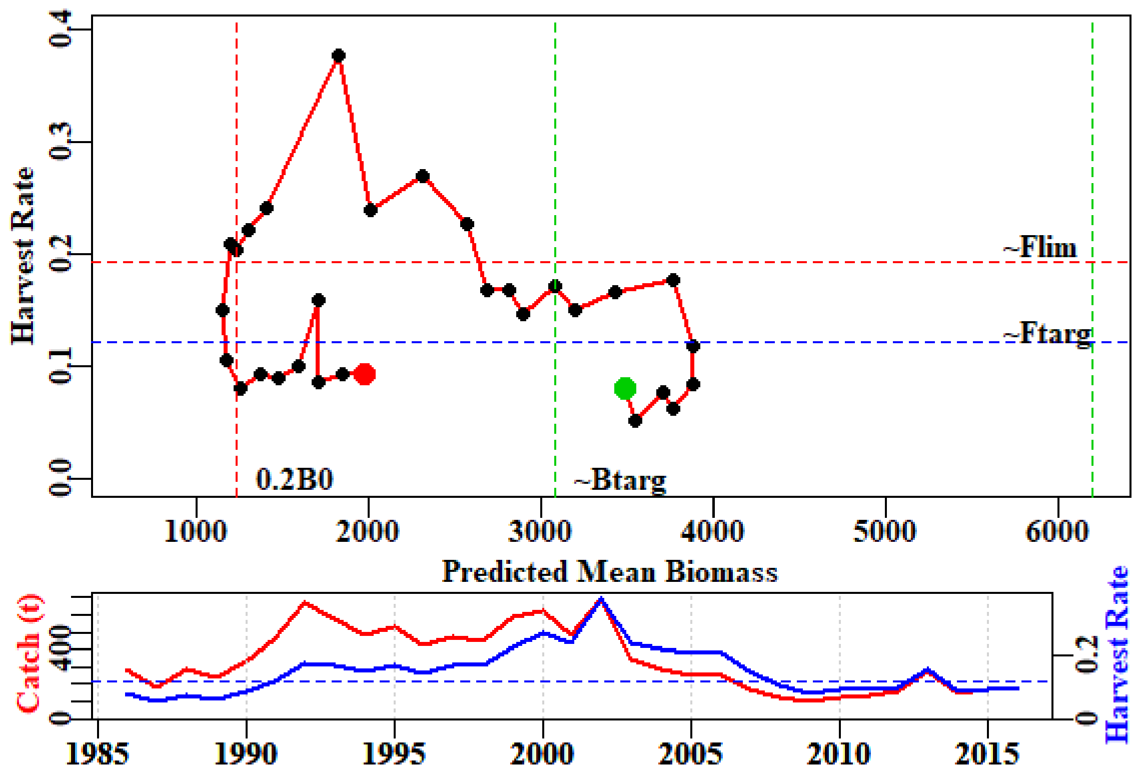


Figure 14. A phase plot of the man predicted biomass and harvest rates through the years observed. The first year of data is a green point and the last a red point. The axes for the bottom plot are identified through the colour of the axis titles

The results are synthesized within the contents of *out*.

```
str(out)

## List of 6
##  $ medianB: Named num [1:31] 3499 3552 3711 3775 3887 ...
##   ..- attr(*, "names")= chr [1:31] "1986" "1987" "1988" "1989" ...
##  $ medianH: Named num [1:31] 0.081 0.0522 0.0771 0.0621 0.0849 ...
##   ..- attr(*, "names")= chr [1:31] "1986" "1987" "1988" "1989" ...
##  $ msy    : num 375
##  $ Bmsy   : num 3088
##  $ Hmsy   : num 0.121
##  $ Hlim   : num 0.193
```

Other Potential Outputs from catch-MSY

A useful summary of the catch-MSY analysis containing the primary results can be obtained using the *summarycMSY* function.

```
summcMSY <- summarycMSY(answer,fish,final=TRUE)
str(summcMSY,max.level=1)  # try max.level = 2
```

```
## List of 11
##  $ countcolour: Named num [1:5] 3989 694 210 89 18
##   ..- attr(*, "names")= chr [1:5] "red_0" "black_2" "blue_4"
"yellow_6" ...
##  $ meanmsy    : num [1:4, 1:2] 381 375 381 278 72 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ meanr      : num [1:4, 1:2] 0.252 0.2419 0.2522 0.14 0.0723 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ meanK      : num [1:4, 1:2] 6315 6197 6322 3911 1176 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ r          : num [1:1011] 0.345 0.168 0.309 0.219 0.22 ...
##  $ K          : num [1:1011] 5526 6913 5047 7823 6690 ...
##  $ msy        : num [1:1011] 477 290 390 428 368 ...
##  $ pickC      :List of 6
##  $ years      : num [1:31] 1986 1987 1988 1989 1990 ...
##  $ parbound   :List of 5
##  $ fish       :'data.frame': 31 obs. of  7 variables:
```

As can be seen from the series of large objects in *summcMSY*, such as the *r*, *K*, and *msy* components, only a portion of the *n=5000* trials succeeded in meeting the constraints that define an acceptable trajectory. This has removed all the parameter combinations that were not productive enough or those that were too productive and retained only those that were at least realistic/plausible enough to be consistent with what is known about the fishery. If you run the last three lines of R code a few times and examine the structure of the *summcMSY* object you will notice that each time the length of the *r*, *K*, and *msy* objects usually differs. This merely exemplifies the fact that the randomly selected parameter combinations lead to different numbers of successful trajectoris each time through. To obtain a visual representation of a selection of the successful trajectories you can use the function *plottrajectory* (see its help function for a description of all the options).

It should be noted that the catch-MSY method uses a two stage strategy. Unless set otherwise, it first sets the initial *K* values between the maximum catch and 60 times the maximum catch. This invariably leads to very many successful trajectories that suggest a very large intiial biomass combined with a very low population growth rate. While mathematically this may match the productivity of the stock, as suggested by the time-series of catches, it is biologically less plausible than lower *K* values associated with higher *r* values (these two parameters are negatively correlated). To avoid the less plausible combinations in their original code Martell & Froese (2013) search for the smallest *K* value that will still give rise the overall average *MSY* value across the successful trajectories found in the first run through. Once a more restricted initial *K* range has been found then the *n* replicates are repeated and the final result put into the *R1* object inside *answer*. The results from the first run through the replicates are put inside the *Rfirst* object.

Once the *summcMSY* object has been generated (**Figure 16**) the results can be summarized by using the *plotcMSY6* function. This generates a plot of the combinations of *r* and *K* and whether they succeeded or not. In addition it displays the distribution of the successful *r*, *K*, and *MSY* values.

```
out <- plottrajectory(answer$R1,fish$year,fish$catch,answer$parbound,
                      oneplot=FALSE,Bmax=25000,
                      scalar=1.0,plotout=TRUE,plotall=15)
```

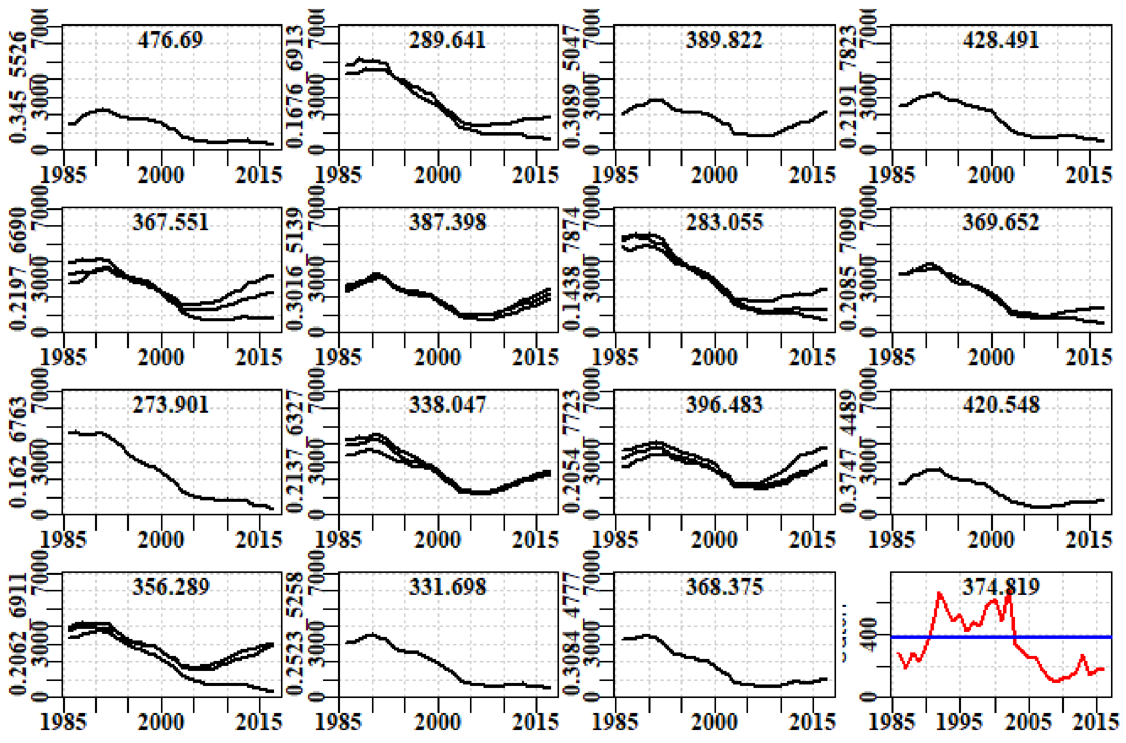


Figure 15. 15 examples of different *r-K* combinations only illustrating those trajectories that were plausible, the number at the top of each plot is the predicted MSY for the given parameter pair. The r and K values are the y-axis labels in each case. The final plot is the catch history with the predicted average MSY as the blue line.

```
plotcMSY6(summcMSY,fish[,"catch"],label=glb$spsname)
```

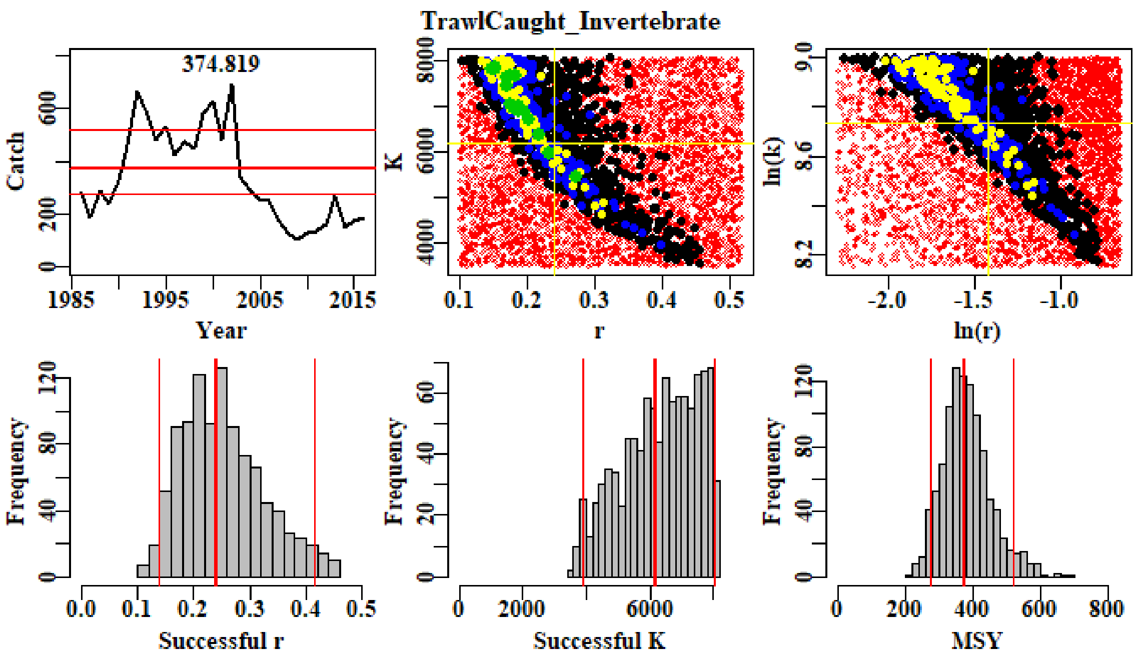


Figure 16. A plot of the catch trajectory with the MSY and its 90% percentiles. The two colour plots are a plot of the *K* vs *r* combinations with the red dots depicting failure and then the increasing colours depicting more combinations of initial depletion that succeeded for each *r-K* pair. The right-hand plot is the log-transformed version of the left-hand plot. The histograms describe the distributions of the successful *r-K* pairs and the resulting MSY. The red lines are the median and the 90th percentile confidence intervals.

Interpretation of the plotcMSY6 plot

The histograms along the bottom row of **Figure 16** illustrate the distribution of the successful values of each parameter and the resulting MSY from the parameter pairs. Because of the truncation of the *K* values in the two phase operation of the analysis it should not be surprising if the 'Successful K' histogram is flat up against the right hand bound. However, if either of the *r* or left-hand side of the *K* plots is hard up against the bounds, it would be worthwhile exploring whether different initial values of the *r* and *K* values are less constrained. This can be done using a different *resilience* value or by explicitly altering the *start_r* and the *start_K* values. Around the jurisdictions examples were seen where parts of the *r* or *K* histograms were flat topped, which suggests there is insufficeint information in the catches to adequately discriminate between potentially successful trajectories and failures.

Reading Martell and Froese (2013) one can see that the method relies on the notion that a fishery is expected to begin small, build up catches, and then those catche slead to depletion so that they cannot be maintained and the catches drop. Of course, in Australia there can be many reasons for catches to drop other than the stock becoming depleted. There may be management interventions that lead to reduced catches (imposed catch limits, large marine closures, gear restrictions, and other management steps). In such situations where catches are reduced before stock depletion leads to reduced catches the catch-MSY seems likely to generate conservative estimates of sustianable production. Other issues that can arise are that catches have only ever increased or stayed stable. Without an on-going decline in catches at some period in the fishery, the model used inside the catch-MSY will have no may in which to characterize maximum productivity and it may give somply an approximate estimate of the average catches. Obviously care is needed when interpreting the outputs from the catch-MSY analysis.

The Catch-MSY method requires a catch history and some prior notion of the relative resilience or expected productivity of the species being fished. The basic idea is that the method begins with a given range of initial depletion and final depletion levels along with initial ranges of the two parameters describing the Schaefer model; these are the *r* and *K* parameters representing the un-restricted population growth rate and the population carrying capacity respectively. The catch-MSY method is based on a review of very many catch histories and relies on an expectation of catches increasing as a fishery develops and then decreasing as the catches impact the stock. The statistics can begin at the start of a fishery or after it has already developed, hence the initial depletion is set by comparing the initial catches with the maximum catches. The initial ranges for the model parameters depend on the assumed productivity or resilience of the species of which four options exist - "verylow", "low", "medium", and "high", with associated *r* ranges of (0.015 - 0.125), (0.1 - 0.6), (0.3 - 0.8), and (0.6 - 1.5). In the original code associated with the Martell and Froese (2013) paper they only used three resilience categories, omitting the "medium". The initial range of the unfished biomass is also very broad with a minimum set at the maximum catch and a maximum set at 60 times the maximum catch.

In case these particular combinations do not suit what is known or suspected of a particular species, the *run_cMSY* function now includes the option to set your own limits on both the initial values for *r* and *K*. A call to either *formals(run_CMSY)* or to *?run_cMSY* will illustrate the syntax and the names of *start_r* and *start_K*. In each case they require a short vector of two numbers (e.g. *start_r=c(0.015,0.3)*)

Results from the Catch-MSY Analysis

The plots above illustrate the results and provide some information but it is also helpful to obtain a tabulation of the outcome of the analysis. For that we use the *pulloutStats* function. It generates another object containing a matrix of r, K, MSY, and current depletion percentiles and mean values. In addition, it contains the successful biomass trajectories and those same trajectories translated into depletion levels.

```
results <- pulloutStats(answer$R1)
print(str(results,max.level=1))

## List of 3
##  $ output : num [1:4, 1:8] 1.38e-01 4.19e+03 2.61e+02 6.95e-02
2.42e-01 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ traject: num [1:2208, 1:35] 1479 4303 4816 2069 2498 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ deplet : num [1:2208, 1:35] 0.268 0.622 0.697 0.41 0.319 ...
##   ..- attr(*, "dimnames")=List of 2
## NULL
```

Table 2. The results$output statistics from the Catch-MSY analysis. The 2.5Perc and 97.5Perc are the respective percentile describing the spread of the trajectories (i.e. not confidence intervals around the mean). The % columns are the quantiles so that the 50% columns represents the medians.

	2.5%Perc	Mean	97.5%Perc	2.5%	5%	50%	95%	97.5%
r	0.14	0.24	0.43	0.14	0.15	0.24	0.39	0.41
K	4188.76	6196.66	9167.06	3911.06	4220.37	6479.79	7940.26	8016.45
MSY	260.69	374.82	538.92	264.36	277.55	373.90	518.97	554.18
CurrDepl	0.07	0.32	0.57	0.07	0.09	0.34	0.49	0.49

These results are fine as far as they go but in order to obtain some notion of stock status it is necessary to trace the successful trajectories in terms of how their depletion has changed through time. For this we can use the same *plottrajectory* function as used previously only this time changing the *oneplot* parameter to TRUE, which overrides the plotall parameter. The Bmax parameter can be adjusted to obtain an acceptable spread of the results.

```
out <- plottrajectory(answer$R1,fish$year,fish$catch,answer$parbound,
                      oneplot=TRUE,scalar=1.0,plotout=TRUE,plotall=7)
```

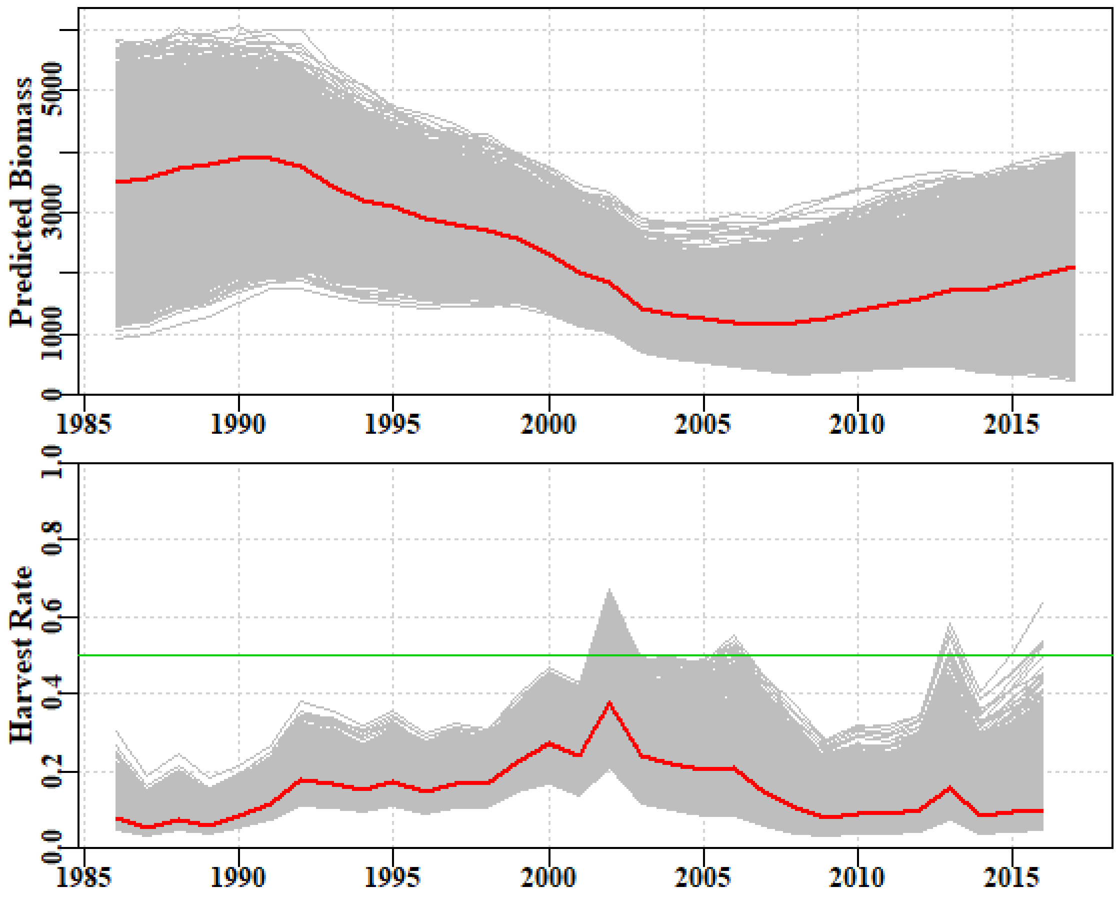


Figure 17. The top plot is of the successful biomass trajectories and the red line is the mean in each year. The bottom plot is of the annual harvest rate; note a few seemingly successful trajectories lead to harvest rates > 0.5. Associated with these high harvest rates note also that the final biomass for many trajectories is very low, which may also be deemed unrealistic.

Given each biomass trajectory has an associated *K* value it is possible to translate the biomass trajectories into depletion trajectories. There are two ways to do this, the simplest way is to use the *pulloutStats* function (see example above) which outputs both the biomass and depletion matrices as a byproduct to estimating the mean MSY and current depletion.

```
results <- pulloutStats(answer$R1)
# Note the use of constC=0, we are not doing any projections yet so no
constant catches
effectC <-
plotconstC(results$deplet,endyear=2017,constC=0,limit=0.2,target=0.4,
                      console=TRUE,intensity=NA,contours=TRUE)

##       Year    PltLim%    PgtTarg%      Mean    Median Pincrease
## 2013 2013 0.2595109 0.05525362 0.2682119 0.2715415 0.1942935
## 2014 2014 0.2758152 0.07835145 0.2682592 0.2754918 0.1376812
## 2015 2015 0.2468297 0.16666667 0.2873085 0.3004043 0.1621377
## 2016 2016 0.2318841 0.26811594 0.3039083 0.3187994 0.1884058
## 2017 2017 0.2182971 0.35009058 0.3201295 0.3389702 0.0000000

abline(h=c(0.2,0.3,0.4),col=3,lwd=2)
```

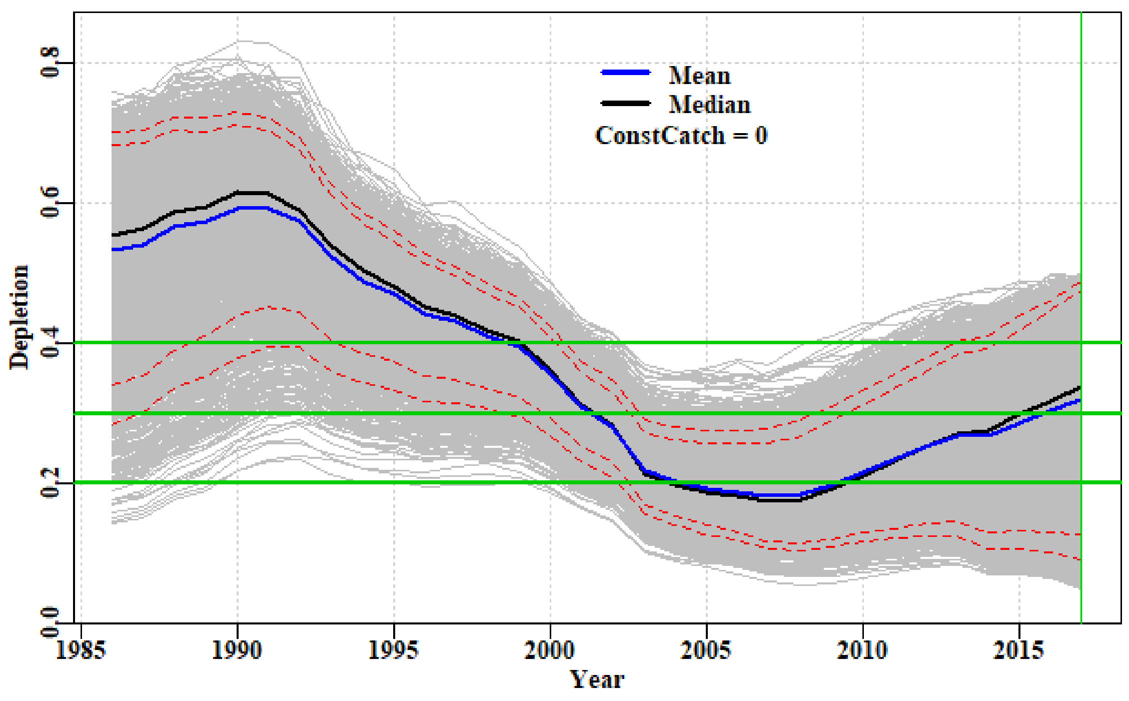


Figure 18. A plot of the successful depletion trajectories with the mean and median annual depletion marked. The lower red line is the default 0.2B0 limit reference point, while the upper is the input target reference point. The green line denotes the end of the final year in which data are available. If the abbreviated table under the plot is not wanted (all the information will be in *effectC*) then the *console=TRUE* should be set to *console=FALSE*. The red dashed lines are the inner 80th and 90th percentile bounds but if not wanted set *contours=FALSE*.

The PltLim% column in the console output is the proportion of trajectories that are < *limit*, whereas the PgtTarg% is the proportion of trajectories that are greater than the target. Given the great uncertainty in these analyses it should not be surprising that both of these increase, which suggests the total spread of the outcomes is increasing. The question that needs answering is whether the average is increasing or decreasing here both the mean and median are increasing though only about 4 - 5 % across the five years prior to the final year of data. As the average stock size increases the rate of increase would be expected to first increase and then decline as the stock moved on average above the biomass that produces the maximum productivity (B_{MSY}).

The default plot of the depletion in **Figure 18** has grey lines of equal density but if one wants to attempt to generate a plot with the density of colour matched to the density of trajectories then it is possible to include a number in the *intensity* parameter. The value input determines the density of the trajectories required to obtain full colour intensity and this will undoubtedly vary by fishery and so will need to be used interactively to find a value that generates a satisfactory plot. If the plot is to be printed then be sure to save the plot as a bit map or .png file so that the varying density and transparency is retained in the plot.

```
effectC <-
plotconstC(results$deplet,endyear=2017,constC=0,limit=0.2,target=0.4,
                      console=FALSE,intensity=30)
abline(h=c(0.2,0.3,0.4),col=3,lwd=2)
```

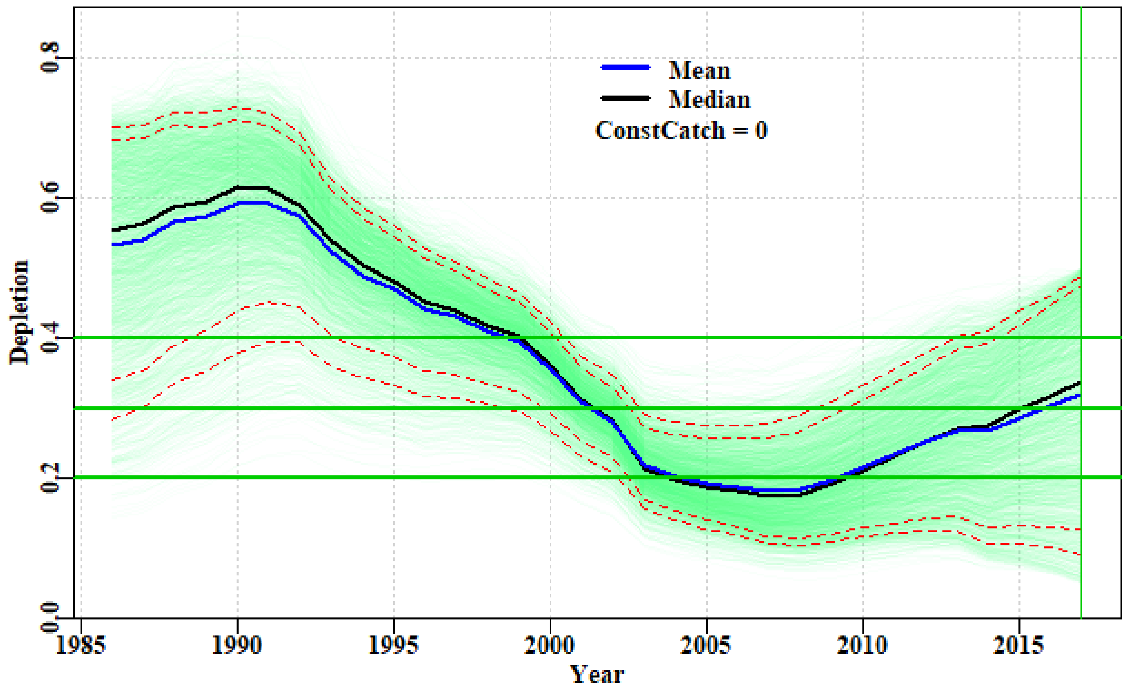


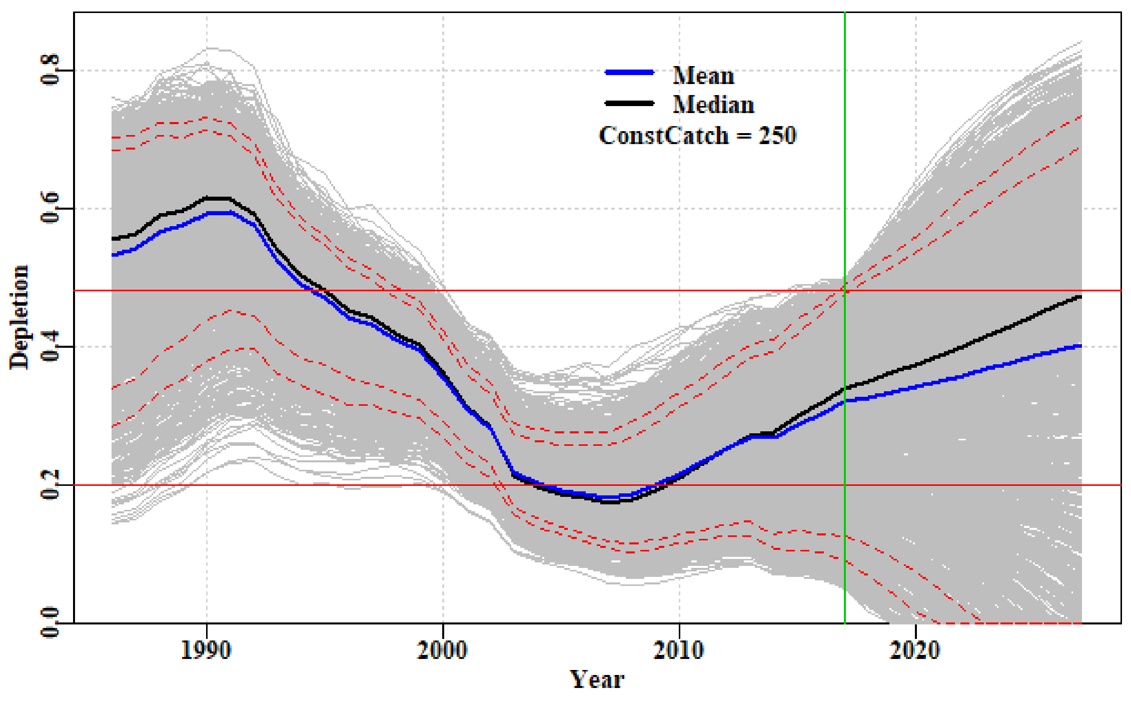
Figure 19. A plot of the successful depletion trajectories with the mean and median annual depletion marked with the density of trajectories represented by different intensity of colour. The lower red line is the default 0.2B0 limit reference point, while the upper is the input target reference point. The green line denotes the end of the final year in which data are available. The intensity value may need to be adjusted empirically to obtain a satisfactory plot.

In **Figure 19** different intensity of colours are used to denote the paths followed by most trajectories. In this case between 1995 - about 2001 most lines are towards the top of the overall path but are moving downwards until between 2002 - 2008 the median line is biased low with it being closer to the bottom of all paths than the top. After that most paths are moving upwards with the median once again shifting closer to the upper margin than the lower.

Constant Catch Projections

From the table and the plot it is clear that the average depletion is quite a way below the selected target. We need to remember that this is a data-poor assessment and that currently there are no agreed harvest strategies or harvest control rules. Nevertheless, if we project the currently potentially successful trajectories forward under different constant catch scenarios we will be able to determine what levels of catch will lead to the stock increasing (on average) and what will lead to decreases (on average). Depending on where the assessment indicates a species is laying will then determine what management action to advise to manage a stock in a desired direction (whether that be to increase or decrease its current state).

```
output <-
doconstC(answer$R1,projn=10,constCatch=250,lastyear=2017,limit=0.2,
                         target=0.48,console=FALSE)
```



```
##      Year   PltLim%   PgtTarg%      Mean    Median Pincrease
## 2017 2017 0.2182971 0.08333333 0.3201295 0.3389702 0.0000000
## 2018 2018 0.2264493 0.15670290 0.3272867 0.3501942 0.7047101
## 2019 2019 0.2336957 0.22780797 0.3344058 0.3627125 0.7047101
## 2020 2020 0.2382246 0.28260870 0.3420144 0.3745919 0.7047101
## 2021 2021 0.2409420 0.32201087 0.3502623 0.3864780 0.7047101
## 2022 2022 0.2418478 0.35552536 0.3589386 0.4010272 0.7047101
## 2023 2023 0.2454710 0.38768116 0.3678768 0.4158672 0.7047101
## 2024 2024 0.2477355 0.42527174 0.3769153 0.4300547 0.7047101
## 2025 2025 0.2527174 0.45018116 0.3859903 0.4440718 0.7047101
## 2026 2026 0.2558877 0.47554348 0.3948929 0.4597548 0.7047101
## 2027 2027 0.2567935 0.49592391 0.4036456 0.4741102 0.7047101
```

Figure 20. A plot of the successful depletion trajectories followed by five years of projection. The limit and target reference points are depicted by the two fine red lines. The green line denotes the end of the final year in which data are available, with projections to the right. The mean depletion varies away from the median because an array of trajectories go extinct on projection

The projections suggest that a total catch of about 250 t would be expected to lead, on average, a small increase in stock levels over the five years. By exploring the outcomes with smaller and larger constant catches the implications can be made clear and appropriate decisions about catch levels or triggers could then be made in an attempt to meet whatever objective is desired for the fishery. What seems to be implied by **Figure 20** is that the stock in question was overfished between about 2003 - 2010 but as the plot of catches vs MSY from the plotcMSY6 analysis illustrate, after 2003 catches were reduced significantly and that led to the stock rebuilding until in 2017 it was estimated (with great uncertainty) to be about 30%B0. In 2017, given the stock is rebuilding and is already above the limit reference point it can be claimed that it is sustainable and that over-fishing is not occurring.

Robustness of MSY Estimate

While it is true that the ranges of the successful *r* and *K* combinations is quite wide, because there is a strong negative correlation between these two parameters the spread of the MSY estimates is not so large in relative terms. This can be illustrated by using the *trendMSY* function which estimates the MSY implied by a series of slices down the K axis from the scatter of r - K combinations. Given that MSY is estimated from the r - K combinations it is possible to map the estimated MSY onto the plot of r - K combinations, along with the estimated trend from the successful r - K combinations (*Figure 6*).

```
r <- summcMSY$r
K <- summcMSY$K
meanmsy <- trendMSY(summcMSY$r,summcMSY$K,inc=300)
centr <- central(r)
centK <- central(K)
means <- central(summcMSY$msy)
avMSY <- means["Geometric","Mean"]
# plotprep(width=7,height=4.0)
par(mfrow=c(1,1),mai=c(0.45,0.45,0.05,0.05),oma=c(0.0,0,0.0,0.0))
par(cex=0.85, mgp=c(1.35,0.35,0), font.axis=7,font=7,font.lab=7)
plot(r,K,type="p",pch=16,col=rgb(1,0,0,1/3),panel.first=grid())
rval <- seq(0.1,0.6,length=100)
kval <- (4 * avMSY)/rval  # calculate the K value that would generate
avMSY
lines(rval,kval,col=4,lwd=2)
pickvalid <- which(meanmsy[,"N"] > 0)
lines(meanmsy[pickvalid,"rcenter"],meanmsy[pickvalid,"Kcenter"],lwd=2,
col=3)
points(centr[2,1],centK[2,1],pch=16,cex=2,col=1)
```

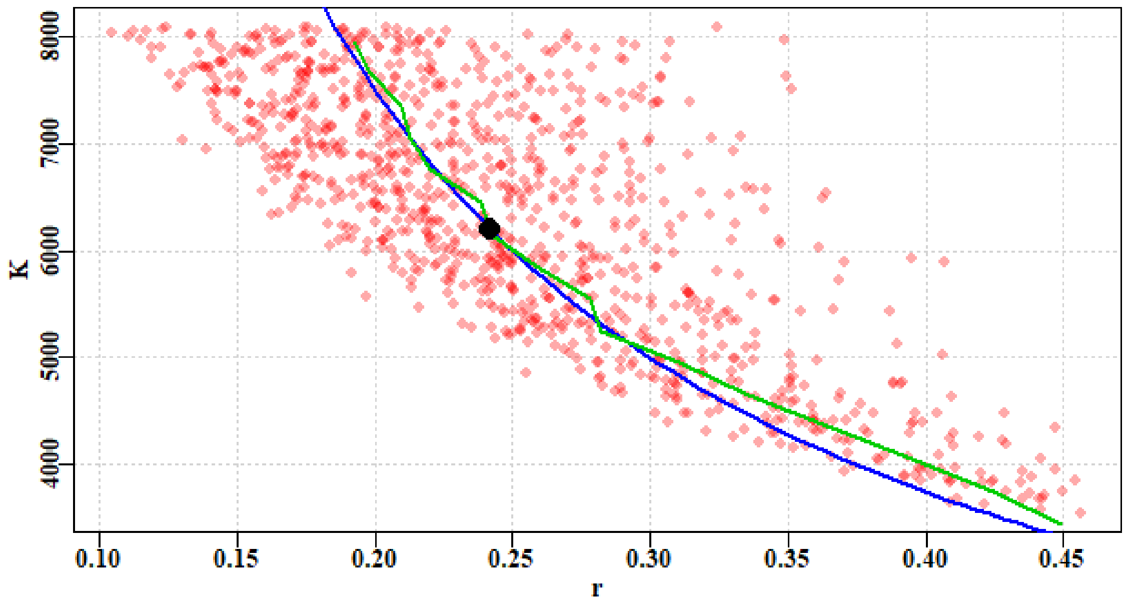


Figure 21. A plot of the successful r - K combinations with the estimated mean MSY over-plotted in blue and the mean MSY for horizontal slices of the cloud of successful r - K combinations derived from *trendMSY*. Note that the overall mean MSY (the black dot) coincides approximately with the point at which the blue and the green lines begin to diverge.

One can imagine contours of MSY running across the r - K parameter space and the catch-MSY analysis is discovering which r - K combinations are consistent with the possibilities (*Figure 7*).

```
r <- summcMSY$r
K <- summcMSY$K
avMSY <- seq(100,600,100)
nMSY <- length(avMSY)
# plotprep(width=7,height=4.0)
par(mfrow=c(1,1),mai=c(0.45,0.45,0.05,0.05),oma=c(0.0,0,0.0,0.0))
par(cex=0.85, mgp=c(1.35,0.35,0), font.axis=7,font=7,font.lab=7)
plot(r,K,type="p",pch=16,col=rgb(1,0,0,1/3),panel.first=grid(),ylim=c(
2500,9000))
rval <- seq(0.1,0.6,length=100)
for (i in 1:nMSY) {
   kval <- (4 * avMSY[i])/rval  # calculate the K value that would
generate avMSY
   lines(rval,kval,col=4,lwd=2)
}
x <- seq(0.15,0.375,length=nMSY)
yval <- 4 * (avMSY/x)
for (i in 1:nMSY) text(x[i],yval[i],avMSY[i],cex=1,font=7,pos=4)
```

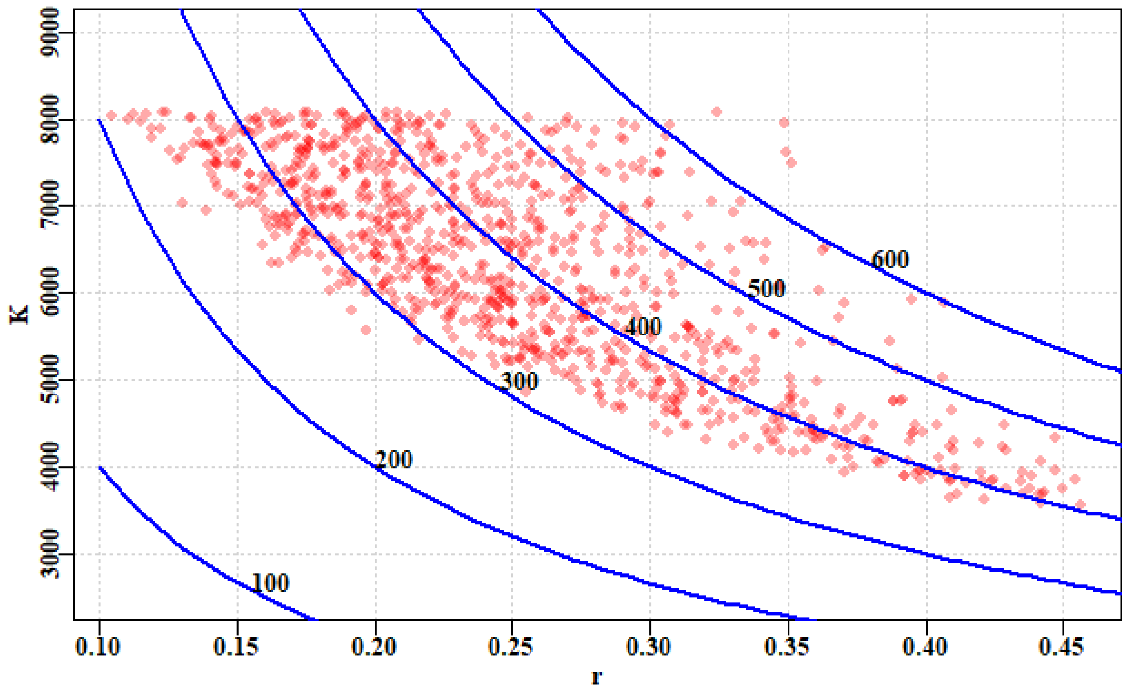


Figure 22. A plot of the successful r - K combinations with an array of implied MSY values as implied contours across the r - K parameter space. Some contours would under-estimate the implied productivity while others would over-estimate what was consistent with available observations.

Significant Unknown Recreational Catches

It is often the case that obtaining accurate estimates of recreational catch is rarely done even when it is known that they can rival commercial catches. In States where there are significant recreational catches of commercial species there remains a need to assess the stock status of such species and likely also provide management advice as to what would constitute sustainable commercial catches. As long as there is some notion of the proportion of recreational catches it is still possible to apply the catch-MSY assessment method to a time-series of commercial catches. To demonstrate the effect of having significant amounts of recreational catch on top of a commercial catch we can conduct an analysis using one of the data sets internal to **simpleSA**, then that can be compared to the same analysis after the catches have been proportionally reduced in some manner as if the total catch was made up of known commercial catches and unknown recreational catches. In this preliminary exploration we will examine a deterministic reduction keeping just 80%, and 60% of known commercial catches while repeating the analysis. To include some variation we can reduce the original catches by a randomly varying proportion around some selected mean, finally we can conduct the analysis on a deterministically reducing commercial proportion down from say 80% to 50% over the time period of data available.

We can use the *invert* data set. First we obtain the expected outcome by analysing the full data set.

```
library(simpleSA)
data(invert)
fish <- invert$fish
glb <- invert$glb

reps <- 5000  # one would run at least 20000, preferably more
answer <- run_cMSY(fish,glb,n=reps,sigpR=0.025)
summcMSY <- summarycMSY(answer,fish,final=TRUE)
ans <- pulloutStats(answer$R1)
round(ans$output[,1:3],3)

##           2.5%Perc     Mean 97.5%Perc
## r            0.127    0.231     0.421
## K         4309.742 6480.576  9744.866
## MSY        253.399  374.364   553.074
## CurrDepl     0.059    0.312     0.566
```

So the mean current depletion is estimated to be about 0.31, with an MSY of about 370 tonnes. Now we can answer what would happen if the catch data was only 80% of the original (with variation about that 80% each year; this implies there would be unknown recreational catches of about 20% plus or minus some variable amount each year) while the cpue data remains unchanged. The deterministic reductions gave very precise results so this exposition begins with the somewhat more realistic variation around a given average proportion of recreational catches

```
commprop <- 0.8
propcom <- rnorm(31,mean=commprop,sd=0.025) # 31 equals the number of
years
fishC <- fish
fishC[,"catch"] <- fishC[,"catch"]*propcom
answerC <- run_cMSY(fishC,glb,n=5000,sigpR=0.025)
ansC <- pulloutStats(answerC$R1)
#str(ans10)
round(ansC$output[,1:3],3)

##           2.5%Perc     Mean 97.5%Perc
## r            0.119    0.222     0.412
## K         3540.525 5416.386  8286.128
## MSY        196.170  299.956   458.653
## CurrDepl     0.056    0.308     0.561
```

```
round(ans$output[,1:3],3)
```

```
##          2.5%Perc     Mean 97.5%Perc
## r           0.127    0.231     0.421
## K        4309.742 6480.576  9744.866
## MSY       253.399  374.364   553.074
## CurrDepl    0.059    0.312     0.566
```

```
msy <- ansC$output["MSY","Mean"]/ans$output["MSY","Mean"]
depl <- ansC$output["CurrDepl","Mean"]/ans$output["CurrDepl","Mean"]
cat("reduced catch MSY/full catch MSY = ", msy, mean(propcom),"\n")
```

```
## reduced catch MSY/full catch MSY =  0.8012429 0.8057442
```

```
cat("Proportion of Current Depetion = ",depl,"\n")
```

```
## Proportion of Current Depetion =  0.9873974
```

You could try different values for the *commprop* value to see the effect of the recreational proportion increasing. But it appears that the impact of a 20% decrease in known catches is to reduce the sustainable productionavailable to commercial operators by about 20% (and similarly a 40% reduction leads to about a 40% reduction in production).

If instead of the mean recreational proportion staying approximately stable through time it is also possible to examine the effect of a trend in the proportion of commercial catches. Here we inspect the effect of the commercial proportion changing from 80% to 50% (unknown recreational catches change from 20% to 50%) over a 31 year period. Normally one would use at least 20000 replicates rather than 5000.

```
propcom <- seq(0.8,0.5,length=31) # 31 equals the number of years
fishC <- fish
fishC[,"catch"] <- fishC[,"catch"]*propcom
answerC <- run_cMSY(fishC,glb,n=5000,sigpR=0.025)
ansC <- pulloutStats(answerC$R1)
round(ansC$output[,1:3],3)
```

```
##          2.5%Perc     Mean 97.5%Perc
## r           0.111    0.205     0.376
## K        3321.514 4953.451  7387.198
## MSY       165.480  253.686   388.910
## CurrDepl    0.063    0.313     0.563
```

```
round(ans$output[,1:3],3)
```

```
##          2.5%Perc     Mean 97.5%Perc
## r           0.127    0.231     0.421
## K        4309.742 6480.576  9744.866
## MSY       253.399  374.364   553.074
## CurrDepl    0.059    0.312     0.566
```

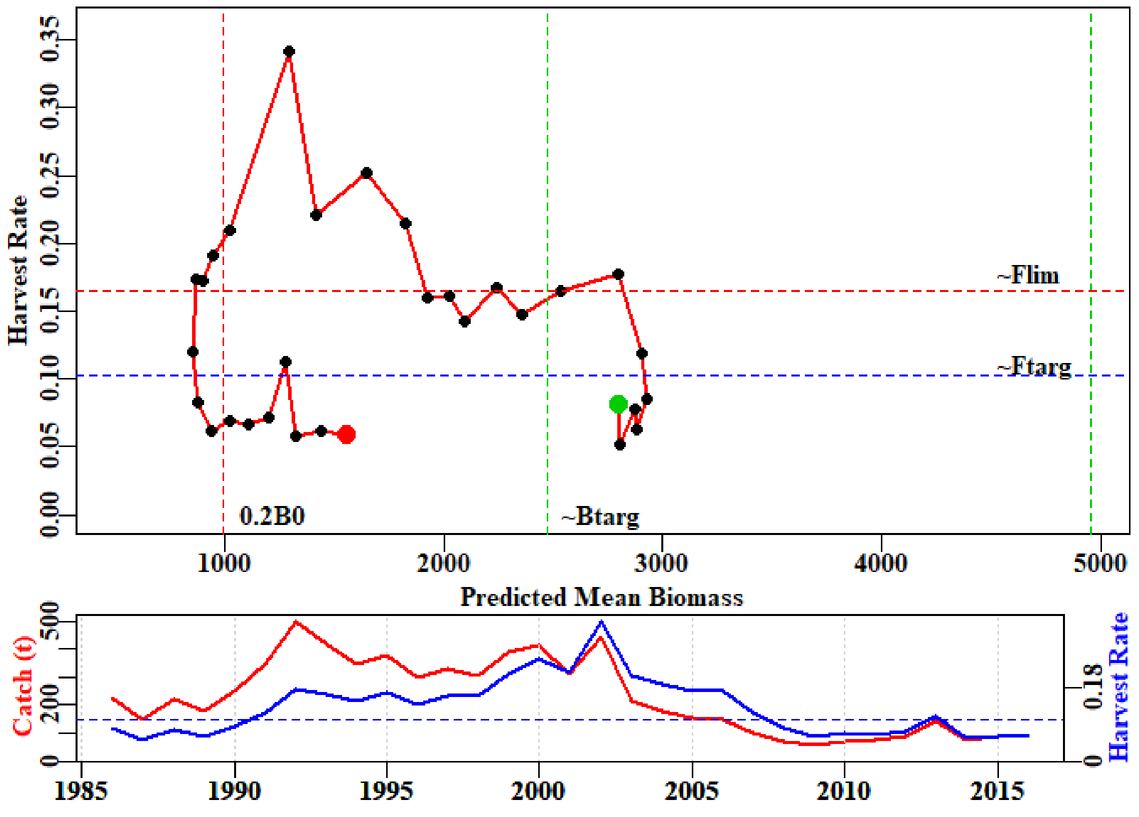
```
msy <- ansC$output["MSY","Mean"]/ans$output["MSY","Mean"]
depl <- ansC$output["CurrDepl","Mean"]/ans$output["CurrDepl","Mean"]
cat("reduced catch MSY/full catch MSY = ", msy, mean(propcom),"\n")
```

```
## reduced catch MSY/full catch MSY =  0.6776468 0.65

cat("Proportion of Current Depletion = ",depl,"\n")

## Proportion of Current Depletion =  1.003029
```

If this code chunk is run numerous times the outcomes suggest that the estimate of sustainable commercial catch may be biased slightly high but, on the other hand the depletion level is invariably very close to that obtained when using all the catch data.

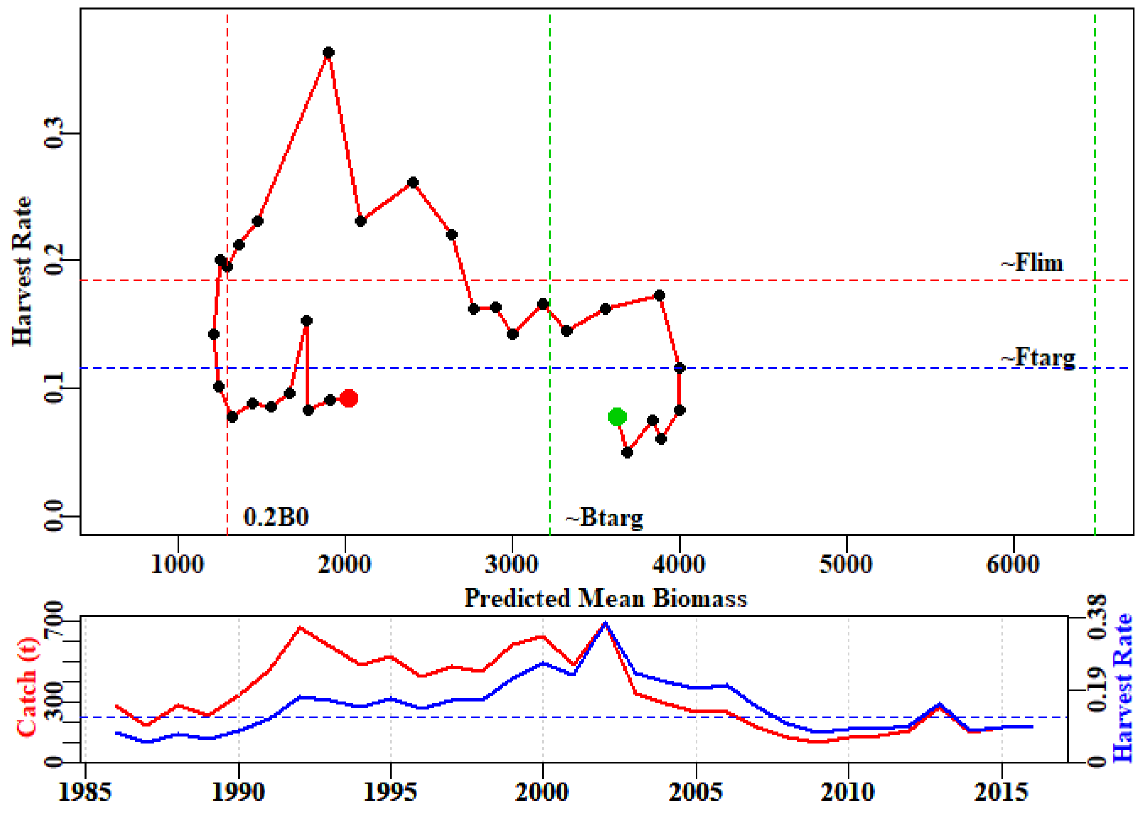
These two sets of scenarios suggest that if we consider an unknown recreational catch to reflect one of these two scenarios (randomly varying around some relatively constant proportion of the commercial catch or approximateely following some trajectory from one proportion to a higher proportion of recreational catch) then it should still be possible to get an estimate of the stock's current depletion level while, at the same time generating potential management advice concerning the commercial fishery. These explorations need to be more fully investigated but, at least with the Catch-MSY approach, the conclusions appear to be acceptable. It can even provde a phase plot of predicted biomass against harvest rate, but obviously only for the commercial component. These results are consistent with those of Rudd and Branch (2017) who find very similar results using fitted stock assessments in a management strategy evaluation framework rather than catch-only methods.

```
cMSYphaseplot(answerC,fishC)
```



which can then be compared with the analysis that derived from the full dataset of catches:

```
cMSYphaseplot(answer,fish)
```



There are clear differences in that the analysis of partial catch suggests that there was a time that the stock went below the limit reference point but overall, the trajectory obtained is approximately the same as that for the full catch data set.

It is not surprising that differences occur between those analyses that capture all catches and those that include a proportion of unknown catches because the productivity now appears to be different. Nevertheless, the same general pattern observed with all data remains, even with the truncated data. In reality, with a fishery having a significant recreational proportion, we would only have the 'truncated' data but these explorations indicate that it should still be possible to obtain both a status and management advice (at least for the commercial proportion of the fishery) under the assumption that the relative proportion would not change further. Of course, if there is a trend in the proportion of recreational take relative to the commercial take, then only advice in the short term could be deemed useful.

Plausible Levels of Harvest Rate

Even though the overall productivity of the diverse Australian fisheries may be relatively high, the productivity of many individual Australian fisheries tends not to be as great as equivalent fsheries in the northern hemisphere. Many commercial Australian fish species seem to live longer than northern hemisphere equivalents and live less densely on the sea bed, and hence naturally cannot be as productive. It seems unlikely, therefore, that the fishing mortalities observed in places such as the North Sea, where *F* values of 1.0 (equivalent to harvest rates of between 0.6 and 0.7 each year) have been common, are not plausible and certainly not sustainable here. The catch-MSY method only uses the available time-series of catches and its random selection of *r*-*K* combinations can select highly productive combinations which enable the stock to grow well beyond plaiuble bounds (rejected), there are other combinations whose implied productivity (from the Schaefer model used within the catch-MSY to reflect the stock dynamics) is far too low so that any trajectories go extinct (rejected). Then there are the combinations that imply a productivity that allows for the catches taken but stays within the other constraints implied by the resilience and starting catch levels for the species (accepted trajectories). However, within those accepted trajectories there will be some having implied productivities that only barely survive within the constraints. It is also implied that some of these trajectories reflect relatively high annual harvest rates. For example, using the *invert* data set we can obtain some trajectories with harvest rates greater than 0.5 (50% of exploitable biomass taken every year).

```
# library(simpleSA)
data(invert)
fish <- invert$fish
glb <- invert$glb
answer <-
run_cMSY(fish,glb,n=5000,sigpR=0.025,finaldepl=c(0.05,0.5),maximumH=1.
0)
ans <- pulloutStats(answer$R1)
round(ans$output[,1:3],3)

##            2.5%Perc     Mean 97.5%Perc
## r             0.116    0.216     0.403
## K          4477.619 6897.759 10625.978
## MSY         243.057  372.339   570.386
## CurrDepl      0.056    0.309     0.561

out <- plottrajectory(answer$R1,fish$year,fish$catch,answer$parbound,
                      oneplot=TRUE,scalar=1.0,plotout=TRUE,plotall=7)
```

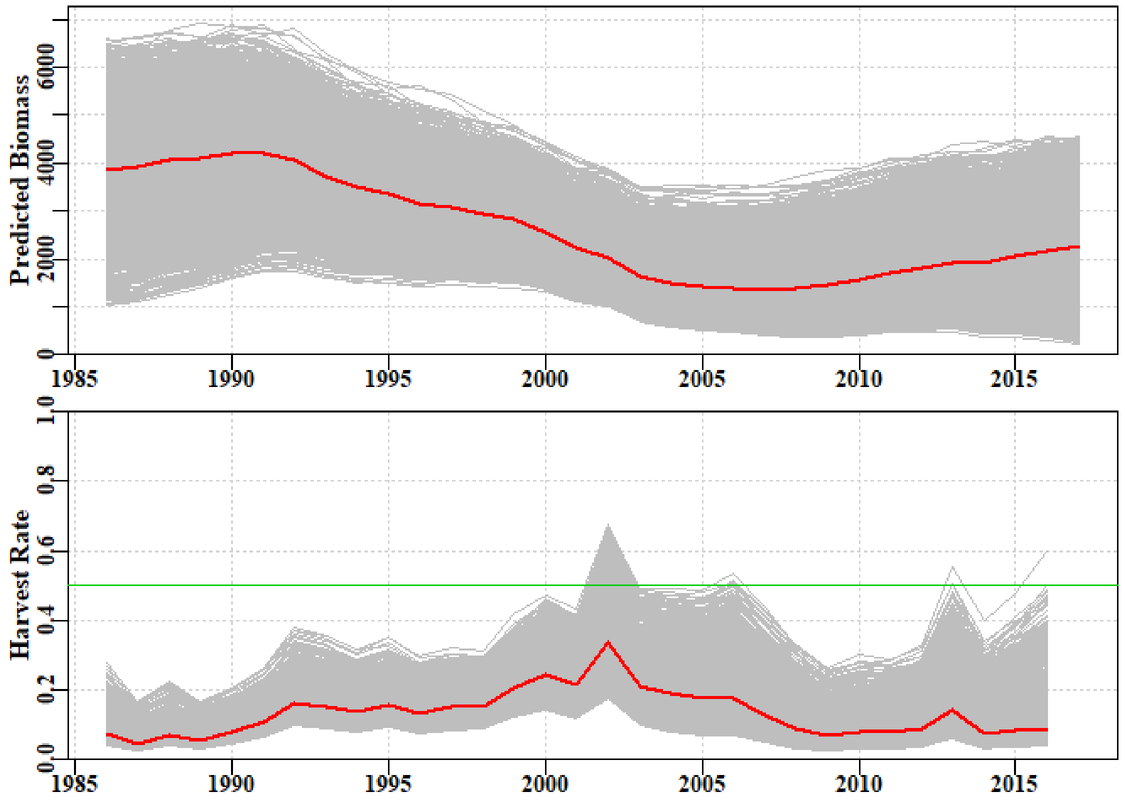


Figure 23. A plot of the implied successful biomass trajectories and their implied annual harvest rates generated by the function *plottrajectory* with the option *oneplot* set to *TRUE*. The red lines are the median values, the green line is at a harvest rate of 0.5, for reference.

In **Figure 23.** some of the harvest rates in the final years approach 0.5 but between 2000 and about 2007 some trajectories fully breach the 0.5 line and often stay close to it. Part of this reflects the default final depletion range of c(0.05,0.5) given if catches in the final year are less than half the maximum catch (a selection which ignores the fact that management intervention or marketing issues may have controlled catches rather than an inability to catch). However, even if we change the lower final depletion valie to, for example, 0.15, while that reduces most of the final year high harvest rates that survive, the period from 2000 still appears to be exceptional.

In order to conduct a sensitivity on the implications of there being an upper limit of harvest rate a new *maximumH* parameter has been added to the *run_cMSY* function. Its default value is set to 1.0, so that all harvest rates are possible (obviously one could not take more than 100% of what is present!). If we alter tha value to 0.5 the outcome changes.

With this extra constraint the successful biomass trajectories are much more restricted and the harvest rates far lower. This change only improves the depletion level by about 3%, but the productivity, as measured by the MSY is reduced from about 370 t down to 333 t, a reduction of 10%. It may appear counterintuitive that removing some of the lower trajectories (note the increase in the lower 95th percentile) led to a lower productivity, but the important parts are reflected in the mean values of *r*, the population growth rate. By comparing the outputs related to **Figures 23 and 24** the reduction in *r* should be clear. It is this reduction that has led to the decrease in productivity.

Such a sensitivity on this parameter (*maximumH*) is very dependent upon local knowledge of the history of any fishery. Productivity is also partly determined by the resilience attributed to a species. One could also run sensitivities on what resilience was given to a species. The importance of that is that the resilience determines the implied bounds on *r* used in the search for successful trajectories. Of course, it is now possible to modify these if it is felt necessary by directly entering values for *start_r* as a vector of two numbers.

```
answer <-
run_cMSY(fish,glb,n=5000,sigpR=0.025,finaldepl=c(0.05,0.5),maximumH=0.
5)
ans <- pulloutStats(answer$R1)
round(ans$output[,1:3],3)

##            2.5%Perc     Mean 97.5%Perc
## r             0.114    0.174     0.265
## K          6038.608 7737.059  9913.227
## MSY         234.142  336.988   485.009
## CurrDepl      0.085    0.325     0.564

out <- plottrajectory(answer$R1,fish$year,fish$catch,answer$parbound,
                      oneplot=TRUE,scalar=1.0,plotout=TRUE,plotall=7)
```

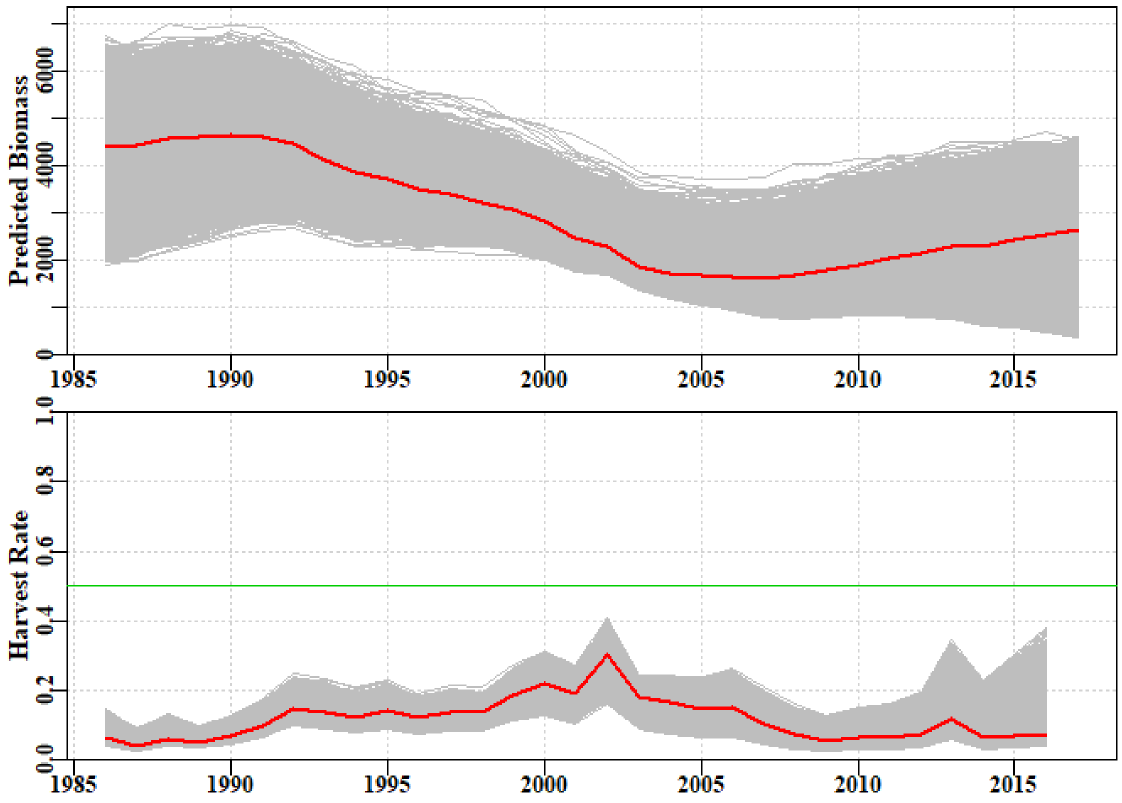


Figure 24. A plot of the implied successful biomass trajectories and their implied annual harvest rates generated by the function *plottrajectory* with the option *oneplot* set to *TRUE* and the *maximumH* parameter set to 0.5. The red lines are the median values, the green line is at a harvest rate of 0.5, for reference.

Other Sensitivities

We have seen it is possible to examine the implications of changing the *initdepl* and *finaldepl*, along with changing the *start_r* and *start_K*. A further sensitivity that ought to be conducted relates to the process error term *sigpR*. This term aims to attempt to capture un-accounted for variation in the stock dynamics between the years. The somple model is so simple it failts to capture many sources of natural variation. It is worthwhile examining alternative values for *sigpR*. In particular, it is worthwhile effectively turning this source of variation off to see the effect of almost deterministic dynamics. This can be achieved by setting *sigpR = 0*.

```
# library(simpleSA)
# normally one would run 20000+ replicates, but for speed we use 5000
answer <-
run_cMSY(fish,glb,n=5000,sigpR=0,finaldepl=c(0.05,0.5),maximumH=1.0)
ans <- pulloutStats(answer$R1)
round(ans$output[,1:3],3)

##            2.5%Perc      Mean  97.5%Perc
## r             0.127     0.231      0.421
## K          4279.238  6586.085  10136.503
## MSY         261.929   380.815    553.661
## CurrDepl      0.054     0.307      0.560

out <- plotconstC(ans$deplet,endyear=2017,constC=0)
```

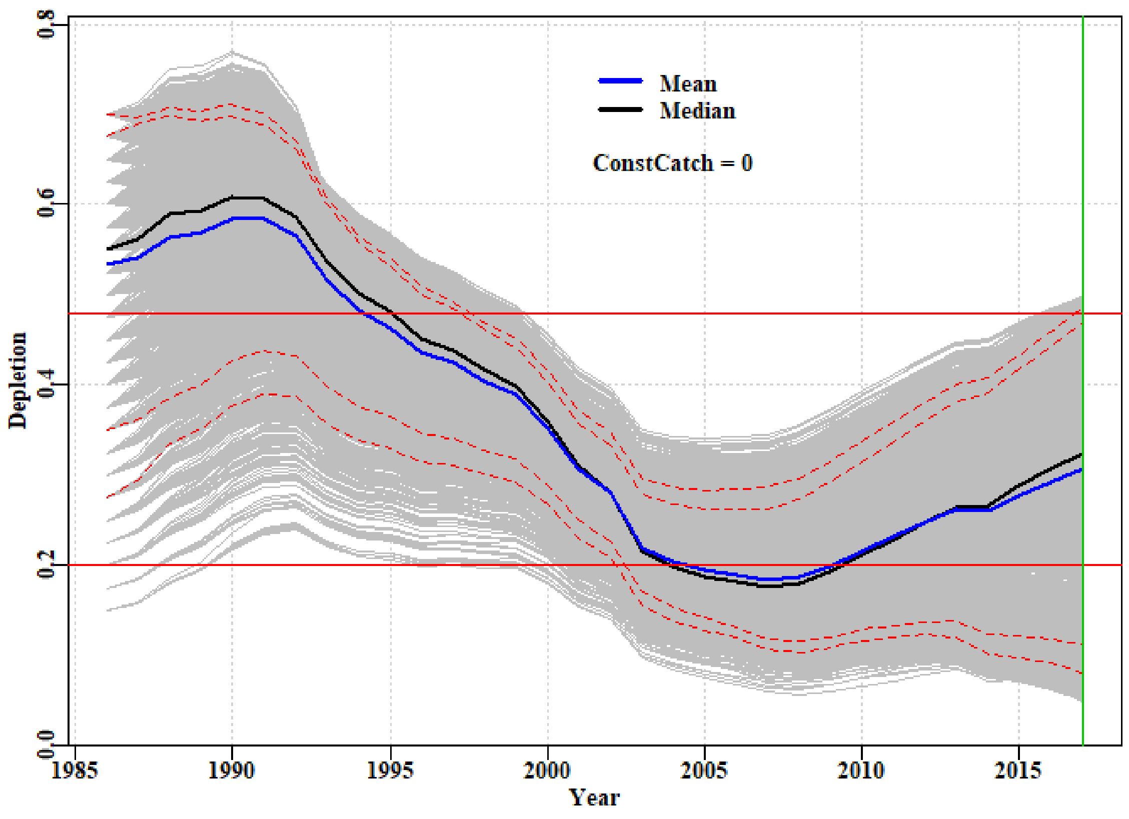


Figure 25. A plot of the implied successful depletion trajectories when the *sigpR* parameter is set to 0. The 23 different starting depletion levels between the default 0.15 - 0.7 can now be clearly seen. The lowest values are the most sparse in terms of successful trajectories.

```
##      Year   PltLim%   PgtTarg%      Mean    Median Pincrease
## 2013 2013 0.2857741 0.00000000 0.2619774 0.2644233 0.2230126
## 2014 2014 0.3087866 0.00000000 0.2610978 0.2668415 0.1631799
## 2015 2015 0.2744770 0.00000000 0.2780376 0.2879090 0.1824268
## 2016 2016 0.2560669 0.00292887 0.2927127 0.3070091 0.1945607
## 2017 2017 0.2476987 0.06820084 0.3065656 0.3233666 0.0000000
```

The assumption that a final depletion level of 5% (0.05) is plausible should be questioned for many Australian fisheries when the catches remain about 1/3 of the maimum and the CPUE barely changes. As always, such data-poor stock assessments should only form part of a weight-of-evidence argument supporting a claim for a set of management advice and stock status determinations.

```
plotfishery(fish,glb)
```

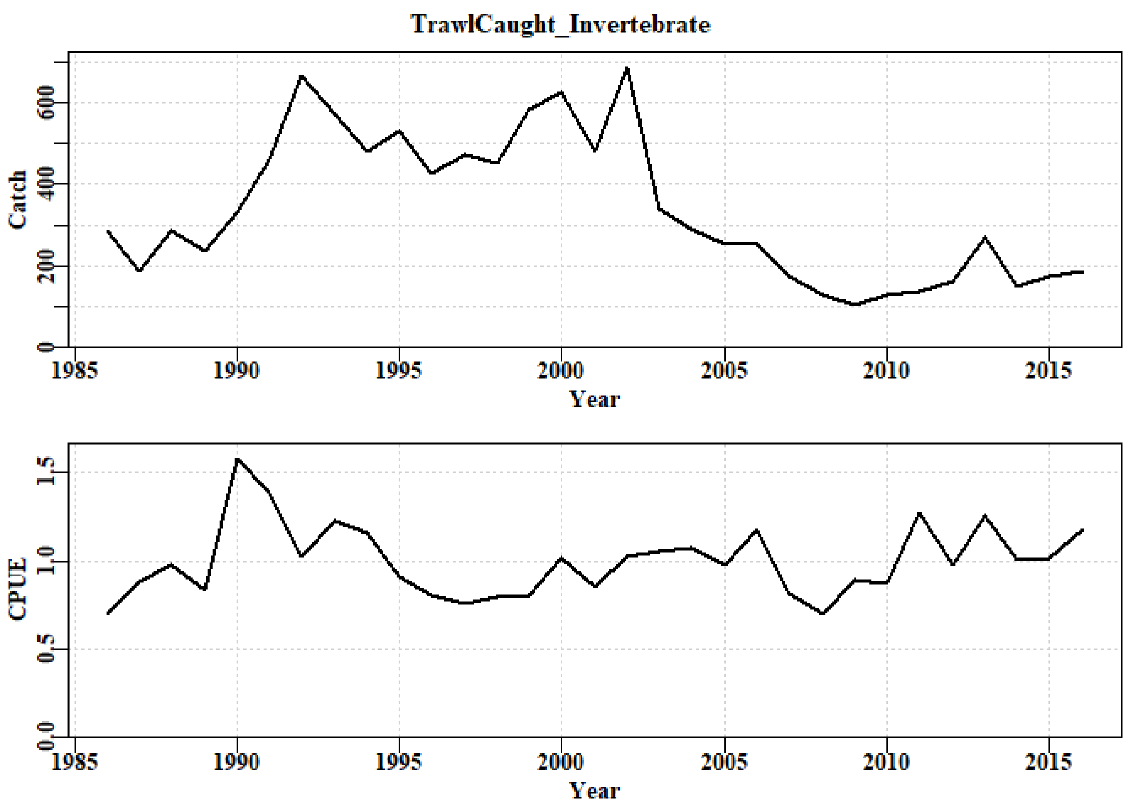


Figure 26. A plot of the reported catch history and the standardized CPUE for the example data set used in the vignette.

spm - Surplus Production Models

Introduction

Which Assessment to Apply

Which stock assessment method to apply to fisheries for data-poor to data-moderate species will depend upon what fisheries and biological data are available but also, importantly, on what management objectives need to be met within the juridiction in question. It may be the case that the fishery for a particular species is of sufficient size and value to warrant on-going monitoring and management towards some defined goal for the stock. In such a case the assessment used should obviously be capable of generating some notion of the current state of the fishery and indicate what management actions may be required to eventually achieve the agreed management goals. But some fisheries may be so minor that trying to actively manage them would be inefficient. Nevertheless, to meet the requirements of the Status of key Australian Fish Stocks (SAFS) one still requires some form of defensible stock assessment capable of determining whether the current level of fishing is sustainable.

Surplus Production Modelling = spm

Surplus production models are one of the simplest analytical methods available that provides for a full fish stock assessment of the population dynamics of the stock being examined. First described in the 1950s (Schaefer, 1954, 1957), modern versions with discrete dynamics are relatively simple to apply. This is partly because they pool the overall effects of recruitment, growth, and mortality (all the aspects of positive production) into a single production function. The stock is considered solely as undifferentiated biomass; that is, age and size structure, along with sexual and other differences, are ignored (one reason these models are also called biomass-dynamic models). Details of the equations used in the following analyses are provided in the Appendix of this vignette. You will also find details of the parameters and other aspects in the help files for each of the functions (try *?spm* or *?simpspm*, or even *simpfox*). In brief, the model parameters are r the net population rate of increase (combined individual growth in weight, recruitment, and natural mortality), K the population carrying capacity or unfished biomass (B_0), which is not to be confused with B_{init} (sometimes, confusingly, also called B_0), which is only required if the index of relative abundance data (usually cpue) only becomes available after the fishery has been running for a few years and after the stock has been depleted to some extent. B_{init} is set equal to K if no initial depletion is assumed. If early catches are known to be small relative to the maximum catches taken (< 10 - 25% of maximum) then it may well be true that initial depletion is only minor and might not be distinguishable from unfished.

The minimum data requirements needed to estimate parameters for such models are time-series of an index of relative abundance and of associated catch data. The catch data can extend either end of the index data if it is available. In Australia the index of relative stock abundance is most often catch-per-unit-effort (cpue), but could be some fishery-independent abundance index (e.g., from trawl surveys, acoustic surveys), or both could be used. The analysis will permit the production of on-going management advice as well as a determination of stock status.

In this section we will describe the details of how to conduct a surplus production analysis, how to extract the results from that analysis as well as plot out illustrations of those results. Example code will be developed that a user can take and modify to suit their own needs.

A standard workflow might consist of:

1. read in time-series of catch and relative abundance data. It would also be possible to use the function *checkdata* to ensure all data columns required are present; you could also consider the functions **makespmdat** and **checkspmdat** for similar purposes, check their help files.
2. use a **ccf** analysis to determine whether the cpue data are likely to be informative and thus provide plausible results from the spm (or not!).
3. define/guess a set of initial parameters containing r and K, and optionally B_{init} = initial biomass - used if it is suspected that the fishery data starts after the stock has been somewhat depleted. The mean values from a catch-MSY analysis might work.
4. use **DisplayModel** to plot up the implications of the assumed initial parameter set for the dynamics. This is useful when searching for plausible initial parameters sets.
5. use **optim** or **fitSPM** to search for the optimum parameters once a potentially viable initial parameter set are input. See discussion.
6. use **DisplayModel** using the optimum parameters to illustrate the implications of the optimum model and its relative fit (especially using the residual plot).
7. ideally one should examine the robustness of the model fit by using multiple different initial parameter sets as starting points for the model fitting procedure. Example code for doing that is given in the vignette.
8. once satisfied with the robustness of the model fit use **spmphaseplot** to plot out the phase diagram so as to determine and illustrate the stock status visually .
9. use **bootspm** to characterize uncertainty in the model fit and outputs.
10. Document and defend any conclusions reached.

Two versions of the dynamics are currently available: the classical Schaefer model (Schaefer, 1954) and the Fox model (Fox, 1970), both are described in Haddon (2011). Prager (1994) provides many additional forms of analysis that are possible using surplus production models and practical implementations are also provided in Haddon (2011). See the model equations in the appendix for more details of each model.

Application of the spm assessment

The minimum data requirements for a surplus production analysis is a time series of catches and a time series of an index of relative abundance (cpue). The years of catch information can be longer than the years of cpue data. We will use the *dataspm* data set built into the package to illustrate the use of the *spm* function. The *checkdata* function indicates it would be possible to apply the catch-MSY, teh spm, and the aspm analyses to this data.

```
# library(simpleSA)  # include the library before starting analysis
data(dataspm)
fish <- dataspm$fish
# check data.frame has all required columns; not usually required; use
head(fish,10)
checkdata(dataspm)
```

```
##                      Method Possible
## catch-MSY              TRUE     TRUE
## spm                    TRUE     TRUE
## aspm                   TRUE     TRUE
## catch-curves catch-curves    FALSE
```

Are the CPUE data informative?

The first step is to try to discover whether or not the available index data provide more information than the catches alone. With a developed fishery if the fishing, that is the catches, do indeed influence the stock dynamics, which would in turn affect the cpue, then we would expect that if catches increased the catch rates would eventually respond by declining and also if catches declined then the cpue would, in turn, eventually increase. Such a time-lagged negative correlation is something we can look for statistically if we have sufficiently long time-series of catches and associated cpue. To determine whether there are any significant negative correlations of cpue with catches we need to conduct a sequence of correlation analyses on the original data (timelag=0), then lag the cpue data backwards by one year and repeat the correlation analysis (with n-1 years this time, lag=-1), and repeat that for an array of negative time lags. Rather than do this manually it is possible to use a built in R function *ccf*. To simplify things further there is a wrapper function in simpleSA called *getlag*. This uses the *ccf* function to run the analysis, which can also generate a plot immediately or its output can be used to produce a more visually appealing plot for inclusion in a report (check out *str(ans)*).

```
ans <- getlag(fish,plotout=FALSE)
par(mfrow=c(1,1),mai=c(0.5,0.45,0.05,0.05),oma=c(0.0,0,0.0,0.0))
par(cex=1.0, mgp=c(1.35,0.35,0), font.axis=7,font=7,font.lab=7)
plot(ans,lwd=3,col=2,main="",ylab="Cross-Correlation")
minim <- which.min(ans$acf)
text(0,-0.6,paste0("Optimum Lag =
",ans$lag[minim]),cex=1.1,font=7,pos=4)
```

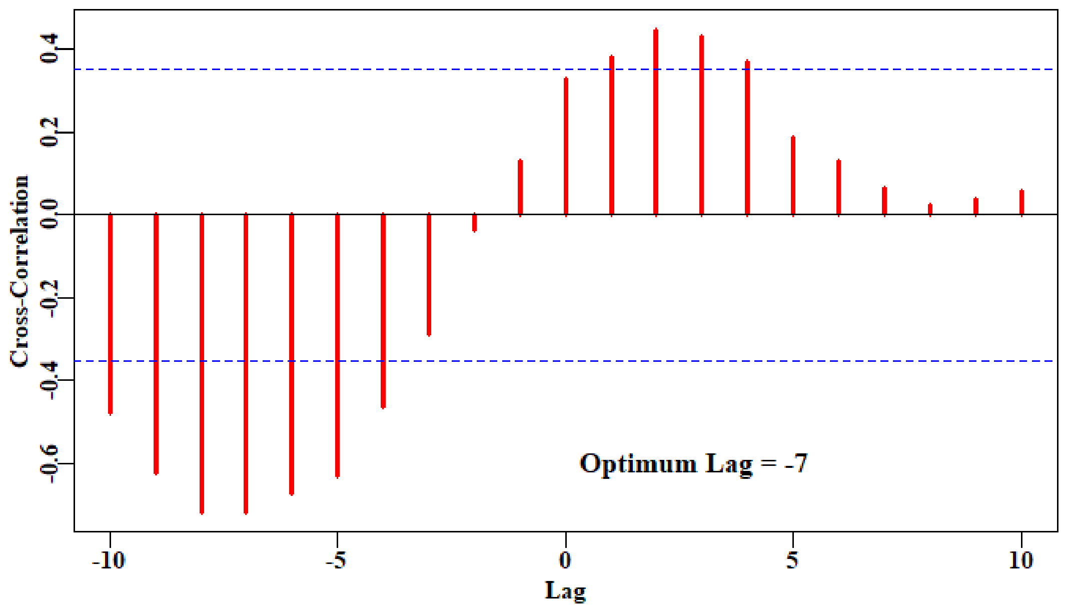


Figure 27. A plot of the *ccf* function output when applied to the fishery data within the built in *dataspm* data set. The strong negative correlation at lags of -7 and -8 suggest an spm or aspm analysis will likely generate plausible results, although the relatively strong correlation at a lag of 0, may weaken any valid realtionship between the catches and subsequent cpue values.

It should not be forgotten that with catches and cpue, we have catches on both sides of the correlation analysis so we should expect to see correlations. If effort were relatively constant however then if availability increased and catchability remained the same then catches would increase and so would cpue, thus the correlation at lag = 0 would be high. But despite such complications it still appears to be the case that if no significant correlations at negative time lags occur then valid spm or aspm model fits to the cpue data do not seem to occur.

Fitting the spm Model

The first step is to start guessing initial parameter sets until an approximate solution presents itself. This eases the way for the minimization of the negative log-likelihood (-verLL) used to fit the model to the data. If the initial guesses are too far from the final solution then the numerical methods sued to conduct the minimization may fail to converge on a viable solution. If the stock is thought to have been depleted before the collection of data began then three parameters are required, otherwise only two are needed. In the example below these parameters are *r = 0.16*, *K = 6700*, and *Binit = 3500* (which implies an initial depletion of 3500/6700, which is ~52%). When making these initial guesses it is helpful to both plot the implied predicted trajectory of cpue against the observed cpue, and include the estimation of the negative log-likelihood. In that way, if the guessed model fit to the data begins to improve it is easily detected by the -veLL getting smaller (more negative in this case).

The plot in **Figure 28** merely represents the first guess at the parameters so it is not surprising that the model fit to data is not particularly impressive. Guessing a set of parameters and displaying their implications is an important first step to get the predicted cpue even approximating the observed cpue. Only once such an approximation is produced should you think about trying to fit the model to the data formally. Model fitting may produce sensible results before an approximate solution is found but often it does not. It is possible to use the catch-MSY method to generate first guesses at the *r* and *K* parameters, but producing initial values for *Binit*, if it is to be used, is currently down to trial and error.

```
# display the dynamics for a set of guessed initial parameters
pars <- c(0.16,6700,3500)  # r, K, and Binit
negLL(pars,fish,simpspm)

## [1] -2.862565

ans <- displayModel(pars,fish,schaefer=TRUE,target=0.48,addrmse=TRUE)
```

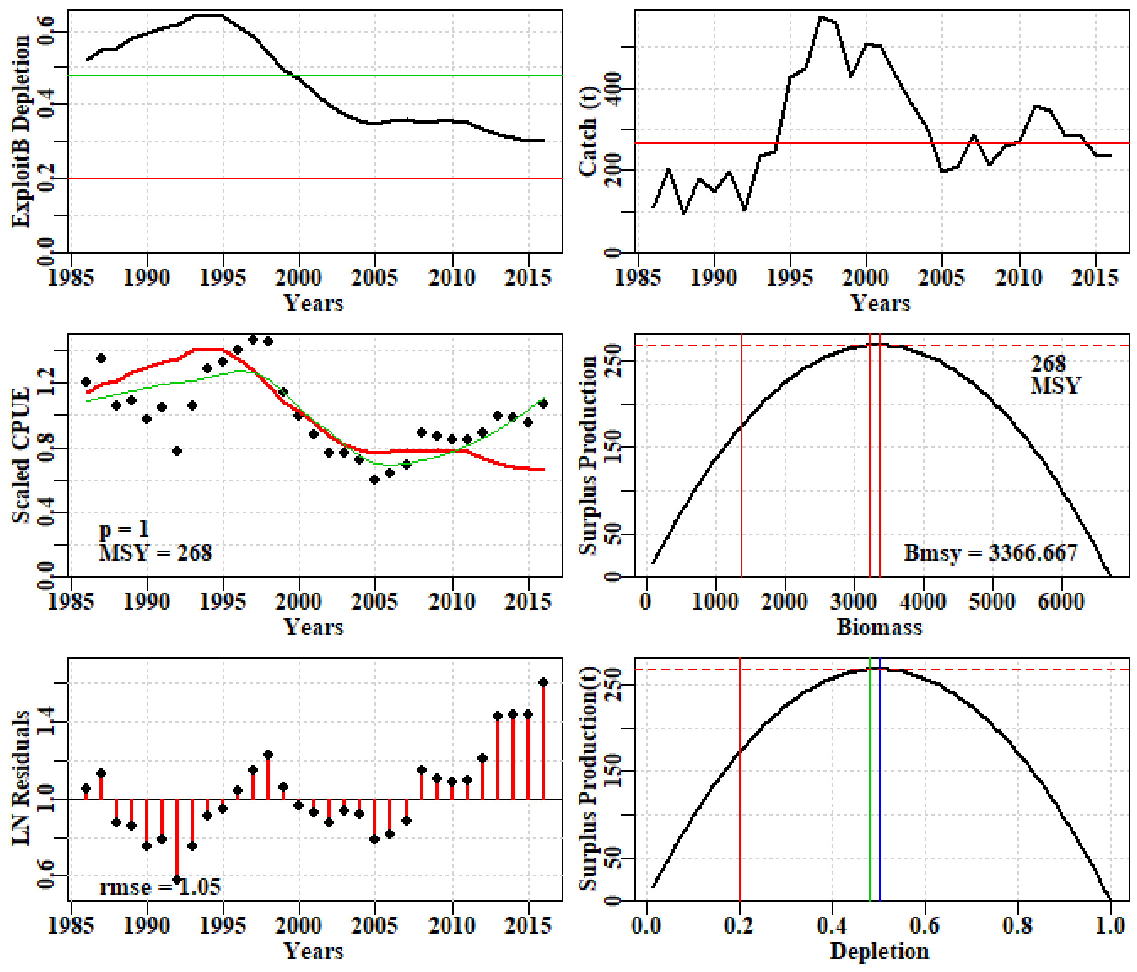


Figure 28. A summary plot depicting the fit of the model using the guessed input parameters. The plots include the predicted depletion and catch through time. Then the fit to cpue, where the green line is a loess fit to the observed cpue and the red line is the fit using the parameters. The surplus production curve is given twice, once with biomass and once with depletion on the x-axis, finally the log-normal residuals between the fit and the data are illustrated. These initial parameters have not been fitted.

Rules of thumb can be used when attempting to find suitable starting points. A possible approximation for the unfished biomass (*K* parameter) can be obtained by multiplying the maximum catch by 10 to 15. As in the catch-MSY method, if the initial catches in the available data are a reasonable proportion of the maximum catch then the stock can be assumed to have been somewhat depleted prior to data collection. A value between 0.5 to 0.7 times the selected *K* value could be used for a *Binit* value. One might then search for an *r* value perhaps starting at something like 0.3. Remember that the *r* parameter as a combination of both the reproductive rate and the natural mortality rate is a net population growth rate not only the reproductive rate. This means it may be lower than a simple consideration of the species' life history might suggesst. A plausible set of parameters would be where the implied predicted cpue trajectory even roughy approximates the observed trajectory. The next step after finding a plausible starting point is to attempt to use numerical methods to fit the data to the model.

The production curves are an expression of the *r* - *K* parameters where, for the Schaefer model, the MSY is assumed to occur at half the unfished biomass (*K*; see the model equations appendix). Using the production curve it will be possible to estimate the predicted catch at B_{Target} and B_{Limit}, and from those it will be possible to estimate the harvest rates at the target and limit reference points. Two of the potential input parameters to the *displayModel* function are the *limit* and *target*, which default to 0.2 and 0.48, obviously these can be set to whatever obtains in the jurisdiction involved.

Fitting the Model to the Data

There are three functions available for working with the dynamics of surplus production models. The most comprehensive in its output is *spm*, which generates the full dynamics and returns a matrix of all values (try *out <- spm(inp=pars, indat=fish, schaefer = TRUE)*. The default Schaefer model dynamics is defined by the *schaefer = TRUE* parameter value, to use the Fox model the option *schaefer = FALSE* should be used. The model fitting only compares the observed cpue with the predicted cpue and does not require the full dynamics to be produced, so, in order to speed up the iterative process of numerically searching for a model fit, we also have a simplified *simpspm* function that only outputs the predicted cpue. This function also has a *schaefer* parameter to determine whether to use a Schaefer or the Fox model. The *simpspm* function is used in the model fitting whereas, if you inspect the code for *displayModel* you will find that it uses *spm*. In the following we will use *optim* which is a solver or minimizer that comes as part of the base R installation. Check out *?optim* or *formals(optim)*, or more thoroughly you could read the *Optimization* task view at https://cran.r-project.org/. The code below is a template that should work for you. The control list is not always needed but it usually helps to keep the solver stable. If you check the code for *negLL* by typing that in the R console without brackets, you will see there is code (*na.rm=TRUE*)to exclude records where there may be catch data but no observed cpue data. Ideally one has data for both in each year, which generally leads to more stable outcomes, but in reality this does not always occur.

Instead of using *optim* or one of the other non-linear minimizers in R it is possible to use *fitSPM*, which is a wrapper within **simpleSA** that contains two runs through the optim fitting routine (see later).

```
pars <- c(0.16,6700,3500) # r, K, and Binit, implies a depletion of
~0.52
cat("initial -ve log-likelihood ",negLL(pars,fish,simpspm),"\n")

## initial -ve log-likelihood  -2.862565

bestSP <- optim(par=pars,fn=negLL,callfun=simpspm,indat=fish,
                control=list(maxit = 1000, parscale = c(1,1000,1000)))
outoptim(bestSP) # outoptim just prints the results more compactly.

## $par         :  0.2424071 5173.363 2845.831
## $value       :  -12.12879
## $counts      :  192 NA   iterations, gradient
## $convergence :  0
## $message     :

# read the help files to understand what each component means.
# Schaefer model dynamics is the default so no need for schaefer=TRUE
```

Now plot up the optimum solution, in this case using the Schaefer model. Note in this case that the final depletion is close to the approximate target green line defined by the loess smoother fit to the observed data (a result of the *addrmse=TRUE*). Note also that the years in which catches were greater than the MSY (the red line in the top right plot) were the years in which the stock declined.

```
ans <- displayModel(bestSP$par,fish,schaefer=TRUE,addrmse=TRUE)
```

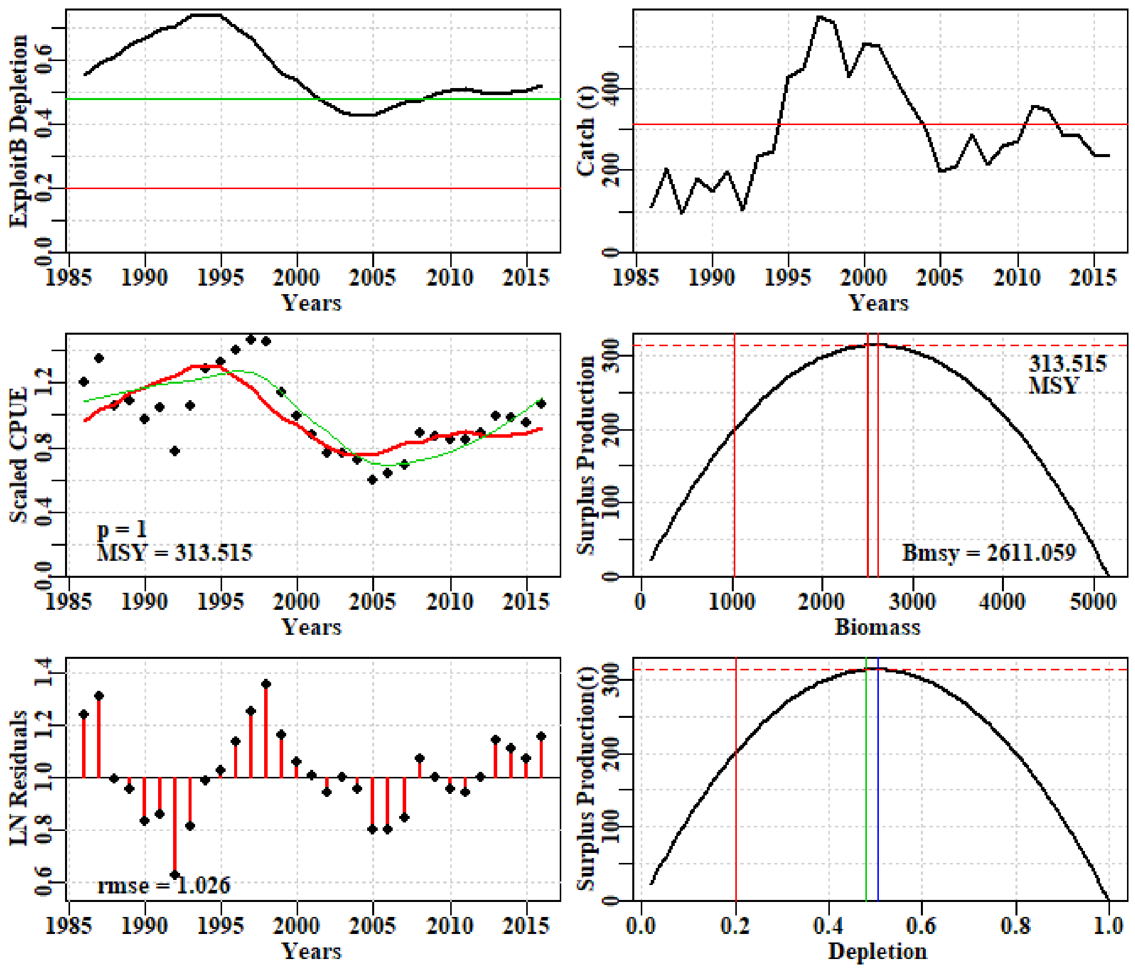


Figure 29. A summary plot depicting the fit of the optimum parameters. The plots include the predicted depletion and catch through time. Then the fit to cpue, where the green line is a loess fit to the data and the red line is the fit using the parameters. The surplus production curve is given with biomass and depletion on the x-axes (note the symmetry of the production curve), finally the residuals between the fit and the data are illustrated.

There are some patterns in the residuals which may suggest there are other factors at play on the fishery dyanamics, but it is clear there are some changes occurring which the model is not sufficiently flexible to enable a fit. Such simple surplus production models ignore good and bad recruitment years and so some deviations from observed are to be expected. Whether this is a sufficient reason for such patterned deviations would require extra independent evidence to be determined.

```
str(ans)

## List of 11
##  $ Dynamics :List of 5
##   ..$ outmat    : num [1:31, 1:8] 2846 3043 3141 3344 3448 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:31] "1986" "1987" "1988" "1989" ...
##   .. .. ..$ : chr [1:8] "ModelB" "Catch" "CPUE" "PredCE" ...
##   ..$ q         : num 0.00034
##   ..$ msy       : num 314
##   ..$ parameters: num [1:3] 0.242 5173.363 2845.831
##   ..$ sumout    : Named num [1:9] 0.242 5173.363 2845.831 313.515 1
...
##   .. ..- attr(*, "names")= chr [1:9] "r" "K" "B0" "msy" ...
##  $ BiomProd : num [1:100, 1:2] 100 151 202 254 305 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:2] "x" "y"
##  $ rmseresid: num 1.03
##  $ MSY      : num 313
##  $ Bmsy     : num 2611
##  $ Dmsy     : num 0.505
##  $ Blim     : num 1022
##  $ Btarg    : num 2509
##  $ Ctarg    : num 313
##  $ Dcurr    : Named num 0.522
##   ..- attr(*, "names")= chr "2016"
##  $ rmse     : num 0.137
```

The object output from `displayModel` has a number of parts including the dynamics, various statistics relating to the model fit, the model parameters, and the production curve. These are used by other functions to summarize and plot the outcomes. However, first it would be best to be confident that one has a unique and optimum model fit.

The model fitted was the Schaefer model so it may be worth considering the outcome had the alternative Fox model been used. This can be implemented by making simple changes to the same functions as before. In this particular case the initial values that enabled a solution for the Schaefer model also work for the Fox model, which is not always the case. Note the fit is slightly better but in fact the predicted cpue trajectories barely differ. Note also the use of magnitude to estimate the scale of each parameter and hence simplify the use of the *parscale* option within the control list.

```
pars <- c(0.16,6700,3500) # r, K, and Binit, implies a depletion of
~0.52
cat("initial -ve log-likelihood
",negLL(pars,fish,simpspm,schaefer=FALSE),"\n")

## initial -ve log-likelihood  -5.998584

parscl <- magnitude(pars)
bestSP <-
optim(par=pars,fn=negLL,callfun=simpspm,indat=fish,schaefer=FALSE,
                control=list(maxit = 1000, parscale = parscl))
outoptim(bestSP   )

## $par         :  0.1382109 6129.655 2757.171
## $value       :  -12.35283
## $counts      :  160 NA   iterations, gradient
## $convergence :  0
## $message     :

# Fox model dynamics is set by schaefer=FALSE

ans <- displayModel(bestSP$par,fish,schaefer=FALSE,addrmse=TRUE)
```

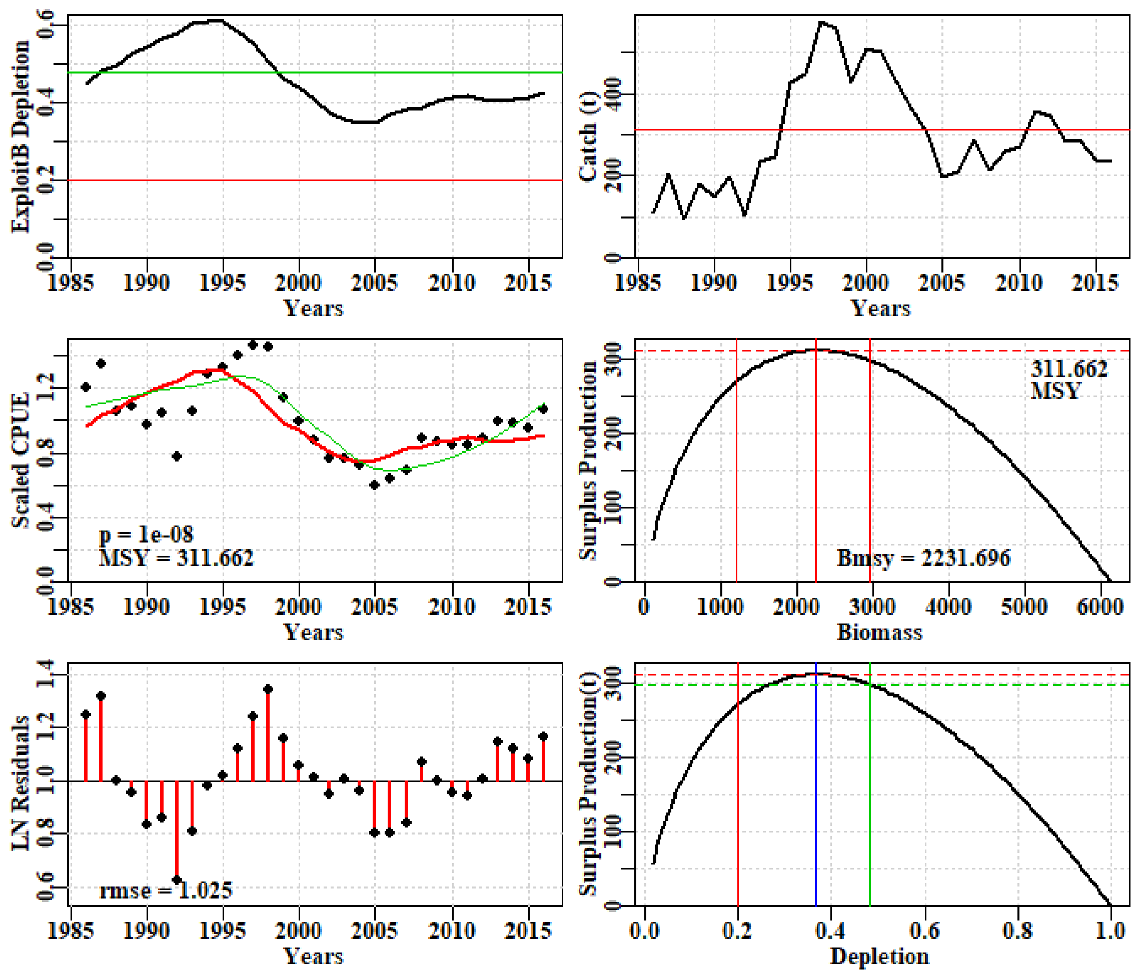


Figure 30. A summary plot depicting the fit of the optimum parameters. The plots include the predicted depletion and catch through time. Then the fit to cpue, where the green line is a loess fit to the data and the red line is the fit using the parameters. The surplus production curve is given with biomass and depletion on the x-axes (note the asymmetry of the production curve), finally the residuals between the fit and the data are illustrated.

```
str(ans)

## List of 11
##  $ Dynamics :List of 5
##   ..$ outmat    : num [1:31, 1:8] 2757 2949 3041 3240 3342 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:31] "1986" "1987" "1988" "1989" ...
##   .. .. ..$ : chr [1:8] "ModelB" "Catch" "CPUE" "PredCE" ...
##   ..$ q         : num 0.00035
##   ..$ msy       : num 312
##   ..$ parameters: num [1:3] 0.138 6129.655 2757.171
##   ..$ sumout    : Named num [1:9] 1.38e-01 6.13e+03 2.76e+03
3.12e+02 1.00e-08 ...
##   .. ..- attr(*, "names")= chr [1:9] "r" "K" "B0" "msy" ...
##  $ BiomProd : num [1:100, 1:2] 100 161 222 283 344 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:2] "x" "y"
##  $ rmseresid: num 1.03
##  $ MSY      : num 312
##  $ Bmsy     : num 2232
##  $ Dmsy     : num 0.364
##  $ Blim     : num 1196
##  $ Btarg    : num 2963
##  $ Ctarg    : num 298
##  $ Dcurr    : Named num 0.425
##   ..- attr(*, "names")= chr "2016"
##  $ rmse     : num 0.137
```

Is the Solution the Global Optimum?

In the ideal world you should always try solving from multiple starting points. A general function is under development but in the meantime you could try using something like this. Here we have used the fitSPM function, which runs through the fitting process twice in case the first run through failed to find the best solution (try *?fitSPM* or just *fitSPM* with no brackets):

```
  pars <- c(0.16,6700,3500) # the original starting point
  origpar <- pars
  N <-  20        # the number of random starting points to try
  scaler <- 10    # how variable; smaller = more variable
  # define a matrix for the results
  columns <- c("ir","iK","iB0","iLike","r","K","Binit","-
veLL","MSY","Iters")
  results <-
matrix(0,nrow=N,ncol=length(columns),dimnames=list(1:N,columns))
  pars <- cbind(rnorm(N,mean=origpar[1],sd=origpar[1]/scaler),
                rnorm(N,mean=origpar[2],sd=origpar[2]/scaler),
                rnorm(N,mean=origpar[3],sd=origpar[3]/scaler))
  # this randomness ignores the strong correlation between r and K
  for (i in 1:N) {
     bestSP <- fitSPM(pars[i,],fish,schaefer=TRUE)
     opar <- bestSP$par
     msy <- opar[1]*opar[2]/4
     origLL <- negLL(pars[i,],fish,simpspm)
     results[i,] <-
c(pars[i,],origLL,bestSP$par,bestSP$value,msy,bestSP$counts[1])
  }
  cat("\n\n\n")

kable(results[order(results[,"-
veLL"]),],digits=c(0,3,3,3,3,3,3,3,3,0))
```

	ir	iK	iB0	iLike	r	K	Binit	-veLL	MSY	Iters
12	0	7054.497	4293.170	-3.715	0.242	5173.514	2846.040	-12.129	313.513	284
13	0	8145.097	2969.235	3.457	0.242	5173.634	2846.102	-12.129	313.513	174
17	0	6990.304	3561.189	12.675	0.242	5173.596	2846.153	-12.129	313.514	176
11	0	7268.679	3221.805	-7.749	0.242	5173.411	2846.070	-12.129	313.517	152
8	0	6261.529	3812.002	-5.681	0.242	5173.579	2845.957	-12.129	313.514	148
15	0	8272.553	3682.467	-4.294	0.242	5173.522	2846.076	-12.129	313.512	286
19	0	7360.068	3673.318	-10.776	0.242	5173.531	2845.885	-12.129	313.514	178
4	0	6662.154	3159.226	-4.168	0.242	5173.644	2846.229	-12.129	313.515	232
3	0	6962.042	3375.847	-8.338	0.242	5173.402	2845.870	-12.129	313.517	196
16	0	6670.589	3317.342	-10.705	0.242	5173.573	2846.273	-12.129	313.514	204
18	0	6350.766	3392.096	19.590	0.242	5173.231	2846.066	-12.129	313.519	150

6	0	6025.200	2799.990	55.220	0.242	5173.480	2845.871	-12.129	313.510	110
2	0	7157.476	3917.461	-10.720	0.242	5173.980	2846.404	-12.129	313.509	156
14	0	6166.552	3612.957	10.208	0.242	5173.961	2846.218	-12.129	313.511	126
10	0	6549.646	3707.303	54.352	0.242	5173.852	2845.990	-12.129	313.506	154
7	0	7244.823	3637.724	14.967	0.242	5173.886	2846.159	-12.129	313.513	212
20	0	5961.728	3425.592	53.339	0.242	5173.220	2845.616	-12.129	313.511	112
5	0	6463.095	2924.876	47.923	0.242	5174.464	2846.192	-12.129	313.497	122
1	0	6189.280	2904.257	43.275	0.242	5174.868	2847.300	-12.129	313.490	92
9	0	6374.195	3467.723	12.473	0.243	5168.933	2842.278	-12.129	313.572	124

```
cat("\n\n")

kable(apply(results,2,range),digits=c(0,3,3,3,3,3,3,3,3,3,0))
```

ir	iK	iB0	iLike	r	K	Binit	-veLL	MSY	Iters
0	5961.728	2799.99	-10.776	0.242	5168.933	2842.278	-12.129	313.490	92
0	8272.553	4293.17	55.220	0.243	5174.868	2847.300	-12.129	313.572	286

```
cat("\n\n")

round(apply(results,2,median),3)

##        ir        iK       iB0     iLike         r         K     Binit      -
veLL       MSY     Iters
##    0.160 6666.371 3446.658    6.832    0.242 5173.576 2846.073  -
12.129  313.513  155.000
```

By repeating that analysis multiple times (more than 20, and try changing the *scaler* variable) you will eventually find solutions that fall far from the most common optimum. This is valuable in demonstrating that the initial starting values can be influential on the final solution and that one should not immediately trust a solution found numerically until it has been tested and pushed around to determine how robust it is to initial conditions. Notice that any outlying values, if they occur, are not common, so the use of the median will usually exclude them from an estimate of the central tendency for the parameters and stock properties such as MSY. We can use the median values of the optimum estimates when we are plotting up a phase plot of biomass against fishing mortality to illustrate the stock status. Had we used a single *optim* run instead of the two found inside *fitSPM* more failures to find the optimum are likely to have occurred.

Produce a Phase Plot

In the SAFS process determining the stock status is an important component. This is simply achieved once you have run the function *displayModel* using the optimum parameters. First fit an optimum model (here we use the median estimates of a set of runs like those produced in the previous example). Once we have applied the *displayModel* function using the optimum parameter estimates we can produce the phase plot.

```
pars <- c(0.2424, 5173.5972, 2846.0953) # r, K, and Binit, median
values
bestSP <- fitSPM(pars,fish,schaefer=TRUE)
ans <- displayModel(bestSP$par,fish,schaefer=TRUE,addrmse=TRUE)
```

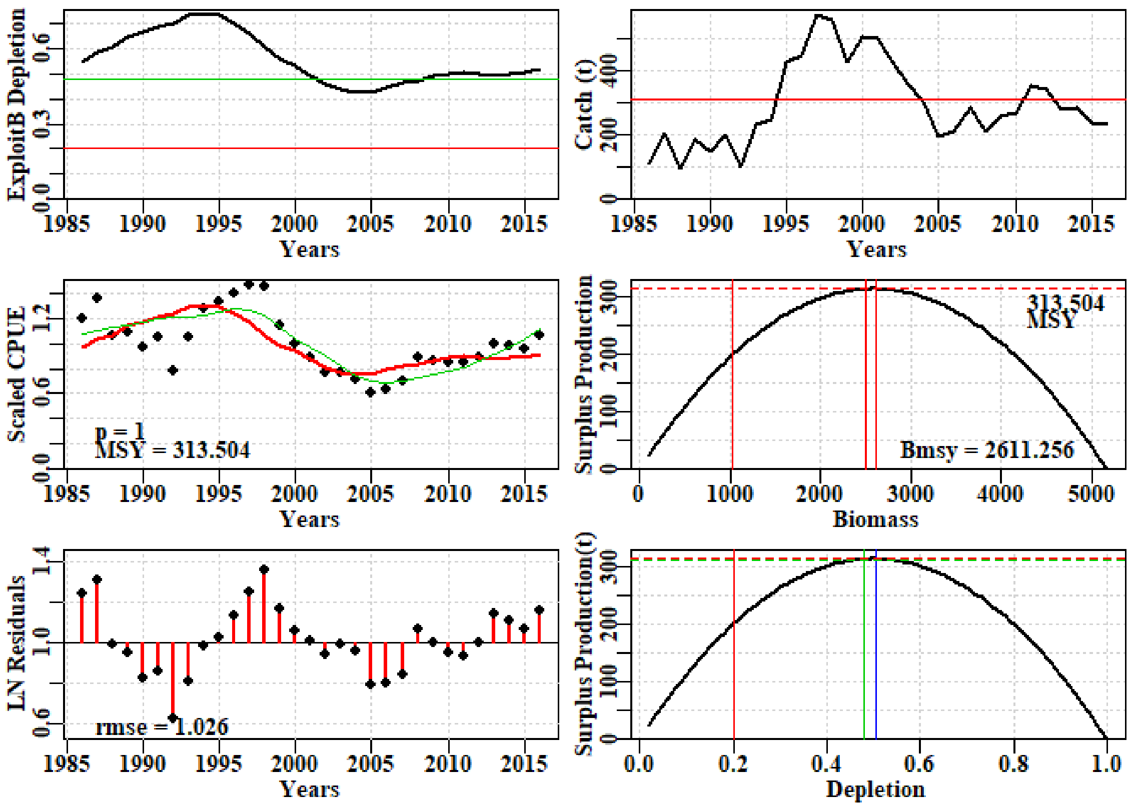


Figure 31. A summary plot depicting the fit after using the median optimum parameters from the repeated starting points depicted in the analysis above. The plots include the predicted depletion and catch through time. Then the fit to cpue, where the green line is a loess fit to the data and the red line is the fit using the parameters. The surplus production curve is given with biomass and depletion on the x-axes, finally the residuals between the fit and the data are illustrated.

The object output by *displayModel* is used to generate a phase plot of the trajectory of stock status implied by the surplus production model's description of the fishery's dynamics.

```
# plotprep(width=7,height=5.5) # to avoid using the RStudio plot
window
spmphaseplot(ans,fnt=7)
```

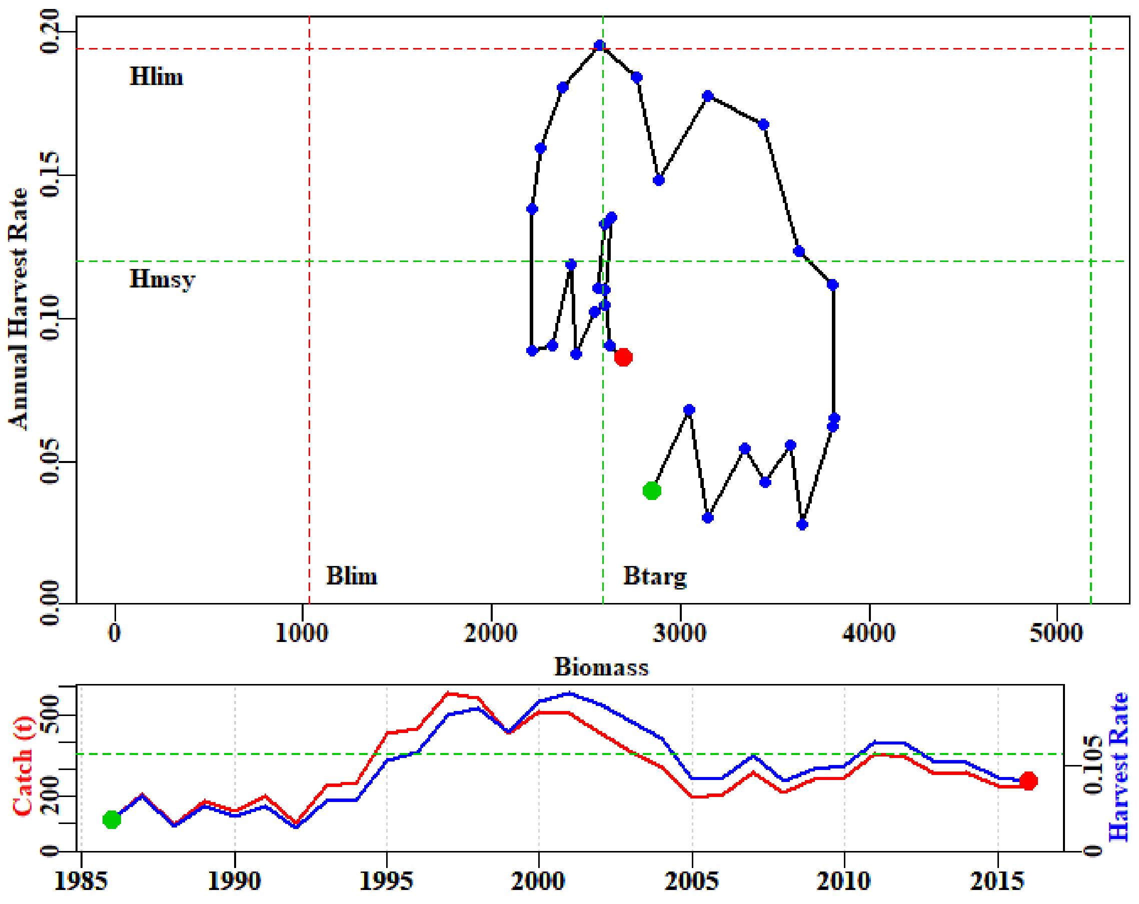


Figure 32. The top plot is a phase plot of the predicted stock biomass versus the predicted annual harvest rate for the optimum model fit. The green dot is the starting year and the red dot the final year. the limit reference points are included and tentative target reference points also to aid interpretation. The catch history and related harvest rates are illustrated in the lower plot also with an aim to aid interpretation of the status trajectory (in which time is only implied).

In the case illustrated the current status is both above the limit and the target biomass reference points, and below the limit and target harvest rate reference points and so can be safely concluded to be sustainable. A more detailed summary would conclude no over-fishing and not over-fished. The catch history indicates that the first eight years of catches led to harvest rates sufficiently low as to allow the stock to increase in size but that as catches increased from 1994 onwards they quickly rose above the harvest rate that would, in theory, maintain the stock at its target and that led to the stock declining over the following 10 years, after which catches declined again so that the stock slowly recovered back up to the target biomass over the next 10 years, with occasional increases in harvest rate up close to the target rate.

Generate Bootstrap Confidence Intervals

Every fishery analysis is invariably uncertain and an effort should always be made to include at least an indication of that uncertainty into any report of a stock assessment. One such characterization included in **simpleSA** for the *spm* and *aspm* models is the generation of percentile confidence intervals around parameters and model outputs (MSY, etc) by taking bootstrap samples of the log-normal residuals associated with the cpue and using those to generate new bootstrap cpue samples with which to replace the original cpue time-series (Haddon, 2011). Each time such a bootstrap sample is made the model is re-fit and the solutions obtained stored for further analysis to conduct such an analysis one uses the *bootspm* function. Once we have found suitable starting parameters we use the *fitSPM* function, which uses optim twice sequentially with a negative likelihood approach to obtain an optimum fit.

```
data(dataspm)
fish <- dataspm$fish
colnames(fish) <- tolower(colnames(fish))
pars <- c(r=0.25,K=5500,Binit=2900)
ans <- fitSPM(pars,fish,schaefer=TRUE,maxiter=1000) #Schaefer version
answer <- displayModel(ans$par,fish,schaefer=TRUE,addrmse=TRUE)
```

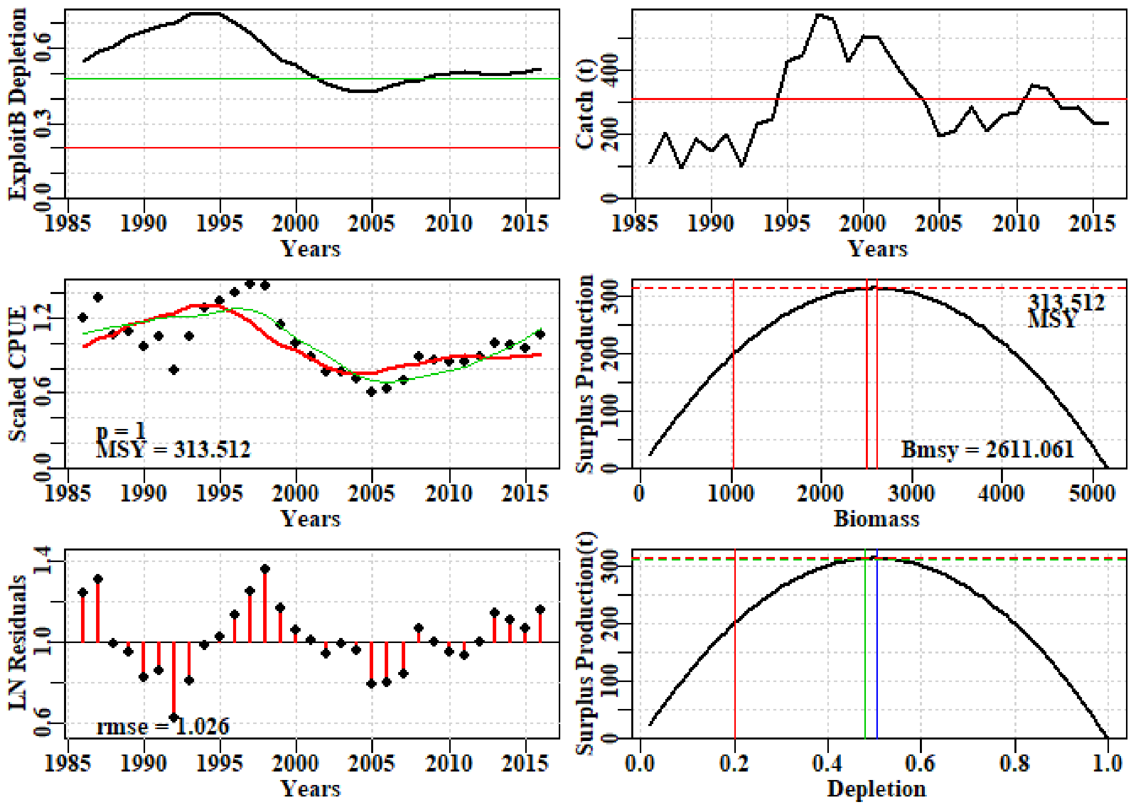


Figure 33. A summary plot depicting the fit of the optimum parameters to the dataspm dataset. The residuals between the fit and the cpue data are illustrated at the bottom left. These are what are bootstrapped and each sample multiplied by the optimum predicted cpue time-series to obtain each bootstrap cpue time-series.

Once we have an optimum fit we can proceed to conduct a bootstrap analysis. Here we are only running 100 replicates for speed but you would usually run at least 1000 even though that might take a few minutes to complete.

```
reps <- 100        # this might take ~60 seconds, be patient
startime <- Sys.time()  # how long will this take
boots <- bootspm(ans$par,fishery=fish,iter=reps,schaefer=TRUE)
print(Sys.time() - startime)

## Time difference of 4.887969 secs

str(boots)

## List of 3
##  $ dynam       : num [1:100, 1:31, 1:5] 2846 3553 2739 3480 2860
...
```

```
##   ..- attr(*, "dimnames")=List of 3
##   .. ..$ : chr [1:100] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:31] "1986" "1987" "1988" "1989" ...
##   .. ..$ : chr [1:5] "ModelB" "BootCE" "PredCE" "Depletion" ...
##  $ bootpar     : num [1:100, 1:6] 0.242 0.271 0.31 0.265 0.234 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:100] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:6] "r" "K" "Binit" "MSY" ...
##  $ negativepars: num 0
```

The output contains the dynamics of each run with the predicted model biomass, each bootstrap cpue sample, the predicted cpue for each bootstrap sample, the depletion time-series, and the annual harvest rate time-series. Each of these can be used to illustrate and summarize the outcomes and uncertainty within the analysis. Given the relatively large residuals in **Figure 7** one might expect a relatively high degree of uncertainty.

```
dynam <- boots$dynam
bootpar <- boots$bootpar
rows <- colnames(bootpar)
columns <- c(c(0.025,0.05,0.5,0.95,0.975),"Mean")
bootCI <- matrix(NA,nrow=length(rows),ncol=length(columns),
                 dimnames=list(rows,columns))
for (i in 1:length(rows)) { # i=1
   tmp <- sort(bootpar[,i])
   qtil <-
quantile(tmp,probs=c(0.025,0.05,0.5,0.95,0.975),na.rm=TRUE)
   bootCI[i,] <- c(qtil,mean(tmp,na.rm=TRUE))
}
kable(bootCI,digits=c(3,3,3,3,3,3))
```

	0.025	0.05	0.5	0.95	0.975	Mean
r	0.109	0.143	0.256	0.359	0.373	0.250
K	3635.081	3671.058	5082.288	7894.464	10086.318	10509.205
Binit	1648.772	1801.246	2796.433	5668.114	7736.649	3161.843
MSY	268.633	280.265	319.086	353.151	361.007	342.134
Depl	0.387	0.394	0.541	0.655	0.682	0.526
Harv	0.047	0.055	0.085	0.118	0.124	0.087

Such percentile confidence intervals can be visualized using histograms and including the respective selected percentile CI.

```
par(mfrow=c(3,2),mai=c(0.45,0.45,0.15,0.05),oma=c(0.0,0,0.0,0.0))
par(cex=0.85, mgp=c(1.35,0.35,0), font.axis=7,font=7,font.lab=7)
hist(bootpar[,"r"],breaks=25,col=2,main="",xlab="r")
abline(v=c(bootCI["r",c(2,3,4,6)]),col=c(3,3,3,4),lwd=c(1,2,1,2))
hist(bootpar[,"K"],breaks=25,col=2,main="",xlab="K")
abline(v=c(bootCI["K",c(2,3,4,6)]),col=c(3,3,3,4),lwd=c(1,2,1,2))
hist(bootpar[,"Binit"],breaks=25,col=2,main="",xlab="Binit")
abline(v=c(bootCI["Binit",c(2,3,4,6)]),col=c(3,3,3,4),lwd=c(1,2,1,2))
hist(bootpar[,"MSY"],breaks=25,col=2,main="",xlab="MSY")
abline(v=c(bootCI["MSY",c(2,3,4,6)]),col=c(3,3,3,4),lwd=c(1,2,1,2))
hist(bootpar[,"Depl"],breaks=25,col=2,main="",xlab="Final Depletion")
abline(v=c(bootCI["Depl",c(2,3,4,6)]),col=c(3,3,3,4),lwd=c(1,2,1,2))
hist(bootpar[,"Harv"],breaks=25,col=2,main="",xlab="Final Harvest
Rate")
abline(v=c(bootCI["Harv",c(2,3,4,6)]),col=c(3,3,3,4),lwd=c(1,2,1,2))
```

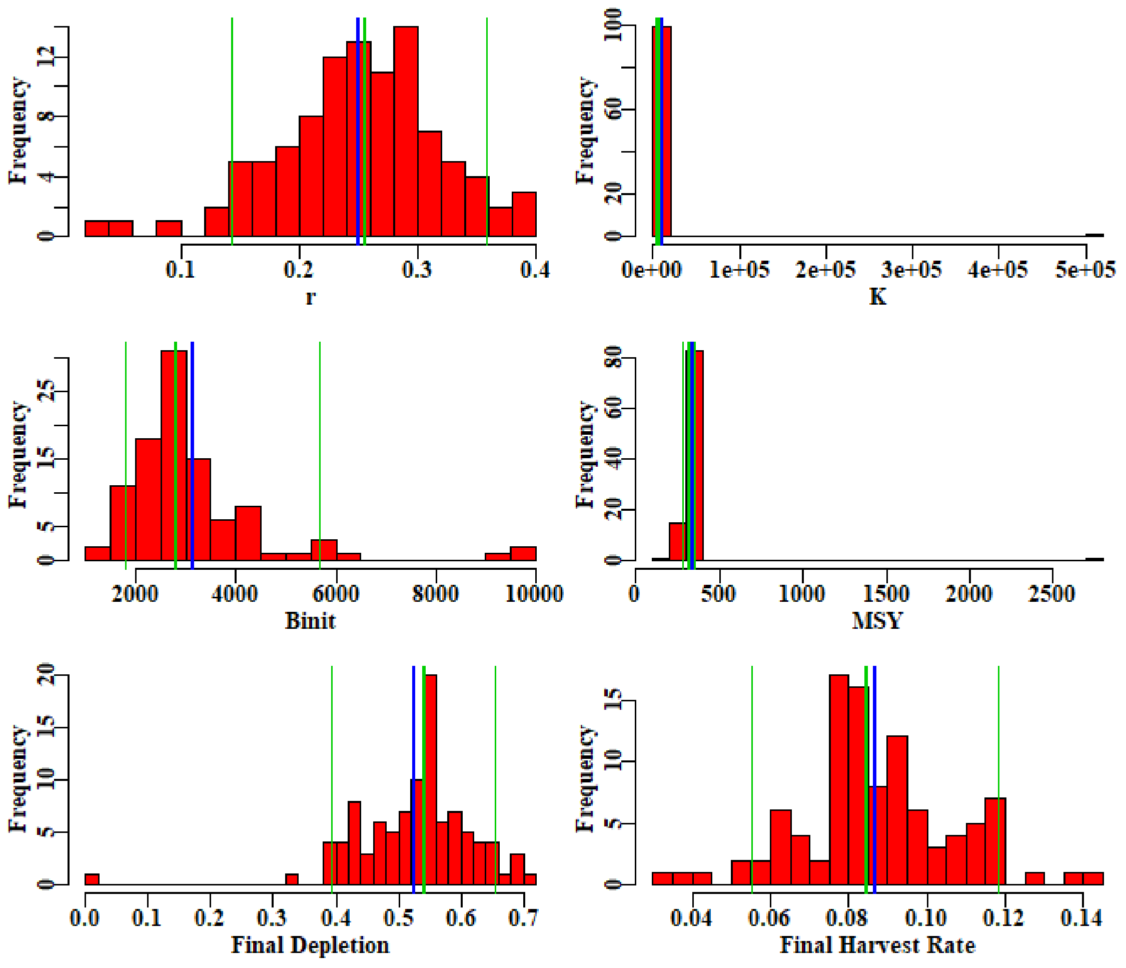


Figure 34. The bootstrap replicates from the optimum spm fit to the dataspm data set. The vertical green lines, in each case, are the median and 90th percentile confidence intervals and the blue lines are the mean values. There is little evidence of bias with the ‘r’ and ‘MSY’, estimates, although both the median ‘K’ and ‘Binit’ values appear somewhat biased low, while the depletion and harvest rates appear slightly biased high.

Of course, with only 100 replicates in such a variable analysis these results remain only very approximate. One would expect 1000 replicates would provide for a smoother response and more representative confidence bounds (100 were used simply to demonstrate the principle within a short time; this may be sped up later using C++). Note that the confidence bounds are not necessarily symmetrical around either the mean or the median estimates. Notice also that with the final year depletion estimates the 10th percentile CI is well above 20%B_0, implying that even though this analysis is uncertain the current depletion level is above the default limit reference point with more than a 90% likelihood (again more replicates are required before we could validly claim this).

The fitted trajectories can also provide an indication of the uncertainty surrounding the analysis.

```
years <- fish[,"year"]
par(mfrow=c(1,1),mai=c(0.45,0.45,0.05,0.05),oma=c(0.0,0,0.0,0.0))
par(cex=0.85, mgp=c(1.35,0.35,0), font.axis=7,font=7,font.lab=7)
ymax <- getmaxy(c(dynam[,,"PredCE"],fish[,"cpue"]))
plot(fish[,"year"],fish[,"cpue"],type="n",ylim=c(0,ymax),xlab="Year",
     ylab="CPUE",panel.first = grid())
for (i in 1:reps) lines(years,dynam[i,,"PredCE"],lwd=1,col="grey")
```

```
lines(years,answer$Dynamics$outmat[,"PredCE"],lwd=2,col=2)
points(years,fish[,"cpue"],cex=1.1,pch=16,col=4)
percs <- apply(dynam[,,"PredCE"],2,quants)
arrows(x0=years,y0=percs["5%",],y1=percs["95%",],length=0.03,angle=90,
code=3,col=2)
```

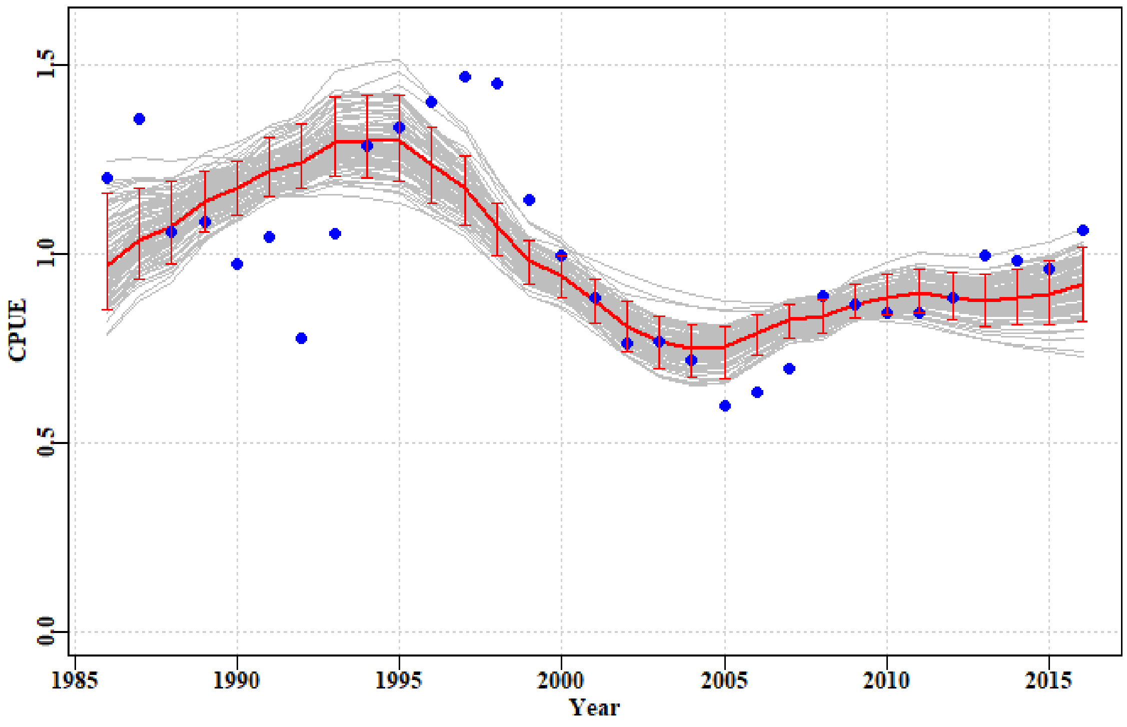


Figure 35. A plot of the original observed CPUE (blue dots), the optimum predicted CPUE (solid red line), the bootstrap predicted CPUE (the grey lines), and the 90th percentile confidence intervals around those predicted values.

There are clearly some major deviations between the predicted and the observed CPUE values but the median estimates and the confidence bounds around them remain well defined.

Discussion

In the Southern and Eastern Scalefish and Shark Fishery (SESSF), rather than using surplus production models or other simple dynamics approaches, empirical harvest strategies have been developed that use such time-series in empirical relationships that give rise directly to management related advice on catch levels (Little et al., 2011; Haddon, 2014). Such empirical harvest strategies can provide the needed management advice but do not determine stock status unless the reference period, used in such approaches, is assumed to be a proxy for the target reference point (and associated limit reference point) for sustainability. In the SESSF, this Tier 4 harvest strategy is used to determine whether a stock is over-fished or not, but currently cannot be used to determine whether over-fishing is occurring. In addition, there is the strong assumption made that the commercial catch rate are a direct reflection of the stock biomass. There are, however, some species, for example mirror dory (*Zenopsis nebulosa*) where catch rates increase when catches increase and then decline once catches begin to decline (see the discussion above about whether the CPUE is informative). They appear to be fisheries based on availability rather than the fishery being the major influence on the stock biomass other aspects of the environment of the species appear to be driving its dynamics. The use of CPUE may this be misleading in such cases.

Management Advice with spm

If working with a species that requires on-going management then it is necessary to produce advice with respect to acceptable catches that will lead to a sustainable fishery or whatever other management goal is in place for the fishery. To generate such advice some form of harvest strategy (HS) is required to allow the outputs from the assessment to be converted into a recommended biological catch (RBC; informal harvest strategies are less repeatable than formal HS). This RBC may then be modified by fishery managers taking into account potential rates of change within a fishery or social or economic drivers of management decisions. It was possible to put forward suggestions for new harvest strategies using the catch-MSY method because none were available previously and that put forward was only a suggestion for a possible consideration. Putting forward a proposed harvest control rule for the spm approach without consultation with jurisdictional fisheries managers could produce suggestions incompatible with a particular jurisdictions objectives. There are harvest control rules that can be used once limit and target reference points are agreed upon and these can be utilized where considered appropriate. The SAFS process, however, does not currently require a target reference point even though most harvest control rules do require one.

Appendix: Surplus Production Model Equations

The surplus production model is used to describe the dynamics of the stock in terms of its exploitable biomass. In general terms where the dynamics are designated a function of the biomass at the start of a given year *t*:

$$B_{t+1} = B_t + f(B_t) - C_t$$

where B_t_ represents the stock biomass at the start of year t, $f(B_t)$ represents a production function in terms of stock biomass (accounting for recruitment or new individuals, growth of biomass of current individuals, and natural mortality), and C_t being the catch in year t.

The model dynamics are also used to generate predicted values for the index of relative abundance in each year:

$$\hat{I}_t = \frac{C_t}{E_t} = qB_t$$

where $\hat{I}_t$ is the predicted or estimated value of the index of relative abundance, which is compared to the observed indices to fit the model to the data, E_t is the fishing effort in year t, and q is the catchability coefficient (defined as the amount of biomass/catch taken with one unit of effort).

A number of functional forms have been put forward to describe production. In **simpleSA** we have implemented two, the Schaefer model and a modified form of the Fox model:

The Schaefer (1954) model uses:

$$f(B_t) = rB_t\left(1 - \frac{B_t}{K}\right)$$

while the Fox (1970) uses:

$$f(B_t) = Ln(K)rB_t\left(1 - \frac{Ln(B_t)}{Ln(K)}\right)$$

Alternatively, a formulation from Polacheck *et al.* (1993) provides a general equation for the population dynamics that can be used for both the Schaefer and the Fox model depending on the value of a single parameter p.

$$B_{t+1} = B_t + \frac{r}{p}B_t\left(1 - \frac{B_t}{K}\right)^p - C_t$$

Thus, when p is set to 1.0 this equation becomes the same as the Schaefer model. But when p is set to a very small number, say 1e-0.8, then the formulation becomes equivalent to the Fox model's dynamics.

The Schaefer model assumes a symmetrical production curve with maximum surplus production (MSY) at 0.5_K_. The Fox model generates asymmetrical production curves with the maximum production at some lower level of depletion. The Schaefer model can be regarded as more conservative than the Fox in that it requires the stock size to be higher for maximum production and generally leads to somewhat lower levels of catch.

Other production functions could be added later. r represents a population growth rate that includes the balance between recruitment, growth in current biomass, and natural mortality, K is the maximum population size (the carrying capacity),the part in brackets, $\left(1-\frac{B_t}{K}\right)$, represents a density dependent term that trends linearly to zero as B_t tends to wards K in the Schaefer model. In the Schaefer mdoel the production curve is symmetric with maximum production occurring at $0.5K$. In the Fox model the density dependent terms introduces some non-linearity which generates an asymmetrical production curve with the point of maximum production moved to $< 0.5K$ (see later diagrams of the prodution curves; See Haddon (2011) for further details).

Thus for the Schaefer model we would have:

$$B_{t+1} = B_t + rB_t\left(1-\frac{B_t}{K}\right) - C_t$$

where $B_0 = K$ or B_{init} depending on whether the stock was deemed to be depleted when data from the fishery first became available.

It appears that fitting the model to data would require at least three parameters, the r, the K, and the q (B_{init} might also be needed). However, it is possible to used what is known as a "closed-form" method for estimating the catchability coefficient q:

$$\hat{q} = \exp\left(\frac{1}{n}\sum Ln\left(\frac{I_t}{B_t}\right)\right)$$

which is the back-transformed geometric mean of the observed CPUE divided by the exploitable biomass.

Sum of Squared Residuals

Such a model can be fitted using least squares or, more properly, the sum of squared residual errors:

$$ssq = \sum\left(Ln(I_t) - Ln\left(\hat{I}_t\right)\right)^2$$

the log-transformations are required as CPUE is considered typically to be distributed log-normally and the least-squares method implies normal random errors. The least squares approach tends to be more robust when first searching for a set of parameters that enable a model to fit to available data. However, once close to a solution more options become available if one then uses maximum likelihood methods.

Maximum likelihood methods, as the name dictates entail maximizing the likelihood of the available data given the model and a proposed set of parameters. Very often the likelihoods involved when fitting models are very small numbers. To avoid rounding errors (even when using 64 bit computers) it is standard to use log-likelihoods rather than likelihoods (in that way the log-likelihoods can be individually added together rather than multiply the individual likelihoods). Additionally, rather than maximizing a log-likelihood, minimization often best matches our intuitions about model fitting and this involves minimizing the negative log-likelihood. The full log-normal negative log likelihood is:

$$L(data|B_{init}, r, K, q) = \prod_t \frac{1}{I_t\sqrt{2\pi}\,\hat{\sigma}} e^{\frac{-\left(LnI_t - Ln\hat{I}_t\right)}{2\hat{\sigma}^2}}$$

Fortunately, the negative log-likelihood can be simplified (Haddon, 2011), to become:

$$-veLL = \frac{n}{2}\left(Ln(2\pi) + 2Ln(\hat{\sigma}) + 1\right)$$

where the maximum likelihood estimate of the standard deviation, $\hat{\sigma}$ is given by:

$$\hat{\sigma} = \sqrt{\frac{\sum\left(Ln(I_t) - Ln\left(\hat{I}_t\right)\right)^2}{n}}$$

Note the division by *n* rather than by *n-1*. Strictly the -veLL is followed by an additional term:

$$-\sum Ln(I_t)$$

the sum of the log-transformed observed catch rates. But as this will be constant it is usually omitted.

Estimating Management Statistics

The Maximum Sustainable Yield can be calculated for the Schaefer model simply by using:

$$MSY = \frac{rK}{4}$$

However, for the more general equation using the *p* parameter from Polacheck *et al* (1993) one needs to use:

$$MSY = \frac{rK}{(p+1)^{\frac{(p+1)}{p}}}$$

aspm - Age-Structured Production Models

Introduction

Which Stock Assessment?

Which stock assessment method to apply to fisheries for data-poor to data-moderate species will depend upon what fisheries and biological data are available but also, importantly, on what management objectives need to be met within the jurisdiction in question. It may be the case that the fishery for a particular species is of sufficient size and value to warrant on-going monitoring and management towards some defined goal for the stock. In such a case the assessment used should obviously be capable of generating some notion of the current state of the fishery and indicate what management actions may be required to eventually achieve the agreed management goals. But some fisheries may be so minor that trying to actively manage them would be inefficient. Nevertheless, to meet the requirements of the Status of key Australian Fish Stocks (SAFS) one still requires some form of defensible stock assessment capable of determining whether the current level of fishing is sustainable.

Age-Structured Production Model

Introduction to ASPM

The age-structured production model (ASPM or **aspm**) is literally a surplus production model which is based upon an age-structured model of production rather than an accumulated biomass model (see the vignette on *spm*).

There are some specific data requirements for fitting an age-structured production model to fishery data. Data from the fishery need to included, as a minimum, an accurate catch time-series plus an index of relative abundance for at least some of the years within the catch time-series. In addition, information (or defensible assumptions) is needed for the species concerned in relation to the description of its natural mortality, its growth, its maturation, and the selectivity of the fishery (maturity and selectivity could be knife-edge). If just the catches and CPUE data are available then one might try fitting a simple, aggregated biomass surplus production model. But if these biological data and information are available then an age-structured production model opens the way to on-going improvements with respect to the inclusion of occasional age-composition data or other observations that could be predicted by a suitable model and hence included in the model fitting process.

More details on age-structured production models can be found in Punt et al. (1995). The model equations are provided in the appendix. It would be helpful to the user to read the spm vignette as the theory in there also applies to age-structured production models.

A Typical Workflow

A typical workflow for using an age-structured production model might be something like:

1. read in the available data and use *checkdata* to ensure it can be used with **aspm**.
2. search for suitable initial parameter values using *dynamics*, *aspmLL*, and *plotASPM*, this will include deciding on the use of a two parameter model (no

initial depletion) or a three parameter model accounting for an initial depletion level.

3. given suitable initial parameters use *fitASPM* or more basically *optim* to fit the model to the available data.
4. once successfully fitted it is best to plot the solution to determine visually how close the model fit is to the data using *plotASPM*. One approach to improving this is to include confidence intervals around the index of relative abundance (cpue). This is done first using *getLNCI* and then including the output of this into *plotASPM* or, for a closer look in *plotceASPM*. Eventually improved confidence intervals for the model outputs can be obtained using boostrap samples (see below).
5. A better test of the robustness of the solution is to test how sensitive it is to the initial conditions. This can be done by randomly varying the initial parameters and determining whether the model fitting outcomes vary. Suitable example code is given in the vignette.
6. After finally deciding on the overall optimum solution it would be sensible to use the optimum parameters to determine the implied production curve so that management statistics such as MSY, the target catch, and the limit and target reference points can be defined. This can be done using *getProductionC*, the *C* post-fix denoting this is a C++ function (used for speed), the results of which can be plotted and summarized using *prodASPM*.
7. One can produce a classical phase plot of predicted biomass vs harvest rate to aid in the determination of the stock status, and for this we would use *aspmphaseplot*.
8. Finally, to obtain a better notion of the uncertainty in the analysis, and we can do that using *bootASPM*, which facilitates the application of a bootstrap analysis.

An Example

The data requirements for the **aspm** are described above and for the next example we will use the *dataspm* built in data set. First we will examine an example using only two parameters (assuming the population begins in an unfished state) and then extend the model fitting to include the possibility of an initially depleted state. The two parameters being fitted are the average unfished recruitment level and the standard deviation of the errors around the CPUE data.

We load *dataspm*, which contains a dataframe of the catches and CPUE by year (*fish*) and other parameters used for the age-structured production model.

```
library(simpleSA)
data(dataspm)
fish <- dataspm$fish
props <- dataspm$props # length-, weight-, maturity- and selectivity-
at-age
```

While a simple, aggregated biomass surplus production model only requires the specification of the species name in the *glb* data object an **aspm** requires information on the growth, selectivity, weight-at-age, steepness of the stock-recruit relationship and natural mortality. The global parameters (*glb*), in addition to the *spsname*, need to contain the population ages (*maxage* and *ages*), natural mortality (*M*), von Bertalanffy growth parameters (*Linf*, *K* and *t0*), weight-at-age parameters (*Waa* and *Wab*), age at 50% maturity, delta (*M50a* and *deltaM*), age that 50% of the population are selected by the gear and delta (*sela50* and *deltaS*) and the *steepness* of the stock-recruitment relationship. The number of years (*nyrs*) of over which catch and CPUE are available (including missing years) is calculated from the *fish* dataframe. A starting value for the log of the initial recruitment (*R0*) also needs to be provided, although this will be estimated along with the standard deviation of the errors around the CPUE data.

We inspect the global parameters specified in *dataspm*.

```
(glb <- dataspm$glb)

## $maxage
## [1] 20
##
## $M
## [1] 0.225
##
## $Linf
## [1] 103.4
##
## $K
## [1] 0.2
##
## $t0
## [1] -3.139
##
## $Waa
## [1] 0.0029
##
## $Wab
## [1] 3.139
##
## $M50a
## [1] 5
##
## $deltaM
## [1] 2.5
##
## $steep
## [1] 0.75
##
## $R0
## [1] 13.7
##
## $sela50
## [1] 3.5
##
## $deltaS
## [1] 1
##
## $resilience
## [1] "low"
##
## $nages
## [1] 21
##
## $ages
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##
```

```
## $nyrs
## [1] 31
##
## $spsname
## [1] "TrawlCaught_Scalefish"
```

Just as with the **spm** model Fitting an **aspm** model entails first finding initial parameter estimates that lead to predicted cpue time-series that approximate the observed series. Thus, one might begin something like this and uses aspmLL to determine the negative log-likelihood (-verLL), then *dynamics* to calculate the dynamics of the **aspm**, and finally *plotASPM* to illustrate the quality of fit to help decide whether changes are required to the initial guesses.:

```
pars <- c(12.9,0.25)
aspmLL(pars,infish=fish,inglb=glb,inprops=props)

## [1] 386.8754

fishery <- dynamics(pars,infish=fish,inglb=glb,inprops = props)
plotASPM(fishery)
```

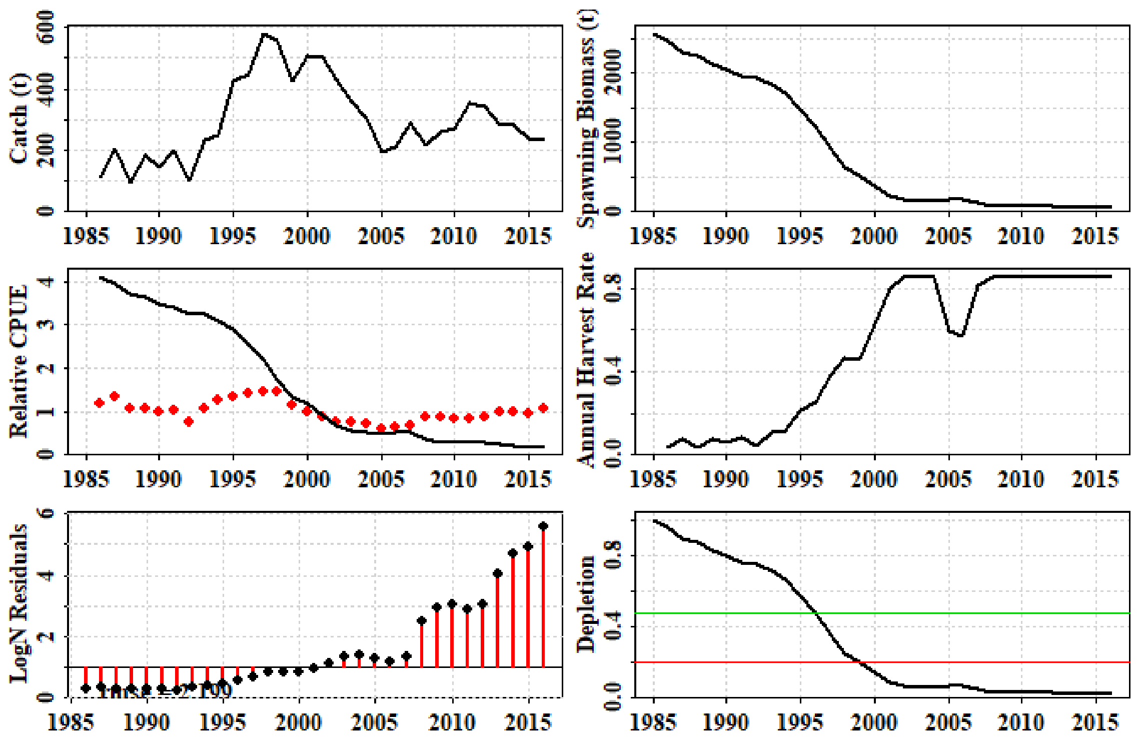


Figure 36. The outcome of a first guess at a two parameter version of the **aspm**. clearly the fit is very poor and the strong trends in the predicted cpue, the log-normal residuals, and the annual harvest rate bumping up against the built-in upper limit of 0.85 across numerous years, provide a strong indication that the initial guess at unfished average recruitment parameter (*R0*) is too small. Try increasing it slowly to see its effect on the model fit to the data.

Once reasonable starting values have been found for the parameters (R_0 the unfished average recruitment and $\hat{\sigma}_I$, the standard deviation associated with fitting the observed cpue) then an attempt at fitting the model to the data formally can be made using code something like this:

```
pars <- c(13.7,0.19)
ans <- fitASPM(pars,infish=fish,inglb=glb,inprops=props)
outoptim(ans) # a tidier way of printing the list output from optim

## $par          :  13.69138 0.189471
## $value        :  -7.582633
## $counts       :  55 NA   iterations, gradient
## $convergence :  0
## $message      :

fishery <- dynamics(ans$par,infish=fish,inglb=glb,inprops = props)
```

Table 1. The output from the *fitASPM* function and the *dynamics* function.

```r
kable(fishery,digits=c(0,3,3,3,3,3,4,4,3))
```

	Year	Catch	PredC	SpawnB	ExploitB	FullH	CPUE	PredCE	Deplete
0	1985			5643.463	6216.733				1.000
1	1986	112.9	112.9	5547.856	6119.538	0.018	1.2006	1.2731	0.983
2	1987	206.3	206.3	5381.150	5955.372	0.034	1.3547	1.2532	0.954
3	1988	95.7	95.7	5325.183	5910.455	0.016	1.0585	1.2196	0.944
4	1989	183.1	183.1	5208.120	5799.303	0.031	1.0846	1.2104	0.923
5	1990	147.4	147.4	5136.775	5735.954	0.025	0.9738	1.1876	0.910
6	1991	198.9	198.9	5034.632	5638.377	0.035	1.0437	1.1747	0.892
7	1992	102.1	102.1	5027.222	5638.064	0.018	0.7759	1.1547	0.891
8	1993	235.5	235.5	4914.029	5523.921	0.042	1.0532	1.1546	0.871
9	1994	247.8	247.8	4801.433	5414.330	0.045	1.2840	1.1312	0.851
10	1995	426.8	426.8	4552.038	5165.789	0.079	1.3327	1.1088	0.807
11	1996	448.0	448.0	4310.173	4933.418	0.087	1.4014	1.0579	0.764
12	1997	577.4	577.4	3991.273	4623.718	0.117	1.4687	1.0103	0.707
13	1998	558.5	558.5	3726.747	4374.663	0.121	1.4493	0.9469	0.660
14	1999	427.9	427.9	3607.166	4275.655	0.098	1.1420	0.8959	0.639
15	2000	509.3	509.3	3447.541	4121.138	0.119	0.9957	0.8756	0.611
16	2001	502.4	502.4	3314.139	3991.335	0.122	0.8818	0.8440	0.587
17	2002	429.6	429.6	3256.173	3938.955	0.108	0.7635	0.8174	0.577
18	2003	360.2	360.2	3264.635	3951.252	0.091	0.7668	0.8066	0.578
19	2004	306.2	306.2	3319.664	4007.258	0.077	0.7198	0.8092	0.588
20	2005	195.7	195.7	3460.581	4149.242	0.049	0.5997	0.8206	0.613
21	2006	210.0	210.0	3578.106	4256.624	0.051	0.6336	0.8497	0.634
22	2007	287.3	287.3	3615.276	4278.291	0.067	0.6936	0.8717	0.641
23	2008	214.2	214.2	3698.766	4357.911	0.050	0.8894	0.8761	0.655
24	2009	260.6	260.6	3733.374	4387.110	0.060	0.8644	0.8924	0.662
25	2010	272.2	272.2	3751.418	4403.947	0.062	0.8442	0.8984	0.665
26	2011	356.9	356.9	3696.506	4346.717	0.081	0.8427	0.9019	0.655
27	2012	345.0	345.0	3654.393	4308.073	0.079	0.8849	0.8902	0.648
28	2013	282.7	282.7	3669.508	4331.070	0.066	0.9964	0.8822	0.650
29	2014	285.1	285.1	3686.377	4351.140	0.066	0.9804	0.8869	0.653
30	2015	237.8	237.8	3742.829	4410.295	0.055	0.9570	0.8911	0.663
31	2016	233.3	233.3	3799.558	4464.794	0.053	1.0629	0.9032	0.673

Note the predicted catches are identical to the observed catches. The catches are assumed to be known accurately and so as to ensure a close match between the predicted and observed catches *aspmLL* has a simple sum-of-squared deviations penalty built into it (try *aspmLL*, without brackets in the console). This is usually sufficient to force the solution to generate a close match once plausible parameter values are found. Note that with respect to reproduction only the average unfished recruitment is estimated. In its current form the **aspm** cannot take into accoutn strong and weak cohorts; this remains a very simple model of the dynamics.

To visualize the output of the model fitting we can plot some of the variables from the fishery information generated by the *dynamics* function.

```r
plotASPM(fishery,CI=NA)
```

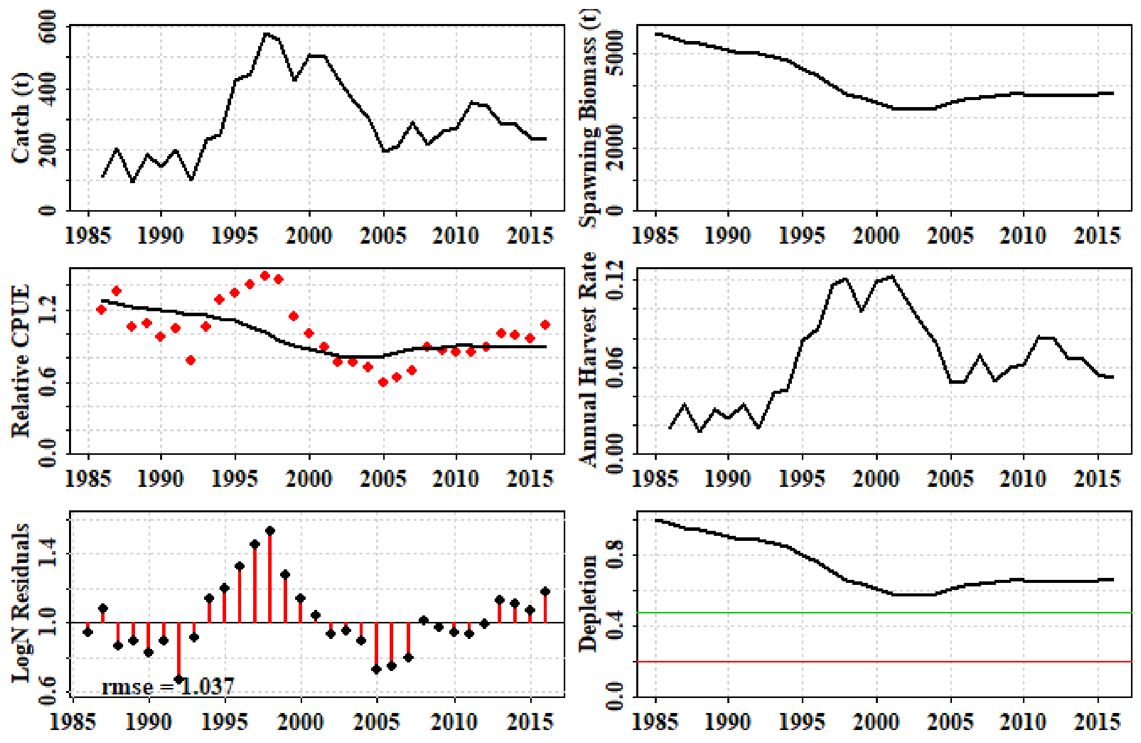


Figure 37. The outcome of fitting a two parameter age-structured production model to the available data. Included is a plot of the catch history, the predicted spawning biomass, the CPUE, the harvest rate, the log-normal residuals for the cpue, and the predicted depletion.

The two-parameter model manages to capture the main cpue trends but fails to capture some of the more obvious and more rapid consistent changes in cpue (**Figure 2**). The example fishery in question was known to have been fished prior to 1985 so in the year the data begin to be available the stock can be expected to be depleted to some extent. Thus, an alternative might be to fit the model using three parameters. The first approach, with only one parameter of real interest, required data from the beginning of the fishery. However, there are many fisheries for which data are only available after the fishery has been running for a number of years. In such cases it is necessary to estimate the level of depletion and its effect upon recruitment and thus requires two parameters of interest to the dynamics. The first is, as before, the unfished recruitment level, R_0, but then we use a parameter that defines the initial depletion at the start of the observations (D_{init}. If this parameter is present in *pars* a search is made for the constant harvest rate that when applied to the initial unfished stock leads to the predicted initial depletion level, only then does the model fitting proceed. Any initial depletion will influence the recruitment depending on the assumed steepness of the stock recruitment relationship, which is assumed to be a Beverton-Holt relationship. The *dataspm* data set is not particularly suited to a two-parameter model even though that arrangement was able to provide a result; these assessments should not be done automatically, it always takes some background knowledge to ensure that any methods or choices applied are valid.

As an alternative two-parameter example, the deep water fishery data *data(fishdat)* was a fishery with catch data from the very start of the fishery, which means it is better suited to using a simple two parameter model

```
data(fishdat)
fish <- fishdat$fish
glb <- fishdat$glb
props <- fishdat$props
```

```
pars <- c(14,0.3)
ans <- fitASPM(pars,infish=fish,inglb=glb,inprops = props)
outoptim(ans)

## $par         :  13.98282 0.2956361
## $value       :  3.606029
## $counts      :  71 NA   iterations, gradient
## $convergence :  0
## $message     :
```

The output contains the estimates of the optimum parameters, the final log-likelihood estimate, the number of iterations it needed to find the optimum, and some diagnostic information. The statement *convergence: int 0* implies the solution appears valid, and no warning messages is also encouraging; but see later concerning how to test the robustness of such model fitting. If we put the fitted optimum parameters into the function *dynamics* we can see the time-series of the more important population variables implied by the model fit.

```
fishery <- dynamics(ans$par,infish=fish,inglb=glb,inprops = props)
kable(fishery,digits=c(0,3,3,1,1,3,3,3,3))
```

	Year	Catch	PredC	SpawnB	ExploitB	FullH	CPUE	PredCE	Deplete
0	1985			18014.3	17693.0				1.000
1	1986	3924.912	3924.912	14248.2	13994.0	0.222		7.553	0.791
2	1987	5117.988	5117.988	9428.1	9259.9	0.366		5.974	0.523
3	1988	4722.200	4722.200	5129.8	5038.3	0.510		3.953	0.285
4	1989	1365.128	1365.128	4105.5	4032.3	0.271	1.490	2.151	0.228
5	1990	801.567	801.567	3641.7	3576.7	0.199	1.849	1.721	0.202
6	1991	625.407	625.407	3369.2	3309.1	0.175	1.501	1.527	0.187
7	1992	1108.241	1108.241	2707.0	2658.7	0.335	2.997	1.413	0.150
8	1993	964.409	964.409	2201.1	2161.8	0.363	1.415	1.135	0.122
9	1994	800.618	800.618	1858.5	1825.4	0.370	1.139	0.923	0.103
10	1995	962.399	962.399	1413.5	1388.3	0.527	0.700	0.779	0.078
11	1996	1180.349	1180.031	845.1	830.1	0.850	0.469	0.593	0.047
12	1997	297.003	297.003	925.3	908.8	0.358	0.303	0.354	0.051
13	1998	316.131	316.131	1007.3	989.4	0.348	0.356	0.388	0.056
14	1999	210.529	210.529	1165.6	1144.8	0.213	0.390	0.422	0.065
15	2000	169.337	169.337	1365.7	1341.4	0.148	0.439	0.489	0.076
16	2001	200.843	200.843	1559.3	1531.5	0.150	0.489	0.573	0.087
17	2002	255.735	255.735	1722.4	1691.7	0.167	0.431	0.654	0.096
18	2003	217.502	217.502	1916.4	1882.2	0.129	0.520	0.722	0.106
19	2004	283.110	283.110	2064.5	2027.7	0.150	0.777	0.804	0.115
20	2005	264.607	264.607	2225.4	2185.7	0.130	1.322	0.866	0.124
21	2006	139.316	139.316	2484.6	2440.2	0.064	1.412	0.933	0.138
22	2007	28.571	28.571	2838.9	2788.3	0.012		1.042	0.158
23	2008	3.331	3.331	3224.3	3166.8	0.001		1.190	0.179
24	2009	13.859	13.859	3608.2	3543.8	0.004		1.352	0.200
25	2010	21.440	21.440	3988.3	3917.1	0.006		1.513	0.221
26	2011	31.426	31.426	4358.7	4281.0	0.008		1.672	0.242
27	2012	17.253	17.253	4737.4	4652.9	0.004		1.828	0.263
28	2013	35.940	35.940	5093.6	5002.8	0.008		1.986	0.283
29	2014	22.087	22.087	5453.2	5355.9	0.004		2.136	0.303
30	2015	16.206	16.206	5806.4	5702.9	0.003		2.287	0.322

The use of *fitASPM* is basically shorthand for using *bestL <- optim(pars, aspmLL, method="Nelder-Mead", infish=fish, inglb=glb, inprops=props, control=list(maxit = 1000, parscale = c(10,1)))* twice in a row. Examine its code by using *fitASPM* without brackets in the R console window.

Note that the harvest rate in 1996 appears to have bumped up against the upper limit of 0.85 hard wired into the R-code (try typing *dynamics* into the R console without brackets to see the code). So that 1180 tonnes catch in 1996 was likely to be damaging. However, the predicted catch is only slightly less than the reported catch so this spike in harvest rate has some support in the data. Plotting up the fishery results enables the visualization of the impact of the fishery and he effect of cutting back catches.

```
ceCI <- getLNCI(fishery[,"PredCE"],ans$par[2])
plotASPM(fishery,CI=ceCI)
```

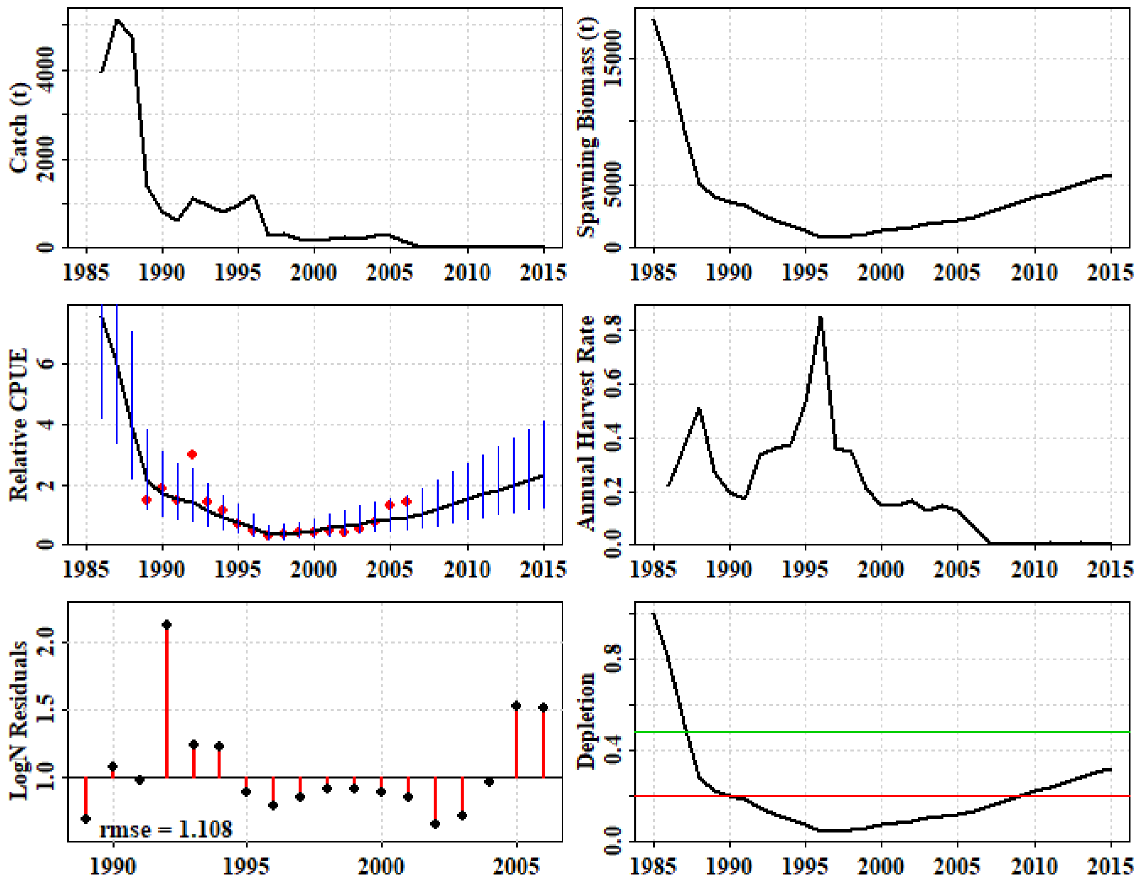


Figure 38. The outcome of fitting a two parameter age-structured production model to the deep-water fishery data. If the function *plotceASPM* is used then only the CPUE with confidence intervals are plotted.

The modelling suggests that once catches were reduced to an average of about 246 t between 1997 - 2005 the stock began to very slowly recover, then, after catches were further reduced from 2007 onwards (due to a deepwater closure and cessation of targeting) the rate of recover was predicted to have increased. The model predicts that the stock breached above the $0.2B_0$ limit reference point in about 2010. However, extrapolation beyond the data must always be treated with a great deal of caution. The uncertainty about the catch rates is relatively large, especially because of the records from 1992, 2005, and 2006 deviate so much from the rest of the trend. Without confirmation from other data the predicted recovery from 2007 onwards is purely dirven by the predictions from the fitted model. The predicted recovery should not be accepted on the basis of the model fit alone, there needs to be other data confirming such a recovery. Whether the predicted recruitment from the model actually happened would require auxilliary information or data to corroborate the prediction. With Orange Roughy, because the fishery was so short lived and they only mature between 31 - 35 years of age, the biology suggested that the fishery is still receiving unfished recruitment levels, which can be expected to decline once the recruits produced from the depleted population are supposed to begin entering the fishery. However, given the lowest point of the stock is predicted to have occurred in 1996 the expected minimum in recruitment should only occur in about 2026 - 2030.

One way of estimating confidence intervals around the cpue is to use the standard deviation estimates from the likelihood fitting of the CPUE (parameter 2) and set up log-normal error bars, but later we will consider bootstrap percentile confidence intervals as an alternative.

It is not surprising the bounds on the predicted CPUE become vary wide in the regions where there are no cpue data. But even where there are data the bounds are wide. A better estimation of the uncertainty is more likely to be generated by the bootstrap analysis. The variation expressed is primarily driven by the elevated values in 1992 and in 2005 and 2006. The period from 1989 - 2006 is assumed to related to when aggregations were not being targeted but occasionally, no doubt, smaller aggregations would have added heterogeneity to the cpue data. Certainly given this uncertainty it remains questionable whether one could validly claim that the likelihood of the limit reference point being passed was high based only on these data. It would be best to test the robustness of this result to the initial conditions by trialing the model fit using a large array of initial parameter values just to see whether the optimum result was repeatable and stable. We will consider this after introducing the three parameter models.

A three parameter model

Returning to the *dataspm* data set it is possible to use a three parameter model to fit this data accepting that the observations began when the stock had already been fished and would be expected to be partially depleted. There are details that need attention if we are to assume such a model structure. These are mathematical models and they can exhibit mathematical behaviour, such as negative recruitment or initial depletions much greater than 1.0, unless such behaviour is controlled. To avoid initial depletions > 1.0 we have implemented a slightly different maximum likelihood function so we should use *aspmPENLL* instead of *aspmLL*. In this case, where the initial depletion is estimated to be a long way below 1.0 it might not matter, but such penalties can stabilize even such supposedly safe parameter sets.

```r
data(dataspm)
fish <- dataspm$fish
glb <- dataspm$glb
props <- dataspm$props
pars <- c(14,0.19,0.6) # Fit 3 par__aspm__with penalty
# pars <- c(13.2794439,0.1731744,0.4933178) # for a second time
through
scalepar <- magnitude(pars)
bestL <- optim(pars,aspmPENLL,method="Nelder-Mead",
              infish=fish,inglb=glb,inprops=props,
              control=list(maxit = 1000,parscale=scalepar))
outoptim(bestL)

## $par         :  13.33983 0.1746212 0.6233444
## $value       :  -10.1113
## $counts      :  116 NA   iterations, gradient
## $convergence :  0
## $message     :

fisheryPen <- dynamics(bestL$par,infish=fish,inglb=glb,inprops=props)
ceCI <- getLNCI(fisheryPen[,"PredCE"],bestL$par[2])
plotASPM(fisheryPen,CI=ceCI)
```

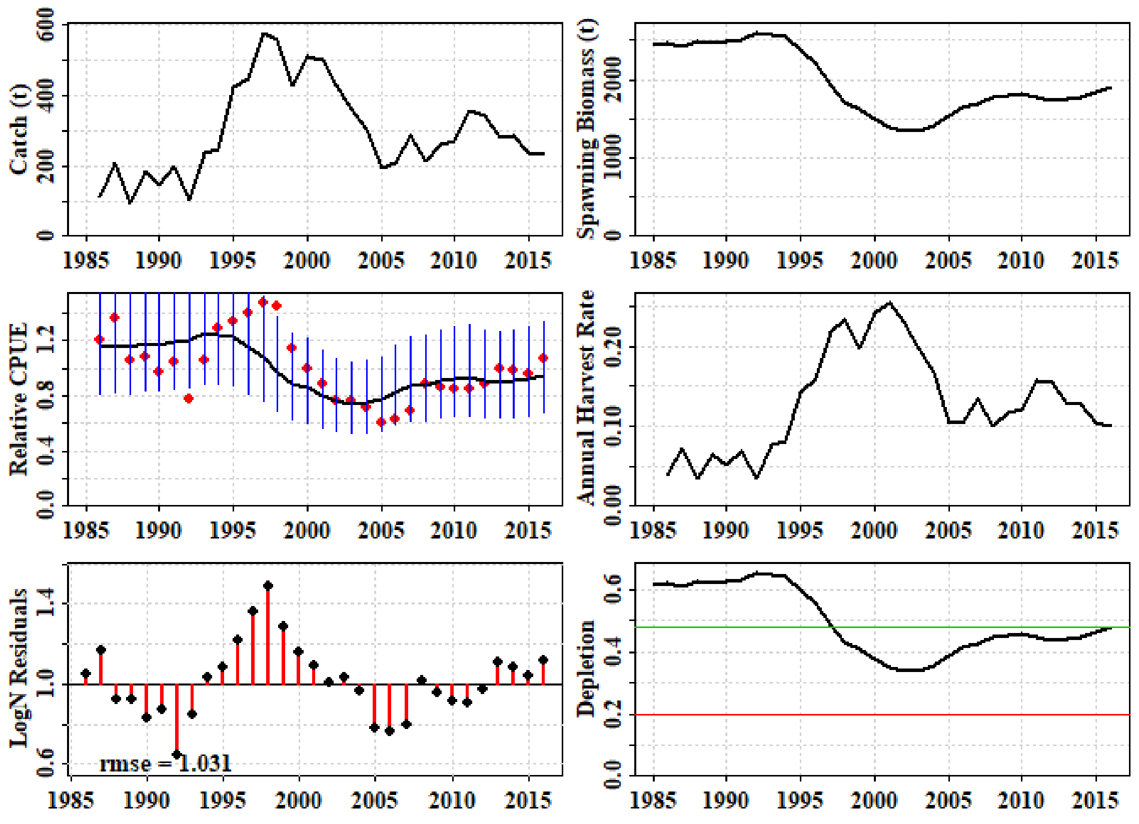


Figure 39. The outcome of fitting a three-parameter age-structured production model to the slope fishery data.

Once again the model fails to capture the more rapid changes in the predicted dynamics but does capture the general trends through time (**Figure 39**). Unlike the two-parameter model it predicts a final depletion close to 50% rather than 60% but this time suggests the starting depletion was about 65% rather than 100%. However, the -ve log-likelihood in this case is -9.95 rather than -7.58 as with the two-parameter model, indicating a slightly better fit (an improvement > 1.96 for each additional parameter suggests a better fit; Venzon and Moolgavkar, 1988).

Despite the improved fit to the data the confidence intervals around the CPUE remain very large. Thus, using the three parameter model one should test the robustness of this fit to the initial conditions to determine whether or not the outcome is stable following the model fitting or whether there is variation (uncertainty) and if so how much.

Testing the Robustness of the Model Fit

The sensitivity of the model fit to the initial parameter values is important to explore to gain greater confidence that the solution one finds is a global optimum rather than some local minima on the log-likelihood surface.

To test for robustness of the model fit we can use the original optimal model parameters or the original guesses, add variation to them, and re-fit the model. This process should enable an analysis of the stability of the modelling outcomes. If the optimum parameters are used then add more variation to enusre the parameter space is covered. The first parameter is $Log(R_0)$ so to simplify the selection of random variations away from the original it helps to return that parameter to the linear scale and only when finished return it to the log-scale.

```
  set.seed(12335)  # to get repeatable results, normally not done
  data(fishdat)
  fish <- fishdat$fish
  glb <- fishdat$glb
  props <- fishdat$props
  pars <- c(14,0.3)
  out <- robustASPM(pars,fish,glb,props,scaler=20,N=15,console=FALSE)
  str(out,max.level=1)

## List of 3
##  $ results: num [1:15, 1:9] 14 14 14.1 14 14 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ range  : num [1:2, 1:9] 13.94 14.09 0.27 0.32 3.76 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ medians: Named num [1:9] 13.993 0.3 9.119 13.983 0.296 ...
##   ..- attr(*, "names")= chr [1:9] "iLnR0" "isigmaCE" "iLike" "LnR0"
...

  kable(out$results,digits=c(3,3,2,3,3,4,1,1,0))
```

	iLnR0	isigmaCE	iLike	LnR0	sigmaCE	-veLL	MSY	B0	Iters
9	14.034	0.290	9.12	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	71
15	13.992	0.303	4.40	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	71
5	14.055	0.285	11.63	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	71
11	13.993	0.307	4.50	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	71
8	13.977	0.304	10.27	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	71
12	14.014	0.283	7.03	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	71
2	14.009	0.270	6.78	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	69
10	14.092	0.308	13.68	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	69
14	14.012	0.281	6.85	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	69
4	13.985	0.290	3.76	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	69
7	14.038	0.303	8.98	13.983	0.296	3.6060	2.399000e+02	1.801430e+04	67
6	13.966	0.297	49.14	42.176	0.660	18.0638	4.209864e+14	3.161862e+16	23
13	13.937	0.301	314.81	41.204	0.660	18.0638	1.592507e+14	1.196069e+16	25
3	13.960	0.320	91.58	46.792	0.660	18.0638	4.254969e+16	3.195738e+18	21
1	13.939	0.300	292.89	41.314	0.660	18.0638	1.777260e+14	1.334829e+16	21

Starting with the deep water fishery data *fishdat* we find that 11 out of 15 generate one solution, which appears to be optimum, while the remaining four, which all began with highly unlikely first guesses (*iLike*, the initial likelihood was large) all gave implausible outcomes. It would be sensible to explore this lack of robustness further by using many more iterations. However, given the variation in the cpue data this is not a surprising result.

If we test the robustness of the model fit to the *dataspm* data set (a three parameter model) similar outcomes arise.

```r
set.seed(12235)
data(dataspm)
fish <- dataspm$fish
glb <- dataspm$glb
props <- dataspm$props
pars <- c(14,0.2,0.6)
out <- robustASPM(pars,fish,glb,props,scaler=15,N=10,console=FALSE)
kable(out$results,digits=c(2,2,2,2,3,2,2,2,1,1,0))
```

	iLnR0	isigmaCE	iDepl	iLike	LnR0	sigmaCE	Depl	-veLL	MSY	B0	Iters
10	13.94	0.21	0.63	-0.91	13.279	0.17	0.49	-	343.8	3738.2	102
7	13.93	0.20	0.58	1.44	13.279	0.17	0.49	-	343.8	3738.2	261
6	13.83	0.21	0.57	-0.81	13.279	0.17	0.49	-	343.8	3738.2	104
9	14.10	0.18	0.64	3.23	13.279	0.17	0.49	-	343.8	3738.2	112
4	14.07	0.23	0.51	4.64	13.279	0.17	0.49	-	343.8	3738.2	183
2	13.97	0.17	0.59	4.60	13.280	0.17	0.49	-	343.8	3738.3	181
1	14.04	0.21	0.60	1.67	13.282	0.17	0.49	-	344.7	3747.6	186
3	14.04	0.20	0.53	5.71	13.301	0.17	0.55	-	351.4	3821.2	125
8	14.09	0.18	0.58	5.74	13.301	0.17	0.55	-	351.4	3821.1	119
5	13.92	0.18	0.60	2.11	13.315	0.17	0.57	-	356.2	3872.8	124

```r
kable(out$range,digits=c(2,2,2,2,3,2,2,2,1,1,0))
```

iLnR0	isigmaCE	iDepl	iLike	LnR0	sigmaCE	Depl	-veLL	MSY	B0	Iters
13.83	0.17	0.51	-0.91	13.279	0.17	0.49	-10.37	343.8	3738.2	102
14.10	0.23	0.64	5.74	13.315	0.17	0.57	-10.27	356.2	3872.8	261

Here we find that four final negative log-likelihoods differ from the optimum, although in this case the differences are not too far from the optimum. Very slight differences in the parameters even with the optimum -veLL lead to small differences in the derived statistics such as MSY and B_0. Once again the variation in the cpue data is what leads to this instability. Whatever the case it is to be hoped that these examples illustrate that one should never accept the final result of fitting a model even if the diagnostics look acceptable (the plot, the -veLL value, and optim gives convergence = 0). Without testing the robustness it is possible that one is settling for only a local minima. This is one reason why it is usually a good idea to run a fitting routine twice, once from the initial parameter guesses, the second time from the solution of the first time.

When testing the robustness ideally one would run very many trials (at least 100 to allow for proportional attribution of variation), in which case it becomes a reasonable proposition to plot the results. The correlations between the parameters can also be calculated (they tend ot be very high).

```
cor(out$results[,c("LnR0","Depl","-veLL","MSY")])  # correlations
between outputs

##            LnR0      Depl     -veLL       MSY
## LnR0  1.0000000 0.9926413 0.9882819 0.9999921
## Depl  0.9926413 1.0000000 0.9986348 0.9922690
## -veLL 0.9882819 0.9986348 1.0000000 0.9877160
## MSY   0.9999921 0.9922690 0.9877160 1.0000000

 #plotprep(width=8,height=6)
intensity <- 2   #  how many points overlapping = maximum colour
pairs(out$results[,c("LnR0","Depl","-veLL","MSY")],pch=16,
      col=rgb(1,0,0,1/intensity),font=7,font.labels = 7)
```

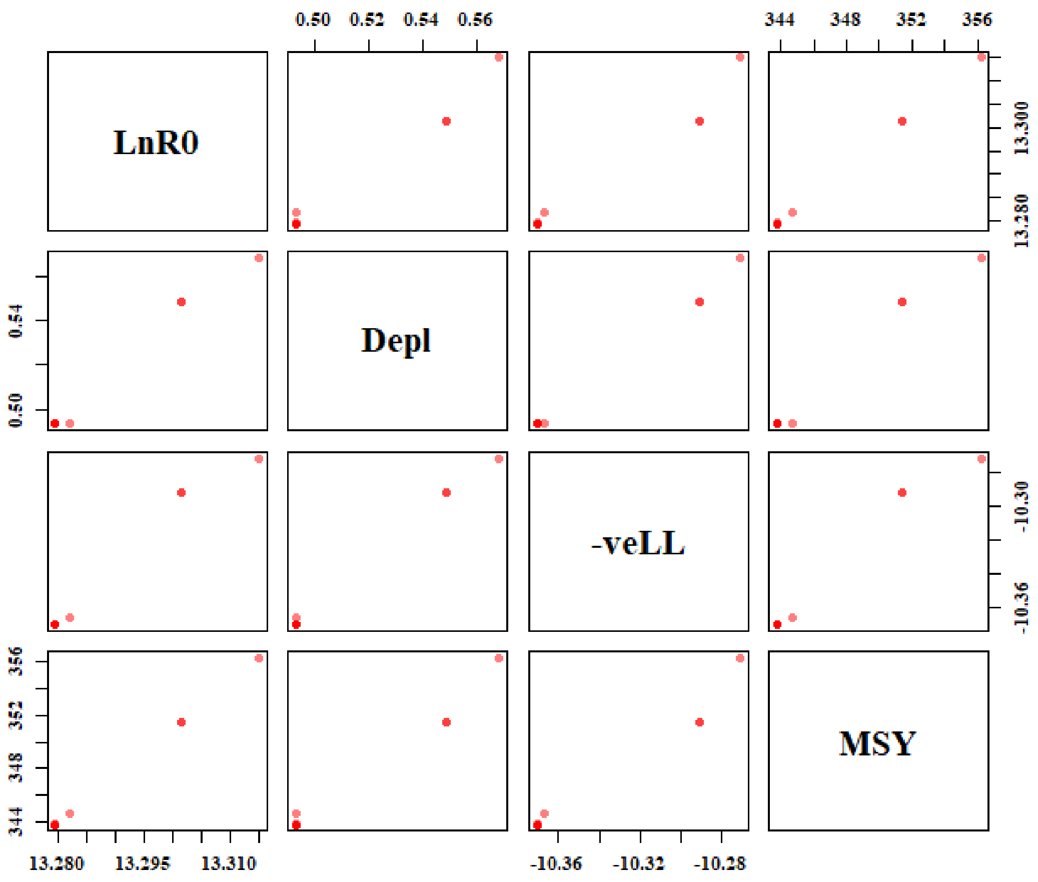
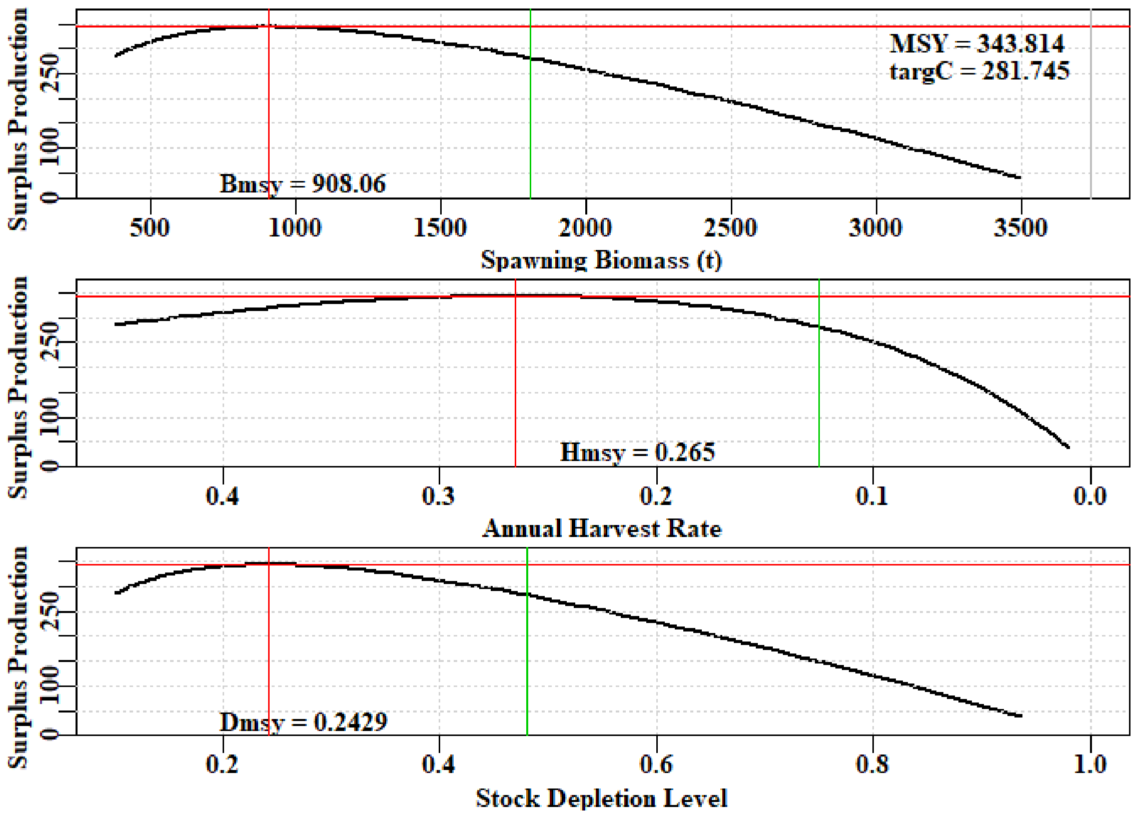


Figure 40. The correlations between outputs from repeated trials starting from different initial parameter values. Usually one would use many more trials than the example of 10, then these plots might be more informative. Histograms of these values might also indicate the variation present.

The Production Curve and Statistics

Using two runs through the *optim* function each time the median of the different trials is very similar to the optimum model fit so we will use those values to determine the production curve predicted by the model We can then use that to estimate the biomass at the target (default = $0.48B_0$) and at the limit reference point of $0.2B_0$. In addition, by estimating the yield expected at those reference points and dividing that through by the biomass at those reference points we can calculate the target and limit harvest rate reference points. The contents of *prod* can be used to determine other statistics such as the sustainable yield over the range of the current predicted depletion levels.

```
data(dataspm)
fish <- dataspm$fish
glb <- dataspm$glb
props <- dataspm$props
pars <- c(13.75,0.189667,0.6) # Fit 3 par__aspm__with penalty
bestL <- optim(pars,aspmPENLL,method="Nelder-Mead",
               infish=fish,inglb=glb,inprops=props,
               control=list(maxit = 1000, parscale = c(10,1,0.1)))
# two times through
bestL <- optim(bestL$par,aspmPENLL,method="Nelder-Mead",
               infish=fish,inglb=glb,inprops=props,
               control=list(maxit = 1000, parscale = c(10,1,0.1)))
par <- bestL$par
print(par)

## [1] 13.2794896  0.1731791  0.4934525

prod <- getProductionC(exp(par[1]),fish,glb,props,
                       Hrg=c(0.01,0.45,0.005),nyr=50)
head(round(prod,3),6)

##       Harvest   SpawnB ExploitB   Yield Depletion
## 0       0.000 3738.229 4117.963      NA     1.000
## 0.01    0.010 3503.303 3896.605  38.966     0.937
## 0.015   0.015 3393.404 3792.697  56.890     0.908
## 0.02    0.020 3288.158 3692.955  73.859     0.880
## 0.025   0.025 3187.299 3597.142  89.929     0.853
## 0.03    0.030 3090.581 3505.037 105.151     0.827

tail(round(prod,3),6)

##       Harvest  SpawnB ExploitB   Yield Depletion
## 0.425   0.425 425.115  702.960 298.758     0.114
## 0.43    0.430 414.797  688.847 296.204     0.111
## 0.435   0.435 404.686  674.933 293.596     0.108
## 0.44    0.440 394.777  661.213 290.934     0.106
## 0.445   0.445 385.066  647.683 288.219     0.103
## 0.45    0.450 375.547  634.339 285.452     0.100

anspen <- prodASPM(prod,target=0.48,console=FALSE,plot=TRUE)
```



```
round(anspen,3)

##       MSY      Bmsy      Hmsy      Dmsy        B0     targC     Htarg 
Btarg 
##   343.814   908.060     0.265     0.243  3738.229   281.745     0.125 
1809.504
```

Figure 41. Production curves for the optimum fitting three parameter age-structured production model fitted to the slope fishery data in *dataspm*. The target in this case is $0.48B_0$ designated by the vertical green lines. The results contained within *anspen* are used as labels. In this case it is suggesting that B_{MSY} is down at $0.243B_0$ so in this case using a target of $0.48B_0$ means that the harvest rate, and presumably effort, would be halved, the stock kept at a much higher presumably more resilient level, and the catch only reduced on average by about 18%.

A Phase Plot

The final part of age-structured production modelling would entail generating a phase plot of predicted biomass against the predicted harvest rates. The previous functions and analyses will provide all the information we require to feed into the function *aspmphaseplot*.

```
#   plotprep(width=7,height=5.5)
fisheryPen <- dynamics(bestL$par,infish=fish,inglb=glb,inprops=props)
outs <- aspmphaseplot(fisheryPen,prod,anspen,Blim=0.2,fnt=7)
```

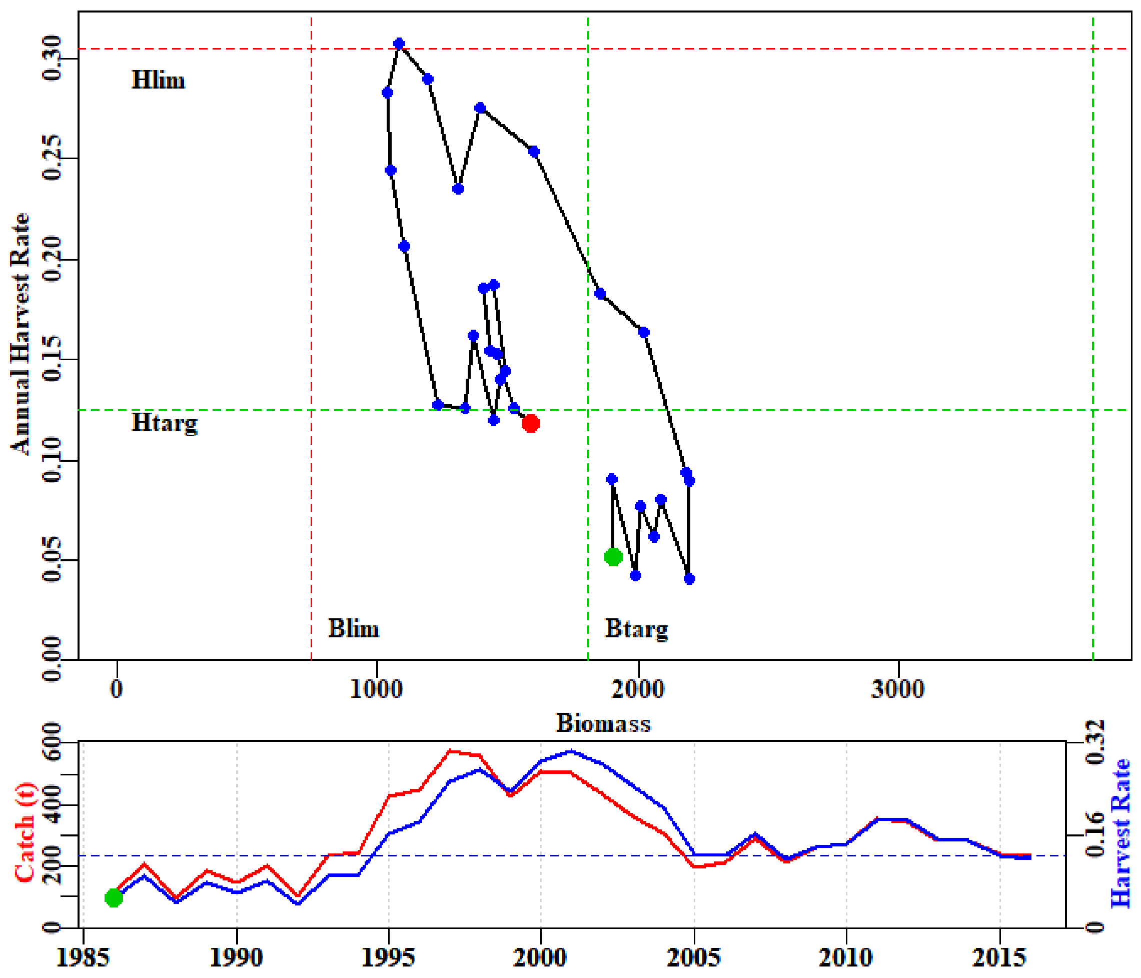


Figure 42. Phase plot of predicted biomass vs predicted harvest rate for the optimum fitting three parameter age-structured production model fitted to the slope fishery data in *dataspm*. The target in this case at $0.48B_0$ is designated by the green lines, while the limit reference points are designated by the red lines.

The phase plot (**Figure 42**) suggests that the biomass is a little below the target but the fishing mortality is very close to its target. In addition the fishery appears relatively stable at present indicating it is not declining. In the SAFS system this fishery could defensibly be claimed to be sustainable although the uncertainty in the analysis would need to be noted explicitly.

Characterization of Uncertainty

When only fitting to CPUE it is possible to use many replicate bootstrap samples followed by reanalysis to generate a detailed characterization of uncertainty. The following example code illustrates the approach. First we need to obtain the optimum solution.

```
#  library(simpleSA)
library(simpleSA)
data(dataspm)
fish <- dataspm$fish
glb <- dataspm$glb
props <- dataspm$props
pars <- c(13.5,0.18,0.5)
bestL <- fitASPM(pars,fish,glb,props,callfun=aspmPENLL)
```

```
fishery <- dynamics(bestL$par,fish,glb,props)
kable(fishery,digits=c(0,1,1,3,3,3,3,3,3))
```

	Year	Catch	PredC	SpawnB	ExploitB	FullH	CPUE	PredCE	Deplete
0	1985			1844.493	2201.515				0.493
1	1986	112.9	112.9	1905.994	2281.700	0.051	1.201	1.075	0.510
2	1987	206.3	206.3	1897.882	2279.923	0.090	1.355	1.115	0.508
3	1988	95.7	95.7	1986.162	2377.159	0.042	1.058	1.114	0.531
4	1989	183.1	183.1	2005.773	2396.363	0.077	1.085	1.161	0.537
5	1990	147.4	147.4	2059.854	2478.526	0.062	0.974	1.171	0.551
6	1991	198.9	198.9	2082.974	2518.471	0.080	1.044	1.211	0.557
7	1992	102.1	102.1	2193.351	2640.207	0.041	0.776	1.230	0.587
8	1993	235.5	235.5	2193.924	2633.084	0.089	1.053	1.290	0.587
9	1994	247.8	247.8	2180.129	2615.893	0.094	1.284	1.286	0.583
10	1995	426.8	426.8	2019.514	2445.747	0.163	1.333	1.278	0.540
11	1996	448.0	448.0	1851.779	2279.545	0.183	1.401	1.195	0.495
12	1997	577.4	577.4	1599.185	2025.283	0.253	1.469	1.114	0.428
13	1998	558.5	558.5	1389.703	1821.336	0.276	1.449	0.989	0.372
14	1999	427.9	427.9	1310.586	1759.085	0.235	1.142	0.890	0.351
15	2000	509.3	509.3	1188.966	1631.779	0.290	0.996	0.859	0.318
16	2001	502.4	502.4	1082.984	1518.972	0.308	0.882	0.797	0.290
17	2002	429.6	429.6	1038.707	1473.711	0.283	0.764	0.742	0.278
18	2003	360.2	360.2	1049.518	1485.656	0.244	0.767	0.720	0.281
19	2004	306.2	306.2	1099.426	1537.854	0.206	0.720	0.726	0.294
20	2005	195.7	195.7	1227.569	1673.737	0.127	0.600	0.751	0.328
21	2006	210.0	210.0	1336.509	1772.734	0.125	0.634	0.818	0.358
22	2007	287.3	287.3	1368.358	1784.647	0.162	0.694	0.866	0.366
23	2008	214.2	214.2	1441.816	1857.291	0.120	0.889	0.872	0.386
24	2009	260.6	260.6	1471.771	1884.847	0.140	0.864	0.907	0.394
25	2010	272.2	272.2	1489.268	1907.019	0.144	0.844	0.921	0.398
26	2011	356.9	356.9	1441.380	1859.230	0.187	0.843	0.932	0.386
27	2012	345.0	345.0	1407.928	1830.919	0.186	0.885	0.908	0.377
28	2013	282.7	282.7	1430.753	1865.664	0.154	0.996	0.894	0.383
29	2014	285.1	285.1	1457.524	1897.810	0.153	0.980	0.911	0.390
30	2015	237.8	237.8	1522.403	1968.634	0.125	0.957	0.927	0.407
31	2016	233.3	233.3	1588.181	2033.085	0.119	1.063	0.962	0.425

Having run the model through optim twice inside *fitASPM* the optimum fit is used to characterize the dynamics using *dynamics*. The basis of the bootstrap sampling is that the log-normal residuals (*CPUE*/*PredCE*) are randomly sampled with replacement with each such bootstrap then being multiplied by the optimum model's predicted CPUE. If, for example, we take the original residuals and multiply them by the original predicted CPUE we would re-generate the original observed CPUE. All we are doing in the bootstrap procedure is reordering the residuals by randomly resampling them with replacement. The 'with replacement' bit implies that some values may be omitted and others may be repeated more than once.

Such bootstrap samples are generated within *bootASPM*. This function generates replicate numbers of optimal fitting parameters in *param*, estimates of unfished biomass in *B0*, and finally a matrix of five time-series of Spawning Biomass, Fully selected harvest rate, each bootstrap CPUE series, the optimum predicted CPUE, and the depletion level through time. Here we are only running 100 replicates so as to speed the process, but in a real analysis one might use at least 1000 replicates

```
reps <- 20
starttime <- Sys.time()
answer <- bootASPM(fish,glb,props,bestL$par,iter=reps)

## 20

Sys.time() - starttime

## Time difference of 13.00663 secs

str(answer,max.level=1)

## List of 3
##  $ result: num [1:20, 1:32, 1:5] 1844 2330 1607 1883 2442 ...
##   ..- attr(*, "dimnames")=List of 3
##  $ B0    : num [1:20] 3738 4004 3741 3816 3973 ...
##  $ param : num [1:20, 1:3] 13.3 13.3 13.3 13.3 13.3 ...
##   ..- attr(*, "dimnames")=List of 2
```

Once the bootstraps are completed there are multiple ways of displaying such information. Initially one can generate classical percentile confidence intervals from the bootstrap replicates (Haddon, 2011).

```
yrs <- fishery[,"Year"]
nyrs <- length(yrs)
par(mfrow=c(2,2),mai=c(0.45,0.45,0.05,0.05))
par(cex=0.85, mgp=c(1.35,0.35,0), font.axis=7,font=7,font.lab=7)
label <- names(answer$result[1,1,])
label <- label[-3]  # remove CPUE
numvar <- length(label)
bootvar <- answer$result[,nyrs,label[1]]
for (i in 1:numvar) { # i=3
   bootvar <- answer$result[,nyrs,label[i]]
   quantCI <- quantile(bootvar,probs=c(0.05,0.5,0.95),na.rm=TRUE)
   hist(bootvar,breaks=30,main="",xlab=label[i],col="red")
   abline(v=quantCI,col=c(4,4,4),lwd=c(1,2,1))
}
```

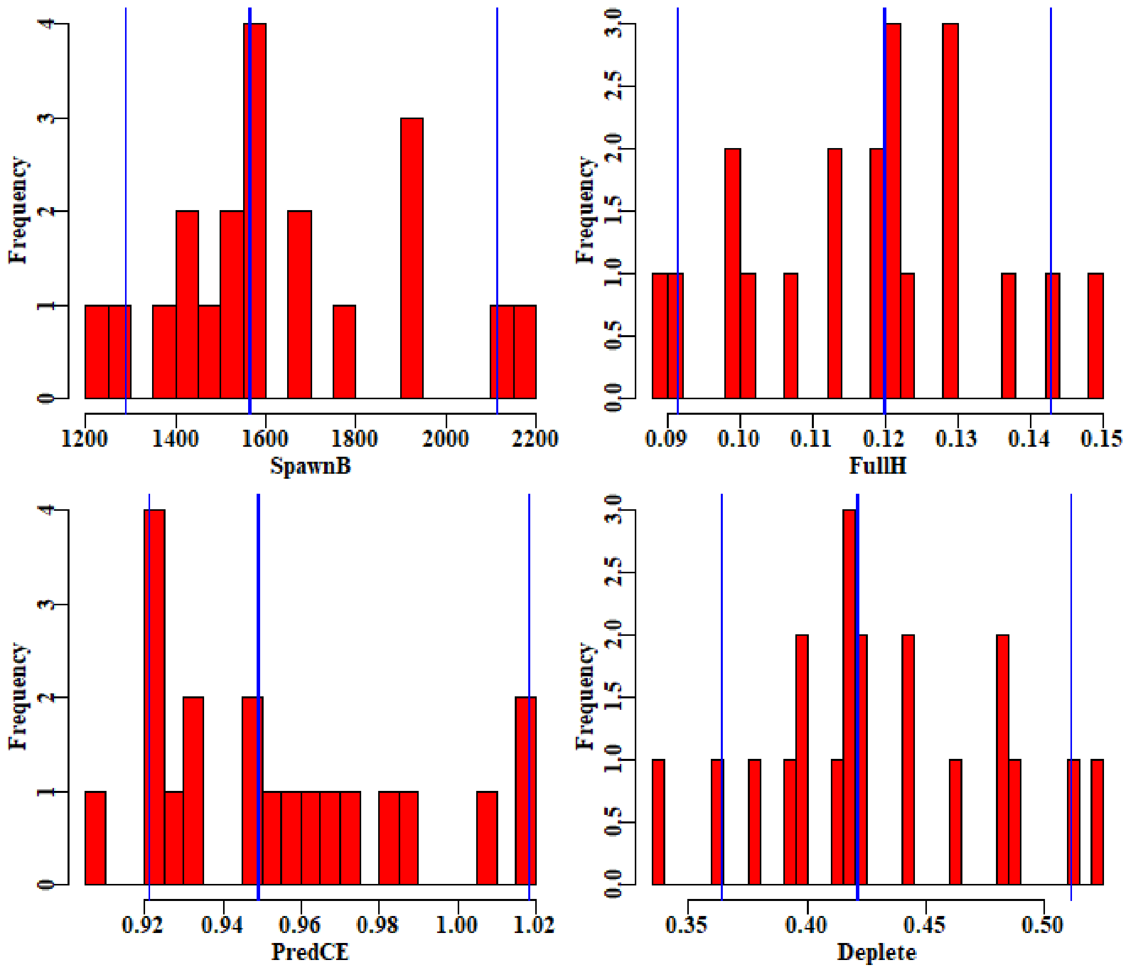


Figure 43. Histograms of the final years' spawning biomass, fully selected harvest rates, predicted CPUE, and the stock depletion level. Of course 20 replicates is completely inadequate but each bootstrap replicate can take a significant time (note the time taken in the example). One thing that can be noted is the asymmetrical percentile confidence bounds.

With only 20 replicates no conclusions can be drawn but the plots still illustrate the principle behind the bootstraps. The percentile confidence intervals can illustrate the uncertainty in the assessments and the potential risk of falling below limit reference points.

```
pickvar <- "Deplete"
bootvar <- answer$result[,,pickvar]
yrs <- as.numeric(colnames(bootvar))
nyrs <- length(yrs)
quantCI <- t(apply(bootvar,2,quants))
kable(quantCI,digits=c(3,3,3,3,3,3))
```

	2.5%	5%	50%	90%	95%	97.5%
1985	0.302	0.319	0.497	0.650	0.670	0.702
1986	0.328	0.346	0.513	0.652	0.672	0.700
1987	0.339	0.357	0.511	0.637	0.657	0.681
1988	0.375	0.393	0.534	0.648	0.666	0.687
1989	0.392	0.410	0.540	0.643	0.659	0.678
1990	0.417	0.435	0.554	0.648	0.663	0.679
1991	0.431	0.449	0.560	0.646	0.661	0.676
1992	0.469	0.486	0.589	0.668	0.682	0.695
1993	0.476	0.493	0.590	0.662	0.676	0.688
1994	0.480	0.496	0.586	0.654	0.668	0.679
1995	0.443	0.460	0.541	0.609	0.625	0.635
1996	0.404	0.421	0.495	0.563	0.581	0.590
1997	0.342	0.360	0.427	0.495	0.516	0.525
1998	0.291	0.309	0.370	0.439	0.462	0.470
1999	0.274	0.292	0.349	0.417	0.440	0.449
2000	0.246	0.264	0.316	0.384	0.409	0.417
2001	0.220	0.239	0.288	0.355	0.382	0.389
2002	0.209	0.228	0.276	0.343	0.371	0.378
2003	0.212	0.229	0.279	0.346	0.374	0.381
2004	0.224	0.241	0.292	0.360	0.387	0.395
2005	0.257	0.274	0.326	0.393	0.421	0.428
2006	0.285	0.303	0.355	0.422	0.449	0.456
2007	0.292	0.309	0.364	0.431	0.458	0.465
2008	0.310	0.327	0.383	0.451	0.478	0.484
2009	0.316	0.334	0.391	0.460	0.486	0.492
2010	0.320	0.337	0.396	0.465	0.491	0.497
2011	0.307	0.323	0.383	0.453	0.479	0.485
2012	0.298	0.314	0.374	0.445	0.470	0.476
2013	0.305	0.320	0.380	0.450	0.475	0.481
2014	0.314	0.327	0.387	0.456	0.481	0.487
2015	0.332	0.345	0.405	0.472	0.496	0.502
2016	0.352	0.364	0.422	0.488	0.512	0.517

```
ymax <- getmaxy(bootvar)
par(mfrow=c(1,1),mai=c(0.45,0.45,0.05,0.05))
par(cex=0.85, mgp=c(1.35,0.35,0), font.axis=7,font=7,font.lab=7)
plot(yrs,bootvar[1,],type="n",lwd=1,col=0,ylim=c(0,ymax),
     panel.first = grid(),xlab="",ylab=pickvar)
for (i in 1:reps) lines(yrs,bootvar[i,],lwd=1,col="grey")
lines(yrs,quantCI[,"50%"],lwd=2,col="red")
arrows(x0=yrs,y0=quantCI[,"5%"],y1=quantCI[,"95%"],
       col=2,lwd=1,length=0.035,angle=90,code=3)
```

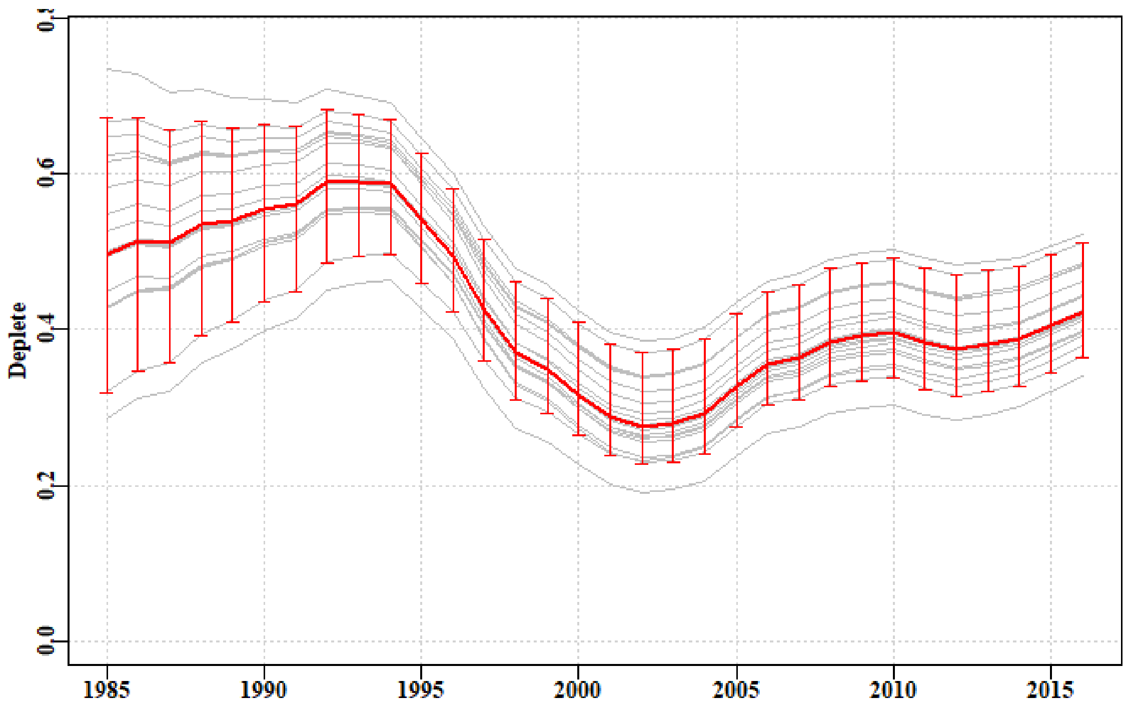


Figure 44. The bootstrapped trajectories of stock depletion of the dataspm data set. Note that 20 replicates are far too few to provide sensible or valid percentile confidence intervals.

The output from the *bootASPM* function includes the bootstrap optimum parameters. These can be used along with the *fish*, *glb*, and *props* objects from the data set used to generate productivity curves and determine target catches, MSY, and other fishery outputs for each set of parameters. This means that percentile confidence intervals can be generated for such assessment outputs.

Management Advice with aspm

If working with a species that requires on-going management then it is necessary to produce advice with respect to acceptable catches that will lead to a sustainable fishery or whatever other management goal is in place for the fishery. To generate such advice formal harvest strategies are required to allow the outputs from the assessment to be converted into a recommended biological catch. This may then be modified by fishery managers taking into account potential rates of change within a fishery or social or economic drivers of management decisions. It was possible to put forward suggestions for new harvest strategies using the catch-MSY method because none were available previously and that put forward was only a suggestion for a possible consideration. Putting forward a proposed harvest control rule for the spm approach without consultation with jurisdictional fisheries managers could produce suggestions incompatible with a particular jurisdictions objectives. There are harvest control rules that can be used once limit and target reference points are agreed upon and these can be utilized where considered appropriate. The SAFS process, however, does not currently require a target reference point even though most harvest control rules do require one.

Appendix: Age-Structured Production Model Equations

Initiation of an Age-Structured Model

At equilibrium, in an un-exploited population, the age-structure is assumed to be the result of natural mortality acting alone upon constant average unfished levels of recruitment. The equilibrium result would be a stable age distribution determined by those constant average recruitments and natural mortality. At the start of a time series, let us say in year 1, this is defined as:

$$N_{a,1} = \begin{cases} N_{0,1} = R_0 & a = 0 \\ N_{a-1,1}e^{-M} & 1 \leq a < a_x \\ N_{a_x-1,1}e^{-M}/(1-e^{-M}) & a = a_x \end{cases}$$

where $N_{a,1}$ is the numbers of age a, in year 1, a_x is the maximum age modelled (the plus-group), and M is the instantaneous rate of natural mortality. In a pre-exploitation population there is no fishing mortality and the final component the above equation (where a = a_x), is referred to as the plus group because it is the series which combines ages a_x and all older ages that are not modelled explicitly. This requires the inclusion of the (1− e^{-M}) divisor to force the equation to be the sum of an exponential series. The $N_{0,1}$ is the constant unfished recruitment level, R_0. Sometimes this also has an e^{-M} term, depending on the timing of spawning. If the natural mortality term is included then the estimated R_0 value will be somewhat higher than if it is omitted (by $1/e^{-M}$), so it is usually simpler to omit it. This stable age distribution can also be obtained by first calculating the numbers-at-age for a recruitment of 1, or the numbers-at-age per recruit, and then multiplying that vectors of numbers by R_0., which is how it is implemented in *simpleSA::dynamics*

Biological Characteristics

Length-at-age of fish is defined by the von Bertalanffy growth function:

$$L_a = L_\infty(1 - e^{-k(a-t_0)})$$

where L_a is the mean length at age a, L_∞ is the asymptotic average maximum length, k is the grow rate coefficient, and t_0 is the length at age zero.

The mass-at-age relationship is defined as:

$$w_a = W_{aa}L^{W_{ab}}$$

where w_a is the mass at age a, and W_{aa} and W_{ab} are the coefficients that define the power relationship between length and mass.

Spawning Stock Recruitment Relationship

The biomass A_0 can be defined as the mature stock biomass that would develop given a constant recruitment level of one (i.e. $N_{0,1} = 1$ in the above equation). Thus, at a biomass of A_0, distributed across a stable age distribution, the resulting average recruitment level would be $R_0 = 1$. A_0 acts as a scaling factor in the recruitment equations by providing the link between R_0 and B_0

$$A_0 = \sum_{a=1}^{a_x} n_{a,1}\, m_a w_a$$

where m_i is the proportion mature at age a, $n_{a,1}$ is the virgin number of animals per recruit of age a in year 1, and w_a is the weight of an animal of age a. The average unfished recruitment level, R_0, is directly related to the virgin mature, or recruited, biomass, B_0

$$R_0 = B_0/A_0$$

By determining A_0, from a constant recruitment level of one, the recruitment levels from realistic B_0 levels can be obtained by applying the above equation. Once R_0 has been determined the unfished number at age distribution can be obtained by substituting R_0 into the first equation. The spawning stock – recruitment relationship can be described by the deterministic form of the Beverton – Holt relationship:

$$R_{y+1} = \frac{aB_y^{Sp}}{b + B_y^{Sp}}$$

where B_y^{Sp} is the mature, or spawning biomass in the in year y.

A re-parameterization of the Beverton-Holt parameters in terms of steepness, h, and B_0 is to specify a and b such that:

$$a = \frac{4hR_0}{5h-1} \qquad \text{and} \qquad b = \frac{B_0(1-h)}{5h-1}$$

Using this re-parameterization the the number of recruits produced in year y from the spawning biomass in year $y-1$ is:

$$N_{0,y} = \frac{4hR_0B_{y-1}^{Sp}}{(1-h)B_0 + (5h-1)B_{y-1}^{Sp}}.$$

Stock dynamics

To describe the dynamics subsequent to population initiation (i.e. the generation of $N_{a,y}$, the number at age a in year y, for years other than 0), requires the inclusion of the stock recruitment relationship and the impact of fishing mortality. Not all age classes are necessarily fully selected, thus the fishing mortality term must be multiplied by the selectivity associated with the fishing gear for age a, s_a, described by a logistic curve:

$$s_a = \frac{1}{\left(1 + e^{\left(\frac{a-a_{50}}{\delta}\right)}\right)}$$

where a_{50} is the age at which 50% of individuals are selected by the fishing gear, and δ is a parameter that determines the width or steepness of the selectivity ogive. Such logistic curves are also used to describe the development of maturity within he population but in such a case the a_{50} refers to the age at 50% maturity.

A term is also needed for the recruitment in each year (stock-recruit relationship above), and this is assumed to be a function of the spawning biomass of the stock at the end of the previous year y, B_y^{Sp}.

The spawning biomass for a year y is:

$$B_y^{Sp} = \sum_{a=0}^{a_x} w_a\, m_a N_{a,y}$$

If this is applied to the unfished stable age distribution this would provide an estimate of the unfished spawning biomass-per-recruit. When using difference equations (rather than continuous differential equations) the dynamics of the fishery, in terms of the order in which growth, natural, and fishing mortality occur, are important when defining how the numbers at age change. If the transition of numbers at age in year y into numbers at age in year $y + 1$ is made in a number of steps this simplifies the calculation of internally consistent estimates of exploitable biomass, catch rates, and harvest rates. If it is assumed that the dynamics of a population entails that fish first grow from year $y - 1$ to year y, then undergo half of natural mortality before they are fished and only then undergo the final half of natural mortality this would imply two steps to define the transition from one year to the next. The first step entails recruitment, growth from each age class to the next, and the application of the effect of half of natural mortality:

$$N_{a,y^*} = \begin{cases} N_{0,y} & a = 0 \\ N_{a-1,y-1} e^{-M/2} & 1 \le a < a_x - 1 \\ \left(N_{a_x-1,y-1} + N_{a_x,y-1}\right) e^{-M/2} & a = a_x \end{cases}$$

where $N_{0,y}$ is defined by the stock - recruit relationship, ages 1 to a_x-1 are modelled by adding 1.0 to the previous year's ages 0 to a_x – 2 and imposing the survivorship from half the natural mortality, and the plus group (a_x) is modelled by adding 1.0 to the previous year's age a_x - 1 and adding those to the numbers in the previous year's age a_x and then applying the survivorship from half the natural mortality. The above equation thus leads to the mid-year exploitable biomass (mid-year being the reason for the $e^{-M/2}$) in year y being defined as:

$$B_y^E = \sum_{a=0}^{a_x} w_a\, s_a N_{a,y^*}$$

The dynamics within any year are completed by the application of the survivorship following fishing mortality across all ages (expressed as an annual harvest rate), followed by the survivorship following the remainder of natural mortality. Natural mortality is not applied directly to the new recruits until they grow into the next year:

$$N_{a,y} = \begin{cases} N_{0,y^*} & a = 0 \\ N_{a,y^*}\left(1 - s_a \hat{H}_y\right) e^{-M/2} & 1 \le a \le a_x \end{cases}$$

In the above equation, the $N_{a,y}$ refer the numbers in age a at the end of year y (i.e. after all the dynamics have occurred). The predicted harvest rate, ., given an observed or recommended catch level in year y, C_y, is estimated as

$$\hat{H}_y = \frac{C_y}{B_y^E}$$

where B_y^E is defined above. The catch at age, in numbers, is therefore defined by:

$$C_{a,y}^N = N_{a,y^*} s_a \hat{H}_y$$

and the total catch by mass is the sum of the separate catches at age multiplied by their respective average weights for all ages:

$$C_y = \sum_{a=0}^{a_x} w_a \, C_{a,y}^N$$

Predicted catch rates also derive from the exploitable biomass and the average catchability coefficient, q:

$$I_y = qB_y^E.$$

Likelihoods

Maximum likelihood methods, as the name dictates entail maximizing the likelihood of the available data given the model and a proposed set of parameters. Very often the likelihoods involved when fitting models are very small numbers. To avoid rounding errors (even when using 64 bit computers) it is standard to use log-likelihoods rather than likelihoods (in that way the log-likelihoods can be individually added together rather than multiply the individual likelihoods). Additionally, rather than maximizing a log-likelihood, minimization often best matches our intuitions about model fitting and this involves minimizing the negative log-likelihood. The full log-normal negative log likelihood for the **aspm** is similar to that used for the **spm** but with a few parameter changes and it estimates the $\hat{\sigma}_I$ directly rather than using a closed form:

$$-veLL\left(data|R_0, \hat{\sigma}_I\right) = -\sum_t Ln\left[\frac{1}{I_t\sqrt{2\pi}\,\hat{\sigma}_I} e^{\frac{-\left(LnI_t - Ln\hat{I}_t\right)}{2\hat{\sigma}_I^2}}\right]$$

References

Dick, E.J. and A.D. MacCall (2011) Depletion-based stock reduction analysis: a catch-based method for determining sustainable yields for data-poor fish stocks. *Fisheries Research* **110**(2): 331-341

Haddon, M. (2014) Tier 4 analyses in the SESSF, including deep water species. Data from 1986 – 2012. Pp 352 – 461 in Tuck, G.N. (ed) (2014) *Stock Assessment for the Southern and Eastern Scalefish and Shark Fishery 2013. Part 2.* Australian Fisheries Management Authority and CSIRO Marine and Atmospheric Research, Hobart. 313p.

Haddon, M., Klaer, N., Wayte, S., and G. Tuck (2015) *Options for Tier 5 approaches in the SESSF and identification of when data support for harvest strategies are inappropriate.* CSIRO. FRDC Final Report 2013/200. Hobart. 115p.

Kimura, D.K. and J.V. Tagart (1982) Stock Reduction Analysis, another solution to the catch equations. *Canadian Journal of Fisheries and Aquatic Sciences* **39**: 1467 - 1472.

Kimura, D.K., Balsiger, J.W., and D.H. Ito (1984) Generalized stock reduction analysis. *Canadian Journal of Fisheries and Aquatic Sciences* **41**: 1325–1333.

Little, L.R., Wayte, S.E., Tuck, G.N., Smith, A.D.M., Klaer, N., Haddon, M., Punt, A.E., Thomson, R., Day, J. and M. Fuller (2011) Development and evaluation of a cpue-based harvest control rule for the southern and eastern scalefish and shark fishery of Australia. *ICES Journal of Marine Science* **68**(8): 1699-1705.

Martell, S. and R. Froese (2013) A simple method for estimating MSY from catch and resilience. *Fish and Fisheries* **14**: 504-514

Polacheck, T., Hilborn, R. and A.E. Punt (1993) Fitting surplus production models: Comparing methods and measuring uncertainty. *Canadian Journal of Fisheries and Aquatic Sciences* **50**: 2597–2607.

Punt, A.E., Butterworth, D.S. and A.J. Penney (1995) Stock assessment and risk analysis for the South Atlantic population of albacore *Thunnus alalunga* using an age-structured production model *South African Journal of Marine Science* **16**: 287-310. http://dx.doi.org/10.2989/025776195784156476

R Core Team (2017). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/. see also https://cran.r-project.org/

RStudio (2016) www.rstudio.com

Rudd, M.B. and T.A. Branch (2017) Does unreported catch lead to over-fishing? *Fish and Fisheries* **18**: 313-323.

Schaefer, M.B. (1954) Some aspects of the dynamics of populations important to the management of the commercial marine fisheries. *Bulletin, Inter-American Tropical Tuna Commission*, **1**: 25-56.

Schaefer, M.B. (1957) A study of the dynamics of the fishery for yellowfin tuna in the Eastern Tropical Pacific Ocean. *Bulletin, Inter-American Tropical Tuna Commission*, **2**: 247-285

Venzon, D.J. and S.H. Moolgavkar (1988) A method for computing profile-likelihood-based confidence intervals. *Applied Statistics*, **37**: 87-94.

Walters, C.J., Martell, S.J.D. and J. Korman (2006) A stochastic approach to stock reduction analysis. *Canadian Journal of Fisheries and Aquatic Sciences* **63**: 212 - 223.