Introduction to Catch-MSY

Owen Liu

December 14, 2016

Introduction to Catch-MSY

Catch-MSY is a data limited assessment method for fisheries data, suggested by Martell and Froese (2013)*. It utilizes a time series of catch data, along with simple relationships and assumptions about resilience to estimate maximum sustainable yield (MSY), biomass at MSY (B_{msy}), and fishing mortality rate that produces MSY (F_{msy}). The entire method, with examples, is described in Martell and Foese (2013). It has also been incorporated into an R package called fishmethods, which we'll use here.

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

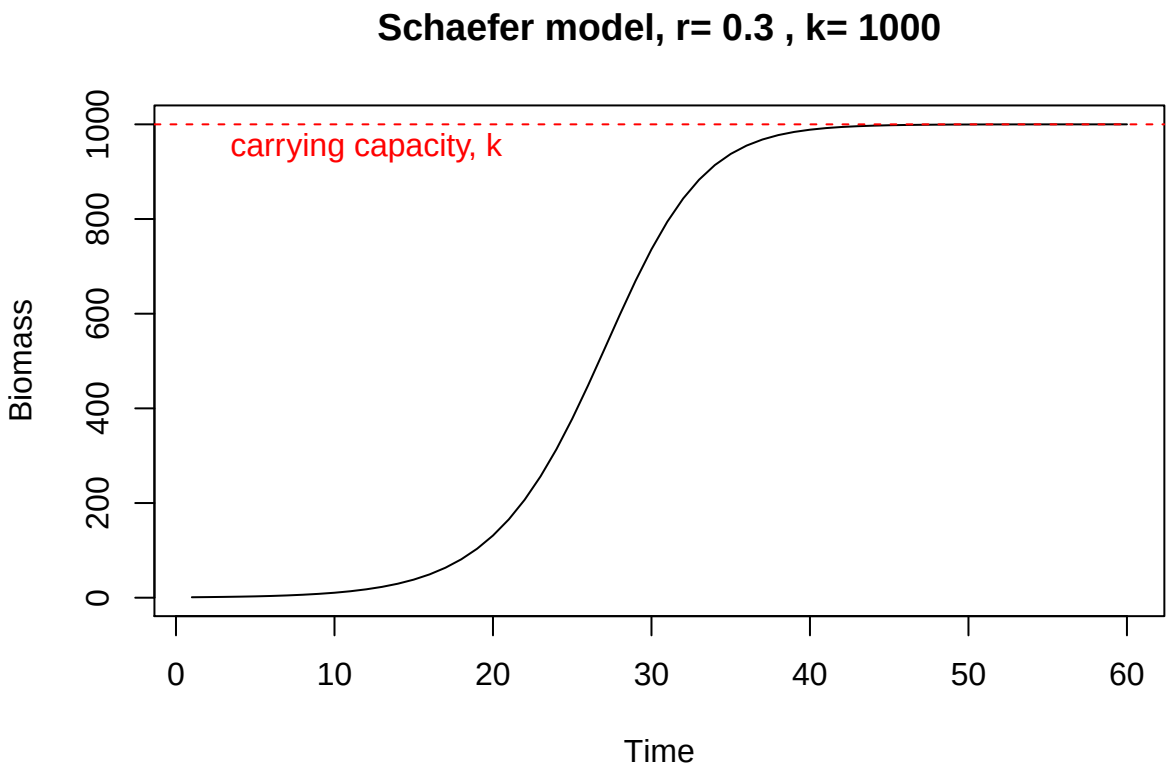
Schaefer model and MSY

The Catch-MSY method is based on a Schaefer production model, which is basically a logistic population growth model that includes a term for fishing. The Schaefer model is:

$$B_{t+1} = B_t + rB_t(1 - B_t/k) - c_t$$

where B_{t+1} is biomass in year $t+1$, r is the population intrinsic growth rate, k is the carrying capacity, and c_t is the catch in year t. In a very basic sense, this is just saying that the biomass of the stock next year is the biomass this year, plus some growth (which is dependent on how close the stock is to its carrying capacity), minus catch. The assumed growth of the fish stock from a low level looks like this (assuming some arbitrary values of 0.3 for the growth rate r and 1000 for the carrying capacity k):

```
schaefer <- function(r = 0.3, k = 1000, B0 = 1, ts = 60) {
    out <- numeric(ts)
    out[1] <- B0
    for (i in 1:(ts - 1)) {
        out[i + 1] <- out[i] + r * out[i] * (1 - out[i]/k)
    }

    # Produce a plot of time vs. biomass
    plot(1:ts, out, type = "l", xlab = "Time", ylab = "Biomass",
        main = paste("Schaefer model, r=", r, ", k=", k))

    # Add label for carrying capacity
    abline(h = k, lty = 2, col = "red")
    text(0.2 * ts, 0.95 * k, col = "red", "carrying capacity, k")
    return(out)
}

schaefer_example <- schaefer()
```



One can also add process error to the model (in this case, lognormal error),

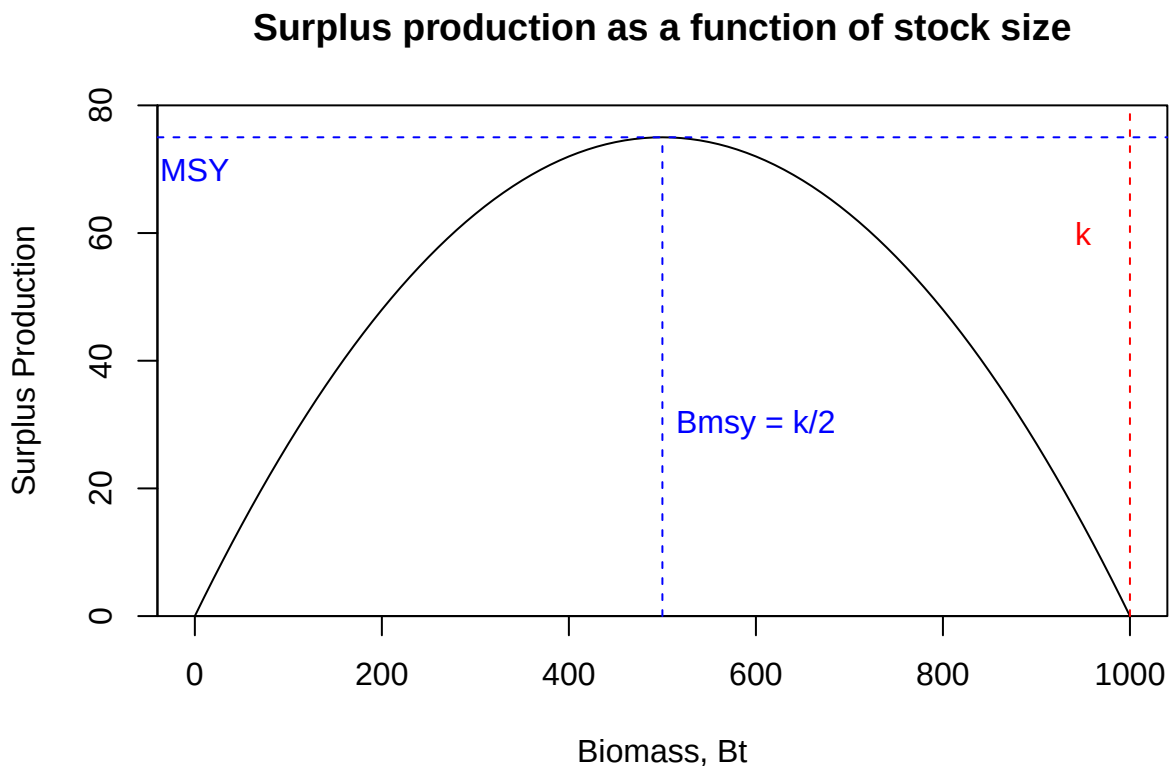
$$B_{t+1} = [B_t + rB_t(1 - B_t/k) - c_t]exp(\nu_t)$$

where $\nu \sim$ normal $(0, \sigma_\nu)$.

In the Schaefer model, the "surplus production" part of the model is equal to $rB_t(1 - B_t/k)$, and is basically the growth of the stock in the year (if catch were equal to zero). We could catch all of this surplus production (theoretically) without reducing the size of the stock if we set $c_t = rB_t(1 - B_t/k)$. Without going into the math, you can see very simply that this surplus production (and therefore, the amount that we can sustainably catch), depends on r, k, and B_t. Looking at it graphically as a function of B_t with r and k held constant, you can see clearly that surplus production has a maximum at half its carrying capacity:

```
r = 0.3
k = 1000
biomass <- seq(0, 1000, by = 10)
surplus_prod <- r * biomass * (1 - biomass/k)

# Plot
plot(biomass, surplus_prod, main = "Surplus production as a function of stock size",
    xlab = "Biomass, Bt", ylab = "Surplus Production", type = "l",
    yaxs = "i", ylim = c(0, 80))

# Line to indicate carrying capacity k
abline(v = k, lty = 2, col = "red")
text(950, 60, "k", col = "red")
```

```
# Line to show Bmsy
segments(x0 = k/2, y0 = 0, x1 = k/2, y1 = max(surplus_prod),
    col = "blue", lty = 2)
text(600, 30, "Bmsy = k/2", col = "blue")

# Line to show MSY
abline(h = max(surplus_prod), col = "blue", lty = 2)
text(0, 70, "MSY", col = "blue")
```



Logically, this happens because in the Schaefer/logistic model, the annual growth of the stock is maximized when it is at half of its carrying capacity. In other words, the steepest part of the curve in the first figure above. This is how maximum sustainable yield is defined. A little math can show that:

$B_{msy} = k/2$

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

$F_{msy} = r/2$

Catch-MSY procedure

Because r and k are not normally known for a data-limited fish stock, the Catch-MSY method uses *prior distributions* of user-provided possible values of r and k to estimate distributions of the outputs of interest, based on the catch history. In other words, the algorithm chooses a random (but potentially plausible) value for r and k, asks if the observed catches are possible with those values of r and k, and, if so, calculates MSY and the other management quantities of interest from that pair of r and k. By doing this many, many times,

trying different values of r and k, what results is ranges of possible r and k values according to the data, and, most usefully, a range of possible MSY values.

The Catch-MSY method needs three things from the user, with one possible extra piece of information:

- Prior estimates of r and k, defined as ranges of possible values.
- Estimates of relative depletion in the first and last years of the catch time series, l_0 and l_t. Relative depletion, B/k is between 0 and 1, and is the user's best guess as to how depleted the stock is at the beginning and end of the observation period. These are defined as ranges. For example, l_0 could have a lower bound of 0.5, and an upper bound of 0.9, meaning that we think that the biomass at the beginning of the time series is between 50 and 90% of carrying capacity. Similarly, we could set bounds for l_t, the relative depletion at the end of the time series, of 0.3 and 0.7, meaning that we think that the biomass at the end of the time series is between 30 and 70% of carrying capacity.
- A time series of catches. This is used to test the possible values of r and k against the possible values of depletion by running simulations.

For an exact description of the steps, see Martell and Froese (2013), but basically, the procedure goes as follows:

1. **Define the lower and upper bounds for depletion at the beginning and end of the time series, l_0 and l_t**
2. **Choose values for r and k randomly, based on the user's defined ranges**
3. **Update the population for next year, using the Schaefer model, with the chosen values of r and k and using an initial biomass of $l_0 k$**
4. **If the calculated value of B_{t+1}/k falls within the range of l_t, assign that combination of r and k a likelihood of 1; otherwise, assign it a likelihood of 0**
5. **Repeat (2) through (4) many, many times, trying different values of r and k and keeping those that are plausiable given the data**
6. **For all of the parameters sets that were assigned 1, calculate MSY, B_{msy} and F_{msy}, and plot their distributions**

Choosing parameter ranges

So, how do you choose which values of r and k to try, and which ranges of l_0 and l_t to define? These are tough questions, but the simple answer is that they should be based on the user's best judgement. Martell and Froese (2013) recommend basing values of r on resilience metrics found in FishBase. If, for example, a species is "highly" resilient, a good prior assumption of r might be between 0.6 and 1.5. These can obviously be adjusted for different applications of Catch-MSY to see how sensitive the output is to which values are chosen.

For estimating k, the carrying capacity, Martell and Froese used a uniform distribution between the maximum catch in the time series, and 100 times the maximum catch in the time series. However, they emphasize that the best available local knowledge should be used to make this prior assumption more realistic.

Similarly, for depletion ranges l_0 and l_t, Martell and Froese used ranges based on examination of the catches relative to the maximum catch in the time series (see Table 1 in Martell and Froese (2013)). If, for example, catch was low relative to maximum catch at the beginning of the time series, it is logical to think that the stock was starting at a relatively lightly-depleted point, and therefore an estimate of l_0 might be between 0.5 and 0.9. Similar rational arguments could be used to define bounds for l_t.

Note: these are all user-defined quantities, based on that user's best judgement. But the user isn't stuck with one assumption. In fact, it might be best to examine a number of different possible assumptions about r, k, and depletion, and see what results they produce

---

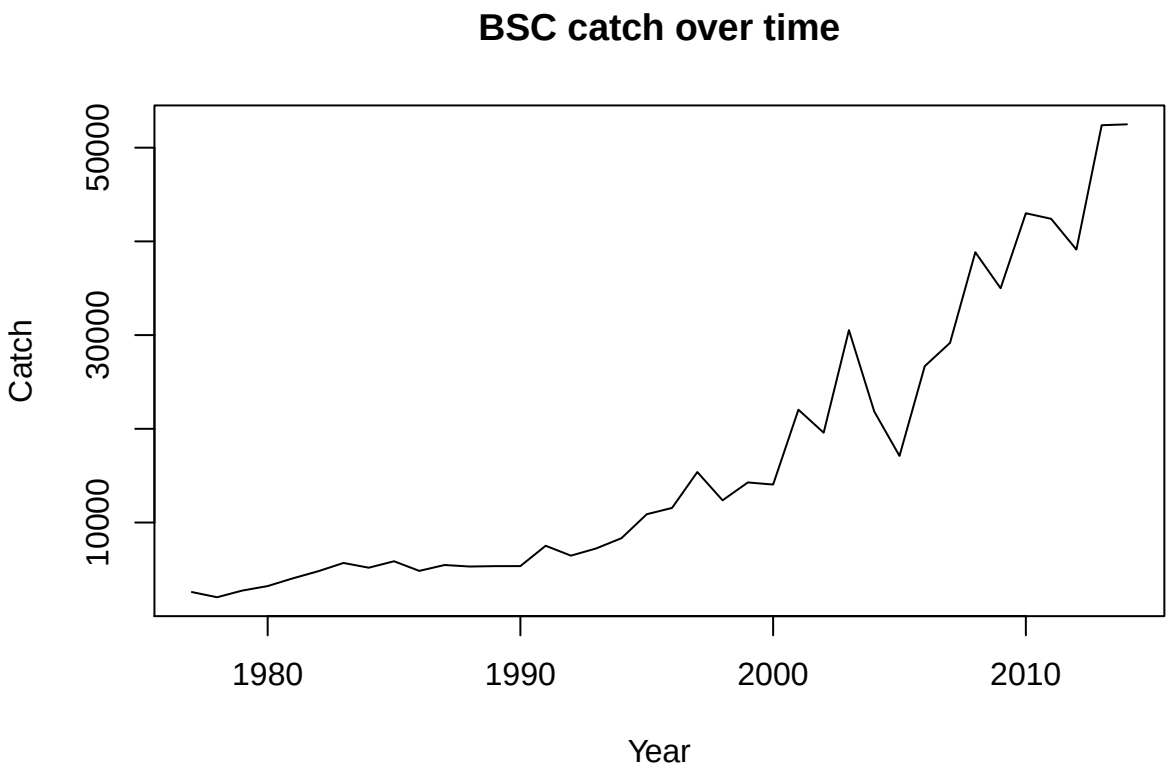
Example: Applying Catch-MSY to BSC

Now that we have an idea of the way the method works, let's apply it to some data from blue swimming crab.

```
# our working directory is where the .csv of catches is
# stored
WD <- getwd()

# fishmethods library. If you don't have this installed,
# download it by running the command
# install.packages('fishmethods') Then load it using:

library(fishmethods)

# Load the time series of catches
catches <- read.csv(paste0(WD, "/BSC_catchdata.csv"))

# quick look at the structure of the data. The first column
# is year, and the second column is catch
head(catches)
```

```
##   year catch effort     cpue
## 1 1977  2583  10819 0.238747
## 2 1978  2028   9564 0.212045
## 3 1979  2741   7405 0.370155
## 4 1980  3218   8947 0.359674
## 5 1981  4046   7935 0.509893
## 6 1982  4802   6610 0.726475
```

```
# And a graph of the time series
plot(catches$year, catches$catch, main = "BSC catch over time",
    xlab = "Year", ylab = "Catch", type = "l")
```



The fishmethods package is very large, and we're only using one particular function in it, called catchmsy(). Let's look at the important parts of the function so we understand what we're telling it to do. You can always read about the inputs and methods of an R function by typing a ? and then the function name:

```
# ?catchmsy
```

In this help dialogue, you can see descriptions of the important input parameters. Let's set them for our BSC example.

```
# the year argument should be a vector containing the time
# series of numeric year labels. We have this in our data
year <- catches$year

# similarly, catch is the time series of catches
catch <- catches$catch

# catchCV and catargs are used if we want to randomly
# resample the catch time series (bootstrapping). We won't
# use these for now

# l0 and lt: a list of argments for initial and final
# depletion levels, including the lower and upper bounds
# described above.  Since the BSC fishery seems to be a
# relatively young (lightly depleted) fishery, we choose
# ranges that are not too close to zero.  by default, the
# algorithm chooses a value of l0 and lt uniformly from these
# ranges. Set step=0 to use this option. Otherwise, you can
```

```
# explicitly set the discrete steps between the lower and
# upper bounds for the algorithm to check. For lt, we also
# need a reference year, which is the year of catch data we
# use to assess the validity of each r,k pair

l0 <- list(low = 0.5, high = 0.9, step = 0)
lt <- list(low = 0.3, high = 0.8, step = 0, refyr = 2014)

# sigv defines process error (see the Schaefer model with
# lognormal error above). For now, we'll just choose zero (no
# error, deterministic model)
sigv = 0

# k is a list of the possible values of k, and the
# statistical distribution from which to draw them. By
# default, the values of r and k are drawn from a uniform
# distribution. Following Martell and Froese (for now), we
# choose values between the maximum catch in the time series,
# and 100 times the maximum catch.
k <- list(dist = "unif", low = max(catch), up = 3e+06)

# for r, since BSC is an invertebrate it is not on FishBase.
# However, since invertebrates can be quite productive, we
# choose a relatively resilient (high) range of possible r
# values
r <- list(dist = "unif", low = 0.6, up = 1.5)

# We can leave all the other values at their defaults, for
# now
```

Now we can run the model! This function actually stores the output, not in R's memory, but as a physical .csv file in our working directory. We can set the working directory manually so we know where the output data will end up.

```
# set output directory
setwd(paste0(WD, "/catchmsy output"))

# Run the assessment, using the parameters we defined above.
# This will take awhile, and we can time it.  start the clock
ptm <- proc.time()

# BSC_catchMSY <-
# catchmsy(year=year,catch=catch,l0=l0,lt=lt,sigv=0,k=k,r=r)
# (not run)

# stop the clock
proc.time() - ptm
```

```
##    user  system elapsed
##       0       0       0
```