

Importing results from a statistical catch-at-age model

Arni Magnusson

June 20, 2017

Introduction

This is a brief description of how to import statistical catch-at-age model output from any stock assessment into **scape**. The user is expected to be familiar with basic R concepts, such as reading data from text files and manipulating lists and data frames.

A general introduction to the **scape** and **plotMCMC** packages is in the vignette *R goes fishing*.

First, take a look at how the example assessments were imported using `importCol()`, which is custom-made for Coleraine output files:

```
x.cod <- importCol("c:/scape/data/cod.res", Dev=T, Survey=T, CAc=T, CAs=T)
x.oreo <- importCol("c:/scape/data/oreo.res", CPUE=T, Survey=T, CLc=T, CLs=T, LA=T)
x.sbw <- importCol("c:/scape/data/sbw.res", Dev=T, Survey=T, CAc=T)
x.ling <- importCol("c:/scape/data/ling.res", Dev=T, CPUE=T, Survey=T, CAs=T, CLc=T)
edit(importCol)
```

The `cod`, `ling`, and `sbw` assessments were run in Coleraine 3.2 and all data and fitted values are stored in ‘data/cod.res’ and ‘data/sbw.res’. The `oreo` assessment, on the other hand, was run in Coleraine 4.2 which incorporates uncertainty about the von Bertalanffy growth curve. The `oreo` data and fitted values are extracted from ‘oreo.res’, ‘oreo.txt’, and ‘l_at_age.dat’. Although the ‘*.res’ files may not be the best examples of how to organize model output, the `importCol()` function demonstrates some parsing techniques. The key functions are:

```
?read.table # creates data frame
?scan       # creates vector, where each element is one token
?readLines  # creates vector, where each element is one line of text
```

Now take a close look at each element:

```
library(gdata)
ll(x.cod)
x.cod$N      # N@A
x.cod$B      # biomass, yield, recruitment
x.cod$SelMat # selectivity, maturity
x.cod$Dev     # recruitment deviates
x.cod$Survey # survey abundance indices
x.cod$CAc    # commercial C@A
```

```
x.cod$CAs      # survey C@A
```

Notice how tabular data like NA and CA are in long format, and look more familiar when cross-tabbed:

```
xtabs(N~Year+Age, data=x.cod$N)
```

Three approaches

There are mainly three approaches for importing assessment model results into **scape**:

1. Read in text file(s) in whatever format output by the model, and rearrange.

This approach is recommended when the assessment model code should not be changed and/or when file size matters. Coleraine output files, for example, have remained the same for years and other software depends on the current format. R is quite good at parsing data from text files, and any standardized format will do. A well formed output file might start each entry with a unique label, followed by the dimensions, and then the data in a compact layout:

```
Commercial C@A
Years 1971-2002
Ages  1-8
SS     50
Obs
0.000000000 0.000000000 0.084548800 ...
0.000000000 0.000000000 0.069520400 ...
0.000000000 0.000000000 0.270907000 ...
...
Fit
0.000638965 0.005390050 0.051482100 ...
0.000409886 0.013909100 0.051432000 ...
0.000714247 0.008321580 0.123764000 ...
...
```

2. Read in text file(s) where results are rolled out, like in **scape**.

This approach is recommended for model developers who would like to make it easy to import the results into R and **scape**. The assessment model is specifically coded to write output files that are easily digested by **scape**, resulting in a simple `importModel()` function, and verbose text files:

```
Commercial C@A
Gear Year SS Sex Age Obs Fit
1 1971 50 Unisex 1 0.00000000 0.000638965
1 1971 50 Unisex 2 0.00000000 0.005390050
1 1971 50 Unisex 3 0.08454880 0.051482100
...
```

3. Read in R code.

The estimation model could output R code, instead of text files. For example, 'results.R' could look similar to the R source code created by

`dump("x.cod", file="x.cod.R")`. Prager and Williams (<http://cran.r-project.org/contrib/extra/x2r/00ReadMe-X2R.html>) have written C++ and Fortran libraries to enable the model developer to write data as R objects.