A link between Isis-Fish 2.0.27 and FLR

using an R server

ISIS-Fish[1] is a spatial and seasonal simulation model describing the dynamics of resources, exploitation and management. It has been developed to explore the impact of a range of management measures upon fisheries dynamics. It enables to compare the respective impacts of conventional management measures like catch and effort controls, and measures like Marine Protected Areas (MPA)

FLR[2] (Fishery Libraries in R) is a library that consists of a number of R packages. The FLR library under developement is intended to provide a framework within which bio-economic simulation models of fishery and ecological systems can be implemented. FLR users aim at running stock assessement models and/or fishery dynamic models.

The goal of a connection between Isis-Fish and FLR is to suggest both models to work in synergy for modelling the fishery dynamics. At this moment, Isis-Fish is a spatially and seasonnally simulation framework (with a Graphical User Interface) for combining dynamics of stocks and exploitation taking into account of various bioeconomical parameters at fleet and metier levels. FLR libraries rather provide a stock assessement-oriented type of data storage (without a Graphical User Interface) and are at the initial stage toward a multi-fleet modelling of fisheries. If Isis-Fish could gain at reusing FLR' modules for stock assessment, Isis-Fish would be helpful in the development of FLR in a spatially explicit way by comparing findings between both approaches. Furthermore, the development of simulations with FLR could gain to reuse the powerful management rules and the fisher's reactions already performed by isis.

Isis-Fish has been implemented in Java. All the Java objects defined in Isis-Fish are available for modifications via the internal Ecmascript[3] editor using the Isis-Fish graphical user interface. Then, The main loop of the simulation (i.e. The loop which computes at each time step the abundances and the catches from fishing mortalities and calls the management rules) is an Ecmascript source (see Figure 2). The management rule in Isis-Fish are equally implemented as Ecmascript code in order to be possibly modified by the users via the Management Rule Editor interface (see Figure 1).

In the present manual, we aimed at describing a way to create a link between Isis-Fish and FLR using opensource tool boxes which enable us to call R inside Isis-Fish Ecmascript Editor. These tools (Rserve[4] and the associated Java client called JavaClient.jar) enable R commands to be encapsulated in java methods. In our case, R commands are encapsulated in Ecmascript codes. There is two types of Ecmascript codes in Isis :

- A script which defines the main simulation loop we called "SimulateurFLR".
- Several scripts defining each management rules we called "FLRnameOfRules".

The present documentation aims at describing the use and the structure of these particular scripts.

1 http://www.ifremer.fr/isis-fish/objectivesen.php

2 http://flr-project.org/doku.php

3 http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

4 http://stats.math.uni-augsburg.de/Rserve/index.shtml

Isis'side : getting and running ISIS-Fish

Requirement

Running the version 2 of ISIS-Fish requires to have at least the version 1.5 of Java installed on your computer. To know which version is installed in your computer, open a DOS windows and type "java -version". If you end up with an error message, it means that Java is not installed. If you have a version of Java less than 1.5, you need to download the last version of Java Runtime Environment (JRE.). This can be done through the Sun Microsystems website: download java 1.5

Download Isis-Fish and its help manual

Various version of ISIS-Fish can be dowloaded on isis-fish web site (version 1, version 2 or soon version 3). The files containing the software and everything that is needed to run are labelled "ifremer-simulateur-all-xxx.jar" where xxx is the version number. Copy the file "ifremer-simulateur-all-xxx.jar" to a specific folder. "ifremer-language-xxx.jar" jar that contains the language files for running ISIS-Fish in different language could be also downloaded. The isis'manual gathering step by step description for beginners could be equally found on the web site.

Running Isis-Fish

On Windows, a .bat file could be use to store the command line running Isis-Fish. A .bat file is a command launcher, located in the same folder as ISIS-Fish, it will start ISIS-Fish with particular properties only by clicking on this .bat file. The .bat files are created with any text editor, saving the file with the file name extension .bat.

```
java -Xmx512M -jar ifremer-simulateur-all-xxx.jar > erreur.txt 2>&1
```

"-Xmx1400M" is the total virtual memory of your computer allocated by ISIS-Fish during the simulation. You should allocate nearly the maximum of your total available virtual memory (Check in control panel, Performances and maintenance, system for Windows).

R'side : Getting and Running R with Rserve

What Rserve is

Rserve can be download on Rserve web site. Rserve is a TCP/IP server which allows other programs to use facilities of R (see www.r-project.org) from various languages without the need to initialize R or link against R library. Every connection has a separate workspace and working directory. Client-side implementations are available for popular languages such as C/C++ and Java. Rserve supports remote connection, authentication and file transfer. Typical use is to integrate R backend for computation of statstical models, plots etc. in other applications.

The following Java code illustrates the easy integration of Rserve:

```
Rconnection c=new Rconnection ();
double d[]=c.eval("rnorm(10)").asDoubleArray();
```

'd' now contains 10 random samples from the N(0,1) distribution if there is a runing Rserve on the local machine. The Rconnection doesn't have to be created more than once in your application.

Rserve itself is the server, that is a program that responds to requests from clients. It listens for any

incoming connections and processes incoming requests. You need to have R-1.5.0 or higher installed on your system in order to be able to use Rserve.
The server is nothing without clients and that's why a java client (among others) have been developed as a framework: JRclient - a full client suite that allows any Java application (JDK 1.1 or higher) to access an Rserve. The suite is written entirely in Java. It provides automatic type translation for most objects such as int, double, arrays, String or Vector and classes for special R objects such as RBool, RList etc. 'JavaDoc' documentation of JRclient-classes is available online.

Rserve installation on Windows

Copy the binary Rserve.exe to the same directory where R.dll is located (by default C:\Program files\R\rw2011\bin). Rserve automatically detects latest installed version from the registry. A .bat file could be created to store the command line to start Rserve such as :

```
set R_HOME=C:\program files\R\R-2.2.0
"%R_HOME%"\bin\Rserve
```

Running FLR in R

FLR libraries (at least FLCore and FLXSA; see at FLR web site) must be installed in the R workspace by calling install.packages(repos = "http://flr-project.org/R") before to be able to use it in an R session.

Running Isis-Fish with Rserve

The command line for running isis have to be modified to load the 'JRClient' jar file in the same time that the 'isis' jar file. Then the .bat file used to launch isis should be modified as following:

```
java -Doptimization=true -Xmx512M -cp ifremer-simulateur-all-
2.0.24.jar;JRclient.jar fr.ifremer.IfremerSimulation -l en -c US > erreur.txt
2>&1
```

⚠ Once isis have been launched, *Rserve have to be manually launched* . Remember that Rserve must be launched before running a simulation (if not, the simulation will fail).

Although ISIS-Fish is running properly, at this stage it does not contain any database yet. You may load the tutorial database: Click on 'File' then 'Restore the database'. The software then asks whether you want to save the current databases, before importing data. Since you have no database already loaded, you should click on 'NO'. The software then prompts you to ask whether you want to delete current data in the database. Since you have no database already loaded, you should click on 'YES', because it is a recommended option. Using the browser, select the file containing the database (it is a file with the extension .xml, you could find base examples on the web site but the ecmascript have to be deleted before use of database with FLR) and click on 'Open' ISIS-Fish loads the file.

At this stage, the Ecmascript code required to run the main simulation loop is not in isis yet because it is not contained in the base .xml (be careful that if you use a base with default scripts already present in the database, then isis won't be able to load the FLR-Ecmascript). The Ecmascript-FLR have to be loaded by clicking on *File* then 'Restore the database' then 'NO' and then 'NO' again, because database have already been loaded and you don't want to erase it. Using the browser, select the file containing the FLR-Ecmascript (it is also a file with the extension .xml).

Description of the simulation loop Ecmascript using FLR

This script is a modified version we called "simulatorFLR" of the script defining the main simulation loop in isis. As usual, on the Isis side, for each population, the operating model loop computes dynamics of abundances and catches in relation to the dynamic of fleets. In addition, as usual, this script calls at each time step the scripts of management rules (if activited) which impact the stock and fleet dynamics. In contrary to the default script, a connection with R is created inside this modified version. Then, on the R side, FLR objects (i.e. FLBiol, FLStock, FLXSA and FLSR) are created over the script and act as collectors of the data created by the isis simulation loop.

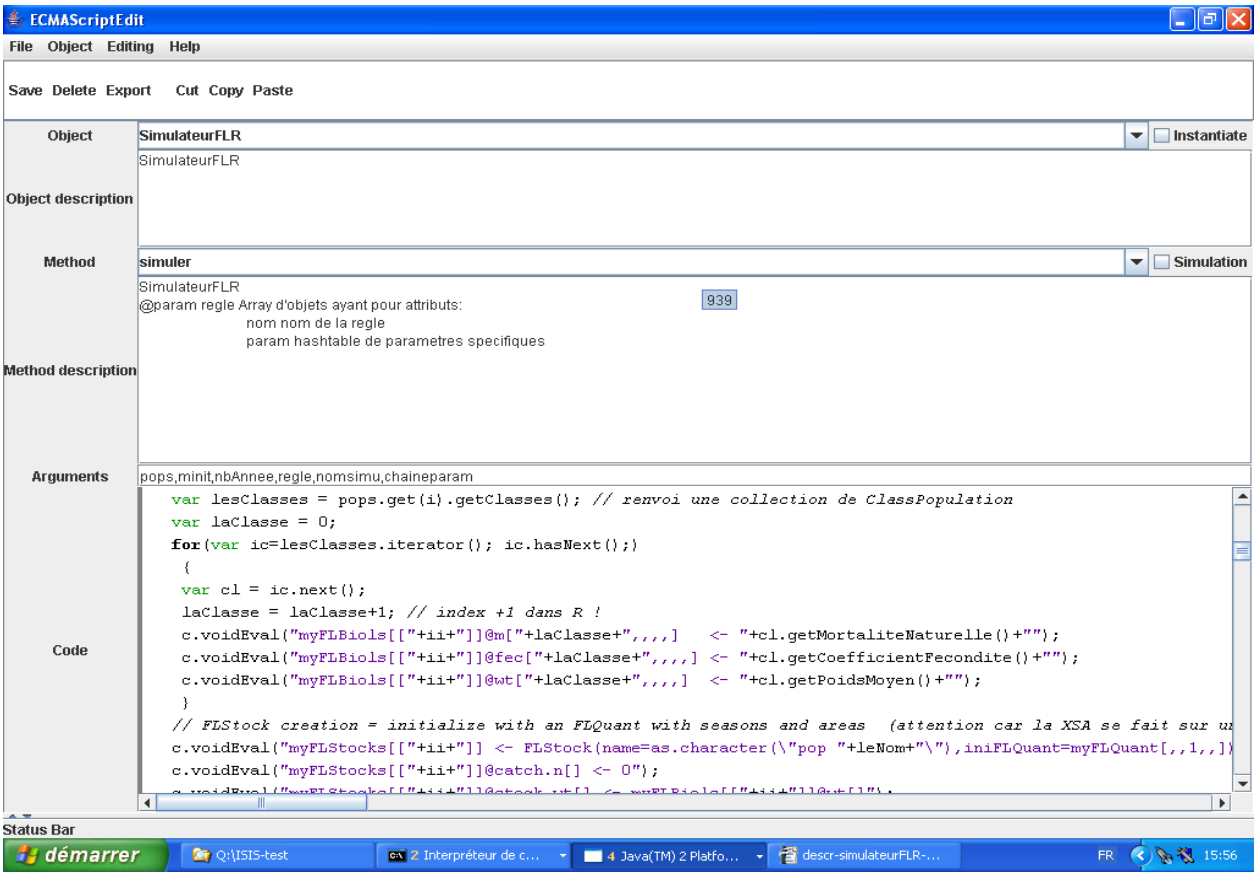

Figure 1: A view of the Ecmascript Editor with a code chunck from the "simulatorFLR" script

Structure of the algorithm

```
// creation of the R connection calling Rserve
var c = new Packages.org.rosuda.JRClient.Rconnection()

// loading FLR libraries in R
c.voidEval("library(FLCore)");
c.voidEval("library(FLXSA)");

// loading other sources if desired
c.voidEval("source(\"Q:/ISIS-test/VPA.r\")");
```

```
**************************************************************************
*************initialization***********************************************
**************************************************************************


// for each population
for(var i=0; i<pops.size(); i++)
   {

   // initial creation of FLBiol, FLStock, FLXSA and FLSR objects (one per pop)
   // FLBiol : underlying truth
   // FLStock : for stock assessment and bioeconomic model

   // natural mortality, fecondity, mean weight : get input data from isis

   // creation of an FLFleet 'survey' for xsa tuning
   //(catches for survey are no spatially explicit
   // and occur on a yearly time step)

   // import catchabilities from isis
   // for  'survey' (mean by age over season)

    c.voidEval("mySurveys[["+ii+"]]@catches[[1]]@q["+laClasse+",,,,]<-
                                                    mean(catchabilities));
    c.voidEval("mySurveys[["+ii+"]]@catches[[1]]@catch.sel[] <- survey.sel");
    c.voidEval("mySurveys[["+ii+"]]@effort[] <- 1");

   // XSA control object
    c.voidEval("xsa.control <- FLXSA.control(tol= 1e-19, maxit   = 50,min.nse =
0.3,  fse  =  0.5,rage=  -1,qage=  6,shk.n=  TRUE,shk.f=  TRUE,shk.yrs  =  5,shk.ages=
2,window  = 100,tsrange = 99,tspower = 0) ");

   } // end for population



**************************************************************************
*************simulation loop**********************************************
**************************************************************************



****stock assessment*************************************
    if(date.getMois().getNumMois() == 0 && date.getAnnee()>= 2) // if january
      {
       for(var m=0; m<populations.size(); m++)
         {
          // running xsa with FLR
          c.voidEval("myFLXSAs[["+xx+"]] <-
            FLXSA(obs, temp.tuning, FLXSA.control())");

         //FLStock update with XSA results:
         //(divided by the number of seasons + number of zones)

         // FLSR update for Spawing Stock Biomass slot
         //SSBcurrent gives the SSB of the last year

          } // end if date
        } // end for population



**************management rule ****************************
```

```
// test if the each of the management rules has to be applied for
 // the current time step
// (for example, the area have to be closed if the biological reference point is
reached, etc.)
// if true, apply the activated rules on each metier.

***************abundances and catches computation*************


  for (var k=0; k<populations.size(); k++)
     {
      writeln("calculation for the current month");
     simulator.calculate(matriceAbundance, populations.get(k), date);
     }


*******************update for FLR objects *******************

for(var w=0; w<populations.size(); w++)
       {
     // update for FLBiol
  c.voidEval("myFLBiols[["+ww+"]]@n["+classeC+","+laDateD+",1,"+laSaisonS+","+la
ZoneZ+"]<-"+abundances+"");

    // update for FLStocks
       c.voidEval("myFLStocks[["+ww+"]]@catch.n["+classeC+","+laDateD+",1,"+laSa
isonS+","+laZoneZ+"]<-"+lesCatches+"");

    //update for FLFleet 'survey'
    //computation of fishing mortalities for the survey
    //cumul for the 'survey' catches

      } // end for populations


*******************date incrementation*****************************
  date.inc();

}

-------------------------writeln("end of loop")----------------------

// file exports in .csv
Export.exportFLRabundances(pops,resManager, buffer); // export for FLBiol
Export.exportFLRxsa(pops,resManager, buffer); // export for FLStock@stock.n
Export.exportFLRcatches(pops,resManager, buffer); // export for FLStock@catch.n
```

Description of the Ecmascript management rules using FLR

The main purpose of the simulation model is to quantitatively assess the relative performances of a range of management measures applicable to a composite fishery. We aim at comparing the impact of conventional management measures including catch Limitation (Total Allowable Catch (TAC)), effort limitation (Total Allowable Effort (TAE)) and gear restrictions, to measures like Marine Protected Areas (MPA). Because our model is spatially explicit with a monthly time step, we may consider management measures that apply either during some months throughout the year, and are either global (at the scale of the region) or local (within a particular zone). In this way, a large variety of MPA designs may be evaluated. In the model, each management measure is defined by several parameters. Each measure results in one or several constrains on the exploitation, that lead fishing unit to modify fishing effort in some respect. These changes are termed “fisher’s reaction to management measures”. The different management measures that are currently implemented are: "PermanentChangeGearParameter","GearProhibition","PermanentGearProhibition","GearSpecific MPA", "PermanentGearSpecificMPA", "MPA", "MinimumLandingSize", "TAC".

The results of the computation using FLR objects in the ecmascript monitoring the main loop of the simulation could be easily reused inside management rule scripts, in particular to decide if management rules have to be applied or not. As an example of a connection between isis and FLR, a modified version of the script for a management rule type "MPA" have been written and we called it "boxesWithBpaFLR". This script controls the area total closure in relation to the results of an XSA performed with FLR and an arbitrary Biomass Reference Point (Bpa) as input (rule => if Spawing Stock Biomass < Bpa then closure of the area).

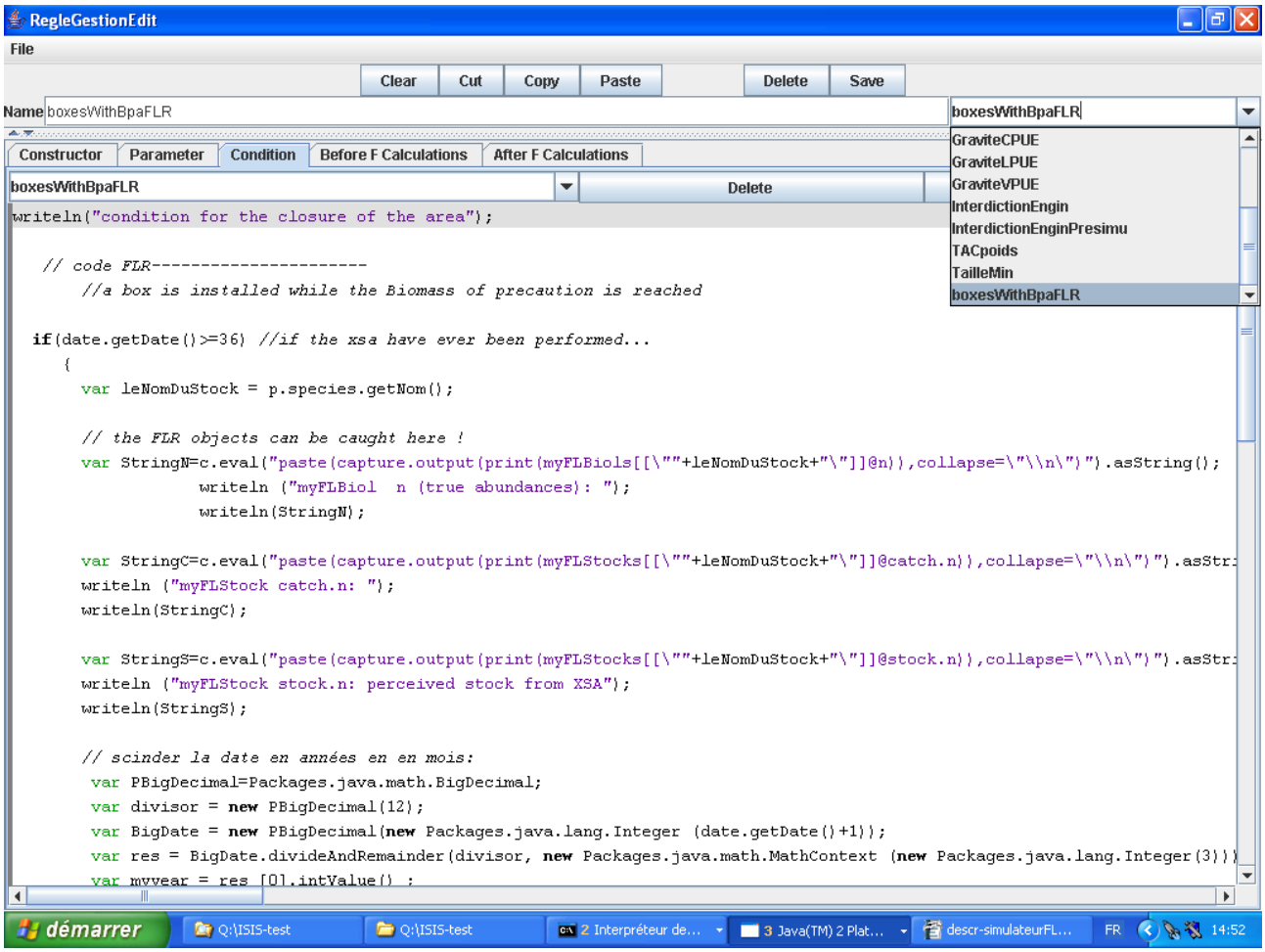

Figure 2 : Ecmascript Editor for management rules with a code chunk using FLR

Example of results

As example, a database describing a no spatial hake/nephrops mixed fisheries in the Bay of Biscay is used. Running the Ecmascript containing the main simulation loop on our database lead to the computation at each time step (every month in isis) of the abundances and the catches on the two stocks depending on the dynamic of fleets (Figure 3). At each yearly step (every 1st January), an XSA using the module FLXSA given by FLR tools have been performed (Figure 3). In a next step, management rule can occur over the simulation depending on the FLR computation. FLR objects can be caught in isis as inputs for the decision of applying a management rules. Then, Figure 4 shows the results of a simulation when a closure of the area is decided while a Biomass Reference Point is reached. In this way, FLR objects can reuse the poweful management rules perfomed by isis.

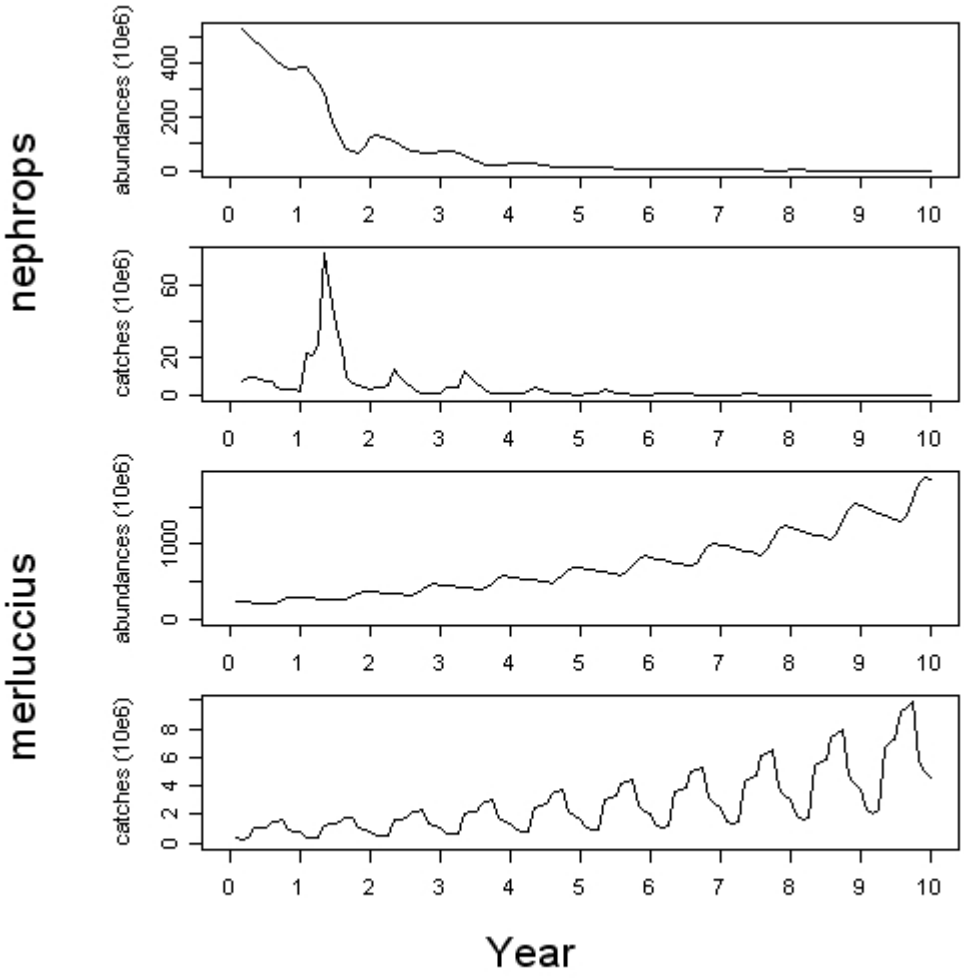

Figure 3 : Time series for simulated abundances and catches using isis on a database describing the hake/nephrops fisheries in the Bay of Biscay.

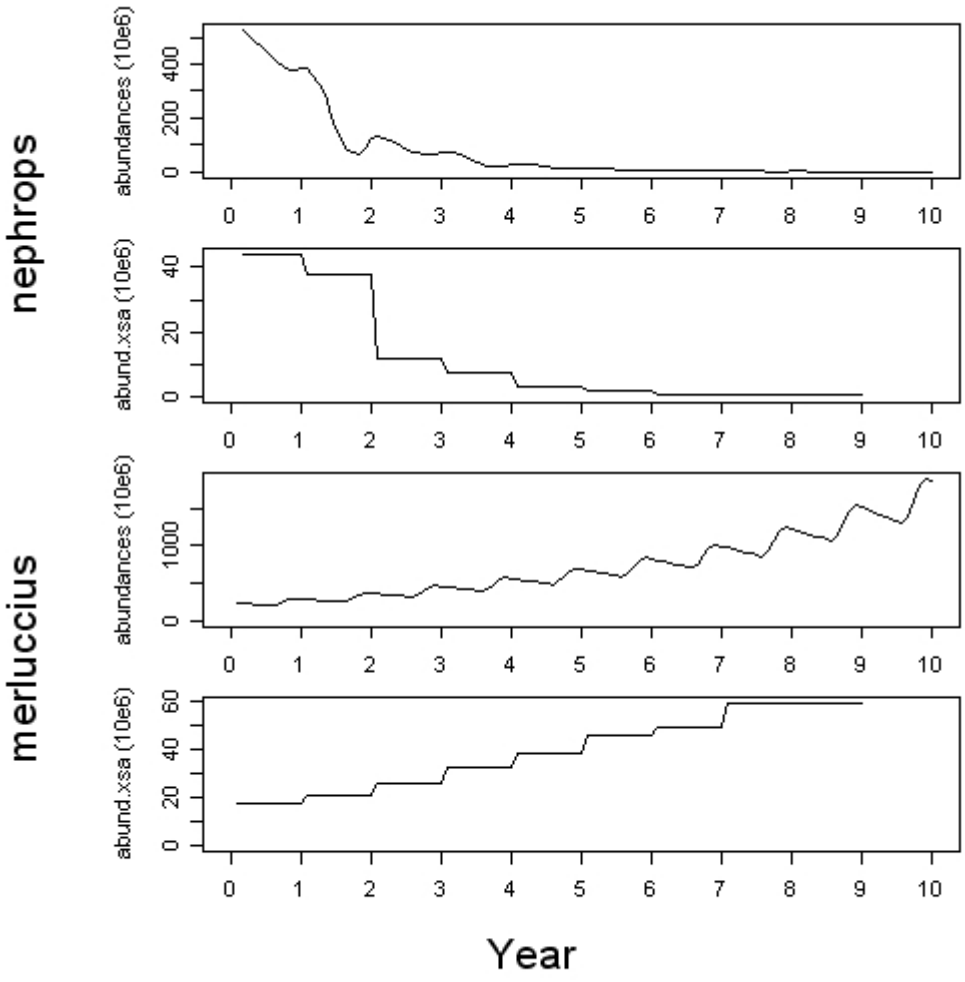

Figure 4 : Time series for simulated abundances (from Figure 1) and estimated abundances using XSA retro-calculation from FLR inside isis Ecmascript loop.