A users guide to the COBECOS software

COBECOS Work Package 6

Tom Carruthers & Charles Edwards
Imperial College London

October 2008

Contents

1 Introduction

The COBECOS software uses object oriented programming and pre-written R functions to provide a powerful and flexible means of undertaking cost-benefit analysis of enforcement strategies. It is suitable for analyses involving a single exploited population (stock) and a single type of enforcement.

The COBECOS software is designed to (1) be accessible to users without a background in R programming whilst (2) providing more experienced R programmers with sufficient control and flexibility to carry out management strategy evaluation using the COBECOS functions and methods.

2 Installing the necessary software

The COBECOS software requires the installation of two programs in order to work.

2.1 The R package

R can be found at the Comprehensive R Archive Network (CRAN): http://cran.r-project.org/bin/windows/base. Because of continual changes to both R and the COBECOS code it is important that you have the latest version installed.

2.2 An R-compatible text editor: Tinn-R

In order to easily run the command lines of the tutorial (and use R efficiently) you require a text editor that will send lines of code to the R console. A great piece of freeware for doing this is TINN-R. The latest version can be found at https://sourceforge.net/projects/tinn-r.

2.3 Running the tutorial

Once R and TINN-R are installed then open the file `Tutorial.r` with TINN-R. Open R from within TINN-R (this ensures R is working within the correct directory) using the 'R→Start preferred Rgui' command. If an error returns stating that the Rgui cannot be found then it is necessary to change the search path used by TINN-R. Go to 'Options→Main' and click on the 'R→General' tab. Then change the search path so that it points to Rgui.exe. The tutorial can then be run according to the instructions.

R commands can be sent directly from TINN-R to R. Just drag your mouse over the line or lines you would like to run from the R console and click the 'Send line' or 'Send selection' button on the tool bar.

Another useful tip for running the COBECOS commands is to ensure that focus is not immediately returned to the TINN-R program after some code has been sent to R. This allows you to examine outputs in R before continuing to run other code from TINN-R. To do this, enter the 'Options→Main→Application' menu from the top of the screen. In this menu click on the 'R' tab. Midway down this menu is a check box labelled 'Return focus after sending to R'. Uncheck this box.

3 Functionality of the COBECOS software

The key functional aspects of the COBECOS software are listed below.

1. Defines a range of theoretical relationships between enforcement effort and enforcement cost (EC), and enforcement effort and probability of detecting violations (EP).
    - Relationships can be specified in a control file or selected according to their fit to the data
2. Fits theoretical EC and EP relationships to observed data.
    - Accepts raw EC and EP data
    - The parameters defining the functional form of the relationship are estimated
    - If no data are available, parameters can be specified in the control file
3. Defines the functional form of private and social benefit functions
    - Default functional forms exist and can also be specified by the user
4. Evaluates the private and social benefits at the user defined level of enforcement effort
    - User defines the enforcement effort (that can be 'real world' or hypothetical)
    - Returns the social and private benefits at the effort level given the EC and EP relationships
    - Plots the marginal social benefit(s) across the enforcement level
    - Returns the predicted level of illegal catch
5. Evaluates the private and social benefits at optimal levels of enforcement effort.
    - Optimisation routines selects the enforcement effort to maximise the social benefit
    - Returns the social and private benefits at this effort level given the EC and EP relationships
    - Plots the marginal benefits across enforcement level
    - Returns the predicted level of illegal catch
6. Allows the user to specify types of stochastic uncertainty and evaluate social and private benefits
    - Implementation error introduces uncertainty in the fisher's response to a specific enforcement effort level
    - Estimation error introduces uncertainty around the estimated EC and EP relationships
    - Either or both types of uncertainty can be included
    - Enforcement level can be optimised or set to a user defined level

4 Overview of the COBECOS methods

The full functionality of the COBECOS software can be accessed using the commands `new` and `BCalc`. These functions are described here.

4.1 The initialisation method: `new`

The `new("COBECOS",...)` command calls an initialisation method: it creates a new COBECOS object from the files in a specified folder. An example (taken from the tutorial) is:

```
HakePath<-"C:/COBECOS/COBECOS_data/Hake/"
Hake <- new("COBECOS",path=HakePath,fitinfo=TRUE,graphics=TRUE)
```

Arguments to the `new("COBECOS",...)` command are:

`path`: Character string specifying the directory in which data and control files are stored

`fitinfo`: Logical value specifying whether parameter values should be returned

`graphics`: Logical value indicating whether a graphical output should be returned

The control and data files in the `path` directory should be comma delimited and named `Control.csv`, `CostData.csv` and `ProbData.csv`. An example of each is distributed with the tutorial. Note that if the functional form is specified completely in the `Control.csv` file, then a data file does not need to be supplied.

`Control.csv` should have a single row with the column headers:

`Type`: Character string giving the name of the enforcement type

`EPMod`: Functional form of the effort-probability (EP) relationship

`EPpar1`: EP parameter 1

`EPpar2`: EP parameter 2

`EPpar3`: EP parameter 3

`ECMod`: Functional form of the effort-cost (EC) relationship

`ECpar1`: EC slope parameter

`ECpar2`: EC intercept parameter

`ECpar3`: EC power parameter

`Effort_Min`: Minimum enforcement effort

`Effort_Max`: Maximum enforcement effort

All headers should be populated. The EP relationship can be either logistic or exponential, the latter with or without an intercept. The EC relationship can be either linear or non-linear, both with or without an intercept. The functional forms and their parameters are detailed in section 5. Given a control file, and (optionally) effort-probability (EP) and effort-cost (EC) data, in the `path` directory, this initialisation method can perform one of two functions:

1. If EPMod and/or ECMod are labelled as `NA` in the `Control.csv` file, then the most appropriate functional forms will be selected (using the Akaike Information Criterion), estimating the relevant parameters
2. If EPMod and/or ECMod are labelled with the relevant functional forms, namely `logistic/exponential` or `linear/nonlinear` respectively, then these functional forms are fitted to the data. Only those parameters labelled `NA` are estimated. All other parameters are assumed equal to those specified in the control file

The functional EP and EC relationships are specified over the effort range given in the `Control.csv` file under the headings `Effort_Min` and `Effort_Max`. This allows the user flexibility to explore effort levels beyond those recorded in the data.

A COBECOS object contains slots in which information is stored. Once initialised, slots in the new COBECOS object (in this example `Hake`) will be populated with all of the EP and EC data in addition to the fitted or user specified EP and EC relationships (see the Appendix for a list of slots).

4.2 The benefit calculator: `BCalc`

The `BCalc` command is designed to estimate the social and private benefits associated with the enforcement level. The enforcement level can be either specified by the user or selected to maximise the expected social benefit. An example of a `BCalc` function is:

```
Hake<-BCalc(.Object=Hake,Optimize=TRUE,stoch=0,graphics=TRUE)
```

Arguments to the `BCalc(...)` command are:

`.Object`: Character string specifying the name of the COBECOS object

`Optimize`: Logical value specifying whether enforcement levels should be selected according to the maximal social benefit or left at the user defined values

`stoch`: Numeric value specifying stochastic uncertainty as `0`: no uncertainty, `1`: implementation error, `2`: estimation error, `3`; both implementation and estimation error

`grahics`: Logical value indicating whether a graphical output should be returned

The `BCalc` function performs two optimisations. It will calculate the expected harvest through a maximisation of the private benefit function (Equation 8), and, if `Optimize=TRUE`, it will optimize effort to maximise social benefits (Equation 10). The `BCalc` function presents two primary graphical outputs in the form of a marginal social benefit profiles against enforcement effort and the private benefit profile against harvest. Both should be examined carefully to ensure that optimisations were successful, particularly when stochasticity is included. The two forms of stochasticity are dealt with in more detail in section 7.

Note that `BCalc` creates a new object from another COBECOS object. This allows users to either overwrite a previous object or use it as a template for making a series of modified objects. Information resulting from the calculations performed by `BCalc` are stored in object slots (see Appendix).

5 Functional forms

The functional forms relating the enforcement effort to enforcement cost and enforcement effort to the probability of sanction are described here. In each case E refers to enforcement effort, and p_1, p_2 and p_3 to parameters 1 to 3 (i.e. `ECpar?` and `EPpar?`). C refers to cost and P to probability.

5.1 Effort-Probability relationship

Logistic

$$P(E) = \frac{p_1}{1 + e^{(p_2 - E)/p_3}} \tag{1}$$

Exponential with intercept

$$P(E) = (1 - e^{-p_1 E})(1 - p_2) + p_2 \tag{2}$$

Exponential

$$P(E) = 1 - e^{-p_1 E} \tag{3}$$

5.2 Effort-Cost relationship

Non-linear with intercept

$$C(E) = p_1 E^{p_3} + p_2 \tag{4}$$

Non-linear

$$C(E) = p_1 E^{p_3} \tag{5}$$

Linear with intercept

$$C(E) = p_1 E + p_2 \tag{6}$$

Linear

$$C(E) = p_1 E \tag{7}$$

6 Economic models

The economic relationships implemented by `BCalc` are described here. Default private and social benefit functions are given, matching those described in the numerical example of COBECOS Memo 4.

Private Benefit function

$$B_P(q, E) = pq - c\frac{q^2}{x} - f\pi(E)q \tag{8}$$

Harvest response function

$$q^* = \arg\max_q [B_P(q, E)] \tag{9}$$

Social Benefit function

$$B_S(E) = (p - \lambda)q^* - c\frac{(q^*)^2}{x} - \gamma(E) \tag{10}$$

Optimal enforcement effort

$$E^* = \arg\max_E [B_S(E)] \tag{11}$$

Both the private and social benefit functions are stored in object slots. This allows the user flexibility to define benefit functions of their choice.

6.1 Function definitions

To define new private and social benefit functions, it is simply a question of allocating a new function to the appropriate slot in the COBECOS object. The default functions, corresponding to Equation 8 and Equation 10, are:

```
.Object@PrivateBFunc <- function(Harvest,Price,FCost,Biomass,
                          ExpFine,UnitTax,TAC) {
       Price*Harvest-FCost*((Harvest*Harvest)/Biomass)
               -ExpFine*Harvest }

.Object@SocialBFunc <- function(Price,ShadowVB,Harvest,FCost,
                         Biomass,TotalCost,ExpFine,UnitTax,TAC) {
       (Price-ShadowVB)*Harvest-FCost*((Harvest*Harvest)/Biomass)
               -TotalCost }
```

Note that the function definitions include a number of arguments that may not be defined in the function itself. These reserve slots in the COBECOS object should they need to be used. When defining a function, all arguments must be included in the correct order. The function itself can consist of any combination of the following arguments

Private benefit function

`Harvest`: Total extractions from the fishery (q)

`Price`: Market price per unit of fish (p)

`FCost`: Private cost incurred per unit of fish (c)

`Biomass`: Total biomass of the exploited stock (x)

`ExpFine`: Expected fine incurred by a fisher per unit of fish ($f\pi$)

`UnitTax`: Tax per unit of fish

`TAC`: Total allowable catch

Social benefit function

`Price`: Market price per unit of fish (p)

`ShadowVb`: Shadow value of biomass (λ)

`Harvest`: Total extractions from the fishery (q)

`FCost`: Private cost incurred per unit of fish (c)

`Biomass`: Total biomass of the exploited stock (x)

`TotalCost`: Total cost of enforcement (γ)

`ExpFine`: Expected fine incurred by a fisher per unit of fish ($f\pi$)

`UnitTax`: Tax per unit of fish

`TAC`: Total allowable catch

6.2 Changing slot values

Before running `BCalc` the values of parameters included in the defined economic models must be specified. This is easily achieved by changing a slot value in the current R session. A complete list of slots is given in the Appendix. For example, if biomass is equal to 5000 mass units and the COBECOS object is called 'Hake' (i.e. `.Object=Hake`) then simply type

```
Hake@Biomass <- 5000.
```

7 Stochastic uncertainty

The `BCalc` function allows stochastic uncertainty to be taken into account when estimating the social benefit associated with enforcement effort. Uncertainty can take two forms:

1. **Implemention error:** Harvest taken by fishers to maximise their own private benefit (Equation 9) is uncertain, being implemented with an assumed log-normal error distribution

$$q' = q^* e^{\varepsilon} \tag{12}$$

$$\varepsilon \sim N(0, \sigma^2)$$

2. **Estimation error:** Effort-Probabilty and Effort-Cost relationships are uncertain, assuming a normal error distribution around the predicted values

$$P'(E) = P(E) + \varepsilon \tag{13}$$

$$C'(E) = C(E) + \varepsilon \tag{14}$$

$$\varepsilon \sim N(0, \sigma^2)$$

It is assumed that $P' \geq 0$ and $C' \geq 0$, so that the error distribution may be truncated at small values of E.

7.1 Optimisation under uncertainty

With reference to Equation 11 the intention is to optimise the expected social benefit over uncertainty (ε) in implementation and/or estimation.

Optimal enforcement effort under uncertainty

$$E^* = \arg\max_E [\mathrm{E}[B_S(E, \varepsilon)]] \tag{15}$$

To evaluate E^* would require the expected social benefit to be estimated across stochastic error(s) for each enforcement level, and then maximised over the enforcement level. This is computationally demanding. Since error is (largely) independent of E, Equation 15 is equivalent to:

$$E^* = \mathrm{E}[\arg\max_E [B_S(E, \varepsilon)]] \tag{16}$$

which is a more convenient form to analyse. Monte Carlo simulation is used. Thus a specified number of stochastic samples are drawn, and optimal enforcement levels estimated for each sample. E^* is taken as the median of the resultant distribution of effort values.

Note that when plotting the marginal social benefit(s) against enforcment effort, Equation 15 is evaluated, taking the median social benefit at each level of enforcement. This is a robust method of visualising the optimisation, but it is necessarily of lower resolution. Nevertheless it is important to ensure that the effort level returned by the optimisation correspond approximately to the visualized maximum, to ensure it was successful.

A COBECOS object slots

A list of the slots in a COBECOS object is given here. Each can be accessed using the `.Object@slotName` command in R. A list of slot names can be obtained using `slotNames(.Object)`.

`slotName`	Class	Default	Description
`control`	data frame	Supplied as `Control.csv`	Control file for the fitting of effort-probability and effort-cost relationships
`EPData`	data frame	Supplied as `ProbData.csv`	Empirical data on enforcement effort against probability of sanction
`ECData`	data frame	Supplied as `CostData.csv`	Empirical data on enforcement effort against enforcement cost
`Ename`	character	`.Object@control$Type`	Name of enforcement type
`EPFitFunc`	function	`NA`	Effort-probability relationship definitions for each enforcement type
`ECFitFunc`	function	`NA`	Effort-cost relationship definitions for each enforcement type
`EPFitPar`	numeric	`NA`	Parameters for effort-probability function
`ECFitPar`	numeric	`NA`	Parameters for effort-cost function
`Effort`	numeric	`NA`	Vector of enforcement efforts
`Fine`	numeric	`NA`	Vector of fines per unit harvest for violations according to each enforcement type
`Biomass`	numeric	`NA`	Resource biomass available to fishing
`FCost`	numeric	`NA`	Cost of fishing per unit harvest
`Price`	numeric	`NA`	Price of fish per unit harvest
`ShadowVB`	numeric	`NA`	Shadow value of resource biomass
`Effort_Min`	numeric	`NA`	Minimum of effort range
`Effort_Max`	numeric	`NA`	Maximum of effort range
`UnitTax`	numeric	`NA`	Tax per unit harvest
`TAC`	numeric	`NA`	Total Allowable Catch
`PrivateBFunc`	function	See Equation 8	Private benefit function
`FishingRFunc`	function	See Equation 9	Optimisation routine for estimating harvest based on the private benefit function

`SocialBFunc`	function	See Equation 10	Social benefit function
`SocialOBJFunc`	function		Sub-routine for optimising social benefit
`Cost`	numeric		Vector of costs for each enforcement type
`Prob`	numeric		Vector of probabilities (of sanction) for each enforcement type
`ExpFine`	numeric		Expected Fine
`PrivateB`	numeric		Estimated private benefit
`Harvest`	numeric		The resource harvest
`SocialB`	numeric		Estimated social benefit
`Optimized`	list		Output from optimisation routine when estimating optimal enforcement levels
`StochN`	numeric	`1000`	Number of replicates
`Stochastic`	logical	`FALSE`	Is estimate of social benefit subject to stochastic uncertainty?
`ImpError`	logical	`FALSE`	Is estimate of social benefit subject to uncertainty regarding the fishing response?
`EstError`	logical	`FALSE`	Is estimate of social benefit subject to uncertainty in our estimation of the effort-probability and effort-cost relationships?
`ImpErrorSE`	numeric	`0.01`	Log-normal standard error for implementation uncertainty
`ECEstErrorSE`	numeric	`0.01`	Standard error for effort-cost estimation uncertainty for each enforcement type
`EPEstErrorSE`	numeric	`0.01`	Standard error for effort-probability estimation uncertainty for each enforcement type
`HarvestErr`	numeric	`1.00`	Multiplicative log-normal error in harvest (implementation uncertainty)
`ECErr`	numeric	`0.00`	Additive normal error in estimation of cost (estimation uncertainty)
`EPErr`	numeric	`0.00`	Additive normal error in estimation of probability of sanction (estimation uncertainty)
`StochCost`	numeric		Cost values for each stochastic sample across enforcement types

`StochProb`	numeric	Probabilty (of sanction) values for each stochastic sample across enforcement types
`StochExpFine`	numeric	Expected fine for each stochastic sample
`StochEffort`	numeric	Effort values for each stochastic sample across enforcement types
`StochPrivateB`	numeric	Private benefit for each stochastic sample
`StochHarvest`	numeric	The resource harvest for each stochastic sample
`StochSocialB`	numeric	Social benefit for each stochastic sample