# Reconstructing the diagrams and results in Polhill et al. 2013

There are four main steps:

1. Setup: Run a script to create all the parameter files that the model is going to use
2. Run: Run a script that will make all the runs of the model and produce the output files
3. Process: Run a script that reads all of the model output files and creates a summary file for use with R. [Bugs in this script meant, if I remember rightly, that I had to make some corrections to this file using R, sed, awk, etc.]
4. Analyse: Run various scripts to create the diagrams and analysis. The main figures of interest in the paper are: figure 3 (sunflower plots with classification tree partition lines), table 4 (results of various tests for nonlinearity), figure 4 (sunflower plots with various GAM and linear fits for selected runs, also showing decision tree partition lines), figure 5 (main decision tree summarising all the results), appendix A (as per figure 4, but for runs not shown therein).

## Setup

* SSS-StopC2-Cluster-expt.pl is a Perl script that creates the parameter files for a single run.
* SSS-StopC2-Cluster-create.sh is a shell script that calls the above Perl script to create all the parameter files for the experiment. This creates the first batch of parameter files for the runs described in table 2 of the paper. The corresponding script for creating the second batch of parameter files (in table 3) is no longer available, but could be created from this one using the table in the paper.

## Run

* SSS-StopC2-Cluster-run.sh is a shell script that runs FEARLUS-SPOMM in batches of 12 runs at a time (the two asps, the two bets and the three rats). Again, the run script that did the second batch of runs in table 3 of the paper is not available, but could easily be created from this file using the table.

The compressed tar files contain the output to here. (It is not intended to attempt to reproduce them in MIRACLE.)

## Process

* analysege\_gp.pl is a Perl script that processes the outputs from FEARLUS-SPOMM and creates a CSV file containing all the variables used in the analysis. It takes a single argument, which is a number indicating the kind of experiment that has been done, and determines the output that will be produced. Argument 8 should be used for the first batch of runs, argument 9 for the second. Numbers of versions of this script exist on different machines. The two main machines are the Mac laptop I was working on sometimes, and the fgridln*X* servers I was running the model on
  + The Mac version of analysege\_gp.pl correctly uses $report[3] rather than $report[1] when getting the income of land managers from ManagerIncomeReport. It also has a column heading for Expenditure, which the fgridlnX version does not. Further, it initialises $n\_years to 0 in get\_occupancy, which the fgridlnX version does not. Another difference is in the directory used for different batches of runs, which was due to the different ways the results were stored on different machines, and is unimportant.
  + On the Mac, I have analysege\_gp2.pl, which is identical to the Mac version of analysege\_gp.pl, except that it wrongly uses $report[1] to get the income.
  + On the Mac, I have analysege\_gpLU.pl, which (when compared with the Mac version of analysege\_gp.pl) seems to add functionality to compute the occupancy of each land use and add these to the columns reported. It further appears to contain some bug fixes:
    - In get\_income, if income is 0, it will not add $reward / $income to $total\_preward
    - In get\_extinction, it puts ‘NA’ if a species does not go extinct instead of ‘no’
    - In get\_occupancy, $shannon, $richness and $equitability are initialised to 0 rather than not defined on declaration.
  + On fgridlnX, I have analysege\_gpLU2.pl. Apart from an irrelevant change for where to look for results from the second batch of runs, this computes the occupancy differently when compared with analysege\_gpLU.pl, using $prop \* $nspp for $this\_occup[$i – 1] instead of just $prop. This seems to correct for a division by $nspp in an earlier computation for $prop.
  + On fgridlnX, I have analysege\_gpLU3.pl, which differs from analysege\_gpLU2.pl in adding options 10A-D and 11A-D to the set of experiment output files the script can generate a CSV for. These seem to cover an extra set of runs not reported on in the paper, in which I used a new class of land manager with an ‘optimism’.

Ideally, therefore, I would have used analysege\_gpLU2.pl to create whatever CSV file eventually contained the data on which the analyses in the paper are based. As things stand, I have no record to prove this to be the case. A provenance database would have fixed this problem because I would know which script had been run to generate which CSV file (and later, which CSV file had been used with which analysis).

The commands that were run would have been something like this:

./analysege\_gpLU2.pl 8 > batch1.csv  
./analysege\_gpLU2.pl 9 > batch2.csv  
cp batch1.csv all-results.csv  
tail +1 batch2.csv >> all-results.csv

It is all-results.csv that I would then have read in to R (e.g. using read.csv()) to do the manual steps.

Further processing was done by hand, probably interactively with R, to remove runs having expenditure or rates of bankruptcy deemed to be too high as described in the paper. CSV files with ‘LOBE’ or ‘lobe’ in the name, refer to ‘LOw Bankruptcy and Expenditure’.

## Analyse

Notes in the README file state that diagrams in the EMS paper were produced using cmp-scenarios.R, cmpgam3-5any4.R, cmpgam3-4any16.R and economy.R. The last of these is of less interest, as it produces the diagrams showing the economic time series in figure 2 of the paper. The other scripts have been used in earlier versions of the paper, and not the final version. Some of the R scripts clearly contain corrections aimed at handling some of the bugs in the analysege\_gp\*.pl scripts discussed above. The scripts actually used are described below:

### figure 3, figure 4 and appendix A

These figures were all created with the same family of scripts figure2-3\*.R. Four versions of this script exist on the Mac. In order of creation, these are:

* figure2-3.R. In the usage statement for this, it is called figure2-3K4.R, which suggests a previous version of this program existed. Various programs feature ‘K4’ in their name, and I think this pertains to building GAMs with *k* = 4. This program takes a data file, a variable to plot on the y-axis, and a PDF file, and produces a plot for each scenario. It has the options –showdata, -plotlm, -plotgam4, -splits and –bs. The –showdata option switches off plotting the data using sunflower plots; the –plotlm option switches off plotting a linear model fit to the data; the –plotgam4 option switches off plotting GAMs; the –bs options switches on using bs = “ts” in the smooth function s() when constructing the GAM; the –splits option switches on drawing vertical lines at each of the split points of an rpart() explaining the y-axis using the x-axis (which is always Incentive).
* figure2-3s.R. The differences between this and figure2-3.R seem to be that rather than producing a diagram for all scenarios, you choose a ‘main’ scenario and four ‘small’ scenarios, and the program just produces graphs for those. These seem very much akin to the diagrams in figure 4 of the paper, and is very much my best guess for the origin of this figure.
* figure2-3small.R. The only difference between this and figure2-3s.R is that this one is programmed to plot a specific arbitrary set of scenarios in the ‘small’ style. This is my best guess for the origin of the figures in appendix A.
* figure2-3part.R. This program seems to have been used to produce figure 3. It takes a configuration file (fig2-3part.csv), which specifies the scenarios to do, and for each scenario the minimum and maximum incentive to use. Based on the ‘usage’ statement in this script, it was developed from figure2-3s.R, but it involves a significant number of changes from a view of diff output. Many of these changes are down to Richness being hardcoded as the y-axis variable in figure2-3part.R, when it is specified as a command line argument in the other versions of this script. Default arguments have changed, with linear models and GAMs not being plotted by default, and the rpart() separations plotted. Note that the latter are different from the separation lines drawn by the other versions of this script.

### table 4

The nonlinear\*.R family of R scripts were used to create the data used in Table 4. In date order, there are the following scripts:

* nonlinearKWholm.R. Uses a Kruskal-Wallis test to check for nonlinearity, calling p.adjust() in R, with method = “holm” to adjust P-values. This seems to be an early method for testing for nonlinearity that is unrelated to any tests conducted in the final version of the paper.
* nonlinear.R. Unrelated to nonlinearKWholm.R, this script performs a variety of tests similar to those in the paper, having two arguments: the CSV file summarising the output of the model, and a CSV file to contain the results of all the tests. However, it uses Expenditure as the explanatory variable rather than Incentive as in the paper.
* nonlinearK4.R. Differs from nonlinear.R only in that k = 4 is used for the GAM rather than k = 3. k = 4 is used in the paper.
* nonlinearK4bs.R. Differs from nonlinearK4.R in that it adds a test using GAMs with bs = “ts” as an option in the smooth function s(). This uses thin plate regression splines (as does the default (which is bs = “tp”), but with a modification to the smoothing penalty (quoting directly from the smooth.terms help page in R) “so that the null space is also penalized slightly and the whole term can therefore be shrunk to zero”. There seems to be a bug in this program, because the computation of a.gam.bs has the argument ‘familty’ instead of ‘family’ in the call to gam().
* nonlinearK4I.R. Differs from nonlinearK4.R in that Incentive is used as the explanatory variable, and the deviance explained by the GAMs is also captured in the output.
* nonlinearK4bsI.R. Does to nonlinearK4I.R what nonlinearK4bs.R does to nonlinearK4.R, including the ‘familty’ bug.

Given that the paper reports using thin-plate regression splines, fitting Richness to a smooth of Incentive, also checking a GAM with k = 4, and that the nonlinear\*bs.R programs have a bug in the gam() call to construct functions with bs = “ts”, nonlinearK4I.R would have been my best guess for the program used to create the table. However, gam() calls will tolerate familty as an argument without complaint, and this does have an effect on the returned result (as might be expected). Further, the deviance explained that appears in the table in brackets after each scenario label only appears in LOBEC.clustq.nonlinK4bsIdevExpl.csv (the name of which suggests nonlinearK4bsI.R was used).

The table in the paper clearly does not contain output directly from the script. From the contents of various directories, the CSV file output by the script has been loaded into Excel, and converted to a spreadsheet, with formulas used to compute the results of various tests. The CSV file LOBEC.clustq.nonlinK4bsIdevExpl.csv features as the third sheet in LOBEC.clustq.nonlinK4bs.xlsx, in which column H (dev.expl.gam) contains the values used in the table. However, the IF() formulas used in Excel (columns AX, AY and AZ) do not correspond to results reported in the paper. (I wonder if the table was loaded in R and generated that way.) However the table was produced, I have been able to reproduce the results using the LOBEC.clustq.nonlinK4bsIdevExpl in LOBEC.clustq.nonlinK4bs.xlsx (now saved as LOBEC.clustq.nonlinK4bs-20150126.xlsx) with the following formulas (bearing in mind that the ordering of the scenarios is slightly different in the spreadsheet than in the table):

|  |  |  |
| --- | --- | --- |
| Test | Excel formula | Condition expression using column headings |
| 1 | =IF(AND(E3>3,F3<=$B$1),"\*","-") | edf.gam > 3 &&  smooth.gam.P <= 0.0001 |
| 2 | =IF(AND(AU3<$B$1,AP3>10000,D3<Y3),"\*","-") | anova.gam.P < 0.0001 &&  diff.gam.klm > 10000 &&  ssq.gam < ssq.gam.klm |
| 3 | =IF(B3<=W3-2,"\*","-") | aic.gam <= aic.gam.klm – 2 |
| 4 | =IF(AND(AW3<$B$1,AR3>10000,D3<AM3),"\*","-") | anova.lm.P < 0.0001 &&  diff.lm > 10000 &&  ssq.gam < ssq.lm |
| 5 | =IF(B3<=AK3-2,"\*","-") | aic.gam <= aic.lm – 2 |

Since this spreadsheet contains a series of columns pertaining to the bs = “ts” results (AD-AJ, AQ, AV) I conclude that nonlinearK4bsI.R was used to generate the CSV file used in the results. Since none of the reported results use data from the buggily computed bs = “ts” GAM, the existence of this bug is not a problem as far as misreported results in the paper are concerned. I suspect from the nomenclature that the file LOBEC.clustq.csv was used as input to this script to generate the file LOBEC.clustq.nonlinK4bsIdevExpl.csv. LOBEC.clustq.csv is the output from the processing stage, including some columns from other analysis that has not been used here, tidying to remove runs with high expenditure/bankruptcy, and columns added as aliases for the other columns (especially those pertaining to scenario variables) to enable their data to be displayed succinctly. This processing will have been done manually using R (i.e. interactively using the R terminal). We should expect LOBEC.clustq.csv to have been used as input to the other scripts in the analysis stage (this is confirmed for figure 5 below). Looking at differences between LOBEC.clustq.csv and the output from analysege\_gpLU2.pl with options 8 and 9 (concatenated into a single file) should tell us something about the manual steps that were taken.

### figure 5

The treehist\* family of Perl scripts were used to generate the output. The program opens a call to R and processes the output, and a diagram built using PowerPoint from the results. The relevant PowerPoint file seems to be Tree Figure 2b.pptx, and here I seem at least to have had the sense to use the ‘notes’ part of the slide to record the command I used and a transcript of the output:

$ ./treehist3.pl -cp **0.0075** LOBEC.clustq.csv LOBEC.rpart3Xfr.pdf Richness G,M,B,A,X

There are three versions of treehist\*.pl, and although from the above treehist3.pl was used (and the arguments to it included), it might be worth just recording some of the differences. treehist2.pl adds to treehist.pl functionality to set the cp argument to the R rpart() call to a non-default value, and includes boxplots of land use occupancy in the results. treehist3.pl seems to had minor cosmetic changes to treehist2.pl, including using a separate PDF file to plot all the graphs at each leaf node (auto-added –leaves before the .pdf suffix).

treehist3.pl has four command-line arguments: the results file (LOBEC.clustq.csv), the PDF to create (LOBEC.rpart3Xfr.pdf – which means leaf nodes will be saved to LOBEC.rpart3Xfr-leaves.pdf), the response variable (Richness), and a comma-separated list of explanatory variables (G,M,B,A,X, where G is short for Government, M for Market, B for Break-Even Threshold, A for Aspiration and X for expenditure; note that LOBEC.clustq.csv already contains columns G, M, B and A – which will have been added after processing – the X column is added to the data as an alias for Expenditure whilst treehist3.pl is interacting with R).

Various command-line options are available. –R gives the location of R, -Rargs arguments to R (--vanilla is the default), -tmp is a working directory (/var/tmp by default), -asfactor does not the response variable as a factor (default is to do so), -cex, -bex, -margin, -uniform and –size make cosmetic control adjustments to the various plots, and –cp sets the complexity parameter for rpart(). The value used (0.0075) is lower than the default 0.01, and the reported decision tree is therefore bushier than would otherwise have been the case.