SUMMARY

This report describes algorithms implementing the *Rapfish* rapid appraisal technique for fisheries status. The algorithms are coded as software as a VBA 'add-in' for Microsoft's Excel spreadsheet (versions 97 to XP) that calls a Dynamic Link Library (DLL). A *Rapfish* toolbar added to Excel allows analysis for a specified evaluation field to be run from a standard interface form, which allows different options for standardisation and also contains options for running leverage and Monte Carlo error analyses. Graphical and tabular output is provided into a working spreadsheet containing a copy of the data, from which results may be transferred to the user's spreadsheet for storage. Data and *Rapfish* anchor points must be entered in a particular layout and format described here. The report presents programming details of the *Rapfish* multidimensional scaling ordination, scaling, anchoring and standardisation methods, leverage analysis of each scored attribute, and Monte Carlo simulations to evaluate uncertainty. Prospects for the further development of the *Rapfish* algorithms are discussed and a version of *Rapfish* that employs a multiplicative utility function is also described.

INTRODUCTION

Rapfish is a statistical technique for rapid appraisal of the relative status of entities (=fisheries), judged quantitatively against pre-defined sets of attributes grouped into 'evaluation fields' or disciplines. Approximate scores for each attribute are awarded on a scale from the worst to the best possible imaginable: subsequent ordination is referenced to this scale, from 'good', or 100% the best possible score, to 'bad', or 0%, the worst possible score[1] (See Figure 1). *Rapfish* has been under development at the Fisheries Centre at the University of British Columbia since 1998 (the first published paper was Pitcher *et al.* 1998). The most general published introductions to the technique are Pitcher and Preikshot (2001) and Alder *et al.* (2000), which contain examples and some numerical validations of the technique. A list of papers published up to the end of 2003 using the *Rapfish* technique is provided in Annex 1. The reader of this report is assumed reasonably familiar with this work.

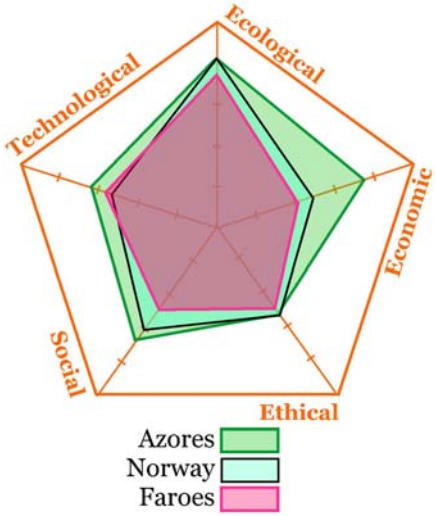


Figure 1. Example of a *Rapfish* analysis for three fisheries in five evaluation fields presented simultaneously as 'kite' signatures. Outer rim represents best possible scores. (Note: *Diagram prepared with Coreldraw, not Excel.*)

The original rationale for developing for *Rapfish* was to evaluate sustainability, and examples from that modality are largely used in this document. Fisheries scientists grade fisheries according to a large set of 'attributes'. Attributes are grouped in ecological, economic, social, technological, and ethical categories, or 'evaluation fields', so that 'sustainability' can be considered from various points of view. The *Rapfish* technique is flexible such that other modalities of status may be used, such conformity with a set of specified objectives or compliance with a code of conduct (e.g., Pitcher 1999). *Rapfish* applies a statistical ordination technique called Multi-Dimensional Scaling (MDS) to reduce the NxM matrix of fisheries statistics for N fisheries and M attributes into a N x 2 dimensional space which has similar distance properties as the N x M statistics. In this 2D attribute space, one dimension (x-axis) is the score representing the statut (degree of sustainability) from 'bad' to 'good', and the other dimension (y-axis) represents other factors, unrelated to sustainability (or whatever status is being scored), which distinguish fisheries.

[1] Some have argued that the use of the terms 'good' and 'bad' is unscientific and that these labels should be 'best' and 'worst' (D. Policansky, pers.comm.). But, since the terms are defined and always used in parentheses, we think this is a semantic point.

The MDS routine ALSCAL in the statistical package SPSS was used in the development and testing of the *Rapfish* technique. SPSS batch programming facilities software were written (Kavanagh 1999) to automate the *Rapfish* procedure, including routines for attribute leveraging and Monte Carlo error analysis. Problems with this software were inflexibility and awkwardness in re-configuring parameters due to limitations in the SPSS command language.

This report describes a more portable and easy-to-use *Rapfish* software implementation, implemented in Microsoft Excel and its programming language, Visual Basic for Applications (VBA). Excel is a popular and low-cost application and the majority of fisheries scientists are familiar and comfortable using it for statistical data analysis. The original ALSCAL FORTRAN code for multi-dimensional scaling (Young 2000) has been re-written and built as a dynamic link library routine (DLL) called from an Excel/VBA program. This Excel/VBA/FORTRAN implementation of *Rapfish* is portable (users need only Excel and not SPSS), is easy to programme for various repeat analyses such as leveraging and Monte Carlo, and has a handy graphical user interface to control processing and visualize results.

This report describes the new *Rapfish* software design and how to use it. The report is organized as follows:

An introductory section, including definition of the scope of the *Rapfish* software development project and the understanding of the requirements for this fisheries analysis tool;
Software design description of the Excel/VBA *Rapfish* application, including descriptions of the processing algorithms, user interface, and software structure;
A user's guide to the Excel/VBA *Rapfish* application, including installation, run and test instructions;
Discussion of stability, error analysis, and interpretation of *Rapfish* ordinations;
Suggestions for future improvements to *Rapfish*;
Listing of the full VBA code.

Rapfish Software Development Requirements

The specified requirements for developing the *Rapfish* software were:

- Work with *Rapfish* researchers to define features an Excel-compatible interface for a *Rapfish* software package appropriate for fisheries researchers and managers. Discussions and analysis of methodology with fisheries scientists (D. Tesfamichael, J. Alder, D. Preikshot, C. J. Walters);
- Port a FORTRAN version of ALSCAL (as used in SPSS multi-dimensional scaling, exactly as used in previous *Rapfish* analyses, to an Excel-accessible 'add-in' routine so that *Rapfish* can be run inside Excel independent of SPSS statistical package;
- Add user interface VB form in Excel to facilitate data entry, data checking, parameter entry, plotting, etc. Work with users to refine the design;
- Test method to ensure numerically consistent with SPSS-based *Rapfish;*
- Write Rapfish scoring error Monte Carlo and attribute leveraging software inside Visual Basic, so that Rapfish users can assess the confidence in the ordination;
- Document the new *Rapfish* software;
- Add VBA automatic plotting software for basic *Rapfish*, Monte Carlo, and attribute leveraging;
- Calculate the median estimate for the Monte Carlo runs and plot;
- Calculate and plot error bars for Monte Carlo (in addition to the raw 'scatter' plots) to quantify the statistics of the scatter and to estimate median ordination;
- For Monte Carlo, add user-defined error range for each attribute in the scoring spreadsheet, in addition to blanket percentage error option;
- Experiment with different ways of standardizing the raw data so that registration is similar when separate *Rapfish* runs are made with the same set of attributes and anchor/reference fisheries.

In the next section we describe the Excel 'add-in' software that has been developed to date, and is currently available for download from the 'Rapfish' pages of the UBC Fisheries Centre web site.

RAPFISH EXCEL VBA SOFTWARE DESIGN DESCRIPTION

Software architecture

Figure 2 illustrates the main components of the *Rapfish* software. The ovals show the *Visual Basic for Applications* (VBA) code modules, each containing one or more VBA subroutines. The **Main_Initialize** routine in the Main module is the entry point for the program, which causes the **RapfishForm** user interface to open. The VBA code associated with **RapfishForm** handles parameter entry, storage, and initiation control of the analysis via three Run buttons, one for each of the main routines: the basic *Rapfish* analysis, attribute leveraging, and Monte Carlo error analysis. The FORTRAN DLL **g77ALSCAL.dll** is used by all three of the processing routines to run Multi-Dimensional Scaling.

Rapfish scoring data is read from the **RapScores** spreadsheet. Analysis output and plots are written to one of three spreadsheets **RapAnalysis**, **Leveraging**, or **MonteCarlo** depending on which Run button is selected. The RapAnalysis worksheet is repeatedly written to for each iteration of the leveraging and Monte Carlo processing routines.

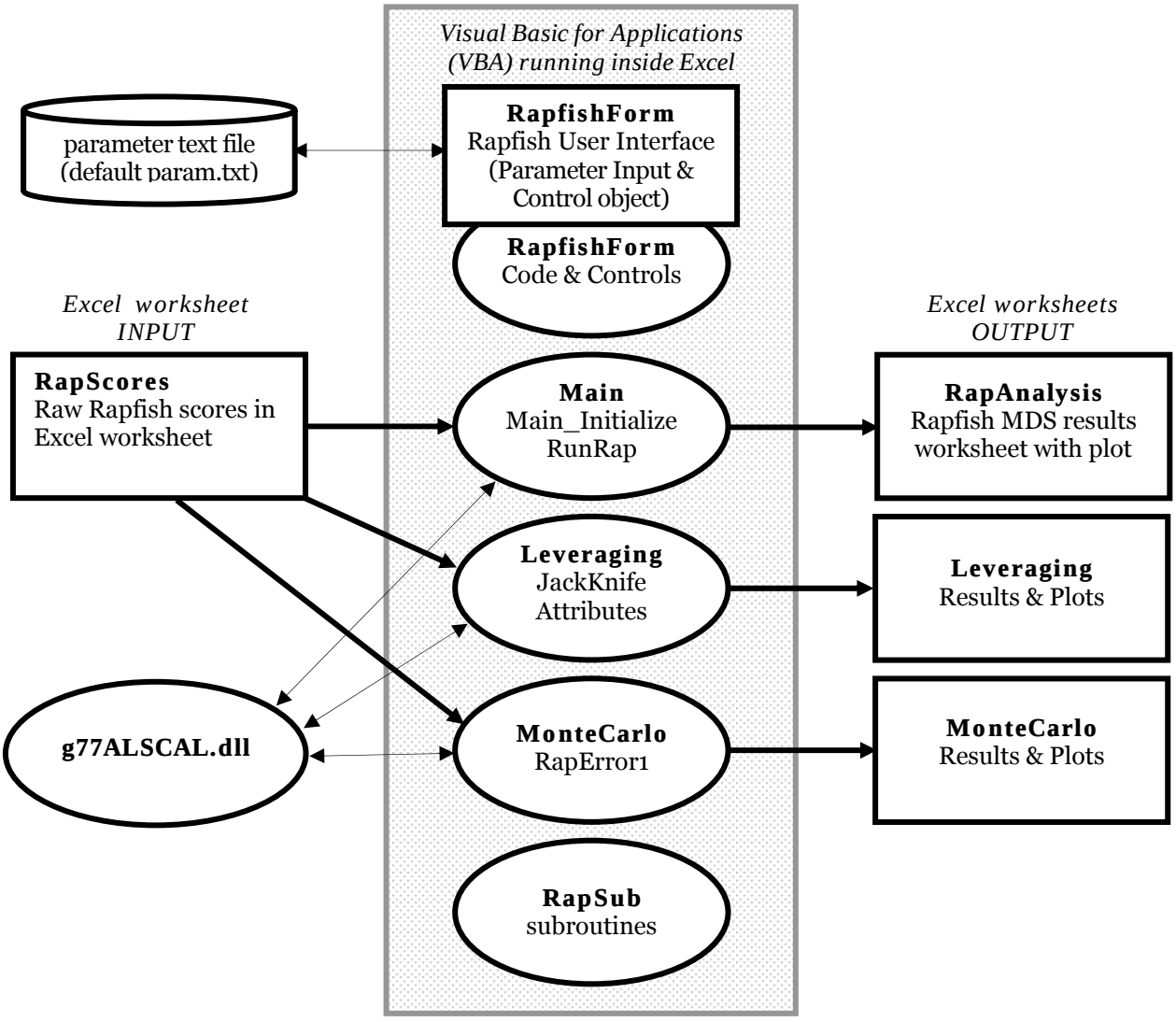


Figure 2. Block diagram showing *Rapfish* Excel/ VBA/ FORTRAN software architecture.

User Interface Form: enter data, run *Rapfish* analysis and options

To initiate a *Rapfish* analysis, enter Excel and click the *Rapfish* toolbar button:

The following user form (Figure 3) will appear (see page 23 if it does not). NOTE that values must be entered into all boxes on the form even if you are not using all the options.

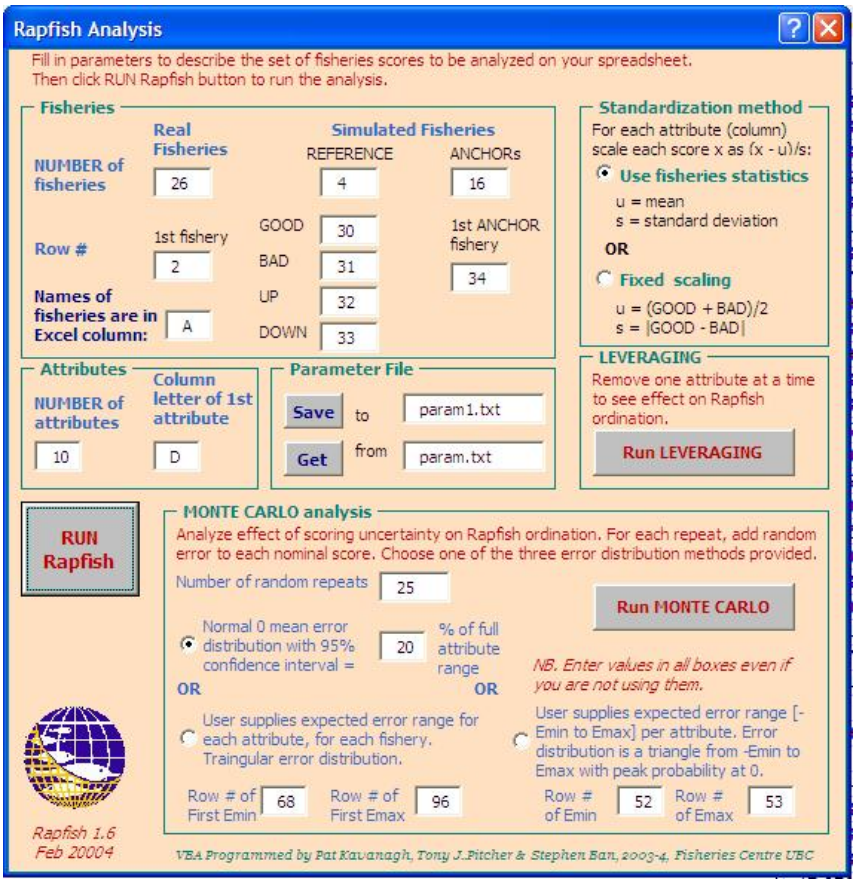


Figure 3. The *Rapfish* VBA interface: run analysis and options choice form (version 1.6, February 2004) which appears when the *Rapfish* toolbar button is clicked. Form is shown as populated from 'Ecological' data in 'Red Sea' test parameter file.

Column letters refer to Excel spreadsheet columns and row numbers refer to Excel spreadsheet numbers for the worksheet where you have stored the data for analysis (see pages 7 & 18 below). Fisheries are set out in rows and attribute scores are in vertical columns. NOTE that the data MUST be stored in the layout described in this handbook (page 7) and shown the example 'RedSeas.xls' file provided.

It is recommended that, for each data set you are working on, you create a special **operating file** for running your Rapfish analyses, and another **results file** into which to paste the results you want to save, since the operating file will be overwritten each time you run a new analysis. Your operating file could conveniently be based on a renamed Redsea.xls.

If **Get** is clicked and a valid file text file of that name is present in the working directory (see below for location) then the form will be populated with parameter values and option buttons values from the default parameter file **param.txt** (or your chosen file name). After populating the form, if **Save** is clicked, the parameter values for this analysis will be saved to a text file in the working directory with the name in the Save box.

The usr interface form has been tested under a wide range of conditions, but the *Rapfish* team would appreciate your reports of any problems.

VBA Code for the user interface form is listed in Annex 3, pages 69 to 72.

Rapfish Analysis Procedure with the User Interface Form

Parameter Input and Selection

The *Rapfish* userform (Figure 3) can be opened with parameters read from file **param.txt**. This file contains a default set of processing parameters for *Rapfish* analysis of the ecological attributes of the Red Sea data file **RedSea.xls**. For a different data set, you must enter appropriate values in the **Fisheries** and **Attributes** frames to read your data from the RapScores spreadsheet. Fisheries are set out in rows and attribute scores are in vertical columns. The data layout is as follows:

- starting row of real fisheries, reference fisheries, and anchor fisheries
- numbers of fisheries, reference fisheries, and anchor fisheries
- column letter of the names of the fisheries (or column with abbreviated names)
- column letter of the first attribute in the set to be analyzed
- number of attributes in the set

In formatting data for the **RapScores** spreadsheet, the main rules are:

- fisheries rows and anchor rows must each be contiguous groups, but need not be together
- reference anchor fisheries can be anywhere (row of each is defined)
- each evaluation field (= group of attributes) must be in contiguous columns

If in doubt, follow the formatting in the example **RedSea.xls** file.

The **Standardization** frame on the form specifies one of two types of standardization of the RapScores fisheries data. In **Use fisheries statistics**, each score x is Normalised as (x-u)/s so that each attribute is evenly weighted and differences among measurement scales is removed. By default u = mean and s = standard deviation of each attribute column of fisheries (real+anchor+reference). The standardization will change with the statistics of the data. Alternatively, standardization can be done with **Fixed scaling** defined by the 'bad' and 'good' scores for each attribute: u = (GOOD+BAD)/2 and s = |GOOD-BAD|. The standardization will not change with the set of fisheries in this case. The fixed scaling method of standardization may give better registration consistency for multiple *Rapfish* analyses with different sets of fisheries since the standardization scaling is the same for all. But differences among their mean scores may be thought to be realistic, in which case the normalisation technique will be better.

Alternatively, one could calculate the mean and sigma over *only* the reference and anchor fisheries, so that the standardization scaling stays fixed regardless of the set of fisheries analyzed, but this has not yet been tried. A future version of *Rapfish* could be programmed to generate the anchor fisheries automatically from the reference 'good'/'bad' and 'up'/'down' fisheries (the **Stabilizer** frame in earlier releases). At present (release 1_6), the **RapScores** data worksheet must contain the reference and anchor fisheries to be used for each set of attributes.

Anchor fisheries are of two types: the reference anchors 'bad','good','up','down', and a ring of anchors:

> 'bad', 'good': generated by all maximum or all minimum socres respectively;
> 'up' and 'down': first half of attribute scores maximum, second half score minimum, and vice versa;
> ring of anchors; a set of half minimum and half maximum scores, the pattern for each anchor shifted on one column from previous (see example in Redsea.xls).

Before running, save your parameter set by specifying a text file filename in the Parameter File frame and clicking **Save** to button. Next time, simply type in your parameter filename and click **Get** from button to load the same parameters (see page 22 as to where these files should be located).

Rapfish Algorithm

After all parameters have been entered into the userform, click the **RUN Rapfish** button to cause subroutine **RunRap** in the **Main** module to be executed.

The **RunRap** processing algorithm is outlined in pseudocode below. VBA subroutines **CalcStats**, **Standardize**, **Proximities, Rotate**, and **FlipNScale** are described on pages 15 to 18, and the multi-dimensional scaling FORTRAN DLL **g77ALSCAL** is described on page 19. The full VBA code is listed in Appendix 3 (pages 40-72).

An example *Rapfish* output plot is given in Figure 4.

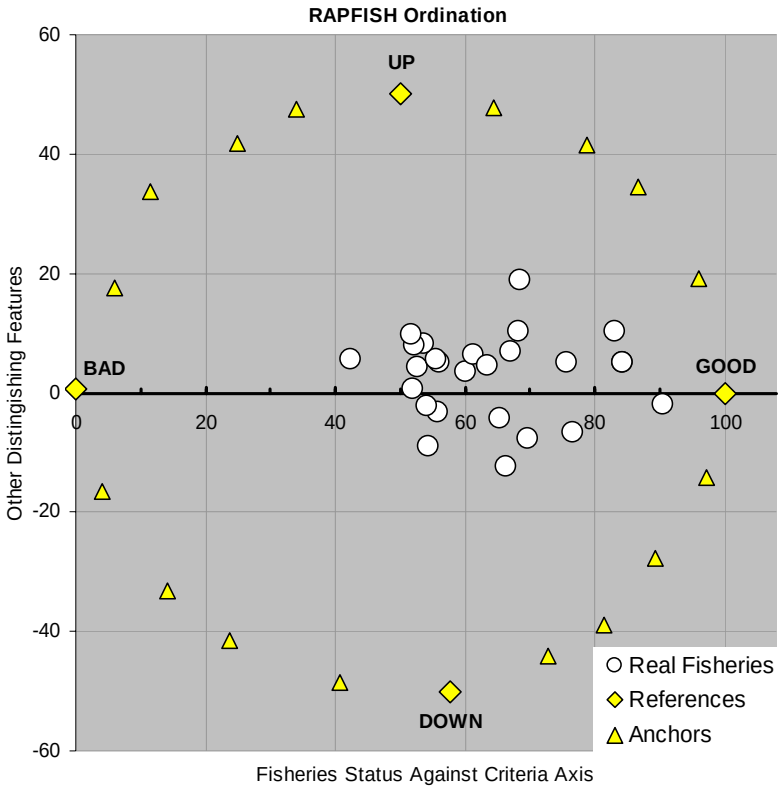


Figure 4. Example output from a *Rapfish* Ordination using the Excel software. Status axis is drawn horizontal by convention, increasing from left to right. Anchor points are shown. Note: Rapfish output graph format uses default Excel settings which may easily be customised by the user.

Anchor points are shown by diamond and triangle symbols. Reference anchor points are fixed at 100% ('good') and 0% ('bad') on the horizontal scale, and at -50 ('down') and plus 50% ('up') on the vertical scale. Ordination results are scaled to these extremes. Other anchor points ordinate in ring and serve to lock the analysis down against the problem of the upper and lower halves of the graph 'flipping' with small changes in input (see Pitcher and Preikshot 2001).

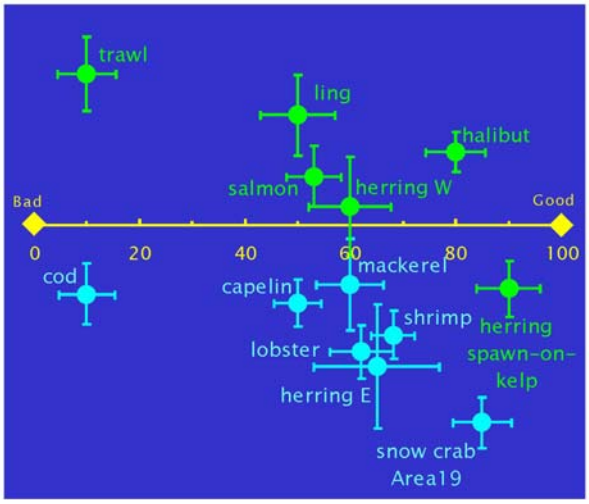


Figure 5. Example of 2-D Rapfish plot with fishery points labelled and Excel graph settings adjusted for Powerpoint presentation.

The ordination point for each fishery is plotted on the two first orthogonal axes of the MDS ordination. The horizontal axis is the status scale, running from zero to one hundred percent. The vertical scale expresses differences among fisheries, derived from the ordination of the scores, that are not related to status. Note that results from very similar scores may overlay each other.

Users may wish to label the data points with the names of the fisheries. For some obscure reason there is no facility to do this in any releases of Excel spreadsheet to date. **Chartlabl.xls** containing a VBA routine to do this is available for download from the Fisheries Centre website, the code and user form is listed in Appendix 4 (page 73), and an example is shown in Figure 5.

Pseudocode for **RunRap** processing algorithm.

```
Sub RunRap()
Load current parameters from RapfishForm into local variables
Fill matrix RapScores(i,j) with scores from selected fisheries and attributes
from RapsScores spreadsheet
Calculate standardization vectors mu(j) and sigma(j) for each attribute j, using
method selected in RapfishForm:
  1) Calculate fisheries statistics (subroutine CalcStats)
        mu(j) = sample mean over all fisheries
        sigma(j) = sample standard deviation over all fisheries
  2) Fixed scaling based on extreme scores BAD and GOOD
        mu(j) = (GOOD(j) + BAD(j))/2 for each attribute
        sigma(j) = |GOOD - BAD| for each attribute
Standardize each attribute column of RapScores (subroutine Standardize)
     Rap01(i,j) = (RapScores(i,j) – mu(j)) / sigma(j)
Calculate Euclidean Squared distance between each pair of fisheries (rows) in
Rap01 matrix to obtain a NxN Distance matrix (subroutine Proximities)
Invoke g77ALSCAL Fortran DLL to do metric Multi-dimensional Scaling on the
Distance matrix to create a Nx2 dimension matrix MDSout which has approximately
the same distance properties as the input matrix Rap01, except for a monotonic
transformation so that rank order of distance is maintained
(N = total number of fisheries). See section 3.6 for g77ALSCAL parameters.
Rotate the 2D MDSout matrix so BAD to GOOD vector is horizontal, left to right
(x-axis) to obtain MDSrotate (subroutine Rotate)
Correct for ambiguous reflected ordination solutions by “flipping” MDSrotate
vertically (y coordinate), as needed, so that UP reference fishery is above DOWN
reference fishery on the 2-dimensional ordination:
    If MDSrotate(UProw,2) < MDSrotate(DOWNrow,2)
    Then MDSfns(i,2) = -MDSrotate(i,2) for all fisheries i
    Else MDSfns(i,2) = MDSrotate(i,2) for all fisheries I
Then, scale MDSfns arbitrarily so BAD to GOOD x-axis range is 0 to 100, DOWN
fishery y-axis -50 and UP fishery y-axis is +50. (subroutine FlipNScale)
Write results to output spreadsheet RapAnalysis with fisheries names, including:
   - MDSout = the raw MDS ordination (before rotate, flip, scale operations)
   - MDSrotate = MDSout rotated so BAD to GOOD vector is horizontal
   - MDSfns = MDSrotate flipped so UP above DOWN and scaled
   - Stress
   - Squared Correlation (RSQ)
   - number of iterations
   - iteration history (stress and ∆stress for each iteration)
   - memory required (words)
   - rotation angle (between MDSout and MDSrotate)
   - g77ALSCAL return value (0 for normal exit, otherwise an error
     or warning code indicating the reason for g77ALSCAL exiting)
Plot rotated, scaled, and flipped Rapfish 2D ordination using Excel plot
facilities. Indicate reference and anchor fisheries with different colours.
```

Attribute Leveraging

Attribute leveraging analysis shows the effect of removal of one attribute at a time on the ordination of the fisheries. For M attributes, the *Rapfish* analysis is run M+1 times, first with all attributes and then M times with a different attribute removed with each iteration. Standardization is perfomed once with all attributes present and the attribute removal is done on the standardized data. This type of analysis is also called 'JackKnife' in the statistical literature (Alder *et al.* 2000). A form of JackKnife could also be done by removing fisheries one at a time, or in groups, to see the effect on the ordination stability, but this is not yet implemented.

Once all the basic *Rapfish* parameters have been selected, click the **Run Leveraging** button. Results will appear in the **LeverageAttributes** worksheet. Examples of the two plots generated are shown in Figures 6 and 7.

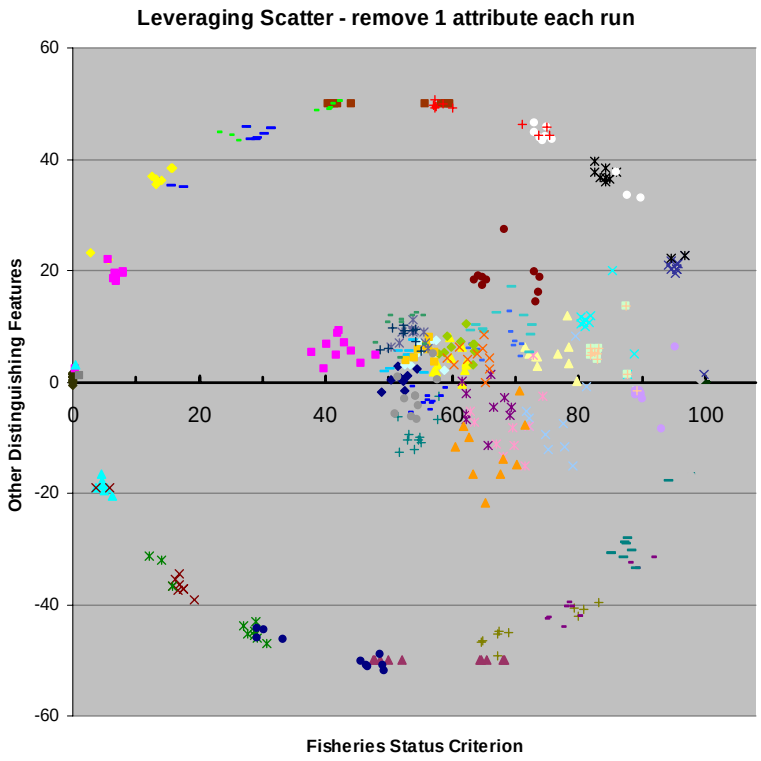


Figure 6. Graphical output from *Rapfish* Leverage analysis where attributes are removed one at a time. Note ring of anchor points and clusters of fisheries ordination points. For further discussion, see text. Colours may not show in black-and-white

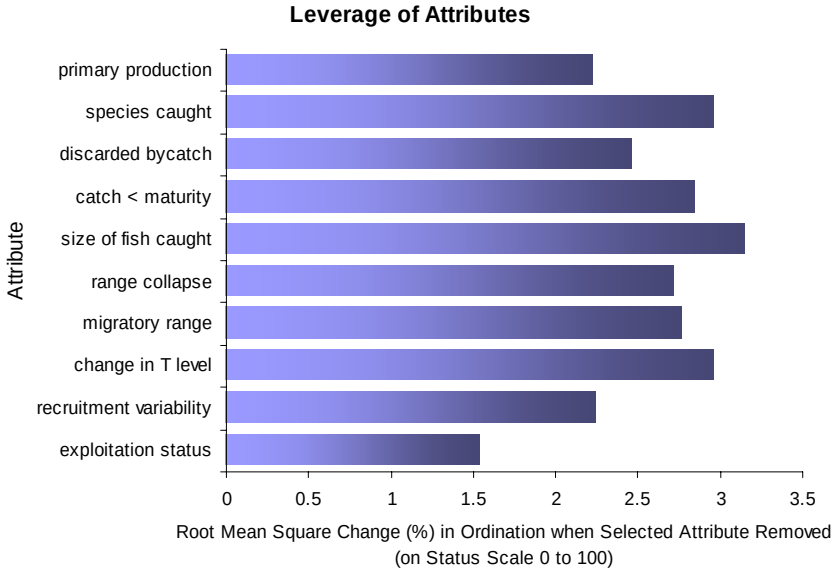


Figure 7. Leverage (sensitivity) analysis in *Rapfish*. Bars show percentage root mean square change in ordination score on the status axis when each attribute is removed from the ordination. For presentation, it would be best to sort the attribute bars by size.

An interesting observation from the Leveraging Scatter plot in Figure 6 is that some of the anchor and reference fisheries (the points in the circular formation) seem to have registered at an adjacent anchor point for several of the attributes. This is another example of ambiguous MDS solutions, similar to the vertical flipping problem. Thus, some of the leveraging variation in the real fisheries may actually be due to the absolute registration ambiguity of the MDS solutions, rather than the missing attribute. To correct this problem we might consider detecting an angle

ambiguity by noting the phase angle of all of the anchors and reference fisheries (not just the BAD and GOOD fisheries as is done now), and then estimating an average correction angle to minimize the angular error, and rotating the fisheries by this angle. This is not yet implemented.

In Figure 7, the length of each bar shows the percent difference that removing that attribute has on the ordination value along the status axis. It is roughly the average radius of the leveraging scatter for the status dimension. The longest bar indicates the attribute which seems to have the most influence on the fishery ordination. For presentation it is recommended that the attributes be sorted from the largest leverage to the least.

Monte Carlo

A 'Monte Carlo' analysis is a statistical simulation method to evaluate the effects of random error on a process, and to estimate the 'true' value of a statistic of interest. Random errors from computer random number generators are added to the phenomena under test (like a roulette table) and a 'scatter' plot and other statistics generated. In addition to estimating the most likely *Rapfish* ordination positions, a Monte Carlo procedure is useful for studying the:

- effects of scoring error caused by imperfect knowledge of the fisheries or misunderstandings of the Rapfish attributes and scoring guidelines
- effects of scoring variation due to differences in opinion or judgement by different people
- stability of MDS method for successive runs (quality of the anchor and reference stabilization)
- incomplete convergence (high stress)
- data entry errors or missing data
- ambiguous (flipped or rotated) solutions

Monte Carlo Parameters

One of two types of random error distributions may be selected in the user interface:

1. *Normal (Gaussian) Error* with user-specified 95% confidence interval expressed as a percentage of the full attribute score range for each attribute;

2. *Asymmetric Triangular Error* distribution with user-entered attribute score minimum and maximum error limits for each attribute, or for each fishery for each attribute (see entries in **RapScores** spreadsheet).

For either case, enter the number of iterations required (25 is the default, 100 are recommended). Click the **Run MONTE CARLO** button when ready to do the analysis. Results will appear in the **MonteCarlo** spreadsheet, and should be copied to your results file for storage.

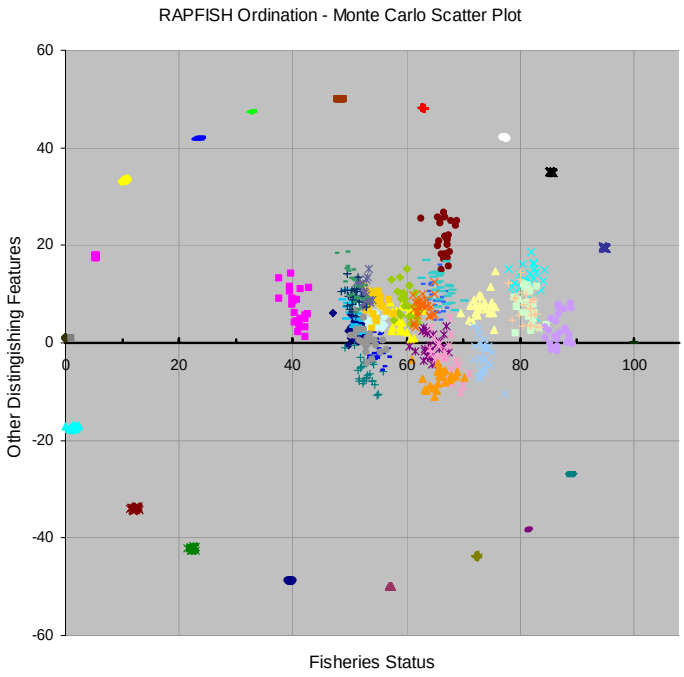


Figure 8. Monte Carlo error analysis scatter plot for 25 repeats adding random zero mean Normally distributed error with 95% confidence interval set to 20% of full range for each attribute. Ring of anchor points would not normally be shown (see text). (NOTE: Colours may not show clearly in printed report).

Monte Carlo Algorithm

Three types of plots are produced by the Monte Carlo routine:

1) A scatter plot showing the ordination location for each fishery for every *Rapfish* Monte Carlo random repeat, overlayed on the same plot (Figure 8).
2) The median estimate of the Rapfish ordination location for each fishery, with error bars set to estimated 95% confidence interval in the median estimate (the larger the number of repeats, the better is the estimate of the median ordination for the fisheries) (Figure 9).
3) The median estimate of the Rapfish ordination location for each fishery, with error bars on the plot set to the 50%tile region (inter-quartile range) of the Monte Carlo scatter for the given input scoring noise statistics (Figure 10).

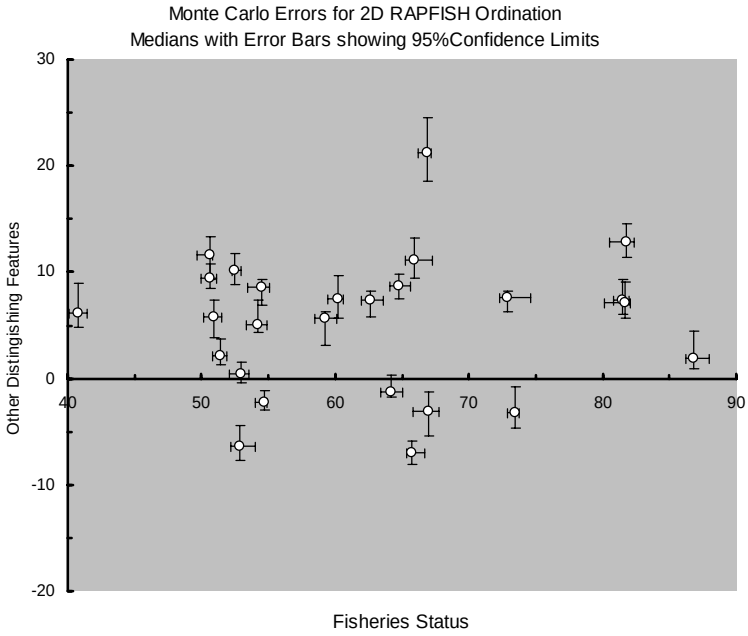


Figure 9. Close-up of Monte Carlo plot produced by *Rapfish* software, showing median estimates of ordination positions with error bars indicating 95% confidence intervals on the medians. (From a run with 25 repeats.)

Figure 8 is a plot of the 'raw' Monte Carlo ordination scatter for all of the repeats. On the output spreadsheet, a different colour is used for each fishery. Notice the 'tight' distribution of plotted points for the reference and anchor fisheries; scores for these fisheries have not been perturbed. The real fisheries show scatter due to the Monte Carlo perturbed scores. Figure 9 is a plot of the median ordination value for each fishery, with error bars showing 95% confidence limits. Asymmetry derives from the use of a triangular PDF for the Monte Carlo errors. Figure 10 shows a plot of the median ordination score for each fishery with error bars indicating the inter-quartile boundaries, also asymmetrical. The plots in Figures 9 and 10 look similar, but in practice the confidence limits on the median become much smaller than the interquartile range as the number of repeats increases. Pseudocode for the Monte Carlo algorithm is given below (details of the *Rapfish* logic is omitted for clarity of Monte Carlo logic), with descriptions of the two types of random number generation and procedures for calculating medians and error bars. The VBA code is listed in Appendix 3.

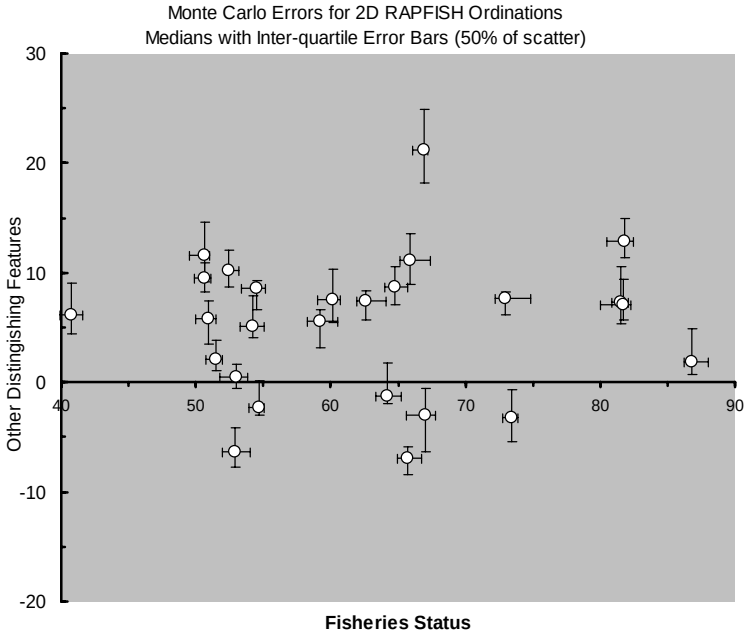


Figure 10. Part of one of the Monte Carlo ordination plots produced by the *Rapfish* software. Each fishery is represented by its median position calculated from the scatter of points produced by the Monte Carlo runs shown in Figure 6. Bars indicate interquartile ranges, or 50% of the scatter. (From a run with 25 repeats.)

Pseudocode for **RapError1** Monte Carlo algorithm.

```
Sub RapError1 (Rapfish Monte Carlo)

Load current parameters from RapfishForm into local variables,
including Monte Carlo error distribution type, spreadsheet row numbers for Emin
and Emax, and # random repeats (Nspins)
Fill matrix RapScores(i,j) with scores from selected fisheries and attributes
from RapsScores spreadsheet
Calculate standardization vectors mu(j) and sigma(j) for each attribute j, using
statisical or fixed scaling method, selected in RapfishForm:
Call Standardize
Similar to basic Rapfish:
Call Proximities to calculate the fisheries distance
Call g77ALSCAL do metric Multi-dimensional Scaling.
Call Rotate
Call FlipNScale
   → to produce nominal ordination matrix Rap01
If doing user-entered error limits, read user-entered error limits Emin(j) and
Emax(j) for each attribute j from the RapScores spreadsheet
For each random repeat 1 to Nspins:
   Generate independent random samples from selected error distribution
   for each real fishery and attribute score and add to nominal ordination
   (keep reference and anchor fisheries clean, no noise)
           NoisyRap01 = Rap01 + noise
   Truncate the score+noise samples NoisyRap01 to force the values
   outside BAD and GOOD limits to BAD or GOOD limits
   Do Rapfish, as usual:
   Call Proximities to calculate the fisheries distance
   Call g77ALSCAL do metric Multi-dimensional Scaling.
   Call Rotate
   Call FlipNScale
      → to produce perturbed ordination matrix Rap01
   Write 2D MDS results to output spreadsheet MonteCarlo, as rows.
Plot Monte Carlo scatter plot (each fishery is a column in spreadsheet)
Calculate median ordination in each dimension for each fishery over all repeats.
Calculate the estimated confidence interval on the median. Plot median with
confidence interval error bars.
Calculate the degree of scatter in the ordinations by measuring the inter-
quartile limits on the scatter (where 50% of the scatter points are contained).
Plot the median ordination again, this time with error bars set to the inter-
quartile limits.
```

```
DEFAULT scoring noise distribution:
Generate Normally distributed scoring error with 0 mean and standard deviation
sigma selected so that the 95% confidence interval is X percent of the full
attribute range, where X is selected by user. By default X=20%.
From Gaussian tables, 95% 2-sided confidence interval has a width of 3.92*sigma,
so we can calculate the noise sigma for attribute j as:
     noise_sigma = (X/100) * |GOOD-BAD attribute j score| / 3.92
Use subroutine GRV to generate a Gaussian random sample with 0 mean and
noise_sigma standard deviation.
Also write out the equivalent Emin = Emax in the original attribute units to
Rapscores spreadsheet at EminRow + 5.
   Emin = Emax = (3.92/2 * X)*sigma + mu
where mu and sigma are the values used to standardize the original scores.
```

ALTERNATIVE scoring noise distribution:

Read Emin and Emax from the RapScores spreadsheet at rows EminRow and EmaxRow.
Generate error random sample using an assymetric triangle PDF with peak
probability at 0, minimum value -Emin, and maximum value Emax
(subroutine **TriangleRV**).

VBA Subroutines

In addition to various Excel Worksheet and VBA standard functions, the **RapSub** module defines custom VBA subroutines invoked by the user interface, basic *Rapfish*, Monte Carlo, and Leveraging routines. Each subroutine is summarized briefly below. See the VBA code in Appendix 3 for further programming details and variable input/output.

CalcStats – Calculate sample mean (mu) and standard deviation (sigma) of each attribute column j of matrix X:

$$\text{mu}(j) = \sum_{i=1 \text{ to } N} X(i,j) / N$$

$$\text{sigma}(j) = \left\{ \left[\sum_{i=1 \text{ to } N} X(i,j)^2 - \left[\left(\sum_{i=1 \text{ to } N} X(i,j) \right)^2 \right] / N \right] / (N-1) \right\}^{1/2}$$

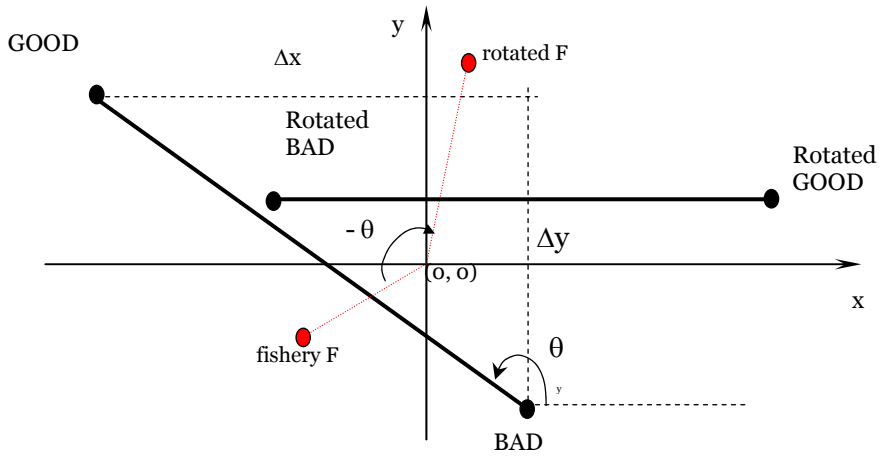
where N = # fisheries.

Standardize – Normalize the attribute scoring scales for each column j of input matrix **X**:

$$\mathbf{X}01(i,j) = \mathbf{X}(i,j) - \text{mu}(j)) / \text{sigma}(j)$$

Assuming a Normal (Gaussian) distribution of fisheries scores in each attribute column and assuming mu and sigma are good estimates of mean and standard deviation of each attribute column, then causes each attribute column of standardized scores X01 to have 0 mean and standard deviation 1.

Rotate – Use trigonometry methods to rotate the 2-dimensional input matrix **V** to obtain matrix **Vrotate** which has the BAD to GOOD case vector horizontal (parallel to x-axis) with BAD on left and GOOD on right:



1) Calculate angle of vector from BAD to GOOD:

$\Delta x = \mathbf{V}(\text{Igood}, 1) - \mathbf{V}(\text{Ibad}, 1)$ Igood = row # of GOOD case; Ibad = row # of BAD case

$\Delta y = \mathbf{V}(\text{Igood}, 2) - \mathbf{V}(\text{Ibad}, 2)$

$\theta = \tan^{-1}(\Delta y / \Delta x)$ (ensure correct quadrant for inverse tangent)

2) Rotate matrix **V** by –θ

For each row i = 1 to N of matrix **V**:

a) Convert **V** from rectangular (x, y) to polar coordinates (magnitude, phase)

$x = \mathbf{V}(i,1)$ $\quad y = \mathbf{V}(i,2)$

$mag = (x^2 + y^2)^{1/2}$ $\quad phase = \tan^{-1}(y/x)$

b) New phase = phase – θ (again, check for correct quadrant for phase angles)

c) Switch back to rectangular coordinates with new phase:

Vrotate(i,1) = mag * cos(new phase)

Vrotate(i,2) = mag * sin(new phase)

Proximities – Given an input matrix **X** of (standardized) fisheries scores (rows are fisheries, columns are attributes), calculate Euclidean distance squared for each pair of fisheries rows i and j over all attribute columns k:

$$\mathbf{Seuclid}(i, j) = \sum_{k = 1 \text{ to } \# \text{ attributes}} \{X(i, k) - X(j, k)\}^2 \qquad \text{evaluated for } i = 1 \text{ to } N-1, \text{ and } j = i \text{ to } N$$

Output matrix **Seuclid** is of dimension NxN where N = number of fisheries; the diagonal and symmetric upper triangle values are not calculated (just set to zero) to reduce computations. The **g77ALSCAL** routine expects a distance matrix of this form.

FlipNScale – Correct DOWN to UP positioning and scale axes.

Ambiguous, "mirror-image" solutions may occur because the MDS iterative calculation does not guarantee a unique ordination solution. With the anchor fisheries, ambiguous solutions are rare. Assume a vertical ambiguity if the UP reference fishery is less than the DOWN fishery in the vertical (y) coordinate. To correct such an ambiguity, "flip" the y-coordinate (second column) of all points in input matrix V to force UP above DOWN:

If **V**(Iup, 2) < **V**(Idown, 2)
Then **Vfns**(i, 2) = –**V**(i,2) for all fisheries I (flip (reflect) across x-axis)
Else **Vfns**(i, 2) = **V**(i,2) for all fisheries I (no flip)

Scale and shift **Vfns** arbitrarily so BAD to GOOD x-axis range is 0 to 100, DOWN fishery y-axis -50 and UP fishery y-axis is +50. Finally, shift BAD to GOOD horizontal axis to y=0

```
For i = 1 To N
        Vfns(i, 1) = 100 * (V(i, 1) - V(Ibad, 1)) / (V(Igood, 1) - V(Ibad, 1) )
        Vfns(i, 2) = ( 100 * (V(i, 2) - V(Idown, 2)) / (V(Iup, 2) - V(Idown, 2)) ) – 50
Next i
For i = 1 To N
         Vfns(i, 2) = Vfns(i, 2) - Vfns(Igood, 2)
Next i
```

Definitions: Iup = row # of UP fishery; Idown = row # of DOWN fishery
Ibad = row # of BAD fishery, Igood = row # of GOOD fishery

GaussianRV – Generate two independent random variables (RV) G1 and G2 from a Gaussian (Normal) random number generator with specified mean and sigma (standard deviation) using the 'Box-Muller' inverse transformation method. Use VBA **Rnd** function to create a pair of Uniform independent random variables U1 and U2 distributed evenly between 0 and 1. Then calculate the following functions of random variables U1 and U2:

$$r = \{-2 * \ln(U1)\}^{1/2}$$
$$\theta = 2\pi * U2$$

Note r (magnitude) is Rayleigh distributed and θ is (phase) Uniform distributed over 0 to 1. A pair of Gaussian random variables with 0 mean and unit standard deviation are calculated as:

$$X1 = r * \cos(\theta)$$
$$X2 = r * \sin(\theta)$$

Adjust mean and sigma:

$$G1 = mean + (X1 * sigma)$$
$$G2 = mean + (X2 * sigma)$$

GRV – Generate a Gaussian random variable with specified mean and standard deviation, using the VBA Uniform random number generator with a transformation. Samples from successive calls to **GRV** will be statistically independent. Use same algorithm as **GaussianRV**, but return only one Gaussian sample.

NOTE: **GRV** and **GaussianRV** are approximate. The quality of the Gaussian and independence of successive samples can be evaluated using the Excel Histogram and Correlation.

TriangleRV – Return a sample RV from an asymmetric triangle probability distribution function (PDF) with maximum probability at 0 and user-entered minimum and maximum values.

Inputs:

ErrLow = minimum expected error (-ve)
ErrHi = maximum expected error (+ve)

The RV is generated with the following algorithm:

1) Generate independent samples U1 and U2 from a Uniform (flat) distribution on range 0 to ErrLow. Use U1 and U2 to generate a sample from a lower triangle (negative only) PDF as T1 = -|U1 - U2|.
2) Generate independent samples U3 and U4 from a Uniform (flat) distribution on range 0 to ErrHi. Use U3 and U4 to generate samples from a lower triangle (positive only) PDF as T2 = |U3 - U4|.
3) Calculate the desired probability of being negative as the area of the lower triangle:
   P(RV<0) = ErrLow/(ErrLow+ErrHi)
4) Generate uniform RV sample F over 0 to 1.
5) Generate output TriangleRV PDF sample as:
   TriangleRV = T1 if F < P
   or T2 if F >= P

NOTES:
The PDF for the sum of 2 independent RVs is the convolution of the individual PDF's. This is how we get the triangle from 2 uniform RVs. The mean of TriangleRV is not exactly 0, but the most likely value is 0. The signs of ErrLow and ErrHi are ignored (code uses |ErrHi| and |ErrLow|) so the user

need not worry about sign. A test spreadsheet is available as a check of **TriangleRV**.

ParamSummary – Write out a summary of the *Rapfish* processing parameters to the **RapAnalysis** spreadsheet.

ExcelColNum – Convert an Excel column letter designator (1 or 2 letters) to a column number as follows:

A, B, C, ... Z, AA, AB, ... BA, BB, ...
To 1, 2, 3, ... 26, 27, 28, ... 52, 53, ...

With the column as a number we can use Worksheet().Cells to access spreadsheet cells numerically.

ExcelColLetters – Convert an Excel column number to a 1 or 2 letter designator as follows:

1, 2, 3, ... 26, 27, 28, ... 52, 53, ...
to A, B, C, ... Z, AA, AB, ... BA, BB, ...

With the column as a letter we can use Worksheet().Range to access spreadsheet ranges.

Multi-Dimensional Scaling Using ALSCAL FORTRAN DLL

The ALSCAL Multi-Dimensional Scaling (MDS) computer code was obtained by free download from Forrest Young's web site (Young 2000). This routine dates back about 20 years and is written in the FORTRAN language. Young's code can be run from a DOS window and uses file input for data and parameters, and outputs the results to a text file (including crude graphics). Young's FORTRAN ALSCAL routine cannot be invoked from Excel or any other language. The routines have evidently been integrated into the SPSS statistical package (Norusis 1990), and although we have used this software package for the initial Rapfish analyses, SPSS is expensive and not easily or efficiently interfaced to Excel spreadsheets or other programming languages (Kavanagh 1999). Early Rapfish analyses using SPSS, were time consuming and clumsy, and it was almost impossible to carry out Monte Carlo iterations. Schiffman *et al.* (1981) present a description of various multi-dimensional scaling techniques including ALSCAL

Microsoft Excel is a popular statistical analysis application at the UBC Fisheries Centre and elsewhere. One of the goals in *Rapfish* development is to make the software more accessible, so it was thought that the ability to run analyses from Excel would make the *Rapfish* procedure accessible to a wider group. In this current project, the original ALSCAL FORTRAN code was re-written and built as a dynamic link library routine (DLL) invoked from a Visual Basic program inside Excel. A number of changes were required in the source code to make the DLL work:

- replace the user input/output (terminal control from DOS window) with a DLL function interface
- replace all file input/output of data with variables passed into and returned from a DLL function
- build the code as a DLL
- write VBA code to declare and invoke ALSCAL for specified data matrix
- remove built-in plotting and replace with Excel/VBA plotting
- all error conditions detected inside ALSCAL converted to a function error return code, instead of the terminal and/or file print-out of an error message (error messages are then displayed via Excel/VBA)
- add comments
- fix (hard code) the ALSCAL (MDS) analysis options to those used for *Rapfish* as described below.

The parameter settings for the DLL FORTRAN ALSCAL were fixed to those previously determined to work well for the *Rapfish* procedure (Pitcher and Preikshot 2001) in order to simplify the DLL interface and to reduce the amount of time for testing and debugging of the ALSCAL routine. If other modes for the MDS in ALSCAL are needed in future, then the ALSCAL DLL can be modifed at that time. Appendix A-4 shows a section of the ALSCAL DLL code with the comments block defining each parameter and the values used; these match the *Rapfish* parameters used for SPSS MDS (see Norusis 1990, Chapter 25 for SPSS function parameters):

- Model – Ratio data option, (LEVEL = RATIO(1) with 1 degree of polynomial transformation)
- Symmetric proximity data matrix (SHAPE = SYMMETRIC - *default*)
- Conditional on subject (CONDITION = MATRIX - *default*)
- Scaling model Euclidean distance (MODEL = EUCLID - *default*)
- Other criteria all set to *defaults*, the most interesting of these being: CRITERIA = CONVERGE(0.001) STRESSMIN(0.005) ITER(30) CUTOFF(0) DIMENS(2,2) which means: Keep iterating to improve solution until the reduction in s-stress is less than 0.001 (or is negative) *or* 30 iterations have been completed *or* the minimum s-stress of 0.005 has been reached. Treat all negative distances as 0. Minimum and maximum number of dimensions for scaled solution is 2 as we want a 2 dimensional ordination.

The ALSCAL code does *not* include the calculation of the squared Euclidean distances between cases, implemented in the PROXIMITIES routine in SPSS. In SPSS the MDS options menu combines the PROXIMITIES and the ALSCAL routines together. Therefore a new **Proximities** subroutine was written in Excel/VBA which is functionally equivalent to the SPSS PROXIMITIES function for the following settings:

- Euclidean distance squared calculation on Interval data (MEASURE = SEUCLID)
- No standardization since separate VBA code does the data normalization (STANDARDIZE = NONE)

The new **Proximities** VBA subroutine in module **RapSub** calculates a distance matrix where each element is the sum of the squared differences between pairs of fisheries scores over all attributes. For N fisheries, the Distance matrix is of dimension NxN (but only half the entries are calculated as the diagonal is all zeros and the matrix is symmetric).

Normalization of the scoring data is done as a separate VBA processing step (see section 2.4). In SPSS MDS normalization (also called 'standardization' here) is performed as part of the PROXIMITIES operation.

To build the software, we had to first obtain a FORTRAN compiler. The freeware GNU FORTRAN compiler (g77) was downloaded from the Internet. The ALSCAL code supplied by F. Young was first compiled and tested as a DOS application. The MDS output was compared to SPSS MDS output for sample runs to ensure that the old FORTRAN ALSCAL is functionally equivalent to the SPSS ALSCAL routine. This test was successful, so we then re-wrote the FORTRAN code so it could run as a DLL callable from Excel (see above). Note however, that we did not do exhaustive comparison testing with many data sets so it is possible that there are subtle differences between the SPSS and FORTRAN versions of ALSCAL. We did compare the 'Red Sea' test data fisheries using Excel *Rapfish* and the SPSS *Rapfish,* obtaining the same results to within machine precision.

Annex 2a explains how to build the **g77ALSCAL.dll** file used by Excel *Rapfish*. Annexes 2b and 2c, 2d illustrate the ALSCAL DLL function syntax and how to invoke it from VBA routines. Source code provided with the *Rapfish* software is provided for more detailed understanding of the ALSCAL algorithm..

RAPFISH INSTALLATION AND USE

Installation Description

Rapfish software is provided as an Excel 'Add-In' that one installs into a Microsoft Excel 2000 spreadsheet application. An example fisheries scores spreadsheet file is provided to help users familiarize themselves with the program. Also supplied is a *Rapfish template* spreadsheet file with the *Rapfish* attributes, reference fisheries, and *Rapfish* 'anchor' fisheries included, but with fisheries scores and names left blank. *Rapfish* software is provided for internet download only. The software is in a Zipfile of about 900 kilobytes in size.

Rap1_x.xla – This is the Excel 2000 or XP 'Add-In' required to run *Rapfish*. Letter x is replaced with the version number of the release (as of Feb 2004, version 6, **Rap1_6.xla**). The Add-In is a Visual Basic for Applications (VBA) macro with code for the *Rapfish* user interface, reading Excel score spreadsheet, *Rapfish* processing, writing results to output spreadsheet, and plotting results in Excel. In XP you may find that *Rapfish* userform has to be closed (click top right) before you can access the Excel spreadsheets.

Rap97.xla – This is an alternative Excel'97 'Add-In' for those still running older Excel'97 spreadsheets. It was created by saving the *.xla file as an Excel'97 Add-In file (with Save As). Some minor function deficiencies occur with this version because of missing features in Excel'97. In particular, the *Rapfish* userform must be closed before you can access the Excel spreadsheets and you may not be able to move the form out of the way after running (during to missing Modality functionality). It is recommended that the Excel 2000 or XP version of *Rapfish* be used rather than this one.

g77ALSCAL.dll – This is the ALSCAL Multi-Dimensional Scaling FORTRAN program. The program originated from a DOS version (Young 2000) and was then rewritten and rebuilt as a Microsoft DLL (dynamic link library) routine that is accessible from Microsoft Excel or other Microsoft applications. See section 2.5 and Appendix A.

param.txt – This is a text file that stores the default processing parameters entered in the *Rapfish* spreadsheet at startup. File param.txt must be located in your default Excel directory path. The user can change the parameters and save to another file (or to **param.txt**). The **param.txt** supplied with the *Rapfish* software is set up to analyze the Ecological attributes for the **RedSea.xls** example file. To speed up repetitive analyses, it is recommended that you save parameter files for each of the attribute groupings in your analysis to avoid error-prone manual entry of parameters for each run.

RapTemplate.xls – This file is a *Rapfish* scoring template spreadsheet. The RapScores sheet contains the template that the user fills in with fishery scores. Each of the attributes in the standard 'sustainability' *Rapfish* is a column in this template, with categories ecological, economic, social, technological, and ethical. Reference fisheries (attributes all maximum GOOD, all maximum BAD, half GOOD, half BAD) and anchor fisheries for ordination consistency are provided in the spreadsheet. For other kinds of status evaluation using *Rapfish*, it is recommended that the user gains some familiarity with data layout for this example first (see page 7). In later versions of *Rapfish* it is possible that user data entry forms will be provided, but there is little point in this until more experience and feedback from use of the technique has been gained.

IMPORTANT NOTE: In order to compare ordinations from separate *Rapfish* analyses it is necessary to use the *same attributes*, the *same order* of attribute columns, and *the same set of reference and anchor* fisheries; these are all provided with the template so your *Rapfish*

analyses will all be standard if you do not change these (see page 7).

RedSea.xls – This file is a full example of the *Rapfish* procedure, including preliminary fisheries scores from the Red Sea (courtesy of Dawit Tesfamichael, 2001 and 2004), and with sample analysis output for the 'ecological' evaluation field.

Software Installation Steps

NOTE: none of the software has been tested on the versoin of Excel for Mackintosh computers.

Configure Excel for using Rapfish software

1) Copy the files from the Zip file to a suitable directory on your computer hard drive, for example C:\Rapfish\RapfishExcels. This directory should contain the files:

   g77ALSCAL.dll
   param.txt
   Rap1_6.xla (for Excel 2000 and XP)
   Rap97.xla (for Excel'97 only)
   RedSea.xls

   With the exception of the Redsea.xls example, these files will not be used – these are your clean copies for reference. You may want to copy the PDF file for this manual to another subdirectory, and create other Rapfish directories for your data and the results of your analyses.

2) Copy **g77ALSCAL.dll**, **param.txt**, **Rap1_v.xla** (for Excel 2000 or XP) from the \RapfishExcels to the Excel add-in directory. This is the root directory that XP expects you as the current user of this copy of Excel to be working from. (Note: This root may be in an unexpected place, such as under whatever Windows XP thinks your current identity is). For example, add-ins for your current 'identity's use of Excel are stored under the 'Documents and Settings' directory: (e.g., if you are currently working under the 'administrator' identity on the machine, C:\Documents and Settings\Administrator\Application Data\Microsoft\AddIns.) If you are working on a LAN workstation, ask your LAN adminstrator what your identity is and where its unique set of files are stored. By default, this will be your **working directory**. It is strongly suggested that you copy results out of the Rapfish working file into a seperately named results file as you run analyses. NOTE: *In different versions of the Windows operating system and Microsoft Office we have found a range of situations with regard to the location of your working directory –if the above does not work, we regret that you may have to experiment your computer.*

   In Excel'97, you may have to alter your default spreadsheet directory and copy the param, .dll and .xla files there. See Tools → Options → General to find the place where you enter the default file location directory. Open Excel and change the Excel 'Default file location' directory to the \RapfishExcels directory you just created.

3) Open Excel and go to the Tools → Add-Ins menu. Browse to select either Rap1_6.xla (in Excel 2000 or XP) or Rap97.xla (if you are using Excel'97). [In Windows XP, it may say that a file of that name already exists and asks do you want to replace it - say NO.] You should now see a Rapfish line listed in the Add-In list with an X marked beside it. As a further check, on opening the Visual Basic environment you should see the *Rapfish* VBA code listed. Here you may open each module of code to inspect if you choose. Unlike commercial add-ins, the Rapfish code is not 'locked', so fiddle with the code if you want, but be it at your own peril. In Windows XP, if an older named version of the Rapfish *.xla addin file is also be present in the add-in sub-directory, it may

be used even if you don't want it to. Make sure older versions are cleared out, although you may want to keep them elsewhere.)

NOTE: If the Add-In menu is greyed out, open a blank workbook, and you should then be able to do the *Rapfish* Add-In above. 'Enable macros' if it asks permission on entry to Excel (*Rapfish* runs as a VBA macro). The software will work best on the lowest level of Microsoft's 'macro security', especially if you want to try your own modifications, so you should have independent anti-virus software installed on the machine.

4) From now on, each time you open Excel you should see the *Rapfish* software listed as an Add-In. If you don't (and it is quite possible, especially in XP), go back and start again.

The Rapfish Toolbar and Trouble Shooting

To open the *Rapfish* user interface, run the macro subroutine **Main_Initialize** in the *Rapfish* Main module. A custom *Rapfish* toolbar is provided to speed this process. In Excel, toolbars cannot be stored with an Add-In (.xla) file, but they can be attached to a spreadsheet. To configure your Excel environment to access the *Rapfish* toolbar, open the example **RedSea.xls** or the **RapTemplate.xls** file. You should be able to see the *Rapfish* toolbar in the header and it should be listed under the Toolbars menu. Beware that a macro of exactly the same name may exist in some commercial Excel add-ins (e.g., Acrobat PDF maker) – if you experience this problem, identify what you want by using the full address. **Rap1_6.xla!Main_Initialize**

If you exit Excel and then re-open you should still see the *Rapfish* toolbar. You can make the *Rapfish* toolbar invisible in the usual way (de-select in the toolbar menu).

What to do if you do not see the Rapfish toolbar button: 

If you lose the *Rapfish* toolbar for some reason, first ensure that you can run *Rapfish* directly by invoking **Main_Initialize** using the Excel Tools → Macros → Macro menu. If this fails, please start the installation process from step 2 above, and check your Excel installation. Otherwise, continue by opening the **RedSea.xls** example file. This should re-activate your *Rapfish* custom toolbar. If this fails for some reason you can try starting over at step 2, or you can re-create a *Rapfish* toolbar as:

- Right click in the Toolbars area. A list comes up of all the active toolbars. If *Rapfish* is listed, make sure to tick on; this should re-activate your Rapfish toolbar. If *Rapfish* is missing from list, do not lose heart, keep going ...
- Under the Toolbars menu click Customize
- Go to Toolbars index card and click "New" and name your new toolbar (a blank toolbar box pops up)
- Go to Commands index card, select Macros, and drag the "Custom Menu Item" to your new toolbar
- Right click on your toolbar (or click "Modify Selection" while selected) to bring up a menu describing your toolbar. Use this menu to design your toolbar:
  - Click "Image and Text"
  - In the Name: box type *Rapfish*
  - Click "Edit button image" and select the picture you want (e.g. fish)
  - Click "Assign Macro" and type in **Main_Initialize**
- Open a *Rapfish* workbook. Go back to the Toolbars index card in the Customize menu and click "Attach…". Select your toolbar from the list and click on "Copy >>" to attach the toolbar to the current workbook. Save the workbook. This should keep you toolbar active for any Excel session.
- Test your new *Rapfish* toolbar

What to do if you see the Rapfish toolbar button but it gives the error message "macro Main_Initialize not found"

Select customize after right clicking the Rapfish menu button. In Office 2000 and XP, then select Toolbar box. Right click on Rapfish menu button again and select "assign macro"
Type in:

Rap1_6.xla!Main_Initialize

What to do if Excel wont remember the macro assignation when Excel is restarted?

In Office XP and 2000, if you get problems with the toolbar button not retaining its memory of the assigned macro on closing and re-opening Excel, check where your Excel10.xla or Excel.xla file is being stored. This file stores all your customised items including menu buttons. Check that you are using only ONE Excle10.xla file under your current identity. (NOTE there are several places where these files can live. Youll want ONE stored where your idenity lives.)

NOTE that some LAN systems (weekly at FAO for example) will set your customised Excel options to default settings during regular 'file maintenance'. If this happens, you can try pleading with the LAN administrator. Or you may have been smart enough to store a back-up copy of the .xla file somewhere that XP would never expect.

A *Rapfish* Test Run

Once you have installed the *Rapfish* software into your Excel environment you are ready to use it to analyze fisheries. Before trying your own data set, familiarize yourself with the procedure by opening the example file **RedSea.xls** and exercising each of the *Rapfish* facilities. This section gives a step-by-step procedure for testing the *Rapfish* procedure on an example test file **RedSea.xls**.

1) Click your *Rapfish* toolbar button (fish icon) to open the *Rapfish* user interface form. This has the controls and parameter settings you will need to specify location and extent of the input scores and the processing to do.

2) The user form will pop up and display the default parameter values from the **param.txt** file. The form will be filled with parameters which correspond to the 'ecological' evaluation field in the Red Sea example sheet. If it isn't populated, click the **Get** button on the form. Examine the **RapScores** worksheet data and note how the parameter values relate to the data worksheet. If you want to change parameters to look at a different evauation field set of attributes, edit the parameters.

3) To allow repeated running of *Rapfish*, without having to re-enter the parameters, use the **Save** button to save sets of parameters to a text file of your choice. The file will be in the working directory, unless you give it a full path. The next time you run *Rapfish*, enter this parameter file name in the **Get** box and the saved parameters will re-appear in the form.

4) Click the **RUN Rapfish** button to do a RAPFSIH analysis for the data set defined by the current parameters. Results should appear in **RapAnalysis** worksheet.

5) Click the **Run MONTE CARLO** button to examine the effect of scoring variability or error on the *Rapfish* ordination. View results in the **MonteCarlo** worksheet. Try changing the number of random repeats and the type and magnitude of the scoring error distribution to see the effect on the ordination scatter. You may specify either a Normal error distribution or triangular distribution based on user-entered error limits (either the same across each attribute, or for each individual data value) on the spreadsheet (see rows near bottom of the RapScores sheet).

6) Click the **Run LEVERAGING** button see the effect of removing one attribute at a time on the ordination. View results in the **Leveraging** worksheet.

Analyzing Your Own Fisheries Scoring Data

When you are ready to analyze your own set of fisheries scoring data, copy the **Redsea.xls** spreadsheet to a new name (e.g., **myworking_rapanalysis.xls)** by doing a **Save As** to a new spreadsheet *file (do not forget this step or you will lose the example file! Save your sanity…please keep backups*). Then, enter the names of your fisheries on the **Rapscores** sheet in the first column (and optionally name abbreviations in second column for use in chart labeling) and corresponding fisheries scores for each of the attributes listed. Ensure that the scores for each attribute lie between the worst case BAD attribute values and the best case GOOD attribute values (these are listed further down the spreadsheet). Add or delet fisheries rows as needed.

Once the data is entered, open the *Rapfish* user form by double-clicking on the *Rapfish* toolbar 'fish' button. The user form will pop up and display the default parameter values from the **param.txt** file. Examine your RapScores spreadsheet data and enter the new fisheries row numbers, attribute column letters, and numbers of fisheries and attributes. The parameters should be for only one set of attributes at a time (ecological, economic, technological, social, or ethical). To allow repeated running of *Rapfish*, without having to re-enter the parameters, use the Parameters **Save** to button to save your parameters to a text file of your choice. The file will be in your working directory unless you give it a full path. The next time you run *Rapfish*, enter this parameter file name and your saved parameters will re-appear by clicking the **Get** from button.

Click each of the 3 Run buttons in turn to run the analyses. Don't forget to copy your results from each evaluation fields to an appropriate results spreadsheet, or earlier work will be over written.

How to Edit the *Rapfish* Add-In for Further Development

When new features or bug fixes are needed for the *Rapfish* Add-In, open your refence copy of **Redsea.xls**, and follow these guidelines:

- In the Visual Basic for Applications editor inside Excel, find the Rap1_v.xla add-in, locate the **ThisWorkbook** item, and set the **IsAddIn** property of the **ThisWorkbook** object listed under the RapfishV.xla Add-In to **False**. The Add-In's hidden workbooks will now appear in Excel. You can use these to test your code or load your own spreadsheets. Make your modifications to the VBA code. When testing is complete, set the **IsAddIn** property of the **ThisWorkbook** object back to **True**. Then do a **Save** in the VBA environment. The modified Add-In is now saved and can be accessed by Excel (and you have overwritten the old version). Your existing *Rapfish* toolbar should still work. (NOTE: the *.xla add-in may be hidden away in an unexpected location if you are using Windows XP –see above).

- You should always be able to run *Rapfish* for newer versions of Excel than it was developed for (if we can trust Microsoft policy). You *should not* assume that you can run a newer Excel version of *Rapfish* on an older version of Excel. To save development and software maintenance time you remind to remind users and collaborators to get a version of Excel the same or more recent than the *Rapfish* software; the problem with this is that some people do not maintain Excel updates for cost reasons. To maintain multiple *Rapfish* copies for specific versions of Excel (e.g. Excel'97, Excel 2000, Excel XP, etc.. make your modifications to the most recent version of Excel and then convert these to the old Excel workbook version and test.

RAPFISH: WHAT DOES IT ALL MEAN?

Interpretation

This section provides a brief overview of how to interpret *Rapfish* ordinations. More detailed explanations of the *Rapfish* technique can be found in Pitcher and Preikshot (2001), Pitcher (1999), and other references cited therein.

The reason for developing a rapid appraisal procedure such as *Rapfish* is to provide a quantitative tool to monitor and assess the status (or 'health') of fisheries, the ecosystems in which they are embedded, and the complex relationships among the human communities and economies which rely on fisheries. Rapid appraisal relies on approximate scoring of a set of attributes on simple scale. Although many things may be measured, attributes for Rapfish are chosen that have a clear directionality form 'bad' (as conceivable) to 'good' (as conceivable). The initial choice of attributes to score for a particular status field is at first critical, but once a set has achieved a reasonable consensus, single additions and deletion make a progressively lesser effect on the overall ordination results (and this effect may be judged using the leverage analysis). Given clear metrics describing fisheries health, tracked over time and for various locations and species of fish, fisheries managers can make more informed decisions about for fisheries policy and management. The effects of various policy and management choices on fisheries health can be tracked over time, and forecasts made of the impacts of alternative policies using *Rapfish*.

The two-dimensional *Rapfish* ordination gives a graphical summary of the differences in fisheries as described by the *Rapfish* attribute scores. The position of a fishery on the "sustainability" horizontal x-axis gives a relative ranking of the fisheries between the two extreme reference fisheries with all BAD and all GOOD attribute scores. The position of the vertical y-axis indicates variation in fisheries attributes that are *not related* to sustainability. The vertical axis is arbitrary and depends on the particular selection of the UP and DOWN reference fisheries. By convention, the UP fishery has the first half of the attributes set to the worst case BAD values and the other half all GOOD; the DOWN reference is the opposite of the UP fishery.

The **Multi-Dimensional Scaling (MDS)** operation at the heart of *Rapfish* is described in detail in Schiffman *et al.* 1981, Norusis (1990) and Alder *et al.* (2000). Essentially, MDS transforms multi-dimensional statistics (in our case a group of fisheries with multiple attribute scores) into a lower dimension representation which maintains similar distance properties (between cases (fisheries). For *Rapfish*, a two-dimensional representation is selected with a *Ratio level* of measurement allowing a transformation polynomial of order 1 (default). Selecting Ratio means that a metric analysis is used. Given an input matrix S of distances (dissimilarities) between multi-attribute fisheries (squared Euclidean distances from Proximities function), the *Rapfish* ALSCAL algorithm generates a two-dimensional set of fisheries scores with squared Euclidean distance matrix D^2 approximately equal to matrix S transformed by a monotonic linear function with 0 intercept:

$$\zeta\{\mathbf{S}\} = \mathbf{D}^2 + \mathbf{E}$$

where symbol ζ means "a monotonic linear transformation with 0 intercept" and **E** is an error (residual) matrix equal to the difference between the $\zeta\{\mathbf{S}\}$ and $\mathbf{D}^2$ matrices. The ALSCAL iterative algorithm strives to minimize the error **E** in a least squares sense. Iterations stop when the s-stress metric is *below* a specified minimum (0.005 by default).

$$\text{s-stress} = (\text{stress})^{1/2}$$
$$\text{stress} = ||\mathbf{E}|| \,/\, ||\zeta\{\mathbf{S}\}||$$

where ||**X**|| means "sum of squares of all the elements of matrix X". A low value of s-stress (or stress) is desired to give us confidence that the solution is a good representation of the input data. Iterations may also stop if a maximum number of iterations has occurred or if the solutions stop reducing s-stress. The solution is not generally unique and/or an exact solution may not exist. The error return code from the *Rapfish* g77ASLCAL function indicates the reason for the iterations stopping.

The rotating, flipping, and scaling operations are intended to allow successive *Rapfish* ordinations to be compared on the same axes. The design and use of the anchor and reference fisheries also provides repeatability in the *Rapfish* 2-dimenional ordinations. The numbers 0 to 100 and –50 to 50 are *entirely arbitrary* and have not been 'calibrated' to any other biological or fisheries management value metric. However the BAD and GOOD reference fisheries indicate what we believe to be, respectively, fisheries with all the worst and all the best features for the status being measured (fishery health in the basic *Rapfish*). The position of a fishery on the BAD to GOOD 'sustainability' axis and its position relative to other fisheries helps us get a sense of the relative sustainability of the set of fisheries, for sustainability defined for each of the *Rapfish* attribute sets. We do not (yet) know how a particular *Rapfish* sustainability measure might relate to the actual sustainability of the fisheries ecosystem and its involved human economy, communities, and social organization.

It is *important* that the same sets of attributes, order of attributes, scoring definitions, anchor fisheries, and reference fisheries be used for all *Rapfish* analyses, in order that results can be compared. For instance, if one wanted to compare the Red Sea fisheries with the North Atlantic, without running all together in the same *Rapfish* spreadsheet, one must ensure that all the above conditions are the same in both analysis runs, otherwise comparisons would not be consistent.

Rapfish Error and Variability

Some sources of error in the *Rapfish* technique are:

- attribute scoring error due to imperfect knowledge of the fisheries, misunderstandings of the *Rapfish* attributes and scoring guidelines, differences in opinion or judgement by several people, and data entry errors.
- incomplete convergence of the MDS solution (higher stress values)
- convergence on an ambiguous solution (for instance, the mirror image of "correct" solution)
- attributes not applicable for some fisheries

The Monte Carlo simulation allows the user to test the effect of various amounts of scoring error on the *Rapfish* ordination. The anchor and reference fisheries should, ideally, not move for Monte Carlo runs.

Sustainability 'Calibration'

MDS is useful for illustrating differences among multi-attribute-scored fisheries. The coordinate along the 'bad' to 'good' fishery axis gives a relative measure of 'sustainability', as defined by the *Rapfish* multiple attribute scores. However, we have not defined sustainability for fisheries as a whole, but rather only specific attributes of sustainability. How a particular combination of attribute scores relates to concepts of 'sustainability' is yet to be defined. Further study is required to define and 'calibrate' the sustainability axis, probably through some kind of scaling before or after MDS, so that for each combination of attribute scores the sustainability measure makes sense and tends to agree with other methods of fisheries health assessment.

Another sort of calibration desired is that for each of the attribute sets (ecological, technological, etc.) the sustainability scales are not biased relative to each other. At present, *Rapfish* does not consider this possibility. *Rapfish* weights all attributes evenly on entry to the algorithm, although the final

results produce implicit weightings. A technique for determining relative weightings of attributes in relation to the perception of status (as opposed to actual status) has been explored by Power (2003; Power and Chuenpagdee, 2004) .

Dominant Attributes

The 'leveraging' calculation allows the user to check for changes in ordination when successive attributes are left out. If the status of attributes truly reflects the status being assessed, one expects them to contribute more or less equally to the final result. In many Rapfish ordinations performed to date, leverage values range between 2% and 6%. No one attribute should dominate if a truely multivariate situation is in existance, and a rough rule of thumb is that the ordination score should not be influenced by more than 8% in any one or two attributes. Where this is the case, status should probably determined by just taking the attributes that have the high leverage values.

FUTURE *RAPFISH* DEVELOPMENT

This section assumes familiarity with the recent literature on Rapfish (e.g., Pitcher and Preikshot 2001, Pitcher 2004), and suggests areas for further development to the algorithms.

- The option to allow selection of the amount of error expected for each attribute and fishery score allows Monte Carlo scoring error tests to see the effect of imperfect or incomplete scoring. This new feature has not yet been fully been tested by users.

- The method of standardization should be examined to see how it affects the Monte Carlo and Leveraging scatter. At present, Monte Carlo and Leveraging assume a fixed standardization based on the raw input scores before any noise is added (or attributes removed). Another way to do 'fixed' standardization would be to use only the anchor and reference fisheries (and not the real fisheries) in calculating the standardization vectors mu and sigma. The analyses could be modified to see whether statistical standardization *for each iteration* changes the amount of ordination scatter.

- The 'anchor' fisheries, used to stabilize the ordination for repeated analyses, could be generated algorithmically rather than depending on the anchors manually entered in the RapScores spreadsheet. Once this is done, an analysis of the Monte Carlo scatter versus number and arrangement of anchors should be done. It would also be interesting to see whether the status measures vary with the *ordering* of the attributes and the particular selections of UP and DOWN reference fisheries. Also of interest are the convergence properties of the MDS solution; for example, do the anchor definitions affect the quality of the MDS fit (low stress) and what conditions cause ambiguous solutions or failure to converge to a low stress value? A new graph could be added to the Monte Carlo run plotting the stress value (a measure of fit) for each iteration to help assess how much of the ordination scatter is caused by imperfect numerical match of the distances between the input multiple attribute scored fisheries versus the 2-dimensional MDS output.

- Although the table of attributes and notes for the 'sustainability' Rapfish analysis has undergone several rounds of revisions, some difficulties still occur in obtaining consistent *Rapfish* fisheries scores, perhaps due to imperfect attribute definitions and scoring guidelines. The various underlying causes of confusion should be identified so that improved attribute definitions, scoring rules, and examples be produced to streamline the scoring process. *Rapfish* training workshops would also help.

- An interesting study would be to measure the 'bias' of several stakeholder groups (e.g. fishers, government officials, fish processing plant owners, fisheries scientists, etc.) in judging the same sets of fisheries. This analysis would also tell us the amount of variability to be expected in the scores due to differences in opinion and point of view, inaccurate or incomplete knowledge of the state of each fishery, and politics. Power (2003, Power and Chuenagdee 2004) has attempted some analyses on this point.

- It has been argued that the *Rapfish* sustainability scale is deceptive in how it relates (or implies a relation) to the real world. *Rapfish* tells us if one fishery is more healthy than another; but the question of 'how much better' is not well quantified. The scale is meaningful since the attribute definitions and scoring were developed by fisheries experts who have a detailed understanding of ecosystems, sustainability, and fisheries operations that promote or endanger an ecosystem's health. One suggestion from C. Walters is to try combining attribute scores multiplicatively rather than linearly – see below. Further study of the *Rapfish* algorithm in combination with decision theory and other techniques such as fuzzy logic might help in adding quantitative 'calibration' to

the *Rapfish* sustainability ordination and confidence in the measures for decision-making.

- To benefit fisheries management, *Rapfish* needs to be tested by fisheries scientists and managers and applied to fisheries throughout the world. To encourage use, this *Rapfish* software is provided for free download from the Fisheries Centre Web site, along with papers showing example analyses and fisheries management recommendations. With a wider audience using the method, *Rapfish* procedures can be improved and eventually applied with confidence as a policy and planning tool.

- *Rapfish* analyses need to be completed for many fisheries and compared to other methods of fisheries analysis, for the same sets of fisheries. These analyses need to be compared to other methods of fisheries health analysis to verify whether the *Rapfish* ordering of fisheries in the sustainability dimension agrees with other measures of fisheries health obtained by other means. Power (2003) has completed some work in this respect. *Rapfish* analysis should help to more fully understand the state of fisheries in this world of declining stocks and damaged ecosystems.

- Some decision-making guidelines could be developed; for example, for given *Rapfish* scores, what are the suggested fisheries management changes which should be made to correct a failing fishery. *Rapfish* scores should be tracked over several years to see the effects of management or policy changes on the health of fisheries. A trade-off analysis of the 'sustainability' suggested by each of the evaluation fields (ecological, ethical, social, technological, economics) might also be examined.

MULTIPLICATIVE UTILITY: *RAPFISH-M*

It has been suggested (Keeney 1977, C. Walters, *pers,comm.)* that human perception of utility is multiplicative, rather than additive as in the basic Rapfish analysis. This technique has the advantage that a zero score for any attribute will reduce the overall score in an evaluation field to zero without operating 'killer' attributes (see guide to Rapfish sustainability attributes and Pitcher 2004). It has the disadvantage that there is no inherently obvious way of deconvoluting the individual attribute scores so that they can be used in an ordination. The following outlines one approach, termed *Rapfish-m* using a 2/3 power factor that has given reasonably satisfactory results.

Scoring Normalization for *Rapfish-m*

Let S(i, j) be the score for attribute j for fishery i. Each attribute is scored with a value between a specified BAD(j) and GOOD(j) value for that attribute. Different attributes typically have different scales. To ensure equal handling of attributes, before calculating the overall utility, all the scores are *normalized* so that BAD(j) is at 0 and GOOD(j) is at 1 for all attributes j. Here's how:

$$Snorm(i, j) = (S(i, j) - BAD(j)) / (GOOD(j) - BAD(j))$$

Overall Multiplicative Utility Function

Define N_a = # attributes (varies depending on whether ecological, social, economic, etc. group). Define an arbitrary power transform:

$$P = c * N_a \qquad \text{(by default } c = 2/3\text{)}$$

We define the overall fisheries multiplicative utility score for fishery i to be the product of the normalized attribute scores over all attributes, raised to the power 1/P and expressed as a percentage:

$$U(i) = 100\% * \{ \prod_j Snorm(i, j) \}^{(1/P)}$$

Rationale

The score U can be looked at as a measure of expected fishery sustainability, based on the defined attribute scores and the judgment of the persons doing the scoring. This value is plotted on the x-axis.

This transform uses multiplicative utility, but scales the individual scores with a power function so that they can be input as individual scores to the MDS ordination. The 2/3 scaling is used rather than a fixed number so that the algorithm works with all numbers of attributes from 3 upwards. A range of values have been tried here. When $P = N_a$, the algorithm reduces to the additive solution.

Attribute Mix

A measure of attribute mix is shown on the y-axis on the Rapfish plots. A mix value of 0 is assigned if all attributes are the same. Otherwise, the value assigned is the sum of the differences between each attribute score and the equivalent score for a fishery with the same utility U but with all attributes the same.

Discussion

Note that in this scheme, any score of zero reduces multiplicative overall utility to zero, so the killer factors used in Rapfish 1 are not necessary. High status ratings are hard to achieve under this method, which may or may not be thought helpful. Note also that all attributes have to have the same scale (e.g. zero to 1), and all have to increase in the same direction (i.e. zero = bad; 1 = good). At present, many of the anchor points developed for the standard Rapfish do not behave properly in Rapfish-m.

REFERENCES

Alder, J. Pitcher, T.J., Preikshot, D., Kaschner, K. and Ferriss, B. (2000) Rapfish estimates - how good is good? Pages 136-182 in Pauly, D. and Pitcher T.J. (eds) Methods for assessing the impact of fisheries on marine ecosystems of the North Atlantic. Fisheries Centre Research Reports 8(2): 195pp.

Kavanagh, P. (1999) *Rapfish* SPSS Automation and Analysis of Technique. UBC Fisheries Centre, *unpublished report*.

Keeney, R. (1977) The art of assessing multi-attribute utility functions. Organizational Behavior and Human Performance 19: 267-310.

Norusis, M.J. (1990) Chapter 24, ALSCAL. Chapter 25, SPSS PROXIMITIES routine. SPSS Base System User's Guide. SPSS Inc., USA.

Papoulis, A. (1965) Probability, Random Variables, and Stochastic Processes. McGraw Hill.

Pitcher, T.J. (1999) Rapfish, A Rapid Appraisal Technique For Fisheries, And Its Application To The Code Of Conduct For Responsible Fisheries. FAO Fisheries Circular No. FIRM/C: No. 947: 47pp.

Pitcher, T.J. (ed) (2004) Rapfish, A Rapid Appraisal Technique For Tracking Fisheries Status. FAO Fisheries Technical Report *(in prep*).

Pitcher, T.J. and Preikshot, D.B. (2001) Rapfish: A Rapid Appraisal Technique to Evaluate the Sustainability Status of Fisheries. Fisheries Research 49(3): 255-270.

Pitcher, T.J., Bundy, A., Preikshot, D., Hutton, T. and Pauly, D. (1998) Measuring the unmeasurable: a multivariate interdisciplinary method for rapid appraisal of health of fisheries. Pages 31-54 in Pitcher, T.J., Hart, P.J.B. and Pauly, D. (eds) Reinventing Fisheries Management Chapman and Hall, London. 435pp.

Power, M.P. (2003) Fishing for Justice: A Groundwork for Ethical Fisheries Policies in Canada. PhD. Thesis, Resource Management, University of British Columbia, Vancouver, Canada. 195 pp.

Power, M.P. and Chuenpagdee, R. (2004) A Paired-Choice Method for Cross-Validating Rapfish Results. FAO Fisheries Technical Report *(in prep*.)

Schiffman, S.S., Reynolds, M.L. and Young, F.W. (1981) Introduction to Multidimensional Scaling: Theory, Methods, and Applications. Academic Press.

Tesfamichael, D. (2001) *Rapfish* Red Sea Analysis. UBC Fisheries centre, *unpublished report.*

Tesfamichael, D. (2004) *Rapfish* Red Sea Analysis. FAO Fisheries Technical Report (*in prep*.)

Young, Forrest W. (2000) web site http://forrest.psych.unc.edu

ANNEXES

ANNEX 1: PUBLISHED PAPERS USING RAPFISH

Anonymous (2002) Laporan Teknis Riset Indikator Kelautan dan Perikanan Tahun 2002 [Technical Report of the Research on Indicators of Marine Affairs and Fisheries in 2002]. Research Centre for Product Processing and Socio-Economics of Marine Affairs and Fisheries. Ministry of Marine Affairs and Fisheries. Jakarta. 74 p. [Rapfish guide in Bahasa, Indonesia].

Alder, J., Pitcher, T.J., Preikshot, D., Kaschner, K. and Ferriss, B. (2000) Rapfish estimates - how good is good? Pages 136 - 182 in Pauly, D. and Pitcher T.J. (eds) Methods for assessing the impact of fisheries on marine ecosystems of the North Atlantic. Fisheries Centre Research Reports 8(2): 195pp.

Alder, J., Zeller, D., Pitcher, T.J. and Sumaila, U.R. (2002) A Method for Evaluating Marine Protected Area Management. Coastal Management 30: 121-131.

Pitcher, T.J. (2003) The compleat angler and the management of aquatic ecosystems. Pages 3-7 in Coleman, A.P.M. (ed) Regional Experiences for Global Solutions. Proceedings of the 3rd World Recreational Fisheries Conference, Darwin, Australia May 2002. Northern Territories Fisheries Report 67: 269 pp.

Pitcher, T.J. (1999) Rapfish, A Rapid Appraisal Technique For Fisheries, And Its Application To The Code Of Conduct For Responsible Fisheries. FAO Fisheries Circular No. FIRM/C: No. 947: 47pp.

Pitcher, T.J. and Preikshot, D. (1998) Rapid Appraisal of the Status of West African Distant Water Fleet Fisheries Relative to Home Fleets using the RAPFISH Technique. Pages 90 - 93 in Bonfil, R., Munro, G., Sumaila, U.R., Valtysson, H., Wright, M., Pitcher, T., Preikshot, D., Haggan, N. and Pauly, D. (eds) Distant water fleets: an ecological, economic and social assessment. Fisheries Centre Research Reports 6 (6): 111pp.

Pitcher, T.J. and Preikshot, D.B. (2001) Rapfish: A Rapid Appraisal Technique to Evaluate the Sustainability Status of Fisheries. Fisheries Research 49(3): 255-270.

Pitcher, T.J. and Preikshot, D.B. (1999) Rapfish: A Rapid Appraisal Technique to Evaluate the Sustainability Status of Fisheries. In Craig, J. (ed) ICLARM Workshop On Lake Nasser's Fisheries.

Pitcher, T.J. and Power, M.P. (2000) Fish Figures: Quantifying the Ethical Status of Canadian Fisheries, East and West. Pages 225-253 in Coward, H., Ommer, R. and Pitcher, T.J. (eds), Just Fish: the ethics of Canadian fisheries. Institute of Social and Economic Research Press, St John's, Newfoundland, 304 pp.

Pitcher, T.J., Bundy, A., Preikshot, D., Hutton, T. and Pauly, D (1998) Measuring the unmeasurable: a multivariate interdisciplinary method for rapid appraisal of health of fisheries. Pages 31-54 in Pitcher, T.J. Hart, P.J.B. and Pauly, D. (eds), Reinventing Fisheries Management Chapman and Hall, London. 435pp.

Pitcher, T.J., S. Mackinson, M. Vasconcellos, L. Nøttestad and D. Preikshot (1999) Rapid appraisal of the status of fisheries for small pelagics using multivariate, multidisciplinary ordination. Pages 759-782 in T.J. Quinn II, F. Funk, J. Heifetz, J.N. Ianelli, J.E. Powers, J.F. Schweigert, P.J. Sullivan, C.-I. Zhang (eds), Fishery Stock Assessment Models. Alaska Sea Grant, Fairbanks.

Preikshot, D.B. and D. Pauly (1999) A multivariate interdisciplinary assessment of small-scale tropical fisheries. Pages 803-814 in T.J. Quinn II, F. Funk, J. Heifetz, J.N. Ianelli, J.E. Powers, J.F. Schweigert, P.J. Sullivan, C.-I. Zhang (eds), Fishery Stock Assessment Models. Alaska Sea Grant, Fairbanks, USA.

Preikshot, D.B., Nsiku. E., Pitcher, T.J. and Pauly, D. (1998) An interdisciplinary evaluation of the status and health of African lake fisheries using a rapid appraisal technique. J. Fish Biol. 53 (Suppl. A): 382-393.

ANNEX 2: THE ALSCAL FORTRAN DLL

This annex provides detailed description of the ALSCAL FORTRAN DLL:

Annex-2a lists the FORTRAN source files and explains the procedure for building the DLL
Annex-2b gives the ALSCAL FORTRAN DLL function definition
Annex-2c shows how to declare and call the ALSCAL DLL from Visual Basic
Annex-2d lists the parameter settings hard-coded inside the ALSCAL DLL

For further details, please see the ALSCAL and *Rapfish* VBA source code provided.

Annex-2a ALSCAL DLL Build Procedure

The source file for the new ALSCAL DLL is:

g77ALSCAL.f

To build the DLL you also need the the following makefile and definitions file:

Makefile.g77ALSCAL
g77ALSCAL.def

Before building, first install the Generalized Compiler Collection (GCC) GNU which contains the FORTRAN77 compiler G77. Go to the Internet at **http://cygwin.com** where you will find instructions on how to download and install the popular GNU development tools and utilities for Windows. Click on the "Install Cywin now" icon to download a **setup.exe** file. Execute this setup file to download the GCC tools from one of several "mirror" sites (for example http://planetmirror.com). Select "Download from Internet" to download to your chosen directory. Select **gcc** and **bash** from the CygwinSetup window list of packages for download (versions gcc 2.95.3-5 and bash 2.05-6 as of Aug 2001). After the download is complete (this can take awhile) the compiler can be installed by running setup.exe again, but this time selecting "Install from local directory" and point to the directory that the download was done; select gcc and bash for install. More detailed instructions are contained at the cygwin.com site.

To build the ALSCAL DLL, open the Cygwin GCC Bash Shell (created when you install GCC). Then type the following commands, with *sourcedir* replaced with the directory path of your ALSCAL source code:

```
>cd sourcedir
>make –f Makefile.g77ALSCAL  >&g77ALSCAL.log
```

Change the directory to the one containing your ALSCAL source files (for

The files generated from the build procedure are:

g77ALSCAL.o – object file (before linking)
g77ALSCAL.log – build log file
g77ALSCAL.dll – the ALSCAL Dynamic Link Library (DLL) routine

Follow the *Rapfish* Excel installation instructions in section 4 to use g77ALSCAL.dll for *Rapfish*.

Annex 2b: ALSCAL FORTRAN DLL Function Interface Syntax

```
C     VERSION 01.1/DLL notes:
C     Author: Pat Kavanagh, University of British Columbia Fisheries Centre
C
C     Re-wrote ALSCAL 84.3F/PC as a DLL (dynamic linked library) routine
C     that can be invoked from Microsoft Windows environments such as Excel.
C     All of the file, terminal, and printer I/O has been removed and
C     replaced with variable transfer in the DLL call. An Excel Visual Basic
C     user interface has been added to allow users to access ALSCAL
C     directly from Excel, rather than having to use SPSS. All results
C     printing and plotting is now to be carried out by a calling routine
C     for example in Excel/ Visual Basic) which looks after data I/O
C     and graphics.
C
C     DLL was built and tested using GNU Compiler Colection (GCC) which
C     includes a FORTRAN compiler g77 which roughly follows FORTRAN 77.
C
C     Any line of code with C*PK* at start was commented out by Pat K as
C     no longer relevant for the DLL version.
C
C     Input arguments:  NROWS - # rows in Xinput
C                       NCOLS - # columns in Xinput
C                       Xinput - proximity matrix input
C
C     Output arguments: g77ALSCAL - error or warning code (see definitions below)
C                       MDSout - NROWSx2 matrix containing MDS ordination
C                       Metric – array containing stress values and change in
C                                stress C for each iteration 1 to Niter
C                       Sout – array with s-stress value and squared
C                              correlation (RSQ)
C                       Niter – number of iterations needed for the solution
C                       Xdebug, Idebug - debug matrices
C                       Xdebug(9) has the # words memory required)
C
C     Function return codes (in g77ALSCAL):
C     0    no errors
C
C     Main routine errors/warnings:
C     902  FATAL ERROR: #rows must equal #columns for non-rectangular data
C     903  FATAL ERROR: #stimuli less than 3
C     904  FATAL ERROR: total #matrices less than 1
C     999  FATAL ERROR: not enough memory (MAXSIZ exceeded)
C     9910 WARNING:     inconsistent control parameters
C     9920 FATAL ERROR: computations terminated (# parameters > #observations)
C
C     STEP1 errors:
C     901  FATAL ERROR: all data missing for matrix
C
C     STEP2 errors/warnings:
C     200  FATAL ERROR: Max # tie-blocks exceeds NDMX (subroutine BLOC2)
C
C     STEP3 errors/warnings:
C     299  WARNING: iterations stopped since #iterations has reached MAXIT
C     141  WARNING: iterations stopped since achieved S-STRESS less than STMIN
C     143  WARNING: iterations stopped since S-STRESS improvement less than EPSI
C     145  WARNING: iterations terminated due to negative S-STRESS improvement
C                   (data may be ill-conditioned or could be programm error?)
```

```
C    1 FATAL ERROR: computations interrupted since a dimension has only 0 weights
C
C     MINV errors/warnings (called by POLYF
C                                     via DISTP, via STEP3A, STEP2, STEP3),
C                            INIT (via STEP2), INSWM (via STEP2), or STEP3):
C     555  FATAL ERROR: Attempt to take inverse of singular matrix
C
C     LINT warnings (called by POLYF (via DISTP, via STEP3A, STEP2, STEP3)
C     9998 WARNING: a linear transformation of your data has negative slope
C-----------------------------------------------------------------------

INTEGER function g77ALSCAL(NROWS, NCOLS, Xinput, MDSout, Metric,
+                           Sout, Niter, Xdebug, Idebug)
```

Annex 2c: Visual Basic Calling Syntax for ALSCAL DLL

Below is a section of Visual Basic for Applications (VBA) code showing the calling syntax for the ALSCAL multi-dimensional scaling Dynamic Link Library (DLL) function

```
'FORTRAN DLL for multi-dimensional scaling
'(derived from Forrest Young code)

Declare Function g77alscal Lib "g77alscal.dll" _
Alias "g77alscal_" _
(ByRef Nrows As Long, _
ByRef Ncols As Long, _
ByRef Xinput As Single, _
ByRef MDSout As Single, _
ByRef Metric As Single, _
ByRef Sout As Single, _
ByRef Niter As Long, _
ByRef Xdebug As Single, _
ByRef Idebug As Long) As Long
.
.
‘Definitions of parameters
Dim Distance() As Single       ‘proximity matrix Ncases x Ncases
Dim MDSout() As Single         ‘MDS output matrix Ncases x 2
Dim Sout(2) As Single          ‘s-stress and squared correlation (RSQ)
Dim Niter As Long              ‘number of iterations in solution
Dim Metric(60) As Single       ‘s-stress and delta for each iteration 1 to Niter
Dim Xdebug(5000) As Single     ‘debug array
Dim Idebug(30) As Long         ‘debug array
                               ‘Idebug(9) contains # words memory
                               ‘ required by ALSCAL DLL)
Dim ierr As Long
Dim N As Long
..
‘after Ncases is set equal to the number of fisheries, can redimension the array
ReDim Distance(Ncases, Ncases)
ReDim MDSout(Ncases, 2)
.
.
‘Call ALSCAL with by reference pointers to first element of each
‘parameter, matrix, array
‘Distance is the input proximity matrix (N x N)
‘MDSout is the output MDS ordination (N x 2)

N = CLng(Ncases)
ierr = g77alscal(N, N, Distance(1, 1), MDSout(1, 1), Metric(1), _
                 Sout(1), Niter, Xdebug(1), Idebug(1))
```

Annex 2d: Parameters for ALSCAL DLL Multi-Dimensional Scaling

Below is a section of FORTRAN code from the *Rapfish* ALSCAL DLL showing the parameter definitions and values hard-coded for *Rapfish*.

```
C-----------------------------------------------------------------------
C     INPUT DATA SPECIFICATIONS
C     -----------------------------------------
C     NROW     NUMBER OF ROW STIMULI (NO MAXIMUM)
C     NCOL     NUMBER OF COLUMN STIMULI (NO MAXIMUM)
C     NS       NUMBER OF MATRICES (NO MAXIMUM)
C     NDTYP    MEASUREMENT LEVEL
C                   =1   RATIO (POLYNOMIAL W/O INTERCEPT)
C                   =2   INTERVAL (POLYNOMIAL WITH INTERCEPT)
C                   =3   ORDINAL(DEFAULT)
C                   =4   NOMINAL
C     NSIM     DATA FORM
C                   =0   SYMMETRIC-DISSIMILARITY(DEFAULT)
C                   =1   SYMMETRIC-SIMILARITY
C                   =2   ASYMMETRIC-DISSIMILARTY
C                   =3   ASYMMETRIC-SIMILARITY
C                   =4   RECTANGULAR-DISSIMILARITY
C                   =5   RECTANGULAR-SIMILARITY
C     NPS      MEASUREMENT PROCESS (ONLY WHEN NDTYP=3)
C                   =1   DISCRETE(DEFAULT)
C                   =2   CONTINUOUS
C     NWC      MEASUREMENT CONDITIONALITY
C                   =1   MATRIX CONDITIONAL(DEFAULT)
C                   =2   ROW CONDITIONAL
C                   =3   UNCONDITIONAL
C     NDEG     DEGREES IN POLYNOMIAL (WHEN NDTYP=1 OR 2)
C     NDMX     MAXIMUM NUMBER OF ORDINAL OBSERVATION CATEGORIES.
C                   (DEFAULT= TOTAL NUMBER OF CELLS, OR 1000,
C                             WHICHEVER IS SMALLER)
C     CUT      CUTOFF FOR MISSING DATA (DEFAULT 0.0)
C
C     ANALYSIS SPECIFICATIONS
C     ---------------------------------------
C     NWE      MODEL TYPE
C                   =0   SIMPLE EUCLIDIAN MODEL (DEFAULT)
C                   =1   INDIVIDUAL DIFFERENCES MODEL
C                   =2   MULTIPROCESS ASYMMETRIC MODEL
C                   =3   MULTIPROCESS ASYMMETRIC INDIVIDUAL DIFFERENCES
C                        MODEL
C                   =4   GENERALIZED EUCLIDEAN MODEL (GEMSCAL)
C     NDIM     NUMBER OF DIMENSIONS IN THE SOLUTION  (MAXIMUM)
C     NDMN     NUMBER OF DIMENSIONS IN THE SOLUTION  (MINIMUM)
C     NNC=1    IF WEIGHTS NOT CONSTRAINED TO BE POSITIVE
C              (DEFAULT IS NONNEGATIVITY RESTRICTIONS)
C
C     MAXIT    MAXIMUM NUMBER OF ITERATIONS (DEFAULT IS 30)
C     EPSI     CONVERGENCE CRITERION (DEFAULT IS 0.001)
C     STMIN    MINIMUM STRESS (DEFAULT IS .005)
C     NDIR     NUMBER OF GEMSCAL DIRECTIONS
C
C     I/O OPTIONS   (not applicable for DLL version)
C     -------------------------------------------------------------
C     NDT      PRINT INPUT DATA
```

```
C                  =0 NO
C                  =1 YES
C     NPT      PLOT RESULTS
C                  =0 NO
C                  =1 PLOT SPACES AND OVERALL FIT
C                  =2 ALSO PLOT TRANSFORMATION AND FIT FOR EVERY
C                     PARTITION (CAN BE VERY MANY PAGES OF OUTPUT)
C     NPH      PUNCH RESULTS
C                  =0 DO NOT PUNCH
C                  =1 PUNCH DERIVED CONFIGURATION
C                  =2 PUNCH INITIAL AND DERIVED CONFIGURATION
C     INDATA   DATA INPUT UNIT
C                  =0   DATA READ FROM CARDS
C                  =N   DATA READ FROM UNIT N
C     INITX    INITIAL STIMULUS COORDINATES
C                  =0   COMPUTE
C                  =1   COMPUTE AND PRINT
C                  =2   READ AND PRINT
C                  =3   READ, PRINT AND FIX
C     INITXC   INITIAL COLUMN STIMULUS COORDINATES
C                  =0   COMPUTE
C                  =1   COMPUTE AND PRINT
C                  =2   READ AND PRINT
C                  =3   READ, PRINT AND FIX
C     INITW    INITIAL SUBJECT WEIGHTS
C                  =0   COMPUTE
C                  =1   COMPUTE AND PRINT
C                  =2   READ AND PRINT
C                  =3   READ, PRINT AND FIX
C     INITWS   INITIAL STIMULUS WEIGHTS
C                  =0   COMPUTE
C                  =1   COMPUTE AND PRINT
C                  =2   READ AND PRINT
C                  =3   READ, PRINT AND FIX
C
C     OTHER PARAMETERS
C     -------------------------------
C     NOULB    UPPER AND LOWER BOUNDS TO ESTIMATE MISSING DATA
C                  =0   YES (DEFAULT)
C                  =1   NO  (ALSCAL 4 METHOD)
C     ICNSTR   CONSTRAIN MISSING DATA (NOT FULLY IMPLEMENTED)
C     DEBUG    FULL DEBUGGING OUTPUT (1 = DEBUG ON)
C
C-----------------------------------------------------------------------

C NROWS and NCOLS are DLL function Input arguments

      BIG=9.0E20
      NROW = NROWS
      NCOL = NCOLS
      NS = 1
      NDTYP = 1    → Ratio Level of measurement (metric analysis) with polynomial
                   → transformation of order 1
      NSIM = 0     → Symmetric Dissimilarity (distance) input matrix
                   → (see Proximities subroutine)
      NPS = 1
      NWC = 1
      CUT = 0.0
```

```
NWE = 0
NDIM = 2
NDMN = 2
NNC = 0

MAXIT = 30          → iteration parameters
EPSI = 0.001
STMIN = 0.005
NDIR = 0

NDT = 1       NPT = 0       NPH = 0
INDATA = 0
INITX = 1
INITXC = 0
INITW = 0
INITWS = 0

ICNSTR = 0
NOULB = 0
DEBUG = 0
```

ANNEX 3: VBA CODE FOR *RAPFISH* EXCEL ADD-IN

```
MODULE MAIN
'-------------------------------------------------------------
'This module contains the main Rapfish routine.
'Subroutines used are in RapSub.
'The Fortran DLL g77alscal.dll must be resident in the
'Excel default directory for the code to work.
'
'     Function ierr return values (from g77ALSCAL):
'     0    no errors
'
'     Main errors:
'     902  FATAL ERROR: #rows must equal #columns for non-rectangular data
'     903  FATAL ERROR: #stimuli less than 3
'     904  FATAL ERROR: total #matrices less than 1
'     999  FATAL ERROR: not enough memory (MAXSIZ exceeded)
'
'     STEP1 errors:
'     901  FATAL ERROR: all data missing for matrix (STEP1)
'
'     STEP2 errors:
'     200  FATAL ERROR: Max # tie-blocks exceeds NDMX (subroutine BLOC2)
'
'     STEP3 returns:
'     299  WARNING: iterations stopped since #iterations has reached MAXIT
'     141  WARNING: iterations stopped since achieved S-STRESS less than STMIN
'     143  WARNING: iterations stopped since S-STRESS improvement less than EPSI
'     145  WARNING: iterations terminated due to negative S-STRESS improvement
'          (data may be ill-conditioned or could be programm error?)
'     1    FATAL ERROR - computations interrupted since a
'                        dimension has only 0 weights
'
'     MINV errors/warnings (called by POLYF (via DISTP, via STEP3A, STEP2, STEP3),
'                           INIT (via STEP2), INSWM (via STEP2), or STEP3):
'     555  FATAL ERROR: Attempt to take inverse of singular matrix
'
'     LINT warnings (called by POLYF (via DISTP, via STEP3A, STEP2, STEP3)
'     9998 WARNING: a linear transformation of your data has negative slope
'
'History:
'  P.Kavanagh   March  2001    new code
'  T. Pitcher   Feb    2004    updates & graph codes adjusted
'-------------------------------------------------------------

Option Explicit
Option Base 1     'indices for arrays/matrices start at 1 (not 0)

Dim msg As String

'Fortran DLL for multi-dimensional scaling
'(derived from Forrest Young code)
Declare Function g77alscal Lib "g77alscal.dll" _
Alias "g77alscal_" _
(ByRef Nrows As Long, _
ByRef Ncols As Long, _
ByRef Xinput As Single, _
ByRef MDSout As Single, _
ByRef Metric As Single, _
ByRef Sout As Single, _
ByRef Niter As Long, _
ByRef Xdebug As Single, _
ByRef Idebug As Long) As Long

Sub Main_Initialize()
'Open Rapfish parameter entry and run control form
    Load RapfishForm
    RapfishForm.Top = 80
    RapfishForm.Left = 385
    Call RapfishForm.Show
    RapfishForm.GetParam_Click
End Sub
```

```
Sub RunRap()
'This is baseline Rapfish, including standardization,
'proximities, ALSCAL, rotation for bad to good horizontal,
'flipping vertically so Up is Up, and arbitrary scaling.

'Number of fisheries
Dim Nfish As Integer      'real fisheries
Dim Nref As Integer       'good, bad, up, down
Dim Nanchor As Integer    'fake fisheries to assist anchoring
Dim Ncases As Integer

'Fisheries attributes and categories
Dim Ncategories As Integer  'eg. technological, etc.
Dim Nattrib As Integer  '# attributes per category

'Excel spreadsheet location for fishery 1, attribute 1
'for each of the Ncategories of attributes
Dim InitRow As Integer
Dim InitCol As Integer

'Row indices to fisheries data
Dim Ifish As Integer
Dim Igood As Integer
Dim Ibad As Integer
Dim Iup As Integer
Dim Idown As Integer
Dim Ianchor As Integer

'Column index to attributes
Dim Col1stAttrib As Integer

'Data
Dim SheetIn As Variant
Dim SheetTemp As Variant
Dim SheetOut As Variant

Dim RapScore() As Single
Dim Rap01() As Single     'standardized RapScore

Dim Distance() As Single
Dim MDSout() As Single
Dim MDSrotate() As Single
Dim MDSfns() As Single
Dim Sout(2) As Single
Dim Niter As Long
Dim Metric(60) As Single
Dim Xdebug(5000) As Single
Dim Idebug(30) As Long
Dim ierr As Long
Dim N As Long
Dim ColFishNames As Integer
Dim fishnames(200) As String

'Other
Dim i As Integer, j As Integer, k As Integer
Dim mu() As Single
Dim sigma() As Single
Dim errstring As String
Dim range_str As String
Dim ch As Variant   'chart object ID

'angle of the BAD to GOOD vector in degrees before rotation
'rotate each point in 2D MDS output by -theta
'to make the BAD to GOOD line horizontal
Dim theta As Single

'Error handling
'On Error GoTo 0    ' Turn off error trapping.
On Error GoTo RapErrorHandler

'Default values of parameters
Ifish = 1
Igood = 27
Ibad = 28
```

```
Iup = 29
Idown = 30
Ianchor = 31

Nfish = 26
Nref = 4
Nanchor = 16
Nattrib = 10

'Load current parameters from RapfishForm into local variables.
Nfish = RapfishForm.Numfish.Value
Nref = RapfishForm.Numref.Value
Nanchor = RapfishForm.Numanchor.Value
Nattrib = RapfishForm.Numattrib.Value

Ifish = RapfishForm.Row1stFISH.Value
Igood = RapfishForm.RowGOOD.Value
Ibad = RapfishForm.RowBAD.Value
Iup = RapfishForm.RowUP.Value
Idown = RapfishForm.RowDOWN.Value
Ianchor = RapfishForm.Row1stAnchor.Value

'Convert Excel column letters to column numbers
Col1stAttrib = RapSub.ExcelColNum(RapfishForm.Col1stATTRIBUTE.Value)
ColFishNames = RapSub.ExcelColNum(RapfishForm.ColFishNames.Value)

'Redimension arrays for selected data set
Ncases = Nfish + Nref + Nanchor

ReDim RapScore(Ncases, Nattrib) As Single
ReDim Rap01(Ncases, Nattrib) As Single
ReDim Distance(Ncases, Ncases) As Single
ReDim MDSout(Ncases, 2) As Single
ReDim MDSrotate(Ncases, 2) As Single
ReDim MDSfns(Ncases, 2) As Single
ReDim mu(Nattrib) As Single
ReDim sigma(Nattrib) As Single

' Name the spreadsheets used and clear results sheets
Set SheetIn = Worksheets("RapScores")
Set SheetTemp = Worksheets("Distances")
Set SheetOut = Worksheets("RapAnalysis")
SheetOut.Cells.ClearContents
SheetTemp.Cells.ClearContents
SheetOut.ChartObjects.Delete

'Summarize parameters on the RapAnalysis sheet
Call RapSub.ParamSummary(Ncases + 15)

'Populate matrix RapScores from RapScores spreadsheet
For i = 1 To Nfish    'real fisheries
   For j = 1 To Nattrib
      RapScore(i, j) = SheetIn.Cells(i + Ifish - 1, j + Col1stAttrib - 1)
   Next j
Next i

For j = 1 To Nattrib    'reference fisheries
   RapScore(i, j) = SheetIn.Cells(Igood, j + Col1stAttrib - 1)
   RapScore(i + 1, j) = SheetIn.Cells(Ibad, j + Col1stAttrib - 1)
   RapScore(i + 2, j) = SheetIn.Cells(Iup, j + Col1stAttrib - 1)
   RapScore(i + 3, j) = SheetIn.Cells(Idown, j + Col1stAttrib - 1)
Next j

k = i + 4
For i = k To Ncases     'anchor fisheries
   For j = 1 To Nattrib
      RapScore(i, j) = SheetIn.Cells(i + Ianchor - k, j + Col1stAttrib - 1)
   Next j
Next i

'Get fisheries names
For i = 1 To Nfish
   fishnames(i) = SheetOut.Cells(Ifish + i - 1, ColFishNames)
Next i
```

```
'Update fisheries row indices to be indices in RapScore matrix
Ifish = 1
Igood = Nfish + 1
Ibad = Nfish + 2
Iup = Nfish + 3
Idown = Nfish + 4
Ianchor = Idown + 1

'Standardize each column of attribute scores x
'according to formula (x-mu)/sigma
'Parameter mu and sigma are either calculated as:
' 1) mean and stand deviation statistics of each attribute
'    column over all fisheries (use routine CalcStats)
' or
' 2) mu = (GOOD + BAD)/2 for each attribute
'    sigma = |GOOD - BAD| for each attribute
'see RapfishForm check box
'Method 2) may give better absolute registration consistency
'for multiple Rapfish analyses with different sets of fisheries
'since the standardization scaling is the same for all.

If RapfishForm.OptionStandardize1.Value Then
   'use attribute stats
   Call RapSub.CalcStats(Ncases, Nattrib, RapScore(), mu(), sigma())
Else
   'use range of BAD to GOOD to calibrate
   For i = 1 To Nattrib
       mu(i) = (RapScore(Igood, i) + RapScore(Ibad, i)) / 2
       sigma(i) = Abs(RapScore(Igood, i) - RapScore(Ibad, i))
   Next i
End If

Call RapSub.Standardize(Ncases, Nattrib, RapScore(), _
                 mu(), sigma(), Rap01())

'Calculate Euclidean Squared distance between each
'pair of fisheries.
Call RapSub.Proximities(Rap01(), Ncases, Nattrib, Distance())

'Store Distance on temporary sheet
For i = 1 To Ncases
   For j = 1 To Ncases
      SheetTemp.Cells(i, j) = Distance(i, j)
   Next j
Next i

'Do Multi-dimensional scaling to reduce to 2 dimensions
'Send array Distance to ALSCAL Fortran DLL
N = CLng(Ncases)
ierr = g77alscal(N, N, Distance(1, 1), MDSout(1, 1), Metric(1), _
                 Sout(1), Niter, Xdebug(1), Idebug(1))

'Check for error return
Select Case ierr
   Case 0
       errstring = "no errors"
   Case 1
       errstring = "FATAL error since a dimension only has 0 weights"
   Case 141
       errstring = "Iterations stopped since achieved S-STRESS less than STMIN"
   Case 143
       errstring = "Iterations stopped since S-STRESS improvement less than EPSI"
   Case 145
       errstring = "Iterations terminated due to negative S-STRESS improvement"
   Case 299
       errstring = "Iterations stopped since #iterations has reached MAXIT"
   Case 901
       errstring = "FATAL error: all data missing for matrix (STEP1) "
   Case 902
       errstring = "FATAL error: #rows must equal #columns for non-rectangular data "
   Case 903
       errstring = "FATAL error: #stimuli less than 3 "
   Case 904
```

```
        errstring = "FATAL error: total #matrices less than 1 "
    Case 999
        errstring = "FATAL error: not enough memory (MAXSIZ exceeded) "
    Case Else
        errstring = "Unexpected error code"
End Select

If ierr = 1 Or ierr = 200 Or ierr > 900 Then
    GoTo ALSCALError
End If

'Rotate the 2D result so BAD to GOOD is horizontal left to right
Call RapSub.Rotate(MDSout(), MDSrotate(), Ibad, Igood, Ncases, theta)

'If UP fishery y coordinate < DOWN y coord then negate the
'y coordinate for all rotated 2D fisheries
'Scale so BAD is 0%, GOOD is 100%, UP is 50, and DOWN is -50
Call RapSub.FlipNScale(MDSrotate(), MDSfns(), Ibad, Igood, Iup, Idown, Ncases)

'--------------------------------------------------
'Show results in output sheet with fisheries names
'--------------------------------------------------
ColFishNames = 1
For i = 1 To Nfish
    SheetOut.Cells(i + 1, 1) = fishnames(i)
Next i
SheetOut.Cells(Igood + 1, 1) = "GOOD"
SheetOut.Cells(Ibad + 1, 1) = "BAD"
SheetOut.Cells(Iup + 1, 1) = "UP"
SheetOut.Cells(Idown + 1, 1) = "DOWN"
SheetOut.Cells(Ianchor + 1, 1) = "ANCHORS:"

'Column headings
SheetOut.Cells(1, ColFishNames + 1) = "2D MDS Results"
SheetOut.Cells(1, ColFishNames + 4) = "Rotated"
SheetOut.Cells(1, ColFishNames + 7) = "& Flipped & Scaled"

'MDS results
For i = 1 To Ncases
    For j = 1 To 2
        SheetOut.Cells(i + 1, j + ColFishNames) = MDSout(i, j)
    Next j
Next i

'Rotated results
For i = 1 To Ncases
    For j = 1 To 2
        SheetOut.Cells(i + 1, j + ColFishNames + 3) = MDSrotate(i, j)
    Next j
Next i

'Rotated, flipped, scaled results
For i = 1 To Ncases
    For j = 1 To 2
        SheetOut.Cells(i + 1, j + ColFishNames + 6) = MDSfns(i, j)
    Next j
Next i

i = 1
j = Ifish + Ncases + 5
SheetOut.Cells(j, i) = "Stress = "
SheetOut.Cells(j, i + 1) = Sout(1)
SheetOut.Cells(j + 1, i) = "Squared Correlation (RSQ) = "
SheetOut.Cells(j + 1, i + 1) = Sout(2)
SheetOut.Cells(j + 2, i) = "Number of iterations = "
SheetOut.Cells(j + 2, i + 1) = Niter
SheetOut.Cells(j + 3, i) = "Memory needed (words) = "
SheetOut.Cells(j + 3, i + 1) = Idebug(9)
SheetOut.Cells(j + 4, i) = "Rotation angle (degrees) = "
SheetOut.Cells(j + 4, i + 1) = theta

SheetOut.Cells(j + 5, i) = " "
SheetOut.Cells(j + 6, i) = "Return value ierr = " & Str(ierr)
SheetOut.Cells(j + 7, i) = errstring
```

```
i = 5
SheetOut.Cells(j, i) = "Iteration"
SheetOut.Cells(j, i + 1) = "Stress"
SheetOut.Cells(j, i + 2) = "Delta "
For k = 1 To Niter
SheetOut.Cells(j + k, i) = k
SheetOut.Cells(j + k, i + 1) = Metric(2 * k - 1)
SheetOut.Cells(j + k, i + 2) = Metric(2 * k)
Next k

'-----------------------------
'Plot Rapfish 2D Ordination
'  - rotated, scaled, flipped
'-----------------------------
Application.ScreenUpdating = False  'suppress plot output till end for efficiency

'Create a chart in the Rapfish Analysis spreadsheet
'and specify its position and size all in screen points:
'   Top & Left - location relative to cell A1
'   Width & Height - chart size parameters
Set ch = SheetOut.ChartObjects.Add(50, 50, 500, 400)
ch.Activate           'paramaters (Top,Left,Width,Height)

'Define chart characteristics
range_str = "A2:A" & CStr(Nfish + 1)
With ActiveChart
   .SetSourceData Source:=Sheets("RapAnalysis").Range("H2")
   .Location Where:=xlLocationAsObject, Name:="RapAnalysis"
   .ChartType = xlXYScatter
   .HasLegend = True
   .Legend.Position = xlLegendPositionRight
   .Legend.Border.LineStyle = xlLineStyleNone
   'the following switches chart border off ... tjp
   .ChartArea.Border.LineStyle = xlLineStyleNone
End With

'Plot real fisheries in series 1
range_str = "A2:A" & CStr(Nfish + 1)
With ActiveChart.SeriesCollection(1)
   .XValues = SheetOut.Range(range_str).Offset(0, 7)
   .Values = SheetOut.Range(range_str).Offset(0, 8)
   .MarkerStyle = xlMarkerStyleCircle
   .MarkerSize = 8
   .MarkerBackgroundColorIndex = 2
   .MarkerForegroundColorIndex = 1
   .Name = "Real Fisheries"
End With

'Plot reference fisheries in series 2
ActiveChart.SeriesCollection.NewSeries
range_str = "A" & CStr(Nfish + 2) & ":A" & CStr(Nfish + Nref + 1)
With ActiveChart.SeriesCollection(2)
   .XValues = SheetOut.Range(range_str).Offset(0, 7)
   .Values = SheetOut.Range(range_str).Offset(0, 8)
   .Name = "Reference anchors"
   .MarkerStyle = xlMarkerStyleDiamond
   .MarkerSize = 9
   .MarkerBackgroundColorIndex = 6
   .MarkerForegroundColorIndex = 1
   .Points(1).HasDataLabel = True
   .Points(1).DataLabel.Text = "Good"
   .Points(1).DataLabel.Position = xlLabelPositionAbove
   .Points(2).HasDataLabel = True
   .Points(2).DataLabel.Text = "Bad"
   .Points(2).DataLabel.Position = xlLabelPositionAbove
   .Points(3).HasDataLabel = True
   .Points(3).DataLabel.Text = "Up"
   .Points(3).DataLabel.Position = xlLabelPositionAbove
   .Points(4).HasDataLabel = True
   .Points(4).DataLabel.Text = "Down"
   .Points(4).DataLabel.Position = xlLabelPositionBelow
End With
```

```
'Plot anchor stabilizer fisheries (simulated) in series 3
ActiveChart.SeriesCollection.NewSeries
range_str = "A" & CStr(Nfish + Nref + 2) & ":A" & CStr(Ncases + 1)
With ActiveChart.SeriesCollection(3)
   .XValues = SheetOut.Range(range_str).Offset(0, 7)
   .Values = SheetOut.Range(range_str).Offset(0, 8)
   .MarkerStyle = xlMarkerStyleTriangle
   .MarkerSize = 5
   .MarkerBackgroundColorIndex = 6
   .MarkerForegroundColorIndex = 1
   .Name = "Anchors"
End With

'Add chart titles, etc.
With ActiveChart
    .HasTitle = True
    .ChartTitle.Characters.Text = _
        "RAPFISH Ordination"
End With

With ActiveChart.Axes(xlCategory)   'y axis
    .HasTitle = True
    .AxisTitle.Characters.Text = _
        "Fisheries Status"
    .HasMajorGridlines = True
    .MajorGridlines.Border.Color = RGB(169, 169, 169)
    .MajorGridlines.Border.LineStyle = xlContinuous
End With

With ActiveChart.Axes(xlValue)   'x axis
    .HasTitle = True
    .AxisTitle.Characters.Text = _
        "Other Distingishing Features"
    .HasMajorGridlines = True
    .MajorGridlines.Border.Color = RGB(169, 169, 169)
    .MajorGridlines.Border.LineStyle = xlContinuous
End With
Exit Sub

RapErrorHandler:
msg = "Check Rapfish parameters & data. Error # = " & Str(Err) & ": " & Error(Err)
If MsgBox(msg, vbExclamation + vbOKCancel) = vbOK Then
'   Resume Next
End If
Exit Sub

ALSCALError:
If MsgBox(errstring, vbExclamation + vbOKCancel) = vbOK Then
End If

End Sub

MODULE RAPSUB

'-------------------------------------------------------------
'This module contains Rapfish Visual Basic subroutines:
'   CalcStats
'   Standardize
'   Proximities
'   Rotate
'   FlipNScale
'   GaussianRV
'   ParamSummary
'   ExcelColNum
'
'History:
'  P.Kavanagh   March 2001     new code
'-------------------------------------------------------------
Option Explicit

Dim msg As String

Sub CalcStats(Nrow As Integer, Ncol As Integer, X() As Single, _
            mu() As Single, sigma() As Single)
```

```
'Calculate mean and standard deviation of a vector
Dim i As Integer, j As Integer
Dim sum As Single, sumsq As Single

For j = 1 To Ncol
   sum = 0
   sumsq = 0
   For i = 1 To Nrow
      sum = sum + X(i, j)
      sumsq = sumsq + (X(i, j) ^ 2)
   Next i
   mu(j) = sum / Nrow
   sigma(j) = Sqr((sumsq - (sum ^ 2) / Nrow) / (Nrow - 1))
Next j
End Sub

Sub Standardize(Nrow As Integer, Ncol As Integer, _
                X() As Single, _
                mu() As Single, sigma() As Single, _
                X01() As Single)

'Standardize each column of matrix X by subtracting mean
'and dividing by the standard deviation, so that the
'returned matrix X01 has mean=0 and sigma=1

Dim i As Integer, j As Integer

For j = 1 To Ncol
   For i = 1 To Nrow
      X01(i, j) = (X(i, j) - mu(j)) / sigma(j)
   Next i
Next j

End Sub

Sub Proximities(X() As Single, _
                Nrow As Integer, Ncol As Integer, _
                Seuclid() As Single)

'Calculate the squared Euclidean distance matrix Seuclid
'for a matrix X by calculating the squared Euclidean
'distance between each pair of row vectors in the matrix.
'Resulting distance matrix is symmetric with zeros along
'the diagonal, so only need to calculate the lower left
'triangle below the diagonal.

Dim i1 As Integer, i2 As Integer, j As Integer
Dim S As Single

For i1 = 1 To (Nrow - 1)
   For i2 = (i1 + 1) To Nrow
      S = 0
      For j = 1 To Ncol
         S = S + (X(i1, j) - X(i2, j)) ^ 2
      Next j
      Seuclid(i2, i1) = S
   Next i2
Next i1

End Sub

Sub Rotate(V() As Single, Vrotate() As Single, _
           Ibad As Integer, Igood As Integer, _
           N As Integer, theta As Single)

Dim X As Single
Dim y As Single
Dim mag As Single
Dim phase As Single
Dim pi As Single
Dim Xdiff As Single
Dim Ydiff As Single
Dim i As Integer
```

```
Dim diddlysquat As Single

'Error handling
'On Local Error Resume Next
On Local Error GoTo RotateErrorHandler

'DoEvents

'Rotate the MDS 2D result so that the BAD to GOOD axis
'is horizontal. Then check for the UP fishery at the top
'relative to the DOWN fishery; if NOT then FLIP (negate)
'the Y cordinate (2nd vertical) of all rotated results.

'Calculate angle of vector from BAD to GOOD
'Note: atan gives angle in range -pi to +pi

diddlysquat = 1E-20
Xdiff = 0
Xdiff = V(Igood, 1) - V(Ibad, 1)
Ydiff = V(Igood, 2) - V(Ibad, 2)
'pi = 3.14159265359
pi = 4 * Atn(1)
If Abs(Xdiff) < diddlysquat Then
   Xdiff = diddlysquat      'avoid divide by zero or v. small number
End If
theta = Atn(Ydiff / Xdiff) * 180 / pi
If Xdiff < 0 Then
   theta = theta + 180   'adjust quadrant
End If

'Rotate all values by -angle. Convert each MDSout value
'to polar coordinates (magnitude and angle), apply the
'rotation, and then convert back to xy coordinates
For i = 1 To N
   X = V(i, 1)
   y = V(i, 2)
   mag = Sqr(X ^ 2 + y ^ 2)
   If Abs(X) < diddlysquat Then
      X = diddlysquat 'avoid divide by zero or small number
   End If
   phase = Atn(y / X) * 180 / pi
   If X < 0 Then
      phase = phase + 180   'adjust quadrant
   End If
   phase = (phase - theta) * pi / 180
   Vrotate(i, 1) = mag * Cos(phase)
   Vrotate(i, 2) = mag * Sin(phase)
Next i
DoEvents

Exit Sub

RotateErrorHandler:
msg = "BAD news!!!! Rotate error # = " & Str(Err.Number) & ": " & Error(Err)
If MsgBox(msg, vbExclamation + vbOKCancel) = vbOK Then
   Resume Next
End If

End Sub

Sub FlipNScale(V() As Single, Vfns() As Single, _
           Ibad As Integer, Igood As Integer, _
           Iup As Integer, Idown As Integer, _
           N As Integer)

Dim i As Integer
Dim squat As Single

squat = 1E-20

'On Local Error Resume Next
On Local Error GoTo FlipErrorHandler

'Negate y coordinate (vertical) if Up fishery < Down fishery
```

```
If V(Iup, 2) < V(Idown, 2) Then
   For i = 1 To N
      V(i, 2) = -V(i, 2)
   Next i
End If

'Scale ARBITARILY so Badx to Goodx is 0 to 100, Downy is -50 and Upy is +50
For i = 1 To N
   Vfns(i, 1) = 100 * (V(i, 1) - V(Ibad, 1)) / (V(Igood, 1) - V(Ibad, 1) + squat)
   Vfns(i, 2) = (100 * (V(i, 2) - V(Idown, 2)) / (V(Iup, 2) - V(Idown, 2) + squat)) - 50
Next i

'Put Bad to Good horizontal axis at y=0
For i = 1 To N
   Vfns(i, 2) = Vfns(i, 2) - Vfns(Igood, 2)
Next i

DoEvents

Exit Sub

FlipErrorHandler:
msg = "BAD News!!!! Flip error # = " & Str(Err.Number) & ": " & Error(Err)
If MsgBox(msg, vbExclamation + vbOKCancel) = vbOK Then
   Resume Next
End If

End Sub

Sub GaussianRV(mean As Single, sigma As Single, _
               G1 As Single, G2 As Single)
'Generate two independent gaussian random variables
'X1 and X2 each with specified mean and standard deviation.
'Use the Visual Basic Uniform random number generator
'with a transformation to do this.

Dim X1 As Single
Dim X2 As Single
Dim U1 As Single
Dim U2 As Single
Dim r As Single
Dim theta As Single
Dim pi As Single

'pi = 3.14159265359
pi = 4 * Atn(1)

U1 = Rnd      'uniform rv over [0 to 1]
U2 = Rnd
r = Sqr(-2 * Log(U1))
theta = 2 * pi * U2

X1 = mean + (r * sigma * Cos(theta))
X2 = mean + (r * sigma * Sin(theta))

End Sub

Function GRV(mean As Single, sigma As Single) As Single
'Return a sample from a gaussian distribution
'with specified mean and standard deviation.
'Use the Visual Basic Uniform random number generator
'with a transformation to do this.

Dim X1 As Single
Dim X2 As Single
Dim U1 As Single
Dim U2 As Single
Dim r As Single
Dim theta As Single
Dim pi As Single

'pi = 3.14159265359
pi = 4 * Atn(1)
```

```
U1 = Rnd      'uniform rv over [0 to 1]
U2 = Rnd
r = Sqr(-2 * Log(U1))
theta = 2 * pi * U2

X1 = mean + (r * sigma * Cos(theta))
X2 = mean + (r * sigma * Sin(theta))

GRV = X1
End Function

Function TriangleRV(ErrLow As Single, ErrHi As Single) As Single
'Return a sample from an assymetric triangle probability distribution
'function with peak probability at 0.
'  Inputs: ErrLow = minimum expected error (-ve)
'          ErrHi = maximum expected error (+ve)
'The RV is generated as follows:
'1) Generate independent samples U1 and U2 from a Uniform (flat)
'   distribution on range 0 to ErrLow. Generate sample from a
'   lower triangle (neagitive only) PDF as T1 = -|U1 - U2|.
'2) Generate independent samples U3 and U4 from a Uniform (flat)
'   distribution on range 0 to ErrHi. Generate sample from a
'   lower triangle (positive only) PDF as T2 = |U3 - U4|.
'3) Calculate the desired probability of being negative
'   as the area of the lower triangle P = ErrLow/(ErrLow+ErrHi)
'4) Generate uniform RV sample F over 0 to 1
'4) Generate output TriangleRV PDF sample as:
'             TriangleRV = T1 if F < P
'                          T2 if F >= P
'NOTES: The PDF for the sum of 2 independent RVs is the
'       convolution of the individual PDF's. This is how
'       we get the triangle from 2 uniform RVs)
'The mean is not 0, but the most likely value is 0.
'The signs of ErrLow and ErrHi are ignored (code uses |ErrHi|
'and |ErrLow|) so user need not worry about sign.

Dim U1 As Single
Dim U2 As Single
Dim U3 As Single
Dim U4 As Single
Dim T1 As Single
Dim T2 As Single
Dim F As Single
Dim P As Single

ErrLow = Abs(ErrLow)
ErrHi = Abs(ErrHi)
U1 = ErrLow * Rnd
U2 = ErrLow * Rnd
T1 = -Abs(U1 - U2)
U3 = ErrHi * Rnd
U4 = ErrHi * Rnd
T2 = Abs(U3 - U4)
F = Rnd
P = ErrLow / (ErrLow + ErrHi)
If F < P Then
   TriangleRV = T1
Else
   TriangleRV = T2
End If

End Function

Function SkewRV(sigma1 As Single, sigma2 As Single) As Single
'Return a sample from a 0 mean "skewed gaussian" distribution
'The skewed random number generator is implemented as:
'      - create 2 independent Gaussian RVs x1 and x2, each with
'        0 mean and standard deviations sigma1 and sigma2
'      - a skewed PDF RV is calculated as:
'            y = |x1| - |x2|
'        (= convolution of |x1| and |x2|)
'Adjust the mean to be zero, if desired.
'The problem with this is it there's no obvious way to pre-compute
'the sigma1 and sigma2 to use to get desired shape of tails.
```

```
Dim r1 As Single
Dim r2 As Single

r1 = GRV(0, sigma1)  'Gaussian RV
r2 = GRV(0, sigma2)  'another one
SkewRV = Abs(r1) - Abs(r2)

End Function

Sub ParamSummary(row As Integer)

Dim SheetParam As Variant
Dim col1 As Integer
Dim col2 As Integer

Set SheetParam = Worksheets("RapAnalysis")

col1 = 1
col2 = 2
SheetParam.Cells(row, col1) = "RAPFISH PARAMETERS USED FOR THIS ANALYSIS"
row = row + 1
SheetParam.Cells(row, col1) = "# fisheries =  "
SheetParam.Cells(row, col2) = RapfishForm.Numfish.Value
row = row + 1

SheetParam.Cells(row, col1) = "# reference fisheries =  "
SheetParam.Cells(row, col2) = RapfishForm.Numref.Value
row = row + 1

SheetParam.Cells(row, col1) = "# anchor fisheries =  "
SheetParam.Cells(row, col2) = RapfishForm.Numanchor.Value
row = row + 1

SheetParam.Cells(row, col1) = "Row# of 1st fishery =  "
SheetParam.Cells(row, col2) = RapfishForm.Row1stFISH.Value
row = row + 1

SheetParam.Cells(row, col1) = "Row# of GOOD fishery =  "
SheetParam.Cells(row, col2) = RapfishForm.RowGOOD.Value
row = row + 1

SheetParam.Cells(row, col1) = "Row# of BAD fishery =  "
SheetParam.Cells(row, col2) = RapfishForm.RowBAD.Value
row = row + 1

SheetParam.Cells(row, col1) = "Row# of UP fishery =  "
SheetParam.Cells(row, col2) = RapfishForm.RowUP.Value
row = row + 1

SheetParam.Cells(row, col1) = "Row# of DOWN fishery =  "
SheetParam.Cells(row, col2) = RapfishForm.RowDOWN.Value
row = row + 1

SheetParam.Cells(row, col1) = "Column letter with fisheries names =  "
SheetParam.Cells(row, col2) = RapfishForm.ColFishNames.Value
row = row + 1

SheetParam.Cells(row, col1) = "Row# of 1st anchor fishery =  "
SheetParam.Cells(row, col2) = RapfishForm.Row1stAnchor.Value
row = row + 1

SheetParam.Cells(row, col1) = "# attributes =  "
SheetParam.Cells(row, col2) = RapfishForm.Numattrib.Value
row = row + 1

SheetParam.Cells(row, col1) = "Column letter of 1st attribute =  "
SheetParam.Cells(row, col2) = RapfishForm.Col1stATTRIBUTE.Value
row = row + 1

End Sub

Function ExcelColNum(S As Variant) As Integer
'Convert an Excel column letter designator (1 or 2 letters)
```

```
'to a column number as follows:
'   A, B, C, .... Z, AA, AB, ... BA, BB, ..
'   1, 2, 3, ... 26, 27, 28, ... 52, 53
'With the column as a number we can use Worksheet().Cells to access
'spreadsheet cells numerically (similar to rows).
'I could NOT find a readymade function to do this simple task!!

Dim string2 As String * 2
Dim N As Integer, N2 As Integer

string2 = S
N = Asc(string2) - 64                '1st character converted
'N2 = Asc(StrReverse(string2)) - 64   '2nd character converted
N2 = Asc(Right(string2, 1)) - 64  '2nd character converted
If (N2 >= 1 And N2 <= 26) Then   'if 2 letters present, then add together,
   N = (N * 26) + N2             'scaling 1st character by # letters in alphabet
End If
ExcelColNum = N

End Function

Function ExcelColLetters(colnum As Integer) As Variant
'Convert an Excel column number to a 1 or 2 letter designator as follows:
'   1, 2, 3, ... 26, 27, 28, ... 52, 53
'   A, B, C, .... Z, AA, AB, ... BA, BB, ..
'With the column as a letter we can use Worksheet().Range to access
'spreadsheet ranges.
'I could NOT find a readymade function to do this simple task!!

Dim N As Integer, N2 As Integer

N = Fix(colnum / 26)
N2 = colnum - (N * 26)
If N <= 0 Then
   ExcelColLetters = Chr(64 + N2)
Else
   ExcelColLetters = Chr(64 + N) & Chr(64 + N2)
End If

End Function

MODULE LEVERAGING

'---------------------------------------------------------------
'This module contains the Rapfish leveraging
'analysis routines:
'     JackKnifeAttributes - repeat analysis each time removing
'                           an attribute and compare to original
'     JackKnifeFisheries - repeat analysis each time removing
'                           a fishery and compare to original
'                           (STILL TO DO - NOT A PRIORITY)
'
'Subroutines used are in RapSub.
'
'The Fortran DLL g77alscal.dll must be resident in the
'Excel default directory for the code to work.
'See Main for description of error returns from ALSCAL
'
'History:
'  P.Kavanagh   March 2001     new code
'---------------------------------------------------------------

Option Explicit
Option Base 1     'indices for arrays/matrices start at 1 (not 0)

Dim msg As String

'Fortran DLL for multi-dimensional scaling
'(derived from Forrest Young code)
Declare Function g77alscal Lib "g77alscal.dll" _
Alias "g77alscal_" _
(ByRef Nrows As Long, _
ByRef Ncols As Long, _
ByRef Xinput As Single, _
```

```
ByRef MDSout As Single, _
ByRef Metric As Single, _
ByRef Sout As Single, _
ByRef Niter As Long, _
ByRef Xdebug As Single, _
ByRef Idebug As Long) As Long

Sub JackKnifeAttributes()
'Run Rapfish multiple times with different groups of
'(real) fisheries and check the variation in anchor
'and reference fishery locations. Use a FIXED mean
'and sigma to standardize all the fisheries.
'  first run - use the entered fisheries
'            - save the mean and sigma for each attribute
'  runs i = 1 to Nattrib - remove attribute i and repeat MDS

'Number of fisheries
Dim Nfish As Integer      'real fisheries
Dim Nref As Integer       'good, bad, up, down
Dim Nanchor As Integer    'fake fisheries to assist anchoring
Dim Ncases As Integer

'Fisheries attributes and categories
Dim Ncategories As Integer  'eg. technological, etc.
Dim Nattrib As Integer  '# attributes per category

'Excel spreadsheet location for fishery 1, attribute 1
'for each of the Ncategories of attributes
Dim InitRow As Integer
Dim InitCol As Integer

'Row indices to fisheries data
Dim Ifish As Integer
Dim Igood As Integer
Dim Ibad As Integer
Dim Iup As Integer
Dim Idown As Integer
Dim Ianchor As Integer

'Column index to attributes
Dim Col1stAttrib As Integer

'Data
Dim SheetIn As Variant
Dim SheetTemp As Variant
Dim SheetOut As Variant
Dim SheetMC As Variant

Dim RapScore() As Single       'Rapfish scores from users
Dim Rap01() As Single          'standardized RapScore
Dim Rap01LevAttrib() As Single     'with error added

Dim Distance() As Single
Dim MDSout() As Single
Dim MDSrotate() As Single
Dim MDSfns() As Single
Dim MDSfns_all() As Single
Dim Sout(2) As Single
Dim Niter As Long
Dim Metric(60) As Single
Dim Xdebug(5000) As Single
Dim Idebug(30) As Long
Dim ierr As Long
Dim N As Long
Dim ColFishNames As Integer
Dim fishnames(100) As String

'Other
Dim i As Integer, j As Integer, k As Integer
Dim mu() As Single
Dim sigma() As Single
Dim ch As Variant
Dim case1_xstr As Variant
Dim Nspins As Integer
```

```
Dim levsumX() As Single
Dim levsumY() As Single
Dim AttributeNames As Variant
Dim strattr As String

'angle of the BAD to GOOD vector in degrees before rotation
'rotate each point in 2D MDS output by -theta
'to make the BAD to GOOD line horizontal
Dim theta As Single

'On Error GoTo 0    ' Turn off error trapping.
'On Error Resume Next
On Error GoTo LevErrorHandler

'Default values of parameters
Ifish = 1
Igood = 27
Ibad = 28
Iup = 29
Idown = 30
Ianchor = 31

Nfish = 26
Nref = 4
Nanchor = 16
Nattrib = 10

Nspins = Nattrib      'for attribute levergaing

'Load current parameters from RapfishForm into local variables.
Nfish = RapfishForm.Numfish.Value
Nref = RapfishForm.Numref.Value
Nanchor = RapfishForm.Numanchor.Value
Nattrib = RapfishForm.Numattrib.Value

Ifish = RapfishForm.Row1stFISH.Value
Igood = RapfishForm.RowGOOD.Value
Ibad = RapfishForm.RowBAD.Value
Iup = RapfishForm.RowUP.Value
Idown = RapfishForm.RowDOWN.Value
Ianchor = RapfishForm.Row1stAnchor.Value

'Convert Excel column letters to column numbers
Col1stAttrib = RapSub.ExcelColNum(RapfishForm.Col1stATTRIBUTE.Value)
ColFishNames = RapSub.ExcelColNum(RapfishForm.ColFishNames.Value)

'Redimension arrays for selected data set
Ncases = Nfish + Nref + Nanchor

ReDim RapScore(Ncases, Nattrib) As Single
ReDim Rap01LevAttrib(Ncases, Nattrib - 1) As Single

ReDim Rap01(Ncases, Nattrib) As Single
ReDim Distance(Ncases, Ncases) As Single
ReDim MDSout(Ncases, 2) As Single
ReDim MDSrotate(Ncases, 2) As Single
ReDim MDSfns(Ncases, 2) As Single
ReDim MDSfns_all(Ncases, 2) As Single
ReDim mu(Nattrib) As Single
ReDim sigma(Nattrib) As Single
ReDim levsumX(Nattrib) As Single
ReDim levsumY(Nattrib) As Single

'Name the spreadsheets used
Set SheetIn = Worksheets("RapScores")
Set SheetTemp = Worksheets("Distances")
Set SheetOut = Worksheets("RapAnalysis")
Set SheetMC = Worksheets("LeverageAttributes")

'Clear results sheets
SheetTemp.Cells.ClearContents
SheetOut.Cells.ClearContents
SheetMC.Cells.ClearContents
SheetMC.ChartObjects.Delete
```

```
'Create Range string containing attribute names
AttributeNames = RapfishForm.Col1stATTRIBUTE.Value & "1:" & _
                 RapSub.ExcelColLetters(Col1stAttrib + Nattrib - 1) & "1"

'Summarize parameters on the RapAnalysis sheet
Call RapSub.ParamSummary(Ncases + 15)

'Populate matrix RapScores from RapScores spreadsheet
For i = 1 To Nfish
   For j = 1 To Nattrib
      RapScore(i, j) = SheetIn.Cells(i + Ifish - 1, j + Col1stAttrib - 1)
   Next j
Next i

For j = 1 To Nattrib
   RapScore(i, j) = SheetIn.Cells(Igood, j + Col1stAttrib - 1)
   RapScore(i + 1, j) = SheetIn.Cells(Ibad, j + Col1stAttrib - 1)
   RapScore(i + 2, j) = SheetIn.Cells(Iup, j + Col1stAttrib - 1)
   RapScore(i + 3, j) = SheetIn.Cells(Idown, j + Col1stAttrib - 1)
Next j

k = i + 4
For i = k To Ncases
   For j = 1 To Nattrib
      RapScore(i, j) = SheetIn.Cells(i + Ianchor - k, j + Col1stAttrib - 1)
   Next j
Next i

'Get fisheries names
For i = 1 To Nfish
   fishnames(i) = SheetIn.Cells(Ifish + i - 1, ColFishNames)
Next i

'Update fisheries row indices to be indices in RapScore matrix
Ifish = 1
Igood = Nfish + 1
Ibad = Nfish + 2
Iup = Nfish + 3
Idown = Nfish + 4
Ianchor = Idown + 1

'Standardize each column of attribute scores x
'according to formula (x-mu)/sigma
'Parameter mu and sigma are either calculated as:
' 1) mean and stand deviation statistics of each attribute
'    column over all fisheries (use routine CalcStats)
' or
' 2) mu = (GOOD + BAD)/2 for each attribute
'    sigma = |GOOD - BAD| for each attribute
'see RapfishForm check box
'Method 2) may give better absolute registration consistency
'for multiple Rapfish analyses with different sets of fisheries
'since the standardization scaling is the same for all.

If RapfishForm.OptionStandardize1.Value Then
   'use attribute stats
   Call RapSub.CalcStats(Ncases, Nattrib, RapScore(), mu(), sigma())
Else
   'use range of BAD to GOOD to calibrate
   For i = 1 To Nattrib
       mu(i) = (RapScore(Igood, i) + RapScore(Ibad, i)) / 2
       sigma(i) = Abs(RapScore(Igood, i) - RapScore(Ibad, i))
   Next i
End If

Call RapSub.Standardize(Ncases, Nattrib, RapScore(), _
                 mu(), sigma(), Rap01())

'Calculate Euclidean Squared distance between each
'pair of fisheries.
Call RapSub.Proximities(Rap01(), Ncases, Nattrib, Distance())

'Store Distance on temporary sheet
```

```
For i = 1 To Ncases
   For j = 1 To Ncases
      SheetTemp.Cells(i, j) = Distance(i, j)
   Next j
Next i

'Do Multi-dimensional scaling to reduce to 2 dimensions
'Send array Distance to ALSCAL Fortran DLL
N = CLng(Ncases)
ierr = g77alscal(N, N, Distance(1, 1), MDSout(1, 1), Metric(1), _
                 Sout(1), Niter, Xdebug(1), Idebug(1))

'Rotate the 2D result so BAD to GOOD is horizontal left to right
Call RapSub.Rotate(MDSout(), MDSrotate(), Ibad, Igood, Ncases, theta)

'If UP fishery y coordinate < DOWN y coord then negate the
'y coordinate for all rotated 2D fisheries
'Scale so BAD is 0%, GOOD is 100%, UP is 50, and DOWN is -50
Call RapSub.FlipNScale(MDSrotate(), MDSfns(), Ibad, Igood, Iup, Idown, Ncases)

'-------------------------------------------------
'Show results in output sheet with fisheries names
'-------------------------------------------------
For i = 1 To Nfish
   SheetOut.Cells(i + 1, ColFishNames) = fishnames(i)
Next i
SheetOut.Cells(Igood + 1, ColFishNames) = "GOOD"
SheetOut.Cells(Ibad + 1, ColFishNames) = "BAD"
SheetOut.Cells(Iup + 1, ColFishNames) = "UP"
SheetOut.Cells(Idown + 1, ColFishNames) = "DOWN"
SheetOut.Cells(Ianchor + 1, ColFishNames) = "ANCHORS:"

'Column headings
SheetOut.Cells(1, ColFishNames + 1) = "2D MDS Results"
SheetOut.Cells(1, ColFishNames + 4) = "Rotated"
SheetOut.Cells(1, ColFishNames + 7) = "& Flipped & Scaled"

'MDS results
For i = 1 To Ncases
   For j = 1 To 2
      SheetOut.Cells(i + 1, j + ColFishNames) = MDSout(i, j)
   Next j
Next i

'Rotated results
For i = 1 To Ncases
   For j = 1 To 2
      SheetOut.Cells(i + 1, j + ColFishNames + 3) = MDSrotate(i, j)
   Next j
Next i

'Rotated, flipped, scaled results
For i = 1 To Ncases
   For j = 1 To 2
      SheetOut.Cells(i + 1, j + ColFishNames + 6) = MDSfns(i, j)
   Next j
Next i

i = 1
j = Ifish + Ncases + 5
SheetOut.Cells(j, i) = "Stress = "
SheetOut.Cells(j, i + 1) = Sout(1)
SheetOut.Cells(j + 1, i) = "Squared Correlation (RSQ) = "
SheetOut.Cells(j + 1, i + 1) = Sout(2)
SheetOut.Cells(j + 2, i) = "Number of iterations = "
SheetOut.Cells(j + 2, i + 1) = Niter
SheetOut.Cells(j + 3, i) = "Memory needed (words) = "
SheetOut.Cells(j + 3, i + 1) = Idebug(9)
SheetOut.Cells(j + 4, i) = "Return value (error if > 0) "
SheetOut.Cells(j + 4, i + 1) = ierr
SheetOut.Cells(j + 5, i) = "Rotation angle (degrees) = "
SheetOut.Cells(j + 5, i + 1) = theta

i = 4
```

```
SheetOut.Cells(j, i) = "Iteration"
SheetOut.Cells(j, i + 1) = "Stress"
SheetOut.Cells(j, i + 2) = "Delta "
For k = 1 To Niter
SheetOut.Cells(j + k, i) = k
SheetOut.Cells(j + k, i + 1) = Metric(2 * k - 1)
SheetOut.Cells(j + k, i + 2) = Metric(2 * k)
Next k

'-----------------------------------------------------------------------
'NOW that we're finished with the data entry, setup, and original data run
'repeat Nattrib times, each time removing a differnt attribute.
'Then calculate the standard error for each removed attribute for  X and Y:
'    Calculate sum of the squared differences from the original for X and Y
'    Divide by number of fisheries Nfish
'    Take square root
'Plot these sums in a bar chart
'-----------------------------------------------------------------------

'First save the baseline analysis with all attributes
For i = 1 To Ncases
   MDSfns_all(i, 1) = MDSfns(i, 1)
   MDSfns_all(i, 2) = MDSfns(i, 2)
Next i

For k = 1 To Nattrib

    'Remove attribute k from the standardized Rapfish scores
    'Save in Rap01LevAttrib
    For i = 1 To Ncases
       For j = 1 To k - 1
          Rap01LevAttrib(i, j) = Rap01(i, j)
       Next j
       For j = k + 1 To Nattrib
          Rap01LevAttrib(i, j - 1) = Rap01(i, j)
       Next j
    Next i

    'Calculate Euclidean Squared distance between each pair of fisheries.
    'Number of attributes is reduce by 1
    Call RapSub.Proximities(Rap01LevAttrib(), Ncases, Nattrib - 1, Distance())

    'Do Multi-dimensional scaling to reduce to 2 dimensions
    'Send array Distance to ALSCAL Fortran DLL
    N = CLng(Ncases)
    ierr = g77alscal(N, N, Distance(1, 1), MDSout(1, 1), Metric(1), _
                     Sout(1), Niter, Xdebug(1), Idebug(1))

    If ierr = 1 Or ierr = 200 Or ierr > 900 Then
       GoTo ALSCALError
    End If

    'Rotate the 2D result so BAD to GOOD is horizontal left to right
    Call RapSub.Rotate(MDSout(), MDSrotate(), Ibad, Igood, Ncases, theta)

    'If UP fishery y coordinate < DOWN y coord then negate the
    'y coordinate for all rotated 2D fisheries
    'Scale so BAD is 0%, GOOD is 100%, UP is 50, and DOWN is -50
    Call RapSub.FlipNScale(MDSrotate(), MDSfns(), Ibad, Igood, Iup, Idown, Ncases)

    'Save output to spreadsheet and plot
    For i = 1 To Ncases
        SheetMC.Cells(k, i) = MDSfns(i, 1)
        SheetMC.Cells(k + Nspins, i) = MDSfns(i, 2)
        'instead, plot unscaled results to see if scaling has any effect
        'SheetMC.Cells(k, i) = MDSrotate(i, 1)
        'SheetMC.Cells(k + Nspins, i) = MDSrotate(i, 2)
    Next i

 Next k

'Calculate sum of the squared differences between baseline analysis
'with all attributes and each analysis with attribute k removed.
'Divide by Nfish and take square root to get standard error.
```

```
For k = 1 To Nattrib
   levsumX(k) = 0
   levsumY(k) = 0
   For i = 1 To Nfish
      levsumX(k) = levsumX(k) + (SheetMC.Cells(k, i) - MDSfns_all(i, 1)) ^ 2
      levsumY(k) = levsumY(k) + (SheetMC.Cells(k + Nattrib, i) - MDSfns_all(i, 2)) ^ 2
   Next i
   SheetMC.Cells(Nspins + Nspins + k, 1) = Sqr(levsumX(k) / Nfish) 'Save to spreadsheet
   SheetMC.Cells(Nspins + Nspins + k, 2) = Sqr(levsumY(k) / Nfish)
Next k

'-------------------------------------------------------
'Plot the resulting fisheries repeatedly to observe the
'leveraging scatter for each fishery.
'-------------------------------------------------------
Application.ScreenUpdating = False
Set ch = SheetMC.ChartObjects.Add(30, 10, 450, 450)
ch.Activate           'parameters (Top,Left,Width,Height)

With ActiveChart
   .ChartType = xlXYScatter
   .SetSourceData Source:=Sheets("LeverageAttributes").Range("F10")
   .Location Where:=xlLocationAsObject, Name:="LeverageAttributes"

End With
'Plot each fishery as a series
For k = 1 To Ncases
   ActiveChart.SeriesCollection.NewSeries
   case1_xstr = "A1:A" & CStr(Nspins)
   ActiveChart.SeriesCollection(k).XValues = SheetMC.Range(case1_xstr).Offset(0, k - 1)
   ActiveChart.SeriesCollection(k).Values = SheetMC.Range(case1_xstr).Offset(Nspins, k - 1)
Next k
With ActiveChart
   .HasLegend = False
   .HasTitle = True
   .ChartTitle.Characters.Text = "Leveraging Scatter - remove 1 attribute each run"
   .Axes(xlCategory, xlPrimary).HasTitle = True
   .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = "Fisheries Sustainability"
   .Axes(xlValue, xlPrimary).HasTitle = True
   .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = _
        "Other Distinguishing Features"
   .ChartArea.Border.LineStyle = xlLineStyleNone
End With

'-------------------------------------------------------
'Plot bar chart of the sum of the squared differences between the
'baseline with all attributes and each analysis with attribute k removed
'-------------------------------------------------------
case1_xstr = "A" & CStr(Nspins + Nspins + 1) & ":A" & CStr(Nspins + Nspins + Nattrib)
Set ch = SheetMC.ChartObjects.Add(40, 200, 450, 350)

ch.Activate           'parameters (Top,Left,Width,Height)
ActiveChart.ChartType = xlAreaStacked
ActiveChart.SetSourceData Source:=Sheets("LeverageAttributes").Range("F24")
ActiveChart.Location Where:=xlLocationAsObject, Name:="LeverageAttributes"
ActiveChart.ChartType = xlBarClustered
ActiveChart.ChartArea.Border.LineStyle = xlLineStyleNone
ActiveChart.ChartGroups(1).GapWidth = 45
ActiveChart.SeriesCollection.NewSeries
ActiveChart.SeriesCollection(1).XValues = SheetIn.Range(AttributeNames)
ActiveChart.SeriesCollection(1).Values = SheetMC.Range(case1_xstr)
ActiveChart.HasLegend = False
ActiveChart.HasTitle = True
ActiveChart.SeriesCollection(1).Border.LineStyle = xlLineStyleNone
ActiveChart.SeriesCollection(1).Interior.ColorIndex = 5
ActiveChart.ChartTitle.Characters.Text = "Leverage of Attributes"
ActiveChart.Axes(xlCategory, xlPrimary).HasTitle = True
ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = "Attribute"
ActiveChart.Axes(xlValue, xlPrimary).HasTitle = True
ActiveChart.Axes(xlValue, xlPrimary).HasMajorGridlines = False
ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = _
        "Root Mean Square Change % in Ordination when Selected Attribute Removed (on Status scale
0 to 100)"
With ActiveChart
```

```
    .PlotArea.Interior.ColorIndex = 2
    .PlotArea.Border.LineStyle = xlLineStyleNone
End With
Exit Sub

LevErrorHandler:
msg = "Rapfish goofed while leveraging!!!! error number = " & Str(Err) & ": " & Error(Err)
If MsgBox(msg, vbExclamation + vbOKCancel) = vbOK Then
   Resume Next
End If

ALSCALError:
If MsgBox("ALSCAL error #" & Str(ierr), vbExclamation + vbOKCancel) = vbOK Then
End If

End Sub

MODULE MONTE CARLO
'-------------------------------------------------------------
'This module contains the Rapfish scoring error
'analysis routine RapError1
'Subroutines used are in RapSub.
'
'The Fortran DLL g77alscal.dll must be resident in the
'Excel default directory for the code to work.
'See Main routine for description of eror returns from g77ALSCAL.
'
'History:
'  P.Kavanagh   March 2001     new code
'  Stephen Ban  April 2003     Added code for independent error estimates
'-------------------------------------------------------------

Option Explicit
Option Base 1     'indices for arrays/matrices start at 1 (not 0)

Dim msg As String

'Fortran DLL for multi-dimensional scaling
'(derived from Forrest Young code)
Declare Function g77alscal Lib "g77alscal.dll" _
Alias "g77alscal_" _
(ByRef Nrows As Long, _
ByRef Ncols As Long, _
ByRef Xinput As Single, _
ByRef MDSout As Single, _
ByRef Metric As Single, _
ByRef Sout As Single, _
ByRef Niter As Long, _
ByRef Xdebug As Single, _
ByRef Idebug As Long) As Long

Sub RapError1()
'Run Rapfish multiple times with different groups of
'(real) fisheries and check the variation in anchor
'and reference fishery locations. Use a FIXED mean
'and sigma to standardize all the fisheries.
'  first run - use the entered fisheries
'            - save the mean and sigma for each attribute
'  runs 1 to Nspins - vary the fisheries scores by
'                     adding a random noise with Gaussian
'                     distribution with 0 mean and noise_sigma
'                     standard deviation

'Number of fisheries
Dim Nfish As Integer      'real fisheries
Dim Nref As Integer       'good, bad, up, down
Dim Nanchor As Integer    'fake fisheries to assist anchoring
Dim Ncases As Integer

'Fisheries attributes and categories
Dim Ncategories As Integer  'eg. technological, etc.
Dim Nattrib As Integer  '# attributes per category

'Excel spreadsheet location for fishery 1, attribute 1
```

```
'for each of the Ncategories of attributes
Dim InitRow As Integer
Dim InitCol As Integer

'Row indices to fisheries data
Dim Ifish As Integer
Dim Igood As Integer
Dim Ibad As Integer
Dim Iup As Integer
Dim Idown As Integer
Dim Ianchor As Integer

'Column index to attributes
Dim Col1stAttrib As Integer

'Data
Dim SheetIn As Variant
Dim SheetTemp As Variant
Dim SheetOut As Variant
Dim SheetMC As Variant
Dim WF As Variant

Dim RapScore() As Single       'Rapfish scores from users
Dim Rap01() As Single          'standardized RapScore
Dim NoisyRap01() As Single     'with error added

Dim Distance() As Single
Dim MDSout() As Single
Dim MDSrotate() As Single
Dim MDSfns() As Single
Dim Sout(2) As Single
Dim Niter As Long
Dim Metric(60) As Single
Dim Xdebug(5000) As Single
Dim Idebug(30) As Long
Dim ierr As Long
Dim N As Long
Dim ColFishNames As Integer
Dim fishnames(100) As String

'Monte Carlo parameters
Dim Nspins As Integer      ' number of repeated runs
Dim noise_sigma As Single
Dim noise_percent As Single  ' expressed as % of full attribute scale
Dim Emin() As Single         ' min error (approx 2.5% probability lower tail)
Dim Emax() As Single         ' max error (approx 97.5% probability upper tail)
Dim ErrorRangeLow() As Single 'User entered error parameters for each attribute - low end SB
Dim ErrorRangeHigh() As Single 'User entered error parameters for each attribute - high end SB
Dim EminRow As Integer   'row # in RapScores sheet on Emin
Dim EmaxRow As Integer   'row # in RapScores sheet on Emax
Dim IndEminRow As Integer 'SB
Dim IndEmaxRow As Integer 'SB
Dim Emin01 As Single
Dim Emax01 As Single
Dim noise_sample As Single
Dim Lmin As Single
Dim Lmax As Single

'Other
Dim i As Integer, j As Integer, k As Integer
Dim mu() As Single
Dim sigma() As Single
Dim ch As Variant
Dim case1_xstr As String
Dim Xmc() As Single
Dim Ymc() As Single
Dim row, col, ns As Integer
Dim Lcv As Integer
Dim temp As Single

'angle of the BAD to GOOD vector in degrees before rotation
'rotate each point in 2D MDS output by -theta
'to make the BAD to GOOD line horizontal
Dim theta As Single
```

```
'On Error GoTo 0    ' Turn off error trapping.
On Error GoTo RouletteErrorHandler

'Default values of parameters
Ifish = 1
Igood = 27
Ibad = 28
Iup = 29
Idown = 30
Ianchor = 31

Nfish = 26
Nref = 4
Nanchor = 16
Nattrib = 10

Nspins = 10

'-----------------------------------------------------
'Run Rapfish first using nominal scores (no error).
'-----------------------------------------------------

'Load current parameters from RapfishForm into local variables.
Nfish = RapfishForm.Numfish.Value
Nref = RapfishForm.Numref.Value
Nanchor = RapfishForm.Numanchor.Value
Nattrib = RapfishForm.Numattrib.Value

Ifish = RapfishForm.Row1stFISH.Value
Igood = RapfishForm.RowGOOD.Value
Ibad = RapfishForm.RowBAD.Value
Iup = RapfishForm.RowUP.Value
Idown = RapfishForm.RowDOWN.Value
Ianchor = RapfishForm.Row1stAnchor.Value

'Convert Excel column letters to column numbers
Col1stAttrib = RapSub.ExcelColNum(RapfishForm.Col1stATTRIBUTE.Value)
ColFishNames = RapSub.ExcelColNum(RapfishForm.ColFishNames.Value)

'Redimension arrays for selected data set
Ncases = Nfish + Nref + Nanchor

ReDim RapScore(Ncases, Nattrib)
ReDim NoisyRap01(Ncases, Nattrib)
ReDim Rap01(Ncases, Nattrib)
ReDim Distance(Ncases, Ncases)
ReDim MDSout(Ncases, 2)
ReDim MDSrotate(Ncases, 2)
ReDim MDSfns(Ncases, 2)
ReDim mu(Nattrib)
ReDim sigma(Nattrib)
'ReDim sigma(Ncases, Nattrib) SB

'Name the spreadsheets used
Set SheetIn = Worksheets("RapScores")
Set SheetTemp = Worksheets("Distances")
Set SheetOut = Worksheets("RapAnalysis")
Set SheetMC = Worksheets("MonteCarlo")

'Clear results sheets
SheetTemp.Cells.ClearContents
SheetOut.Cells.ClearContents
SheetMC.Cells.ClearContents
SheetMC.ChartObjects.Delete

'Summarize parameters on the RapAnalysis sheet
Call RapSub.ParamSummary(Ncases + 15)

'Populate matrix RapScores from RapScores spreadsheet
For i = 1 To Nfish
   For j = 1 To Nattrib
      RapScore(i, j) = SheetIn.Cells(i + Ifish - 1, j + Col1stAttrib - 1)
   Next j
```

```
Next i

For j = 1 To Nattrib
   RapScore(i, j) = SheetIn.Cells(Igood, j + Col1stAttrib - 1)
   RapScore(i + 1, j) = SheetIn.Cells(Ibad, j + Col1stAttrib - 1)
   RapScore(i + 2, j) = SheetIn.Cells(Iup, j + Col1stAttrib - 1)
   RapScore(i + 3, j) = SheetIn.Cells(Idown, j + Col1stAttrib - 1)
Next j

k = i + 4
For i = k To Ncases
   For j = 1 To Nattrib
      RapScore(i, j) = SheetIn.Cells(i + Ianchor - k, j + Col1stAttrib - 1)
   Next j
Next i

'Get fisheries names
For i = 1 To Nfish
   fishnames(i) = SheetIn.Cells(Ifish + i - 1, ColFishNames)
Next i

'Update fisheries row indices to be indices in RapScore matrix
Ifish = 1
Igood = Nfish + 1
Ibad = Nfish + 2
Iup = Nfish + 3
Idown = Nfish + 4
Ianchor = Idown + 1

'Standardize each column of attribute scores x
'according to formula (x-mu)/sigma
'Parameter mu and sigma are either calculated as:
' 1)mean and stand deviation statistics of each attribute
'   column over all fisheries (use routine CalcStats)
' OR
' 2)"fixed calibration" method
'     mu = (GOOD + BAD)/2 for each attribute
'     sigma = |GOOD - BAD| for each attribute
'see RapfishForm check box
'Method 2) may give better absolute registration consistency
'for multiple Rapfish analyses with different sets of fisheries
'since the standardization scaling is the same for all.

If RapfishForm.OptionStandardize1.Value Then
   'use attribute stats
   Call RapSub.CalcStats(Ncases, Nattrib, RapScore(), mu(), sigma())
Else
   'use fixed range of BAD to GOOD to calibrate
   For i = 1 To Nattrib
       mu(i) = (RapScore(Igood, i) + RapScore(Ibad, i)) / 2
       sigma(i) = Abs(RapScore(Igood, i) - RapScore(Ibad, i))
   Next i
End If
Call RapSub.Standardize(Ncases, Nattrib, RapScore(), _
                 mu(), sigma(), Rap01())

'Calculate Euclidean Squared distance between each
'pair of fisheries.
Call RapSub.Proximities(Rap01(), Ncases, Nattrib, Distance())

'Store Distance on temporary sheet
For i = 1 To Ncases
   For j = 1 To Ncases
      SheetTemp.Cells(i, j) = Distance(i, j)
   Next j
Next i

'Do Multi-dimensional scaling to reduce to 2 dimensions
'Send array Distance to ALSCAL Fortran DLL
N = CLng(Ncases)
ierr = g77alscal(N, N, Distance(1, 1), MDSout(1, 1), Metric(1), _
                 Sout(1), Niter, Xdebug(1), Idebug(1))

If ierr = 1 Or ierr = 200 Or ierr > 900 Then
```

```
   GoTo ALSCALError
End If

'Rotate the 2D result so BAD to GOOD is horizontal left to right
Call RapSub.Rotate(MDSout(), MDSrotate(), Ibad, Igood, Ncases, theta)

'If UP fishery y coordinate < DOWN y coord then negate the
'y coordinate for all rotated 2D fisheries
'Scale so BAD is 0%, GOOD is 100%, UP is 50, and DOWN is -50
Call RapSub.FlipNScale(MDSrotate(), MDSfns(), Ibad, Igood, Iup, Idown, Ncases)

'---------------------------------------------------------
'Show nominal results in output sheet with fisheries names
'---------------------------------------------------------
For i = 1 To Nfish
   SheetOut.Cells(i + 1, ColFishNames) = fishnames(i)
Next i
SheetOut.Cells(Igood + 1, ColFishNames) = "GOOD"
SheetOut.Cells(Ibad + 1, ColFishNames) = "BAD"
SheetOut.Cells(Iup + 1, ColFishNames) = "UP"
SheetOut.Cells(Idown + 1, ColFishNames) = "DOWN"
SheetOut.Cells(Ianchor + 1, ColFishNames) = "ANCHORS:"

'Column headings
SheetOut.Cells(1, ColFishNames + 1) = "2D MDS Results"
SheetOut.Cells(1, ColFishNames + 4) = "Rotated"
SheetOut.Cells(1, ColFishNames + 7) = "& Flipped & Scaled"

'MDS results
For i = 1 To Ncases
   For j = 1 To 2
      SheetOut.Cells(i + 1, j + ColFishNames) = MDSout(i, j)
   Next j
Next i

'Rotated results
For i = 1 To Ncases
   For j = 1 To 2
      SheetOut.Cells(i + 1, j + ColFishNames + 3) = MDSrotate(i, j)
   Next j
Next i

'Rotated, flipped, scaled results
For i = 1 To Ncases
   For j = 1 To 2
      SheetOut.Cells(i + 1, j + ColFishNames + 6) = MDSfns(i, j)
   Next j
Next i

i = 1
j = Ifish + Ncases + 5
SheetOut.Cells(j, i) = "Stress = "
SheetOut.Cells(j, i + 1) = Sout(1)
SheetOut.Cells(j + 1, i) = "Squared Correlation (RSQ) = "
SheetOut.Cells(j + 1, i + 1) = Sout(2)
SheetOut.Cells(j + 2, i) = "Number of iterations = "
SheetOut.Cells(j + 2, i + 1) = Niter
SheetOut.Cells(j + 3, i) = "Memory needed (words) = "
SheetOut.Cells(j + 3, i + 1) = Idebug(9)
SheetOut.Cells(j + 4, i) = "Return value (error if > 0) "
SheetOut.Cells(j + 4, i + 1) = ierr
SheetOut.Cells(j + 5, i) = "Rotation angle (degrees) = "
SheetOut.Cells(j + 5, i + 1) = theta

i = 4
SheetOut.Cells(j, i) = "Iteration"
SheetOut.Cells(j, i + 1) = "Stress"
SheetOut.Cells(j, i + 2) = "Delta "
For k = 1 To Niter
SheetOut.Cells(j + k, i) = k
SheetOut.Cells(j + k, i + 1) = Metric(2 * k - 1)
SheetOut.Cells(j + k, i + 2) = Metric(2 * k)
Next k
```

```
'---------------------------------------------------------------------
'NOW that we're finished with the data entry, setup, and original data run
'SPIN the roulette table Nspins times by added Gaussian noise with 0 mean
'and standard deviation noise_sigma to the standardized REAL fisheries
'(keeping the reference and anchor fisheries pure).
'Plot the anchor and reference points on same plot for all Nspins to
'examine the scatter.
'---------------------------------------------------------------------

'Get Monte Carlo parameters from the user form
Nspins = RapfishForm.NspinsMC.Value
noise_percent = RapfishForm.NoisePercent.Value
EminRow = RapfishForm.RowNumEmin.Value
EmaxRow = RapfishForm.RowNumEmax.Value
IndEminRow = RapfishForm.RowNumIndEmin 'Added by Stephen B.
IndEmaxRow = RapfishForm.RowNumIndEmax 'Added by Stephen B.
ReDim ErrorRangeLow(Ncases, Nattrib) 'Added by Stephen B.
ReDim ErrorRangeHigh(Ncases, Nattrib) 'Added by Stephen B.
ReDim Xmc(Nspins)
ReDim Ymc(Nspins)
ReDim Emin(Nattrib)
ReDim Emax(Nattrib)
For i = 1 To Nfish
    For j = 1 To Nattrib
        Emax(j) = SheetIn.Cells(EmaxRow, j + Col1stAttrib - 1)
        Emin(j) = SheetIn.Cells(EminRow, j + Col1stAttrib - 1)
    Next j
Next i
For i = 1 To Nfish
    For j = 1 To Nattrib
        ErrorRangeLow(i, j) = SheetIn.Cells(IndEminRow + i - 1, j + Col1stAttrib - 1)
        ErrorRangeHigh(i, j) = SheetIn.Cells(IndEmaxRow + i - 1, j + Col1stAttrib - 1)
    Next j
Next i

'Loop adding independent random noise sample to real fishery attribute score each spin
For k = 1 To Nspins

    'Generate random scoring noise samples and add to the
    'standardized real fisheries scores. One of two methods
    'for random number generation is used, depending on
    'user choice.
    'For both cases, the resulting score is truncated so it
    'always lies between the BAD and GOOD limits (scores
    'beyond GOOD set to GOOD, scores beyond BAD set to BAD).
    'This changes the distribution conditional on the score values
    'so that any part of the error distribtion tail that lies
    'beyond the BAD-GOOD limits becomes an impulse probability
    'at the limit.

    If RapfishForm.OptionMC1.Value Then
       'DEFAULT scoring noise distribution:
       'Generate Normally distributed scoring error that is  with
       '0 mean and standard deviation sigma selected so that the
       '95% confidence interval is noise_percent of the full attribute
       'range, where X is selected by user. By default X=20%.
       'From Gaussian tables, 95% 2-sided confidence interval has
       'a width of 3.92*sigma, so we can calculate sigma from:
       '    3.92*sigma = (noise_percent/100) * |GOOD-BAD attribute values|
       'Also write out the equivalent Emin = Emax in the original
       'attribute units to Rapscores spreadsheet at EminRow + 5.
       '   Emin = Emax = (3.92/2 * noise_sigma)*sigma + mu
       'where mu and sigma are the values used to standardize the
       'original scores.
       SheetIn.Cells(EmaxRow + 4, 1) = "DEFAULT 95% probability scoring error limits"
       SheetIn.Cells(EmaxRow + 5, 1) = "set at " & Str(noise_percent) & "% of full attribute
scale"
       SheetIn.Cells(EmaxRow + 6, 1) = "Error limit above or below score"
       SheetIn.Cells(EmaxRow + 7, 1) = "(assumes 0 mean Normal error distribution)"
       For j = 1 To Nattrib
          temp = (noise_percent / 100) * Abs(Rap01(Igood, j) - Rap01(Ibad, j))
          noise_sigma = temp / 3.92
          For i = 1 To Nfish
             noise_sample = RapSub.GRV(0, noise_sigma)
```

```
            NoisyRap01(i, j) = Rap01(i, j) + noise_sample
         Next i
         SheetIn.Cells(EmaxRow + 6, j + Col1stAttrib - 1) = temp / 2
      Next j

   ElseIf RapfishForm.OptionMC2.Value Then
      'ALTERNATIVE scoring noise distribution:
      'Read Emin and Emax from the RapScores spreadsheet
      'at rows EminRow and EmaxRow. Generate error random
      'sample from an assymetric triangle PDF with peak
      'probability at 0, minimum value -Emin, and maximum
      'value Emax.
      For j = 1 To Nattrib
         'scale the scoring error limits same as scores
         Emax01 = Emax(j) / sigma(j)
         Emin01 = Emin(j) / sigma(j)
         For i = 1 To Nfish
            noise_sample = RapSub.TriangleRV(Emin01, Emax01)
            NoisyRap01(i, j) = Rap01(i, j) + noise_sample
         Next i
      Next j
   ElseIf RapfishForm.OptionMC3.Value Then
      'ALTERNATIVE scoring noise distribution:
      'Read ErrorMins and Maxs from the RapScores spreadsheet
      'at rows EminCol and EmaxCol. Generate error random
      'sample from an assymetric triangle PDF with peak
      'probability at 0, minimum value -Emin, and maximum
      'value Emax.
      For j = 1 To Nattrib
         For i = 1 To Nfish
            'scale the scoring error limits same as scores
            Emax01 = ErrorRangeHigh(i, j) / sigma(j)
            Emin01 = ErrorRangeLow(i, j) / sigma(j)
            noise_sample = RapSub.TriangleRV(Emin01, Emax01)
            NoisyRap01(i, j) = Rap01(i, j) + noise_sample
         Next i
      Next j
   End If

   'Don't add noise to the anchor fisheries scores
   For i = Nfish + 1 To Ncases
      For j = 1 To Nattrib
         NoisyRap01(i, j) = Rap01(i, j)
      Next j
   Next i

   'Truncate the score+noise samples NoisyRap01
   'outside BAD and GOOD limits to BAD or GOOD
   Set WF = Application.WorksheetFunction
   For j = 1 To Nattrib
       'define limits
       Lmin = Rap01(Ibad, j)
       Lmax = Rap01(Igood, j)
       If Lmin > Lmax Then
           Lmax = Rap01(Ibad, j)
           Lmin = Rap01(Igood, j)
       End If
       For i = 1 To Nfish
          NoisyRap01(i, j) = WF.Min(NoisyRap01(i, j), Lmax)
          NoisyRap01(i, j) = WF.Max(NoisyRap01(i, j), Lmin)
       Next i
   Next j

   'Calculate Euclidean Squared distance between each pair of fisheries.
   Call RapSub.Proximities(NoisyRap01(), Ncases, Nattrib, Distance())

   'Do Multi-dimensional scaling to reduce to 2 dimensions
   'Send array Distance to ALSCAL Fortran DLL
   N = CLng(Ncases)
   ierr = g77alscal(N, N, Distance(1, 1), MDSout(1, 1), Metric(1), _
                    Sout(1), Niter, Xdebug(1), Idebug(1))

   'Rotate the 2D result so BAD to GOOD is horizontal left to right
   Call RapSub.Rotate(MDSout(), MDSrotate(), Ibad, Igood, Ncases, theta)
```

```
    'If UP fishery y coordinate < DOWN y coord then negate the
    'y coordinate for all rotated 2D fisheries
    'Scale so BAD is 0%, GOOD is 100%, UP is 50, and DOWN is -50
    Call RapSub.FlipNScale(MDSrotate(), MDSfns(), Ibad, Igood, Iup, Idown, Ncases)

    'Save output to spreadsheet and plot
    For i = 1 To Ncases
        SheetMC.Cells(k, i) = MDSfns(i, 1)
        SheetMC.Cells(k + Nspins, i) = MDSfns(i, 2)
        'for TEST only, plot unscaled results to see if scaling has any effect
        'SheetMC.Cells(k, i) = MDSrotate(i, 1)
        'SheetMC.Cells(k + Nspins, i) = MDSrotate(i, 2)
    Next i

 Next k

'Look up the lower and upper critical values from Binomial distribution
'for 95% confidence interval calculation. Use Large and Small functions
'to get the index into the data.
Set WF = Application.WorksheetFunction
Lcv = WF.CritBinom(Nspins, 0.5, 0.025)
If Lcv < 1 Then Lcv = 1
If Lcv > Nspins Then Lcv = Nspins

'Calculate MEDIAN of each fishery in both X and Y dimensions
'Need to put data in an array in order to access
'functions MEDIAN, LARGE, and SMALL.
'Calculate the estimated 95% confidence region error bars
'for the median estimate. The confidence interval narrows as Nspins increases.
'Store in columns 1 to 6.
row = 3 + Nspins * 2
For k = 1 To Ncases
   For i = 1 To Nspins
       Xmc(i) = SheetMC.Cells(i, k)
       Ymc(i) = SheetMC.Cells(i + Nspins, k)
   Next i

   'Save X dimension median, min, and max estimates, and same for Y
   temp = WF.median(Xmc)
   SheetMC.Cells(row + k, 1) = temp
   SheetMC.Cells(row + k, 2) = WF.Large(Xmc, Lcv) - temp
   SheetMC.Cells(row + k, 3) = temp - WF.Small(Xmc, Lcv)

   temp = WF.median(Ymc)
   SheetMC.Cells(row + k, 4) = temp
   SheetMC.Cells(row + k, 5) = WF.Large(Ymc, Lcv) - temp
   SheetMC.Cells(row + k, 6) = temp - WF.Small(Ymc, Lcv)
Next k

'Also select error bars showing the where 50% of the scatter
'is located around the median (the interquartile range = +-25%)
'Save the MEDIAN, +25% quartile, -25% quartile for each dimension
'(columns 8, 9, 10 for X and 11,12,13 for Y dimension)
For k = 1 To Ncases
   For i = 1 To Nspins
       Xmc(i) = SheetMC.Cells(i, k)
       Ymc(i) = SheetMC.Cells(i + Nspins, k)
   Next i

   'Save X dimension median, +25%, and -25% error bars, and same for Y
   temp = WF.median(Xmc)
   SheetMC.Cells(row + k, 8) = temp
   SheetMC.Cells(row + k, 9) = WF.Quartile(Xmc, 3) - temp   '3rd quartile (75%)
   SheetMC.Cells(row + k, 10) = temp - WF.Quartile(Xmc, 1)   '1st quartile (25%)

   temp = WF.median(Ymc)
   SheetMC.Cells(row + k, 11) = temp
   SheetMC.Cells(row + k, 12) = WF.Quartile(Ymc, 3) - temp
   SheetMC.Cells(row + k, 13) = temp - WF.Quartile(Ymc, 1)
Next k

'----------------------------------------
'Plot the MEDIAN estimate for each fishery
```

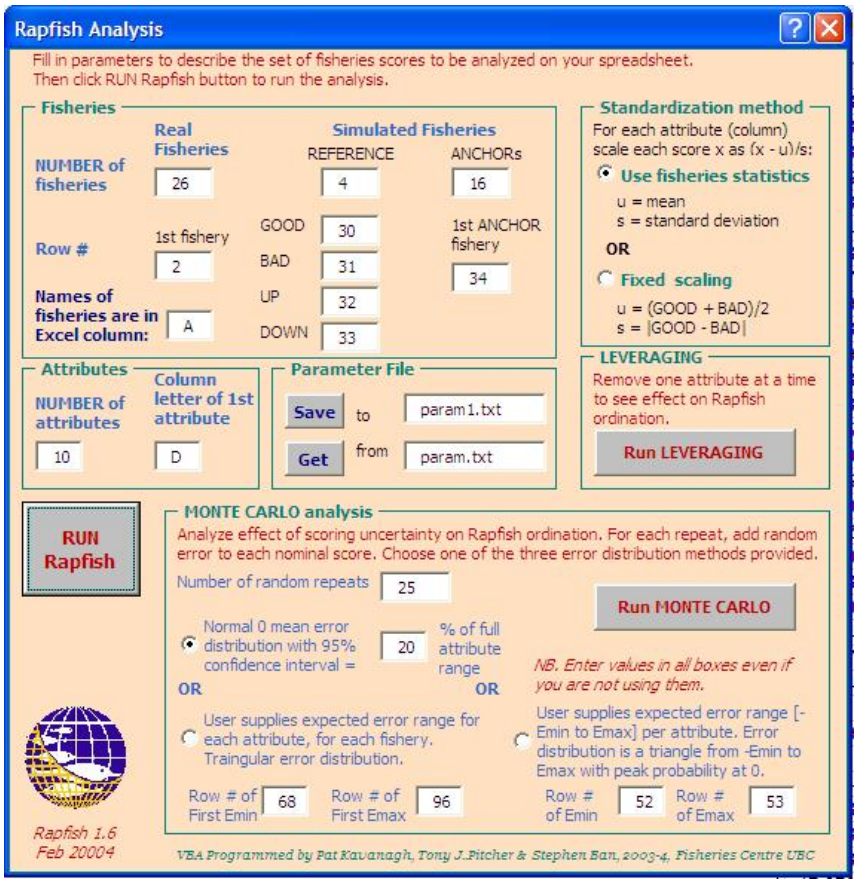
```
'with 95% confidence error bars
'----------------------------------------
Application.ScreenUpdating = False
Set ch = SheetMC.ChartObjects.Add(30, 10, 450, 450)
ch.Activate           'parameters (Top,Left,Width,Height)

ActiveChart.SetSourceData Source:=Sheets("MonteCarlo").Range("F10")
ActiveChart.Location Where:=xlLocationAsObject, Name:="MonteCarlo"
ActiveChart.ChartType = xlXYScatter
ActiveChart.HasLegend = False
ActiveChart.ChartArea.Border.LineStyle = xlLineStyleNone

'Create Median estimate series
case1_xstr = "A" & CStr(row + 1) & ":A" & CStr(row + Ncases)  'X median column
ActiveChart.SeriesCollection.NewSeries
ActiveChart.SeriesCollection(1).XValues = SheetMC.Range(case1_xstr).Offset(0, 0)
ActiveChart.SeriesCollection(1).Values = SheetMC.Range(case1_xstr).Offset(0, 3)
ActiveChart.SeriesCollection(1).MarkerStyle = xlMarkerStyleCircle
ActiveChart.SeriesCollection(1).MarkerBackgroundColorIndex = 1
ActiveChart.SeriesCollection(1).MarkerForegroundColorIndex = 1

'Create X dimension error bars on Median estimate
ActiveChart.SeriesCollection(1).errorbar _
    Direction:=xlX, _
    Include:=xlBoth, _
    Type:=xlCustom, _
    Amount:=SheetMC.Range(case1_xstr).Offset(0, 1), _
    MinusValues:=SheetMC.Range(case1_xstr).Offset(0, 2)

   'Create Y dimension error bars on Median estimate
ActiveChart.SeriesCollection(1).errorbar _
    Direction:=xlY, _
    Include:=xlBoth, _
    Type:=xlCustom, _
    Amount:=SheetMC.Range(case1_xstr).Offset(0, 4), _
    MinusValues:=SheetMC.Range(case1_xstr).Offset(0, 5)

'Adjust colours, line weight for error bars
ActiveChart.SeriesCollection(1).ErrorBars.Select
With Selection.Border
    .LineStyle = xlContinuous
    .ColorIndex = 3
    '.Weight = xlMedium
    .Weight = xlThin
End With
Selection.EndStyle = xlCap

'Add chart titles, axis labels, and gridlines
With ActiveChart
    .HasTitle = True
    .ChartTitle.Characters.Text = _
    "Rapfish Ordination Monte Carlo (Median with 95%Confidence Interval Error Bars)"
End With

With ActiveChart.Axes(xlCategory)   'y axis
    .HasTitle = True
    .AxisTitle.Characters.Text = _
        "Fisheries Status"
    .HasMajorGridlines = True
    .MajorGridlines.Border.Color = RGB(169, 169, 169)
    .MajorGridlines.Border.LineStyle = xlContinuous
End With

With ActiveChart.Axes(xlValue)   'x axis
    .HasTitle = True
    .AxisTitle.Characters.Text = _
        "Other Distingishing Features"
    .HasMajorGridlines = True
    .MajorGridlines.Border.Color = RGB(169, 169, 169)
    .MajorGridlines.Border.LineStyle = xlContinuous
End With
'-----------------------------------------
'Plot the MEDIAN estimate for each fishery
'and inter-quartile error bars
```

```
'(50% of the scatter lies within these bars).
'-----------------------------------------
Application.ScreenUpdating = False
Set ch = SheetMC.ChartObjects.Add(50, 110, 450, 450)
ch.Activate           'parameters (Top,Left,Width,Height)

ActiveChart.SetSourceData Source:=Sheets("MonteCarlo").Range("F10")
ActiveChart.Location Where:=xlLocationAsObject, Name:="MonteCarlo"
ActiveChart.ChartType = xlXYScatter
ActiveChart.HasLegend = False
ActiveChart.ChartArea.Border.LineStyle = xlLineStyleNone

'Create Median estimate series (start at column 8 (H))
case1_xstr = "H" & CStr(row + 1) & ":H" & CStr(row + Ncases)  'X median column
ActiveChart.SeriesCollection.NewSeries
ActiveChart.SeriesCollection(1).XValues = SheetMC.Range(case1_xstr).Offset(0, 0)
ActiveChart.SeriesCollection(1).Values = SheetMC.Range(case1_xstr).Offset(0, 3)
ActiveChart.SeriesCollection(1).MarkerStyle = xlMarkerStyleCircle
ActiveChart.SeriesCollection(1).MarkerBackgroundColorIndex = 1
ActiveChart.SeriesCollection(1).MarkerForegroundColorIndex = 1

'Create X dimension inter-quartile error bars (50% of scatter)
ActiveChart.SeriesCollection(1).errorbar _
    Direction:=xlX, _
    Include:=xlBoth, _
    Type:=xlCustom, _
    Amount:=SheetMC.Range(case1_xstr).Offset(0, 1), _
    MinusValues:=SheetMC.Range(case1_xstr).Offset(0, 2)

   'Create Y dimension inter-quartile error bars (50% of scatter)
ActiveChart.SeriesCollection(1).errorbar _
    Direction:=xlY, _
    Include:=xlBoth, _
    Type:=xlCustom, _
    Amount:=SheetMC.Range(case1_xstr).Offset(0, 4), _
    MinusValues:=SheetMC.Range(case1_xstr).Offset(0, 5)

'Adjust colours, line weight for error bars
ActiveChart.SeriesCollection(1).ErrorBars.Select
With Selection.Border
    .LineStyle = xlContinuous
    .ColorIndex = 3
    '.Weight = xlMedium
    .Weight = xlThin
End With
Selection.EndStyle = xlCap

'Add chart titles, axis labels, and gridlines
With ActiveChart
    .HasTitle = True
    .ChartTitle.Characters.Text = _
    "Rapfish Monte Carlo Ordination (Median with Inter-quartile Error Bars - 50% of scatter)"
End With

With ActiveChart.Axes(xlCategory)   'y axis
    .HasTitle = True
    .AxisTitle.Characters.Text = _
        "Fisheries Status"
    .HasMajorGridlines = True
    .MajorGridlines.Border.Color = RGB(169, 169, 169)
    .MajorGridlines.Border.LineStyle = xlContinuous
End With

With ActiveChart.Axes(xlValue)   'x axis
    .HasTitle = True
    .AxisTitle.Characters.Text = _
        "Other Distingishing Features"
    .HasMajorGridlines = True
    .MajorGridlines.Border.Color = RGB(169, 169, 169)
    .MajorGridlines.Border.LineStyle = xlContinuous
End With
'-----------------------------------------------------
'Plot the Monte Carlo scatter plot for the fisheries
'-----------------------------------------------------
```

```
Application.ScreenUpdating = False
Set ch = SheetMC.ChartObjects.Add(10, 210, 450, 450)
ch.Activate           'parameters (Top,Left,Width,Height)

ActiveChart.SetSourceData Source:=Sheets("MonteCarlo").Range("F10")
ActiveChart.Location Where:=xlLocationAsObject, Name:="MonteCarlo"
ActiveChart.ChartType = xlXYScatter
ActiveChart.HasLegend = False
ActiveChart.ChartArea.Border.LineStyle = xlLineStyleNone
ns = 0
For k = 1 To Ncases
   ActiveChart.SeriesCollection.NewSeries
   ns = ns + 1

   'Plot each group of Monte Carlo samples of the fishery as a series
   case1_xstr = "A1:A" & CStr(Nspins)
   ActiveChart.SeriesCollection(ns).XValues = SheetMC.Range(case1_xstr).Offset(0, k - 1)
   ActiveChart.SeriesCollection(ns).Values = SheetMC.Range(case1_xstr).Offset(Nspins, k - 1)
Next k

'Add chart titles, axis labels, and gridlines
With ActiveChart
    .HasTitle = True
    .ChartTitle.Characters.Text = _
        "Rapfish Ordination - Monte Carlo Scatter Plot"
End With

With ActiveChart.Axes(xlCategory)   'y axis
    .HasTitle = True
    .AxisTitle.Characters.Text = _
        "Fisheries Status"
    .HasMajorGridlines = True
    .MajorGridlines.Border.Color = RGB(169, 169, 169)
    .MajorGridlines.Border.LineStyle = xlContinuous
End With

With ActiveChart.Axes(xlValue)   'x axis
    .HasTitle = True
    .AxisTitle.Characters.Text = _
        "Other Distingishing Features"
    .HasMajorGridlines = True
    .MajorGridlines.Border.Color = RGB(169, 169, 169)
    .MajorGridlines.Border.LineStyle = xlContinuous
End With
Exit Sub

RouletteErrorHandler:
msg = "Rapfish goofed in Monte Carlo!!!! error number = " & Str(Err) & ": " & Error(Err)
If MsgBox(msg, vbExclamation + vbOKCancel) = vbOK Then
   Resume Next
End If

ALSCALError:
If MsgBox("ALSCAL error #" & Str(ierr), vbExclamation + vbOKCancel) = vbOK Then
End If

End Sub
```

```
CODE FOR VBA FORM

'------------------------------------------------------------
'This code is for the Rapfish parameter entry and control form
'
'History:
'  P.Kavanagh   March 2001     new code
'  same gal     May 2 2001     added more button controls
'  Stephen Ban  April 2003     added option for independent error estimates
'  Tony Pitcher Feb   2004     redesigned form
'------------------------------------------------------------

Option Explicit
Option Base 1     'indices for arrays/matrices start at 1 (not 0)

Private Sub AttribLeverageButton_Click()
```

```
Call Leveraging.JackKnifeAttributes
End Sub

'This functions may be added LATER - contorls deleted for now tjp feb 04
'The user must enter their own anchor fisheries in the spreadsheet and
'specify location
Private Sub GenerateAnchors_Click()

End Sub

Private Sub Col1stATTRIBUTE_Change()

End Sub


Private Sub NoisePercent_Change()

End Sub

Private Sub OptionMC1_Click()

End Sub

Private Sub OptionMC2_Click()

End Sub

Private Sub OptionMC3_Click()

End Sub

Private Sub
RapAnalysis1Button_Click()
Call MonteCarlo.RapError1
End Sub
Private Sub Numattrib_Change()

End Sub

Private Sub
RowNumEmax_Change()

End Sub

Private Sub
RowNumEmin_Change()

End Sub

Private Sub
RowNumIndEmax_Change()

End Sub

Private Sub
RowNumIndEmin_Change()

End Sub

Private Sub RunButton_Click()
Call Main.RunRap
End Sub

Private Sub SaveParam_Click()
'Save Rapfish processing parameters to a file, for easy recall

Dim row As Integer, col1 As Integer, col2 As Integer
Dim SheetParam As Variant
Dim filename As String
Dim msg As String

On Error GoTo SaveErrorHandler

filename = Savefilename.Value
```



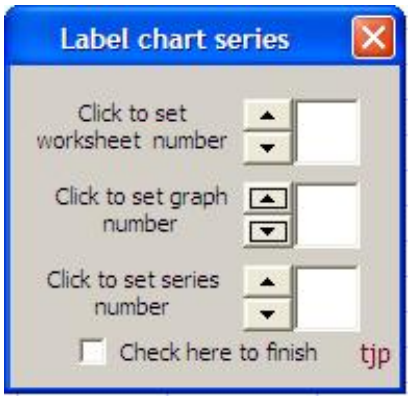
The *Rapfish* VBA interface: run analysis and options choice form (version 1.6, February 2004) which appears when the *Rapfish* toolbar button is clicked. Form is shown as populated from Ecological data in test 'Red Sea' parameter file.

```
If filename <> "" Then
   Open filename For Output As #1

   Write #1, Numfish.Value
   Write #1, Numref.Value
   Write #1, Numanchor.Value
   Write #1, Row1stFISH.Value
   Write #1, RowGOOD.Value
   Write #1, RowBAD.Value
   Write #1, RowUP.Value
   Write #1, RowDOWN.Value
   Write #1, Row1stAnchor.Value
   Write #1, ColFishNames.Value
   Write #1, Numattrib.Value
   Write #1, Col1stATTRIBUTE.Value
   Write #1, NspinsMC.Value
   Write #1, NoisePercent.Value
   Write #1, RowNumEmin.Value
   Write #1, RowNumEmax.Value
   Write #1, RowNumIndEmin.Value
   Write #1, RowNumIndEmax.Value
   If OptionStandardize1.Value Then
      Write #1, "True"
      Write #1, "False"
   Else
      Write #1, "False"
      Write #1, "True"
   End If
   If OptionMC1.Value Then Write #1, "True" Else Write #1, "False"
   'End If
   If OptionMC2.Value Then Write #1, "True" Else Write #1, "False"
   'End If
   If OptionMC3.Value Then Write #1, "True" Else Write #1, "False"
   'End If

   Close #1
Else
   msg = "Please enter a file name for saving Rapfish parameters"
   If MsgBox(msg, vbExclamation + vbOKCancel) = vbOK Then
   End If
End If

Exit Sub

SaveErrorHandler:
   'Display error and exit subroutine
   msg = "Save parameters error #" & Str(Err) & ": " & Error(Err)
   If MsgBox(msg, vbExclamation + vbOKCancel) = vbOK Then
   End If
   Close #1
End Sub


Sub GetParam_Click()
'Read parameters from specified file in
'Rapfish parameter entry sheet
'NOTE: you have to SAVE one first before you can use it
'param.txt is the DEFAULT

Dim i As Integer
Dim row As Integer, col1 As Integer, col2 As Integer
Dim SheetParam As Variant
Dim filename As String
Dim Parameter(23) As String
Dim msg As String

On Error GoTo GetErrorHandler

filename = Inputfilename.Value

If filename <> "" Then

   Open filename For Input As #1
```

```
   For i = 1 To 23
      Input #1, Parameter(i)
   Next i

   Numfish.Value = Parameter(1)
   Numref.Value = Parameter(2)
   Numanchor.Value = Parameter(3)
   Row1stFISH.Value = Parameter(4)
   RowGOOD.Value = Parameter(5)
   RowBAD.Value = Parameter(6)
   RowUP.Value = Parameter(7)
   RowDOWN.Value = Parameter(8)
   Row1stAnchor.Value = Parameter(9)
   ColFishNames.Value = Parameter(10)
   Numattrib.Value = Parameter(11)
   Col1stATTRIBUTE.Value = Parameter(12)
   NspinsMC.Value = Parameter(13)
   NoisePercent.Value = Parameter(14)
   RowNumEmin.Value = Parameter(15)
   RowNumEmax.Value = Parameter(16)
   RowNumIndEmin.Value = Parameter(17)
   RowNumIndEmax.Value = Parameter(18)
   OptionStandardize1.Value = Parameter(19)
   OptionStandardize2.Value = Parameter(20)
   OptionMC1.Value = Parameter(21)
   OptionMC2.Value = Parameter(22)
   OptionMC3.Value = Parameter(23)
   Close #1
Else
   msg = "Please enter a Rapfish parameter file name for input"
   If MsgBox(msg, vbExclamation + vbOKCancel) = vbOK Then
   End If
End If

Exit Sub

GetErrorHandler:
   'Display error and exit subroutine
   msg = "Bad file or parameter access error #" & Str(Err) & ": " & Error(Err)
   If MsgBox(msg, vbExclamation + vbOKCancel) = vbOK Then
   End If
   Close #1
End Sub

Private Sub NspinsMC_Change()

End Sub


Private Sub TextBox4_Change()

End Sub

Private Sub TextBox5_Change()

End Sub

Private Sub UserForm_Click()

End Sub
```

ANNEX4: VBA ROUTINE TO LABEL DATA POINTS IN EXCEL X-Y GRAPH

VBA CODE FOR FORM in Chartlabl.xls (see notes in spreadsheet)




```
Private Sub SpinButton1_change()
SpinButton1.Min = 0
SpinButton1.Max = 10
TextBox1.Text = SpinButton1.Value
yy = SpinButton1.Value
End Sub
Private Sub SpinButton2_Change()
SpinButton2.Min = 0
SpinButton2.Max = 10
TextBox2.Text = SpinButton2.Value
xx = SpinButton2.Value
End Sub
Private Sub SpinButton3_Change()
SpinButton3.Min = 0
SpinButton3.Max = 10
TextBox3.Text = SpinButton3.Value
zz = SpinButton3.Value
End Sub
Private Sub CheckBox1_Click()
Unload Me
End Sub
```

MODULE LABEL-MACRO

```
Public xx, yy, zz As Integer
Sub labelpoints()

Rem Visual Basic procedure to label xyplot points with labels selected in a column
Rem runs with control-L in example (user assigns keys to macro name)
Rem after selecting column of labels or blanks
Rem improved with user form tjp May 99

xx = 0: yy = 0: zz = 0
UserForm1.Show

Worksheets(xx).Activate
nlabel = Application.Selection.Count
If nlabel = 0 Then GoTo 10:
For i = 1 To nlabel
    labell = Selection.Cells(i, 1).Text
    Worksheets(xx).ChartObjects(yy).Chart _
        .SeriesCollection(zz).Points(i).HasDataLabel = True
    Worksheets(xx).ChartObjects(yy).Chart _
        .SeriesCollection(zz).Points(i).DataLabel.Text = labell
Next i
10:
End Sub
```