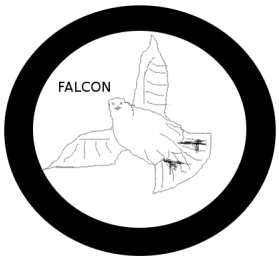
FALCON 1.0 - Instructions

Stephen J. Beckett*, Hywel T. P. Williams

October 10, 2013

College of Life and Environmental Sciences, University of Exeter, Exeter, UK



*author for correspondence: S.J.Beckett@exeter.ac.uk

Contents

1 Introduction

FALCON is software for the analysis of nestedness in bipartite networks. It enables the comparison of different nestedness measurements and null models and provides associated significance scores using an adaptive ensemble method to ensure efficient computation. Despite nestedness being a largely ecologically driven concept in the literature, particularly in respect to plant-pollinator systems nestedness can in principle be used to analyse any type of binary network. You can read more about the methodology and philosophy behind FALCON in `What_FALCON_does.pdf` . FALCON is currently coded in MATLAB, but the codes are currently being rewritten for use in R (to be released in FALCON 1.1). This document details instructions for how to download and use FALCON in MATLAB. To aid reading this document `folder locations are typed in green`, `variables are typed in red` and `file names are typed in blue`.

2 Downloading

If you have not already downloaded FALCON, you can find the main project at https://github.com/sjbeckett/FALCON . GitHub is an online repository that is widely used for sharing and collaborating on coding based projects. You can download FALCON by clicking on the 'Download ZIP' button (which is in the bottom right of the screen at time of writing).

Once the ZIP file has downloaded, move the folder to somewhere where you wish to access it from and then extract the files from their compressed format. You should probably be able to find this in the options list after right clicking the folder. This will extract the various FALCON codes and related documentation in various subfolders under `FALCON`. Within FALCON are two subfolders `Documentation`, which lists various FALCON documentation and `MATLAB`, where the MATLAB software codes used in FALCON are stored.

Inside the software subfolder are other subfolders; `MEASURES` is the folder where the codes to run different ways of measuring nestedness are stored, `NULLS` contains codes to create the different types of null models, `SOLVERS` contains the different types of ensemble generators that can be used in FALCON and `MISC` contains other 'micellaneous functions' that are used for plotting, generating statistics, sorting matrices and evaluating whether nestedness increases with an increasing nestedness score. In addition to these subfolders the software folder also includes the file `PERFORM_NESTED_TEST` which is the main file through which to use FALCON. Using the parameters supplied this file calls on the chosen solver, which then iteratively calls on the chosen null models and chosen measure to build an ensemble of null models for which statistics can be generated. There is an interactive version of FALCON which asks for user input to questions in `InteractiveMode` and also examples of how to use `PERFORM_NESTED_TEST` in the file `examplescript`.

3 MATLAB

You will require a copy of MATLAB to run the MATLAB version of FALCON. If you do not have one installed it may be possible to download a trial edition. If you cannot aquire a version of MATLAB you could try Octave (http://www.gnu.org/software/octave/), which is an open source program that is very similar to MATLAB.

1. Open MATLAB
2. Navigate to where you have stored the FALCON folder.
3. Navigate through to `FALCON/MATLAB/`

The folder `FALCON/MATLAB/` contains multiple functions in separate files which are outlined below and shown in table 1. Inside the software subfolder are other subfolders; `MEASURES` is the folder where the codes to run different ways of measuring nestedness are stored, `NULLS` contains codes to create the different types of null models, `SOLVERS` contains the different types of ensemble generators that can be used in FALCON and `MISC` contains other 'micellaneous functions' that are used for plotting, generating statistics, sorting matrices and evaluating whether nestedness increases with an increasing nestedness score. In addition to these subfolders the software folder also includes the file `PERFORM_NESTED_TEST` which is the main file through which to use FALCON. Using the parameters supplied this file calls on the chosen solver, which then iteratively calls on the chosen null models and chosen measure to build an ensemble of null models for which statistics can be generated. There are examples of how to use `PERFORM_NESTED_TEST` in the file `examplescript`. There is also an interactive version of FALCON which asks for user input to questions in `InteractiveMode`. It contains a simple code that should allow for very simple computation of nestedness metrics through user input via questions at the command line. We will cover this function in the interactive mode section 3.1. If you want something more configurable or for use within other functions then skip this section and move straight into the command line mode section 3.2.

3.1 Interactive mode

Type `InteractiveMode` into the command window and hit enter. This will cause the script in `Interactive.m` to run. This script will ask you questions about the location of the input matrix you wish to assess the nestedness of as well as questions about which nestedness measure and null model you want to use. Interactive mode is currently only set up to evaluate nestedness of a given binary matrix (0's and 1's), though quantitative matrices can be analysed in command line mode. If a non-binary matrix is supplied values greater than 0 will map to 1's, whilst values less than or equal to 0 will map to 0's. However, before interactive mode is used we need an input matrix. Type `quit` to return to the command line if you have a matrix you need to load.

File	Location	Description	Called by	Calls
InteractiveMode	MATLAB	Asks for user input through questions which is sent to PERFOM_NESTED_TEST	-	PERFORM_NESTED_TEST
PERFORM_NESTED_TEST	MATLAB	Main file for running FALCON	InteractiveMode, examplescript	sortMAT, MEASURES, SOLVERS
examplescript	MATLAB	Examples of how to use PERFORM_NESTED_TEST	-	PERFORM_NESTED_TEST
FIXEDSOLVER	SOLVERS	The fixed number ensemble solver	PERFORM_NESTED_TEST	sortMAT,MEASURES,NULLS, performEnsembleStats,plotEnsemble
ADAPTIVESOLVER	SOLVERS	The adaptive ensemble solver	PERFORM_NESTED_TEST	sortMAT,MEASURES,NULLS, performEnsembleStats,plotEnsemble
NODF	MEASURES	The NODF measure	PERFORM_NESTED_TEST,SOLVERS	-
WNODF	MEASURES	The weighted NODF measure	PERFORM_NESTED_TEST,SOLVERS	-
WNODF_REVERSE	MEASURES	The reverse weighted NODF measure	PERFORM_NESTED_TEST,SOLVERS	-
SPECTRAL_RADIUS	MEASURES	The spectral radius measure	PERFORM_NESTED_TEST,SOLVERS	-
MANHATTAN_DISTANCE	MEASURES	The Manhattan distance measure	PERFORM_NESTED_TEST,SOLVERS	-
CREATEBINNULL1	NULLS	EE binary null model	SOLVERS	-
CREATEBINNULL2	NULLS	FF binary null model	SOLVERS	-
CREATEBINNULL3	NULLS	RFRF binary null model	SOLVERS	-
CREATEQUANTNULL1	NULLS	Binary shuffled quantitative null model	SOLVERS	-
CREATEQUANTNULL2	NULLS	Conserve row totals quantitative null model	SOLVERS	shuffleVector
CREATEQUANTNULL3	NULLS	Conserve column totals quantitative null model	SOLVERS	shuffleVector
NESTED_UP_OR_DOWN	MISC	Returns whether nestedness increases with increased nestedness score or not	SOLVERS	MEASURES
performEnsembleStats	MISC	Creates statistics about the measures found in null ensemble	SOLVERS	-
plotEnsemble	MISC	Plots a histogram of the measures found in null ensemble	SOLVERS	-
shuffleVector	MISC	Changes the order of elements in a vector	CREATEQUANTNULL2, CREATEQUANTNULL3	-
sortMAT	MISC	Sorts matrix into most nested configuration	PERFORM_NESTED_TEST,SOLVERS	-

Table 1: Outline of different functions and dependancies contained in FALCON. SOLVERS is shorthand for the functions in the SOLVERS folder, similarly NULLS is short for functions in NULLS and MEASURES short for functions in MEASURES.

3.1.1 Loading a matrix

To run FALCON you need to have an input matrix that you wish to measure the nestedness of. You could write the matrix into MATLAB directly as shown in code snippet 1. Here MY_MATRIX is the variable name of the matrix being created, the square brackets indicate the enclosed is stored as an array, semi-colons indicate the start of each new row and 0's and 1's separated by spaces are the matrix elements:

Algorithm 1 Creating matrix variable MY_MATRIX

```
>> MY_MATRIX= [ 1 0 0 1;0 1 0 1; 0 1 1 1; 0 0 1 1]
```

It creates a matrix that looks like this:

$$\begin{matrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{matrix}$$

Note this is the matrix that is used to illustrate the outputs of FALCON in the rest of this document. If you already have a matrix stored in a spreadsheet it is possible to load it into MATLAB directly. To load a matrix from sheet 1 in the excel spreadsheet text.xls for example and save it to a new variable called MY_MATRIX you can run the code (similar is also shown in examplescript.m):

Algorithm 2 Loading matrix from test.xls and assigning it to MY_MATRIX

```
>> MY_MATRIX = xlsread('test',1)
```

You can search MATLAB's help documentation for 'importing data' to look at other data types that can be imported. Alternatively, if you don't have an input matrix or just want to practise, interactive mode offers you the chance to create a random matrix by typing `generate` and pressing enter as the answer to the first question as shown below.

Algorithm 3 Running InteractiveMode.m

```
>> InteractiveMode

- Enter the variable name of the matrix you wish to test.  If
a variable is not yet assigned type "exit" and create a matrix
variable this before continuing, or type "generate" to generate a
random matrix to test the code:

generate
```

3.1.2 Running interactive mode

Once you have created a matrix to test you can start interactive mode again by typing `InteractiveMode` . Next type in the name of the variable you have saved your matrix to (here we refer to this matrix as `MY_MATRIX`) and press enter.

Algorithm 4 Running `InteractiveMode.m` using a users matrix

```
>> InteractiveMode

- Enter the variable name of the matrix you wish to test.  If
a variable is not yet assigned type "exit" and create a matrix
variable this before continuing, or type "generate" to generate a
random matrix to test the code:

MY_MATRIX
```

The next steps are to answer questions about how to analyse the nestedness of the input matrix, which should be answered as directed. Below, both the questions and some example answers are shown. To move onto the next question you must press enter once you have entered your input. If you enter a command incorrectly, you will be prompted to correct your input. However, if you make a mistake you will have to restart interactive mode after answering all the questions. This example specifies that `MY_MATRIX` should be measured by NODF and tested against an equiprobable rows equiprobable columns (EE) null model using the adaptive ensemble solver:

Algorithm 5 Example of questions and user responses in InteractiveMode.m

```
- Which measure do you wish to use to measure nestedness?  If
unsure we suggest using 1, one of the more popular nestedness
measures.  Select 1 for NODF , 2 for SPECTRAL RADIUS, 3 for
MANHATTAN DISTANCE (used to calculate TAU-TEMPERATURE).
1

- Which null model do you wish to use to measure nestedness?  If
unsure we suggest using 1, where size and fill are conserved but
uniformly randomly shuffled.  Select 1 for EE , 2 for FF, 3 for
RF-RF.
1

- Are you happy to use the adaptive solver?  Adaptive solver (Y),
Fixed solver (N).
y

Now performing the calculations on MY_MATRIX to find the NODF
score; and test it against null model 1 using the ADAPTIVE
solver.  Hold on whilst your output is calculated!
```

Once these answers have been supplied interactive mode passes these variables and the input matrix to PERFORM_NESTED_TEST.m to calculate the nestedness of the input matrix and how significant the nestedness of this matrix is in comparison to those created by the null model. After these calculations have been made the user is shown the output of the null model ensemble in a figure(e.g. figure 1) and also the output ensemble statistics (see section 3.3 for more). The output of these calculations is stored in the variable ind. ind.Matrix shows the matrix that was entered by the user, while ind.index_rows and ind.index_cols show the row and column indexes for the most nested configuration of the input matrix. ind.Bin_t1 shows the statistics associated with the ensemble, where the 1 indicates that null model 1 (EE) was used. This number will change if a different null model was selected.

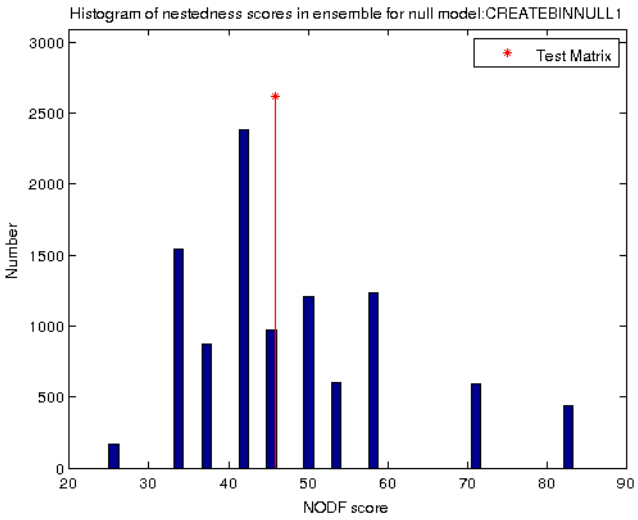


Figure 1: Output histogram of null model nestedness scores for MY_MATRIX (from 1) measured by NODF using the EE null model. Red line indicates the nestedness score of MY_MATRIX.

It is also possible to bypass the interactive mode and call PERFORM_NESTED_TEST.m directly - this is useful when you know what you are doing or if you want to embed FALCON within another script. How to use PERFORM_NESTED_TEST.m is outlined in the next section 3.2.

3.2 Command line mode

3.2.1 Default options

PERFORM_NESTED_TEST.m is the main function containing FALCONS's 'brains'. It takes user options as input and uses them to work out which measure, which null models and which solver should be used. After creating or loading the matrix (here labelled MATRIX but could take any name) you wish to assess into MATLAB you will need to define an empty variable to store output in, then you can run the function as follows:

Algorithm 6 Running PERFORM_NESTED_TEST.m

```
>> store = PERFORM_NESTED_TEST(MATRIX,binary,MEASURE,nulls,thresh,ensNum,plotON);
```

where store is the variable that the functions output will be written to. It is an object that contains the output variables. MATRIX is the matrix to be tested, binary records whether the nested tests the user wishes to assess are binary, quantitative or both:

$$binary = \begin{cases} 0 & quantitative \\ 1 & binary \\ 2 & both \end{cases}$$

MEASURE is the string name of the measure that the user wishes to assess. The string is used to call functions saved in the folder FALCON/MATLAB/MEASURES . If there is a measure that we have not coded but you wish to study it can be saved here and then used. At current the measures of nestedness that we include are:

Name	Code	binary/quantitative
NODF	NODF	Binary
WNODF	WNODF	Quantitative
WNODF-REVERSE	WNODF_REVERSE	Quantitative
Manhattan distance (used to calculate $\tau - temperature$)	MANHATTAN_DISTANCE	Binary
Spectral Radius	SPECTRAL_RADIUS	Both

Table 2: Types of measures included in FALCON

nulls is the variable that decides which null models are to be used in significance testing of the input matrix. Its choice is tied to whether the user wishes to test the matrix in a binary or quantitative sense. The different values nulls can take are shown by the CodeID in table 3. Many null models can be tested with one call to the PERFORM_NESTED_TEST.m function by using a vector e.g. nulls = [1 3] would run the EE and RFRF null models . Alternatively, all possible null models can be called by setting nulls=[]. It is important the user chooses null models that are appropriate with their choice of nestedness measure. Outputs are stored in separate objects within store for each of the different null models (see section 3.3).

Name	CodeID	
Equiprobable-Equiprobable (EE)	1	Binary
Fixed-Fixed(FF)	2	Binary
Relatively Fixed-Relatively Fixed(RFRF)	3	Binary
Shuffle Values	-1	Quantitative
Conserve Row Totals	-2	Quantitative
Conserve Column Totals	-3	Quantitative

Table 3: Types of null models included in FALCON

Each null model is encoded with a number (positive for binary nulls and negative for quantitative nulls) and new null models can be added to the folder FALCON/MATLAB/NULLS . In order to use them you will also need to change some information in the PERFORM_NESTED_TEST.m function, this is outlined in section 3.2.2.

The setting of variables thresh and ensNum tells PERFORM_NESTED_TEST.m which solver (adaptive or fixed) to run and an option to go with it. To run the adaptive solver ensNum should be set as empty (ensNum=[]), while thresh represents the threshold value for the difference in normalised temperature between the two sampling groups i.e. the measure of the test matrix divided by the expected nestedness measure calculated from the ensemble. However, if ensNum is not empty, the value it takes specifies the exact number of null models should be used in the ensemble to test for statistical significance of the nestedness found.

The final input argument plotON is used to indicate whether the user would like to view a histogram of the measures computed in the null ensemble where 1 indicates that the plot should be made and 0 indicates it should not. Plotting arguments can be found in the FALCON/MATLAB/MISC folder in plotEnsemble.m .

You should look at examplescripts.m for some examples of how to use FALCON in this way.

3.2.2 Including additional functions

You may also wish to introduce new functionality to the FALCON package, for example by including additional null models or additional measures of nestedness. This will include writing a function and saving it to the appropriate folder, where the arguments should be the same as given for the other functions there, but also in the case of new null models modifying PERFORM_NESTED_TEST.m .

This file has information about which types of measures and which null models are appropriate for each other based on whether they are binary or quantitative in nature.

If you are adding extra null models the things you need to change in PERFORM_NESTED_TEST.m are:

- The variables BnullNumber and QnullNumber in the %%REFERENCES section which represent the number of available binary and quantitative null models respectively.
- Adding code to tell the function to run your null model and store it in an appropriate object in the %%PERFORM TESTS section in a similar format to the current implementation.

3.3 Interpreting output

Once you have run one of the above nestedness scripts you will be supplied with some sort of output that will look something like this:

```
ind =
```

```
   binary:  1
   MEASURE: 'NODF'
   nulls:  1
   thresh:  1.0000e-04
   ensNum:  []
   Matrix:  [4x4 double]
   NestedConfig:  [4x4 double]
   index_rows:  [4x1 double]
   index_cols:  [4 2 3 1]
   Bin_t1:  [1x1 struct]

ind.Bin_t1 =
   NANcount:  0
   EnsembleNum:  10000
   pvalue:  0.5048
   Mean:  47.5313
   NormalisedTemperature:  0.9643
   StandardDeviation:  12.7592
   sampleZscore:  -0.1331
   Measure:  45.8333
   NestednessUpOrDown:  'Up'
   AdjustedNormalisedTemperature:  0.9643
```

What does it mean? `ind` is where the results of FALCON are stored from section 3.1 (`store` is the corresponding variable for the examples in section 3.2) this object contains the parameters used in calling `PERFORMING_NESTED_TEST.m` (`ind.binary`, `ind.MEASURE`, `ind.nulls`, `ind.thresh` and `ind.ensNum`) as described in section 3.2. In addition it contains the input matrix (`ind.Matrix`), the nested configuration of this matrix (`ind.NestedConfig`) and the orderings of rows and columns from the input matrix to achieve this (`ind.index_rows` and `ind.index_cols` respectively). See table 4 for a description of the output variables for the output from significance tests (here just `ind.Bin_t1`, where Bin stands for binary and t1 stands for test 1, the 1 indicating that null model 1 (EE) was used). It should be noted that the `NormalisedTemperature` calculated for the Manhattan distance is the $\tau - temperature$, where a $\tau - temperature$ greater than 1 indicates it is less nested than expected in comparison to the null model, whilst a $\tau - temperature$ less than 1 indicates it is more nested than expected according to the null model. This is due to more nested matrices having a lower Manhattan distance due to the matrix more likely to appear upper triangular - the shape (for a given number of elements) that will minimize Manhattan distance. This method of measuring nestedness therefore differs from the other measures considered here as the amount of nestedness decreases with increasing score. Thus the above definitions for `NormalisedTemperature` defined for $\tau - temperature$ are reversed when looking at NODF or spectral radius. However, these differences in the way these measures are defined are accounted

for when considering the p-value. pvalue always represents the probability of attaining a more nested matrix than the one under consideration.

If you are running the PERFORM_NESTED_TEST.m function rather than InteractiveMode.m it is important to know that you can run tests for multiple null models with one command, thus the above information in table 4 is computed for each of the null models you decide to compute. Each binary null model set of results is stored in the object store.Bin_t# and every quantitative null model set of results is stored in the object ind.Qua_t#, where the # refers to the null models identifying number set in PERFORM_NESTED_TEST.m (the negative numbers used for quantitative nulls are shown as positives here). In addition the input matrix is saved as ind.Matrix and the ordering of rows and columns used to compute nestedness are saved as ind.index_rows and ind.index_cols respectively. This shows rows and columns sorted into their most likely nested configuration by sorting them in terms of row and column degree respectively and removing rows and columns composed entirely of zero elements. This sorted matrix can be viewed by looking at ind.NestedConfig:

4 R

Coding up the algorithms in R is a current project, once this is done documentation will be addressed.

Name	Description
NANcount	Number of null models in ensemble that returned nestedness score with not a number. Want 0 of these!
EnsembleNum	Number of null models used in ensemble for significance testing
pvalue	The p-value found by significance testing (proportion of null models returning a higher nestedness score i.e the probability of attaining a higher nestedness score than Measure)
Mean	The mean average nestedness score returned by the null model ensemble
NormalisedTemperature	The normalised nestedness temperature (= Measure/Mean)
StandardDeviation	The standard deviation of the test matrix from the ensemble
sampleZscore	The sample Z-score of the matrix against the ensemble
Measure	The nestedness score of the matrix itself against chosen measure
NestednessUpOrDown	Displays whether increased nestedness increases with increasing measure ('Up') or decreasing measure ('Down')
AdjustedNormTemperature	If NestednessUpOrDown is 'Down' this is equal to 1/NormalisedTemperature

Table 4: Description of FALCON output