Coevolution Utilities Documentation

Daniel J. Parente
dparente@kumc.edu

Liskin Swint-Kruse
lswint-kruse@kumc.edu

February 27, 2015

Contents

List of Figures

1 Overview

The Co-evolution Utilities software suite supports both conservation-based and co-evolutionary analyses of multiple sequence alignments (MSAs). Conserved positions are amino acid positions that retain the same amino acid throughout evolution. Co-evolving positions are pairs of sites that mutate in a coordinated manner throughout evolution. To detect co-evolutionary patterns in sequence data, Co-evolution Utilities uses five, mathematically-divergent algorithms: McBASC, SCA, ELSC, OMES, and ZNMI.

The software provides support for

1. Differential conservation analysis among the subfamilies of homologous proteins
2. Automated co-evolutionary analyses (with 5 algorithms) using an "ensemble-average" treatment of the MSA
3. Comparison of co-evolution scores among homologous protein subfamilies
4. Mapping MSA column numbers to a given protein's amino acid positions
5. Generation of shuffled MSAs for determining and subtracting the "noise" from evolutionary data
6. Unthresholded analyses of co-evolutionary scores (Jaccard analysis)
7. Centrality analysis of unthresholded, weighted networks created from co-evolutionary scores

2 Prerequisites

2.1 Programming languages and runtime frameworks

This section describes the configuration of a computing environment able to run co-evolution utilities.

2.1.1 Operating system

Co-evolution utilities were designed to be run under Windows using Cygwin (see Section 2.1.4 Cygwin). The software has been tested on Windows 7, but any version of Windows should be sufficient.

Other operating systems (e.g. Linux or Mac) should be able to run the software with some minor changes (e.g. altering the assumed path to the python executable), but this has not been explicitly tested.

2.1.2 Python

Installation of Python 2.7 with the numpy, scipy and NetworkX and BioPython packages is required. The software assumes that this is installed in C:\Python27\.

For ease of installing these packages, we recommend installing the python(x,y) distribution available at: http://code.google.com/p/pythonxy/. This includes numpy, scipy and NetworkX, but lacks BioPython. BioPython can be installed by opening a command line window[1] and executing:

```
easy_install biopython
```

2.1.3 .NET framework

The .NET 4.0 (or higher) framework is required. Under Windows, this is available at no charge from Microsoft: http://www.microsoft.com/en-us/download/details.aspx?id=17851

Under Linux or Mac[2], a .NET framework is available through the Mono project: http://www.mono-project.com/Main_Page.

2.1.4 Cygwin

The Cygwin package provides access to Linux-like functionality under Windows (e.g. pipelining, GNU utilities, BASH shell scripting). Co-evolution utilities assumes that it is being run in this environment. Cygwin is free software and can be obtained from http://www.cygwin.com/.

2.2 Fodor's co-evolution package

Calculation of ELSC, OMES, McBASC and SCA scores depends on access to Anthony Fodor's Java-based implementation of various co-evolutionary algorithms. These can be obtained from this webpage: http://fodorwebsite.appspot.com/ (direct link: http://fodorwebsite.appspot.com//covariance1_1.zip).

We modified the code of McBASC to remove the filtration of 90% identical sequences. This can be implemented by modifying McBASCCovariance.java to comment out the line that reads:

a = a.getFilteredAlignment(90);

(near line 173).

The Energetics.properties file inside Fodor's package must be configured to set the home directory equal to the directory path in which the package is decompressed.[3]

This software also requires a Java runtime to be installed, which can be obtained at http://www.java.com/.

[1] Depending on your security settings, you may need to open this as an administrator by right clicking on Command Prompt in Start ¿ Programs ¿ Accessories and selecting "Run as Administrator'

[2] This will also work on Windows, if desired.

[3] Failure to do so will result in McBASC generating errors due to inability of the software to find the McLachlan substitution matrix.

2.3 ZNMI and ZNDAMI Algorithms

The ZNMI algorithm was re-implemented in C# to accommodate a specific output format and to improve speed. A modified version of the ZNMI algorithm – Z-normalized decoy adjusted mutual information (ZNDAMI) – has also been implemented. This algorithm performs subtraction analysis for ZNMI (see Section 6.2)[4].

Pre-compiled binaries of both algorithms are found in the zndami-binaries directory as ZNMI.exe and ZNDAMI.exe. Note that ZNMI.exe must be present in the same directory as ZNDAMI.exe for ZNDAMI to function (it references ZNMI internally).

2.4 Biological data

This tutorial assumes that you wish to compare the evolutionary constraints within subfamilies of a larger family of homologous proteins (e.g. the paralogs of the LacI/GalR transcription repressors). Suppose you wanted to compare the evolutionary constraints in three subfamilies. First, designate a single reference sequence for each of the subfamilies. Then, you will require four FASTA-formatted alignments: (a) one alignment for each of the subfamilies ("subfamily alignments") and (b) one alignment establishing the relationship between the three reference sequences ("reference alignment").

2.4.1 Alignment quality considerations

Detection of evolutionary constraint can be confounded by (a) insufficient number of sequences, or (b) biased sampling of available sequences. The former problem can be mitigated by including many (at least 100) homologous sequences.

Biased sampling can be avoided by removing any sequences that are 100% identical[5]. Additionally, a phylogenetic tree should be constructed (using modern methods, e.g. maximum-likelihood tree construction[6]) to verify sequences are dispersed on a tree with stellate appearance, without oversampling any individual branch.

3 Differential Conservation Analysis

Differential conservation analysis is accomplished using the ColoredMSA program. This command line software accepts two arguments, a parameter file (see below) and an output file path (e.g. output.bmp). The software highlights regions of conservation *within each subfamily* (magenta or green) and leaves non-conserved positions empty.[7] The amino acid shown in conserved positions is the most frequent ("consensus") amino acid in that row. Green cells highlight positions that are conserved in only one subfamily.

[4]The procedure of subtracting a "shuffled" co-evolution network from a ZNDAMI co-evolution network will add noise rather than remove it because of the final Z-normalization step of ZNMI. ZNDAMI performs subtraction before this normalization.

[5]Sequences which are merely highly similar, not 100% identical, should not be excluded. Even single amino acid changes are known to alter protein functioning.

[6]Avoid neighbor-joining approaches.

[7]Gaps in the reference sequences are also left empty.

3.1 The Parameter File

The parameter file specifies the alignments (subfamily and reference), the entropic definition of "conserved", an empirical substitution or physiochemical similarity matrix, and the layout geometry of the comparison figure.

To this end, the parameter file contains several lines, specifying these parameters in this order:

1. The path to the reference alignment
2. The width of a cell in the alignment, in pixels
3. The height of a cell in the alignment, in pixels
4. The number of cells of the alignment before looping to the next row
5. The maximum variability a subfamily column can have before being considered "conserved," as defined by an entropy threshold.[8]
6. A similarity matrix (e.g. the BLOSUM62 matrix)
7. Several lines, each giving a the path to a subfamily alignments and the names of the reference sequence in (a) the subfamily alignment and (b) the reference alignment. The order of subfamily declarations must match their ordering in the reference alignment.

Lines containing only whitespace (spaces, tabs, etc) in the parameter file are ignored. Lines beginning with a # are ignored as comments. All characters on a line after a # are also ignored as comments. An example configuration file is shown in Figure 1.

3.2 Set-up

The working directory should contain all of the following[9]:

- The reference alignment
- The subfamily alignments
- The parameter file (ParameterFile.txt)
- The ColoredMSA executable (ColoredMSA2.exe)
- The DJPCommonBioinfo library binary (DJPCommonBioinfo.dll)
- A similarity matrix (BLOSUM62.txt)

[8] In the example, the definition of "conservation" is taken as no more than 5% variability, given by the entropy threshold ($H_{conserved}$):

$$H_{conserved} \leq \left[-\sum p_i \ln[p_i] = -(0.95 \ln[0.95] + 0.05 \ln[0.05]) \approx 0.1985 \right]$$

[9] The last four items can be found in ColoredMSA2\ColoredMSA2\bin\Debug

```
# Comments lines are denoted by the # character
ReferenceAlignment.fasta            # Reference Alignment

# Geometric parameters
75                                  # Cell Width
100                                 # Cell Height
51                                  # Cells Per Row

# Entropy and similarity parameters
0.19851524334587                    # Maximum conserved entropy
BLOSUM62.txt                        # Similarity Matrix

# Subfamily declarations
# Must be in the same order as the rows appear in the reference alignment
# Note the fields are seperated by tabs, not spaces
FirstSubfamily.fasta NameOfReferSeqInSubfamAlign NameOfReferSeqInReferAlign
SecondSubfamily.fasta PurR_EColi PurR
ThirdSubfamily.fasta LacI_EColi LacI
# More lines, one per line in the reference alignment
```

Figure 1: Example parameter file

3.3 Execution

Open a Cygwin command prompt and change the directory (cd) to the working directory above. Execute the following command:

```
./ColoredMSA2.exe Parameter.txt output.bmp
```

This will process the files and produce a color-coded image (output.bmp in the working directory) of the reference alignment showing regions of conservation within each subfamily.

4 Co-evolution analysis

4.1 Strategy

Co-evolutionary analyses may be susceptible to alignment or sequencing errors. To quantify and mitigate the influence of these effects, the alignment can be repeatedly subsampled and re-analyzed. The coevolution utilities scripts automates this subsampling process, executes co-evolution analysis and averages these results into an final network.

Co-evolution analysis a subfamily alignment and a map file, the creation of which are described in this section.

4.2 Creation of map files

Map files serve two functions:

- Establishes a mapping between the (biologically meaningless) column number of the subfamily alignment and the biologically-meaningful residue number of the protein family.
- Declares the set of non-conserved[10] and acceptably-gapped sites. Highly-gapped and highly-conserved sites are excluded from the map file[11]. Positions not included in the map file are dropped from the downstream analyses.

One map file must be created for each subfamily.

Map files are simply tab-delimited text files with two columns per line (see Figure). The first column specifies the number to translate "from". The second column specifies the number to translate "to". These files can be created automatically (described below) or manually in a text editor or spreadsheet. An example is shown in Figure 2.

N.B.: Some software (relevantly, Fodor's co-evolutionary software package) may use a numbering convention where the first amino acid is labeled 0, which is a computer science convention. If map files are being edited manually, care should be taken to determine the appropriate numbering conventions for upstream and downstream analysis.

[10] Based on an entropy criteria, see Section 3.1.

[11] The createMap.py script allows these to be specified as command line arguments. The script to automate global mapping, Get_Global_Map.sh, by default uses a minimum non-conserved entropy of 0.1985 (see footnote to Section 4.1) and a maximum number of gaps in a column of 50%. These are the same values chosen in Parente and Swint-Kruse (2013) PLoS ONE 8(12):e84398.

```
                                MSA numbering
Tens            1          2          3          4          5
Ones  123456789 0123456789 0123456789 0123456789 0123456789 0123456789

Ref   --------- ---------- ---KPVTL-D VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV
A     --------- ---------- --MKPVTLYD VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV
B     --------- ---------- --MKPVTLYD VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV
C     --------- ---------- --MKPVTLYD VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV
D     --------- ---------- --MKPVTLYD VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV
Long  GLIINYPLD DQDAIAVEAA CTMKPVTLYD VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV
E     --------- ---------- --MKPVTLYD VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV
F     --------- ---------- --MKPVTLYD VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV
G     --------- ---------- --MKPVTLYD VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV
H     --------- ---------- --MKPVTLYD VAEYAGVSYQ TVSRVVNQAS HVSAKTREKV

                          Biological numbering
Tens                                      1          2          3
Ones  000000000 0000000000 0001234556 7890123456 7890123456 7890123456
InsC  ABCDEFGHI JKLMNOPQRS TUV     A
```

```
Map file
22      0V
23      1
24      2
25      3
26      4
27      5
28      5A
29      6
 ... etc ...
```

Figure 2: Hypothetical MSA and Map file.

This situation is typical of MSAs. A reference sequence (Ref) has been designated, which determines the biological numbering, and has been aligned with 9 other protein sequences (A-H and Long) one of which has longer length at the N-terminus (Long). In this situation, MSA analyses will yield mostly no information for positions 1-21 (MSA numbering) because it is populated by only one sequence; it is desirable to exclude them from downstream analyses. However, MSA position #22 (and #28) is well-populated except in the reference sequence. Useful information may be contained in these columns and we will want to include them in downstream analyses. The beginning of a map file translating between the MSA numbering and the biological numbering is shown. Note that because MSA positions 1-21 are not included in the map file, that these positions will be deleted when the map file is used to translate numbering. Positions that are gapped but well-populated (#22 and #28) are accommodated by using an insertion code. Important: The "MSA numbering" may be software dependent. Some software begins numbering the first column as 1, while others will label it as 0 (relevantly, Fodor's co-evolution software). It is important to know which convention is being used if writing a map file manually.

4.2.1 Set-up

First, designate a (global) reference sequence for the entire protein family (e.g. for the LacI/GalR proteins, we prefer the LacI numbering system).

Set up a working directory containing the following files:

- The reference alignment
- Each of the subfamily alignments

And the following scripts:

- Get_Global_Map.sh
- createMap.py
- composeRenumberings.py
- invertMap.py

4.2.2 Execution

Open a Cygwin command prompt and change the directory (cd) to the working directory above. Execute[12] the following command for each subfamily:

```
sh ./Get_Global_Map.sh Subfam.fasta Refer.fasta
    SFSFName SFRFName GloName > Subfam.map
```

Where:

- SFSFName is the name of the subfamily reference sequence in the *subfamily* alignment
- SFRFName is the name of the subfamily reference sequence in the *reference* alignment
- GloName is the name of the global reference in sequence in the reference alignment

For example, if the reference sequence for the PurR subfamily was *E. Coli* PurR (named PurR_EColi in the reference alignment and named simply PurR in the reference alignment) and we wanted to map this onto the *E. Coli* LacI numbering system (named LacI in the reference alignment), the command line parameters would be:

```
sh ./Get_Global_Map.sh PurR.fasta Refer.fasta PurR_EColi PurR LacI > PurR.map
```

Map files are a required input for ensemble-based co-evolution analysis (see Section 4.3). They can also be used in conjunction with the ApplyRenumbering script to convert analysis results between number systems (see Section 7.5).

[12] There is not a newline in this command; it is displayed here on two lines only as a consequence of limited space on the page.

4.3 Ensemble-based co-evolution analysis

4.3.1 Set-up

Create a new directory to hold all of the subfamily co-evolutionary analysis. In this directory, create one additional subfolder for each subfamily. Extract Fodor's co-evolution analysis tools into this directory into a folder called covariance1_1_nofilter.

Enter the subfamily directory you wish to perform co-evolution analysis on. Create a subfolder called "scripts" and copy the following files to it:

- fasta2fodor.py
- fodor2fasta.sh
- generate_subsets.sh
- mainscript.sh

Move back to the subfamily directory and create a new folder called "bin" copying the following files to it:

- AverageNetwork.exe[13]
- AverageNetwork.pdb
- ZNMI.exe[14]
- ZNDAMI.exe[15]
- DJPCommonBioinfo.dll[16]
- ApplyRenumbering.exe[17]

Next, move back to the subfamily directory. Create a folder called data_files. Copy the map file for this subfamily to this directory and rename it to `col_to_lac_map.tsv`.

In the subfamily directory, copy the subfamily's FASTA-formatted alignment to the root level. Now, convert the FASTA formatted alignment to Fodor format, by opening a command line to the subfamily directory and executing:

```
cat subfam.fasta | ./scripts/fasta2fodor.py | dos2unix > subfam.fodor
```

Finally, copy Perform_Coevol_Analysis.sh to the subfamily directory. Repeat this process for the other subfamilies. The final directory structure is shown in Figure 3. The directory is now ready to run co-evolutionary analysis.

[13] Found in AverageNetwork\AverageNetwork\bin\Debug
[14] Found in zndami-binaries
[15] Found in zndami-binaries
[16] Found in the same directory as ZNMI
[17] Found in ApplyRenumbering\ApplyRenumbering\bin\Debug

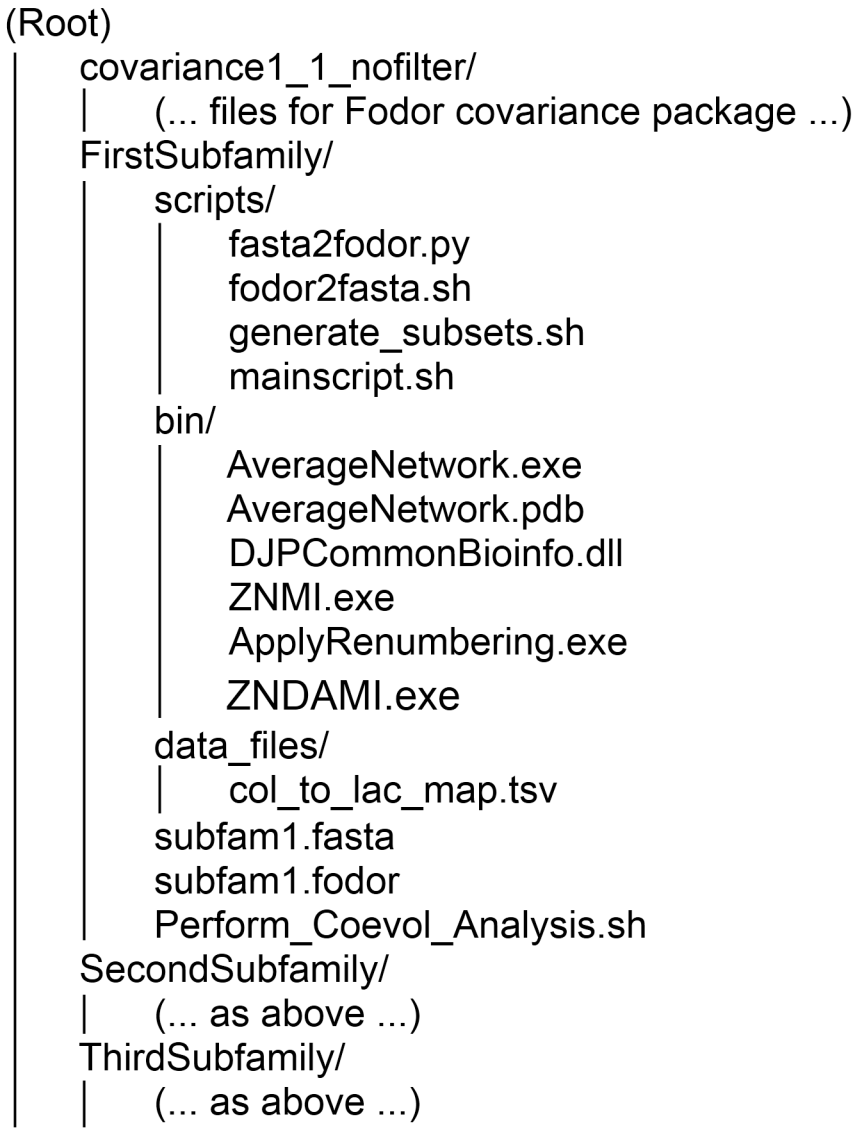

Figure 3: Directory structure needed for co-evolutionary analysis

4.3.2 Execution

We will perform a co-evolutionary analysis on an ensemble of 100 subsampled alignments, each containing 90% of the total number of sequences. Suppose our subfamily alignment contained 200 sequences; each subsampled alignment should contain 180 sequences (90% of 200).

To perform a co-evolutionary analysis, enter each subfamily directory above and execute this command:

```
sh Perform_Coevol_Analysis.sh subfam.fodor 180 100
```

You will need to adjust the "180" parameter based on the number of sequences in each subfamily alignment. The second parameter (100) specifies that 100 subsampled alignments will be analyzed and averaged.

The analysis will take several hours (largely driven by the time required to run McBASC analyses). When completed, the results will be in weighted edgelist (.wel) format in the final_result folder. This folder will contain file names of the form: algorithm.avg.map.wel (e.g. elsc.avg.map.wel or znmi.avg.map.wel).

Intermediate results are also available in the raw_results, and mapped_results directory. The raw folder contains the output of Fodor's software (N.B.: Column numbering is zero-based; the MSA column is column zero, the second, column one, etc.). The mapped directory contains data that have been translated using the map file of Section 4.2.

5 Similarity analysis

5.1 Strategy

Co-evolution analyses produce a complete[18] network with edges between positions (nodes of the network) weighted by the strength of their co-evolution score. To compare the connectivity of the networks, some threshold that discriminates "important" from "unimportant" edges must be adopted. Since co-evolution scores are continuous, with no natural discontinuities, these thresholds cannot be chosen in a data-driven way. Since any specific threshold choice would be arbitrary, we avoid this problem by calculating the similarity of two networks as a function of all possible threshold choices (i.e. from very stringent to extremely liberal).

Similarity of the networks may be determined by examining either the set of top edges or the set of top nodes. Co-evolution utilities allows for both edge- and node-based comparisons.

Quantitatively, the similarity between a set of nodes/edges is calculated by examining the Jaccard index. For two sets, A and B, the Jaccard index (J) is given by:

$$J = \frac{|A \cap B|}{|A \cup B|}$$

[18] A network in which every node is connected to every other node.

Thus highly similar sets will have a Jaccard index approaching one while dissimilar sets will have a Jaccard index approaching zero.

5.2 Extraction of top nodes and edges

Given a co-evolutionary network in weighted edgelist (.wel) format, the top nodes can be extracted by executing the following command

```
cat coevol.wel | ./orderNodes.py > coevol.topnodes
```

Similarly, extract the top edges by:

```
cat coevol.wel | ./orderEdges.py | sed 's/(tab)/_/' > coevol.topedges
```

Where coevol.wel is replaced with the weighted edgelist for a specific co-evolution analysis.

Important: Replace (tab) above with an actual tab, which can be input at the command line by pushing Ctrl+V (and then releasing this combination) followed by the tab key.

5.3 Performing a Jaccard Analysis

Jaccard analysis is supported by the Jaccard.py program. To compare the set of high scoring nodes for two subfamilies, execute the following command[19]:

```
./Jaccard.py subfamily1.topnodes subfamily2.topnodes >
                sub1_alg1_sub2_alg2.nja
```

And, similarly, for edges:

```
./Jaccard.py subfamily1.topedges subfamily2.topedges >
                sub1_alg1_sub2_alg2.eja
```

This will produce nodal jaccard analysis (.nja) and edgewise jaccard analysis (.eja) files which reports the actual agreement for each threshold value and estimates the mean and confidence interval of the random model.

5.4 Constructing a Jaccard Analysis Plots

Jaccard analysis figures can be constructed for nodal analyses by calling:

```
./plotJaccardNodes.py sub1_alg1_sub2_alg2.nja sub1_alg1_sub2_alg2_nodes.png
```

And similarly, for edges:

```
./plotJaccardEdges.py sub1_alg1_sub2_alg2.eja sub1_alg1_sub2_alg2_edges.png
```

[19]The figure construction program assumes that the filenames are meaningful: Four fields, delimited by underscores (_), ending in .nja. This is used to create a title for the plots.

6 Centrality analysis

6.1 Eigenvector centrality

The most recent version of the software suite adds support for eigenvector centrality analysis. To calculate the network centrality for a weighted edgelist (.wel; i.e. the result of Section 4.3.2), copy `evc.py` into a working directory along with the .wel files and execute:

```
./evc.py network.wel > evc-results.txt
```

This will create a tab-delimited text file with two columns: (1) the position, (2) the eigenvector centrality score associated with that position.

If you prefer to have more meaningful file extensions, you may also consider using:

```
./evc.py network.wel > evc-results.evc
```

6.2 Shuffling and subtracting a control MSA

For each MSA to be analyzed with co-evolution analyses, a useful control is to randomly shuffle the amino acids in each column. This process maintains the amino acid composition and sequence entropy of each column, but destroys the co-evolutionary information between columns. The resulting "shuffled" MSA can be analyzed with the co-evolutionary algorithms as described above to determine the "noise" that contaminates the true co-evolutionary signal. To that end,for each position, the shuffled score is subtracted from the un-shuffled score for a final co-evolutionary value.

6.2.1 Execution

For each MSA that is to be shuffled, set up a directory like the example shown in Figure 2.

To the directory level that contains the MSA.fasta file, add the files

- ShuffleMSA.exe
- DJPCommonsBioinf.dll
- subtractNetworks.py

Execute the command:

```
./ShuffleMSA.exe MSA.fasta > MSAshuffle.fasta
```

Use the file "MSAshuffle.fasta" to continue with co-evolution analyses (starting with the fasta2fodor and Perform_Coevolution commands).

Subtract the shuffled co-evolutionary scores from the unshuffuled scores, using the command:

```
./subtractNetworks.py msa.wel msa_shuffle.wel > subtracted.wel
```

Further analysis of this subtracted network can be performed (e.g. EVC calculations).

Note that shuffling and subtraction of networks are not required for ZNDAMI, which performs these steps internally.

7 Additional tools and usage examples

In addition to the applications described above, individual programs can be useful outside their role in the co-evolutionary pipeline. Selected applications are described below.

7.1 Align multiple networks

Script: alignMultiNetworks.py

Input: Several networks in weighted-edgelist (.wel) format. For example, an OMES and ELSC network from the same protein family.

Output: A tab-delimited table of co-evolution scores, arranged by position pair. The first two columns indicate the pair of positions and all subsequent columns indicating the co-evolution score in the N'th network. This facilitates comparisons between methods. If a co-evolutionary method does not output a score for a given position pair, the cell remains blank.

Usage example:

```
./alignMultiNetworks.py net1.wel net2.wel net3.wel > OutputTable.txt
```

7.2 Extract distance networks from a PDB structure

Script: extractDistanceNetwork.py

Input: Two inputs: (1) A PDB file (or pseudo-PDB file containing just ATOM/HETATM records) containing a single model or biological assembly[20]. (2) A set of chain identifiers to be used.

Output: a weighted-edgelist (.wel) text file reporting, for each pair of positions in the molecule, the shortest distance in *quaternary* structure will be reported. A shortest *tertiary* structure distance can be obtained by specifying only a single chain.

Usage example:

./extractDistanceNetwork.py protein.pdb chainIDs > shotest-distances.wel

Usage correlate: Distance networks can be used in conjunction with the align multiple networks and the create scatterplot scripts to create co-evolution score vs. structural distance figures.

7.3 Creating all-vs-all scatterplot matrices

Script: create_scatterplot.py

Input: A collated table of position scores assigned by multiple methods (e.g. alternative EVC scores).

[20]For example, NMR structures with many modelss should be edited down to a single biological assembly

For one-score-per-position types of data (such as EVC or MEW scores), such collated tables can be created by, among other things, `CompileMasterTable.exe`.

For score-per-position-pair types of data (such as RAW co-evolution scores), a collated table can be constructed with `alignMultiNetworks.py`. However, the output of this tool uses the first two columns to denote the position pair. These can be merged into a single column by removing the first tab from each line. One way to do this at the command line is:

```
cat aligned_pairwise.txt | sed 's/(TAB)/_/1' > align_all.txt
```

Note that (TAB) must be a literal tab, enterable on Cygwin by pushing Ctrl+V and then (after letting go) pushing TAB once.

Output: A scatterplot matrix (off-diagonals) comparing alternative algorithm's scores with a kernal density estimate (KDE; similar to a histogram) of the distribution of (univariate) data on the diagonal.

Usage example:

```
./create_scatterplot.py align_all.txt scatterplot-matrix.png
```

7.4 Consensus network construction

Script: compositeNetwork.py

Input: A set of weighted-edgelist (.wel) co-evolutionary networks.

Output: A weighted-edgelist (.wel) file where the edge weight for each pair of positions is average weight taken across all of the input files.

Usage example:

```
./compositeNetwork.py name1.wel name2.wel name_n.wel > average.wel
```

7.5 Apply renumbering with a map file

Script: ApplyRenumbering.exe

Input: ApplyRenumbering acts like a filter, accepting two command line arguments: A map file and the column to act on. A tab-delimited text file is also given on stdin.

Output: On stdout, position numbers in the specified column are re-numbered according to the map file.

Usage example:

```
cat data.out | ./ApplyRenumbering.exe file.map 1 > | dos2unix >
                                newdata.out
```

This remaps column 1 of data.out using file.map as a map file and saves the data as newdata.out. The inclusion of dos2unix sets Windows-style line endings (CRLF) back to Unix-style line endings (LF), and is a recommended, but not required, practice.

8 Summary of available programs

Program	Language	Description
Perform_Coevol_Analysis.sh	Bash	Root-level script that performs a co-evolutionary analysis given an input Fodor-formatted file (i.e. an alignment compatible with A. Fodor's co-evolution package) and a mapping of all non-conserved positions to a common numbering system (e.g. LacI biological numbers).
≫ GenerateSubsets.sh	Bash	Subsidiary script of Perform_Coevol_Analysis that generates an ensemble of sub-sampled MSA containing a specified fraction of the total sequences (e.g. 90% or 50%).
≫ Mainscript.sh	Bash	Subsidiary script of Perform_Coevol_Analysis that (1) carries out co-evolution analyses for each algorithm, on every alignment in the ensemble, (2) filters out highly conserved positions, (3) maps the co-evolution network to a common (e.g. LacI) numbering system, and (4) Averages across the ensemble.
Get_Global_Mapping.sh	Bash	Calculates a table mapping the MSA column number of all non-conserved positions that are ungapped in the reference sequence to a common (e.g. LacI) numbering system.
≫ CreateMap.py	Python	Subsidiary script of Get_Global_Mapping.sh. Establishes a mapping between alignment number and biological number, for a given sequence in an alignment.
≫ ComposeRenumberings.py	Python	Subsidiary script of Get_Global_Mapping.sh. Applies several mappings (e.g. those generated by CreateMap.py) in series, to produce a new (composite) mapping directly linking the first and last map in the chain (i.e. without reference to intermediate mappings).
≫ InvertMap.py	Python	Subsidiary script of Get_Global_Mapping.sh. Inverts the directionality of a mapping (e.g. Column number → LacI number becomes LacI number → Column number).
ApplyRenumbering.exe	C#	Transforms the sequence numbers of one column in a table, using a dictionary mapping (e.g. those created by Get_Global_Mapping.sh).
AverageNetwork.exe	C#	Calculates the mean and standard deviation of the co-evolution scores produced for each alignment in a subsampled ensemble.

Continued on next page

Table 1 – *Continued from previous page*

Program	Language	Description
ColoredMSA.exe	C#	Constructs a color-coded reference alignment figure, highlighting sites of conservation in each subfamily (similar to Figure 3), based on the sequence entropy of each position in the subfamily. Accepts a configuration file that specifies (1) the reference alignment, (2) the subfamily alignments, (3) the entropy-based threshold for "conserved" positions, (4) the layout geometry of the figure.
fodor2fasta.sh	Bash	Accepts a Fodor-formatted file and converts it to FASTA format.
fasta2fodor.py	Python	Accepts a FASTA-formatted file and converts it to Fodor format.
ZNMI.exe	C#	Re-implementation of the ZNMI algorithm in C#.
ZNDAMI.exe	C#	Implementation of the shuffle-and-subtract procedure for ZNMI networks.
AlignMultiNetworks.py	Python	Accepts several co-evolution networks and collates a table of scores for each edge, showing its score in each input co-evolution network.
CompileMasterTable.exe	C#	Accepts tables associating nodes with analysis parameters and collates a table of scores for each node, showing its score in each of the input tables.
OrderEdges.py	Python	Rank-orders the edges of a co-evolution network.
OrderNodes.py	Python	Rank-orders the nodes of a co-evolution network, based on the score of its strongest edge.
Jaccard.py	Python	Calculates the Jaccard index for a pair of rank-ordered positions/edges, for all possible threshold values, and compares to the "random" model. Outputs a table suitable for use with PlotJaccardEdge.py or PlotJaccardNode.py for final figure construction.
PlotJaccardEdge.py	Python	Constructs a plot of the Jaccard index for all threshold values, using the output of Jaccard.py, labeling the axis for edges.
PlotJaccardNode.py	Python	Constructs a plot of the Jaccard index for all threshold values, using the output of Jaccard.py, labeling the axis for nodes.
ExtractDistanceNetwork.py	Python	Extracts a distance network in weighted edgelist format for all pairs of ATOM records in the specified chains of a PDB.
MeasureScoreVsDistance.py	Python	Uses a distance and co-evolution network to construct a table associating physical distance with co-evolution score.
CliContact.py	Python	Detects contacts between a set of protein and ligand ATOM/HETATM records (i.e. compatible with PDB format), implementing the RESMAP contact criteria.

Continued on next page

Table 1 – *Continued from previous page*

Program	Language	Description
FilterFASTABySequence.py	Python	Filters a FASTA-formatted sequence alignment, removing all sequences not contained in an inclusion list.
ExtractNodeData.py	Python	Parses a co-evolution network and reports all scores associated with a given node.
GetPovrayEdgeDesc.py	Python	Writes a PovRay scene descriptor file to display key edges on to a dimeric backbone trace. This script is LacI/GalR specific, but easier modifiable for use with other systems.
IdentifyBasisVectors.exe	C#	Identifies a set of disparate (representative) sequences from a subfamily, using an iterative dynamic programming approach. These can be used as input seed sequences to BLAST.
ShuffleMSA.exe	C#	Scrambles the order of amino acids within each column of an MSA, thus destroying inter-column co-evolutionary signals while preserving intra-column properties (entropy, amino acid distribution).
evc.py	Python	Determines the eigenvector centrality of positions in a co-evolutionary network.
subtractNetworks.py	Python	Calculates a new co-evolutionary network by subtracting the pairwise score assigned to the same edge in one network from another (e.g. a co-evolution network and the shuffled, control, network).

Table 1: Description of available programs