Documentation for ProcessNetwork Software

Original Software Version 1.0 (2008)

Revised Version 1.5, 30 July 2015

Benjamin L. Ruddell[1]

with

Cove Sturtevant
Minseok Kang
Rong Yu

[1]Arizona State University, bruddell@asu.edu

Attribution and Licensing

The ProcessNetwork software was written between 2005 and 2008 at the University of Illinois at Urbana-Champaign, in the Department of Civil Engineering, funded 2006-2008 by a NASA fellowship #NNX06AF71H , and then continued development has occurred at Arizona State University, funded 2013-2015 by a grant from the NSF's Macrosystems Biology program BIO-1241960. Dr. Benjamin L. Ruddell is the primary author of the software, but many collaborators have contributed to its ongoing development. The work is that of the authors, and its accuracy and implications are not necessarily supported by the funding and employing organizations. Those using the software are requested to cite the software using the following three citations, and to communicate with the author regarding modifications, extensions, and applications of the software.

Current Citation for the software:

Ruddell, B.L., C. Sturtevant, M. Kang, and R. Yu (2008), ProcessNetwork Software, version 1.5, accessed at (INSERT URL OR SOURCE HERE) on (INSERT DATE HERE).

Citations for the methods employed in the software:

Ruddell, B. L.*, and P. Kumar (2009a), Ecohydrologic process networks: 1. Identification, *Water Resour. Res., 45*, W03419, doi:10.1029/2008WR007279.

Ruddell, B. L.*, and P. Kumar (2009b), Ecohydrologic process networks: 2. Analysis and characterization, *Water Resour. Res., 45*, W03420, doi:10.1029/2008WR007280.

Ruddell, B.L.*, N. Oberg, P. Kumar, and M. Garcia (2010), Using Information-Theoretic Statistics in MATLAB to Understand How Ecosystems Affect Regional Climates, MATLAB Digest Academic Edition, February 2010, www.mathworks.com/academia.

Sturtevant, C., et al. (2016), Identifying scale-emergent, non-linear, asynchronous processes of wetland methane exchange, *Journal of Geophysical Research - Biogeosciences*, 121(1), pp. 188-204. doi: 10.1002/2015JG003054.

Acknowledgements:

Dr. Praveen Kumar, for advising and co-authorship of publications at the University of Illinois

Dr. Ricky Robertson, for important histogram algorithm advice at the University of Illinois

Numerous students and colleagues who provided help with MATLAB coding

Cove Sturtevant and Dennis Baldocchi of the University of California, Berkeley for redevelopment of the MATLAB code in Version 1.5 to improve the processing speed of individual functions and add a wavelet transformation module, IAAFT surrogate data generation, and plotting functions.

Minseok Kang and Rong Yu for contributions to code versions 1.2 through 1.4

Introduction

The purpose of this software is to delineate Dynamical Process Networks based on observations of information flow between discrete vector variables using information theory, including the Transfer Entropy statistic. The software also computes Shannon Entropy, Mutual Information, Relative Entropy, and many other information theoretic statistics. The software is valid for all kinds of data, but the typical application is to Dynamical information flow and Shannon Entropy, and the typical dataset a multivariate timeseries. The software is written for the MATLAB® scientific computing environment. This basic version of the software does not contain any GUI features and there are limited plotting scripts, but more of these features may be available separately. The basic version of the software contains a small set of preprocessing functions and filters, but others may also be useful.

Files and Functions

Files you need to edit to configure a run

config_runscript_*.m	MATLAB script to define input options and run the ProcessNetwork software

Files you don't need to edit, containing the basic code

ProcessNetwork.m	master function that coordinates the data reading, writing, and processing steps; command-line executable
logwrite.m	function for logging the completed processing steps
initializeOutput.m	function that initialized the results structure
optsCheck.m	function that checks the input options and fills missing or bad options with defaults
preProcess.m	function that organizes and calls the preprocessing steps, including data trimming and transformation
trimNoData.m	function that flags an entire row as NoData if one value is NoData
removePeriodicMean.m	function that filters a periodic mean to transform data into an anomaly
waveletTranform	function that filters the data according the time scale of variations using the maximal overlap discrete wavelet transform (this function requires the WMTSA toolbox and takes computing time)
GetUniformBinEdges.m	function that find bin edges at equal intervals between Bounds-Of-Variability
GetEvenBinEdgesGlobal.m	function that finds the histogram bin edges given global Bounds-Of-Variability
classifySignal.m	function that classifies data
createSurrogates.m	function that coordinates the creation of surrogate data for statistical significance testing (takes computing time)
IAAFTsur.m	function that creates Iterated Amplitude Adjusted Fourier Transform surrogates, redistributed from the Measured of Analysis of Time Series (MATS) toolkit by Dimitris Kugiumtzis (takes computing time)
entropyFunction.m	function that calculates entropy statistics from classified data
getCountMat.m	function that computes histograms (this function takes computing time)
GetShannonBits.m	function that computes marginal and joint Shannon Entropies from histograms
ShannonBitsWrapper.m	function that sorts vectors and performs shuffled surrogate calculations
NormTheStats.m	function that normalizes output statistics
DoProduction.m	function that computes information production and consumption
AdjMatrices.m	function that computes adjacency matrices based on significance thresholds
couplingLagPlot	function that plots a coupling test statistic according to lag (line graph)
multiCouplingSynchronyPlot	function that plots the zero-lag test statistic compared to the maximum test statistic (at any lag) for multiple couplings with the same variable (bar plot)

Files Containing Reading and Examples

Readme_ProcessNetwork_v1.5.pdf	This readme file
Bondville2003InputData	Example of monthly observed data from the Bondville FLUXNET site
ChaosInputData	Example of smooth-chaos system
ToyInputData	Example of AR-type noisy input data
config_runscript_Bondville.m	Script that will run an example of 12 monthly flux tower datasets
config_runscript_Chaos.m	Script that will run an example of a smooth chaos dataset
config_runscript_Toy.m	Script that will run an example of five toy datasets

Interface, Input, and Configuration

Normally these vector variables will be timeseries variables, but they can also be spatial vector variables. The only input required for the software is a clean flat numerical file (or series of files containing the same variables/vectors in the same column order) with no header, where each numerical vector is a column, and each incremental item in the vector is a row. It is important that the time or space intervals in the vector data are equal. These input data can also be contained within native MATLAB .mat files, with the requirement that the data be contained with the "Data" variable.

The suggested structure of "config_runscript_*.m" follows, with a brief explanation of each item. This script defines the options and parameters for processing, organizing them as different fields within the "opts" structure, which is then passed into the ProcessNetwork.m function. Option fields with a default value can be missing from the opts structure (field not present), and if missing will be filled with the default value.

Fields within the options structure (opts.files, opts.varNames, …)

files…….. cell column of strings listing files to process, including path

varNames…… cell row of strings listing the variable names pertaining to the columns of the data. No LaTeX allowed. (default names are V1,V2,V3,etc.)

varSymbols……... cell row of strings listing the variable symbols to use in plotting (LaTeX okay here) (default is same as varNames)

varUnits…… cell row of strings listing the variable units (currently for metadata purposes only)

NoDataCode……. numerical value representing data gaps or data to skip (default = NaN)

trimTheData……. Preprocessing: remove entire rows of data when one or more NoDataCode values are encountered? 0 = no, 1 = yes (default)

transformation……... Preprocessing: 0 = apply no transformation (default), 1 = apply anomaly filter, 2 = apply maximal overlap discrete wavelet transform (requires no data gaps)

anomalyPeriodInData…….... for anomaly filter only: integer representing the length in time steps of the period of the data (default = 48)

anomalyMovingAveragePeriodNumber…… for anomaly filter only: the moving window used for anomaly generation, in # of periods (default = 5)

waveN……... for wavelet transform only: vector of dyadic scales to decompose (default = 1) Dyadic scales are base-2, where scale N represents 2^N time steps

waveName…… for wavelet transform only: a string indicating the mother wavelet to use (default = 'la8', options are 'haar','d4','la8','la16')

waveDorS……. for wavelet transform only: 1 (default) = pull out detail added at the scales listed in waveN (detail over multiple scales will be summed together), 2 = output smooth approximation at the scale in waveN (waveN must be single-valued in this case)

binType…… Preprocessing: 0 = don't do classification (or input data are already classified), 1 = classify using locally bounded bins (default), 2 = classify using globally bounded bins (bin range determined from all files)

binPctlRange…… for binType > 0: vector of [min max] percentile range to determine total bin range for each variable. For example, [1 99] would force the bottom and top 1% of data into the first and last bins, respectively. (default = [0 100])

nBins……number of bins to classify each signal or number of bins the data are already classified into (default = 11). This can be a single value applied to all variables, or a row vector of integers corresponding with each column in the input data

SurrogateMethod…….. Preprocessing: See description below of statistical significance testing with surrogate data. 0 = do not do statistical testing, regardless of whether input files contain surrogate data. 1 = use the surrogate data contained in the loaded files (must be named "Surrogates"). 2 = create and test new surrogates via random shuffle of input data (default); 3 = create and test new surrogates via IAAFT method (requires no data gaps)

nTests……for SurrogateMethod > 0: number of surrogates to create and/or test (default = 100)

doEntropy……. run entropy calculations? 0 = no (default), 1 = yes

lagVect……..for doEntropy = 1: vector of lags (in units of time steps) to evaluate (default = 0:10). Note: 0 will always be included, whether it is entered here or not. These are the "taus" from Ruddell and Kumar (2009 a,b). Negative lags are acceptable, but require careful interpretation.

SurrogateTestEachLag……. for doEntropy = 1: test surrogates at each lag? 0 = no, test last lag only (default), 1 = yes

oneTailZ……for doEntropy = 1: one-tail z-score for 95% significance given number of tests, 1.66 for 100, 1.68 for 50, 1.71 for 25
parallelWorkers…… parallel CPU matlab toolbox flag; if 0 or 1 no parallelization is used (default), if a positive integer MATLAB will attempt to open this number of workers using the parallel toolbox
closeParallelPool…….. close parallel pool after finishing? 0 = no, 1 = yes (default)
savePreProcessed……..save preprocessed data (including surrogates)? 0 = no (default), 1 = yes
preProcessedSuffix……... for savePreProcessed = 1: string indicating the suffix to add onto the end of each file name when saving preprocessed data (default is "_preprocessed")
saveProcessNetwork…….save R output structure (see Output section below) and processLog from ProcessNetwork computations? 0 = no, 1 = yes (default)
outFileProcessNetwork…….for saveProcessNetwork = 1: file name of saved ProcessNetwork output. Default name will include the date and time of the processing run
outDirectory……..directory to save output (both preprocessed files and results) (default is current directory)

To execute a run, configure the opts structure to list the files and processing items to perform, then using either the "config_runscript_*.m" script or the MATLAB command line, execute the ProcessNetwork function as follows:

```
>> [R,opts] = ProcessNetwork(opts)
```

The preprocessing and entropy calculations can be performed in a single processing run or in several stand-alone steps. If a single processing run is desired, choose the appropriate preprocessing steps including any data trimming, transformation, classification, and the creation of surrogate data, set the doEntropy option to 1 with associated parameters, and run the ProcessNetwork function. The preprocessed output and results will be saved to the chosen directory and files, if indicated. This way of running the code does not require storage of anything but the original data files and the results (if desired).

The processing can also be done in stages by saving the preprocessed data and then running other preprocessing and/or entropy computations at a later time (by using the saved preprocessed files instead of the originals). This allows further data splitting or other preprocessing steps to be performed before running the entropy computations. Remember to appropriately adjust the processing steps in the opts structure when running preprocessed files, as any indicated processing steps will be performed (again) on the data. Note also that this more flexible way of running the code requires greater data storage, as the preprocessed data (including surrogate data for statistical significance testing) must be saved to file.

Data preprocessing

The data trimming, transformation, and classification preprocessing steps are done in that order. The transformation option is an either/or situation. Only the anomaly filter or the wavelet filter transformation can be performed in one processing run. If both are desired, multiple processing runs must be done sequentially and the interim preprocessed data must be saved (in order to run the other filter on it).

In order to run the wavelet filter, the WMTSA Toolkit must be installed. This toolkit can be found at http://www.atmos.washington.edu/~wmtsa/ and is also redistributed with this package in the wmtsa-matlab-0.2.6.zip archive. Follow the installation instructions in the INSTALL file in the wmtsa folder of the archive. Installation of the toolkit requires a MATLAB-compatible C compiler to be installed on your computer. To determine if you have one installed, attempt to install the toolkit and MATLAB will tell you if you do not have one installed. If needed, install one by going to http://www.mathworks.com/support/compilers/R2015a/index.html or do an internet search for "MATLAB Supported and Compatible Compilers". If installing the recommended Microsoft Windows SDK 7.1 and you encounter an error, follow the fix at: http://www.mathworks.com/matlabcentral/answers/95039-why-does-the-sdk-7-1-installation-fail-with-an-installation-failed-message-on-my-windows-system.

For information on the maximal overlap discrete wavelet transform (MODWT), see:
Percival, D. and A. Walden (2000). Wavelet methods for time series analysis. Cambridge University Press. New York.

Surrogate Data

Surrogate data allows the assignment of statistical significance to the entropy statistics by running the same preprocessing and entropy computations on random data. Choose the surrogate data generation method that fits your application. The random shuffle method

simply randomly samples (without replacement) the data in each variable, destroying all auto and cross-correlations. The IAAFT method creates surrogate data that has the same autocorrelation and marginal probability density as the original data, but is otherwise random. If the processing is done in stages, it is important to create the surrogate data in the first processing run (preprocessed surrogates are saved in the variable "Surrogates") so that all processing steps done to the original data are done to the surrogate data as well. The Surrogate variable in saved preprocessing files will be of the size [nrows, ncols, nTests], where [nrows and ncols] matches the size of the original data and each level of the 3rd dimension is a replicate surrogate with nTests total replicates.

Outputs

There are quite a few output variables. These are saved out in the "R" structure and can be accessed in the workspace or output file as "R.*". The ProcessNetwork.m function also returns the opts structure with any missing or bad parameters filled with the defaults. Also output as a global variable is the processLog, a cell array of strings recording the status of the processing steps as they occur. This variable can be output to the workspace by typing "global processLog". The R output structure, opts, and processLog are saved to the indicated output file (if desired). The processLog is also saved with the preprocessed data. The name, matrix dimensionality, and meaning of each output in the R. structure are summarized below. In the case of [Vars x Vars] matrix indexing, a directionality is sometimes implied, as in the case of Transfer Entropy; in such cases, the convention is ["from" x "to"].

Definition of Terms

Lags: The number of lags or "taus"
SLags: The number of lags or "taus" evaluated in the surrogate data (either 1 or Lags)
Files: The number of files you listed in the opts structure to process.
Bins: The number of classes in your dataset
Vars: The number of columns in your dataset; variable X or Y, for example.

Output Variables

nRawData: [Files x 1]	Number of raw unprocessed data read in
nVars: [Vars x 1]	Number of variables in the file (also the number of columns in the file)
varNames: [1 x Vars]	List of variable names
varSymbols: [1 x Vars]	List of variable symbols
varUnits: [1 x Vars]	List of variable units
nBinVect: [Bins x 1]	Vector containing the number of bins
nClassified: [Files x 1]	Number of data that were successfully filtered and classified
binEdgesLocal: [Vars x Bins x Files]	Locations of histogram bin edges for local scheme
minEdgeLocal: [Vars x Files]	Locations of lower histogram bin edges for local scheme
maxEdgeLocal: [Vars x Files]	Locations of upper histogram bin edges for local scheme
minSurrEdgeLocal: [Vars x Files]	Locations of lower histogram bin edges for local scheme (min of all surrogates)
maxSurrEdgeLocal: [Vars x Files]	Locations of upper histogram bin edges for local scheme (max of all surrogates)
LocalVarAvg: [Vars x Files]	Average of each variable/column in each file (just before classification)
LocalVarCnt: [Vars x Files]	Number of non-missing data for each variable/column in each file
binEdgesGlobal: [Vars x Bins]	Locations of histogram bin edges for global scheme
minEdgeGlobal: [Vars x 1]	Locations of lower histogram bin edges for global scheme
maxEdgeGlobal: [Vars x 1]	Locations of upper histogram bin edges for global scheme
binSurrEdgesGlobal: [Vars x Bins]	Locations of surrogate data histogram bin edges for global scheme
minSurrEdgeGlobal: [Vars x 1]	Locations of surrogate data ower histogram bin edges for global scheme
maxSurrEdgeGlobal: [Vars x 1]	Locations of surrogate data upper histogram bin edges for global scheme
GlobalVarAvg: [Vars x 1]	Average of each variable/column across all files
lagVect: [Lags x 1]	List of Taus/lags
HXt: [Vars x Vars x Lags x Files]	Shannon Entropy of Variable "X" in pairwise X,Y calculation at lag t
HYw: [Vars x Vars x Lags x Files]	Shannon Entropy of Variable "Y" in pairwise X,Y calculation at lag w
HYf: [Vars x Vars x Lags x Files]	Shannon Entropy of Variable "Y" in pairwise X,Y calculation
HXtYw: [Vars x Vars x Lags x Files]	Joint Shannon Entropy of Xt and Yw
HXtYf: [Vars x Vars x Lags x Files]	Joint Shannon Entropy of Xt and Yf
HYwYf: [Vars x Vars x Lags x Files]	Joint Shannon Entropy of Yw and Yf
HXtYwYf: [Vars x Vars x Lags x Files]	Triply-joint Shannon Entropy of Xt, Yw, and Yf

SigThreshT: [Vars x Vars x SLags x Files]	Statistical Significance threshold of Transfer Entropy from X to Y
SigThreshI: [Vars x Vars x SLags x Files]	Statistical Significant threshold of Mutual Information from X to Y
meanShuffT: [Vars x Vars x SLags x Files]	Mean of Shuffled Surrogates of Transfer Entropy from X to Y
sigmaShuffT: [Vars x Vars x SLags x Files]	Stdev of Shuffled Surrogates of Transfer Entropy from X to Y
meanShuffI: [Vars x Vars x SLags x Files]	Mean of Shuffled Surrogates of Mutual Information from X to Y
sigmaShuffI: [Vars x Vars x SLags x Files]	Stdev of Shuffled Surrogates of Mutual Information from X to Y
nCounts: [Vars x Vars x Lags x Files]	Number of samples actually included in histograms in calculation
I: [Vars x Vars x Lags x Files]	Mutual Information from X to Y
T: [Vars x Vars x Lags x Files]	Transfer Entropy from X to Y
IR: [Vars x Vars x Lags x Files]	Relative Entropy or Relative Mutual Information of X to Y
TR: [Vars x Vars x Lags x Files]	Relative Transfer Entropy of X to Y
SigThreshTR: [Vars x Vars x SLags x Files]	Statistical Significance threshold of Relative Transfer Entropy from X to Y
SigThreshIR: [Vars x Vars x SLags x Files]	Statistical Significant threshold of Relative Mutual Information from X to Y
meanShuffTR: [Vars x Vars x SLags x Files]	Mean of Shuffled Surrogates of Relative Transfer Entropy from X to Y
sigmaShuffTR: [Vars x Vars x SLags x Files]	Stdev of Shuffled Surrogates of Relative Transfer Entropy from X to Y
meanShuffIR: [Vars x Vars x SLags x Files]	Mean of Shuffled Surrogates of Relative Mutual Information from X to Y
sigmaShuffIR: [Vars x Vars x SLags x Files]	Stdev of Shuffled Surrogates of Relative Mutual Information from X to Y
Tplus: [Vars x Lags x Files]	Gross Information Production of X
Tminus: [Vars x Lags x Files]	Gross Information Consumption of X
Tnet: [Vars x Lags x Files]	Net Information Production of X
TnetBinary: [Vars x Vars x Lags x Files]	1/0 flags on whether there is a (positive or negative) net flow
InormByDist: [Vars x Vars x Lags x Files]	Mutual Information from X to Y normalized between 0 and 1 limits
TnormByDist: [Vars x Vars x Lags x Files]	Transfer Entropy from X to Y normalized between 0 and 1 limits
SigThreshInormByDist: [Vars x Vars x SLags x Files]	Statistical Significance threshold of Mutual Information normalized
SigThreshTnormByDist: [Vars x Vars x SLags x Files]	Statistical Significance threshold of Transfer Entropy normalized
Ic: [Vars x Vars x Lags x Files]	Mutual Information converted to statistical confidence level
Tc: [Vars x Vars x Lags x Files]	Transfer Entropy converted to statistical confidence level
TvsIzero: [Vars x Vars x Lags x Files]	Tz statistic; T normalized by zero-lag I
SigThreshTvsIzero: [Vars x Vars x SLags x Files]	Statistical Significance threshold of Tz
IvsIzero: [Vars x Vars x Lags x Files]	Iz statistic; I normalized by zero-lag I
SigThreshIvsIzero: [Vars x Vars x SLags x Files]	Statistical Significance threshold of Iz
Abinary: [Vars x Vars x Lags x Files]	Binary Cut Transfer Entropy Adjacency Matrix
Awtd: [Vars x Vars x Lags x Files]	Weighted Transfer Entropy Adjacency Matrix
AwtdCut: [Vars x Vars x Lags x Files]	Weighted/Cut Transfer Entropy Adjacency Matrix
charLagFirstPeak: [Vars x Vars x Files]	Shortest lag where there is a statistically significant peak in T
TcharLagFirstPeak: [Vars x Vars x Files]	T at charLagFirstPeak
charLagMaxPeak: [Vars x Vars x Files]	Highest T lag where there is a statistically significant peak
TcharLagMaxPeak: [Vars x Vars x Files]	T at charLagMaxPeak
TvsIzerocharLagMaxPeak: [Vars x Vars x Files]	Shortest lag where there is a statistically significant peak in Tz
nSigLags: [Vars x Vars x Files]	Number of statistically significant lags in T of X to Y
FirstSigLag: [Vars x Vars x Files]	Shortest statistically significant lag in T
LastSigLag: [Vars x Vars x Files]	Longest statistically significant lag in T
Hm: [Files x 1]	Mean Shannon Entropy of all variables/vectors in the system
TSTm: [Lags x Files]	Mean Total System Transport of information
HXtNormByDist: [Vars x Vars x Lags x Files]	Shannon Entropy of Variable “X” in pairwise X,Y normalized between 0 and 1

Plotting

Two plotting functions are included with the ProcessNetwork Software and example usage of them is included in the config_runscript_*.m scripts:

couplingLagPlot.m	function to generate a line plot of a coupling statistic (such as I or T) for a single coupling according to lag. Multiple couplings can be plotted on the same graph by multiple calls to this function.
multiCouplingSynchronyPlot.m	Horizontal bar plot showing the zero-lag coupling statistic (black bar) as well as the maximum coupling statistic at any lag (extension to the bar colored according to lag) for couplings between

one variable ("to" variable) and several others ("from" variables).

Each of these functions accepts the R results structure and a "popts" structure of plotting parameters. Option fields within the popts structure with a default value can be missing, and if missing will be filled with the default value. The returned variable of each function is an "H" structure of plotting handles appropriately named for each of the plot components, so that any desired modification to the plot components can be done.

Both plotting functions can be executed within config_runscript_*.m script after the ProcessNetwork.m function is called, or at the command line:

>> [H] = couplingLagPlot(R,popts)
>> [H] = multiCouplingSynchronyPlot(R,popts)

Fields within the plotting options structure (popts.testStatistic, popts.sigThresh, …) for **couplingLagPlot.m**

testStatistic……(required) string indicating the name of the test statistic within the R.* output structure to plot
sigThresh……...string indicating the name of the statistical significance threshold for the test statistic within the R.* output structure to plot. If this field is missing, no statistical significance threshold will be plotted.
vars……cell array of strings (matching R.varNames) or indices within R.varNames indicating the variables in the coupling to plot. The order matters: 1st is FROM variable, 2nd is TO variable (default is [1 2])
fi……an integer indicating the file index (4th dimension of R."testStatistic") (default is 1)
laglim…… a vector of [min max] lags to show (default is entire range)
fignum……...integer indicating the figure number to plot within (default is 1)
subplot…….. a 3-element vector of integers, following the MATLAB format [nrows ncols n], indicating the subplot to plot within (default is [1 1 1])
clearplot…… 1 = clear the current plot before plotting, 0 = plot over whatever is already in the plot (default = 0)
plotProperties……Following the Name, Value linespec syntax in MATLAB, enclose desired linespec properties for the testStatistic in curly brackets (for example {'color',[0 0 0],'linewidth',2}). If missing, the MATLAB default lineSpec is used.
sigThreshPlotProperties……Following the Name, Value linespec syntax in MATLAB, enclose desired linespec properties for the sigThresh in curly brackets (for example {'color',[0 0 0],'linewidth',2}). If missing, the MATLAB default lineSpec is used.
saveFig……..1 = save the figure, 2 = don't save (default = 2)
figName…… file name for saving figure. Default includes file index, the testStatistic, and variables in the coupling
saveFormat……string indicating the format to save the figure in (without the preceding . (dot)). See MATLAB help for the saveas function for allowable formats (default is 'fig')
outDirectory……..relative if absolute directory to save the figure in. Make sure to include ending \ (slash) (default is current directory)

Fields within the plotting options structure (popts.testStatistic, popts.sigThresh, …) for **multiCouplingSynchronyPlot.m**

testStatistic……(required) string indicating the name of the test statistic within the R.* output structure to plot
sigThresh……...string indicating the name of the statistical significance threshold for the test statistic within the R.* output structure to plot. If this field is missing, no statistical significance threshold will be plotted.
ToVar……(required) The "TO" variable in the set of couplings to plot. Can be a string or the numerical index matching one of the names in R.varNames
FromVars…….. (required) The "FROM" variables in the set of couplings to plot. Can be a cell array of strings or vector of indices matching names in R.varNames
fi……an integer indicating the file index (4th dimension of R."testStatistic") (default is 1)
laglim…… a vector of [min max] lags to evaluate (default is entire range)
claglim…….. a vector of [min max] lags to form the limits of the color range used to indicate the lag of the maximum testStatistic (default is range observed in the plotted couplings)
fignum……...integer indicating the figure number to plot within (default is 1)
subplot…….. a 3-element vector of integers, following the MATLAB format [nrows ncols n], indicating the subplot to plot within (default is [1 1 1])
clearplot…… 1 = clear the current plot before plotting, 0 = plot over whatever is already in the plot (default = 0)
saveFig……..1 = save the figure, 2 = don't save (default = 2)
figName…… file name for saving figure. Default includes file index, the testStatistic, and variables in the coupling

saveFormat……string indicating the format to save the figure in (without the preceding . (dot)). See MATLAB help for the saveas function for allowable formats (default is 'fig')
outDirectory……..relative if absolute directory to save the figure in. Make sure to include ending \ (slash) (default is current directory)

Examples

If you run the included "config_runscript_Bondville.m", "config_runscript_Toy.m", and "config_runscript_Chaos.m" files (first creating the Toy and Chaos data by running the scripts in their respective folders) the following command window output will be produced, with R.* results structure for the three example datasets and two example figures saved in their respective folders. The opts structure and processLog variable saved with the results will contain a copy of the parameters used to process the data and the command window output, respectively.

```
>> config_runscript_Bondville

Checking options & parameters...
*** Beginning processing ***
Starting parallel pool (parpool) using the 'local' profile ... connected to 2 workers.
Matlab Pool open with 2 CPU cores employed.
--Processing file # 1: Bondville2003InputData\2003_1_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 2: Bondville2003InputData\2003_2_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 3: Bondville2003InputData\2003_3_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 4: Bondville2003InputData\2003_4_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 5: Bondville2003InputData\2003_5_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 6: Bondville2003InputData\2003_6_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 7: Bondville2003InputData\2003_7_USBo1_L4_30min.txt...
```

Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 8: Bondville2003InputData\2003_8_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 9: Bondville2003InputData\2003_9_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 10: Bondville2003InputData\2003_10_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 11: Bondville2003InputData\2003_11_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
--Processing file # 12: Bondville2003InputData\2003_12_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Computing local statistics
Creating and running the same operations on 100 surrogates using random shuffle method.
Computing global statistics
*** Processing files again, this time using global binning ***
--Processing file # 1: Bondville2003InputData\2003_1_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 2: Bondville2003InputData\2003_2_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 3: Bondville2003InputData\2003_3_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data

Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 4: Bondville2003InputData\2003_4_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 5: Bondville2003InputData\2003_5_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 6: Bondville2003InputData\2003_6_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 7: Bondville2003InputData\2003_7_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 8: Bondville2003InputData\2003_8_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 9: Bondville2003InputData\2003_9_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 10: Bondville2003InputData\2003_10_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data

```
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0  100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 11: Bondville2003InputData\2003_11_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0  100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
--Processing file # 12: Bondville2003InputData\2003_12_USBo1_L4_30min.txt...
Preprocessing data.
Trimming rows with missing data
Applying anomaly filter over 5 periods of 48 time steps per period.
Classifying with [11] global bins over [0  100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method (this may take a while)...
Testing surrogates at final lag only.
Parallel pool using the 'local' profile is shutting down.
Computing final entropy quantities.
Saving results.
Processing run complete.
Elapsed time is 180.384608 seconds.



>> config_runscript_Toy

Checking options & parameters...
No varNames option found. Default names will be used (V1,V2,etc.)
*** Beginning processing ***
Starting parallel pool (parpool) using the 'local' profile ... connected to 2 workers.
Matlab Pool open with 2 CPU cores employed.
--Processing file # 1: ToyInputData\data_ARNoisy.txt...
Setting variable names to (V1,V2,etc.) for rest of processing run.
Warning: # of varSymbols inconsistent with # of varNames. Setting varSymbols to varNames.
Warning: # of varUnits inconsistent with # of varNames. Clearing varUnits.
Preprocessing data.
Trimming rows with missing data
Computing local statistics
Classifying with [11] local bins over [0  100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method.
Testing surrogates at final lag only.
--Processing file # 2: ToyInputData\data_ARNoisy2.txt...
Preprocessing data.
Trimming rows with missing data
Computing local statistics
Classifying with [11] local bins over [0  100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method.
Testing surrogates at final lag only.
--Processing file # 3: ToyInputData\data_ChaosNoisy.txt...
Preprocessing data.
```

Trimming rows with missing data
Computing local statistics
Classifying with [11] local bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method.
Testing surrogates at final lag only.
--Processing file # 4: ToyInputData\data_ChaosNoisy2.txt...
Preprocessing data.
Trimming rows with missing data
Computing local statistics
Classifying with [11] local bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method.
Testing surrogates at final lag only.
--Processing file # 5: ToyInputData\data_NormNoisy.txt...
Preprocessing data.
Trimming rows with missing data
Computing local statistics
Classifying with [11] local bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method.
Testing surrogates at final lag only.
Computing global statistics
Parallel pool using the 'local' profile is shutting down.
Computing final entropy quantities.
Saving results.
Processing run complete.
Elapsed time is 522.383480 seconds.

>> config_runscript_Chaos

Checking options & parameters...
*** Beginning processing ***
Starting parallel pool (parpool) using the 'local' profile ... connected to 2 workers.
Matlab Pool open with 2 CPU cores employed.
--Processing file # 1: ChaosInputData\data_ChaosWithResolutions.mat...
Warning: # of varSymbols inconsistent with # of varNames. Setting varSymbols to varNames.
Warning: # of varUnits inconsistent with # of varNames. Clearing varUnits.
Preprocessing data.
Trimming rows with missing data
Computing local statistics
Classifying with [11] local bins over [0 100] percentile range.
Running entropy function.
Creating and running the same operations on 100 surrogates using random shuffle method.
Testing surrogates at final lag only.
Computing global statistics
Parallel pool using the 'local' profile is shutting down.
Computing final entropy quantities.
Saving results.
Processing run complete.
Elapsed time is 16.901219 seconds.

Notes, Bugs, and Fixes

23 April 2013, Benjamin L. Ruddell [RELEASE VERSION 1.0]

- GUI and plotting removed for publication

- Debug functionality disabled
- Text output is broken
- Log file formatting is broken
- Input errors are not handled

28 May 2014, Benjamin L. Ruddell, Rong yu, Minseok Kang [VERSION 1.1]

- Fixed the descriptions of variables: charLagFirstPeak, TcharLagFirstPeak, charLagMaxPeak, TcharLagMaxPeak
- Updated ProcessNetwork_v1.m to ProcessNetwork_v1.1.m. Fixed calculations for variables: R.LocalVarAvg(v,f), R.GlobalVarAvg(v), R.maxEdgeGlobal(v)

27 June 2014, Juyeol Yun, Minseok Kang, Benjamin L. Ruddell [VERSION 1.2]

- Updated ProcessNetwork.m for the newest version of Matlab: Modified the variables in OPEN PARALLEL TOOLBOX POOL, findResource -> parcluster, localScheduler.Clustersize -> localScheduler.NumWorkers

10 October 2014, Minseok Kang, Benjamin L. Ruddell [VERSION 1.3]

- Updated ShannonBitsWrapper.m: Fixed to generate shuffled surrogate tuple matrix i.e., shuffleMat

16 January 2015 Rong Yu, Benjamin L. Ruddell [RELEASE VERSION 1.4]

- Updated and validated ProcessNetwork.m, NormTheStats.m, and entropyFunction.m for adding three new variables: HXtNormByDist, IvsIzero, and SigThreshIvsIzero
- Changed parallel processing to the new Matlab parallel system (not the older pool system)

2 August 2015 Cove Sturtevant, Benjamin L. Ruddell [RELEASE VERSION 1.5]

- Redevelopment of the code resulting in a substantially different architecture of the main ProcessNetwork.m function in order to add a wavelet transformation module and IAAFT surrogate data generation, and to repeat all processing steps similarly on the surrogate data. Several of the functions were adjusted to reduce computation time. This transformation architecture is generic and can accommodate other types of transformations before information metrics are computed.
- The data range of the "TO" variable in each coupling is held constant over all tested lags. Previously, the evaluated data range of the "TO" variable decreased with each successive lag. This can lead to differences in a test statistic among lags simply due to the difference in data range evaluated, if the data range is not far larger than the scale of time lag or wavelet transformation being employed. The code now keeps the "TO" variable data range constant over all lags by using the data range pertaining to the highest lag. Therefore, the results for lags less than the maximum differ slightly from Version 1.4 of the code due to a different range of data being used. Results for the maximum information flow lag are identical to Version 1.4, which is the most important test.
- The anomaly filter function removePeriodicMean.m was adjusted to center the averaging window over each point instead of each point being aligned with the left edge of the averaging window. Also, at the edges of the data series the averaging window is successively shifted forward (front edge of series) or backward (back edge of series) so that the anomaly of the edge points is computed over a full averaging window. This change allows the total series length to be retained. Therefore, results when using the anomaly filter differ slightly from those of Version 1.4, though only slightly.
- The entropy statistics are computed for a lag of 0 in the same manner as the computations for higher lags. Therefore results at zero lag differ slightly from Version 1.4 of the code, which did not compute some statistics at zero-lag.
- The statistical significance thresholds can differ strongly between Version 1.4 and 1.5 of the code as a result of randomly shuffling the data prior to anomaly filtering and wavelet transform in Version 1.5. Statistical significance thresholds are near-identical between version 1.4 and 1.5 when using the random shuffle method if these preprocessing transformation steps are not performed. The user needs to be aware that if a transformation is employed, significance thresholds are based on a dataset that was time-shuffled before being transformed.