NetStruct

Version 1.1

User Manual

Developed by Gili Greenbaum

Contents

1. Introduction

NetStruct is program implementing a distance-based approach for inference and analysis of population structure using genetic data, set in a network theory framework. Details can be found in _____.

This is the first version of NetStruct, and thus has many issues and bugs, and only limited functionality. All these will be addressed in forthcoming versions. This version is intended to allow basic population structure analysis. This version is also not a standalone software, but is a package that must be run in Mathematica. Do not worry,

You do not need to know anything about

Mathematica in order to use NetStruct!!!

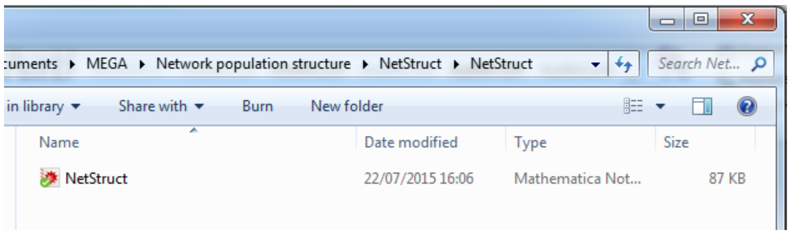
All you need is to have Mathematica (version 8 or later) installed on your computer.

NetStruct is supplied with genetic information, either as individual's genotypes or as a genetic distance matrix. NetStruct then constructs a network, after removing edges below a user defined threshold, and applies a community detection algorithm in order to detect communities (dense subgraphs in the network, i.e. groups of individuals that are strongly related within the group vs. between the groups). The results can be further analyzed for the SAD (Strength of Association) of the detected communities, Strength of Disassociation of the individuals to communities and a permutation-based significance test (for the modularity of the detected community partition).

Please be patient with this version of NetStruct. If input is not defined as NetStruct expects, it will probably crash. You can always close Mathematica, and start again. Future versions are promised to be more user-friendly and better-designed, but the hope is that this version is sufficient to conduct population structure analysis.

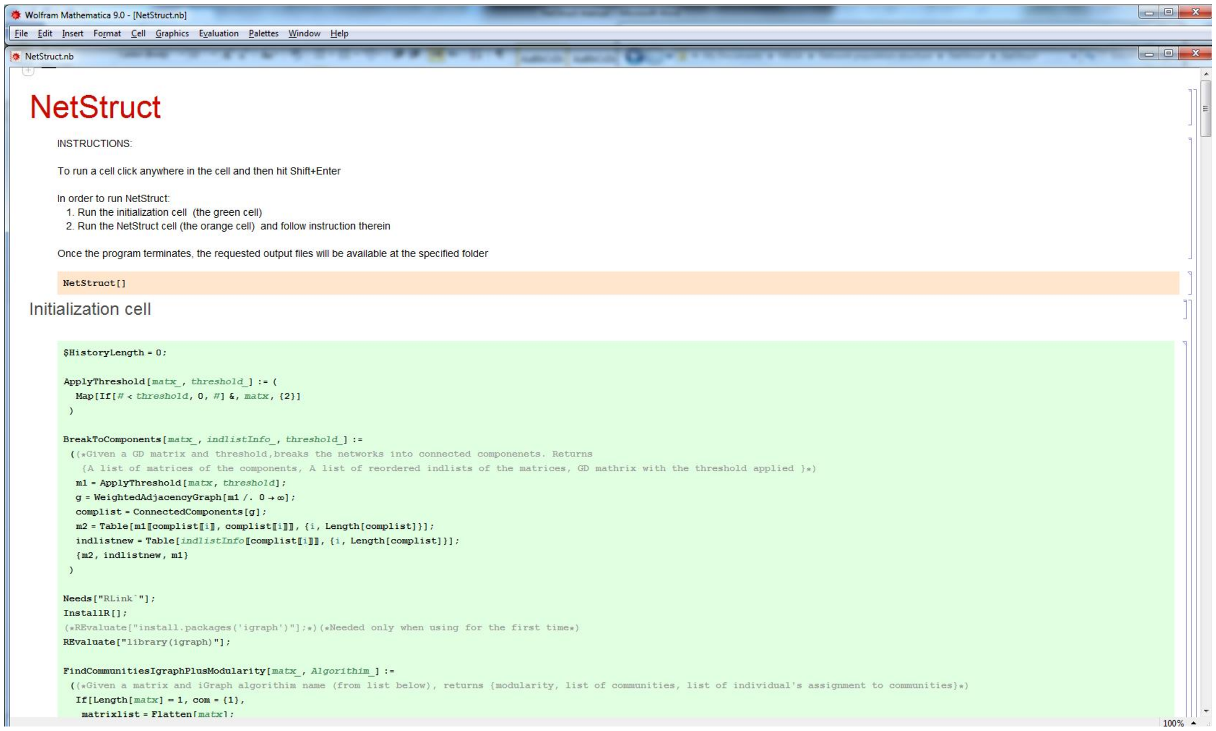
2. Getting started

First step, you need to open the Netstruct Notebook file (NetStruct.nb):


cuments ▸ MEGA ▸ Network population structure ▸ NetStruct ▸ NetStruct
Search Net...
in library
Share with
Burn
New folder

Name	Date modified	Type	Size
NetStruct	22/07/2015 16:06	Mathematica Not...	87 KB

Next Mathematica will initialize, and you will see the following notebook:


Wolfram Mathematica 9.0 - [NetStruct.nb]
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
NetStruct.nb
100%

NetStruct

INSTRUCTIONS:

To run a cell click anywhere in the cell and then hit Shift+Enter

In order to run NetStruct:
1. Run the initialization cell (the green cell)
2. Run the NetStruct cell (the orange cell) and follow instruction therein

Once the program terminates, the requested output files will be available at the specified folder

```
NetStruct[]
```

Initialization cell

```
$HistoryLength = 0;

ApplyThreshold[matx_, threshold_] := (
  Map[If[# < threshold, 0, #] &, matx, {2}]
 )

BreakToComponents[matx_, indlistInfo_, threshold_] :=
 ((*Given a GD matrix and threshold,breaks the networks into connected componenets. Returns
   {A list of matrices of the components, A list of reordered indlists of the matrices, GD mathrix with the threshold applied }*)
  m1 = ApplyThreshold[matx, threshold];
  g = WeightedAdjacencyGraph[m1 /. 0 → ∞];
  complist = ConnectedComponents[g];
  m2 = Table[m1[[complist[[i]], complist[[i]]]], {i, Length[complist]}];
  indlistnew = Table[indlistInfo[[complist[[i]]]], {i, Length[complist]}];
  {m2, indlistnew, m1}
 )

Needs["RLink`"];
InstallR[];
(*REvaluate["install.packages('igraph')"];*)(*Needed only when using for the first time*)
REvaluate["library(igraph)"];

FindCommunitiesIgraphPlusModularity[matx_, Algorithim_] :=
 ((*Given a matrix and iGraph algorithim name (from list below), returns {modularity, list of communities, list of individual's assignment to communities}*)
  If[Length[matx] = 1, com = {1},
   matrixlist = Flatten[matx];
```

In order to run NetStruct, you will need to do follow the following two steps:

1. Click anywhere in the green cell, and hit Shift+Enter (this will run the initialization cell)
2. Click anywhere in the orange cell and hit Shift+Enter (this will run NetStuct)

Once the NetStruct procedure is finished, the program will terminate and the output files will be available at the designated folder.

3. Constructing Genetic-Distance matrix from genotype data

Genotype data may be inputted directly to NetStruct in order to construct a Genetic-Distance matrix (GDmatrix). NetStruct uses a frequency-weighted similarity measure in order to calculate genetic distance between individuals. The similarity in a given locus *l* between individual *i* with alleles a and *b* (with frequencies f_a and f_b respectively) and individual *j* with alleles *c* and *d* (with frequencies f_c and f_d respectively) is defined as:

$$S_{ij,l} = \frac{1}{4}\big((1-f_a)(I_{ac}+I_{ad}) + (1-f_b)(I_{bc}+I_{bd})\big)$$

where I_{ac} is one if alleles *a* and *c* are identical and zero otherwise, and the other indicators are similarly defined. Given a sample of L loci, the genetic distance between *i* and *j* is computed as:

$$G(i,j) = \frac{1}{L}\sum_{l=1}^{L} S_{ij,l}$$

Currently NetStruct supports two formats for genotype input: Mathematica MX files and Comma-separated values (CSV) file. The output file is a Mathematica MX file containing an *n*n* GDmatrix (this file can be used later as a pre-prepared GDmatrix).

The input files should contain the sampled population of the individuals, the name of each individual and the genotypes. Note that the sampled populations are not used during the computation, and they are used only for visualization and data presentation. If there are no known sampled populations, a single sampled population should be assumed. Missing genetic data can be incorporated by setting the genotypes in the missing locus to 0. When calculating the genetic distance between a pair of individuals the loci with missing data for either individual are discarded.

In this version of NetStruct only Codominant allele option is available. Microsatellite data can be used, but no mutation model is assumed for the number of microsatellite repeats.

3.1 Mathematica MX format

The input format should be a list of length *n* (*n*=number of individuals in the sample) of the form {ind$_1$, ind$_2$,…, ind$_n$}. Each individual is described by a list of the form ind$_i$={*Sampled pop*, *Individual ID*, *Genotypes*}. Sampled pop and Individual ID are interpreted as strings and therefore do not need to be necessarily numbers. *Genotypes* is a list of length *L* (*L*=number of Loci in the sample) of the form {gen$_1$,gen$_2$,…gen$_L$}. Each genotype is a list of length 2 (diploid) the form gen$_i$={allele$_1$, allele$_2$}. Each allele is interpreted as a string.

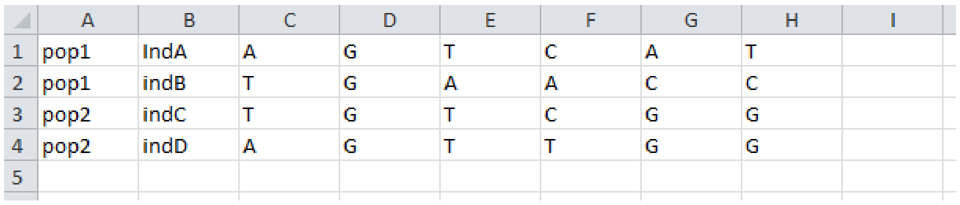
Example of a list of 2 sampled populations, 4 individuals, 3 loci:

Indlist={{“pop1”,”indA”,{{“A”,”G”},{“T”,”C“},{“A”,”T“}},{“pop1”,”indB”,{{“T”,”G”},{“A”,”A“},{“C”,”C“}},{“pop2”,”indC
”,{{“T”,”G”},{“T”,”C“},{“G”,”G“}},{“pop2”,”indD”,{{“A”,”G”},{“T”,”T“},{“G”,”G“}}}

3.2 CSV format

The input file should contain a table with *n* rows (*n*=number of individuals in the sample), where each row contains in the first column the sampled population of the individual and in the second column the individuals’ ID. Columns 3 to 2*L*+2 (*L*=number of Loci in the sample) contain the genotypes, where columns 3-4 contain the alleles for locus 1, columns 5-6 the alleles for locus 2, etc.

Example of a list of 2 sampled populations, 4 individuals, 3 loci:



	A	B	C	D	E	F	G	H	I
1	pop1	IndA	A	G	T	C	A	T	
2	pop1	indB	T	G	A	A	C	C	
3	pop2	indC	T	G	T	C	G	G	
4	pop2	indD	A	G	T	T	G	G	
5									

4. Using Pre-prepared Genetic-Distance matrix

When using the pre-prepared GDmatrix option, two input files are needed: one file containing the genetic distances and one file containing the individuals’ information. The pre-prepared genetic distance can be either the matrix given as an output from the NetStruct process of construction of the GDmatrix, or from an external source. The matrix should be an *n*n* table in MX format. Option for reading matrices from other formats will be available in the following NetStruct version.

The second file should contain the individuals' information (sampled population and individual's ID). This file should be an MX file containing a list of the form {$ind_{1,}$ ind_2,..., ind_n}. Each individual is described by a list of the form ind_i={*Sampled pop*, *Individual ID*}. *Sampled pop* and *Individual ID* are interpreted as strings and therefore do not need to be necessarily numbers. Note that the order of the individuals in this file must match the order of the GDmatrix, so ind_i corresponds to the i^{th} row\column of the GDmatrix.

It is important to make sure that the GDmatrix is symmetric, and that the values on the diagonal are set to 0. All Genetic distance values should be scaled between 0 and 1, with higher values indicating higher genetic similarity.

5. Community Detection

5.1 Community detection algorithms

This version of NetStruct supports 6 different community detection algorithms. These algorithms are implemented in igraph, and the details can be found in the igraph manual (http://igraph.org/r/doc/igraph.pdf).

1. Label propagation (label.propagation.community in igraph) – A stochastic community detection algorithm. Very fast, but does not supply stable results for small networks. Tends to detect communities at higher hierarchical levels.
2. Girvan-Newman (edge.betweenness.community in igraph) – A classic community detection algorithm based on edge betweeness. This algorithm is relatively slow, and should not be used on very large networks.
3. Fast Greedy (fastgreedy.community in igraph)

4. Walktrap (walktrap.community in igraph) – An algorithim based on random walk on networks
5. Spinglass (spinglass.community in igraph)
6. Leading Eigenvector – (leading.eigenvector.community) – An algorithm based on the leading eigenvector of the network's modularity matrix.

5.2 Threshold

The user defined threshold should be between 0 and 1. Edges that represent a genetic distance value below the threshold will be removed from the network. With a low threshold more information will be used, but a higher threshold allows the computation to take into account only the meaningful genetic relations in the population, resulting in a cleaner, more fine-scale, and possibly more significant analysis. However, when the threshold is too high the network breaks down into many disconnected components, and there will be too little information for a meaningful interpretation of the structure. Future versions of NetStruct will have more options and guidelines for determination of the threshold, but for this version it is recommended to try out different thresholds, and use the minimal threshold that reveals a significant structure (removing edges up to a point where significant structure is revealed). Note that different thresholds may reveal structure at different Hierarchical levels.

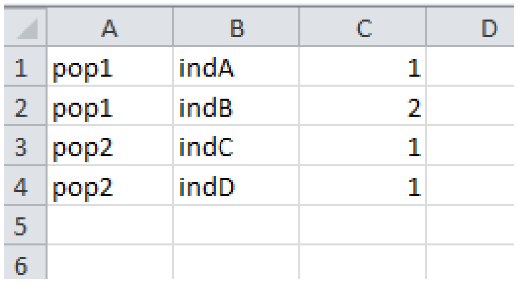
5.3 Community assignment output files

The community detection process produces two output files containing the community assignment information, one as a CSV file and one as an MX file. These files' name end with *_ComAssignment. They are both n*3 tables, with each row containing the sampled population, the individual's ID and the community the individual was assigned to (a number), in this order.

An example of an output MX *_ComAssignment file (2 detected communities, one with 3 individuals and one with one individual):

```
{{pop1,indA,1},{pop1,indB,2},{pop2,indC,1},{pop2,indD,1}}
```

An example of an output CSV *_ComAssignment file (2 detected communities, one with 3 individuals and one with one individual):



	A	B	C	D
1	pop1	indA	1	
2	pop1	indB	2	
3	pop2	indC	1	
4	pop2	indD	1	
5				
6				

Optionally one or two network image files (PDF) can be produced (the file names end with either *_netfig, *_withCoord or *_withoutCoord). These can be without coordinates, and then the standard Mathematica network plotting algorithm is used, with each the node colors correspond to the node's community assignment.
If the option "with coordinates" is selected, then the user will be asked to supply the coordinates of the sampled populations. Additionally, the user will be asked to supply a proximity value, indicating how close to the coordinate to allow the node to be placed (in order that not all nodes from the same population will be placed at the same coordinates). Therefore the proximity value should be smaller than half the distance between the closest sampled populations. It is recommended to use values between 0 and 10 for the coordinates. The colors of the nodes correspond to the assigned communities.

6. SAD analysis

6.1 What is SAD analysis?

SAD (Strength of Association Distribution) analysis explores the distribution of the strengths in which individuals are associated with the community to which they were assigned.

Definitions:

Q_C is the modularity for the community partition *C*. SA (Strength of Association) of individual *i* to community partition *C* is defined as:

$$SA(i,C) = Q_C - \underset{\substack{k \\ C_i(k)\neq C}}{\mathrm{Max}} \; Q_{C_i(k)}$$

Where $C_i(k)$ is the same partition as *C*, except that individual *i* is assigned to community *k* (if *k* is the community *i* is assigned to in C, then $C_i(k) = C$) . The SAD of a community is the distribution of the SA values for the individuals assigned to that community.

One can also define *SdissA*, the Strength of Disassociation. Given a community partition *C*, the disassociation of individual *i* to community *k* is defined as:

$$SdissA(i,C,k) = Q_C - Q_{C_i(k)}$$

SA can be alternatively defined as the minimal Strength of Disassociation of an individual to all non-assigned communities:

$$SA(i,C) = \underset{\substack{k \\ C_i(k)\neq C}}{\mathrm{Min}} \; SdissA(i,C,k)$$

6.2 SAD output files

SAD analysis does not require input files, as it uses the results of the community detection and the GDmatrix used for the community detection. The SAD analysis returns two information files and two chart files.

Information files:

1. CSV *_SAD file, which contain the SA values of all individuals for each community. The file contains *c* rows (*c*=Number of detected communities), with each row containing the SA values for the individuals assigned to the community (first row for community 1, second row for community 2, etc.). The overall number of values should be *n*.

2. CSV *_SADFullSAdetails file. This file contains all the SdissA data. The file contains *n* rows, where each row contains: *Sampled population, Individual's ID*, *assigned community*, $SdissA_1$, $SdissA_2$,…, $SdissA_c$. $SdissA_k$ is the Strength of Disassociation of the individual to community *k*. If the individual is assigned to community *k* then $SdissA_k$ =0.

<u>Chart files:</u>

1. PDF *_SADcharts file which contains a Box-Whisker chart and a Distribution chart of the SAD's of the different communities. The colors correspond to the colors in the network image files.
2. PDF *_SdissAcharts file which contains the SdissA for all individuals. Each bar is labeled with the individual's name and sample population. The colors show the SdissA values of the individual to the non-assigned community (note that the SdissA of an individual to its assigned community is 0, and therefore does not appear).

7. Modularity significance test

Generally, high modularity of a detected community partition indicates a significant structure, but in order to verify that the modularity is higher than would be expected by chance, permutation tests are implemented in NetStruct.

If a pre-prepared GDmatrix was used, the matrix is randomly permuted, except the values on the diagonal which remain 0. The symmetry of the matrix is maintained. If genotypes were used to construct the GDmatrix in NetStruct, then it is possible to permute the genetic data, for each locus independently, and uses the permuted data to construct GDmatrices. While the permutation process in this case is stronger (the GDmatrices do not necessarily contain the same genetic distance values as the original GDmatrix), the process can take a long time when many loci are involved. Therefore, if an analysis is performed using many loci, it is recommended to use permutation of the GDmatrix in order to arrive at reasonable computation times, even if genotype data is available.

The PDF output file, named *_ModularitySignificanceTest, contains a histogram of the modularity values of the community partitions detected by the selected algorithm with the selected threshold of networks constructed from the permuted GDmatrices, or constructed GDmatrices from the permuted genotypes. The dashed lines show the 95% confidence interval (If only one dashed line is presented, then the confidence interval contains only one value). The red lines indicate the modularity for the community partition using the original GDmatrix. If this value is above the 95% interval (the red line is left of the interval defined by the dashed lines), then the modularity is significantly ($p<0.05$) higher than would be expected.