

# ComponentGrapher User Guide

February 2019 / version 1.11



## Table of contents

Table of contents.....	3
1. Pre-requisites .....	4
2. Quickstart.....	4
2.1 Input files.....	5
2.2 Command-line and options.....	6
2.3 Output files .....	8
2.3.1 Network files (_complete.txt, _1.txt, _2.txt, _3.txt, _4.txt).....	10
2.3.2 Node identification (_id.txt) .....	10
2.3.3 Summary files.....	10
2.3.4 Triplets file (triplets.txt).....	14
2.3.5 Graphml files (_graphml.txt) .....	14
3. Miscellaneous .....	15
3.1 External node annotation file .....	15
3.2 Compiling the source code.....	16
4. References .....	17
5. Abbreviations and mathematical formulas.....	18

ComponentGrapher is a multi-platform java program which eases the creation of character-states networks from character matrices in NEXUS or PHYLIP format. This command-line tool also allows the characterization of the resulting networks by searching for global and local articulation points, the numbers of complexes (connect components), betweenness, and other network related statistics. The software and its source code are provided at <https://github.com/etiennelord/ComponentGrapher> using a GPL3 license. This work was developed as a collaboration between the Université de Montréal (UdeM), Université du Québec à Montréal (UQAM) and Sorbonne Université, Université Pierre et Marie Curie, UPMC Paris 6.

## 1. Pre-requisites

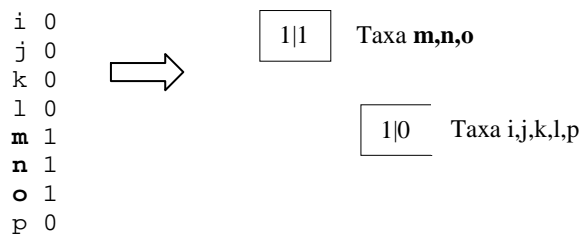
The software ComponentGrapher is implemented as a multi-platform java program. It requires Java SE8 (JDK 8) or later versions of Java. For large matrices (>1000 characters), a fair amount of RAM is required, and should be specified before launching the tool. The *results*, if no output option is specified, will be found in the **results directory**.

**Example for large matrices with allocation of 6 gb of memory [-Xmx6g]:**

```
java -Xmx6g -jar ComponentGrapher.jar sample/matrix.txt
```

## 2. Quickstart

The ComponentGrapher software is to be run as a **command-line utility**. Given an input matrix (section 2.1), the software ComponentGrapher will generate character state networks *i.e.* for each state (e.g. 0,1) of a character, a new node is created (see also table 1 - section 2.3). The created edges will then reflect some characteristics between the character-state nodes (see also table 2 - section 2.3). For example, given a matrix with one character (*Char.1*) and two states (0,1), this will result in 2 nodes (character state):



*Note:* refer to the published paper for more information and section 2.3.

The edges represent the link between pair of nodes. For example, four types of edges can exist if we add another character to this matrix.

	Type I Identical		Type II Inclusion		Type III Overlap		Type IV Disjoint	
	A2	B2	C2	D2	E2	F2	A2	F2
<i>i</i>	-	-	-	-	-	+	-	+
<i>j</i>	-	-	-	-	-	+	-	+
<i>k</i>	-	-	-	+	+	+	-	+
<i>l</i>	-	-	-	+	+	+	-	+
<i>m</i>	+	+	+	+	+	-	+	-
<i>n</i>	+	+	+	+	+	-	+	-
<i>o</i>	+	+	+	+	+	-	+	-
<i>p</i>	-	-	-	+	+	-	-	-

*Note:* the **complete network** is then composed of the link of types 1, 2 and 3.

## 2.1 Input files

The software requires as input either a character matrix either in Phylip format [1] or a character matrix in NEXUS format [2] with **only one** morphological MATRIX tag.

Phylip format (sample/Sample.phy) :

```
5 6
A    110110
B    110000
C    100110
D    001001
E    001110
```

Nexus format (sample/Sample.nex):

```
#NEXUS
[Note: this is a sample matrix]
begin data;
  dimensions ntax=5;
  dimensions nchar=6;
  format datatype=standard gap=- missing=? Interleave=no;
  MATRIX
    A    110110
    B    110000
    C    100100
    D    001001
    E    001110
  ;
end;
```

\*Note that multiple states at one position in the matrix are supported (*e.g.* {0,1}, (0 1), (0,1,2,3) ) as well as interleaves for the NEXUS format. Also note that **undefined states symbol** (*e.g.* ?, -, \*) **are not processed in this version**. For the NEXUS format, column and specific states information can be specified using the CHARSTATELABELS, STATELABELS or CHARLABELS annotation specific to NEXUS format [2].

Nexus format (Sample matrix with multiple states and annotations):

```
#NEXUS
[Note: this is a sample matrix]
begin data;
  dimensions ntax=5;
  dimensions nchar=6;
  format datatype=standard gap=- missing=? Interleave=no;
  CHARSTATELABELS
    1 Mouth / present, absent, 2 'eyes' / two, eight, 3 hands / two,
      'more than 2', 4 hairs / present, absent, 5 'eye color' / blue,
      brown, 6 nose / red, normal
  ;
  MATRIX
    A    110110
    B    1{0,1}0000
    C    100100
    D    00100{0,1}
    E    001110
  ;
end;
```

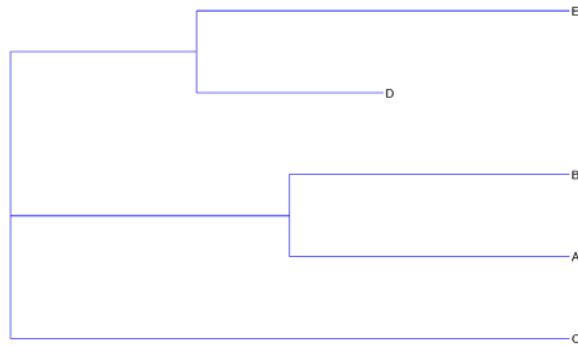
Note: Phylogenetic trees in nexus datasets are not currently supported if phylogenetic permutations are performed (*see below*).

If phylogenetic permutations [-tree] are performed, a tree in the newick format is required.

Newick format (sample/Sample\_tree.newick):

```
((A:1.5,B:1.5):1.5,C:3.0,(D:1.0,E:2.0):1.0);
```

Visualisation of this phylogenetic tree using the trex webserver (<http://www.trex.uqam.ca>)



## 2.2 Command-line and options

### Base command-line

```
java -jar ComponentGrapher.jar matrixfile
java -jar ComponentGrapher.jar matrixfile -tree=treefile
```

### Command-line options

**-tree=file.newick**

Specify the phylogenetic tree in newick format used for the character permutations.

**-perm=100**

Specify the number of permutations to perform.

**Important note:** If no permutation is specified, the default number of permutation is defined as the total number of possible nodes divided by 0.05 (i.e. the *p*-value significance limit).

**-permmode=0**

- 0 : Equiprobable permutation (default option). This results in a random permutation in the state in each columns.
- 1 : Probabilistic permutation. This result in a permutation of the states in each column proportional to the number of each character found in this column.
- 2 : Phylogeny (default if -tree option is used). Perform the phylogenetic permutation defined in the article of *Lapointe and Garland (2001)* [4].
- 3 : Equiprobable of undefined states. Perform the equiprobable permutation but only change the undefined states in the matrix.
- 4 : Probabilistic of undefined states. Perform the probabilistic permutation but only change the undefined states in the matrix.

**Important:** Please specify a random seed using the **-seed** option for better randomness.

**-k=2.0**

Specify the *PhyloPermute* *k* parameter. See the article of Lapointe and Garland (2001) [4].  
The default value is 2.0.

**-filter=...,...**

Filter some nodes from the created networks based on the “*state*” of those nodes. For example, ones might want to remove from the created networks all the nodes containing the state: **absent** indicating that only the present states should be considered. A list separated with commas can be given.

For example, the option `-filter=absent,not_present,?` will remove all nodes with `absent`, `not_present` or `?` states.

**-threads=10**

Specify the number of concurrent threads to be used during the computation. Defaults is 2.

**-edges=0**

Specify edge's inference mode

0 : Treat all edges (default)

1 : Absolute Majority

2 : Majority Rule

>10 : Minimum percent to consider the edge

**-undefined**

Remove columns containing undefined states (e.g.?,\*)

**-multiple**

Remove columns containing multiple states (e.g.[1,2]).

**-bipartite**

Specify that we want to output bipartite files.

**-graphml**

Specify the output of graphml files (Gephi[3] or Cytoscape[10] compatibles, see section 2.3.5).

**-nodeid=file**

Provide a node identification file (see section 2.3.2).

**-output=dir**

Specify output directory. The default is `../results/[matrixfile]`.

**-triplets**

Specify to output the triplets file (`triplets.txt`, see section 2.3.4).

**-seed=42**

Specify the random seed for the random number generator.

### 2.2.1 Base analysis with sample file

```
java -jar ComponentGrapher.jar sample/Sample.phy
```

Results will be located in the [Base component-grapher directory]/results/Sample.phy directory.

### 2.2.2 Analysis without columns containing multiple states and with specified output file [-multiple, -output]

```
java -jar ComponentGrapher.jar matrixfile -multiple -output=result
```

### 2.2.3 Analysis with a phylogenetic permutations

```
java -jar ComponentGrapher.jar sample/Sample.phy -tree=sample/Sample_tree.newick
```

Important note: the supplied phylogenetic tree must be in newick format.

## 2.3 Output files

With a matrix as input, the ComponentGrapher software will generate five network files (Table 3 - below) for each evaluated matrix in either an *edge list* or *graphml* (Gephi [3] and Cytoscape [10] compatible) format, statistics files and a node identification file. In this analysis, each node corresponds to a character-state meaning that each state of a particular character, column in the matrix, will be present in the complete network.

For example, processing the Sample.phy will result in the following nodes (Table 1):

Command-line: `java -jar ComponentGrapher.jar sample/Sample.phy`

```
Sample.phy
5 6
A    110110
B    110000
C    100110
D    001001
E    001110
```

Table 1. Generated nodes found in Sample.phy\_summary\_statistics.tsv<sup>1</sup>

Node ID	Character-state	Character (column)	Associated taxa
<b>0</b>	<b>char. 1 1</b>	<b>char. 1</b>	<b>A,B,C</b>
<b>1</b>	<b>char. 1 0</b>	<b>char. 1</b>	<b>D,E</b>
2	char. 2 1	char. 2	A,B
3	char. 2 0	char. 2	C,D,E
4	char. 3 0	char. 3	A,B,C
5	char. 3 1	char. 3	D,E
6	char. 4 1	char. 4	A,C,E
7	char. 4 0	char. 4	B,D
8	char. 5 1	char. 5	A,C,E
9	char. 5 0	char. 5	B,D
10	char. 6 0	char. 6	A,B,C,E
11	char. 6 1	char. 6	D

<sup>1</sup>In bold, the nodes created for the first state column.

The different files generated correspond to the types of edges (1, 2, 3, or 4) found between each nodes (Table 2).



**Table 2. Type of links (see first section for details)**

Edge type	Directed /Undirected	Link types	Meaning
1	Undirected	Co-existence / identical	Same taxa found in both nodes
2	Directed	Inclusion	One group of taxa is included in the other group of taxa
3	Undirected	Overlap	Overlap between the group of taxa in each node
4*	Undirected	Disjoint	No overlap between the group of taxa for both nodes

\*Note: the disjoint edges (type 4) are not included in the *complete network*.

The generated files for a matrix file **without multiple states** (e.g. more than one possible state per position in the matrix) are shown in Table 3.

**Table 3. Generated networks, node identification and statistics files for a matrix without multiple states.**

Generated files	Content
Filename_summary.txt <sup>1</sup>	Screen output of the simulation (log)
Filename_complete.txt	Edge types: 1,2,3 (complete network/section 2.3.1)
Filename_1.txt	Edge types: 1 (network/section 2.3.1)
Filename_2.txt	Edge types: 2 (network/section 2.3.1)
Filename_3.txt	Edge types: 3 (network/section 2.3.1)
Filename_4.txt	Edge types: 4 (network/section 2.3.1)
Filename_id.txt	Node identification file for the resulting networks (section 2.3.2)
Filename_network_statistics.tsv	Final results for the computed network (section 2.3.3)
Filename_nodes_statistics.tsv	Final results of the computed network nodes
Filename_summary_statistics.tsv	Other statistics for the generated networks (table 5)
	<b>Note: for the generated files, see section 5 for the abbreviations used.</b>
<b>Other generated files</b>	
Randomization_*.json	Internal serialization of the network and statistics during simulation
Replicate_matrix_*.txt	The permuted matrices used for the permutation
Replicate_matrix_*_charstates.txt	The nodes identification for the replicated matrices

<sup>1</sup>Filename refers to the original matrix e.g. Sample.phy\_summary.txt

### 2.3.1 Network files (\_complete.txt, \_1.txt, \_2.txt, \_3.txt, \_4.txt)

The result edge list files are zero-based *tab-separated value* (.tsv) files. Each line corresponds to one edge with a source id (src\_id), destination id (dest\_id), the edge type and the number of common taxa between each node:

Source node ID	Destination node ID	Type of edge	Number of common taxa	Percent of common taxa
----------------	---------------------	--------------	-----------------------	------------------------

```
#src_id(tab)dest_id(tab)edge_type(tab)number_common_taxa(tab)    percent
2                0                2                2                1.0
0                3                3                1                0.5
0                4                1                3                1.0
0                6                3                2                0.75
...
```

\*Note: that there is no fourth and fifth columns (Number of common taxa, percent of common taxa) for the type 4 network.

### 2.3.2 Node identification (\_id.txt)

The result node identification file (*tab-separated values*) contains information about each node in the networks.

Node ID	Name	Char. Label	State label	Encoded State	Edge count	In edge count	Out edge count	Number of taxa	Taxa
---------	------	-------------	-------------	---------------	------------	---------------	----------------	----------------	------

```
0    char.1|1  char.1    1        1        8        1        7        3    A,B,C
1    char.1|0  char.1    0        0        8        1        7        2    D,E
...
```

### 2.3.3 Summary file (.summary.txt, summary\_statistics.tsv, network\_statistics.tsv and nodes\_statistics.tsv)

The software will compute some statistics for each resulting networks and save them in the **results directory**. During operation, the software will also output some statistics (*see below*). Additionally, statistic files are generated for each network. The file ending with "*\_summary.txt*" contains the command-line arguments for this particular run and the screen output, as well as some results such as the number of edges in each network, the number of total nodes and other network statistics (Table 5). The network and nodes statistic files are also available separately in the file "*\_network\_statistics.tsv*" and "*\_nodes\_statistics.tsv*". Finally, a "*\_summary\_statistics.tsv*" is also available and contains some node-specific informations (*see below*).

**Default screen output when executing the software. This output is also found in the \_summary.txt file.**

```
COMPONENT-GRAPHER v1.0.11
Etienne Lord, Jananan Pathmanathan, Vladimir Makarencov,
François-Joseph Lapointe, Éric Baptiste

===== PARAMETERS =====
Command line options      : sample/sample.phy -perm=10 -graphml -triplets
Input                    : sample/sample.phy
Output directory          : results/sample.phy
N taxa                   : 5 (rows)
N characters              : 6 (columns)
Total number of multistate characters: 0
Total number of possible variations*: 1
Total variations tested   : 1
Permutations statistics   : 10
Suggested permutations**  : 240
Remove multiple state columns : false
Remove undefined columns  : false
Total column              : 6
Total undefined column(s) : 0
Total multiple column(s)  : 0
Total treated column      : 6
Total undefined char      : 0
Total multiple char       : 0
Total possible nodes      : 12
Edges selection mode      : treat all edges (default)
Permutations mode         : equiprobable (default)
```

```

Seed : 1001
PhyloPermute k : 2.0
Number of threads : 2
=====
* Indicate the number of state variation for polymorphic characters.
** Indicate the minimum number of permutations for significant p-values.
=====
Total permutations (N replicate) : 0
Total time : 949ms
=====
Trying to allocate memory for 66 possible edges.
=====
Current iteration : 1/1
States variations :
(saving to: results/sample_phy/sample_phy_complete.txt)
=====
6 / 66 ( 0ms )
12 / 66 ( 5ms )
18 / 66 ( 9ms )
24 / 66 ( 13ms )
30 / 66 ( 17ms )
36 / 66 ( 21ms )
42 / 66 ( 24ms )
48 / 66 ( 28ms )
54 / 66 ( 32ms )
60 / 66 ( 35ms )
66 / 66 ( 39ms )
=====
Computing statistics:
=====
(50%) Creating and analyzing intermediate networks ( 11ms ).
(60%) Shortest paths complete network ( 11ms ).
(80%) Shortest paths type 3 network ( 12ms ).
(100%) Done statistics ( 33ms ).
=====
Network statistics (See manual for descriptions)
=====
Statistics Ref p-value sign. N Mean STD Min Max 5% 95%
CC_type1 4.0 0.182 10 1.8 1.135 0.0 4.0 0.0 4.0
CC_type3 1.0 1.0 10 1.0 0.0 1.0 1.0 1.0 1.0
CC_complete 1.0 1.0 10 1.0 0.0 1.0 1.0 1.0 1.0
total_ap_compl. 0.0 1.0 10 0.0 0.0 0.0 0.0 0.0 0.0
n_edge_type1 4.0 0.182 10 1.8 1.135 0.0 4.0 0.0 4.0
n_edge_type2 14.0 0.364 10 16.6 3.534 12.0 22.0 12.0 22.0
n_edge_type3 31.0 0.545 10 31.5 6.276 19.0 38.0 19.0 38.0
n_edge_type4 11.0 0.545 10 10.1 2.331 8.0 15.0 8.0 15.0
n_edge_compl. 49.0 0.545 10 49.9 2.331 45.0 52.0 45.0 52.0
total_ap_type3 0.0 1.0 10 0.0 0.0 0.0 0.0 0.0 0.0
local_ap_type3 7.0 0.091 10 2.5 2.273 0.0 6.0 0.0 6.0
local_ap_compl. 0.0 1.0 10 0.0 0.0 0.0 0.0 0.0 0.0
triplet_type3 107.0 0.091 10 77.8 16.369 39.0 93.0 39.0 93.0
convergence 0.6944 0.364 10 0.623 0.085 0.471 0.72 0.471 0.72
loop4_type3 84.3333 0.545 10 101.267 53.835 23.667 166.333 23.667 166.333
len4_type3 84.3333 0.545 10 101.267 53.835 23.667 166.333 23.667 166.333
den_complete 0.7424 0.545 10 0.756 0.035 0.682 0.788 0.682 0.788
density_type1 0.1429 0.364 10 0.248 0.143 0.0 0.333 0.0 0.333
density_type2 0.2121 0.364 10 0.252 0.054 0.182 0.333 0.182 0.333
density_type3 0.5636 0.545 10 0.591 0.084 0.473 0.691 0.473 0.691
density_type4 0.2444 0.455 10 0.282 0.057 0.222 0.393 0.222 0.393
triangle_type3 14.0 0.182 10 30.8 13.653 10.0 46.0 10.0 46.0
prop_triangle 0.0848 0.091 10 0.193 0.075 0.097 0.279 0.097 0.279
triplet_typeA 107.0 0.091 10 77.8 16.369 39.0 93.0 39.0 93.0
triplet_typeB 23.0 0.091 10 7.8 4.894 0.0 13.0 0.0 13.0
triplet_typeC 32.0 0.455 10 34.2 8.967 16.0 50.0 16.0 50.0
triplet_typeD 28.0 0.091 10 12.6 4.427 4.0 18.0 4.0 18.0
triangle 42.0 0.182 10 92.4 40.959 30.0 138.0 30.0 138.0
time/net. (ms) 12.0 0.091 10 3.0 0.943 2.0 5.0 2.0 5.0
* Some statistic are not available for each permutations, resulting in smaller sample size (N).
=====
Nodes statistics
=====
NodeID 0 Char. 1|0
Statistics Ref p-value Sign. N Mean STD Min Max 5% 95%
in_degree2 1.0 0.091 10 0.3 0.483 0.0 1.0 0.0 1.0
out_degree2 1.0 0.727 10 2.0 1.155 0.0 4.0 0.0 4.0
in_degree3 5.0 0.545 10 4.7 2.541 0.0 9.0 0.0 9.0
in_degree1 1.0 0.091 10 0.5 0.527 0.0 1.0 0.0 1.0
closeness_type3 0.0667 0.909 10 0.062 0.024 0.0 0.091 0.0 0.091
between_type3' 1.6429 0.273 10 1.803 3.201 0.0 10.7 0.0 10.7
$triplet_type3 7.48 0.364 10 7.414 8.924 0.0 31.17 0.0 31.17
triplet_typeA 8.0 0.273 10 6.1 6.919 0.0 24.0 0.0 24.0
triplet_typeB 2.0 0.091 10 0.4 0.843 0.0 2.0 0.0 2.0
triplet_typeC 2.0 0.273 10 2.3 3.653 0.0 12.0 0.0 12.0
triplet_typeD 2.0 0.182 10 1.2 1.874 0.0 6.0 0.0 6.0
triangle 2.0 0.727 10 5.5 4.17 0.0 12.0 0.0 12.0
=====
NodeID 1 Char. 1|1
Statistics Ref p-value Sign. N Mean STD Min Max 5% 95%
...

```

**Table 5. The network and node statistics files: [Filename]\_network\_statistics.tsv and [Filename]\_nodes\_statistics.tsv display the following information:**

Columns	Descriptions
<b>A</b>	Statistics (refer to section 5 for descriptions)
<b>B</b>	Reference value. The value associated with the computed original network.
<b>C</b>	The calculated p-value
<b>D</b>	Significance of the $p$ -value. This indicate if the $p$ -value should be considered significant. If no star (*) is present, the value should not be considered statistically different. Legend: * $p < 0.05$ ** $p < 0.01$ *** $p < 0.001$
<b>E</b>	Number of permutation performed.
<b>F</b>	Minimum value found in the permuted dataset.
<b>G</b>	Maximum value found in the permuted dataset.
<b>H</b>	Mean value found in the permuted dataset.
<b>I</b>	Standard deviation found in the permuted dataset.
<b>J</b>	5% percentile value found in the permuted dataset.
<b>K</b>	95% percentile value found in the permuted dataset.

Example:

Columns: A	B	C	D	E	F	G	H	I	J	K
Statistics	Ref.	p-value	sign.	N	Mean	STD	Min	Max	5%	95%
CC_type1	4.0	0.12	*	30	1.46	1.27	0.0	4.0	0.0	4.0
CC_type3	1.0	1.0		30	1.0	0.0	1.0	1.0	1.0	1.0
CC_complete	1.0	1.0		30	1.0	0.0	1.0	1.0	1.0	1.0
total_ap_compl.	0.0	1.0		30	0.0	0.0	0.0	0.0	0.0	0.0
...										

**Note:  $p$ -values are not calculated if the reference value is either absent (NA) or equal to zero.**

The file [Filename]\_summary\_statistics.tsv contains 21 fields in the following order:

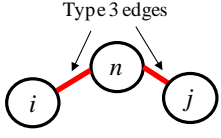
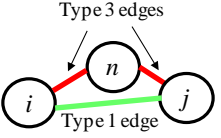
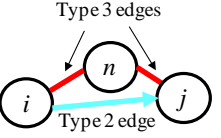
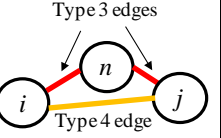
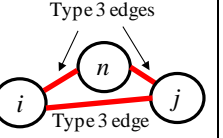
Column	Descriptions
Nodeid	Node numbering (zero-based)
Name	Node name
Found in type 1 network	Node is found in type 1 network (Presence indicate by a x)
Found in type 2 network	Node is found in type 2 network (Presence indicate by a x)
Found in type 3 network	Node is found in type 3 network (Presence indicate by a x)
Found in complete network	Node is found in complete network (Presence indicate by a x)
Column	Column in the matrix (starting at index 1)
Encoded state	Encoded state in the matrix (e.g. 0,1)
Character	If provided in the matrix, the full description of this character and its associated state (e.g. red nose   present)
Connected component type 1 network	The connected component associated with this node in the type 1 network
Connected component complete network	The connected component associated with this node in the complete network
Local articulation point type 3 network	In the type 3 network, is this node a local articulation point? If present, indicated as a x.
Global articulation point type 3 network	In the type 3 network, is this node a global articulation point? If present, indicated as a x.
Local articulation point in complete network	In the complete network, is this node a local articulation point? If present, indicated as a x.
Global articulation point in complete network	In the complete network, is this node a global articulation point? If present, indicated as a x.
Indegree type 2 network	This node "in" degree in the directed type 2 network.
Normalized indegree type 2 network	This node normalized "in" degree in the directed type 2 network.
Betweenness type 3 network	Betweenness centrality of node in type 3 network.
Closeness type 3 network	Closeness centrality of node in type 3 network
Numbers of triplet type 3 network	Number of triplets in the type 3 network containing this node. Here, triplet stands for a linear series of 3 nodes where the terminal nodes are unconnected.
Percent(%) triplet type 3 network	Percent of those triplets over the total number of triplets in the type 3 network.

### 2.3.4 Triplets file (triplets.txt)

If the [-triplets] option is selected, the software will compute the triplets and generate the file triplets.txt. In this file each line corresponds to one triplet with the first node (node 1), the second node which is in the middle of the triplet (central node), the third node (Node 3) and the triplet type (Table 6):

Node 1	Central node	Node 3	Type
7	0	6	Type A
8	0	6	Type A
8	0	6	Type B
8	0	7	Type A
9	0	6	Type A
...			

**Table 6: The different types of identified triplets are described in the following table.**

Triplettype A	Triplet type B	Triplet type C	Triplet type D	Triplet type E
				
Nodes <i>i</i> and <i>j</i> are not connected. Central node <i>n</i> is linked with type 3 edges.	Nodes <i>i</i> and <i>j</i> are connected by a type 1 edge. Central node <i>n</i> is linked with type 3 edges	Nodes <i>i</i> and <i>j</i> are connected by a type 2 edge. Central node <i>n</i> is linked with type 3 edges	Nodes <i>i</i> and <i>j</i> are connected by a type 4 edge. Central node <i>n</i> is linked with type 3 edges	Nodes <i>i</i> and <i>j</i> are connected by a type 4 edge. Central node <i>n</i> is linked with type 3 edges (e.g. it forms a <i>triangle</i> )

### 2.3.5 Graphml files (\_graphml.txt)

If the [-graphml] option is selected, the software will generate a graphml file for each network and compatible with Gephi and Cytoscape, e.g. sample.phy\_\_1.graphml (network type 1) :

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<graphml>
<key id='k2' for='edge' attr.name='type' attr.type='double' />
<key id='k1' for='edge' attr.name='total_shared_taxa' attr.type='double' />
<key id='k3' for='node' attr.name='fullname' attr.type='string' />
<key id='k4' for='node' attr.name='number_of_taxa' attr.type='double' />
<key id='k5' for='node' attr.name='partition' attr.type='string' />
<key id='k6' for='node' attr.name='total_edges' attr.type='double' />
<key id='k61' for='node' attr.name='in_edges' attr.type='double' />
<key id='k62' for='node' attr.name='out_edges' attr.type='double' />
<key id='k7' for='node' attr.name='associated_character_column' attr.type='double' />
<key id='k8' for='node' attr.name='charlabel' attr.type='string' />
<key id='k9' for='node' attr.name='statelabel' attr.type='string' />
<key id='k10' for='node' attr.name='statematrix' attr.type='string' />
<key id='k11' for='node' attr.name='total_taxa' attr.type='double' />
<key id='k12' for='node' attr.name='taxa_id' attr.type='string' />
<graph edgedefault='undirected' id=''>
</graph>
</graphml>
```

### 3. Miscellaneous

#### 3.1 External node annotation file

If needed, you can supply a list of character-state annotation if none is provided in the NEXUS matrix or if you use a Phylip matrix file. To do so, include a list of node character-states with the [-nodeid=] option.

##### Example command-line with the sample.phy matrix

```
java -jar COMPONENTGRAPHER.jar sample/Sample.phy -nodeid=sample/Sample_nodeid.txt
```

This file should be a *tab-separated value* with four columns: column number (character position starting at 1), encoded state, new character label, new state label:

		Column	State	Character label	State label
<u>Sample_nodeid.txt</u>					
1	0	Mouth opening orientation		Terminal	
1	1	Mouth opening orientation		Ventral	
2	0	Pre-oral chamber		Absent	
2	1	Pre-oral chamber		Present	
3	0	Radially arranged circumoral structures		Absent	
3	1	Radially arranged circumoral structures		Present	
4	0	Differentiated circumoral structures		undifferentiated plates	
4	1	Differentiated circumoral structures		three or four enlarged plates	
5	0	Pharynx differentiated from midgut		not differentiated	
5	1	Pharynx differentiated from midgut		differentiated	
6	0	Pharynx reversible		permanently inverted	
6	1	Pharynx reversible		completely reversible	

These annotations will be reflected in the **node identification file** (\_id.txt) and in the **summary file** (\_summary.txt) as well as in the general screen output:

##### Without the nodeid file

NodeID	0	Char.	1 0							
Statistics		Ref	p-value	Sign.	N	Mean	STD	Min	Max	5% 95%

##### With the nodeid file

NodeID	0	Mouth opening orientation terminal								
Statistics		Ref	p-value	Sign.	N	Mean	STD	Min	Max	5% 95%

### 3.2 Compiling the source code

The java source code provided is a **Netbeans project** (<https://netbeans.org>). To compile the source code, download the Netbeans IDE, open the project file and build the project. The required libraries are listed below:

#### **SSJ(<https://github.com/umontreal-simul/ssj>)**

The SSJ library is used for BitVector calculations and for computing random uniform distribution. The `ssj.jar` archive (version 3.2.0) is included in the COMPONENT-GRAPHER distribution and it must be in the CLASSPATH environment variable.

#### **Apache Commons Mathematics Library (<http://commons.apache.org/proper/commons-math/>)**

The Apache common mathematics library is used for distribution and  $p$ -value calculations. The `commons-math3-3.6.1.jar` archive is included in the COMPONENT-GRAPHER distribution and it must be in the CLASSPATH environment variable.

#### **Google gson (<https://github.com/google/gson>)**

The Google gson library is used for serialization/deserialization of the datasets. The `gson-2.6.2.jar` archive is included in the COMPONENT-GRAPHER distribution and it must be in the CLASSPATH environment variable.

#### **Forester.jar (<https://sites.google.com/site/cmzmasek/home/software/forester>)**

Forester is a collection of open source libraries for phylogenomics and evolutionary biology research created by Christian Zmasek. The `forester.jar` archive is included in the COMPONENT-GRAPHER distribution and it must be in the CLASSPATH environment variable.



## 4. References

- [1] Felsenstein, J. 2005. PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.
- [2] Maddison DR, Swofford DL, Maddison WP. 1997. NEXUS: an extensible file format for systematic information. *Syst Biol.* 46(4):590-621.
- [3] Bastian M., Heymann S., Jacomy M. 2009. Gephi: an open source software for exploring and manipulating networks. International AAAI Conference on Weblogs and Social Media.
- [4] Lapointe, F. J., & Garland Jr, T. 2001. A generalized permutation model for the analysis of cross-species data. *Journal of Classification*, 18(1), 109-127.
- [5] Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2), 163-177.
- [6] Brandes, U. (2008). On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2), 136-145.
- [7] Saryüce, A. E., Kaya, K., Saule, E., & Catalyürek, U. V. (2013, October). Incremental algorithms for closeness centrality. In *IEEE International Conference on BigData*.
- [8] Anthonisse, J. M. (1971). The rush in a directed graph. Stichting Mathematisch Centrum. Mathematische Besliskunde, (BN 9/71).
- [9] Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 35-41.
- [10] Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D. & Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11), 2498-2504.
- [11] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT press.
- [12] Schank, T, and Wagner, D. (2005) Finding, counting and listing all triangles in large graphs, an experimental study. International Workshop on Experimental and Efficient Algorithms. Springer Berlin Heidelberg, 2005.

## 5. Abbreviations and mathematical formulas

Some abbreviations can be found in the software outputs.

Abbreviations	Descriptions
<b>N</b>	Number of permutations (replicates)
<b>%triplet_type3</b>	Number of time (in percent) that this node is central in a triplet
<b>between_type3</b>	Node betweenness centrality [5,8-9] in type3 network
<b>between_type3'</b>	Betweenness centrality [5,8-9] of node (type 3 network)
<b>CC_complete</b>	Number of complexes (complete network composed of type 1, 2, and 3 edges)
<b>CC_type1</b>	Number of complexes (type 1 network)
<b>CC_type3</b>	Number of complexes (type 3 network)
<b>closeness_type3</b>	Closeness centrality [5,7] of node (type 3 network)
<b>den_complete</b>	Network density <sup>1</sup> (complete network composed of type 1, 2, and 3 edges)
<b>density_type1</b>	Network density (type 1 network)
<b>density_type2</b>	Network density (type 2 network)
<b>density_type3</b>	Network density (type 3 network)
<b>density_type4</b>	Network density (type 4 network)
<b>in_degree1</b>	Degree of nodes (type 1 network, undirected)
<b>in_degree2</b>	In-degree of nodes (type 2 network)
<b>in_degree3</b>	Degree of nodes (type 3 network, undirected)
<b>loop4_type3</b>	Total loop of length 4 (quad) in (type 3 network (undirected)
<b>len4_type3</b>	Total paths of length 4 in (type 3 network (undirected)
<b>n_edges_complete</b>	Number of edges (complete network composed of type 1, 2, and 3 edges)
<b>n_edges_type1</b>	Number of edges (type 1 network)
<b>n_edges_type2</b>	Number of edges (type 2 network)
<b>n_edges_type3</b>	Number of edges (type 3 network)
<b>n_edges_type4</b>	Number of edges (type 4 network)
<b>out_degree2</b>	Out-degree of nodes (type 2 network)
<b>prop_triangle</b>	Proportion of triangles (type 3 network, over the total number of possible triangles in a complete graph)
<b>sign.</b>	Significance of p-value from the permutation statistics and Bonferroni correction for multiple tests (* p<0.05, ** p<0.01, *** p<0.001)
<b>time/net. (ms)</b>	Time to process a network (in milliseconds)
<b>total_ap_compl.</b>	Number of global articulation points (complete network composed of type 1, 2, and 3 edges)
<b>total_ap_type3</b>	Number of global articulation points (type 3 network)
<b>local_ap_compl.</b>	Number of local articulation points (complete network composed of type 1, 2, and 3 edges)

<b>local_ap_type3</b>	Number of local articulation points (type 3 network)
<b>triangle_type3</b>	Number of triangles (type 3 network)
<b>triplet_type3</b>	Number of non-transitive triplets (type 3 network)
<b>triplet_typeA</b>	Number of triplets of type A. For a node n, the triplet is in the form i -3- n -3- j where i and j are other nodes connected to n with a type 3 edge and i and j are not connected)
<b>triplet_typeB</b>	Number of triplets of type B. For a node n, the triplet is in the form i -3- n -3- j where i and j are other nodes connected to n with a type 3 edge and i and j are connected with a type 1 edge)
<b>triplet_typeC</b>	Number of triplets of type C. For a node n, the triplet is in the form i -3- n -3- j where i and j are other nodes connected to n with a type 3 edge and i and j are connected with a type 2 edge)
<b>triplet_typeD</b>	Number of triplets of type D. For a node n, the triplet is in the form i -3- n -3- j where i and j are other nodes connected to n with a type 3 edge and i and j are connected with a type 4 edge)
<b>triangle</b>	Triangle where node n is present (type 3 network)

<sup>1</sup>The ratio of the number of edges to the number of possible edges in the network.

Some information about the calculated network statistics. The number of **connected components** are computed using the depth-first search algorithm [11]. The nodes **betweenness centrality** [5,8-9] is calculated using formula 1 [5] and the algorithm of Brandes (2008) [6]

**Formula 1.**

$$B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

Where  $\sigma_{st}$  denotes the number of shortest paths from  $s \in V$  to  $t \in V$ , where  $V$  is the vertices (nodes) of a graph. The symbol  $\sigma_{st}(v)$  then indicate the number of shortest paths from  $s$  to  $t$  that some  $v \in V$  lies on (Brandes, 2001[5]). For **closeness centrality**, the algorithm used was from Sarıyüce *et al.* (2013 [7]) and calculated using formula 2. In this later formula [5], the symbol  $d_G(v, t)$  denote the distance between the nodes  $s$  and  $t$ .

**Formula 2.**

$$C_c(v) = \frac{1}{\sum_{t \in V} d_G(v, t)}.$$

To calculate the **number of triangles** in each network, the fast procedure of Schank and Wagner (2005) [11] was used. All the source code can be found in the graph.java source code.