COMPONENT-GRAPHER User Guide

Mars 2016 / version 1.0

Table of contents

[1. Pre-requisites 4](#_Toc445373930)

[2. Quickstart 4](#_Toc445373931)

[2.1 Input files 5](#_Toc445373932)

[2.2 Command-line and options 7](#_Toc445373933)

[2.3 Output file 8](#_Toc445373934)

[2.3.1 Network files (\_complete.txt, \_1.txt, \_2.txt, \_3.txt, \_4.txt) 12](#_Toc445373935)

[2.3.2 Global network files (global\_complete.txt, global\_1.txt, global\_2.txt,…) 12](#_Toc445373936)

[2.3.3 Node identification (\_id.txt) 13](#_Toc445373937)

[2.3.4 Statistic file (\_stat.txt) 13](#_Toc445373938)

[2.3.5 Summary files (\_summary.txt, \_degree.txt) 14](#_Toc445373939)

[3. Miscellaneous 15](#_Toc445373940)

[3.1 External node annotation file 15](#_Toc445373941)

[3.2 Compiling 17](#_Toc445373942)

[3.3 Citation 17](#_Toc445373943)

[4. References 17](#_Toc445373944)

COMPONENT-GRAPHER 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 connected component (CC), betweenness, and other network related statistics. The software and its source code are provided at https://github.com/etiennelord/COMPONENT-GRAPHER 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 Universités, UPMC Paris 6.

# 1. Pre-requisites

The software COMPONENT-GRAPHER is implemented as a multi-platform java program. It requires Java SE5 (JDK 5) or later versions of Java. For large matrices (>2000 characters), a fair amount of RAM is required, and should be specified before launching the tool.

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

java -Xmx6g -jar COMPONENT-GRAPHER.jar large\_matrix.nex

# 2. Quickstart

Given an input matrix, the software COMPONENT-GRAPHER 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 (C*har.1*) and two states, this will result in 2 nodes (character|state):

A 0

Taxa E,F,G

B 0

C 0

D 0

E 1

Taxa A,B,C,D,H

F 1

G 1

H 0

The edges represent the link between pair of nodes. For example, four type of links can exists if we add another character to this matrix (*Char.2*)



The complete network is then composed of the link of types 1 to 3.

### 2.1 Input files

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

Phylip format (Sample.phy) :

5 6

A 110110

B 110000

C 100110

D 001001

E 001110

Nexus format (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;

### 2.2 Command-line and options

**Base command-line**

java -jar COMPONENT-GRAPHER.jar matrixfile

**Command-line options**

-taxa=list : Specify some taxa tagged in the summary file

(list separated by comma e.g. A,B,C or with a star flag

e.g. homo\* for homo\_sapiens, homo\_erectus...).

-mintaxa=9[%]: Sepecify either the minimum number of common taxa or the

percent of common taxa to link two nodes in the final

networks.

-maxiter=999 : Maximum number of iterations to search in case of

undefined states in the input matrix (e.g. {1,2,3}).

\*Note: the first 1000 iterations are ordered and not

random. (default=1).

-random=999 : Specify the number of random iterations that we want

in case of undefined states.

-undefined : Remove column containing undefined states(e.g. ?,-)

-multiple : Remove column containing multiple states (e.g. {1,2,3}).

-graphml : Output graphml file.

-nodeid=file : Provide a node identification file (*see section 3.1.1*)

-output=file : Specify output file name.

-variation=X : Specify the variation string to use.

-summary : Compute and output summary statistic.

**2.2.1 Base analysis**

java -jar COMPONENT-GRAPHER.jar matrixfile

**2.2.2 Base analysis with summary statistic [-summary] (*see section 2.3.4*)**

java -jar COMPONENT-GRAPHER.jar matrixfile -summary

\*This will generate two additional files, a *matrixfile\_summary.txt* file containing and a *matrixfile\_degree.txt* file containing degree information for each nodes.

**2.2.1 Base analysis with summary containing more information for taxon A and B.**

java -jar COMPONENT-GRAPHER.jar matrixfile -summary -taxa=A,B

**2.2.3 Analysis without column containing multiple states and with specified output file [-multiple, -output]**

java -jar COMPONENT-GRAPHER.jar matrixfile -multiple -output=result

**2.2.4 Generate the first one hundred possible variations of a multiple states matrix**

Phylip format (Sample\_n.phy) (128 possible variations) :

5 6

A 210114

B 110{0,1}0{2,3}

C 100112

D {2,3}01001

E 3{0,1}1{1,0}1{1,2,3,4}

Given the matrix **Sample\_n.phy**, this will generate the 100 first variations:

java -jar COMPONENT-GRAPHER.jar Sample\_n.phy -maxiter=100

Alternatively, this will generate 100 random variations (out of the 128 possibles):

java -jar COMPONENT-GRAPHER.jar Sample\_n.phy -random=100

These last two commands will results in the generation of 100 group of files (*see Table 3 below*) in the form: sample\_n.phy\_XXX\_complete.txt where XXX correspond to this analysis iteration (*e.g.* 1 to 100 in the present case). It is worth mentioning that if there is more than 1,000 possible states, the states will automatically be chosen at random using an Uniform distribution provided by the SSJ java package (https://github.com/umontreal-simul/ssj; Pierre L'Ecuyer's Simulation and Optimization Laboratory, located at the Department of Computer Science and Operations Research of Université de Montréal.).

**2.2.5 Use a specific variation [-variation]**

java -jar COMPONENT-GRAPHER.jar Sample\_n.phy -variation=032011

Note that the different position in the matrix are visited from left to right, from first taxon to last taxon.

5 6

A 210114

B 110{0,1}0{2,3}

C 100112

D {2,3}01001

E 3{0,1}1{1,0}1{1,2,3,4}

### 2.3 Output file

With a matrix as input, the COMPONENT-GRAPHER software will generate five network files (Table 3 - below) for each evaluated matrix in either an *edge list* or *graphml* (Gephi [3] compatible) format, a statistic file (.stat.txt) and a node identification file (.id.txt). In this analysis, each node correspond 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):

Sample.phy

5 6

A **1**10110

B **1**10000

C **1**00110

D **0**01001

E **0**01110

**Table 1. Generated nodes from the Sample.phy matrix\***

|  |  |  |  |
| --- | --- | --- | --- |
| **Node ID** | **Character-state** | **Character (column)** | **Associated taxa** |
| **0** | **char. 1|1** | **char. 1** | **A,B,C** |
| **1** | **char. 1|0** | **char. 1** | **D,E** |
| 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 |

\* 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 main publication for more information*)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Edge type** | **Directed**  **/Undirected** | **Link types** | **Meaning** |
| 1 | Undirected | Co-existence /  perfect match | 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 (Table 3) for a matrix file **without multiple states** (*e.g.* more than one possible state per position in the matrix) are as follow:

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

|  |  |
| --- | --- |
| **Generated files** | **Content** |
| Filename\_complete.txt | Edge of types: 1,2,3 |
| Filename\_1.txt | Edge of types: 1 |
| Filename\_2.txt | Edge of types: 2 |
| Filename\_3.txt | Edge of types: 3 |
| Filename\_4.txt | Edge of types: 4 |
| Filename\_id.txt | Node identification |
| Filename\_stat.txt | Summary statistic of the generated networks and command-line options |

In case of multiple states per position in the matrix (Table 4), the generated files are has follow:

**Table 4. Generated networks, node identification and statistics files for a matrix without multiple states**

|  |  |
| --- | --- |
| **Generated files** | **Content** |
| Filename\_XXX\_complete.txt | Edge of type 1,2 and 3 for the XXX variation |
| Filename\_ XXX\_1.txt | Edge of type 1 for the XXX variation |
| Filename\_ XXX\_2.txt | Edge of type 2 for the XXX variation |
| Filename\_ XXX\_3.txt | Edge of type 3 for the XXX variation |
| Filename\_ XXX\_4.txt | Edge of type 4 for the XXX variation |
| Filename\_ XXX\_id.txt | Node identification |
| Filename\_ XXX\_stat.txt | Summary statistic of the generated networks and command-line options |
| Filename\_global\_complete.txt | Edge of type 1,2 and 3 and their appearance (count) over all variations |
| Filename\_global\_1.txt | Edge of type 1 and their appearance (count) over all variations |
| Filename\_global\_2.txt | Edge of type 2 and their appearance (count) over all variations |
| Filename\_global\_3.txt | Edge of type 3 and their appearance (count) over all variations |
| Filename\_global\_4.txt | Edge of type 4 and their appearance (count) over all variations |

Where XXX is replaced by the current iteration number. The current evaluated states are listed in the Filename\_XXX\_stat.txt For example, for the Sample\_n.phy\_2.txt the state string 12**2**011 correspond to the state **2** for the taxon **D** for the column **1** as noted in the statistic file:

Sample\_n.phy

5 6

A 210114

B 110{0,1}0{2,3}

C 100112

D {2,3}01001

E 3{0,1}1{1,0}1{1,2,3,4}

Sample\_n.phy\_2\_stat.txt

...

Input : data/sample\_n.phy

**Total iterations : 6 (Number of generated**

**networks)**

Iteration mode : random

N taxa : 5 (rows)

N characters : 6 (columns)

Total number of multistate characters: 6

**Total number of possible variations : 128 (Total number of**

**possible networks)**

Total undefined column(s) : 0

Total multiple column(s) : 4

Total undefined char : 0

Total multiple char : 6

Remove multiple state columns : false

Remove undefined columns : false

...

States evaluated: 02**2**004

Taxon->Character(column)|value

----------------------------------------

B->4|1

B->6|2

**D->1|2**

E->2|0

E->4|1

E->6|1

### 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\t)dest\_id(tab\t)edge\_type(tab\t) number\_common\_taxa(tab\t)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 Global network files (global\_complete.txt, global\_1.txt, global\_2.txt,…)

These contain the resulting edges of multiple variations i.e. if the original matrix contains multiple states at some positions and the [-maxiter] or [-random] command-line options are used to compute more than one possible variation of the matrix.

These 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 percent of appearance over all variations:

|  |  |  |  |
| --- | --- | --- | --- |
| Source  node ID | Destination  node ID | Type of edge | Percent of appearance |

#src\_id(tab\t)dest\_id(tab\t)edge\_type(tab\t)percent

2 0 2 0.10

0 3 3 1.00

0 4 1 0.75

0 6 3 0.66

...

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

### 2.3.3 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

...

### 2.3.4 Statistic file (\_stat.txt)

This file contains the command-line arguments for this particular run, as well as some results such as the number of edges in each network and the number of total nodes.

Example for the file Sample\_n.nex\_100.stat.txt

=============================== PARAMETERS ================================

Command line options : data/sample\_n.phy -random=100

Input : data/sample\_n.phy

N taxa : 5 (rows)

N characters : 6 (columns)

Total number of multistate characters: 6 (Positions in the matrix with multi-states)

Total number of possible variations : 128 (Number of possible matrices)

Total variations tested : 100 (Total variation of matrices tested)

Iteration mode : random (Either in order or random)

Total undefined char : 0 (Number of undefined characters.)

Total multiple char : 6 (Number of multistate characters)

Total undefined column(s) : 0 (Number of column with undefined char.)

Total multiple column(s) : 4 (Number of column with multistate char.)

Remove multiple state columns : false (Remove column with multistate char.)

Remove undefined columns : false (Remove column with undefined char.)

Total treated column : 6

Total possible nodes : 15

===========================================================================

Current iteration : 100/100 (Current variation number.)

States variations : 023004 (Current variation string)

(saving to: data/sample\_n.phy\_100\_complete.txt)

===========================================================================

Results:

===========================================================================

Edges (total) : 6051 (Total edges in complete network)

Edges type 1 (perfect) : 281 (Total edges in type 1 network)

Edges type 2 (inclusion) : 2367 (Total edges in type 2 network)

Edges type 3 (overlap) : 3403 (Total edges in type 3 network)

Edges type 4 (disjoint) : 3149 (Total edges in type 4 network)

Total nodes evaluated : 15 (Total generated character-state nodes)

Total nodes : 14 (Total nodes included in the networks)

Node (unassigned) : 1 (Total nodes that are not in any network)

Node type 1 (perfect) : 4 (Nodes in type 1 network)

Node type 2 (inclusion) : 14 (Nodes in type 2 network)

Node type 3 (overlap) : 12 (Nodes in type 3 network)

Node type 4 (disjoint) : 15 (Nodes in type 4 network)

...

===========================================================================

States variations : 023004 (Current variation string, *see beginning of section*)

Taxon->Character(column)|Value (Corresponding matrix position)

---------------------------------------------------------------------------

B->4|0

B->6|2

D->1|3

E->2|0

E->4|0

E->6|4

===========================================================================

### 2.3.5 Summary files (\_summary.txt, \_degree.txt)

If the [-summary] option is selected, the software will compute some statistics for each resulting networks. Two files are generated. The first one (\_degree.txt) contains node's degree information for the 3 types of networks in a tab-separated value format (tsb).

\_degree.txt files

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Node  ID | In  degree  type 1 | Out  degree  type 1 | In  degree  type 2 | Out  degree  type 2 | In  degree  type 3 | Out  degree  type 3 |

0 1 1 1 1 5 5

1 1 1 1 1 5 5

2 0 0 0 3 4 4

3 0 0 3 0 7 7

...

\_summary.txt files

The (\_summary.txt) summary file contains 31 fields in the following order:

nodeid,contains\_taxa,found\_in\_type\_1,found\_in\_type\_2,found\_in\_type\_3,found\_in\_complete,column,encoded\_state,char.|states,cc\_type1,cc\_complete,local\_ap\_type3,global\_ap\_type3,local\_ap\_complete,global\_ap\_complete,in\_degree\_type2,norm\_indegree\_type2,betweenness\_type3,closeness\_type3,triplet\_type3,per\_triplet\_type3,triplet\_complete,Per\_triplet\_complete,max\_shortest\_path\_type3,max\_shortest\_path\_complete,convergence,progressive\_transition\_total,progressive\_transition\_end\_node,contains\_taxa,percent\_contained,taxa

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Fields** | **Description** | **Networks** |
| 0 | nodeid | Node numbering (zero-based) | na |
| 1 | contains\_taxa | If the [-taxa=] option is set, indicate if this node contains those particular taxon. (Presence indicate by a x) | na |
| 2 | found\_in\_type\_1 | Node is found in type 1 network  (Presence indicate by a x) | 1 |
| 3 | found\_in\_type\_2 | Node is found in type 2 network  (Presence indicate by a x) | 2 |
| 4 | found\_in\_type\_3 | Node is found in type 3 network  (Presence indicate by a x) | 3 |
| 5 | found\_in\_complete | Node is found in complete network (If present, indicate by a x) | complete |
| 6 | column | Column in the matrix (starting at index 1) | na |
| 7 | encoded\_state | Encoded state in the matrix (*e.g.* 0,1) | na |
| 8 | char\_states | If provided in the matrix, the full description of this character and its associated state (e.g. red nose | present) | na |
| 9 | cc\_type1 | The connected component associated with this node in the type 1 network | 1 |
| 10 | cc\_complete | The connected component associated with this node in the complete network | complete |
| 11 | local\_ap\_type3 | In the type 3 network, is this node a local articulation point? If present, indicated as a x. | 3 |
| 12 | global\_ap\_type3 | In the type 3 network, is this node a global articulation point? If present, indicated as a x. | 3 |
| 13 | local\_ap\_complete | In the complete network, is this node a local articulation point? If present, indicated as a x. | complete |
| 14 | global\_ap\_complete | In the complete network, is this node a global articulation point? If present, indicated as a x. | complete |
| 15 | in\_degree\_type2 | This node "in" degree in the directed type 2 network. | 2 |
| 16 | norm\_indegree\_type2 | This node normalized "in" degree in the directed type 2 network. | 2 |
| 17 | betweenness\_type3 | Betweenness of node in type 3 network. | 3 |
| 18 | closeness\_type3 | Closeness of node in type 3 network. | 3 |
| 19 | triplet\_type3 | 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. | 3 |
| 20 | per\_triplet\_type3 | Percent of those triplets over the total number of triplets in the type 3 network. | 3 |
| 21 | triplet\_complete | Number of triplets in the complete network containing this node. | complete |
| 22 | per\_triplet\_complete | Percent of those triplets over the total number of triplets in the complete network. | complete |
| 23 | max\_shortest\_path\_type3 | Maximum length of the shortest path for this node in the type 3 network. | 3 |
| 24 | max\_shortest\_path\_complete | Maximum length of the shortest path for this node in the complete network. | complete |
| 25 | convergence | The ratio of total loop on length 3 and 4 divided by the total number of length 4 in the type 3 network. | 3 |
| 26 | progressive\_transition | Progressive trantition stands for short paths of length greater than 2 in type 3 network that are not smaller in the complete network. | 3/complete |
| 27 | progressive\_transition\_end\_node | List of the terminal node of the progressive transition. | na |
| 28 | contains\_taxa | Duplicate of column 1 | na |
| 29 | percent\_contained | Ratio of the contains\_taxa over the total number of taxa for this node | na |
| 30 | taxa | The taxa included in this node | na |

# 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 option [-nodeid=].

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

java -jar COMPONENT-GRAPHER.jar sample.phy -nodeid=sample\_nodeid.txt

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

|  |  |  |  |
| --- | --- | --- | --- |
| Column | State | Character label | State label |

sample\_nodeid.txt

|  |  |  |  |
| --- | --- | --- | --- |
| 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).

sample.phy\_id.txt (before)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Node**  **id** | **complete\_name** | **char\_label** | **state\_label** | ... |
| 0 | char. 1|1 | char. 1 | 1 | ... |
| 1 | char. 1|0 | char. 1 | 0 | ... |
| 2 | char. 2|1 | char. 2 | 1 | ... |
| 3 | char. 2|0 | char. 2 | 0 | ... |
|  | ... | ... | ... | ... |

sample.phy\_id.txt (after provided annotations)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Node**  **id** | **complete\_name** | **char\_label** | **state\_label** | ... |
| 0 | Mouth opening orientation|ventral | Mouth opening orientation | ventral | ... |
| 1 | Mouth opening orientation|terminal | Mouth opening orientation | terminal | ... |
| 2 | Pre-oral chamber|present | Pre-oral chamber | present | ... |
| 3 | Pre-oral chamber|absent | Pre-oral chamber | absent | ... |
| 4 | Radially arranged circumoral structures|absent | Radially arranged circumoral structures | absent | ... |
| 5 | Radially arranged circumoral structures|present | Radially arranged circumoral structures | present | ... |
| 6 | Differentiated circumoral structures|three or four enlarged plates | Differentiated circumoral structures | three or four enlarged plates | ... |
| 7 | Differentiated circumoral structures|undifferentiated plates | Differentiated circumoral structures | undifferentiated plates | ... |
| 8 | Pharynx differentiated from midgut|differentiated | Pharynx differentiated from midgut | differentiated | ... |
| 9 | Pharynx differentiated from midgut|not differentiated | Pharynx differentiated from midgut | not differentiated | ... |
| 10 | Pharynx reversible|permanently inverted | Pharynx reversible | permanently inverted | ... |
| 11 | Pharynx reversible|completely reversible | Pharynx reversible | completely reversible | ... |

### 3.2 Compiling

The java source code provided is a Netbeans (https://netbeans.org) project. To compile the source code, download the Netbeans IDE, open the project file and build the project. The required library SSJ.jar can be downloaded from (https://github.com/umontreal-simul/ssj) but is also provided with the software release.

### 3.3 Citation

You can cite this software as:

# 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.