



FILE OPERATIONS

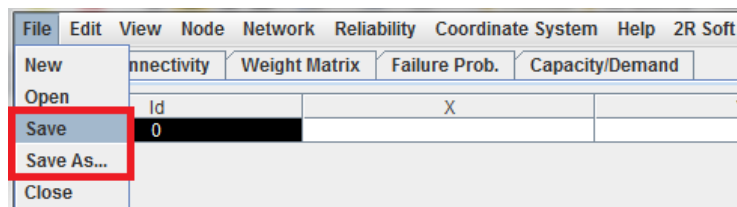
2R Net is capable of reading and writing **.2rn** files, which contain the complete description of a specific network model:

1. Coordinate System.
2. Coordinates and names of all the nodes.
3. Connectivity Matrix.
4. Weight Matrix.
5. Failure Probability Matrix.
6. Capacity and Demand Matrices.

These files are the means for 2R Net users to save their work, as well as the medium of distribution of network models that could be of interest to other 2R Net users.

SAVING NETWORK MODELS

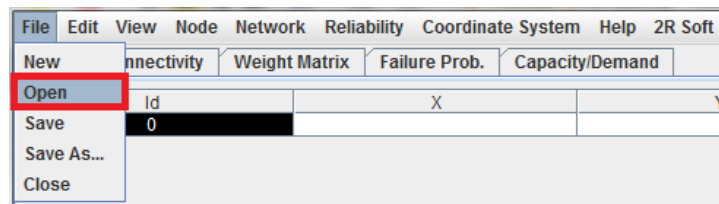
At any given time, a user can decide to save a network model's information for later use. In order to do this, the user must navigate through the **File** menu and select the **Save** or **Save As...** option:



The difference between **Save** and **Save As...** is that, while **Save** will only ask for the file's name and destination once and will then overwrite that same file on any subsequent uses, **Save As...** will ask for the file's name and destination every time it is invoked. Thus, **Save As...** is to be used whenever a user wants to save modifications made to a file without modifying the base file.

OPENING DATA MODELS

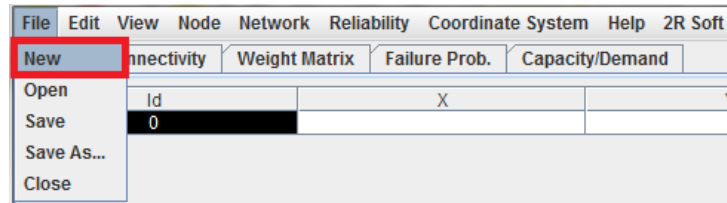
In order to load the information contained inside a **.2rn** file, the user must navigate through the **File** menu and select the **Open** option:



If no errors occur, the loaded model is shown in the main window (Main, Connectivity, Weight Matrix, etc).

STARTING NEW NETWORK MODELS

If a user is done working with a network model and wants to start a new one, the **New** option from the **File** menu provides this functionality:



Before the new model is created, the user is given the chance to save his current work.

DATA INPUT

INPUTTING AND MODIFYING DATA

2R Net offers a straight-forward editing interface in the main window for users to declare the network's nodes, connectivity matrix, weight matrix, etc. The following table summarizes the different actions that a user can carry out in 2R Net's editor:

Action	How To
Add nodes	<ol style="list-style-type: none">1. Select the Main tab's last cell (the one that is always blank).2. Write a coordinate or name to be added.3. Hit the ENTER key.4. Repeat steps 2 and 3 until all the nodes have been added.
Modify a specific cell of a matrix	There are two ways of going about this task: <ul style="list-style-type: none">• Double-click over the cell that contains the value to be edited, make the desired changes, and then hit the ENTER key.• To overwrite a value, select the appropriate cell (single click) and write the new value. Then, hit the ENTER key.
Delete a node	<ol style="list-style-type: none">1. In the Main tab, select a cell located on the row that describes the node that you want to delete.2. Hit the DEL or DELETE key. You can also go to the Edit menu and click on the Delete Selected Data Point (DEL) option.

NAVIGATION WITHIN A MATRIX

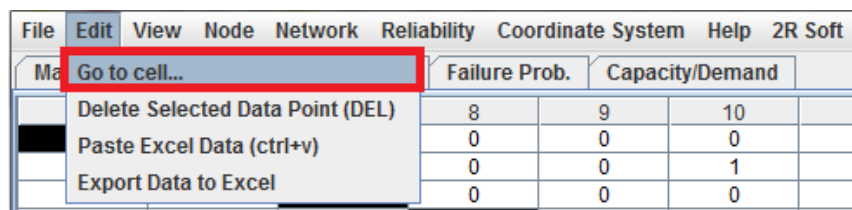
2R Net uses NxN matrices as input when dealing with N nodes. This makes navigation within each matrix (connectivity, weight, demand, etc) relatively difficult if the user limits itself to the arrow keys. For this reason, every matrix explicitly tells the user which cell is currently selected:

Main	Connectivity	Weight Matrix	Failure Prob.	Capacity/Demand			
5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0
0	0	0	0	1	0	1	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Current Cell: (20,7)

Cell (i, j) is that located at row i and column j, and represents a network property (weight, connectivity, etc) FROM i TO j.

If a user wants to jump to a specific cell in a matrix, the **Go to cell...** option in the **Edit** menu enables him to do so. After selecting that menu option, the user is prompted for the row and column of interest.



PASTING DATA FROM EXCEL

Given that Microsoft Excel is the de facto standard for spreadsheet management, 2R Net has been enhanced to allow an intuitive flow of information with that program. Therefore, users can copy and paste data from Excel into the 2R Net user interface with ease, both by using hotkeys and by using explicit menu options.

METHOD 1 - HOTKEYS

1. In Microsoft Excel, highlight the data values that you wish to paste into a 2R Net model:

	A	B	C	D
1	1	0	1	0
2	0	1	1	0
3	1	1	1	0
4	0	1	1	0
5	0	0	1	0
6				

2. Then, press the **CTRL** and the **C** keys **simultaneously** (ctrl+c).

3. In 2R Net's main window (Main, Connectivity, Weight Matrix, Failure Prob., or Capacity/Demand tab), select the cell that will act as the top-left cell of the imported data. **Warning:** the Excel data will overwrite data values of the current model if the position of the imported values coincides with the position of existing values.

Main	Connectivity	Weight Matrix	Failure Prob.	Capacity/Demand			
0	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0

4. Press the **CTRL** and the **V** keys **simultaneously** (ctrl+v).

5. The data is copied to 2R Net:

Main	Connectivity	Weight Matrix	Failure Prob.	Capacity/Demand				
0	1	2	3	4	5	6	7	
1	0	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	
0	0	1	0	0	0	0	0	
0	0	0	1	0	0	0	0	
0	0	0	0	1	0	0	0	
0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	
0	1	0	1	0	0	0	1	
0	0	1	1	0	0	0	0	
0	1	1	1	0	0	0	0	
0	0	1	1	0	0	0	0	
0	0	0	1	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	1	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	1	0	0	

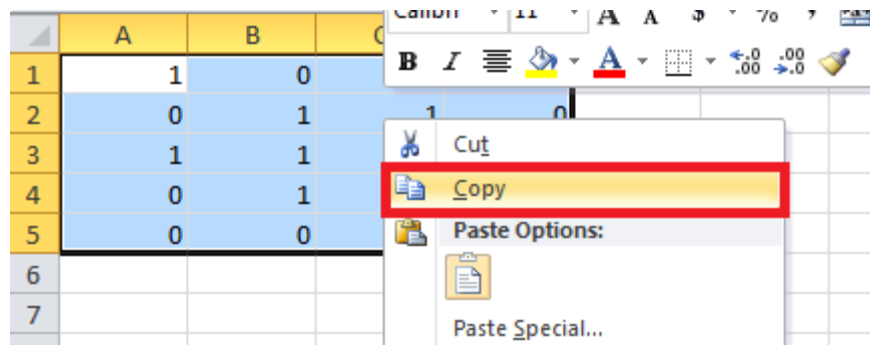
Note that, as the warning from step 3 suggests, some values from the matrix got overwritten by the Excel data.

METHOD 2 – EXPLICIT MENU OPTIONS

1. In Microsoft Excel, highlight the data values that you wish to paste into a 2R Data model:

	A	B	C	D
1	1	0	1	0
2	0	1	1	0
3	1	1	1	0
4	0	1	1	0
5	0	0	1	0
6				

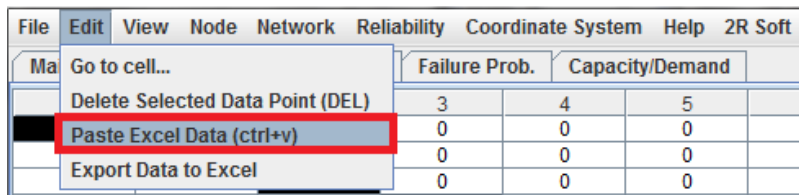
2. Right click over the selection and select the **Copy** option.



3. In 2R Net's main window (Main, Connectivity, Weight Matrix, Failure Prob., or Capacity/Demand tab), select the cell that will act as the top-left cell of the imported data. **Warning:** the Excel data will overwrite data values of the current model if the position of the imported values coincides with the position of existing values.

Main	Connectivity	Weight Matrix	Failure Prob.	Capacity/Demand				
0	1	2	3	4	5	6	7	
1	0	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	
0	0	1	0	0	0	0	0	
0	0	0	1	0	0	0	0	
0	0	0	0	1	0	0	0	
0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	1	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	1	0	0	0	0	0	
0	0	0	0	0	0	1	0	
0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0

5. Navigate through the **Edit** menu and click over the **Paste Excel Data** option.



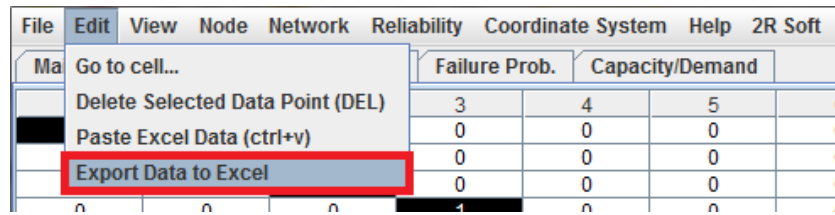
6. The data is copied to 2R Data:

Main	Connectivity	Weight Matrix	Failure Prob.	Capacity/Demand				
0	1	2	3	4	5	6	7	
1	0	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	
0	0	1	0	0	0	0	0	
0	0	0	1	0	0	0	0	
0	0	0	0	1	0	0	0	
0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	
0	1	0	1	0	0	0	1	
0	0	1	1	0	0	0	0	
0	1	1	1	0	0	0	0	
0	0	1	1	0	0	0	0	
0	0	0	1	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	1	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	1	0	0	

Note that, as the warning from step 3 suggests, some values from the matrix got overwritten by the Excel data.

EXPORTING DATA

Even though 2R Net provides a variety of ways to analyze a network, some users might want to carry out operations on a network model's information with other software packages. For that reason, 2R Net users are given the option to export a model's data values to XLSX (Excel 2007) format. The option to do this is located under the **Export** menu and is labeled as **Export Data to Excel**.



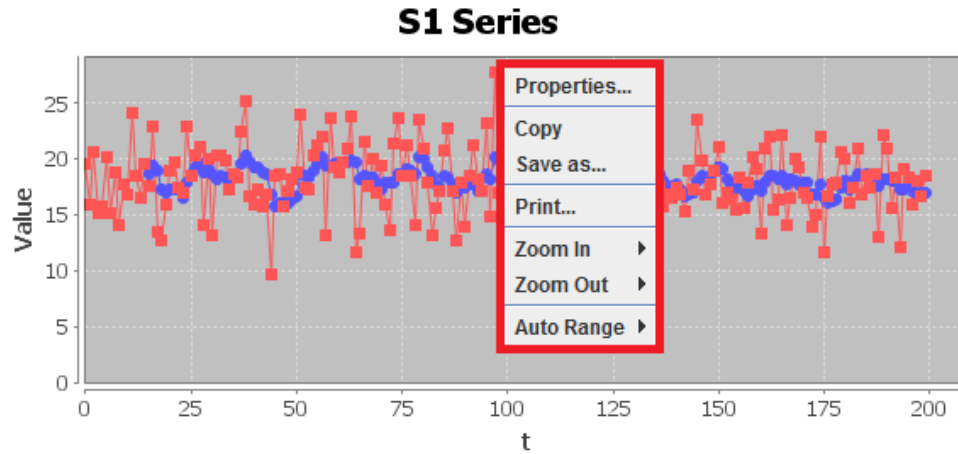
The resulting excel contains all of the model's matrices organized in separate tabs:

	A	B	C	D	E	F	G	H	I	J	K	L
1	0	81.47664	22.29157									
2	1	54.04542	73.30683									
3	2	87.16322	93.55599									
4	3	97.40707	85.19719									
5	4	18.73033	46.41067									
6	5	60.77468	11.87322									
7	6	54.2297	46.63394									
8	7	57.75111	15.91467									
9	8	0.862457	22.58624									
10	9	29.83575	5.45913									
11	10	20.98203	56.81245									
12	11	10.6021	61.31831									
13	12	40.88339	3.064481									
14	13	34.42469	87.91085									
15	14	30.28445	97.9241									
16	15	55.24904	28.9202									
17	16	74.48748	64.81982									
18	17	95.29292	50.01238									
19	18	60.9156	27.46962									
20	19	13.58691	92.99894									
21	20	67.05124	59.64461									
22	21	69.90636	48.50617									
23	22	67.76061	98.63706									
24	23	60.91699	77.51264									
25	24	76.57022	40.92959									
26	25	22.25023	36.04275									
27	26	97.66851	20.30045									

Data / Connectivity Matrix / Weight Matrix / Failure Probability / Capacity Matrix / Demand Matrix

GRAPHS

All of the graphs generated in 2R Soft provide a wide array of options in the form of a context menu. **The context menu appears when you right-click over a graph:**



PROPERTIES PANE

If you select the **Properties...** option, a properties pane appears. The properties pane lets you change the graph title, axis names, axis ranges, and font size.

Title **Plot** **Other**

XY Plot:

Domain Axis **Range Axis** **Appearance**

General:

Label:

Font: **Select...**

Paint: **Select...**

Other

Ticks **Range**

Auto-adjust range:

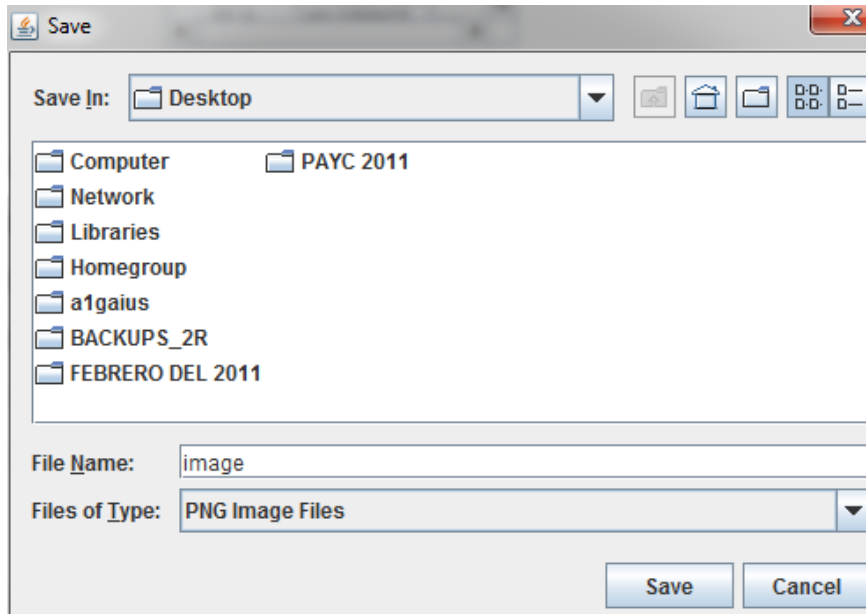
Minimum range value:

Maximum range value:

COPY AND SAVE AS...

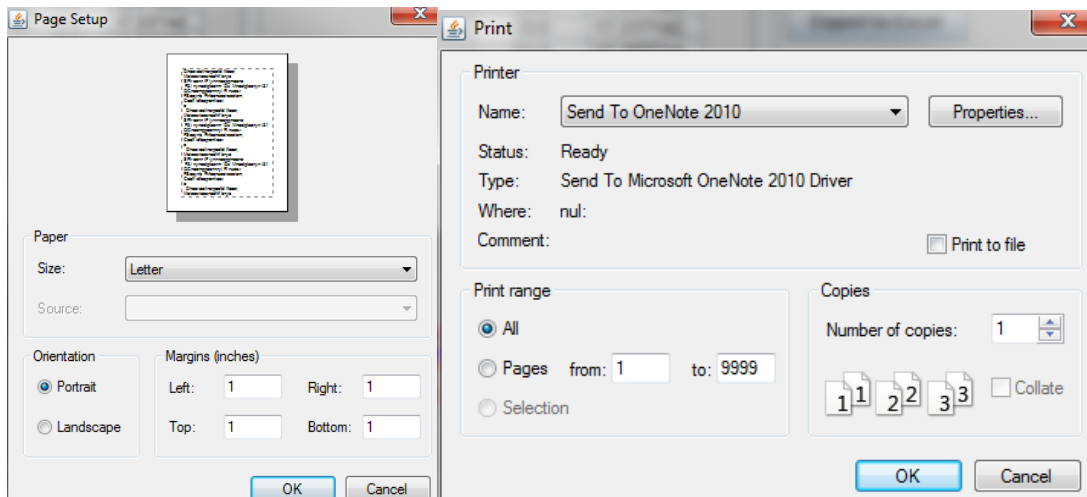
If you select **COPY**, the graph is copied to the system clipboard, so you can **PASTE** it anywhere else (Microsoft Word, Microsoft PowerPoint, etc).

Meanwhile, if **Save As...** is selected, 2R Soft will save the graph as a **PNG** image file in your hard disk after selecting the desired output folder and file name:



PRINT

The **Print** option does just that: it sends the graph to the printer of your choice (local or networked):

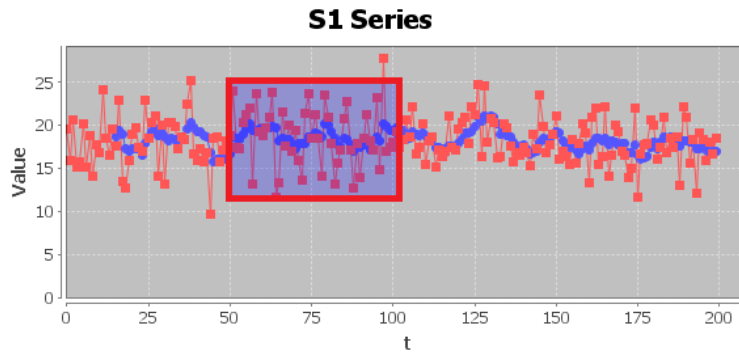


SCALE OPTIONS

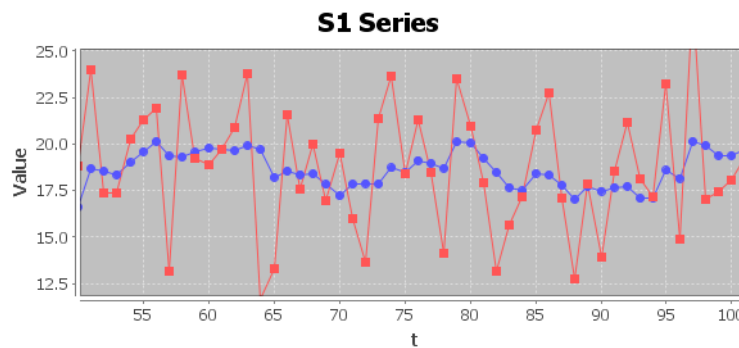
The **Auto Range**, **Zoom In**, and **Zoom Out** options are a quick way to inspect the graph. If a very specific range is needed for an axis, we highly recommend the [Properties Pane](#).

MANUAL ZOOM IN

For user convenience, all **2R Soft graphs support manual zoom in by regions**. If you're interested in a specific region, **hold your left-click and drag the mouse** to generate a highlighted box around that region:

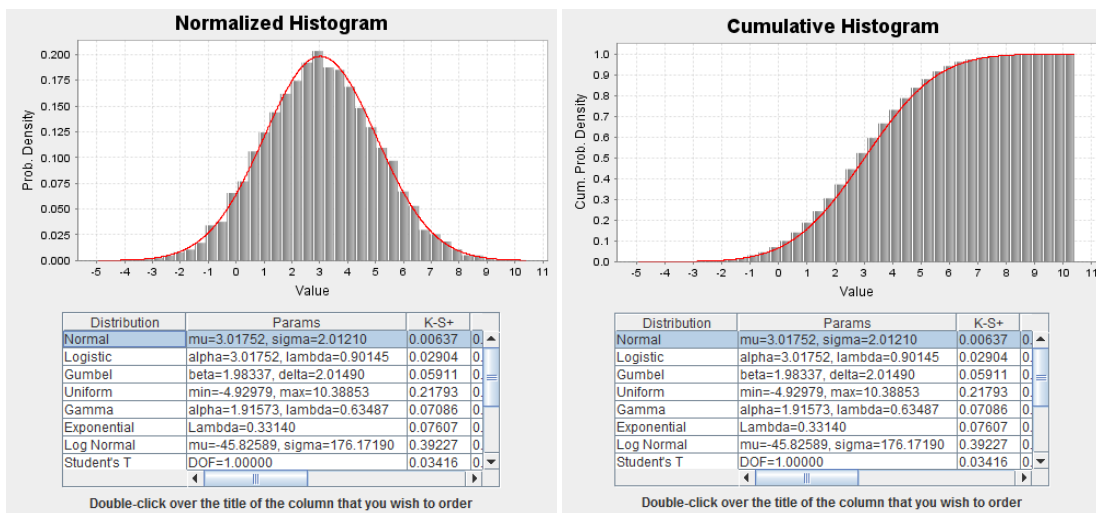


The end result:



NORMALIZED AND CUMULATIVE HISTOGRAMS

Normalized and Cumulative histograms are found all throughout 2R Soft. These two types of graphs are of great importance, considering that they show the stochastic tendencies of a data set. With them, a goodness-of-fit table is displayed with various probability distributions and the best estimates for their parameters, along with different goodness-of-fit tests:



The distribution selected in the goodness-of-fit table is juxtaposed with the histograms (red curve).

PROBABILITY DISTRIBUTION TYPES

When declaring a variable with a known distribution, the user can select one of many types of probability distributions.

Distribution	Description	Parameters
Beta	<p>The <i>beta</i> distribution has shape parameters $\alpha > 0$ and $\beta > 0$ over the interval (a, b), where $a < b$.</p> <p>It has density: $f(x) = (x - a)^{\alpha-1} (b - x)^{\beta-1} / [B(\alpha, \beta)(b - a)^{\alpha+\beta-1}]$ for $a < x < b$, and 0 elsewhere.</p> <p>It has the following distribution function: $F(x) = I_{a, \theta}(x) = \int_a^x (\xi - a)^{\alpha-1} (b - \xi)^{\beta-1} / [B(\alpha, \beta)(b - a)^{\alpha+\beta-1}] d\xi$, for $a < x < b$</p> <p>(Simard)</p>	<p>Alpha – shape parameter, alpha > 0 Beta – shape parameter, beta > 0 a – lower bound of the interval b – upper bound of the interval, b > a</p>
Binomial	<p>The binomial distribution with parameters n and p, where n is a positive integer and $0 \leq p \leq 1$. Its mass function is given by: $p(x) = nCr(n, x)p^x(1 - p)^{n-x} = n!/[x!(n - x)!] p^x(1 - p)^{n-x}$ for $x = 0, 1, 2, \dots, n$,</p> <p>and its distribution function is: $F(x) = \sum_{j=0}^x nCr(n, j) p^j(1 - p)^{n-j}$ for $x = 0, 1, 2, \dots, n$, where $nCr(n, x)$ is the number of possible combinations of x elements chosen among a set of n elements.</p> <p>(Simard)</p>	<p>p – probability of success on each trial ($0 \leq p \leq 1$) n – number of trials (integer), $n > 0$</p>
Chi Square	<p>The <i>chi-square</i> distribution with n degrees of freedom, where n is a positive integer. Its density is: $f(x) = x^{(n/2)-1} e^{-x/2} / (2^{n/2} \Gamma(n/2))$, for $x > 0$ where $\Gamma(x)$ is the gamma function. The <i>chi-square</i> distribution is a special case of the <i>gamma</i> distribution with shape parameter $n/2$ and scale parameter $1/2$.</p> <p>(Simard)</p>	<p>n – degrees of freedom (integer), $n > 0$</p>
Deterministic	<p>Distribution that represents a constant value, <i>val</i>. Consequently: $f(x) = \begin{cases} 1 & \text{if } x = val \\ 0 & \text{if } x \neq val \end{cases}$, $F(x) = \begin{cases} 1 & \text{if } x \geq val \\ 0 & \text{if } x < val \end{cases}$</p>	<p>Value – any real number</p>
Discrete Uniform	<p>The <i>discrete uniform</i> distribution over the integers in the range $[i, j]$. Its mass function is given by: $p(x) = 1/(j - i + 1)$ for $x = i, i + 1, \dots, j$ and 0 elsewhere.</p> <p>The distribution function is: $F(x) = (\text{floor}(x) - i + 1) / (j - i + 1)$ for $i \leq x \leq j$ and its inverse is: $F^{-1}(u) = i + (j - i + 1)u$ for $0 \leq u \leq 1$.</p> <p>(Simard)</p>	<p>Min. – lower bound (integer) Max. – upper bound (integer) (Max. > Min.)</p>
Exponential	<p>The <i>exponential</i> distribution with mean $1/\lambda$ where $\lambda > 0$. Its density is: $f(x) = \lambda e^{-\lambda x}$ for $x \geq 0$, its distribution function is: $F(x) = 1 - e^{-\lambda x}$, for $x \geq 0$, and its inverse distribution function is: $F^{-1}(u) = -\ln(1 - u)/\lambda$, for $0 < u < 1$</p> <p>(Simard)</p>	<p>Lambda – rate parameter, lambda > 0</p>
F-Distribution	<p>The Fisher F distribution with n and m degrees of freedom, where n and m are positive integers. Its density is: $f(x) = \Gamma((n + m)/2) n^{n/2} m^{m/2} / [\Gamma(n/2) \Gamma(m/2)] x^{(n-2)/2} / (m + nx)^{(n+m)/2}$, for $x > 0$. where $\Gamma(x)$ is the gamma function</p> <p>(Simard)</p>	<p>D.O.F. 1 – the n degrees of freedom (integer), D.O.F. 1 > 0 D.O.F. 2 – the m degrees of freedom (integer), D.O.F. 2 > 0</p>

Gamma	<p>The <i>gamma</i> distribution with shape parameter $\alpha > 0$ and scale parameter $\lambda > 0$. The density is: $f(x) = \lambda^\alpha x^{\alpha-1} e^{-\lambda x} / \Gamma(\alpha)$, for $x > 0$, where Γ is the gamma function, defined by: $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx$.</p> <p>In particular, $\Gamma(n) = (n - 1)!$ when n is a positive integer.</p> <p>(Simard)</p>	<p>Alpha – shape parameter, alpha > 0 Lambda – scale parameter, lambda > 0</p>
Geometric	<p>The <i>geometric</i> distribution with parameter p, where $0 < p < 1$. Its mass function is: $p(x) = p(1 - p)^x$, for $x = 0, 1, 2, \dots$ The distribution function is given by: $F(x) = 1 - (1 - p)^{x+1}$, for $x = 0, 1, 2, \dots$ and its inverse is: $F^{-1}(u) = \text{floor}(\ln(1 - u) / \ln(1 - p))$, for $0 \leq u < 1$</p> <p>(Simard)</p>	<p>p – probability of success on each trial ($0 < p < 1$)</p>
Gumbel	<p>The Gumbel distribution, with location parameter δ and scale parameter $\theta \neq 0$. Using the notation $z = (x - \delta) / \theta$, it has density: $f(x) = e^{-z} e^{-e^{-z}} / \theta$, for $-\infty < x < \infty$. and distribution function: $F(x) = e^{-e^{-z}}$, for $\theta > 0$ $F(x) = 1 - e^{-e^{-z}}$, for $\theta < 0$</p> <p>(Simard)</p>	<p>Beta – scale parameter, beta $\neq 0$ Delta – location parameter, any real number</p>
Hypergeometric	<p>The <i>hypergeometric</i> distribution with k elements chosen among l, m being of one type, and $l - m$ of the other. The parameters m, k and l are positive integers where $1 \leq m \leq l$ and $1 \leq k \leq l$. Its mass function is given by:</p> $p(x) = \frac{nCr(m, x)nCr(l - m, k - x)}{nCr(l, k)}$ <p>for $\max(0, k - l + m) \leq x \leq \min(k, m)$ where $nCr(n, x)$ is the number of possible combinations of x elements chosen among a set of n elements.</p> <p>(Simard)</p>	<p>m – number of elements of one type (integer), $m > 0$ l – total elements (integer), $l > 0$ k – number of elements chosen among l (integer), $k > 0$</p>
Logistic	<p>The <i>logistic</i> distribution. It has location parameter α and scale parameter $\lambda > 0$. The density is: $f(x) = (\lambda e^{-\lambda(x-\alpha)}) / ((1 + e^{-\lambda(x-\alpha)})^2)$ for $-\infty < x < \infty$. and the distribution function is: $F(x) = 1 / [1 + e^{-\lambda(x-\alpha)}]$ for $-\infty < x < \infty$.</p> <p>For $\lambda = 1$ and $\alpha = 0$, one can write: $F(x) = (1 + \tanh(x/2)) / 2$.</p> <p>The inverse distribution function is given by: $F^{-1}(u) = \ln(u / (1 - u)) / \lambda + \alpha$ for $0 \leq u < 1$</p> <p>(Simard)</p>	<p>Alpha – location parameter, any real number Lambda – scale parameter, lambda > 0</p>
Lognormal	<p>The <i>lognormal</i> distribution. It has scale parameter μ and shape parameter $\sigma > 0$. The density is: $f(x) = ((2\pi)^{-1/2} \sigma^{-1}) e^{-(\ln(x)-\mu)^2 / (2\sigma^2)}$ for $x > 0$, and 0 elsewhere.</p> <p>The distribution function is: $F(x) = \Phi((\ln(x)-\mu) / \sigma)$ for $x > 0$, where Φ is the standard normal distribution function.</p> <p>Its inverse is given by: $F^{-1}(u) = e^{\mu + \sigma \Phi^{-1}(u)}$ for $0 \leq u < 1$</p> <p>If $\ln(Y)$ has a <i>normal</i> distribution, then Y has a <i>lognormal</i> distribution with the same parameters.</p> <p>(Simard)</p>	<p>log mu – scale parameter, any real number log sigma – shape parameter, log sigma > 0</p>

Negative Binomial	<p>The negative binomial distribution with real parameters γ and p, where $\gamma > 0$ and $0 <= p <= 1$. Its mass function is: $p(x) = \Gamma(\gamma + x)/(x! \Gamma(\gamma))p^\gamma(1 - p)^x$, for $x = 0, 1, 2, \dots$ where Γ is the gamma function.</p> <p>If γ is an integer, $p(x)$ can be interpreted as the probability of having x failures before the γ-th success in a sequence of independent Bernoulli trials with probability of success p.</p>	<p>Gamma – number of failures until the experiment is stopped, Gamma > 0 p – success probability in each experiment, $0 \leq p \leq 1$</p>																		
(Simard)																				
Normal	<p>The normal distribution. It has mean μ and variance σ^2. Its density function is: $f(x) = e^{-((x-\mu)^2/(2\sigma^2))}/((2\pi)^{1/2}\sigma)$ for $-\infty < x < \infty$, where $\sigma > 0$.</p> <p>When $\mu = 0$ and $\sigma = 1$, we have the standard normal distribution, with corresponding distribution function: $F(x) = \Phi(x) = \int_{-\infty}^x e^{-t^2/2} dt/(2\pi)^{1/2}$ for $-\infty < x < \infty$.</p>	<p>Mean – self-explanatory, any real number Standard Deviation – self-explanatory, Std. Dev. > 0</p>																		
(Simard)																				
Pareto	<p>The Pareto family, with shape parameter $\alpha > 0$ and location parameter $\beta > 0$. The density for this type of Pareto distribution is: $f(x) = \alpha\beta^\alpha/x^{\alpha+1}$ for $x \geq \beta$, and 0 otherwise.</p> <p>The distribution function is: $F(x) = 1 - (\beta/x)^\alpha$ for $x \geq \beta$,</p> <p>and the inverse distribution function is: $F^{-1}(u) = \beta(1 - u)^{-1/\alpha}$ for $0 <= u < 1$</p>	<p>Alpha – shape parameter, alpha > 0 Beta – location parameter, beta > 0</p>																		
(Simard)																				
Poisson	<p>The Poisson distribution with mean $\lambda \geq 0$. The mass function is: $p(x) = e^{-\lambda}\lambda^x/(x!)$, for $x = 0, 1, \dots$ and the distribution function is: $F(x) = e^{-\lambda}\sum_{j=0}^x \lambda^j/(j!)$, for $x = 0, 1, \dots$</p>	<p>Lambda – mean, lambda ≥ 0</p>																		
(Simard)																				
Student's T	<p>The Student-t distribution with n degrees of freedom, where n is a positive integer. Its density is: $f(x) = [\Gamma((n+1)/2)/(\Gamma(n/2)(\pi n)^{1/2})][1 + x^2/n]^{-(n+1)/2}$ for $-\infty < x < \infty$, where $\Gamma(x)$ is the gamma function</p>	<p>D.O.F – degrees of freedom (integer), D.O.F > 0</p>																		
(Simard)																				
Triangular	<p>The triangular distribution with domain $[a, b]$ and mode (or shape parameter) m, where $a \leq m \leq b$. The density function is:</p> <table border="1" data-bbox="542 1226 984 1325"> <tbody> <tr> <td>$f(x) = 2(x - a)/[(b - a)(m - a)]$</td> <td>for $a \leq x \leq m$,</td> </tr> <tr> <td>$f(x) = 2(b - x)/[(b - a)(b - m)]$</td> <td>for $m \leq x \leq b$,</td> </tr> <tr> <td>$f(x) = 0$</td> <td>elsewhere,</td> </tr> </tbody> </table> <p>the distribution function is:</p> <table border="1" data-bbox="542 1377 984 1520"> <tbody> <tr> <td>$F(x) = 0$</td> <td>for $x < a$,</td> </tr> <tr> <td>$F(x) = (x - a)^2/[(b - a)(m - a)]$</td> <td>if $a \leq x \leq m$,</td> </tr> <tr> <td>$F(x) = 1 - (b - x)^2/[(b - a)(b - m)]$</td> <td>if $m \leq x \leq b$,</td> </tr> <tr> <td>$F(x) = 1$</td> <td>for $x > b$,</td> </tr> </tbody> </table> <p>and the inverse distribution function is given by:</p> <table border="1" data-bbox="480 1572 1045 1646"> <tbody> <tr> <td>$F^{-1}(u) = a + ((b - a)(m - a)u)^{1/2}$</td> <td>if $0 \leq u \leq (m - a)/(b - a)$,</td> </tr> <tr> <td>$F^{-1}(u) = b - ((b - a)(b - m)(1 - u))^{1/2}$</td> <td>if $(m - a)/(b - a) \leq u \leq 1$</td> </tr> </tbody> </table>	$f(x) = 2(x - a)/[(b - a)(m - a)]$	for $a \leq x \leq m$,	$f(x) = 2(b - x)/[(b - a)(b - m)]$	for $m \leq x \leq b$,	$f(x) = 0$	elsewhere,	$F(x) = 0$	for $x < a$,	$F(x) = (x - a)^2/[(b - a)(m - a)]$	if $a \leq x \leq m$,	$F(x) = 1 - (b - x)^2/[(b - a)(b - m)]$	if $m \leq x \leq b$,	$F(x) = 1$	for $x > b$,	$F^{-1}(u) = a + ((b - a)(m - a)u)^{1/2}$	if $0 \leq u \leq (m - a)/(b - a)$,	$F^{-1}(u) = b - ((b - a)(b - m)(1 - u))^{1/2}$	if $(m - a)/(b - a) \leq u \leq 1$	<p>a – lower bound of the domain, any real number b – upper bound of the domain, any real number mode – shape parameter, any real number</p> <p>"$a \leq mode \leq b$" must be satisfied</p>
$f(x) = 2(x - a)/[(b - a)(m - a)]$	for $a \leq x \leq m$,																			
$f(x) = 2(b - x)/[(b - a)(b - m)]$	for $m \leq x \leq b$,																			
$f(x) = 0$	elsewhere,																			
$F(x) = 0$	for $x < a$,																			
$F(x) = (x - a)^2/[(b - a)(m - a)]$	if $a \leq x \leq m$,																			
$F(x) = 1 - (b - x)^2/[(b - a)(b - m)]$	if $m \leq x \leq b$,																			
$F(x) = 1$	for $x > b$,																			
$F^{-1}(u) = a + ((b - a)(m - a)u)^{1/2}$	if $0 \leq u \leq (m - a)/(b - a)$,																			
$F^{-1}(u) = b - ((b - a)(b - m)(1 - u))^{1/2}$	if $(m - a)/(b - a) \leq u \leq 1$																			
(Simard)																				

Uniform	<p>The <i>uniform</i> distribution over the interval $[a, b]$. Its density is: $f(x) = 1/(b - a)$ for $a \leq x \leq b$, and 0 elsewhere.</p> <p>The distribution function is: $F(x) = (x - a)/(b - a)$ for $a \leq x \leq b$</p> <p>and its inverse is: $F^{-1}(u) = a + (b - a)u$ for $0 \leq u \leq 1$</p> <p>(Simard)</p>	<p>Min. – lower bound Max. – upper bound (Max. > Min.)</p>
Weibull	<p>The <i>Weibull</i> distribution with shape parameter $\alpha > 0$, location parameter δ, and scale parameter $\lambda > 0$. The density function is: $f(x) = \alpha \lambda^\alpha (x - \delta)^{\alpha-1} e^{-(\lambda(x-\delta))^\alpha}$ for $x > \delta$.</p> <p>the distribution function is: $F(x) = 1 - e^{-(\lambda(x-\delta))^\alpha}$ for $x > \delta$,</p> <p>and the inverse distribution function is: $F^{-1}(u) = (-\ln(1 - u))^{1/\alpha} / \lambda + \delta$ for $0 \leq u < 1$</p> <p>(Simard)</p>	<p>Alpha – shape parameter, alpha > 0 Lambda – scale parameter, lambda > 0 Delta – location parameter, any real number</p>

GOODNESS-OF-FIT (GOF) STATISTICS

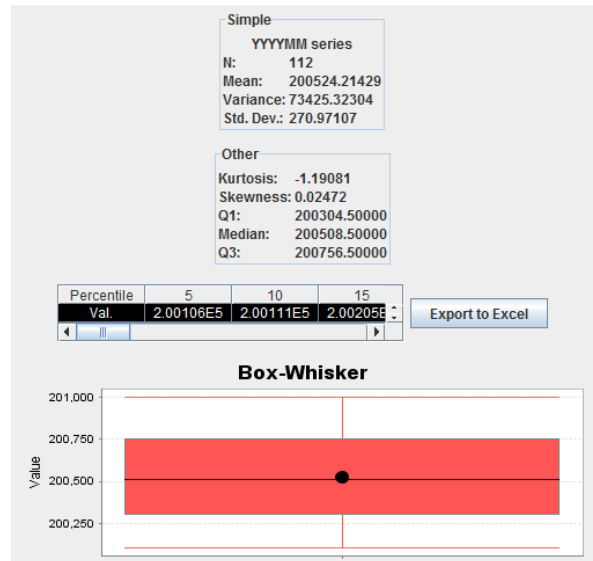
To decide whether to accept or reject a proposed probability distribution for the generated data, it is necessary to at least use one goodness-of-fit statistic as a criterion.

Col.	Test Type	Explanation	Critical Values																				
A-D	Anderson-Darling	<p>The Anderson-Darling test is defined as:</p> <p>H_0: The data follow a specified distribution. H_a: The data do not follow the specified distribution</p> <p>Test Statistic: The Anderson-Darling test statistic is defined as $A^2 = -N - S$</p> <p>where $S = \sum_{i=1}^N \frac{(2i-1)}{N} [\ln F(Y_i) + \ln(1 - F(Y_{N+1-i}))]$ F is the cumulative distribution function of the specified distribution. Note that the Y_i are the <i>ordered</i> data.</p> <p>The test is a one-sided test and the hypothesis that the distribution is of a specific form is rejected if the test statistic, A, is greater than the critical value.</p> <p>(SEMATECH2)</p>	<p>The critical values for the Anderson-Darling test are dependent on the specific distribution that is being tested. (SEMATECH2)</p> <p>For Normal and Lognormal distributions:</p> <table border="1"> <thead> <tr> <th>alpha</th> <th>0.1</th> <th>0.05</th> <th>0.025</th> <th>0.01</th> </tr> </thead> <tbody> <tr> <td>A^2_{crit}</td> <td>0.631</td> <td>0.752</td> <td>0.873</td> <td>1.035</td> </tr> </tbody> </table> <p>For Weibull and Gumbel distributions:</p> <table border="1"> <thead> <tr> <th>alpha</th> <th>0.1</th> <th>0.05</th> <th>0.025</th> <th>0.01</th> </tr> </thead> <tbody> <tr> <td>A^2_{crit}</td> <td>0.637</td> <td>0.757</td> <td>0.877</td> <td>1.038</td> </tr> </tbody> </table> <p>(Annis)</p>	alpha	0.1	0.05	0.025	0.01	A^2_{crit}	0.631	0.752	0.873	1.035	alpha	0.1	0.05	0.025	0.01	A^2_{crit}	0.637	0.757	0.877	1.038
alpha	0.1	0.05	0.025	0.01																			
A^2_{crit}	0.631	0.752	0.873	1.035																			
alpha	0.1	0.05	0.025	0.01																			
A^2_{crit}	0.637	0.757	0.877	1.038																			
K-S	Kolmogorov-Smirnov	<p>The Kolmogorov-Smirnov test is defined by:</p> <p>H_0: The data follow a specified distribution H_a: The data do not follow the specified distribution</p> <p>Test Statistic: The Kolmogorov-Smirnov test statistic is defined as $D = \max_{1 \leq i \leq N} \left(F(Y_i) - \frac{i-1}{N}, \frac{i}{N} - F(Y_i) \right)$ where F is the theoretical cumulative distribution of the distribution being tested which must be a continuous distribution (i.e., no discrete distributions such as the binomial or Poisson), and it must be fully specified.</p> <p>(SEMATECH1)</p>	<p>An attractive feature of this test is that the distribution of the K-S test statistic itself does not depend on the underlying cumulative distribution function being tested. Therefore, the critical values are universal. (SEMATECH1)</p> <p>For samples with more than 35 values: (ERI)</p> <table border="1"> <thead> <tr> <th>Significance</th> <th>0.20</th> <th>0.15</th> <th>0.10</th> <th>0.05</th> <th>0.01</th> </tr> </thead> <tbody> <tr> <td>Critical Value</td> <td>$\frac{1.07}{\sqrt{N}}$</td> <td>$\frac{1.14}{\sqrt{N}}$</td> <td>$\frac{1.22}{\sqrt{N}}$</td> <td>$\frac{1.36}{\sqrt{N}}$</td> <td>$\frac{1.63}{\sqrt{N}}$</td> </tr> </tbody> </table>	Significance	0.20	0.15	0.10	0.05	0.01	Critical Value	$\frac{1.07}{\sqrt{N}}$	$\frac{1.14}{\sqrt{N}}$	$\frac{1.22}{\sqrt{N}}$	$\frac{1.36}{\sqrt{N}}$	$\frac{1.63}{\sqrt{N}}$								
Significance	0.20	0.15	0.10	0.05	0.01																		
Critical Value	$\frac{1.07}{\sqrt{N}}$	$\frac{1.14}{\sqrt{N}}$	$\frac{1.22}{\sqrt{N}}$	$\frac{1.36}{\sqrt{N}}$	$\frac{1.63}{\sqrt{N}}$																		

K-S+ and K-S-	Kolmogorov-Smirnov+ and Kolmogorov-Smirnov-	<p>Given a sample of n independent uniforms U_i over $[0, 1]$, the <i>Kolmogorov-Smirnov+</i> statistic D_n^+ and the <i>Kolmogorov-Smirnov-</i> statistic D_n^- are defined by</p> $D_n^+ = \max_{1 \leq j \leq n} (j/n - U_{(j)}),$ $D_n^- = \max_{1 \leq j \leq n} (U_{(j)} - (j-1)/n),$ <p>where the $U_{(j)}$ are the U_i sorted in increasing order. Both statistics follows the same distribution function, i.e. $F_n(x) = P[D_n^+ \leq x] = P[D_n^- \leq x]$</p> <p>(Simard)</p>	The same from the K-S test.																																																												
CVM	Cramér-von Mises	<p>Given a sample of n independent uniforms U_i over $[0, 1]$, the Cramér-von Mises statistic W_n^2 is defined by $W_n^2 = 1/12n + \sum_{j=1}^n (U_{(j)} - (j-0.5)/n)^2$, where the $U_{(j)}$ are the U_i sorted in increasing order. The distribution function (the cumulative probabilities) is defined as $F_n(x) = P[W_n^2 \leq x]$</p> <p>(Simard)</p>	<p>As with the K-S test, the critical values are universal:</p> <table border="1"> <thead> <tr> <th colspan="6">Significance</th> </tr> <tr> <th>N</th> <th>0.20</th> <th>0.15</th> <th>0.10</th> <th>0.05</th> <th>0.01</th> </tr> </thead> <tbody> <tr> <td>2</td> <td>0.138</td> <td>0.149</td> <td>0.162</td> <td>0.175</td> <td>0.186</td> </tr> <tr> <td>10</td> <td>0.125</td> <td>0.142</td> <td>0.167</td> <td>0.212</td> <td>0.32</td> </tr> <tr> <td>20</td> <td>0.128</td> <td>0.146</td> <td>0.172</td> <td>0.217</td> <td>0.33</td> </tr> <tr> <td>30</td> <td>0.128</td> <td>0.146</td> <td>0.172</td> <td>0.218</td> <td>0.33</td> </tr> <tr> <td>60</td> <td>0.128</td> <td>0.147</td> <td>0.173</td> <td>0.220</td> <td>0.33</td> </tr> <tr> <td>100</td> <td>0.129</td> <td>0.147</td> <td>0.173</td> <td>0.220</td> <td>0.34</td> </tr> </tbody> </table> <p>(ReliaSoftCorp)</p>	Significance						N	0.20	0.15	0.10	0.05	0.01	2	0.138	0.149	0.162	0.175	0.186	10	0.125	0.142	0.167	0.212	0.32	20	0.128	0.146	0.172	0.217	0.33	30	0.128	0.146	0.172	0.218	0.33	60	0.128	0.147	0.173	0.220	0.33	100	0.129	0.147	0.173	0.220	0.34												
Significance																																																															
N	0.20	0.15	0.10	0.05	0.01																																																										
2	0.138	0.149	0.162	0.175	0.186																																																										
10	0.125	0.142	0.167	0.212	0.32																																																										
20	0.128	0.146	0.172	0.217	0.33																																																										
30	0.128	0.146	0.172	0.218	0.33																																																										
60	0.128	0.147	0.173	0.220	0.33																																																										
100	0.129	0.147	0.173	0.220	0.34																																																										
WG	Watson G	<p>Given a sample of n independent uniforms U_i over $[0, 1]$, the G statistic is defined by</p> $G_n = (n)^{1/2} \max_{1 \leq j \leq n} [j/n - U_{(j)} + \bar{u}_n - 1/2]$ $= (n)^{1/2} (D_n^+ + \bar{u}_n - 1/2),$ <p>where the $U_{(j)}$ are the U_i sorted in increasing order, \bar{u}_n is the average of the observations U_i, and D_n^+ is the Kolmogorov-Smirnov+ statistic. The distribution function (the cumulative probabilities) is defined as $F_n(x) = P[G_n \leq x]$</p> <p>(Simard)</p>	<p>From 2R Soft WG CDF code:</p> <table border="1"> <thead> <tr> <th colspan="6">Significance</th> </tr> <tr> <th>N</th> <th>0.20</th> <th>0.15</th> <th>0.10</th> <th>0.05</th> <th>0.01</th> </tr> </thead> <tbody> <tr> <td>2</td> <td>0.609</td> <td>0.636</td> <td>0.670</td> <td>0.718</td> <td>0.787</td> </tr> <tr> <td>10</td> <td>0.692</td> <td>0.728</td> <td>0.773</td> <td>0.843</td> <td>0.979</td> </tr> <tr> <td>20</td> <td>0.711</td> <td>0.747</td> <td>0.794</td> <td>0.866</td> <td>1.010</td> </tr> <tr> <td>30</td> <td>0.719</td> <td>0.755</td> <td>0.802</td> <td>0.875</td> <td>1.021</td> </tr> <tr> <td>60</td> <td>0.728</td> <td>0.765</td> <td>0.813</td> <td>0.887</td> <td>1.034</td> </tr> <tr> <td>100</td> <td>0.734</td> <td>0.770</td> <td>0.818</td> <td>0.893</td> <td>1.041</td> </tr> <tr> <td>1000</td> <td>0.745</td> <td>0.782</td> <td>0.831</td> <td>0.906</td> <td>1.055</td> </tr> <tr> <td>10000</td> <td>0.749</td> <td>0.786</td> <td>0.834</td> <td>0.909</td> <td>1.059</td> </tr> </tbody> </table>	Significance						N	0.20	0.15	0.10	0.05	0.01	2	0.609	0.636	0.670	0.718	0.787	10	0.692	0.728	0.773	0.843	0.979	20	0.711	0.747	0.794	0.866	1.010	30	0.719	0.755	0.802	0.875	1.021	60	0.728	0.765	0.813	0.887	1.034	100	0.734	0.770	0.818	0.893	1.041	1000	0.745	0.782	0.831	0.906	1.055	10000	0.749	0.786	0.834	0.909	1.059
Significance																																																															
N	0.20	0.15	0.10	0.05	0.01																																																										
2	0.609	0.636	0.670	0.718	0.787																																																										
10	0.692	0.728	0.773	0.843	0.979																																																										
20	0.711	0.747	0.794	0.866	1.010																																																										
30	0.719	0.755	0.802	0.875	1.021																																																										
60	0.728	0.765	0.813	0.887	1.034																																																										
100	0.734	0.770	0.818	0.893	1.041																																																										
1000	0.745	0.782	0.831	0.906	1.055																																																										
10000	0.749	0.786	0.834	0.909	1.059																																																										
WU	Watson U	<p>Given a sample of n independent uniforms u_i over $[0, 1]$, the Watson statistic U_n^2 is defined by $W_n^2 = 1/12n + \sum_{j=1}^n [u_{(j)} - (j-0.5)/n]^2$, $U_n^2 = W_n^2 - n(\bar{u}_n - 1/2)^2$, where the $u_{(j)}$ are the u_i sorted in increasing order, and \bar{u}_n is the average of the observations u_i. The distribution function (the cumulative probabilities) is defined as $F_n(x) = P[U_n^2 \leq x]$</p> <p>(Simard)</p>	<p>From 2R Soft WU CDF code:</p> <table border="1"> <thead> <tr> <th colspan="6">Significance</th> </tr> <tr> <th>N</th> <th>0.20</th> <th>0.15</th> <th>0.10</th> <th>0.05</th> <th>0.01</th> </tr> </thead> <tbody> <tr> <td>2</td> <td>0.122</td> <td>0.132</td> <td>0.143</td> <td>0.154</td> <td>0.164</td> </tr> <tr> <td>10</td> <td>0.116</td> <td>0.130</td> <td>0.150</td> <td>0.183</td> <td>0.255</td> </tr> <tr> <td>20</td> <td>0.116</td> <td>0.131</td> <td>0.151</td> <td>0.185</td> <td>0.262</td> </tr> <tr> <td>30</td> <td>0.117</td> <td>0.131</td> <td>0.151</td> <td>0.185</td> <td>0.264</td> </tr> <tr> <td>60</td> <td>0.117</td> <td>0.131</td> <td>0.151</td> <td>0.186</td> <td>0.266</td> </tr> <tr> <td>100</td> <td>0.117</td> <td>0.131</td> <td>0.152</td> <td>0.186</td> <td>0.267</td> </tr> <tr> <td>1000</td> <td>0.117</td> <td>0.131</td> <td>0.152</td> <td>0.187</td> <td>0.268</td> </tr> <tr> <td>10000</td> <td>0.117</td> <td>0.131</td> <td>0.152</td> <td>0.187</td> <td>0.268</td> </tr> </tbody> </table>	Significance						N	0.20	0.15	0.10	0.05	0.01	2	0.122	0.132	0.143	0.154	0.164	10	0.116	0.130	0.150	0.183	0.255	20	0.116	0.131	0.151	0.185	0.262	30	0.117	0.131	0.151	0.185	0.264	60	0.117	0.131	0.151	0.186	0.266	100	0.117	0.131	0.152	0.186	0.267	1000	0.117	0.131	0.152	0.187	0.268	10000	0.117	0.131	0.152	0.187	0.268
Significance																																																															
N	0.20	0.15	0.10	0.05	0.01																																																										
2	0.122	0.132	0.143	0.154	0.164																																																										
10	0.116	0.130	0.150	0.183	0.255																																																										
20	0.116	0.131	0.151	0.185	0.262																																																										
30	0.117	0.131	0.151	0.185	0.264																																																										
60	0.117	0.131	0.151	0.186	0.266																																																										
100	0.117	0.131	0.152	0.186	0.267																																																										
1000	0.117	0.131	0.152	0.187	0.268																																																										
10000	0.117	0.131	0.152	0.187	0.268																																																										

STATISTICS

When appropriate, 2R Soft calculates an array of statistics that can be both relevant and pertinent for users interested in analyzing the behavior of a data set.



The screenshot above is an example of what the user should expect to find in a **Statistics** tab. The following subsections of this document explain each of the statistics that are calculated by 2R Soft.

MEAN

The arithmetic mean of a set of values is the quantity commonly called "the" mean or the average. Given a set of samples $\{x_i\}$, the arithmetic mean is: (Weisstein2)

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

VARIANCE

For a series of data, the sample variance may be computed as: (Weisstein3)

$$s_N^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$$

where \bar{x} is the sample mean.

Note that the sample variance s_N^2 defined above is *not* an unbiased estimator for the population variance, σ^2 . In order to obtain an unbiased estimator for σ^2 , it is necessary to instead define a "bias-corrected sample variance": (Weisstein3)

$$s_{N-1}^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2$$

2R Soft uses the bias-corrected sample variance.

From the mathematical definition of variance, it is clear that this statistic expresses the dispersion of the data with respect to the mean. The larger the variance, the more spread out is the data.

STANDARD DEVIATION

The standard deviation formula is very simple: it is the square root of the variance. It is the most commonly used measure of spread (Lane2). Hence, an unbiased estimator of the population standard deviation, σ , is given by:

$$s_{N-1} = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

KURTOSIS

Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution. That is, data sets with high kurtosis tend to have a distinct peak near the mean, decline rather rapidly, and have heavy tails. Data sets with low kurtosis tend to have a flat top near the mean rather than a sharp peak. A uniform distribution would be the extreme case. For univariate data Y_1, Y_2, \dots, Y_N , the formula for kurtosis is: (SEMATECH3)

$$kurtosis = \frac{\sum_{i=1}^N (Y_i - \bar{Y})^4}{(N-1)s^4}$$

where \bar{Y} is the mean, s is the standard deviation, and N is the number of data points.

SKEWNESS

Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right of the center point. For univariate data Y_1, Y_2, \dots, Y_N , the formula for skewness is: (SEMATECH3)

$$skewness = \frac{\sum_{i=1}^N (Y_i - \bar{Y})^3}{(N-1)s^3}$$

where \bar{Y} is the mean, s is the standard deviation, and N is the number of data points. The skewness for a normal distribution is zero, and any symmetric data should have a skewness near zero. Negative values for the skewness indicate data that are skewed left and positive values for the skewness indicate data that are skewed right. By skewed left, we mean that the left tail is long relative to the right tail. Similarly, skewed right means that the right tail is long relative to the left tail. Some measurements have a lower bound and are skewed right. For example, in reliability studies, failure times cannot be negative. (SEMATECH3)

QUARTILES (Q1, MEDIAN, Q3)

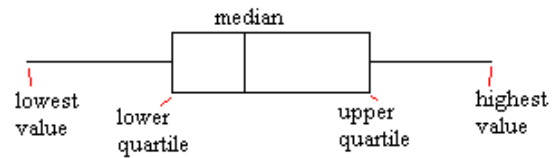
The quartiles of a data set are formed by the two boundaries on either side of the median, which divide the set into four equal sections. The lowest 25% of the data being found below the first quartile value, also called the lower quartile (Q1). The median, or second quartile divides the set into two equal sections. The lowest 75% of the data set should be found below the third quartile, also called the upper quartile (Q3). These three numbers are measures of the dispersion of the data, while the mean, median and mode are measures of central tendency. (EncyclopediaOfStatistics)

BOX-WHISKER DIAGRAM

Given some data, a **box and whisker diagram** (or box plot) can be drawn to show the spread of the data. The diagram shows the quartiles of the data, using these as an indication of the spread. (MathsRevision.net)

The diagram is made up of a "box", which lies between the upper and lower quartiles. The median can also be indicated by dividing the box into two. (MathsRevision.net)

The "whiskers" are straight line extending from the ends of the box to the maximum and minimum values: (MathsRevision.net)



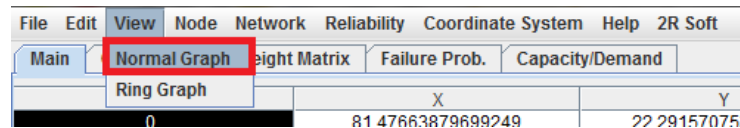
In 2R Soft, the Box-Whisker diagram also contains a black dot, which marks the arithmetic mean of the generated values.

NETWORK VIEWS

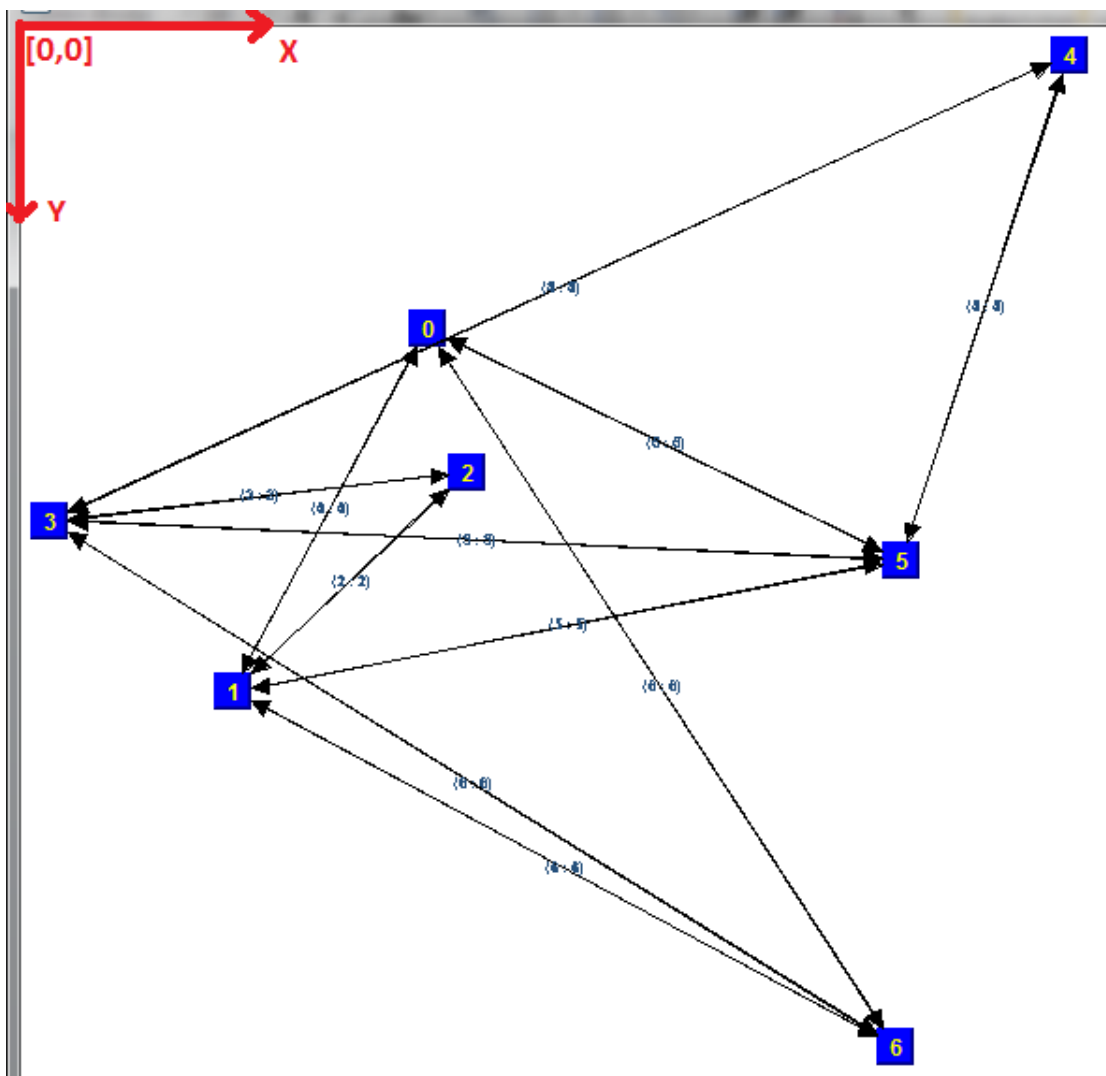
2R Net can graph the network (nodes and links) so that the users can visualize the system that is being analyzed.

NORMAL VIEW

Normal View is a graph of the network where the nodes are placed in a two-dimensional Euclidean Plane in accordance with their coordinates and the existing links are then shown with their respective direction. To obtain this type of graph, the user must navigate through the **View** menu and select the **Normal Graph** option:



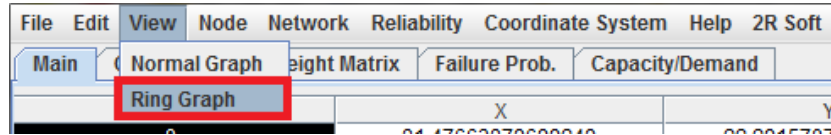
The result:



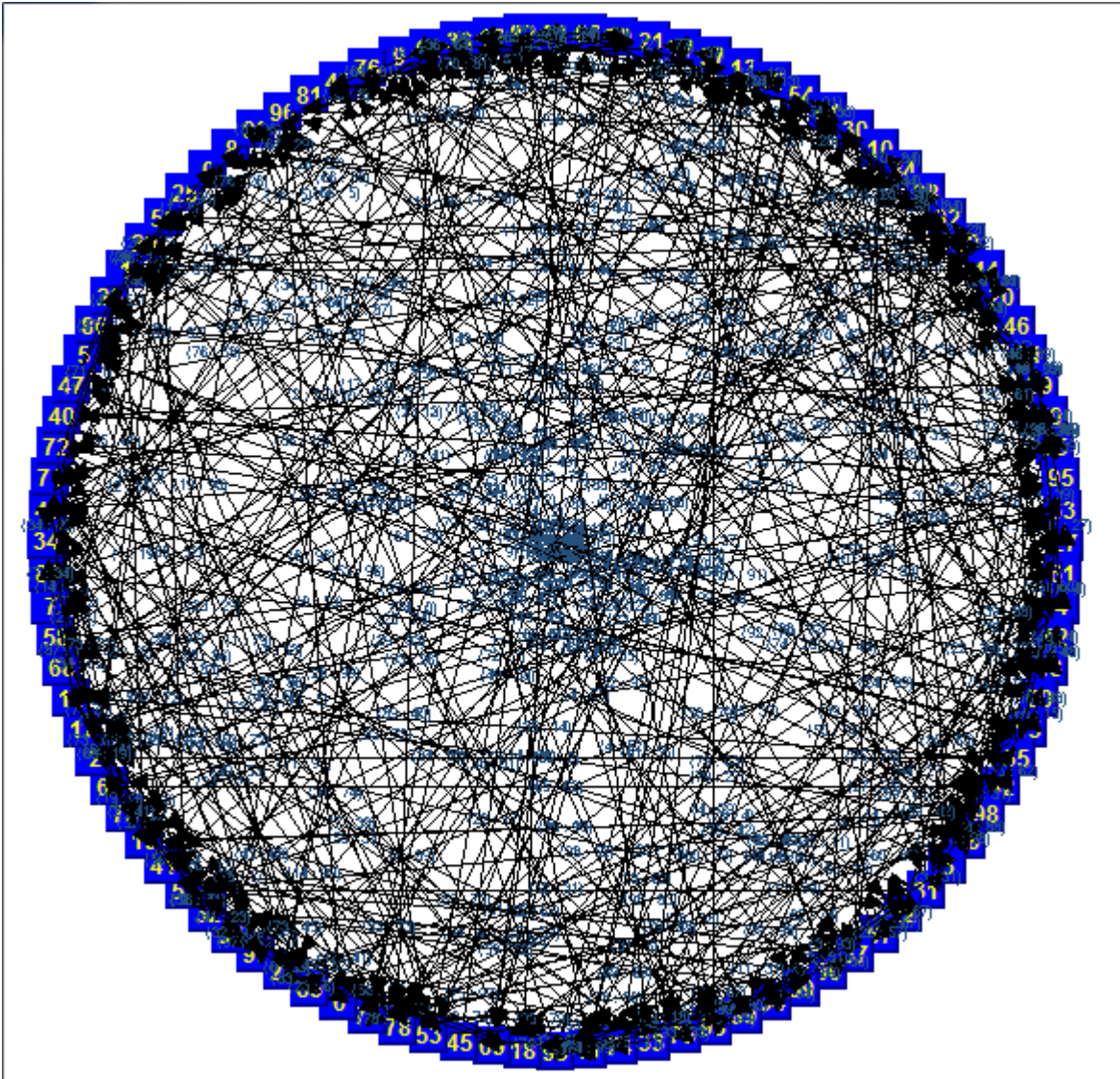
Please note that the origin (coordinate [0,0]) is located at the top left corner. The X coordinate increases to the right, while the Y coordinate increases downwards.

RING VIEW

Ring View is a graph of the network where the nodes are placed in a two-dimensional Euclidean Plane in a circular arrangement (ignoring their actual coordinates) and the existing links are then shown with their respective direction. Consequently, **the network's connection density can be witnessed**. To obtain this type of graph, the user must navigate through the **View** menu and select the **Ring Graph** option:



The result:



The network shown above is an example of high connection density, given that the center of the graph is occupied by a large number of links.

STATISTICS OF A NODE

THEORY

Network nodes (or graph vertices) have different metrics that can be of interest to determine their role and influence within a network.

DEGREE IN

The number of inward directed graph edges from a given graph vertex in a directed graph (Weisstein). That is, the amount of incoming links for a node.

DEGREE OUT

The number of outward directed graph edges from a given graph vertex in a directed graph (Weisstein). That is, the amount of outgoing links for a node.

DEGREE CENTRALITY

Degree is often interpreted in terms of the immediate risk of node for catching whatever is flowing through the network (such as a virus, or some information). For a graph $G = (V, E)$ with n vertices, the degree centrality $C_D(v)$ for vertex v is (Wikipedia 2011):

$$C_D(v) = \frac{\text{deg}(v)}{n - 1}$$

BETWEENNESS CENTRALITY

Vertices that occur on many shortest paths between other vertices have higher betweenness than those that do not.

For a graph $G = (V, E)$ with n vertices, the betweenness $C_B(v)$ for vertex v is computed as follows (Wikipedia 2011):

1. For each pair of vertices (s, t) , compute all shortest paths between them.
2. For each pair of vertices (s, t) , determine the fraction of shortest paths that pass through the vertex in question (here, vertex v).
3. Sum this fraction over all pairs of vertices (s, t) .

Or, more succinctly:

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

Where σ_{st} is the number of shortest paths from s to t , and $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass through a vertex v . This may be normalized by dividing through the number of pairs of vertices not including v , which is $(n-1)(n-2)$ for directed graphs. **2R Net uses the normalized measure of betweenness.**

CLOSENESS CENTRALITY

In the network theory, **closeness** is a sophisticated measure of centrality. It is defined as the mean geodesic distance (i.e., the shortest path) between a vertex v and all other vertices reachable from it (Wikipedia 2011):

$$\frac{\sum_{t \in V \setminus v} d_G(v, t)}{n - 1}$$

where $n \geq 2$ is the size of the network's 'connectivity component' V reachable from v . Closeness can be regarded as a measure of how long it will take information to spread from a given vertex to other reachable vertices in the network.

CLUSTERING COEFFICIENT

The **local clustering coefficient** of a vertex in a graph quantifies how close its neighbors are to being a clique (complete graph). Duncan J. Watts and Steven Strogatz introduced the measure in 1998 to determine whether a graph is a small-world network.

A graph $G = (V, E)$ formally consists of a set of vertices V and a set of edges E between them. An edge e_{ij} connects vertex i with vertex j .

The neighborhood N for a vertex v_i is defined as its immediately connected neighbors as follows:

$$N_i = \{v_j : e_{ij} \in E \wedge e_{ji} \in E\}.$$

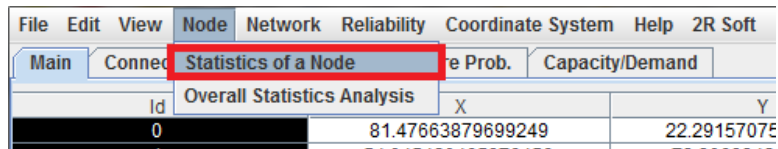
We define k_i as the number of vertices, $|N_i|$, in the neighborhood, N_i , of a vertex.

The local clustering coefficient C_i for a vertex v_i is then given by the proportion of links between the vertices within its neighborhood divided by the number of links that could possibly exist between them. For a directed graph, e_{ij} is distinct from e_{ji} , and therefore for each neighborhood N_i there are $k_i(k_i - 1)$ links that could exist among the vertices within the neighborhood (k_i is the total (in + out) degree of the vertex). Thus, the **local clustering coefficient for directed graphs** is given as (Wikipedia 2011):

$$C_i = \frac{|\{e_{jk}\}|}{k_i(k_i - 1)} : v_j, v_k \in N_i, e_{jk} \in E.$$

INPUT

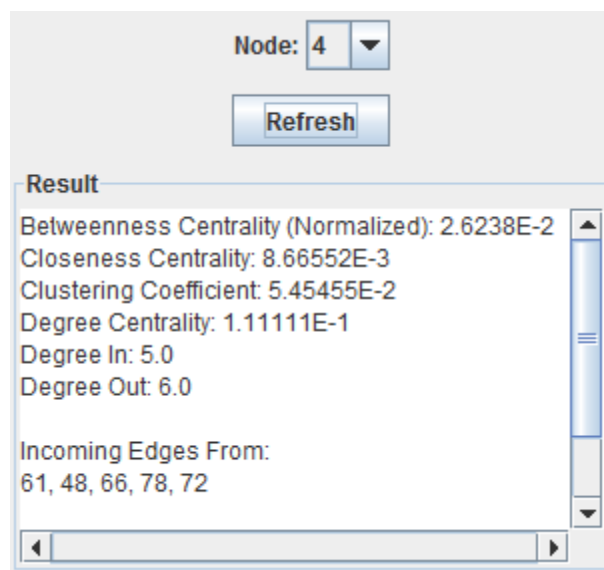
To query the statistics of a node, a user must navigate through the **Node** menu and select the **Statistics of a Node** option:



A new window appears with a drop-down menu containing all of the nodes in the network.

OUTPUT

After selecting the node of interest and pressing the **Refresh** button, the node's statistics are displayed:



Refer to the [Theory](#) section for a description of the statistics and how they are calculated. Below the main statistics, the nodes that are directly connected to the queried node in an incoming or outgoing direction are listed. **The Result pane has a scroll bar to the right-hand side that has to be used in order to view the entire output data.**

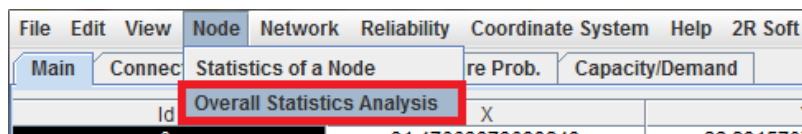
OVERALL STATISTICS ANALYSIS

THEORY

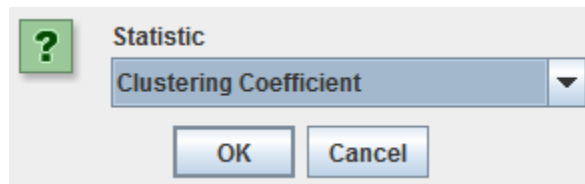
The [Statistics of a Node](#) option focuses on a single node in the network. Nonetheless, the behavior of a statistic, such as connectivity or clustering coefficient, throughout the entire network is also important. This is why an **Overall Statistics Analysis** option is provided. The same statistics described in the [previous Theory section](#) are the subject of study, but this time the objective is generalization rather than specificity.

INPUT

To analyze the tendency of a node statistic, a user must navigate through the **Node** menu and select the **Overall Statistics Analysis** option:

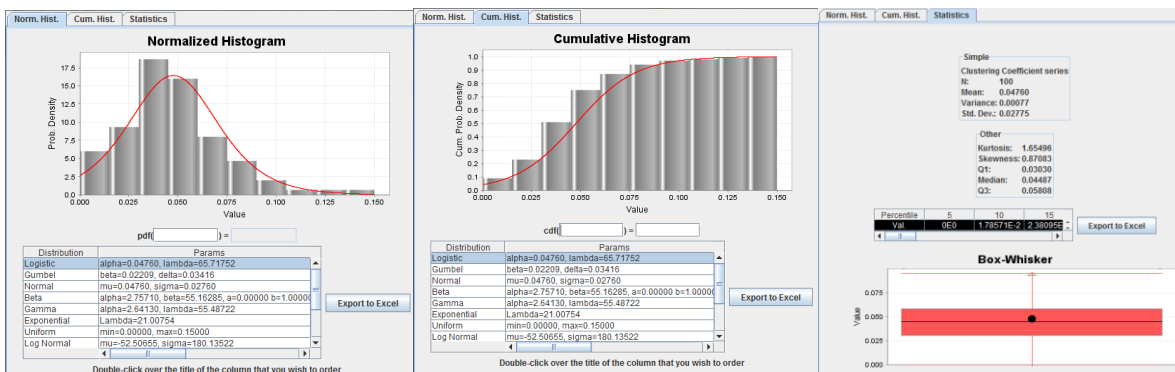


The user must then select the statistic to be analyzed and press the **OK** button:



OUTPUT

The output comes in the form of a Normalized Histogram and a Cumulative Histogram for the values of the statistic calculated on all of the network's nodes, along with an array of statistics regarding the calculated values:



To understand these results, refer to the [Normalized and Cumulative Histograms](#) section and the [Statistics](#) section of this document.

NETWORK-LEVEL ANALYSIS

ARE A AND B CONNECTED?

THEORY

When dealing with a large graph (or network), $G(V, E)$, with V being the set of vertices comprising the graph and E the set of edges connecting them, the question “Are A and B connected?” for a pair of vertices (or nodes) $[A, B]$ is of great importance. If the network is large enough, the computational time required to answer such question can be considerably large. While a [Shortest Path](#) algorithm can provide us with an answer to the question, it also requires an unnecessary amount of computation, since it will try to find the exact path that optimizes the displacement from A to B. A comparison of computational complexity is shown below:

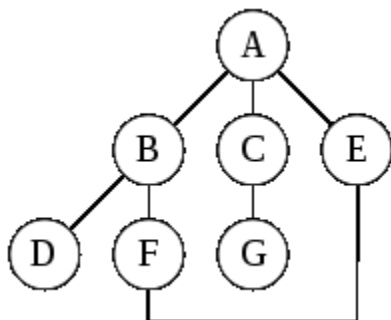
Algorithm	Type	Worst-Case Time Complexity
Dijkstra	Shortest Path	$O(V ^2)$
Bellman-Ford	Shortest Path	$O(V \times E)$
Depth-First	Connectivity Tree	$O(V + E)$
Breadth-First	Connectivity Tree	$O(V + E)$

As the table above shows, if one is not interested on the path from A to B, but only on knowing whether they are connected or not, the Depth-First and Breadth-First algorithms are faster at providing an answer.

DEPTH-FIRST SEARCH

Depth-first search (DFS) is an algorithm for traversing or searching a tree, tree structure, or graph. One starts at the root (selecting some node as the root in the graph case) and explores as far as possible along each branch before backtracking (FilePie 2011).

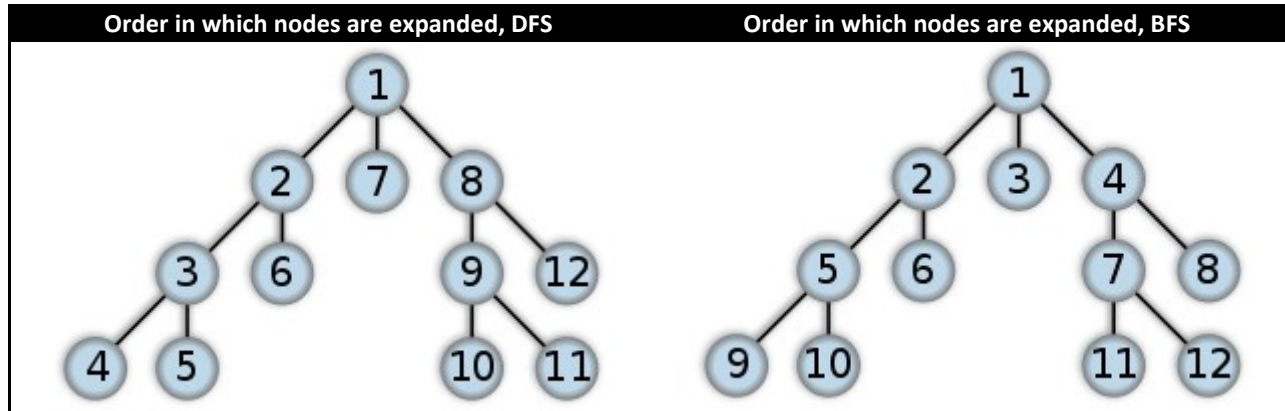
For the following graph:



a depth-first search starting at A, assuming that the left edges in the shown graph are chosen before right edges, and assuming the search remembers previously-visited nodes and will not repeat them (since this is a small graph), will visit the nodes in the following order: A, B, D, F, E, C, G.

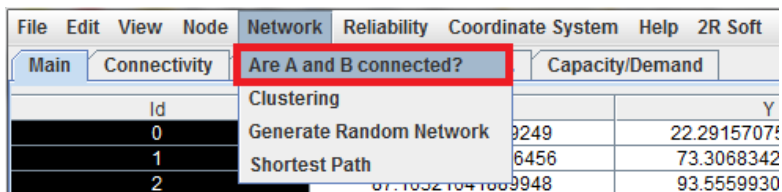
BREADTH-FIRST SEARCH

In graph theory, **breadth-first search (BFS)** is a graph search algorithm that begins at the root node and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal. This results in a different order of traversal with respect to the previous algorithm (FilePie 2011):

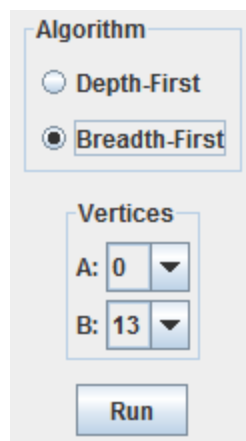


INPUT

To query the connectivity between two nodes, a user must navigate through the **Network** menu and select the **Are A and B Connected?** option:



A new dialog window comes up asking for the search algorithm to be employed and the pair of nodes to test:



Refer to the [Theory](#) section to understand the differences between Depth-First and Breadth-First searches.

OUTPUT

After clicking the **Run** button, 2R Net shows a message indicating whether or not the two nodes are connected:

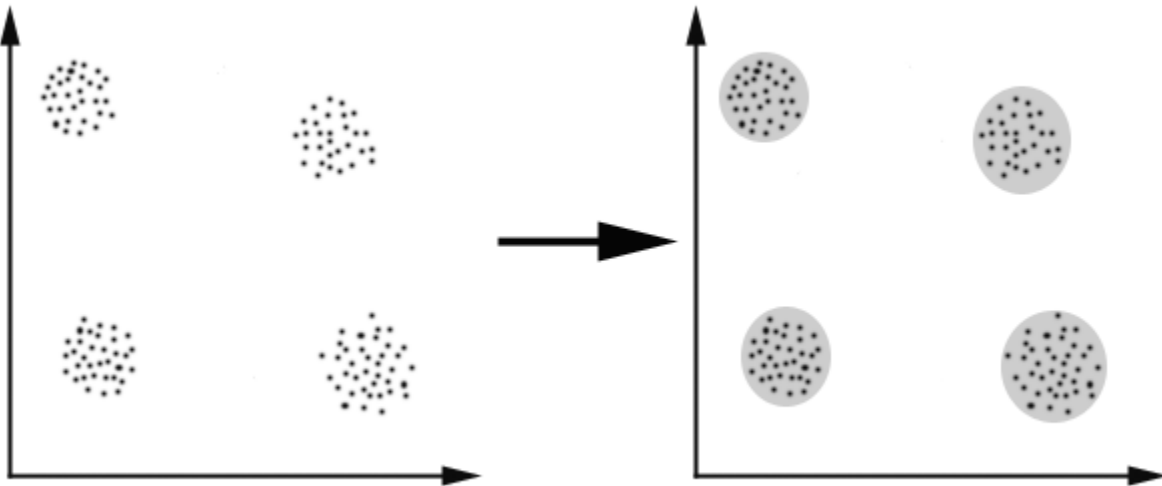
Positive Answer	Negative Answer
<p data-bbox="391 302 505 323">Algorithm</p> <p data-bbox="399 352 561 380"><input type="radio"/> Depth-First</p> <p data-bbox="399 409 586 436"><input checked="" type="radio"/> Breadth-First</p> <p data-bbox="435 485 537 506">Vertices</p> <p data-bbox="435 533 537 560">A: 0 ▼</p> <p data-bbox="435 590 537 617">B: 13 ▼</p> <p data-bbox="467 674 505 701">Run</p> <p data-bbox="313 737 678 764">A and B ARE INDEED connected</p>	<p data-bbox="1029 302 1143 323">Algorithm</p> <p data-bbox="1037 352 1200 380"><input checked="" type="radio"/> Depth-First</p> <p data-bbox="1037 409 1224 436"><input type="radio"/> Breadth-First</p> <p data-bbox="1073 485 1175 506">Vertices</p> <p data-bbox="1073 533 1175 560">A: 0 ▼</p> <p data-bbox="1073 590 1175 617">B: 5 ▼</p> <p data-bbox="1105 674 1143 701">Run</p> <p data-bbox="967 737 1333 764">A and B ARE NOT connected</p>

Of course, the Depth-First and Breadth-First algorithms provide consistent answers. The means of getting to the answers differ, but the results are always the same.

CLUSTERING ANALYSIS

In the words of (Dipartimento Di Elettronica E Informazione):

Clustering can be considered the most important *unsupervised learning* problem; so, as every other problem of this kind, it deals with finding a *structure* in a collection of unlabeled data. A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”. A *cluster* is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters. We can show this with a simple graphical example:



INTRODUCTION TO CLUSTERING ALGORITHMS

Clustering algorithms can be classified into 4 different groups (Dipartimento Di Elettronica E Informazione):

1. **Exclusive clustering:** if a certain datum belongs to a definite cluster then it could not be included in another cluster.
2. **Overlapping clustering:** uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership.
3. **Hierarchical clustering:** based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster. After a few iterations it reaches the final clusters wanted.
4. **Probabilistic clustering:** a completely probabilistic approach.

An important step in any clustering is to select a **distance measure**, which will determine how the *similarity of two elements is calculated*. This will influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another (Temporis). Common distance measures include the Euclidean distance, Manhattan distance, Mahalanobis distance, and Hamming distance. **In the case of 2R Net, LINK WEIGHT is always used as the distance measure during clustering analysis. Refer to the [Weight Matrix](#) section for more information on the way link weight is calculated.**

EDGE BETWEENNESS

THEORY

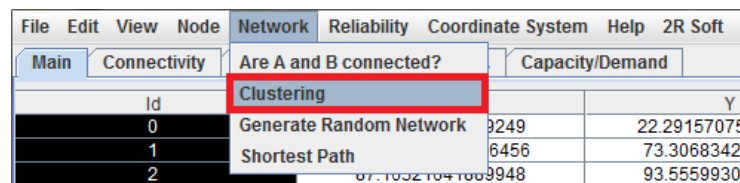
An algorithm for computing clusters (community structure) in graphs based on edge betweenness. The betweenness of an edge is defined as the extent to which that edge lies along shortest paths between all pairs of nodes. This algorithm works by iteratively following the 2 step process (GIRVAN, #160 et al. 2002):

- Compute edge betweenness for all edges in current graph.
- Remove edge with highest betweenness

Running time is: $O(kmn)$ where k is the number of edges to remove, m is the total number of edges, and n is the total number of vertices. For very sparse graphs the running time is closer to $O(kn^2)$ and for graphs with strong community structure, the complexity is even lower.

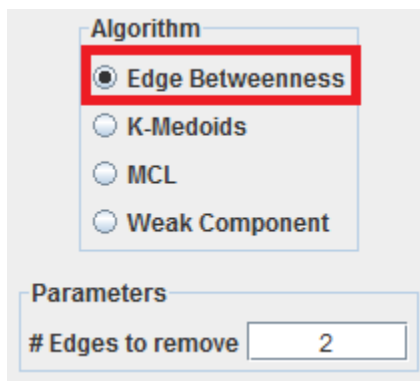
INPUT

To begin a clustering analysis in 2R Net, the user must navigate through the **Network** menu and select the **Clustering** option:



File	Edit	View	Node	Network	Reliability	Coordinate System	Help	2R Soft
Main		Connectivity		Are A and B connected?		Capacity/Demand		
		Id		Clustering		Y		
		0		Generate Random Network		9249		22.29157075
		1		Shortest Path		6456		73.3068342
		2				9948		93.5559930

After selecting **Edge Betweenness** as the clustering algorithm of choice, any relevant parameters are requested:



Algorithm

Edge Betweenness

K-Medoids

MCL

Weak Component

Parameters

Edges to remove

- **# Edges to remove:** the number of edges to be progressively removed from the graph. Refer to the [Theory](#) section to understand the effects of this parameter.

OUTPUT

Jump to the [Clustering Output](#) section to learn about the results obtained after this type of analysis is run.

K-MEDOIDS CLUSTERING

THEORY

K-Medoids is an exclusive clustering algorithm for partitioning \mathbf{N} data points into \mathbf{K} disjoint subsets \mathbf{S}_j containing \mathbf{N}_j data points so as to minimize the sum-of-squares criterion (Weisstein):

$$J = \sum_{j=1}^K \sum_{n \in S_j} |x_n - \mu_j|^2$$

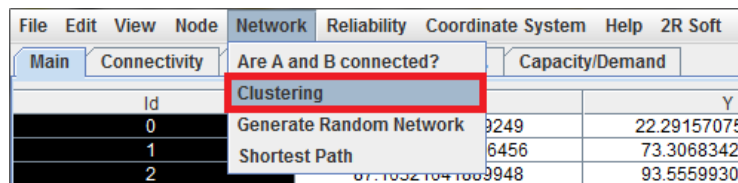
Where x_n is a vector representing the n th data point and μ_j is the geometric centroid (average of all coordinates) of the data points in \mathbf{S}_j .

The algorithm consists on a simple re-estimation procedure as follows (Weisstein). Initially, the data points are assigned at random to the \mathbf{K} sets.

- For step 1, the centroid (average of all instances in the set) is computed for each set (or cluster) and the instance closest to such mean is taken as the cluster's medoid. (Abeel)
- In step 2, every point is re-assigned to the cluster whose medoid is closest to that point.
- These two steps are alternated until a stopping criterion is met, i.e., when there is no further change in the assignment of the data points.

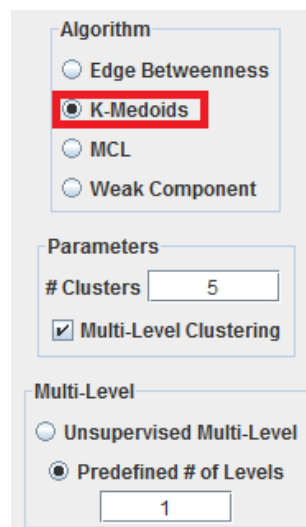
INPUT

To begin a clustering analysis in 2R Net, the user must navigate through the **Network** menu and select the **Clustering** option:



File	Edit	View	Node	Network	Reliability	Coordinate System	Help	2R Soft
Main	Connectivity	Are A and B connected?	Capacity/Demand	Clustering				
Id								Y
0		Generate Random Network	9249					22.29157075
1		Shortest Path	6456					73.3068342
2			07.10321041009948					93.5559930

After selecting **K-Medoids** as the clustering algorithm of choice, any relevant parameters are requested:



Algorithm

Edge Betweenness

K-Medoids

MCL

Weak Component

Parameters

Clusters

Multi-Level Clustering

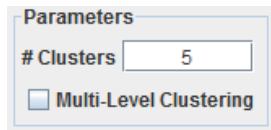
Multi-Level

Unsupervised Multi-Level

Predefined # of Levels

PARAMETERS

When running a one-level clustering analysis, meaning that clusters are only calculated from the original network and no sub-clusters within those clusters are to be calculated, a single parameter has to be entered:

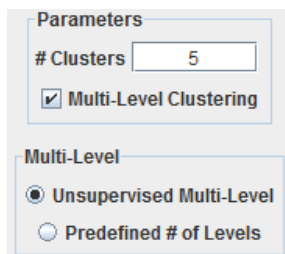


The screenshot shows a window titled "Parameters". It contains a text input field labeled "# Clusters" with the value "5" entered. Below it is a checkbox labeled "Multi-Level Clustering" which is currently unchecked.

- **# Clusters:** the number of clusters that the user wants to obtain from the clustering process.

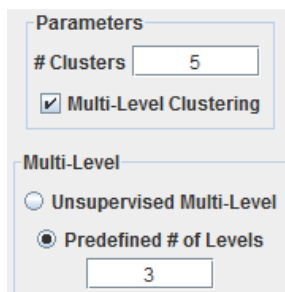
MULTI-LEVEL CLUSTERING

If sub-clusters within clusters are to be calculated, the user will have to activate the **Multi-Level Clustering** checkbox. This will require additional information to be entered in the **Multi-Level** region of the clustering window:



The screenshot shows two windows. The top window, titled "Parameters", has "# Clusters" set to "5" and the "Multi-Level Clustering" checkbox checked. The bottom window, titled "Multi-Level", has two radio button options: "Unsupervised Multi-Level" (which is selected) and "Predefined # of Levels".

- **Unsupervised Multi-Level:** for each calculated cluster, sub-clusters will be found until a point is reached where the sub-sub-...-sub-cluster cannot be divided into **#Clusters** clusters (5 in the screen above). Therefore, for the example above, a tree of clusters is obtained in which each "node" (cluster) has either 0 or 5 "nodes" (clusters) as childs.



The screenshot shows two windows. The top window, titled "Parameters", has "# Clusters" set to "5" and the "Multi-Level Clustering" checkbox checked. The bottom window, titled "Multi-Level", has two radio button options: "Unsupervised Multi-Level" and "Predefined # of Levels" (which is selected). Below the selected option is a text input field containing the value "3".

- **Predefined # of Levels:** this type of multi-level clustering stops after **N** levels of clusters are found (limited depth of search). For each calculated cluster, sub-clusters will be found until a point is reached where the sub-sub-...-sub-cluster cannot be divided into **#Clusters** clusters (5 in the screen above) **or until N levels of clusters have already been calculated**. Therefore, for the example above, a tree of clusters is obtained in which each "node" (cluster) has either 0 or 5 "nodes" (clusters) as childs, **and the maximum depth of search is of sub-sub-clusters with respect to the original clusters**.

OUTPUT

Jump to the [Clustering Output](#) section to learn about the results obtained after this type of analysis is run.

MCL (MARKOV CLUSTER ALGORITHM)

THEORY

In the words of (JUNG 2011):

The basic idea underlying the MCL algorithm is that dense regions in sparse graphs correspond with regions in which the number of k -length paths is relatively large, for small k in N , which corresponds to multiplying probabilities in the matrix appropriately. Random walks of length k have higher probability (product) for paths with beginning and ending in the same dense region than for other paths (JUNG 2011).

The algorithm starts by creating a Markov matrix from the graph, for which first the adjacency matrix is added diagonal elements to include self-loops for all nodes, i.e., probabilities that the random walker stays at a particular node. After this initialization, the algorithm works by alternating two operations, expansion and inflation, iteratively recomputing the set of transition probabilities. The expansion step corresponds to matrix multiplication (on stochastic matrices), the inflation step corresponds with a parameterized inflation operator Γ_r , which acts column-wise on (column) stochastic matrices.

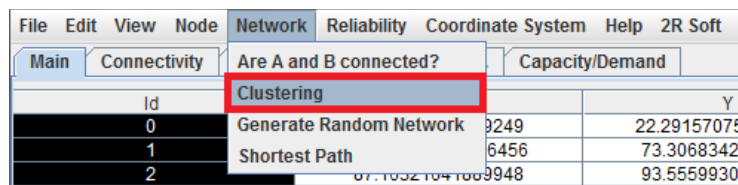
The inflation operator transforms a stochastic matrix into another one by raising each element to a positive power p and re-normalizing columns to keep the matrix stochastic. The effect is that larger probabilities in each column are emphasized and smaller ones deemphasized. On the other side, the matrix multiplication in the expansion step creates new non-zero elements, i.e., edges. The algorithm converges very fast, and the result is an idempotent Markov matrix, $M = M * M$, which represents a hard clustering of the graph into components.

Expansion and inflation have two opposing effects: While expansion flattens the stochastic distributions in the columns and thus causes paths of a random walker to become more evenly spread, inflation contracts them to favored paths.

This description is based on the introduction of Stijn van Dongen's thesis *Graph Clustering by Flow Simulation* (2000); for a mathematical treatment of the algorithm and the associated MCL process, see there.

INPUT

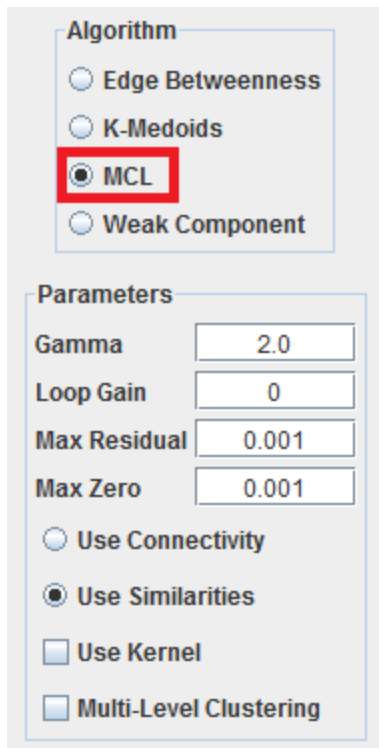
To begin a clustering analysis in 2R Net, the user must navigate through the **Network** menu and select the **Clustering** option:



The screenshot shows the 2R Net software interface. The 'Network' menu is open, and the 'Clustering' option is highlighted with a red box. The menu items are: 'Main', 'Connectivity', 'Are A and B connected?', 'Capacity/Demand', 'Clustering', 'Generate Random Network', 'Shortest Path', and '07.10.02.104.1009948'. The 'Clustering' option is the first item in the list below 'Capacity/Demand'.

File	Edit	View	Node	Network	Reliability	Coordinate System	Help	2R Soft
				Main	Connectivity	Are A and B connected?		Capacity/Demand
				Id	Clustering			Y
				0	Generate Random Network	9249		22.29157075
				1	Shortest Path	6456		73.3068342
				2		07.10.02.104.1009948		93.5559930

After selecting **MCL** as the clustering algorithm of choice, any relevant parameters are requested:



The screenshot shows a configuration window for the MCL algorithm. It is divided into two main sections: 'Algorithm' and 'Parameters'. In the 'Algorithm' section, four radio buttons are present: 'Edge Betweenness', 'K-Medoids', 'MCL' (which is selected and highlighted with a red rectangular box), and 'Weak Component'. The 'Parameters' section contains four input fields: 'Gamma' with the value 2.0, 'Loop Gain' with the value 0, 'Max Residual' with the value 0.001, and 'Max Zero' with the value 0.001. Below the input fields are five checkboxes: 'Use Connectivity' (unchecked), 'Use Similarities' (checked), 'Use Kernel' (unchecked), and 'Multi-Level Clustering' (unchecked).

PARAMETERS

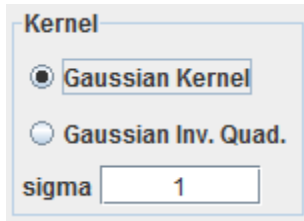
When running a one-level clustering analysis, meaning that clusters are only calculated from the original network and no sub-clusters within those clusters are to be calculated, some parameters have to be entered:

- **Gamma:** inflation exponent for Gamma operator.
- **Loop Gain:** values for cycles.
- **Max Residual:** maximum difference between row elements and row square sum (measure of idempotence).
- **Max Zero:** maximum value considered zero for pruning operations.
- **Affinity Matrix**
 - **Use Connectivity:** the [Connectivity Matrix](#) will be used as the MCL Affinity Matrix.
 - **Use Similarities:** the [Weight Matrix](#) will be used to **generate** the MCL Affinity Matrix. Since a higher link weight indicates a lower affinity, cell [i, j] of the affinity matrix will be taken to be equal of $1/w_{i,j}$, where $w_{i,j}$ is the weight of the link from node i to node j.

To fully understand the meaning of these parameters, it is necessary to go over Stijn van Dongen's thesis *Graph Clustering by Flow Simulation* (2000). If a basic usage of MCL is needed, the default values of these parameters (shown in the screenshot above) should be fine.

USE KERNEL

If the user wants to apply a kernel to the MCL Affinity Matrix, the **Use Kernel** checkbox should be selected. This leads to the **Kernel** region being displayed:



Kernel

Gaussian Kernel

Gaussian Inv. Quad.

sigma

Kernels are applied to enhance the differences between affinity values, so that the affinity matrix becomes less vague:

With Gaussian Kernel:

$$A_{ij} = \begin{cases} e^{-\frac{\|s_i - s_j\|^2}{2\sigma^2}} & \text{if } i \neq j, \\ 0 & \text{if } i = j \end{cases} \quad \text{scaling parameter } \sigma \text{ chosen by the user}$$

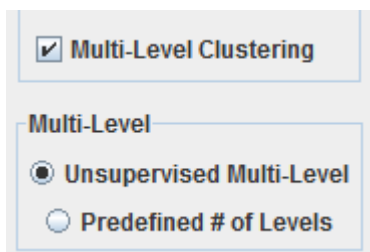
With Inverse Quadratic Kernel:

$$A_{ij} = \begin{cases} \frac{1}{\sqrt{\|s_i - s_j\|^2 + c}} & \text{if } i \neq j, \\ 0 & \text{if } i = j \end{cases} \quad \text{scaling parameter } c \text{ chosen by the user}$$

In both cases, $\|s_i - s_j\|$ stands for a basic measure of similarity. If $\|s_i - s_j\|$ is large, points s_i and s_j are said to be “closer” to each other in accordance with the distance measure being used. Such measure depends on whether the Connectivity Matrix or the Weight Matrix is being used as the base data for the affinity values, as mentioned in the [Parameters](#) section.

MULTI-LEVEL CLUSTERING

If sub-clusters within clusters are to be calculated, the user will have to activate the **Multi-Level Clustering** checkbox. This will require additional information to be entered in the **Multi-Level** region of the clustering window:



Multi-Level Clustering

Multi-Level

Unsupervised Multi-Level

Predefined # of Levels

- **Unsupervised Multi-Level:** for each calculated cluster, sub-clusters will be found until a point is reached where the sub-sub-...-sub-cluster cannot be divided.

Multi-Level Clustering

Multi-Level

Unsupervised Multi-Level

Predefined # of Levels

Predefined # of Levels: this type of multi-level clustering stops after **N** levels of clusters are found (limited depth of search). For each calculated cluster, sub-clusters will be found until a point is reached where the sub-sub-...-sub-cluster cannot be divided **or until N levels of clusters have already been calculated.** Therefore, for the example above, a tree of clusters is obtained in which **the maximum depth of search is of sub-sub-clusters with respect to the original clusters.**

OUTPUT

Jump to the [Clustering Output](#) section to learn about the results obtained after this type of analysis.

WEAK COMPONENT

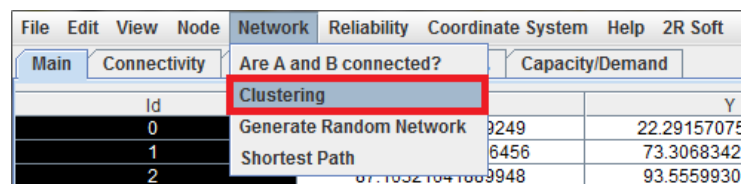
THEORY

Finds all weak components in a graph and regards them as separate clusters. A weak component is defined as a maximal subgraph in which all pairs of vertices in the subgraph are reachable from one another in the underlying undirected subgraph (JUNG 2011).

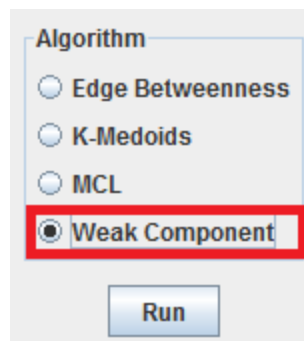
Running time: $O(|V| + |E|)$ where $|V|$ is the number of vertices and $|E|$ is the number of edges.

INPUT

To begin a clustering analysis in 2R Net, the user must navigate through the **Network** menu and select the **Clustering** option:



After selecting **Weak Component** as the clustering algorithm of choice, any relevant parameters are requested:



- In this case, no parameters are required. Refer to the [Theory](#) section to understand why.

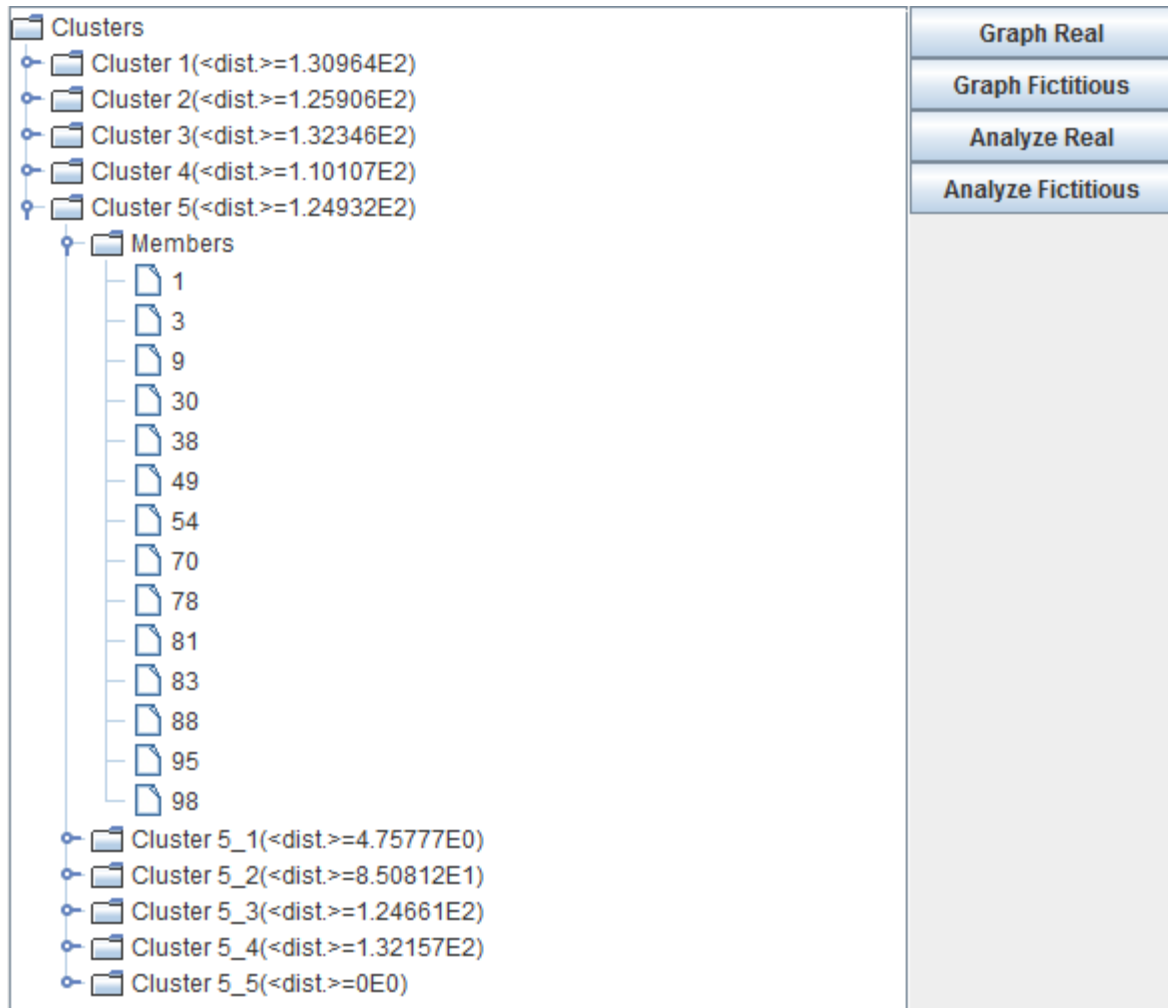
OUTPUT

Jump to the [Clustering Output](#) section to learn about the results obtained after this type of analysis is run.

CLUSTERING OUTPUT

TREE VIEW

After a clustering analysis is run, the results are displayed in the form of a tree view:



The screenshot displays a tree view of clustering results. The root node is 'Clusters', which is expanded to show five main clusters: Cluster 1, Cluster 2, Cluster 3, Cluster 4, and Cluster 5. Cluster 5 is further expanded to show a 'Members' folder containing a list of node IDs: 1, 3, 9, 30, 38, 49, 54, 70, 78, 81, 83, 88, 95, and 98. Below the 'Members' folder, Cluster 5 is further subdivided into five sub-clusters: Cluster 5_1, Cluster 5_2, Cluster 5_3, Cluster 5_4, and Cluster 5_5. Each cluster node includes a distance value in scientific notation (<dist.>=). To the right of the tree view is a vertical panel with four buttons: 'Graph Real', 'Graph Fictitious', 'Analyze Real', and 'Analyze Fictitious'.

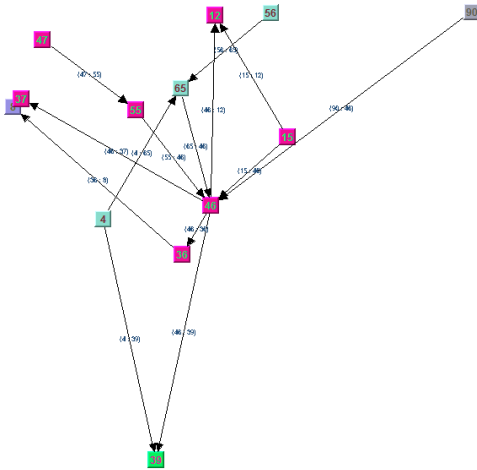
Nodes with a folder icon (clusters and members) can be expanded and contracted by double-clicking on them.

Every cluster contains:

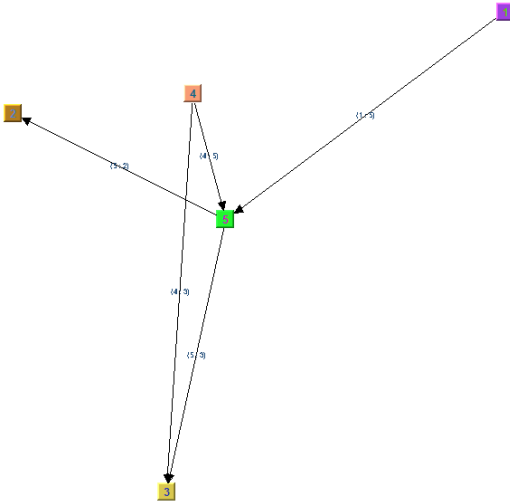
- The average shortest path weight between pairs of nodes inside the cluster (<dist.>).
- A **Members** folder with a list of all the nodes that comprise the cluster.
- A list of sub-clusters when applicable (if running a multi-level clustering).

The tree view provides 4 important options. **They are applied on the selected cluster only. The selected cluster is the one highlighted by the user by clicking on it.**

- **Graph Real:** a graph of the selected cluster's nodes is shown. The members of a cluster are represented with a unique color.



- **Graph Fictitious:** a graph of the selected cluster's sub-clusters is generated, where **each sub-cluster is represented as a single node corresponding to its medoid**. Every sub-cluster's medoid is graphed with a unique color. Most importantly, **the links between the medoids are EQUIVALENT LINKS**.



Equivalent link [A→B], connecting cluster A to cluster B in that direction has the following properties:

- Its **demand** is equal to the **sum** of the demands carried by all of the links connecting a node of cluster A with a node of cluster B **in that direction**.
- Its **capacity** is equal to the **sum** of the capacities carried by all of the links connecting a node of cluster A with a node of cluster B **in that direction**.
- Its **weight** is the **minimum weight** out of all of the links connecting a node of cluster A with a node of cluster B **in that direction**. This is because we assume that traveling within the clusters has an insignificant cost, so all the traffic from cluster A to cluster B will try to travel through the less costly significant segment.
- Its **failure probability** is equal to the **multiplication** of the probabilities of failure of the links connecting a node of cluster A with a node of cluster B **in that direction**. This is because, for the two clusters to become disconnected, all of the mentioned links must fail.

- **Analyze Real:** the selected cluster's nodes are exported to a new 2R Net window for further analysis.
- **Analyze Fictitious:** the selected cluster's sub-clusters are exported to a new 2R Net window for further analysis, where **each sub-cluster is represented as a single node corresponding to its medoid**. Most importantly, **the links between the medoids are EQUIVALENT LINKS**.

Equivalent link [A→B], connecting cluster A to cluster B in that direction has the following properties:

- Its **demand** is equal to the **sum** of the demands carried by all of the links connecting a node of cluster A with a node of cluster B **in that direction**.
- Its **capacity** is equal to the **sum** of the capacities carried by all of the links connecting a node of cluster A with a node of cluster B **in that direction**.
- Its **weight** is the **minimum weight** out of all of the links connecting a node of cluster A with a node of cluster B **in that direction**. This is because we assume that traveling within the clusters has an insignificant cost, so all the traffic from cluster A to cluster B will try to travel through the less costly significant segment.
- Its **failure probability** is equal to the **multiplication** of the probabilities of failure of the links connecting a node of cluster A with a node of cluster B **in that direction**. This is because, for the two clusters to become disconnected, all of the mentioned links must fail.

GENERATE RANDOM NETWORK

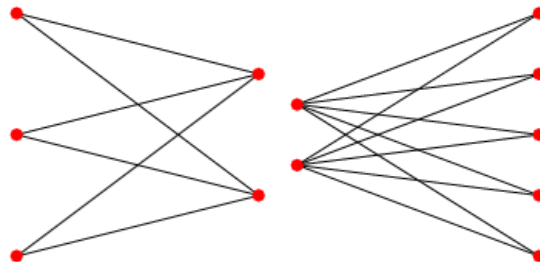
THEORY

Random networks can be created with different end results in mind. Each algorithm aims for a specific type of effect, with special properties in terms of the configuration of nodes and/or links.

COMPLETE NETWORK

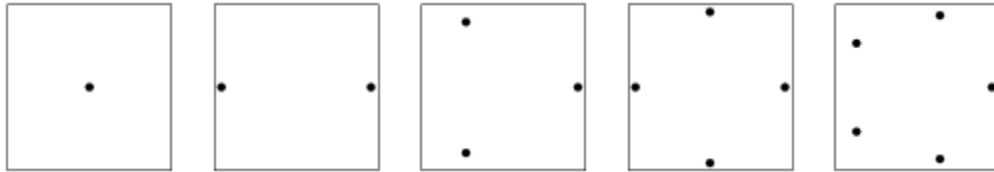
A complete graph is a graph where every vertex shares an edge with every other vertex. If it is a directed graph, then edges must always exist in both directions. On a side note, a complete graph is the least efficient possible graph in terms of memory and CPU usage.

COMPLETE BIPARTITE NETWORK



A complete bipartite graph is a bipartite graph (i.e., a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent) such that every pair of graph vertices in the two sets are adjacent (Weisstein 2011).

EMPTY NETWORK



An empty graph on n nodes consists of n isolated nodes with no edges. Such graphs are sometimes also called edgeless graphs or null graphs (Weisstein 2011).

HYPERCUBE NETWORK

This is a graph that can be represented by bit strings, so for an n -dimensional hypercube each vertex resembles an n -length bit string. Then, two vertices are adjacent if and only if their bitstring differ by exactly one element.

LINEAR NETWORK

A linear graph with N vertices has an N -step path that traverses the entire network in a single direction.

RANDOM NETWORK

This is the most usual type of algorithm. N nodes are placed randomly and the M links to create are then randomly selected from the $N*(N-1)$ possible links. No special properties are expected to result from this type of procedure.

RING NETWORK

A ring graph is a graph that contains a single cycle that passes through all its vertices exactly once. For a directed graph, the generated edges are oriented consistently around the ring.

SCALE FREE NETWORK

Scale-free network is a connected graph, where degrees of vertices are distributed in an unusual way. There are many vertices with small degrees and only small amount of vertices with big degrees.

STAR NETWORK

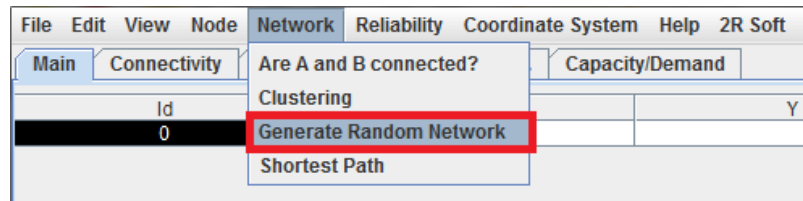
This is a graph where every vertex has exactly one edge with a center vertex.

WHEEL NETWORK

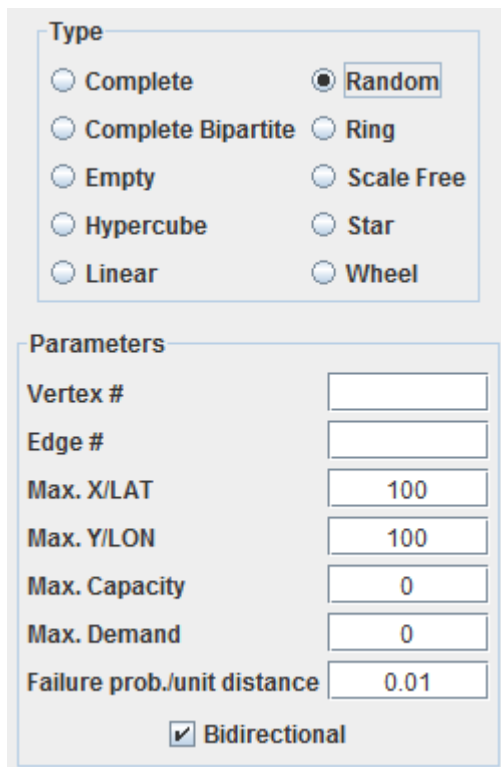
Reminding a bicycle wheel, a wheel graph has a hub vertex in the center and a rim of vertices around it that are connected to each other (as a ring). The rim vertices are also connected to the hub with edges that are called "spokes".

INPUT

To generate a random network, a user must navigate through the **Network** menu and select the **Generate Random Network** option:



A new dialog window will pop up asking for a variety of parameters:

A dialog window for network generation. It has two main sections: 'Type' and 'Parameters'. The 'Type' section contains radio buttons for: Complete, Random (selected), Complete Bipartite, Ring, Empty, Scale Free, Hypercube, Star, Linear, and Wheel. The 'Parameters' section contains input fields for: Vertex #, Edge #, Max. X/LAT (100), Max. Y/LON (100), Max. Capacity (0), Max. Demand (0), and Failure prob./unit distance (0.01). At the bottom, there is a checked checkbox for 'Bidirectional'.

- **Type:** the type of network that is to be generated. Refer to the [Theory](#) section for information on the different types of networks available.

PARAMETERS IN COMMON

All of the network generation algorithms have some parameters in common:

- **Max. X/LAT:** the X coordinate (or latitude) for each generated node will be in **[0, Max. X/LAT]**.
- **Max. Y/LON:** the Y coordinate (or longitude) for each generated node will be in **[0, Max. Y/LAT]**.
- **Max. Capacity:** the capacity for each generated link will be in **[0, Max. Capacity]**.
- **Max. Demand:** the demand for each generated link will be in **[0, Max. Demand]**.
- **Failure prob./unit distance:** the proportionality factor that will be used to compute failure probabilities for the generated links based on the distance covered by them. Refer to the [Failure Probability Matrix](#) section for more information regarding this parameter.

UNIQUE PARAMETERS

- **Complete Network**
 - **Vertex #:** number of vertices to be generated.
- **Complete Bipartite Network**
 - **Vertex #, P1:** the number of vertices in the first partition.
 - **Vertex #, P2:** the number of vertices in the second partition.
- **Empty Network**
 - **Vertex #:** number of vertices to be generated.
- **Hypercube Network**
 - **Dimension #:** dimension of the hypercube. (The length of the bitstrings to be used).
- **Linear Network**
 - **Vertex #:** number of vertices to be generated.
- **Random Network**
 - **Vertex #:** number of vertices to be generated.
 - **Edge #:** number of links to be generated.
 - **Bidirectional:** if this checkbox is selected, the edges that are generated are bidirectional (cover both directions).
- **Ring Network**
 - **Vertex #:** number of vertices to be generated.
- **Scale Free Network**
 - **Vertex #:** number of vertices to be generated.
- **Star Network**
 - **Vertex #:** number of vertices to be generated.
- **Wheel Network**
 - **Vertex #:** number of vertices to be generated.

OUTPUT

The active 2R Net window will be populated with the generated network after the user hits the **Generate** button.

SHORTEST PATH

THEORY

2R Net can run three different shortest path algorithms on a network (or graph), $G(V, E)$, with V being the set of vertices comprising the graph and E the set of edges connecting them. Here's a summary of their computational complexity and their use:

Algorithm	Use	Worst-Case Time Complexity
Dijkstra	Finding a single shortest path	$O(V ^2)$
Bellman-Ford	Finding a single shortest path	$O(V \times E)$
Floyd-Warshall	Finding the total weight of all shortest paths in the network	$O(V ^3)$

While the Dijkstra and Bellman-Ford algorithms find the weight and the step-by-step path from a specific node in the network to another node, the Floyd-Warshall algorithm gives a matrix of shortest path weights for the entire network. Thus, the first two algorithms are more detailed in their answer, but the latter has a wider coverage and is much more efficient than running the other two algorithms $|V|^2$ times for a network-wide analysis. **In the case of 2R Net, LINK WEIGHT is always used as the distance measure during shortest path analysis. Refer to the [Weight Matrix](#) section for more information on the way link weight is calculated.**

DIJKSTRA ALGORITHM

Let the node at which we are starting be called the **initial node**. Let the **distance of node Y** be the distance from the **initial node** to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step (FilePie 2011).

1. Assign to every node a distance value: set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes as unvisited. Set initial node as current.
3. For current node, consider all its unvisited neighbors and calculate their *tentative* distance. For example, if current node (A) has distance of 6, and an edge connecting it with another node (B) is 2, the distance to B through A will be $6+2=8$. If this distance is less than the previously recorded distance, overwrite the distance. All unvisited neighbors are added to an *unvisited set*.
4. When we are done considering all neighbors of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.
5. The next *current node* will be the node with the lowest distance in the *unvisited set*.
6. If all nodes have been visited, finish. Otherwise, set the unvisited node with the smallest distance (from the initial node, considering all nodes in graph) as the next "current node" and continue from step 3.

BELLMAN-FORD ALGORITHM

The **Bellman–Ford algorithm** computes single-source shortest paths in a weighted digraph. For graphs with only non-negative edge weights, the faster [Dijkstra's algorithm](#) also solves the problem. Thus, Bellman–Ford is used primarily for graphs with negative edge weights. The algorithm is named after its developers, Richard Bellman and Lester Ford, Jr (FilePie 2011).

Bellman–Ford is in its basic structure very similar to [Dijkstra's algorithm](#), but instead of greedily selecting the minimum-weight node not yet processed to relax, it simply relaxes *all* the edges, and does this $|V| - 1$ times, where $|V|$ is the number of vertices in the graph. The repetitions allow minimum distances to accurately propagate throughout the graph, since, in the absence of negative cycles, the shortest path can only visit each node at most once. Unlike the greedy approach, which depends on certain structural assumptions derived from positive weights, this straightforward approach extends to the general case (FilePie 2011).

FLOYD-WARSHALL ALGORITHM

In the words of (FilePie 2011):

The Floyd–Warshall algorithm compares all possible paths through the graph between each pair of vertices. It is able to do this with only $\Theta(|V|^3)$ comparisons in a graph. This is remarkable considering that there may be up to $\Omega(|V|^2)$ edges in the graph, and every combination of edges is tested. It does so by incrementally improving an estimate on the shortest path between two vertices, until the estimate is optimal.

Consider a graph G with vertices V , each numbered 1 through N . Further consider a function $\text{shortestPath}(i, j, k)$ that returns the shortest possible path from i to j using vertices only from the set $\{1, 2, \dots, k\}$ as intermediate points along the way. Now, given this function, our goal is to find the shortest path from each i to each j using only vertices 1 to $k + 1$.

There are two candidates for each of these paths: either the true shortest path only uses vertices in the set $\{1, \dots, k\}$; or there exists some path that goes from i to $k + 1$, then from $k + 1$ to j that is better. We know that the best path from i to j that only uses vertices 1 through k is defined by $\text{shortestPath}(i, j, k)$, and it is clear that if there were a better path from i to $k + 1$ to j , then the length of this path would be the concatenation of the shortest path from i to $k + 1$ (using vertices in $\{1, \dots, k\}$) and the shortest path from $k + 1$ to j (also using vertices in $\{1, \dots, k\}$).

If $w(i, j)$ is the weight of the edge between vertices i and j , we can define $\text{shortestPath}(i, j, k)$ in terms of the following recursive formula: the base case is

$$\text{shortestPath}(i, j, 0) = w(i, j)$$

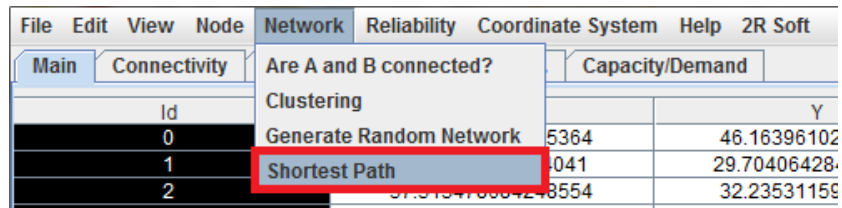
and the recursive case is

$$\text{shortestPath}(i, j, k) = \min(\text{shortestPath}(i, j, k-1), \text{shortestPath}(i, k, k-1) + \text{shortestPath}(k, j, k-1))$$

This formula is the heart of the Floyd–Warshall algorithm. The algorithm works by first computing $\text{shortestPath}(i, j, k)$ for all (i, j) pairs for $k = 1$, then $k = 2$, etc. This process continues until $k = n$, and we have found the shortest path for all (i, j) pairs using any intermediate vertices.

INPUT

To run a shortest path analysis, a user must navigate through the **Network** menu and select the **Shortest Path** option:



A new window will show up asking for the algorithm of choice and the pair of node to test (where appropriate):

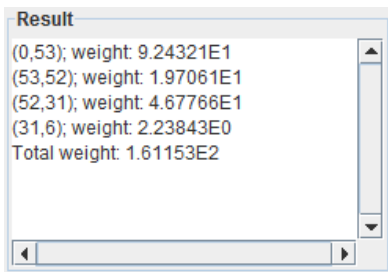
The dialog box has two sections: 'Algorithm' and 'Vertices'. In the 'Algorithm' section, there are three radio buttons: 'Dijkstra' (selected), 'Bellman-Ford', and 'Floyd-Warshall'. In the 'Vertices' section, there are two dropdown menus: 'Start: 16' and 'End: 16'. At the bottom of the dialog box is a 'Run' button.

- **Algorithm:** the shortest path algorithm to run. Refer to the [Theory](#) section for more information.
- **Vertices (Dijkstra and Bellman-Ford only) - ORDER DOES MATTER. The shortest path from A to B isn't necessarily the same as the one from B to A!**
 - **Start:** the node from which the algorithm should start. **Only nodes with incoming or outgoing links are shown.**
 - **End:** the node at which the algorithm should end. **Only nodes with incoming or outgoing links are shown.**

OUTPUT

DIJKSTRA AND BELLMAN-FORD

The result pane shows the step-by-step shortest path and the weight associated with each step:



(i, j) refers to the edge going from node i to node j. In the screenshot above, the shortest path from node 0 to node 6 is being described. **The result pane has a scroll bar to the right-hand side that has to be used to scroll through the result if the shortest path covers a large number of steps.**

FLOYD-WARSHALL

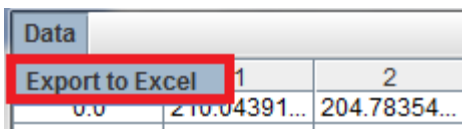
A new window pops up containing a matrix of shortest path weights for the entire network:

Data

0	1	2	3	4	5	6	7
0.0	210.04391...	204.78354...	225.22667...	205.69673...	152.46031...	161.15320...	208.14...
143.63179...	0.0	118.57475...	149.51426...	129.98432...	78.164181...	107.13929...	141.26...
167.23145...	92.651723...	0.0	131.85606...	134.72151...	135.04462...	99.797096...	138.47...
152.84678...	79.967089...	140.40859...	0.0	128.02787...	104.91329...	61.534370...	163.59...
124.89637...	139.39374...	106.52093...	130.61061...	0.0	106.02610...	62.890585...	149.57...
170.16811...	94.404684...	96.532336...	149.95201...	224.38901...	0.0	118.84593...	157.52...
120.36126...	129.90666...	115.08793...	166.99942...	169.86487...	74.139278...	0.0	124.26...
162.18132...	130.47761...	132.60526...	198.37677...	149.54502...	96.580117...	148.54399...	0.0
247.96689...	108.29896...	135.52356...	167.25289...	228.69370...	183.63026...	174.57033...	171.59...
184.59349...	128.87102...	135.61911...	229.66516...	215.40886...	172.11054...	127.55213...	167.40...
153.55134...	115.95448...	123.99721...	148.31742...	136.08908...	60.162791...	91.545549...	110.28...
121.36681...	105.04605...	142.11884...	148.17072...	151.03617...	90.616105...	46.057691...	140.74...
173.28931...	117.56684...	233.68264...	237.60050...	218.07055...	195.73102...	195.68592...	194.91...
162.84502...	90.170599...	53.039792...	174.98332...	134.23682...	143.44321...	131.11457...	191.51...
155.55923...	67.944790...	116.39835...	169.46086...	155.08589...	124.80852...	118.40560...	174.45...
83.387770...	112.32484...	77.897591...	167.87217...	109.76327...	17.920160...	103.00196...	104.17...
174.01344...	92.279987...	65.399423...	164.25942...	154.49046...	54.693538...	111.46407...	51.689...
79.159294...	88.704695...	73.885966...	182.50695...	162.97700...	128.94893...	63.659719...	135.64...

Current Cell: (-1,-1)

Cell (i, j) contains the total weight of the shortest path traversing from node i to node j. The matrix can be exported to Microsoft Excel by selecting the **Export to Excel** option from the **Data** menu located at the top-left corner:



RELIABILITY ANALYSIS

CONNECTIVITY TYPES

When testing the reliability of a network, three different connectivity definitions can be employed. A system that is highly reliable under one definition of connectivity can be remarkably unreliable when using a different definition for the same property.

- **Weak Connectivity:** node A is said to be *weakly connected* to node B if there is a path from A to B in a network where link directions are ignored. That is, every existing link is regarded as bidirectional for the connectivity test.
- **Normal Connectivity:** node A is said to be *normally connected* to node B if there is a path from A to B **OR** from B to A in the network.
- **Strong Connectivity:** node A is said to be *strongly connected* to node B if there is a path from A to B **AND** from B to A.

It can be shown that strongly connectivity implies normal and weak connectivity as well. Also, normal connectivity implies weak connectivity.

ANALYZABLE SYSTEMS

Three types of analyzable systems can be derived from a network when studying reliability:

- **The Entire Network:** the connectivity definition is applied to every pair of nodes inside the network. If at least one pair of nodes is disconnected, then the entire network is said to be disconnected.
- **Pair of Nodes (A<->B):** the connectivity definition is applied to a specific pair of nodes (A, B).
- **Path (A->B):** this type of system is considered as connected **if and only if** there is a path from node A to node B **in that direction**. This system has its own definition of connectivity, which is the midpoint between normal and strong connectivity of a pair of nodes (the **OR** is too weak, but the **AND** is too strong).

THEORY

There are two types of cut sets (Weisstein 2011):

- **Edge-Cut Set:** A set of edges of a graph (or network) which, if removed (or "cut"), disconnects the graph.
- **Vertex-Cut Set:** A set of vertices of a graph (or network) which, if removed (or "cut"), disconnects the graph.

Clearly, the "disconnects the graph" part of the definition depends on the [type of connectivity](#) that is being tested (weak, normal, or strong).

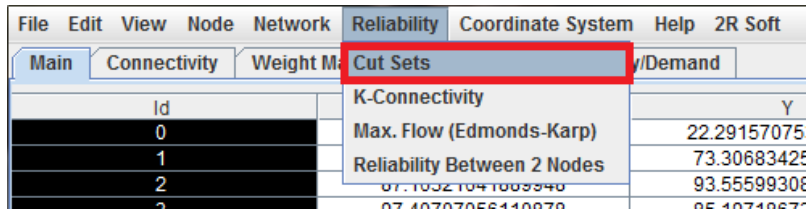
In reliability analysis, only the **minimal cut sets** are of interest. Those are found by testing sets of only 1 element, then testing sets of 2 elements, and so on. If a cut set of N elements contains a previously found cut set of M elements, where $M < N$, then the N-member cut set **isn't** a **minimal cut set**.

The process of finding cut sets in a network entails testing all of the possible edge or vertex subsets one at a time, leading to **non-polynomial running times**. This is why 2R Net's user interface provides real-time information about the minimal cut sets that have been found and the number of cut sets that have been tested. In addition, the algorithm can be stopped at any moment. **The following optimizations have been made in 2R Net to reduce the running time:**

1. Weak Connectivity and Strong Connectivity network-wide inspection algorithms rely on graph theory theorems with computational complexities much lower than the ones associated with the brute-force approach. This means that the Normal Connectivity network-wide inspection is considerably slower than the other two alternatives, as it requires a brute-force methodology.
2. When testing vertex cut sets, the vertices are sorted in descending degree order (incoming plus outgoing links), so that the vertices with a higher probability of affecting the network as a whole are tested first.
3. Cut sets containing previously found minimal cut sets aren't tested. Therefore, the process accelerates as more minimal cut sets are found, since the number of remaining possible cut sets is reduced.

INPUT

To begin a cut set search, the user must navigate through the **Reliability** menu and select the **Cut Sets** option:



A new window pops up with a wide array of options:

The dialog box is titled 'Cut Sets' and contains the following sections:

- Find:** Radio buttons for Edge-Cut Sets and Vertex-Cut Sets.
- System to Analyze:** Radio buttons for All the Network, A<->B, and A->B.
- Connectivity Type:** Radio buttons for Weak Connectivity, Normal Connectivity, and Strong Connectivity.
- Parameters:** A text box for 'Max. Cut Set Size' with the value '2', and two dropdown menus for 'A:' and 'B:', both currently showing '0'.
- Run:** A blue button labeled 'Run'.
- Sets tried:** A text box showing '0'.
- Size of the last cut set that was tested:** A text box showing '0'.
- Results:** A large empty text area for displaying results.

- The difference between edge-cut sets and vertex-cut sets is covered in the [Theory](#) section. Meanwhile, the [Analyzable Systems](#) and [Connectivity Types](#) sections explain the options contained in the **Connectivity Type** and **System to Analyze** regions.
- There are three parameters:
 - **Max. Cut Set Size:** the maximum cut set size to be tested by the algorithm.
 - **A and B (only when testing a pair of nodes or a path)**
 - Nodes A and B are the initial and final nodes in the A<->B and A->B systems. Note that, when analyzing a path (A->B), the **order does matter**.

OUTPUT

Cut sets are listed in the **Results** region. Every line represents a cut set and members of the same cut sets are separated by a comma (.). Edge-cut set members are presented in the form H->K, where H->K is the link that goes from node H to node K. Meanwhile, vertex-cut set members are presented in the form H, where H is the node ID of the vertex in question.

The image displays two side-by-side software windows. The left window is titled "Edge-Cut Sets" and features a "Stop" button at the top. Below it, a progress bar indicates "Sets tried: 25519". A text label states "Size of the last cut set that was tested: 2". The "Results" section contains a list of edge cut sets: 96->30, 0->59, 51->18, 67->76, 44->20, 87->20, 95->24, 63->24, and 97->56, 97->72. The right window is titled "Vertex-Cut Sets" and also has a "Stop" button. Its progress bar shows "Sets tried: 13463". A text label indicates "Size of the last cut set that was tested: 3". The "Results" section lists vertex cut sets: 56, 72; 61, 35; 61, 3; 45, 87; 29, 7; 97, 25; and 79, 82, 34. Both windows have scroll bars on the right and bottom.

Note: the results can be copied (ctrl+c) and pasted (ctrl+v) in an external program (e.g. Microsoft Word).

K-CONNECTIVITY

THEORY

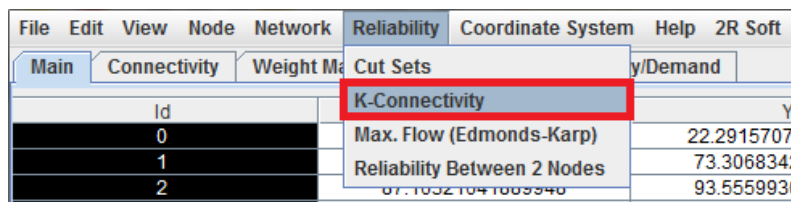
A graph (or network) G is said to be **k-vertex connected** if there does not exist a set of $k-1$ vertices (or nodes) whose removal disconnects the graph (Weisstein 2011). Similarly, a graph (or network) G is said to be **k-edge connected** if there does not exist a set of $k-1$ edges (or links) whose removal disconnects the graph.

From the definition, a network's k -connectivity is found by determining its smallest minimal cut set. If the network is initially disconnected, then its k -vertex and k -edge connectivity is equal to 0.

2R Net's k -connectivity algorithm is the same as the one used for the [Cut Sets option](#), but is limited to the entire network ($A \leftrightarrow B$ and $A \rightarrow B$ systems don't make sense in this case) and **halts as soon as the first minimal cut set is found**.

INPUT

To calculate the k -connectivity of a network, a user must navigate through the **Reliability** menu and select the **k-connectivity** option:



A new dialog window is shown asking for some parameters:

The 'Find' dialog window is shown. It has two sections: 'Find' and 'Connectivity Type'. In the 'Find' section, 'K-Vertex Connectivity' is selected with a radio button. In the 'Connectivity Type' section, 'Weak Connectivity' is selected with a radio button. There is a 'Run' button and a 'Sets tried: 0' indicator at the bottom.

- The difference between k -vertex connectivity and k -edge connectivity is covered in the [Theory](#) section. Meanwhile, the [Connectivity Types](#) section explains the options contained in the **Connectivity Type** region. Keep in mind that the [Entire Network](#) is the system being analyzed in this case.

OUTPUT

A message pops up as soon as the first minimal cut set is found. The message mentions the cut set that led to the k-connectivity conclusion:



Cut set members are separated by a comma (,). Edge-cut set members are presented in the form H->K, where H->K is the link that goes from node H to node K. Meanwhile, vertex-cut set members are presented in the form H, where H is the node ID of the vertex in question.

MAX. FLOW (EDMONDS-KARP)

THEORY

In the words of (Wikipedia 2011):

The **Ford–Fulkerson algorithm** (named for L. R. Ford, Jr. and D. R. Fulkerson) computes the maximum flow in a flow network. It was published in 1956. The name "Ford–Fulkerson" is often also used for the **Edmonds–Karp algorithm**, which is a specialization of Ford–Fulkerson.

The idea behind the algorithm is very simple: As long as there is a path from the source (start node) to the sink (end node), with available capacity on all edges in the path, we send flow along one of these paths. Then we find another path, and so on. A path with available capacity is called an augmenting path.

Let $G(V,E)$ be a graph, and for each edge from u to v , let $c(u,v)$ be the capacity and $f(u,v)$ be the flow. We want to find the maximum flow from the source s to the sink t . After every step in the algorithm the following is maintained:

Capacity constraints: $f(u, v) \leq c(u, v)$ The flow along an edge cannot exceed its capacity.

Skew symmetry: $f(u, v) = -f(v, u)$ The net flow from u to v must be the opposite of the net flow from v to u (see example).

Flow conservation: $\sum_{w \in V} f(u, w) = 0$ That is, unless $u = s$ or $w = t$. The net flow to a node is zero, except for the source, which "produces" flow, and the sink, which "consumes" flow.

This means that the flow through the network is a *legal flow* after each round in the algorithm. We define the **residual network** $G_f(V, E_f)$ to be the network with capacity $c_f(u, v) = c(u, v) - f(u, v)$ and no flow. Notice that it can happen that a flow from v to u is allowed in the residual network, though disallowed in the original network: if $f(u, v) > 0$ and $c(v, u) = 0$ then $c_f(v, u) = c(v, u) - f(v, u) = f(u, v) > 0$.

Inputs Graph G with flow capacity c , a source node s , and a sink node t

Output A flow f from s to t which is a maximum

1. $f(u, v) \leftarrow 0$ for all edges (u, v)
2. While there is a path p from s to t in G_f , such that $c_f(u, v) > 0$ for all edges $(u, v) \in p$:
 1. Find $c_f(p) = \min \{c_f(u, v) : (u, v) \in p\}$
 2. For each edge $(u, v) \in p$
 1. $f(u, v) \leftarrow f(u, v) + c_f(p)$ (Send flow along the path)
 2. $f(v, u) \leftarrow f(v, u) - c_f(p)$ (The flow might be "returned" later)

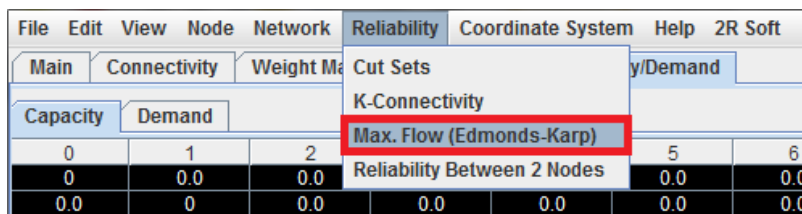
The path in step 2 can be found with for example a [breadth-first search](#) or a [depth-first search](#) in $G_f(V, E_f)$. If you use the former, the algorithm is called **Edmonds–Karp**.

When no more paths in step 2 can be found, s will not be able to reach t in the residual network. If S is the set of nodes reachable by s in the residual network, then the total capacity in the original network of edges from S to the remainder of V is on the one hand equal to the total flow we found from s to t , and on the other hand serves as an upper bound for all such flows. This proves that the flow we found is maximal.

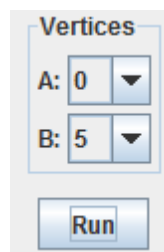
INPUT

2R Net's Maximum Flow (Edmonds-Karp) algorithm uses the network's [Capacity Matrix](#) as its main input.

To begin a Maximum Flow analysis, a user must navigate through the **Reliability** menu and select the **Max. Flow (Edmonds-Karp)** option:



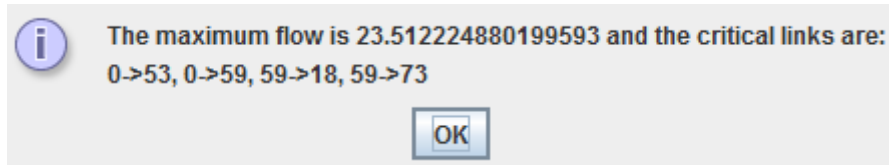
A new window will ask for the **source node, A**, and the **sink node, B**, for the analysis:



Refer to the [Theory](#) section to understand the role of the source (A) and the sink (B) nodes in this type of analysis.

OUTPUT

A message pops up with the maximum flow from A to B under the capacity constraints given by the [Capacity Matrix](#). In addition, the **critical links** are enumerated in the form H->K, where H->K is the link that goes from node H to node K. **If a critical link's capacity decreases, the source-sink system's maximum flow decreases as well.**



RELIABILITY BETWEEN 2 NODES

THEORY

Reliability analysis circles around failure probabilities, given that it's the standard way of quantifying risk. In 2R Net, failure probabilities of the network's links and nodes are declared in the [Failure Probability Matrix](#). They can then be used in conjunction with a cut set analysis to determine the probability of failure of a two-node system (A->B) or a path (A->B). The cut set finding procedure is the same as the one described in the [Cut Set Analysis theory section](#), but the cut sets are then used to calculate the joint failure probability of the system.

EDGE-CUT SET FAILURE PROBABILITY

Let $\{e_1, e_2, \dots, e_m\}$ be a minimal edge-cut set of network $G(V, E)$. Each edge (or link), e_i , has a probability of failure that is explicitly found in the [Failure Probability Matrix](#). For example, if e_i is the edge that connects node A with node B, then its failure probability is equal to the value located at cell (A, B) of the previously mentioned matrix. **Assuming that edge (or link) failures are independent events**, the failure probability associated with an edge-cut set is given by the multiplication of the failure probabilities of its members:

$$P_f(\{e_1, e_2, \dots, e_m\}) = \prod_{\{e_1, e_2, \dots, e_m\}} P_f(\text{edge})$$

VERTEX-CUT SET FAILURE PROBABILITY

Let $\{v_1, v_2, \dots, v_m\}$ be a minimal vertex-cut set of network $G(V, E)$. Each vertex, v_i , has a failure probability, but its value can be one of two alternatives:

- **Arbitrary value in the [Failure Probability Matrix \(FPM\)](#):** as mentioned in the section of this document dedicated to the FPM, the FPM's main diagonal contains the failure probabilities of the network nodes. That is, cell (i, i) gives the failure probability of node i. Nonetheless, these cannot be automatically calculated and must be manually entered by the user. Thus, this alternative is only useful when the user is well-acquainted with the network that is being modeled.
- **Automatically estimated based on edge failure probabilities:** if the [Failure Probability Matrix](#) is missing values in its main diagonal, vertex failure probabilities can be estimated by **assuming that a node's failure is equivalent to a total disconnection from the network. That is, a vertex is said to have failed when all of its incoming and outgoing links have failed.** Consequently, **assuming that edge failures are independent events**, $P_f(v_i) = \prod_{\{\text{incoming and outgoing}\}} P_f(\text{edge})$

Regardless of the chosen alternative and **assuming that vertex (or node) failures are independent events**, the failure probability associated with a vertex-cut set is given by the multiplication of the failure probabilities of its members:

$$P_f(\{v_1, v_2, \dots, v_m\}) = \prod_{\{v_1, v_2, \dots, v_m\}} P_f(\text{vertex})$$

SYSTEM FAILURE PROBABILITY

If we regard cut sets as independent events, then **the system's failure probability is the probability of at least one of its minimal cut sets occurring**. The following theorem is of great help in the calculation of that result (ProofWiki 2011):

Let $\mathcal{E} = (\Omega, \Sigma, \text{Pr})$ be a probability space.

Let $A_1, A_2, \dots, A_m \in \Sigma$ be independent events in the event space of \mathcal{E} .

Then the probability of at least one of A_1 to A_m occurring is:

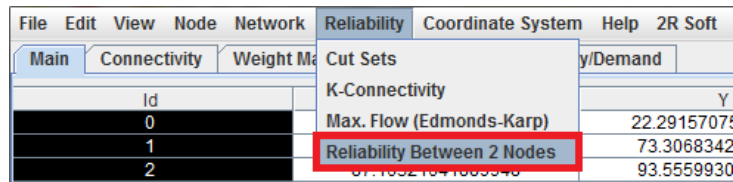
$$1 - \prod_{i=1}^m (1 - \text{Pr}(A_i))$$

From this theorem, one may assert that a system's failure probability resulting from its minimal cut sets can be calculated as:

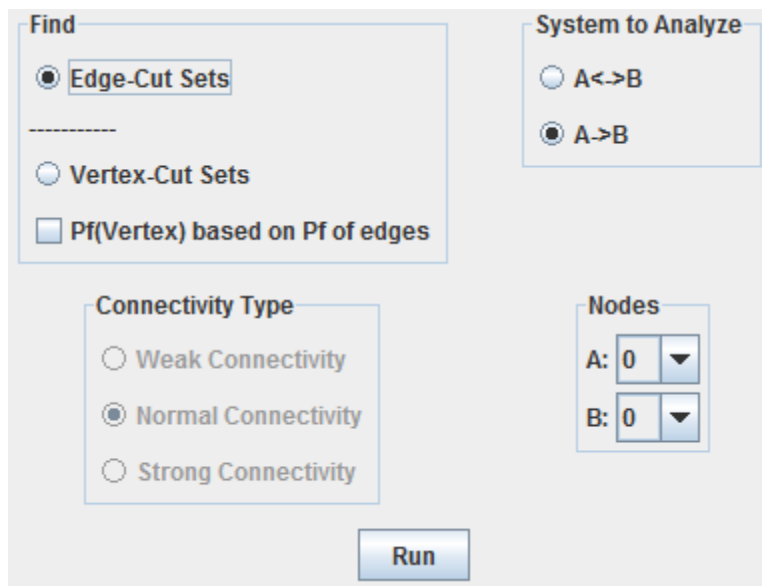
$$P_f(\text{system}) = 1 - \prod_{\text{minimal cut sets}} [1 - P_f(\text{cut set})]$$

INPUT

To investigate the reliability of a system comprised by two nodes, a user must navigate through the **Reliability** menu and select the **Reliability Between 2 Nodes** option:



A new window pops up with a wide array of options:



- The differences between edge-cut set and vertex-cut set failure probability calculations are discussed in the [Theory](#) section. Meanwhile, the [Analyzable Systems](#) and [Connectivity Types](#) sections explain the options contained in the **Connectivity Type** and **System to Analyze** regions.
- If the **Pf(Vertex) based on Pf of edges** checkbox is ticked, the second alternative mentioned in the [Vertex-Cut Set Failure Probability](#) section is employed. Otherwise, the first alternative (matrix values) is applied.
- **A and B**
 - Nodes A and B are the initial and final nodes in the A<->B and A->B systems. Note that, when analyzing a path (A->B), the **order does matter**.

OUTPUT

The result table is filled out as new minimal cut sets are found. It has four columns:

- **Cut Set:** Cut set members are separated by a comma (,). Edge-cut set members are presented in the form H->K, where H->K is the link that goes from node H to node K. Meanwhile, vertex-cut set members are presented in the form H, where H is the node ID of the vertex in question.
- **Cut Set Pf:** the probability of failure of the specific cut set as an independent event. Refer to the [Theory](#) section for information on how this is calculated.
- **Delta Pf:** the system failure probability increase that the cut set generated with respect to the previous value of the **Pf** column. That is, $(\text{Delta Pf})_i = \text{Pf}_i - \text{Pf}_{i-1}$.
- **Pf:** the resulting system failure probability after the cut set was found. Refer to the [Theory](#) section for information on how this is calculated.

Sets tried: 21769417

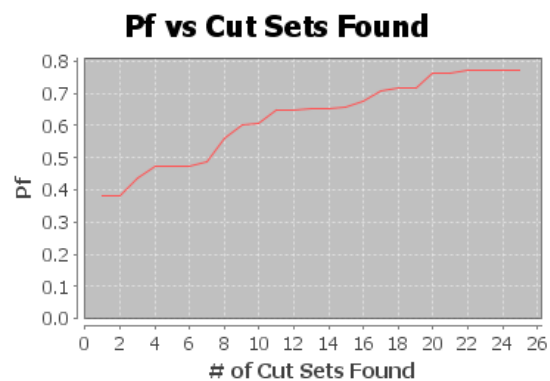
Size of the last cut set that was tested: 6

Results

Cut Set	Cut Set Pf	Delta Pf	Pf
4->5, 7->5	3.81335E-1	3.81335E-1	3.81335E-1
0->4, 0->9, ...	1.82808E-3	1.13097E-3	3.82466E-1
4->5, 1->8, ...	8.73146E-2	5.39197E-2	4.36386E-1
4->5, 6->7, ...	6.29784E-2	3.54955E-2	4.71881E-1
0->4, 13->1...	9.94196E-4	5.25054E-4	4.72406E-1
0->4, 10->4	1.75324E-3	9.24998E-4	4.73331E-1

The result table's contents can be easily exported to Microsoft Excel with the *Export to Excel* button located below it.

If two or more minimal cut sets have been found, a **Pf vs Cut Sets Found** plot pops to show the system failure probability's convergence:



As the number of minimal cut sets increases, the contribution of each new individual cut set towards the system's failure probability lowers in significance to the point where a clear asymptote in the **Pf** value is reached. In the graph shown above, the asymptote was reached after 22 cut sets were found. Given that the cut set finding process has **non-polynomial running time**, it is usually impractical to test all the possible cut set combinations. Hence, the **Pf** convergence point is what the user should employ as his analysis termination criterion.

After the user hits the **Stop** button, the **Graph Selected Cut Set** and the **Graph All Cut Sets** buttons become activated:

Run

Sets tried: 3230347

Size of the last cut set that was tested: 5

Results

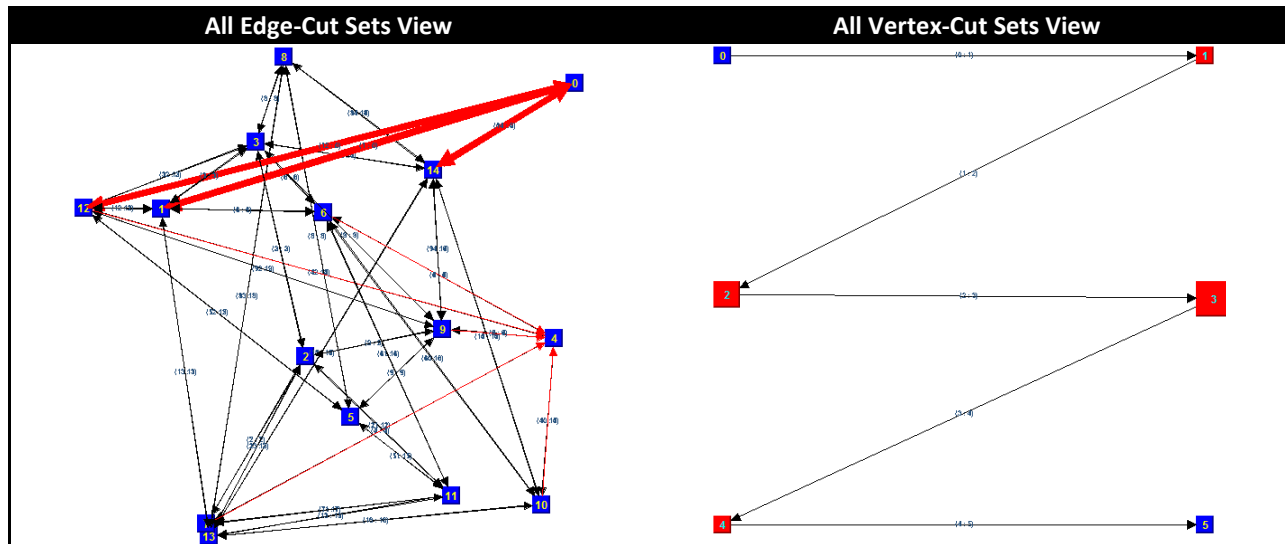
Cut Set	Cut Set Pf	Delta Pf	Pf
0->12, 0->1...	1.98753E-1	1.98753E-1	1.98753E-1
1->0, 12->0...	1.98753E-1	1.5925E-1	3.58003E-1
12->4, 10->...	1.67715E-2	1.07673E-2	3.6877E-1

Export to Excel

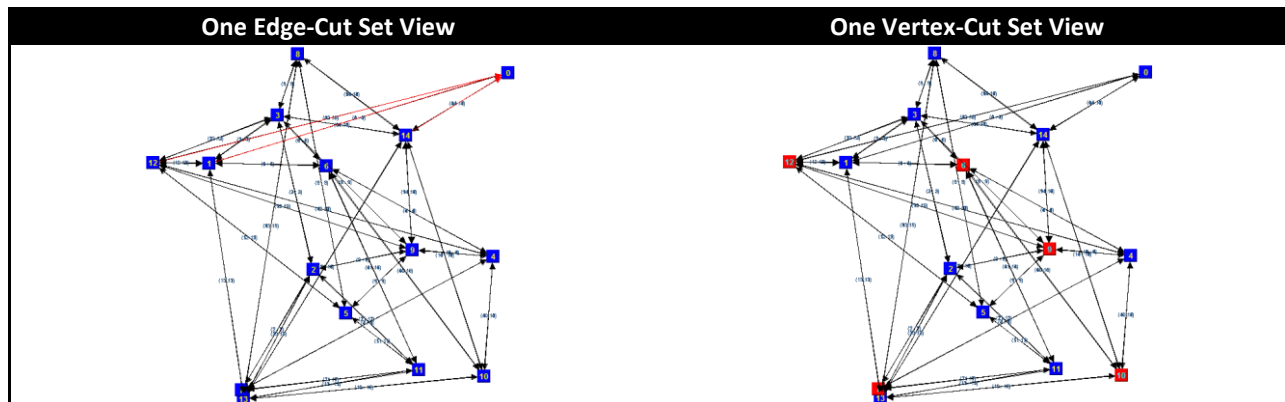
Graph Selected Cut Set

Graph All Cut Sets

The **Graph All Cut Sets** button generates a view of the network in which the cut sets are displayed in **red**. If edge-cut sets are being used, **thicker links indicate a cut set with a higher failure probability**. On the other hand, if vertex-cut sets are being used, **larger nodes indicate a cut set with a higher failure probability**.



If the user wants to view a specific cut set, he must select the cut set in the result table (left click) and then hit the **Graph Selected Cut Set** button. A network view is generated in which the cut set is displayed in **red**.



BIBLIOGRAPHY

Abeel, T. "K-Medoids." from <http://java-mml.sourceforge.net/api/0.1.6/net/sf/javaml/clustering/KMedoids.html>.

Annis, C. "Goodness-of-Fit tests for Statistical Distributions." Retrieved 05/17/2010, from <http://www.statisticalengineering.com/goodness.htm>.

Dipartimento Di Elettronica E Informazione, P. D. M. "A Tutorial on Clustering Algorithms." 2011, from http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/.

EncyclopediaOfStatistics. "Quartiles." Retrieved 05/17/2010, from <http://books.google.com/books?id=56LkkdZPpyoC&pg=PT19&lpg=PT19>.

ERI. "Kolmogorov-Smirnov Test." Retrieved 05/17/2010, from <http://www.eridlc.com/onlinetextbook/index.cfm?fuseaction=textbook.appendix&FileName=Table7>.

FilePie (2011). "Bellman–Ford algorithm." from http://www.filepie.us/?title=Bellman%E2%80%93Ford_algorithm.

FilePie (2011). "Breadth-first search." from http://www.filepie.us/?title=Breadth-first_search.

FilePie (2011). "Depth-first search." from http://www.filepie.us/?title=Depth-first_search.

FilePie (2011). "Dijkstra's algorithm." from http://www.filepie.us/?title=Dijkstra%27s_algorithm.

FilePie (2011). "Floyd–Warshall algorithm." from http://www.filepie.us/?title=Floyd%E2%80%93Warshall_algorithm.

GIRVAN, #160, et al. (2002). Community structure in social and biological networks. Washington, DC, ETATS-UNIS, National Academy of Sciences.

A number of recent studies have focused on the statistical properties of networked systems such as social networks and the Worldwide Web. Researchers have concentrated particularly on a few properties that seem to be common to many networks: the small-world property, power-law degree distributions, and network transitivity. In this article, we highlight another property that is found in many networks, the property of community structure, in which network nodes are joined together in tightly knit groups, between which there are only looser connections. We propose a method for detecting such communities, built around the idea of using centrality indices to find community boundaries. We test our method on computer-generated and real-world graphs whose community structure is already known and find that the method detects this known structure with high sensitivity and reliability. We also apply the method to two networks whose community structure is not well known—a collaboration network and a food web—and find that it detects significant and informative community divisions in both cases.

JUNG (2011). "MarkovClustering." from <http://java-sourceforge.net/api/0.1.1/net/sf/javaml/clustering/mcl/MarkovClustering.html>.

JUNG (2011). "WeakComponentClusterer." from <http://jung.sourceforge.net/doc/api/edu/uci/ics/jung/algorithms/cluster/WeakComponentClusterer.html>.

Lane2, D. "Standard Deviation and Variance." Retrieved 05/17/2010, from <http://davidmlane.com/hyperstat/A16252.html>.

MathsRevision.net. "Box and Whisker Diagrams." Retrieved 05/17/2010, from <http://www.mathsrevision.net/alevel/pages.php?page=50>.

ProofWiki (2011). "Probability of Occurrence of At Least One Independent Event." from http://www.proofwiki.org/wiki/Probability_of_Occurrence_of_At_Least_One_Independent_Event.

ReliaSoftCorp. "Critical Values for Cramér-von Mises Test." Retrieved 05/17/2010, from http://www.weibull.com/RelGrowthWeb/Appendix_B_Critical_Values_for_Cramer-von_Mises_Test.htm.

SEMATECH1, N. "Kolmogorov-Smirnov Goodness-of-Fit Test." Retrieved 05/17/2010, from <http://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm>.

SEMATECH2, N. "Anderson-Darling Test." Retrieved 05/17/2010, from <http://www.itl.nist.gov/div898/handbook/eda/section3/eda35e.htm>.

SEMATECH3, N. "Measures of Skewness and Kurtosis." Retrieved 05/17/2010, from <http://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm>.

Simard, R. "Package umontreal.iro.lecuyer.probdist." Retrieved 05/17/2010, from <http://www.iro.umontreal.ca/~simardr/ssj/doc/html/umontreal/iro/lecuyer/probdist/package-summary.html>.

Temporis, S. "Cluster Analysis." 2011, from <http://www.spiritus-temporis.com/cluster-analysis/distance-measure.html>.

Type, M. (2011). "Calculate distance, bearing and more between Latitude/Longitude points." from <http://www.movable-type.co.uk/scripts/latlong.html>.

Weisstein2, E. W. "Arithmetic Mean." Retrieved 05/17/2010, from <http://mathworld.wolfram.com/ArithmeticMean.html>.

Weisstein3, E. W. "Variance." Retrieved 05/17/2010, from <http://mathworld.wolfram.com/Variance.html>.

Weisstein, E. W. "Indegree." from <http://mathworld.wolfram.com/Indegree.html>.

Weisstein, E. W. "K-Means Clustering Algorithm." from <http://mathworld.wolfram.com/K-MeansClusteringAlgorithm.html>.

Weisstein, E. W. "Outdegree." from <http://mathworld.wolfram.com/Outdegree.html>.

Weisstein, E. W. (2011). "Complete Bipartite Graph." from <http://mathworld.wolfram.com/CompleteBipartiteGraph.html>.

Weisstein, E. W. (2011). "Cut Set." from <http://mathworld.wolfram.com/CutSet.html>.

Weisstein, E. W. (2011). "Empty Graph." from <http://mathworld.wolfram.com/EmptyGraph.html>.

Weisstein, E. W. (2011). "k-Connected Graph." from <http://mathworld.wolfram.com/k-ConnectedGraph.html>.

Wikipedia (2011). "Centrality." from <http://en.wikipedia.org/wiki/Centrality>.

Wikipedia (2011). "Clustering coefficient." from http://en.wikipedia.org/wiki/Clustering_coefficient.

Wikipedia (2011). "Ford–Fulkerson algorithm." from http://en.wikipedia.org/wiki/Ford%E2%80%93Fulkerson_algorithm.