Network Analysis in Complex Environment

Analysis of weighted networks

Vladimir Batagelj

IMFM Ljubljana, IAM UP Koper

Spinaker Summer School

Kraków, Poland, 3-14. July 2023

European Funds
Knowledge Education Development

Republic of Poland

NAWA
POLISH NATIONAL AGENCY FOR ACADEMIC EXCHANGE

European Union
European Social Fund

Outline

PROGRAM SPINAKER

Vladimir Batagelj: vladimir.batagelj@fmf.uni-lj.si

Last version of slides (June 28, 2023, 03 : 09): Wnets.pdf

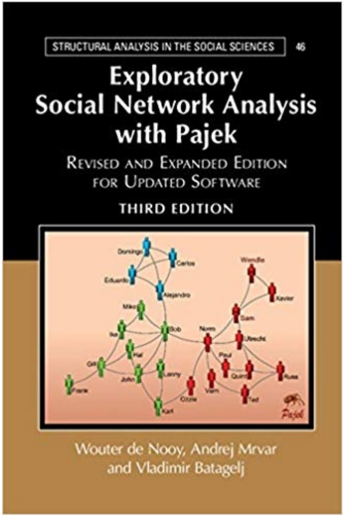
Info

Timetable

Fri	7^{th}	14:00 – 17:15
Mon	10^{th}	15:45 – 17:15
Tue	11^{th}	11:30 – 13:00
Wed	12^{th}	15:45 – 17:15
Thu	13^{th}	14:00 – 15:30

https://github.com/bavla/wNets

ESNA **Pajek**



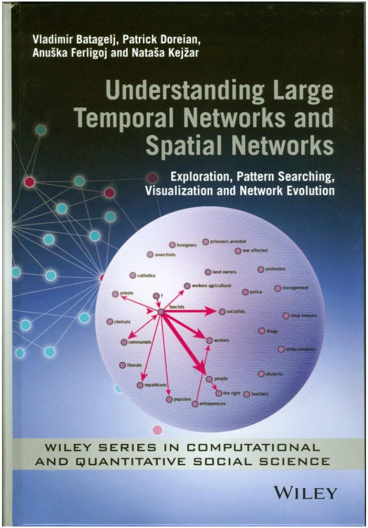
An introduction to social network analysis with **Pajek** is available in the book ESNA 3 (de Nooy, Mrvar, Batagelj, CUP 2005, 2011, 2018).

ESNA in Japanese was published by Tokyo Denki University Press in 2010; and in Chinese by Beijing World Publishing in November 2012.

Pajek – program for analysis and visualization of large networks is freely available, for noncommercial use, at its web site.

http://mrvar.fdv.uni-lj.si/pajek/

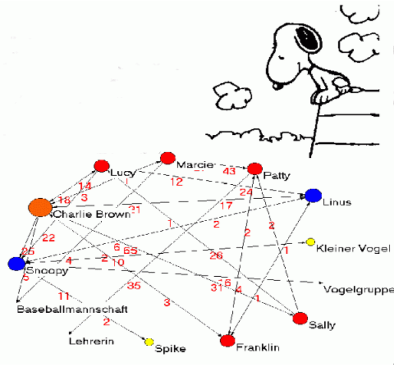
Understanding large networks



This course is closely related to chapters 2 and 3 in the book:

Vladimir Batagelj, Patrick Doreian, Anuška Ferligoj and Nataša Kejžar: Understanding Large Temporal Networks and Spatial Networks: Exploration, Pattern Searching, Visualization and Network Evolution. Wiley Series in Computational and Quantitative Social Science. Wiley, October 2014.

Networks



Alexandra Schuler/ Marion Laging-Glaser: Analyse von Snoopy Comics

A *network* is based on two sets – set of *nodes* (vertices), that represent the selected *units*, and set of *links* (lines), that represent *ties* between units. They determine a *graph*. A link can be *directed* – an *arc*, or *undirected* – an *edge*.

Additional data about nodes or links can be known – their *properties* (attributes). For example: name/label, type, value, . . .

Network = Graph + Data

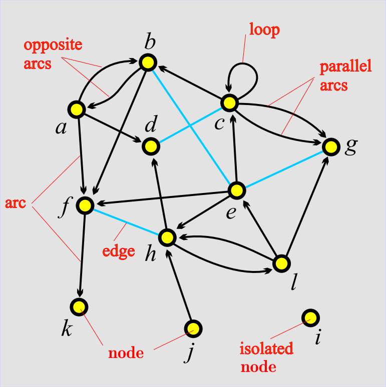
The data can be measured or computed.

Networks / Formally

A *network* $\mathbf{N} = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W})$ consists of:

- a *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{A}$ is the set of arcs, $\mathcal{E}$ is the set of edges, and $\mathcal{L} = \mathcal{E} \cup \mathcal{A}$ is the set of links.
  $n = |\mathcal{V}|$, $m = |\mathcal{L}|$
- $\mathcal{P}$ *node value functions* / properties: $p : \mathcal{V} \to A$
- $\mathcal{W}$ *link value functions* / weights: $w : \mathcal{L} \to B$

Graph

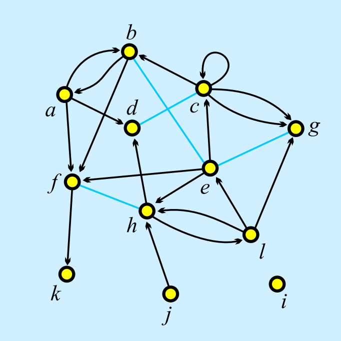


unit, actor – node, vertex
tie, line – link, edge, arc

arc = directed link, (a, d)
a is the *initial* node,
d is the *terminal* node.

edge = undirected link, $(c : d)$
c and d are *end* nodes.

Graph / Sets – NET



$$\mathcal{V} = \{a, b, c, d, e, f, g, h, i, j, k, l\}$$

$$\mathcal{A} = \{(a, b), (a, d), (a, f), (b, a), (b, f), (c, b), (c, c), (c, g)_1, (c, g)_2, (e, c), (e, f), (e, h), (f, k), (h, d), (h, l), (j, h), (l, e), (l, g), (l, h)\}$$

$$\mathcal{E} = \{(b : e), (c : d), (e : g), (f : h)\}$$

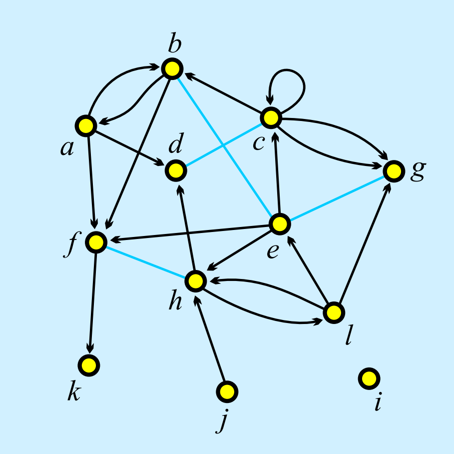
$$\mathbf{G} = (\mathcal{V}, \mathcal{A}, \mathcal{E})$$

$$\mathcal{L} = \mathcal{A} \cup \mathcal{E}$$

$\mathcal{A} = \emptyset$ – *undirected* graph; $\mathcal{E} = \emptyset$ – *directed* graph.

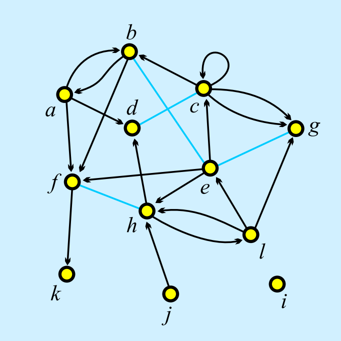
Pajek: local: `GraphSet`; `TinaSet`;
WWW: `GraphSet` / `net`; `TinaSet` / `net`, picture `picture`.

Graph / Sets – NET




```
*Vertices   12
   1 "a"       0.1020   0.3226
   2 "b"       0.2860   0.0876
   3 "c"       0.5322   0.2304
   4 "d"       0.3259   0.3917
   5 "e"       0.5543   0.4770
   6 "f"       0.1552   0.6406
   7 "g"       0.8293   0.3249
   8 "h"       0.4479   0.6866
   9 "i"       0.8204   0.8203
  10 "j"       0.4789   0.9055
  11 "k"       0.1175   0.9032
  12 "l"       0.7095   0.6475
*Arcs
   1  2
   2  1
   1  4
   1  6
   2  6
   3  2
   3  3
   3  7
   3  7
   5  3
   5  6
   5  8
   6 11
   8  4
  10  8
  12  5
  12  7
   8 12
  12  8
*Edges
   2  5
   3  4
   5  7
   6  8
```

Graph / Matrix – `MAT`

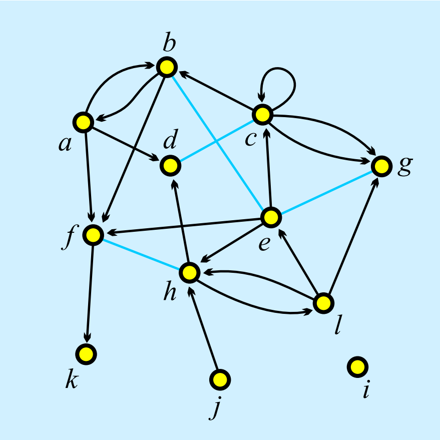


	a	b	c	d	e	f	g	h	i	j	k	l
a	0	1	0	1	0	1	0	0	0	0	0	0
b	1	0	0	0	1	1	0	0	0	0	0	0
c	0	1	1	1	0	0	2	0	0	0	0	0
d	0	0	1	0	0	0	0	0	0	0	0	0
e	0	1	1	0	0	1	1	1	0	0	0	0
f	0	0	0	0	0	0	0	1	0	0	1	0
g	0	0	0	0	1	0	0	0	0	0	0	0
h	0	0	0	1	0	1	0	0	0	0	0	1
i	0	0	0	0	0	0	0	0	0	0	0	0
j	0	0	0	0	0	0	0	1	0	0	0	0
k	0	0	0	0	0	0	0	0	0	0	0	0
l	0	0	0	0	1	0	1	1	0	0	0	0

Pajek: local: `GraphMat`; `TinaMat`, picture `picture`;
WWW: `GraphMat` / `net`; `TinaMat` / `net`, `paj`.

Graph G is *simple* if in the corresponding matrix all entries are 0 or 1.

Graph / Matrix – `MAT`




```
*Vertices   12
   1 "a"     0.1020   0.3226
   2 "b"     0.2860   0.0876
   3 "c"     0.5322   0.2304
   4 "d"     0.3259   0.3917
   5 "e"     0.5543   0.4770
   6 "f"     0.1552   0.6406
   7 "g"     0.8293   0.3249
   8 "h"     0.4479   0.6866
   9 "i"     0.8204   0.8203
  10 "j"     0.4789   0.9055
  11 "k"     0.1175   0.9032
  12 "l"     0.7095   0.6475
*Matrix
  0  1  0  1  0  1  0  0  0  0  0  0
  1  0  0  0  1  1  0  0  0  0  0  0
  0  1  1  1  0  0  2  0  0  0  0  0
  0  0  1  0  0  0  0  0  0  0  0  0
  0  1  1  0  0  1  1  1  0  0  0  0
  0  0  0  0  0  0  0  1  0  0  1  0
  0  0  0  0  1  0  0  0  0  0  0  0
  0  0  0  1  0  1  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  1  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  1  0  1  1  0  0  0  0
```

Node Properties / CLU, VEC, PER

All three types of files have the same structure:

`*vertices` n	n is the number of nodes
v_1	node 1 has value v_1
...	
v_n	

CLUstering – partition of nodes – *nominal* or *ordinal* data about nodes
$v_i \in \mathbb{N}$: node i belongs to the cluster/group v_i;

VECtor – *numeric* data about nodes
$v_i \in \mathbb{R}$: the property has value v_i on node i;

PERmutation – *ordering* of nodes
$v_i \in \mathbb{N}$: node i is at the v_i-th position.

When collecting the network data consider to provide as much properties as possible.

Example: Wolfe Monkey Data

inter.net
```
*Vertices  20
  1 "m01"
  2 "m02"
  3 "m03"
  4 "m04"
  5 "m05"
  6 "f06"
  7 "f07"
  8 "f08"
  9 "f09"
 10 "f10"
 11 "f11"
 12 "f12"
 13 "f13"
 14 "f14"
 15 "f15"
 16 "f16"
 17 "f17"
 18 "f18"
 19 "f19"
 20 "f20"
*Edges
  1   2   2
  1   3  10
  1   4   4
```

inter.net
```
1   6   5
1   7   9
1   8   7
1   9   4
1  10   3
1  11   3
1  12   7
1  13   3
1  14   2
1  15   5
1  16   1
1  17   4
1  18   1
2   3   5
2   4   1
2   5   3
2   6   1
2   7   4
2   8   2
2   9   6
2  10   2
2  11   5
2  12   4
2  13   3
2  14   2
...
```

sex.clu
```
*vertices 20
1
1
1
1
1
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
```

age.vec
```
*vertices 20
15
10
10
 8
 7
15
 5
11
 8
 9
16
10
14
 5
 7
11
 7
 5
15
 4
```

rank.per
```
*vertices 20
 1
 2
 3
 4
 5
10
11
 6
12
 9
 7
 8
18
19
20
13
14
15
16
17
```

Important note: 0 is not allowed as node number.

Pajek's Project File / `PAJ`

All types of data can be combined into a single file – Pajek's *project* file *file*`.paj`.

The easiest way to do this is:

- read all data files in Pajek,
- compute some additional data,
- delete (dispose) some data,
- save all as a project file with `File/Pajek Project File/Save`.

Next time you can restore everything with a single `File/Pajek Project File/Read`.

Wolfe network as a (Pajek's project file).

Sources of networks

Indexes and repositories

- ICON - The Colorado Index of Complex Networks
- Netzschleuder
- Network Data Repository
- Kaggle
- UCI Network Data Repository
- KONECT - Koblenz Network Collection
- Bayesys
- Stanford Large Network Dataset Collection
- Pajek datasets
-

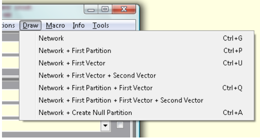
Representations of properties

Properties of nodes $\mathcal{P}$ and links $\mathcal{W}$ can be measured in different scales: numerical, ordinal and nominal. They can be *input* as data or *computed* from the network.

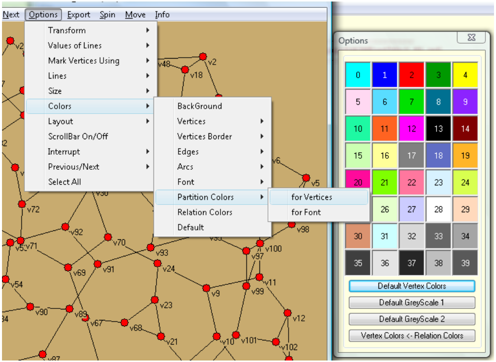
In **Pajek** numerical properties of nodes are represented by *vector*s, nominal properties by *partitions* or as *label*s of nodes. Numerical property can be displayed as *size* (width and height) of node (figure), as its *coordinate*; and a nominal property as *color* or *shape* of the figure, or as a node's *label* (content, size and color).

We can assign in **Pajek** numerical values to links. They can be displayed as *value*, *thickness* or *grey level*. Nominal vales can be assigned as *label*, *color* or *line pattern* (see Pajek manual, section ***4.3***).

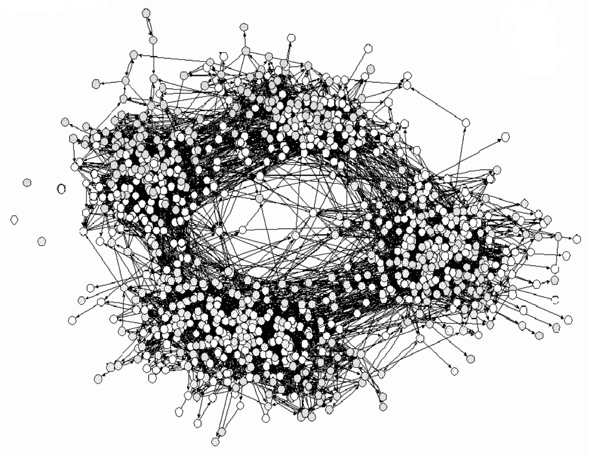
Some related operations




```
Operations/Network+Vector/Transform/Put C
Network/Create Vector/Get Coordinate
[Draw] Options
[Draw] Layout/Energy/Kamada-Kawai/Free
[Draw] Export/2D/EPS-PS
```



"Countryside" school district



James Moody (2001) AJS Vol 107, 3,679–716, friendship relation

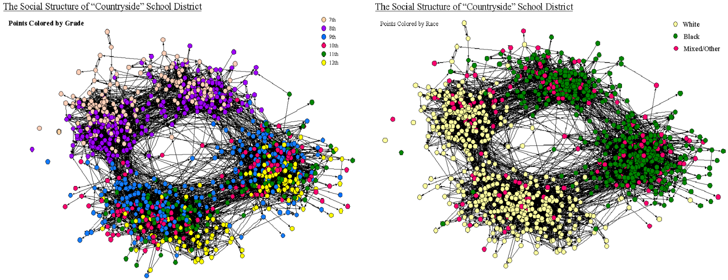
Only small or sparse networks can be displayed readably.

On large networks graph drawing algorithms can reveal their overall structure.

Can we explain the obtained structure?

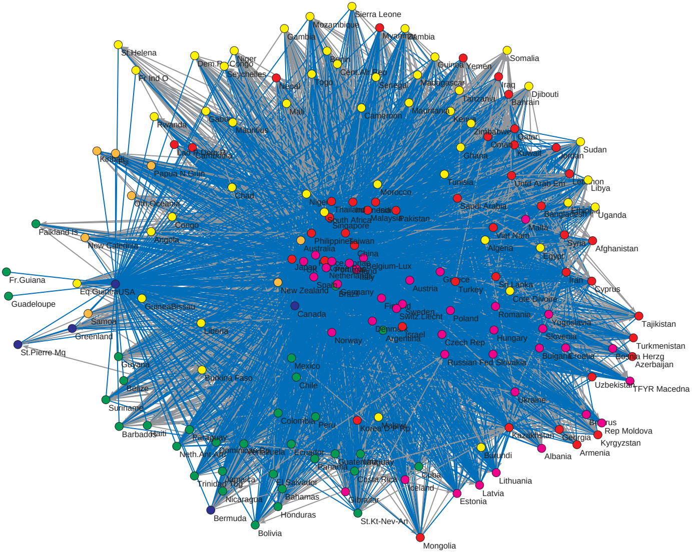
Visualization: initial network exploration, reporting results, story telling.

Display of properties – school (Moody)

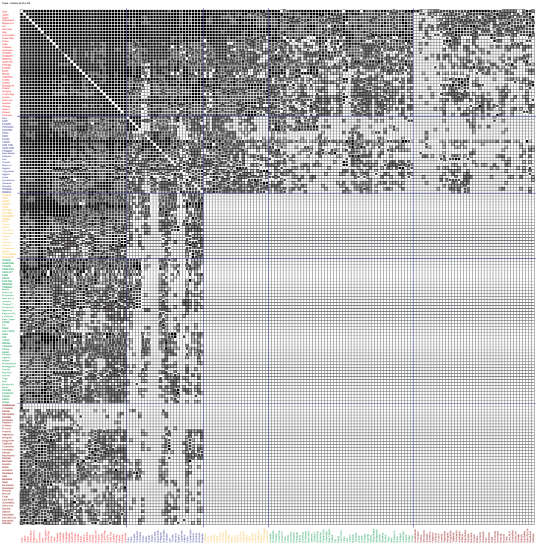


Besides the graph, we need data to understand the network!

A display of World Trade 1999 network



Matrix display of World Trade 1999 network



World trade 1999 clustering

Feenstra, RC, Lipsey, RE, Deng, H, Ma, AC, Mo, H: World Trade Flows: 1962-2000; info → weights in 1000 USD.

```
read World Trade 1999 Pajek project file
button Info network
Network/Info/Line values
Network/Create new network/Transform/Line values/Ln
Network/Create Vector/Centrality/Weighted degree/All
Vector/Make permutation
Permutation/Mirror permutation
File/Network/Export as matrix/EPS/Using permutation

Cluster/Create complete cluster
Operations/Network+cluster/Dissimilarity*/Network based/
   d5 [1]
manually reorder and close hierarchy
Hierarchy/Make partition
Hierarchy/Make permutation
select ln network as the First
File/Network/Export as matrix/EPS/Using
   permutation+partition
```

Types of networks

Besides ordinary (directed, undirected, mixed) networks some extended types of networks are also used:

- *2-mode networks*, bipartite (valued) graphs – networks between two disjoint sets of nodes $\mathbf{N} = ((\mathcal{U}, \mathcal{V}), \mathcal{L}, \mathcal{P}, \mathcal{W})$
- *multi-relational networks* $\mathbf{N} = (\mathcal{V}, (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k), \mathcal{P}, \mathcal{W})$
- *temporal networks*, dynamic graphs – networks changing over time $\mathbf{N}_T = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W}, T)$
- specialized networks: representation of genealogies as *p-graphs*; *Petri's nets*, ...

The network (input) file formats should provide means to express all these types of networks. All interesting data should be recorded (respecting privacy).

Pictures in SVG: ***66 days***.

Multi-relational temporal network – KEDS/WEIS

```
% Recoded by WEISmonths, Sun Nov 28 21:57:00 2004
% from http://www.ku.edu/~keds/data.dir/balk.html
*vertices 325
1 "AFG" [1-*]
2 "AFR" [1-*]
3 "ALB" [1-*]
4 "ALBMED" [1-*]
5 "ALG" [1-*]
 ...
318 "YUGGOV" [1-*]
319 "YUGMAC" [1-*]
320 "YUGMED" [1-*]
321 "YUGMTN" [1-*]
322 "YUGSER" [1-*]
323 "ZAI" [1-*]
324 "ZAM" [1-*]
325 "ZIM" [1-*]
*arcs :0 "*** ABANDONED"
*arcs :10 "YIELD"
*arcs :11 "SURRENDER"
*arcs :12 "RETREAT"
 ...
*arcs :223 "MIL ENGAGEMENT"
*arcs :224 "RIOT"
*arcs :225 "ASSASSINATE TORTURE"
*arcs
224: 314 153 1 [4]                890402 YUG    KSV    224  (RIOT) RIOT-TORN
212: 314 83 1 [4]                 890404 YUG    ETHALB 212  (ARREST PERSON) ALB ETHNI
224: 3 83 1 [4]                   890407 ALB    ETHALB 224  (RIOT) RIOTS
123: 83 153 1 [4]                 890408 ETHALB KSV    123  (INVESTIGATE) PROBING
 ...
42: 105 63 1 [175]                030731 GER    CYP    042  (ENDORSE) GAVE SUPPORT
212: 295 35 1 [175]               030731 UNWCT  BOSSER 212  (ARREST PERSON) SENTENCED
43: 306 87 1 [175]                030731 VAT    EUR    043  (RALLY) RALLIED
13: 295 35 1 [175]                030731 UNWCT  BOSSER 013  (RETRACT) CLEARED
121: 295 22 1 [175]               030731 UNWCT  BAL    121  (CRITICIZE) CHARGES
122: 246 295 1 [175]              030731 SER    UNWCT  122  (DENIGRATE) TESTIFIED
121: 35 295 1 [175]               030731 BOSSER UNWCT  121  (CRITICIZE) ACCUSED
```

Kansas Event Data System *KEDS*

Important nodes and links in network

To identify important / interesting elements (nodes, links) in a network we often try to express our intuition about important / interesting element using an appropriate measure (index, weight) following the scheme

> *larger is the measured value of an element,*
> *more important / interesting is this element*

Too often, in the analysis of networks, researchers uncritically pick some measures from the literature. For formal approach see Roberts.

It seems that the most important distinction between different node *indices* is based on the view/decision of whether the network is considered directed or undirected.

Network properties

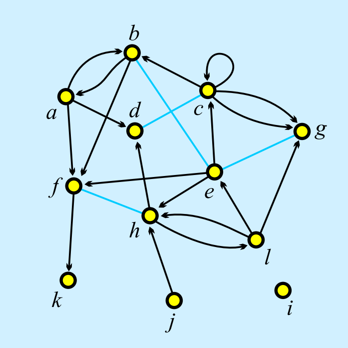
The (importance) measure can be obtained as input data – an *observed* property, or computed from the network description – a *structural* property.

An interesting question is studying associations among structural and observed properties. Can an observed property be explained with some structural property/ies?

measuring, computing, projections of two-mode networks

normalization of weights

Degrees



degree of node v, $\deg(v)$ = number of links with v as an endnode;
indegree of node v, $\mathrm{indeg}(v)$ = number of links with v as a terminal node (endnode is both initial and terminal);
outdegree of node v, $\mathrm{outdeg}(v)$ = number of links with v as an initial node.
initial node $v \Leftrightarrow \mathrm{indeg}(v) = 0$
terminal node $v \Leftrightarrow \mathrm{outdeg}(v) = 0$

$$n = 12,\ m = 23,\ \mathrm{indeg}(e) = 3,\ \mathrm{outdeg}(e) = 5,\ \deg(e) = 6$$

$$\sum_{v\in\mathcal{V}} \mathrm{indeg}(v) = \sum_{v\in\mathcal{V}} \mathrm{outdeg}(v) = |\mathcal{A}| + 2|\mathcal{E}| - |\mathcal{E}_0|,\ \sum_{v\in\mathcal{V}} \deg(v) = 2|\mathcal{L}| - |\mathcal{L}_0|$$

Hubs and authorities

To each node v of a network $\mathbf{N} = (\mathcal{V}, \mathcal{L})$ we assign two values: quality of its content (*authority*) x_v and quality of its references (*hub*) y_v.

A good authority is selected by good hubs; and good hub points to good authorities (see Kleinberg).

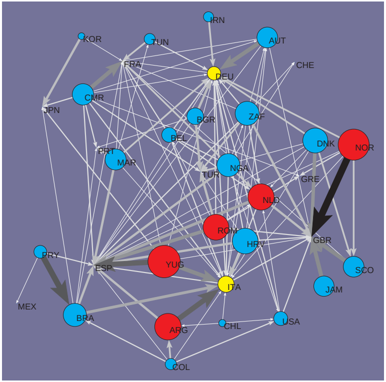
$$x_v = \sum_{u:(u,v)\in\mathcal{L}} y_u \qquad \text{and} \qquad y_v = \sum_{u:(v,u)\in\mathcal{L}} x_u$$

Let $\mathbf{W}$ be a matrix of network $\mathbf{N}$ and $\mathbf{x}$ and $\mathbf{y}$ authority and hub vectors. Then we can write these two relations as $\mathbf{x} = \mathbf{W}^T\mathbf{y}$ and $\mathbf{y} = \mathbf{W}\mathbf{x}$.

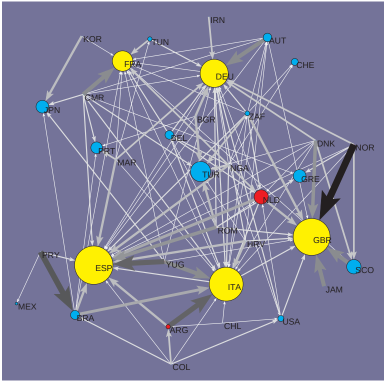
We start with $\mathbf{y} = [1, 1, \ldots, 1]$ and then compute new vectors $\mathbf{x}$ and $\mathbf{y}$. After each step we normalize both vectors. We repeat this until they stabilize.

We can show that this procedure converges. The limit vector $\mathbf{x}^*$ is the principal eigen vector of matrix $\mathbf{W}^T\mathbf{W}$; and $\mathbf{y}^*$ of matrix $\mathbf{W}\mathbf{W}^T$.

. . . Hubs and authorities: football



Exporters (hubs)



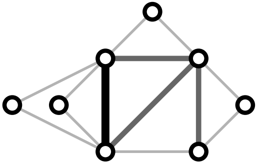
Importers (authorities)

Example: Krebs, Krempl. World Cup 1998 in Paris, 22 national teams. A player from first country is playing in the second country.

Important links – Weighted networks

Obtaining weights

Triangular network



Let $\mathbf{G}$ be a simple undirected graph. A *triangular network* $\mathbf{N}_T(\mathbf{G}) = (\mathcal{V}, \mathcal{E}_T, w)$ determined by $\mathbf{G}$ is a subgraph $\mathbf{G}_T = (\mathcal{V}, \mathcal{E}_T)$ of $\mathbf{G}$ which set of edges $\mathcal{E}_T$ consists of all triangular edges of $\mathcal{E}(\mathbf{G})$. For $e \in \mathcal{E}_T$ the weight $w(e)$ equals to the number of different triangles in $\mathbf{G}$ to which e belongs.

Triangular networks can be used to efficiently identify dense clique-like parts of a graph. If an edge e belongs to a k-clique in $\mathbf{G}$ then $w(e) \geq k - 2$.

```
Network/Create New Network/with Ring Counts/3-Rings
```

Projections of two-mode networks

Normalizations

In networks obtained from large two-mode networks, there are often huge differences in weights. Therefore it is not possible to compare the vertices according to the raw data. First, we have to *normalize* the network to make the weights comparable.

There exist several ways how to do this. Some of them are presented in the following table. They can be used also on other networks.

In the case of networks without loops, we define the diagonal weights for undirected networks as the sum of out-diagonal elements in the row (or column) $w_{vv} = \sum_u w_{vu}$ and for directed networks as some mean value of the row and column sum, for example $w_{vv} = \frac{1}{2}(\sum_u w_{vu} + \sum_u w_{uv})$. Usually, we assume that the network does not contain any isolated node.

. . . Normalizations

$$\mathrm{Geo}_{uv} = \frac{w_{uv}}{\sqrt{w_{uu} w_{vv}}} \qquad \mathrm{GeoDeg}_{uv} = \frac{w_{uv}}{\sqrt{\deg_u \deg_v}}$$

$$\mathrm{Input}_{uv} = \frac{w_{uv}}{w_{vv}} \qquad \mathrm{Output}_{uv} = \frac{w_{uv}}{w_{uu}}$$

$$\mathrm{Min}_{uv} = \frac{w_{uv}}{\min(w_{uu}, w_{vv})} \qquad \mathrm{Max}_{uv} = \frac{w_{uv}}{\max(w_{uu}, w_{vv})}$$

$$\mathrm{MinDir}_{uv} = \begin{cases} \frac{w_{uv}}{w_{uu}} & w_{uu} \leq w_{vv} \\ 0 & otherwise \end{cases} \qquad \mathrm{MaxDir}_{uv} = \begin{cases} \frac{w_{uv}}{w_{vv}} & w_{uu} \leq w_{vv} \\ 0 & otherwise \end{cases}$$

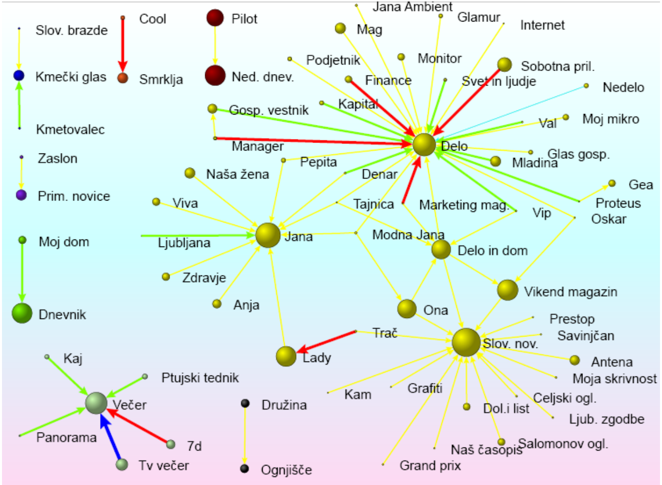
After a selected normalization the important parts of network are obtained by link-cuts or islands approaches.

```
Network/2-Mode Network/2-Mode to 1-Mode/Normalize 1-Mode/
```

Reuters Terror News: GeoDeg, MaxDir, MinDir.

Slovenian journals and magazins.

MinDir of Slovenian journals 2000



Over 100000 people were asked in the years 1999 and 2000 about the journals they read. They mentioned 124 different journals. (source Cati)

Size of network

The size of a network/graph is expressed by two numbers: number of nodes $n = |\mathcal{V}|$ and number of links $m = |\mathcal{L}|$.

In a *simple undirected* graph (no parallel edges, no loops) $m \leq \frac{1}{2}n(n-1)$; and in a *simple directed* graph (no parallel arcs) $m \leq n^2$.

Small networks (some tens of nodes) – can be represented by a picture and analyzed by many algorithms (***UCINET***, ***NetMiner***).

Also, *middle size* networks (some hundreds of nodes) can still be represented by a picture (!?), but some analytical procedures can't be used.

Till 1990 most networks were small – they were collected by researchers using surveys, observations, archival records, . . . The advances in IT allowed to create networks from the data already available in the computer(s). *Large* networks became reality. Large networks are too big to be displayed in detail; special

Large networks

Large network – several thousands or millions of nodes. Can be stored in the computer's memory – otherwise *huge* network. 64-bit computers!

Jure Leskovec: SNAP – Stanford Large Network Dataset Collection

Social networks

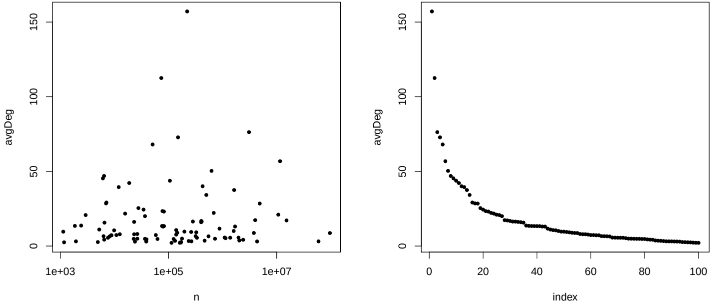
Name	Type	Nodes	Edges	Description
ego-Facebook	Undirected	4,039	88,234	Social circles from Facebook (anonymized)
ego-Gplus	Directed	107,614	13,673,453	Social circles from Google+
ego-Twitter	Directed	81,306	1,768,149	Social circles from Twitter
soc-Epinions1	Directed	75,879	508,837	Who-trusts-whom network of Epinions.com
soc-LiveJournal1	Directed	4,847,571	68,993,773	LiveJournal online social network
soc-Pokec	Directed	1,632,803	30,622,564	Pokec online social network
soc-Slashdot0811	Directed	77,360	905,468	Slashdot social network from November 2008
soc-Slashdot0922	Directed	82,168	948,464	Slashdot social network from February 2009
wiki-Vote	Directed	7,115	103,689	Wikipedia who-votes-on-whom network

Networks with ground-truth communities

Name	Type	Nodes	Edges	Communities	Description
com-LiveJournal	Undirected, Communities	3,997,962	34,681,189	287,512	LiveJournal online social network
com-Friendster	Undirected, Communities	65,608,366	1,806,067,135	957,154	Friendster online social network
com-Orkut	Undirected, Communities	3,072,441	117,185,083	6,288,363	Orkut online social network

Dunbar's number

Average degrees of the SNAP and Konect networks



Average degree $\bar{d} = \frac{1}{n}\sum_{v \in V} \deg(v) = \frac{2m}{n}$. Most real-life large networks are *sparse* – the number of nodes and links are of the same order. This property is also known as a Dunbar's number.

The basic idea is that if each node has to spend for each link a certain amount of "energy" to maintain the links to selected other nodes then since it has a limited "energy" at its disposal, the number of links should be limited. In human networks, Dunbar's number is between 100 and 150.

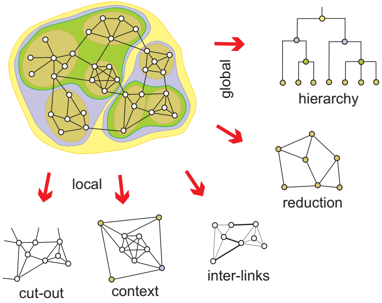
Complexity of algorithms

Let us look to time complexities of some typical algorithms:

	T(n)	1.000	10.000	100.000	1.000.000	10.000.000
LinAlg	O(n)	0.00 s	0.015 s	0.17 s	2.22 s	22.2 s
LogAlg	O($n \log n$)	0.00 s	0.06 s	0.98 s	14.4 s	2.8 m
SqrtAlg	O($n\sqrt{n}$)	0.01 s	0.32 s	10.0 s	5.27 m	2.78 h
SqrAlg	O(n^2)	0.07 s	7.50 s	12.5 m	20.8 h	86.8 d
CubAlg	O(n^3)	0.10 s	1.67 m	1.16 d	3.17 y	3.17 ky

For the interactive use on large graphs already quadratic algorithms, O(n^2), are too slow.

Approaches to large networks



In the analysis of a *large* network (several thousands or millions of nodes, the network can be stored in computer memory) we can't display it in its totality; also there are only a few algorithms available.

To analyze a large network we can use a statistical approach or we can identify smaller (sub) networks that can be analyzed further using more sophisticated methods.

Statistics

Measured properties – Input data

- numeric → **`vector`**
- ordinal → **`permutation`**
- nominal → **`clustering`** (partition)

Computed properties

global: number of nodes, edges/arcs, components; maximum core number, ...

local: degrees, cores, indices (betweeness, hubs, authorities, ...)

inspections: partition, vector, values of lines, ...

Associations between computed (structural) data and input (measured) data.

. . . Statistics

The global computed properties are reported by **Pajek**'s commands or can be seen using the **Info** option. In *repetitive* commands they are stored in vectors.

The local properties are computed by **Pajek**'s commands and stored in vectors or partitions. To get information about their distribution use the **Info** option.

As an example, let us look at The Edinburgh Associative Thesaurus network. The EAT is a network of word associations collected from subjects (students). The weight on the arcs is the count of word associations.

```
File/Network/Read  EATnewSR.net
Info/Network/General
```

It has 23219 nodes and 325593 arcs (564 loops); number of links with value=1 is 227459; and Average Degree = 28.045 .

. . . Statistics

To identify the nodes with the largest degree:

```
Network/Create vector/Centrality/Degree/All
button Info/Vector  +10
```

The largest degrees have the nodes:

	vertex	deg	label
1	12720	1108	ME
2	12459	1074	MAN
3	8878	878	GOOD
4	18122	875	SEX
5	13793	803	NO
6	13181	799	MONEY
7	23136	732	YES
8	15080	723	PEOPLE
9	13948	720	NOTHING
10	22973	716	WORK

What about indegree and outdegree?

Distribution of weights

```
Network/Info/Line values
```

. . . weighted indegrees

To identify the words selected by the largest number of respondents we look to their weighted indegrees

```
Network/Create vector/Centrality/Weighted degree/In
button Info/Vector  +20
```

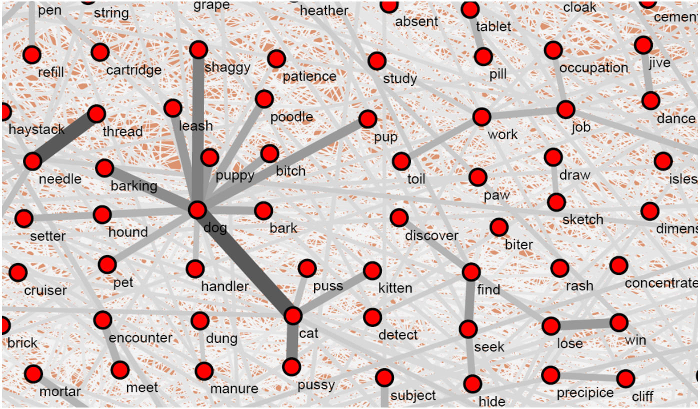
The result is:

	wideg	word	wideg	word	
1	4387	MONEY	11	2019	TREE
2	3299	WATER	12	1988	GOOD
3	2918	FOOD	13	1972	HOUSE
4	2515	ME	14	1896	BIRD
5	2435	MAN	15	1891	UP
6	2434	CAR	16	1881	CHURCH
7	2224	SEA	17	1802	TIME
8	2154	SEX	18	1795	FIRE
9	2100	HORSE	19	1762	SHIP
10	2073	DOG	20	1722	MUSIC

Big picture of EAT at Ars Electronica 2004

Part of the big picture of EAT

The glasses in this case are based on ordering the edges in increasing order of their values and drawing them in this order – stronger edges cover the weaker ones. The picture emphasizes the strongest substructures; the remaining elements form a background.



Statistics with **Pajek** and R

Pajek supports interaction with statistical program R and the use of other external programs as tools (menu `Tools`).

In **Pajek** we determine the degrees of nodes and submit them to R

```
Network/Create Vector/Centrality/Degree/All
Tools/R/Send to R/Current Vector
```

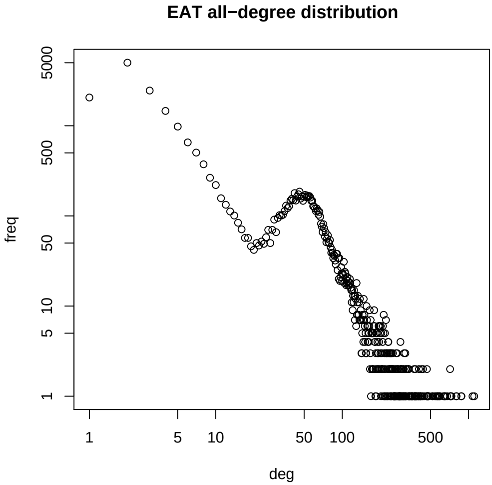
In R we determine their distribution and plot it

```
summary(v2)
t <- table(v2)
x<-as.numeric(names(t))
plot(x,t,log='xy',main='degree distribution',
  xlab='deg',ylab='freq')
```

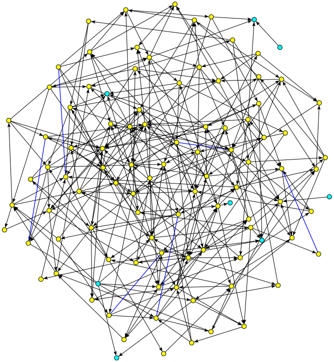
The obtained picture can be saved with `File/Save as` in selected format (PDF or PS for LaTeX; Windows metafile format for inclusion in Word; SVG for WWW).

Attention! The nodes of degree 0 make problems with `log='xy'`.

EAT all-degree distribution



Erdős and Renyi's random graphs



Erdős and Renyi defined a *random graph* as follows: every possible link is included in a graph with a given probability p.

In **Pajek** instead of probability p a more intuitive average degree is used

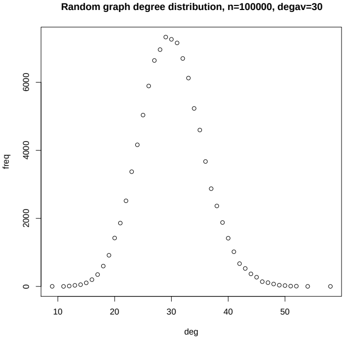
$$\overline{\deg} = \frac{1}{n} \sum_{v \in \mathcal{V}} \deg(v)$$

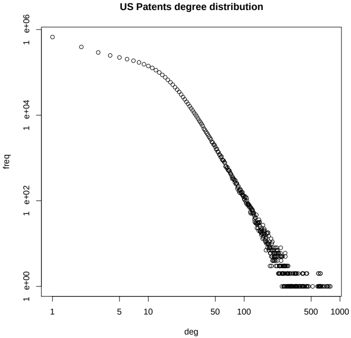
`Network/Create Random Network/Bernoulli/Poisson/Undirected/General [100] [3]`

It holds $p = \frac{m}{m_{max}}$ and, for simple graphs, also $\overline{\deg} = \frac{2m}{n}$.

The random graph in the picture has 100 nodes and an average degree 3.

Degree distribution





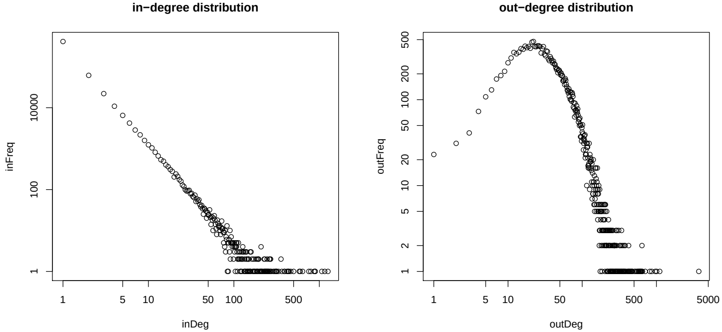
Real-life networks are usually not random in the Erdős/Renyi sense. The analysis of their distributions gave a new view about their structure – Watts (Small worlds), Barabási (nd/networks, Linked).

in/out-degree distributions

We read in **Pajek** the Centrality citation network `cite.net`. First, we remove loops and multiple links. Then we determine the indegrees and outdegrees and call R from **Pajek** submitting all vectors.

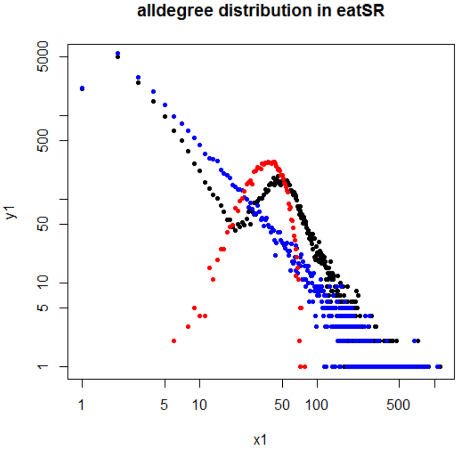
```
########################################
 R called from Pajek
 The following vectors read:
 v1   :  Input Degree of N1 (548600)
 v2   :  Output Degree of N1 (548600)
 v3   :  All Degree of N1 (548600)
------------------------------------------------------------
> inTab <- table(v1)
> indeg <- as.integer(names(inTab))
> inDeg <- indeg[indeg>0]
> inFreq <- as.vector(inTab[indeg>0])
> plot(inDeg,inFreq,log='xy',main="in-degree distribution")
> ouTab <- table(v2)
> outdeg <- as.integer(names(ouTab))
> outDeg <- outdeg[outdeg>0]
> outFreq <- as.vector(ouTab[outdeg>0])
> plot(outDeg,outFreq,log='xy',main="out-degree distribution")
```

in/out-degree distributions



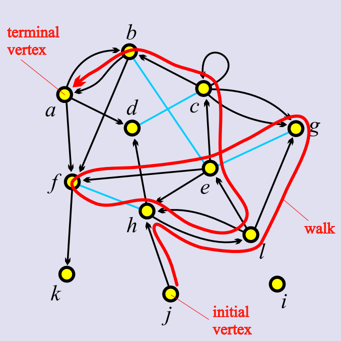
The in-degree distribution is "scale-free"-like – it follows the *power law* *inFreq* $= k \cdot$ *inDeg*$^{-\alpha}$. The parameters *k* and α can be determined using the package of Clauset, Shalizi and Newman. See also Stumpf, et al.: Critical Truths About Power Laws.

EAT all/in/out-degree distributions



It is important to consider the direction of links!

Walks



length $|s|$ of the walk s is the number of links it contains.

$s = (j, h, l, g, e, f, h, l, e, c, b, a)$

$|s| = 11$

A walk is *closed* iff its initial and terminal node coincide.

If we don't consider the direction of the links in the walk we get a *semiwalk* or *chain*.

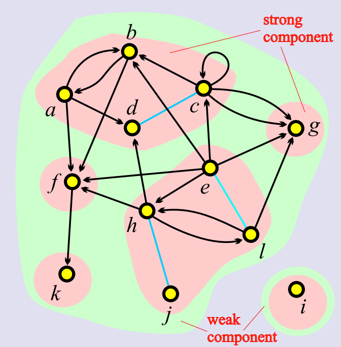
trail – walk with all links different

path – walk with all nodes different

cycle – closed walk with all internal nodes different

A graph is *acyclic* if it doesn't contain any cycle.

Connectivity



Node u is *reachable* from node v iff there exists a walk with initial node v and terminal node u.

Node v is *weakly connected* with node u iff there exists a semiwalk with v and u as its end-nodes.

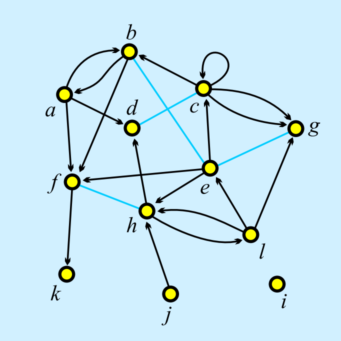
Node v is *strongly connected* with node u iff they are mutually reachable.

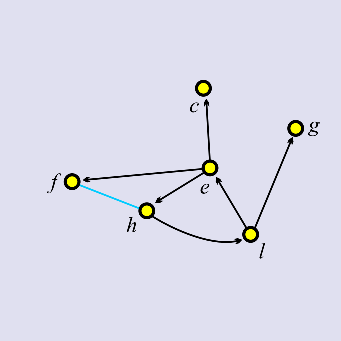
Weak and strong connectivity are equivalence relations.

Equivalence classes induce weak/strong *components*.

`Network/Create Partition/Components/`

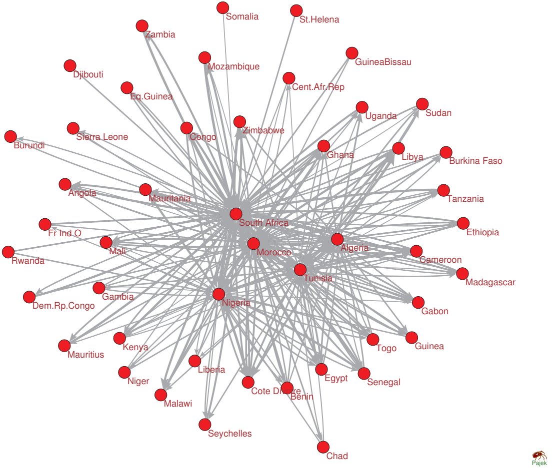
Subgraph





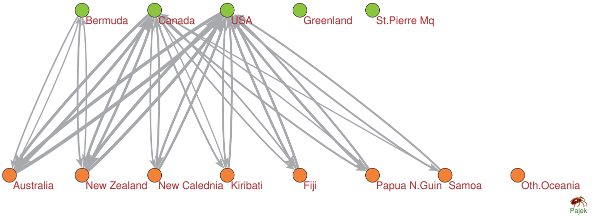
A *subgraph* $\mathbf{H} = (\mathcal{V}', \mathcal{L}')$ of a given graph $\mathbf{G} = (\mathcal{V}, \mathcal{L})$ is a graph which set of links is a subset of set of links of $\mathbf{G}$, $\mathcal{L}' \subseteq \mathcal{L}$, its node set is a subset of set of nodes of $\mathbf{G}$, $\mathcal{V}' \subseteq \mathcal{V}$, and it contains all endnodes of $\mathcal{L}'$.
A subgraph can be *induced* by a given subset of nodes or links. It is a *spanning* subgraph iff $\mathcal{V}' = \mathcal{V}$.
To obtain a *subnetwork* also the properties/weights have to be restricted to $\mathcal{V}'$ and $\mathcal{L}'$).

Cut-out – induced subgraph: World trade Africa




```
select Continet 1999 partition as First
Operations/Network+Partition/Extract/Subnetwork [6]
```

Cut-out: World trade North America : Oceania




```
Operations/Network + Partition/Extract/Subnetwork   [2,7]
Operations/Network + Partition/Transform/Remove lines/
  Inside clusters    [2,7]
```

The nodes can be manually put on a rectangular grid produced by

```
[Draw] Move/Grid
```

Cuts

The standard approach to find interesting groups inside a network is based on properties/weights – they can be *measured* or *computed* from network structure.

The *node-cut* of a network $\mathbf{N} = (\mathcal{V}, \mathcal{L}, p)$, $p : \mathcal{V} \to \mathbb{R}$, at selected level t is a subnetwork $\mathbf{N}(t) = (\mathcal{V}', \mathcal{L}(\mathcal{V}'), p)$, determined by the set

$$\mathcal{V}' = \{v \in \mathcal{V} : p(v) \geq t\}$$

and $\mathcal{L}(\mathcal{V}')$ is the set of links from $\mathcal{L}$ that have both endnodes in $\mathcal{V}'$.

The *link-cut* of a network $\mathbf{N} = (\mathcal{V}, \mathcal{L}, w)$, $w : \mathcal{L} \to \mathbb{R}$, at selected level t is a subnetwork $\mathbf{N}(t) = (\mathcal{V}(\mathcal{L}'), \mathcal{L}', w)$, determined by the set

$$\mathcal{L}' = \{e \in \mathcal{L} : w(e) \geq t\}$$

and $\mathcal{V}(\mathcal{L}')$ is the set of all endnodes of the links from $\mathcal{L}'$.

Cuts in **Pajek**

The threshold value *t* is determined on the basis of distribution of values of weight *w* or property *p*. Usually we are interested in cuts that are not too large, but also not trivial.

Node-cut: *p* stored in a vector

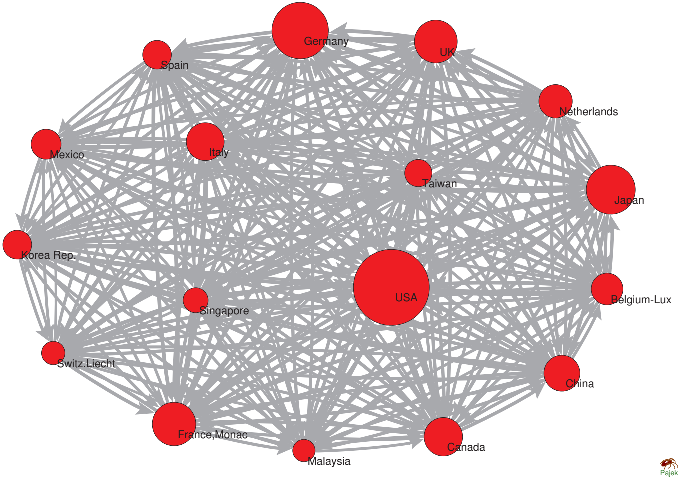
```
Vector/Info  [+30] [#20]
Vector/Make Partition/by Intervals/Selected Thresholds  [t]
Operations/Network + Partition/Extract Subnetwork  [2]
```

Link-cut: weighted network

```
Network/Info/Line values   [#20]
Network/Create New Network/Transform/Remove/Lines with Value/
  lower than   [t]
Network/Create Partition/Degree/All
Operations/Network + Partition/Extract Subnetwork  [1-⋆]
```

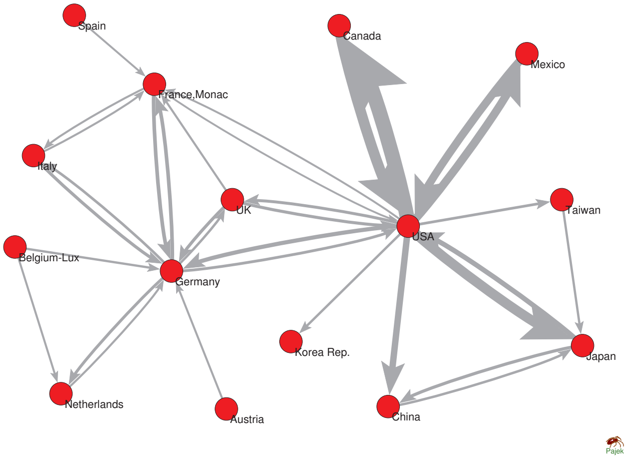
Node-cut: World trade, wdeg=150000

weights devided by 1000



Link-cut: World trade, wdeg=25000

weights devided by 1000



Skeletons

Skeletons are subnetworks in which only the most important links are preserved.

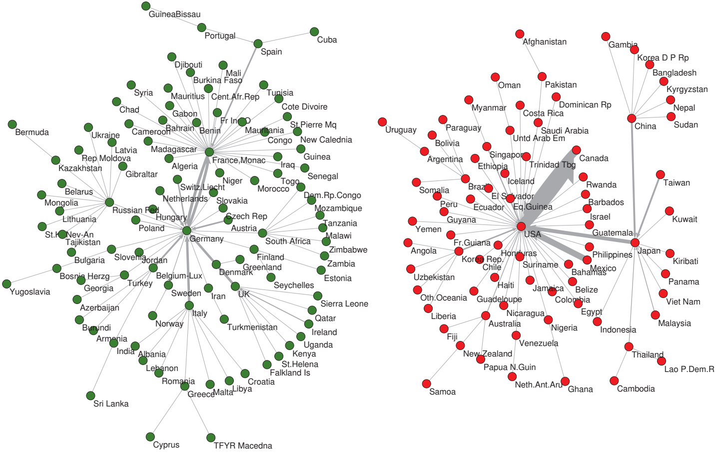
In the *top k-neighbors skeleton* in each node the k links with the highest weights are preserved.

```
Network/Create New Network/Transform/Remove/all arcs from
  each vertex except/ k with highest line values   [k]
```

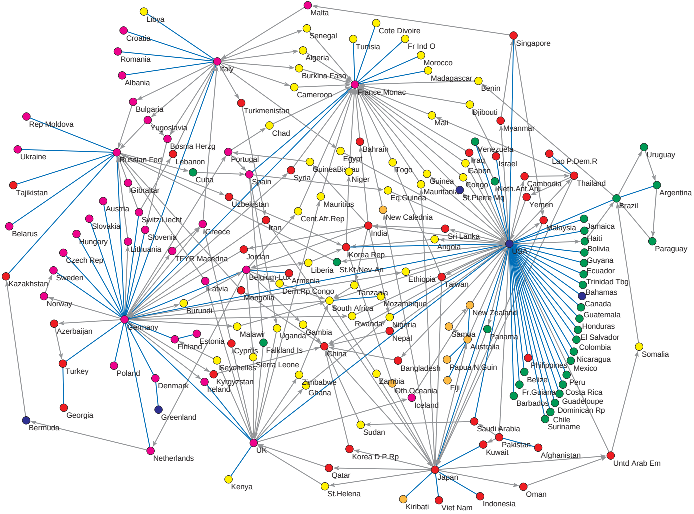
Let the weight w be a dissimilarity ($w' = 1/w$). The Pathfinder algorithm was proposed in eighties by Schvaneveldt [3] for simplification of weighted networks. It removes from the network all links that do not satisfy the triangle inequality – if for a link a shorter path exists connecting its endnodes then the link is removed. The obtained subnetwork is called the *Pathfinder skeleton*.

```
Network/Create New Network/Transform/Line values/Power [-1]
Network/Create New Network/Transform/Reduction/Pathfinder [1]
select the original network as the Second
Networks/Cross-intersection/Second
```

Nearest neighbor skeleton: World trade



Pathfinder skeleton of World Trade 1999 network



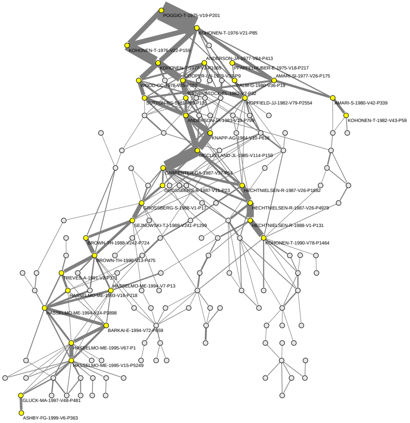
Simple analysis using cuts

We look at the components of $\mathbf{N}(t)$. Their number and sizes depend on t. Usually there are many small components. Often we consider only components of size at least k and not exceeding K. The components of size smaller than k and not containing important nodes/links are discarded as ”noninteresting”; and the components of size larger than K are cut again at some higher level.

The values of thresholds t, k and K are determined by inspecting the distribution of node/link-values and the distribution of component sizes and considering additional knowledge on the nature of network or goals of analysis.

We developed some new and efficiently computable properties/ weights.

Citation weights



The citation network analysis started in 1964 with the paper of Garfield et al. In 1989 Hummon and Doreian proposed three indices – weights of arcs that are proportional to the number of different source-sink paths passing through the arc. We developed algorithms to efficiently compute these indices.

Main subnetwork (arc-cut at level 0.007) of the SOM (selforganizing maps) citation network (4470 nodes, 12731 arcs). See paper.

Islands

Cores

Skeletons

Clustering and blockmodeling

References I

- Wouter De Nooy, Andrej Mrvar, Vladimir Batagelj: Exploratory Social Network Analysis with Pajek; Revised and Expanded Edition for Updated Software. Structural Analysis in the Social Sciences, CUP, July 2018.
- Vladimir Batagelj, Patrick Doreian, Anuška Ferligoj and Nataša Kejžar: Understanding Large Temporal Networks and Spatial Networks: Exploration, Pattern Searching, Visualization and Network Evolution. Wiley Series in Computational and Quantitative Social Science. Wiley, October 2014
- Schvaneveldt, R. W. (Ed.) (1990) Pathfinder Associative Networks: Studies in Knowledge Organization. Norwood, NJ: Ablex.
- Wikipedia: Regular language.