Introduction to Network Analysis using Pajek

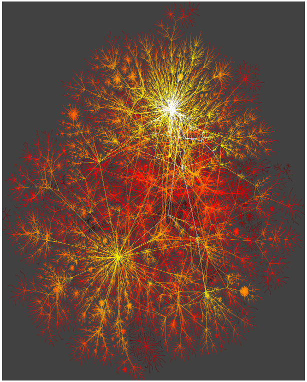
9. What else?

Vladimir Batagelj

IMFM Ljubljana and IAM Koper

Phd program on Statistics
University of Ljubljana, 2017

Outline



K. C. Claffy: Skitter data

e-mail: vladimir.batagelj@fmf.uni-lj.si
wiki: http://vladowiki.fmf.uni-lj.si/doku.php?id=pajek:ev:pde
version: May 9, 2017

Analysis of temporal networks

`Pajek` supports temporal networks from 1999.
A network can be analyzed as a whole (all time points together) or by time slices.

Time series of selected structural characteristics can be exported in statistical packages and analyzed there.

There are no specific network analysis methods for temporal networks in `Pajek` yet.

An interesting approach to analysis/visualization of temporal networks was developed by U. Brandes and his group paper, animations.
NAS: Dynamic Social Network Modeling and Analysis.

In 2014 we started to develop a Python library TQ for analysis of temporal networks based on temporal quantities.

KEDS

Standard approach:

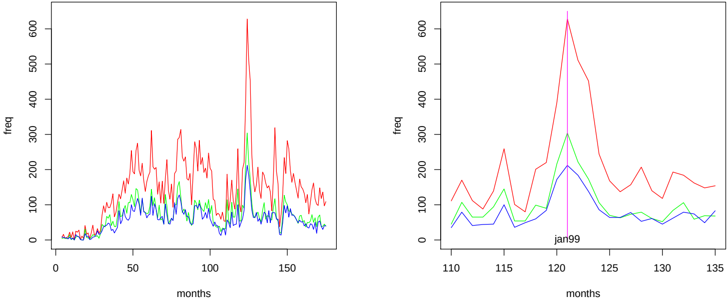
- layout of the entire network using spring embedder
- sequence of time slices
- selected relation

We get a 'rainbow'. Difficult to see something.
We decided to merge actions into 3 groups

Positive (blue)	Neutral (green)	Negative (red)
01 Yield	08 Agree	15 Demand
02 Comment	09 Request	16 Warn
03 Consult	10 Propose	17 Threaten
04 Approve	11 Reject	18 Demonstrate
05 Promise	12 Accuse	19 Reduce Relationship
06 Grant	13 Protest	20 Expel
07 Reward	14 Deny	21 Seize
		22 Force

KEDS statistics

Time changing of numbers of links. Repetitive operations !!!




```
months <- 4:175
plot(months,v3,type="l",ylim=c(0,650),ylab="freq",xlab="months",col="red")
lines(months,v2,col="green"); lines(months,v1,col="blue")

m <- 110:135
plot(m,v3[m],type="l",ylim=c(0,650),ylab="freq",xlab="months",col="red")
lines(m,v2[m],col="green"); lines(m,v1[m],col="blue")
t <- 121; lines(c(t,t),c(0,650),col="magenta"); text(t,0,"jan99")
```

Hall, B.H., Jaffe, A.B. and Tratjenberg M.: The NBER U.S. Patent Citations Data File. NBER Working Paper 8498 (2001).
http://www.nber.org/patents/

- developed between 1975 – 1999
  granted patents from January 1963 – December 1999
- 2923922 patents with text descriptions, 850846 as image
  3774768 nodes
- 16522438 citations (network arcs)

Several variables (properties of nodes) are also available: application year, assignee identifier, technological (sub)category, . . .

Shrinking of network according to categories & time slices

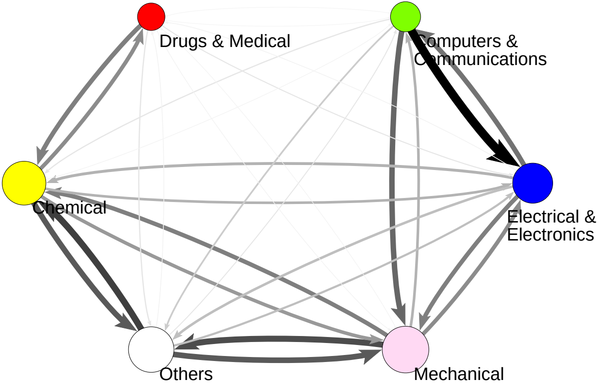
All nodes from the same category in the same time slice are shrunk in one node.
The obtained smaller networks over time are analyzed.
For looking closer to a specific segment of the network subcategories or assignee numbers can be used.

Choice of sliding time window

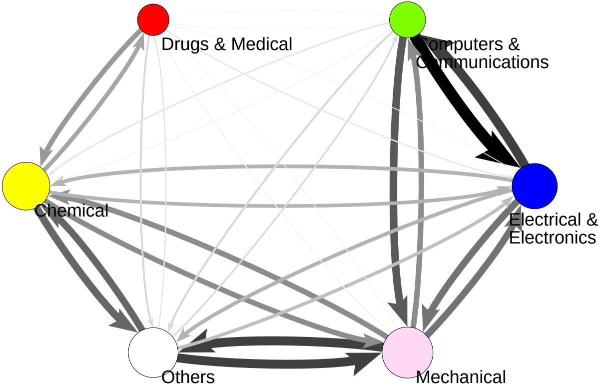
We used the knowledge about backward citation lags (Hall, Jaffe, Trajtenberg), that is the time difference between grant year of the **citing** patent and that of the **cited patents**. The highest number of cited patents were granted 3 and 4 years earlier. For even older patents the number drastically decreases.
Since application year and grant year somehow correlate, we used time slices of **4 years with no history**. All the citations lagged more than 4 years were excluded.
Possible interpretation:

- less lagged citations could be part of the research and development at current time
- other citations used as references to well known methods patented earlier

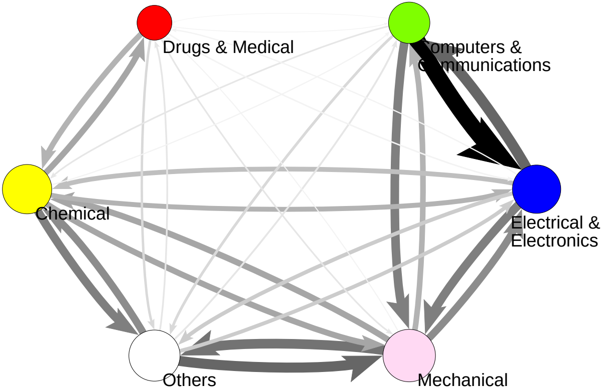
Temporal networks (1984 – 1987)



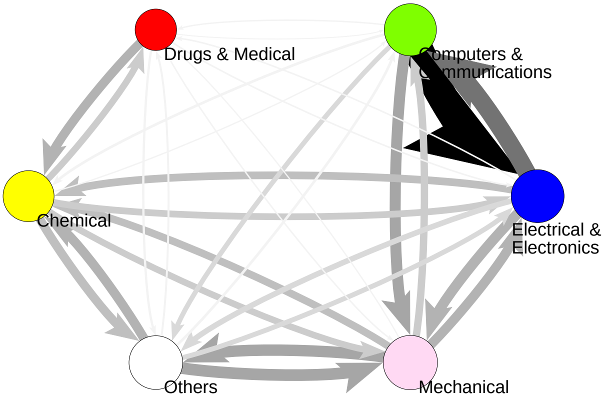
Temporal networks (1987 – 1990)



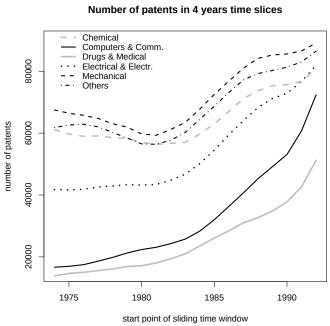
Temporal networks (1990 – 1993)

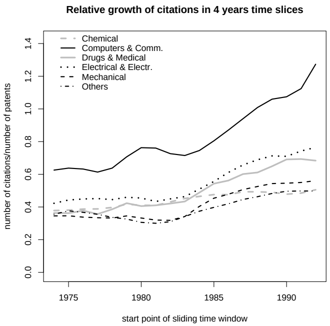


Temporal networks (1993 – 1996)

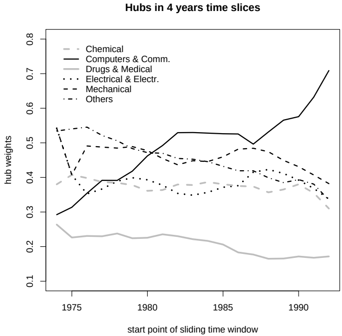


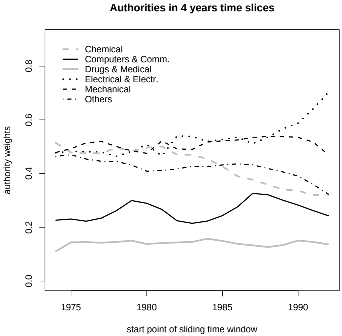
Growth of number of patents and relative growth of citations within category





Development of technological categorie
Hubs and authorities





Categories with large values of hubs (Computers & Communication and Mechanical, Others) are categories which **combine knowledge from other** important technological **categories**.
Categories with large values of authorities (Mechanical and Electrical & Electronic) play very important role in setting the foundations – **basic knowledge**.

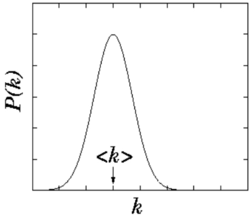
PajektoSVGanim

For movie-like 'smooth' visualization of evolution of a network through time we are developing a separate program `PajektoSVGanim` (implemented by Darko Brvar) that transforms a sequence of Pajek's layouts into a SVG animation.

Similar programs: Skye Bender-deMoll, Daniel A. McFarland, James Moody: `SoNIA` (movies, program, paper).
Peter A. Gloor: `TeCFlow` (examples, program, paper).
Franzosi

Plans: An interesting approach could be search for typical temporal patterns – stories in the network. In Pajek a pattern search is implemented for ordinary networks. For this purpose we intend to extend it also to temporal patterns.

Erdős-Rényi networks



Barabasi: The Architecture of Complexity

Paul Erdős in Alfréd Rényi introduced in 1959 the notion of random graph in which each pair of nodes is linked with a given probability p. The theory of ER random graphs is well developed (see B. Bollobás). Some characteristic results:

- the degree distribution is binomial and most of the nodes have degree (very) close to the average degree;
- for $p \geq \frac{1}{n}$ cycles appear in the graph, and soon also the giant component;
- for $p \geq \frac{\log_2 n}{n}$ almost all graphs are connected;

Real-life networks are usually not random in the Erdős - Rényi sense.

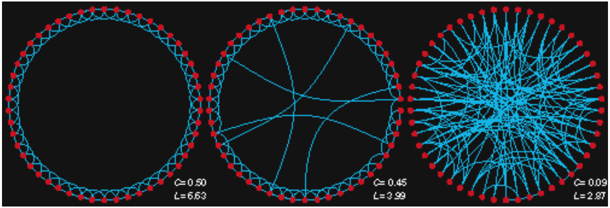
The analysis of their distributions gave a new views about their structure.

Small worlds / six degrees of separation

In 1967 a psychologist Stanley Milgram made his experiment with letters. The letter should reach a target person. The persons involved in experiment were asked to send the letter with these instructions to his or her acquaintance that is supposed to be closer (in the acquaintances network) to the target person. The letter was sent from Boston to Omaha. The average length of the successful paths was 6 – *six degrees of separation*. The average path length on the internet is 19 clicks.
The networks in which the average shortest path length is small are called *small worlds*. netlogo
Similary, Mark Granovetter in 1973 noticed that in social networks groups are formed (*strong ties*) linked among them by *weak ties*.

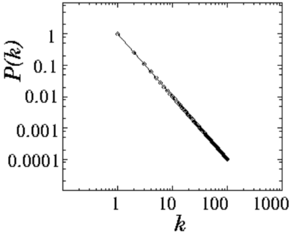
Rewiring

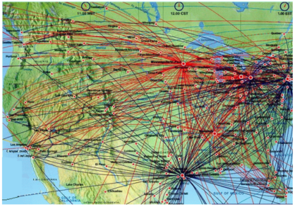
Duncan Watts and Steven Strogatz developed in late 90-ties a procedure for construction of (random) small worlds by rewiring – an edge is randomly selected and one of its endnodes is attached to same other node. After each rewiring step the average length of geodesics is usually descreasing because the rewiring creates shortcuts.



Densmore: Power-Law Networks

Scale-free networks





Barabasi: The Architecture of Complexity

Albert-László Barabási with his group from University of Notre Dame analyzed in 1998 several networks and noticed:

- in a network there exist some nodes with large degree (very unprobable in ER graphs). These nodes link the network into a single component;
- the degree distribution follows the power law – the probability p_d that a node has a degree d equals to $p_d = cd^{-\gamma}$ – in log-log scale its diagram is represented by a line.

It turned out that most of real life networks have such characteristics. Because for these networks their degree distribution has no natural scale they were named scale free networks.

... Scale-free networks

For a discussion about the notion of scale-free network see Li et al. Further research showed that this kind of networks appear in many fields: persons – e-mail, phone calls, sexual contacts (drug users, AIDS), collaboration; movie actors – playing in the same movie; proteins – interactions; words – semantic relations; . . .
The first explanation (Barabási) of free-scale nature of many real-life networks was:

- these networks are growing;
- in this process new nodes are added and linked with new edges to already existing nodes. The random selection of node to which a new node is attached is not uniform but follows the preferential attachment rule – the selection probability is proportional to the degree of a node.

. . . Scale-free networks

Based on this model it can be shown that:

- the degree distribution is the power law;
- the average length of geodesics is $O(\log n)$;
- these networks are resilient against random node or edge removals (random attacks), but quickly become disconnected when large degree nodes (Achilles' heel) are removed (targeted attacks).

In real-life networks nodes often also form groups – clustering.

Several improvements and alternative models were proposed that also produce scale-free networks with some additional properties characteristic for real-life networks: copying (Kleinberg 1999), combining random and preferential attachment (Pennock et al. 2002), R-mat (Chakrabarti et al. 2004), forest fire (Leskovec et al. 2005), aging, fitness, nonlinear preferences, . . .

. . . Scale-free networks – exponent

Problems: large variability at the end, line only on an interval, nonuniform data density, . . .

Also the distribution function is power law

$$\int Cx^{-\tau} = C\frac{x^{1-\tau}}{1-\tau}$$

Newman's estimate

$$\tau = 1 + n(\sum_{i=1}^{n} \ln \frac{x_i}{x_{min}})^{-1}$$

M. E. J. Newman: Power laws, Pareto distributions and Zipf's law and Power-law distributions in empirical data. Packages in R: igraph, plfit / Santa Fe.

power, Pareto

See also

Matthew Effect: Wikipedia, When Do Matthew Effects Occur?
Epidemies: Barthélemy, Barrat, Pastor-Sattoras, Vespignani, Complex Networks Collaboratory.
Searching: Adamic et al.
General: Center for Complex Network Research, Newman, Borner, Sanyal, Vespignani.

Random networks

Batagelj V., Brandes U.: *Efficient Generation of Large Random Networks*. Physical Review E 71, 036113, 2005
web page

overview

Dense networks

The important parts of networks are smaller, but relatively dense. For such networks the standard "nodes and links" visualization is not readable.

Much better visualization can be produced using the matrix representation for an appropriate ordering (determined for example by clustering or blockmodeling).

Another approach is to display only the skeleton of the network obtained by removing less important links. The standard skeleton is a minimal spanning tree; often also Pathfinder skeletons are used.

Pathfinder / Dissimilarities

Joly and Le Calvé theorem:
For any even dissimilarity measure d there is a unique number $p \geq 0$, called its metric index, such that: d^r is metric for all $r \leq p$, and d^r is not metric for all $r > p$.

In the opposite direction we can say: Let d be a dissimilarity and for x, y and z we have $d(x,z) + d(z,y) \geq d(x,y)$ and $d(x,y) > \max(d(x,z), d(z,y))$ then there exists a unique number $p \geq 0$ such that for all $r > p$

$$d^r(x,z) + d^r(z,y) < d^r(x,y)$$

or equivalently

$$d(x,z) \boxed{r} d(z,y) < d(x,y)$$

where $a \boxed{r} b = \sqrt[r]{a^r + b^r}$.

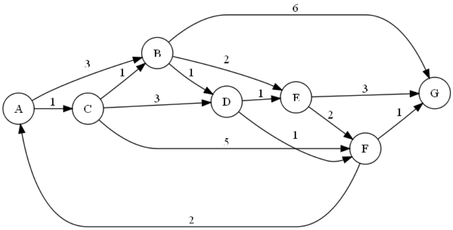
Pathfinder

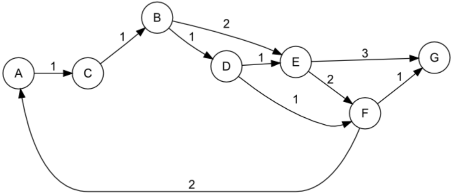
The Pathfinder algorithm was proposed in eighties (Schvaneveldt 1981, Schvaneveldt etal. 1989; Schvaneveldt, 1990) for simplification of weighted networks – it removes from the network all links that do not satisfy the triangle inequality – if for a link a shorter path exists connecting its endnodes then the link is removed. The basic idea of the Pathfinder algorithm is simple. It produces a network $PFnet(\mathbf{W}, r, q) = (\mathcal{V}, \mathcal{L}_{PF})$

compute $\mathbf{W}^{(q)}$;
$\mathcal{L}_{PF} := \emptyset$;
for $e(u, v) \in \mathcal{L}$ **do begin**
  if $\mathbf{W}^{(q)}[u, v] = \mathbf{W}[u, v]$ **then** $\mathcal{L}_{PF} := \mathcal{L}_{PF} \cup \{e\}$
end;

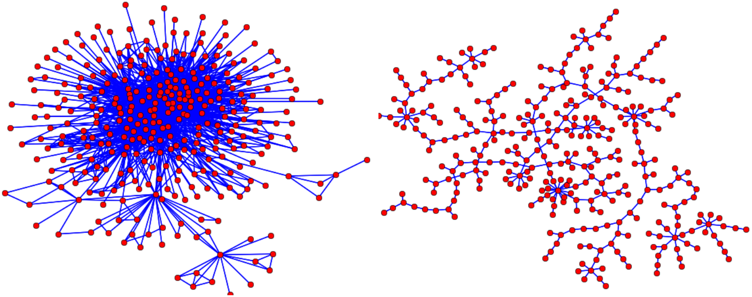
where $\mathbf{W}$ is a network dissimilarity matrix and $\mathbf{W}^{(q)}$ the matrix of values of all walks of length at most q computed over the semiring $(\mathbb{R}_0^+, \oplus, \boxed{r}, \infty, 0)$ where $a \oplus b = \min(a, b)$.

Pathfinder





Pathfinder



PathFindeR

We developed a fast Pathfinder algorithm for large sparse networks. For smaller (up to 1000) networks we wrote a program in R (based on the Fletcher's algorithm) to do the job.

```
# PathFinder
# http://pajek.imfm.si/lib/exe/fetch.php?media=slides:pfxxx.pdf
# by Vladimir Batagelj, December 24-28, 2011
#
# PathFinder(D,r,q) - determines the skeleton of network represented by
# matrix D . The weights in D should be dissimilarities; the value 0
# denotes nonlinked nodes.
# r - is the parameter in Minkowski operation
# q - is the limit on the length of considered paths; if q >= n-1
#     all paths are considered.
#
# PathFinderSim(S,r,q,s) - is a version of PathFinder for the case
# when the weights are similarities.
# s - determines how the similarity is transformed into dissimilarity
#   s = 1 -  D = 1+max S - S
#   s = 2 -  D = 1/S
# In the resulting skeleton the weights are the original similarities.

Multiply <- function(A,B,r){
  n <- nrow(A); C <- matrix(Inf,nrow=n,ncol=n)
  if(is.infinite(r)){
    for(i in 1:n) for(j in 1:n) C[i,j] <- min(pmax(A[i,],B[,j]))
  } else if (r==1){
    for(i in 1:n) for(j in 1:n) C[i,j] <- min(A[i,]+B[,j])
  } else {
    for(i in 1:n) for(j in 1:n) C[i,j] <- min((A[i,]^r+B[,j]^r)^(1/r))
  }
  C
}
```

PathFindeR

```
Closure <- function(W,r){
  n <- nrow(W); W[W==0] <- Inf; diag(W) <- 0
  if(is.infinite(r)){for(k in 1:n) for(i in 1:n) W[i,] <- pmin(W[i,],pmax(W[i,k],W[k,]))
  } else if (r==1){for(k in 1:n) for(i in 1:n) W[i,] <- pmin(W[i,],(W[i,k]+W[k,]))
  } else {for(k in 1:n) for(i in 1:n) W[i,] <- pmin(W[i,],(W[i,k]^r+W[k,]^r)^(1/r)) }
  W
}

Power <- function(W,r,q){
  n <- nrow(W); W[W==0] <- Inf; diag(W) <- 0
  T <- matrix(Inf,nrow=n,ncol=n); diag(T) <- 0
  if (q > 0) {
    i <- q; S <- W
    repeat{
      if ((i %% 2) == 1) { T <- Multiply(T,S,r) }
      i <- i %/% 2; if (i == 0)  break
      S <- Multiply(S,S,r)
    }
  }
  rownames(T) <- colnames(T) <- rownames(W)
  T
}
```

PathFindeR

```
PathFinder <- function(D,r=Inf,q=Inf,eps=0.0000001){
  if(r<1) stop("Error: r < 1")
  if(q>=nrow(D)-1) {D[(D>0)&(abs(D-Closure(D,r))>eps)] <- 0
  } else {D[(D>0)&(abs(D-Power(D,r,q))>eps)] <- 0}
  D
}

PathFinderSim <- function(S,r=Inf,q=Inf,s=1,eps=0.0000001){
  if(r<1) stop("Error: r < 1")
  n <- nrow(S); D <- S
  if(s==1) {D[S>0] <- 1+max(S)-S[S>0]} else {D[S>0] <- 1/S[S>0]};
  if(q>=n-1) {S[(S>0)&(abs(D-Closure(D,r))>eps)] <- 0
  } else {S[(S>0)&(abs(D-Power(D,r,q))>eps)] <- 0}
  S
}

# setwd("C:/Users/Batagelj/work/R/pf")
# PF <- PathFinder(n1,1,Inf)
# savenetwork(PF,'PFtest.net')

# cat(date(),"\n"); PF2 <- PathFinderSim(n2,1,Inf,2); cat(date(),"\n");
# savenetwork(PF2,'PF2500.net'); cat(date(),"\n")
```

PathfindeR / code

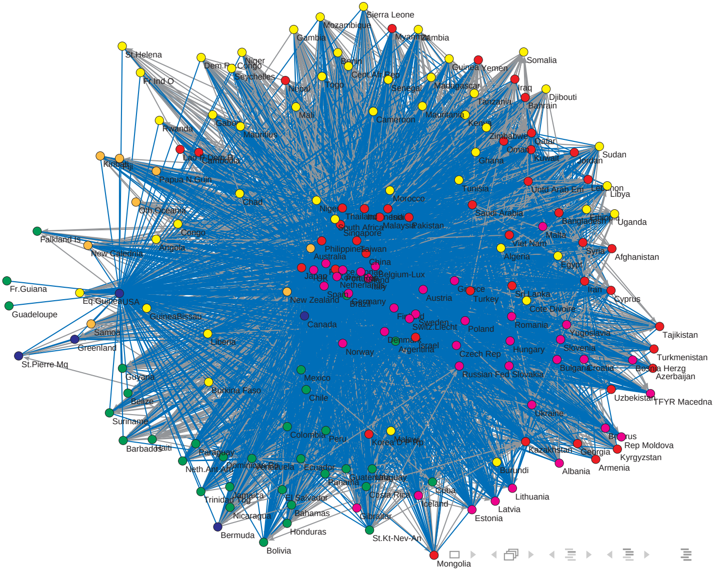
Cosine dissimilarity $d_2 = 1 - \frac{\mathbf{x}.\mathbf{y}}{|\mathbf{x}|.|\mathbf{y}|}$ to be included.

A display of World Trade 1999 network

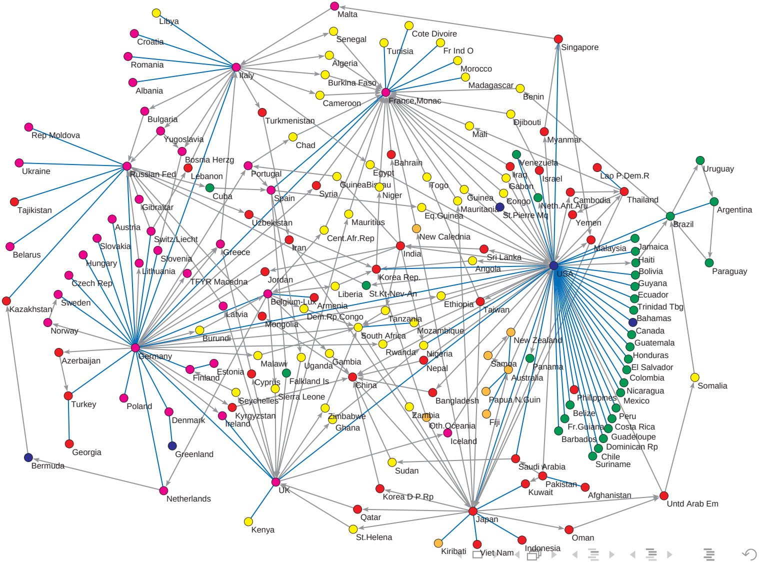
What else?

V. Batagelj

- Temporal networks
- Scale-free networks
- Random networks
- Pathfinder
- Some hints
- What else?



Pathfinder skeleton of World Trade 1999 network



Drawing

- order of links (values)
- INI file
- Unicode, icons
- URL links
- new link
- JSON and D3.js

Memory and dispose

Using the command

`Info/Memory`

we obtain the information about the available computer memory

```
--------------------------------------------------------------------------------
Global Memory Status, 32bit OS, call=1
--------------------------------------------------------------------------------
kilobytes of physical memory:                3.134.820 kB
percent of free memory:                             30 %  (+4% since Pajek start)
kilobytes of free physical memory:             928.696 kB

kilobytes of paging file space:              5.716.496 kB
kilobytes of free paging file space:         2.802.704 kB

kilobytes of virtual address space:          2.097.024 kB
kilobytes of free virtual address space:     1.939.220 kB
```

Using the command

`File/Network/Dispose`

we can remove the current network from register and free the corresponding computer memory.
Similar commands exist also for other data objects.

Names of nodes

Considerable amount of memory is used by node labels. Using the command

`File/Network/Dispose`

the option `Read - Save vertices labels?` can be switched off thus preventing the storage of labels. We can add the labels later to the resulting subnetworks using

`Network/Create New Network/Transform/Add/Vertex Labels/`

Even greater memory economy is done in a special version of `Pajek`– `Pajek`XXL.

Subnetworks

As already mentioned to obtain a subnetwork we have to keep cutting also the corresponding properties. This can be quite annoying. Recently a new command for matching labels of nodes in two networks was implemented in `Pajek`

`Networks/Match Vertex Labels`

This operation is very general with several usages:

- combinig two networks where some (but not all) node labels are the same
- making partitions and vectors compatible with extracted subnetworks (without keeping track of all extractions done so far).

Relabel

In the analysis process in `Pajek` we construct important intermediate data objects. To find them easier in the registers and make the process more readable we can relabel the selected data objects using commands such as

`File/Network/Change Label`

Similar commands exist for all data objects.

What else?

Several topics on network analysis were not covered in these lectures:

- brokerage (see ESNA, ch. 7)
- difussion, epidemics (see ESNA, ch. 8)
- probabilistic models and analysis (p* or exponential random graph models ERGMs, SIENA)
- homophily
- structural holes (Burt)
- spatial networks
- how to?
- ...