Introduction to Network Analysis using **`Pajek`**

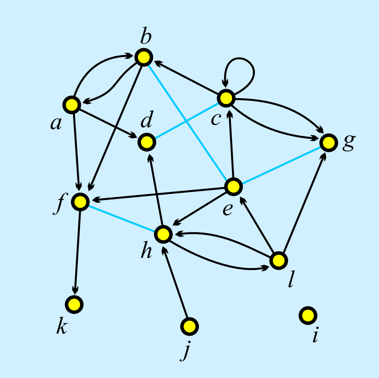
3. Structure of networks: subnetworks

Vladimir Batagelj

IMFM Ljubljana and IAM UP Koper

PhD and MS program in Statistics
University of Ljubljana, 2022

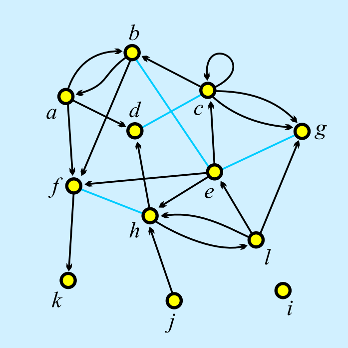
Outline



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

Current version of slides (February 17, 2022 at 02 : 29): slides PDF

Degrees



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

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

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

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 details; special algorithms are needed for their analysis (**Pajek**).

Large networks

Large network – several thousands or millions of nodes. Can be stored in 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

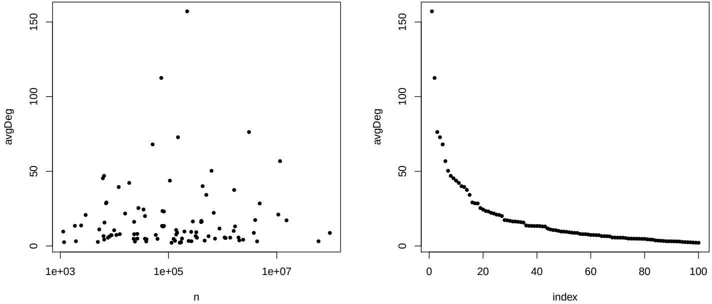
Networks with ground-truth communities

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

Pajek datasets.

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 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 the Dunbar's number is between 100 and 150.

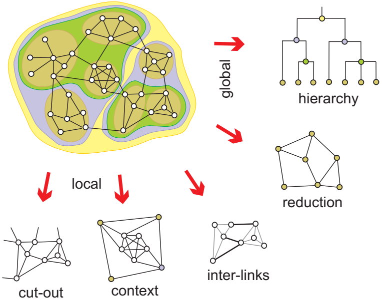
Complexity of algorithms

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

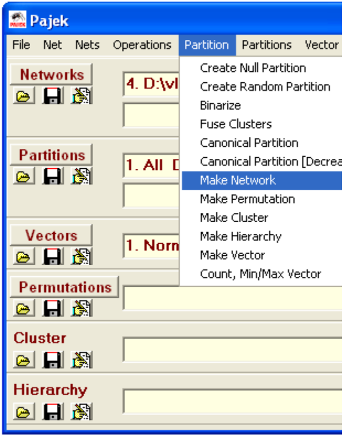


In analysis of a *large* network (several thousands or millions of nodes, the network can be stored in computer memory) we can't display it in its totality; also there are only few algorithms available.

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

Pajek's data types

In **Pajek** analysis and visualization are performed using 6 data types:



- *network* (graph),
- *partition* (nominal or ordinal properties of nodes),
- *vector* (numerical properties of nodes),
- *cluster* (subset of nodes),
- *permutation* (reordering of nodes, ordinal properties), and
- *hierarchy* (general tree structure on nodes).

Pajek supports also *multi-relational*, *temporal* and *two-mode* networks.

Pajek's data types

The power of **Pajek** is based on several transformations that support different transitions among these data structures. Also the menu structure of the main **Pajek**'s window is based on them. **Pajek**'s main window uses a 'calculator' paradigm with list-accumulator for each data type. The operations are performed on the currently active (selected) data and are also returning the results through accumulators.

The procedures are available through the main window menus. Frequently used sequences of operations can be defined as *macro*s. This allows also the adaptations of **Pajek** to groups of users from different areas (social networks, chemistry, genealogy, computer science, mathematics...) for specific tasks. **Pajek** supports also *repetitive operations* on series of networks.

Statistics

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 (betweeneness, 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 association as collected from subjects (students). The weight on the arcs is the count of word associations.

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

It has 23219 nodes and 325624 arcs (564 loops); number of links with value=1 is 227481.

... Statistics

To identify the nodes with the largest degree:

```
Net/Partitions/Degree/All
Partition/Make vector
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

In igraph the function `degree()` has `mode`s `in`, `out` and `all`.

```
> G <- read.graph("links.net",format="pajek")
> deg <- degree(G,mode="all")
> plot(G,vertex.size=deg*3)
```

Degrees in igraph

In the file igraph+.R some additional functions are collected that make network analysis easier. For example, the function top

```
top <- function(v,k){
  ord <- rev(order(v)); sel <- ord[1:k]
  S <- data.frame(name=names(v[sel]),
    value=as.vector(v[sel]))
  return(S)
}
```

returns top k values in the node attribute v.

```
> wdir <- "C:/Users/batagelj/Documents/papers/2017/Moscow
> setwd(wdir)
> library(igraph)
# delete *network and empty line before *vertices
> T <- read.graph("./nets/eatRS.net",format="pajek")
> vcount(T)
[1] 23219
> ecount(T)
[1] 325624
> source("igraph+.R")
> SR <- graph.reverse(T)
> SR$indeg <- degree(SR,mode="in")
```

...Degrees in igraph

```
> top(SR$indeg,10)
      name  value
1       ME   1074
2      MAN   1046
3     GOOD    861
4      SEX    828
5       NO    780
6    MONEY    743
7      YES    718
8     WORK    672
9  NOTHING    672
10    FOOD    665
> SR$windeg <-  strength(SR,mode="in")
> max(SR$windeg)
[1] 4387
> top(SR$windeg,20)
> SR$awindeg <- SR$windeg/SR$indeg
> SR$awindeg[is.nan(SR$awindeg)] <- 0
> top(SR$awindeg,20)
```

Statistics / **Pajek** and R

Pajek (0.89 and higher) 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

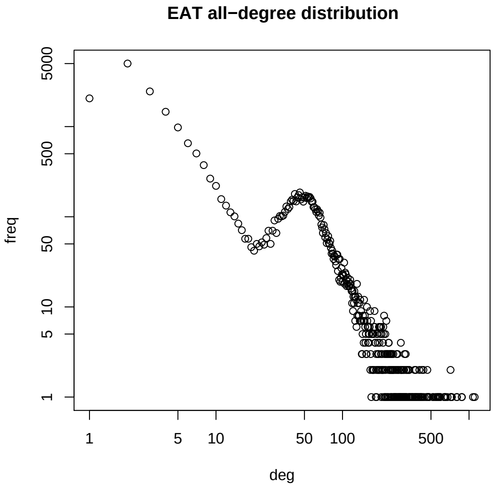
```
Network/Info/General
Network/Create Vector/Centrality/Degree/All
Tools/R/Send to R/Current Vector
```

In R we determine their distribution and plot it

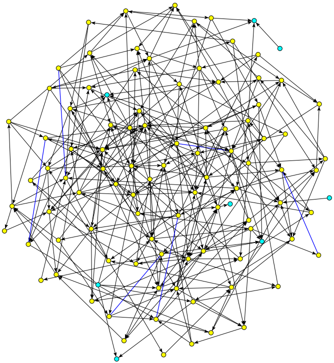
```
summary(v2)
t <- table(v2)
x<-as.numeric(names(t))
plot(x,t,log='xy',main='degree distribution',
  xlab='deg',ylab='freq')
```

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).
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 probabilty p. In **Pajek**

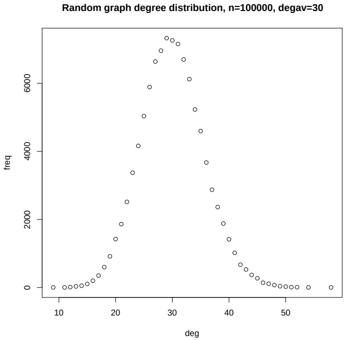
```
Network/Create
Random Network/
Bernoulli/Poisson/Undirected
General [100] [2.5]
```

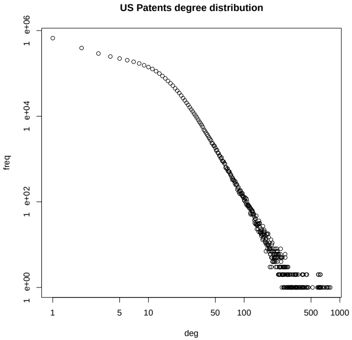
instead of probability p a more intuitive average degree is used

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

It holds $p = \frac{m}{m_{max}}$ and, for simple graphs, also $\overline{\deg} = \frac{2m}{n}$.
Random graph in the picture has 100 nodes and average degree

Degree distribution





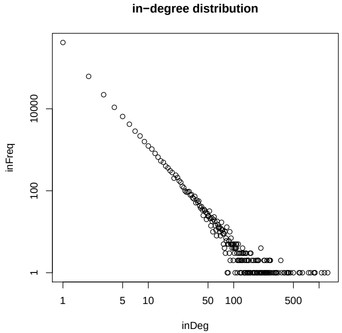
Real-life networks are usually not random in the Erdős/Renyi sense. The analysis of their distributions gave a new view about their structure – Watts (Small worlds), Barabási (nd/networks, Linked).

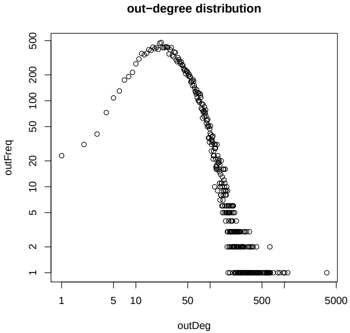
in/out-degree distributions

We read in **Pajek** the 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:
 v3   :  From partition 1 (548600)
 v4   :  From partition 2 (548600)
-------------------------------------------------------------
> inTab <- table(v3)
> 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(v4)
> outdeg <- as.integer(names(ouTab))
> outDeg <- outdeg[outdeg>0]
> outFreq <- as.vector(ouTab[outdeg>0])
> plot(outDeg,outFreq,log='xy',main="out-degree distribution")
```

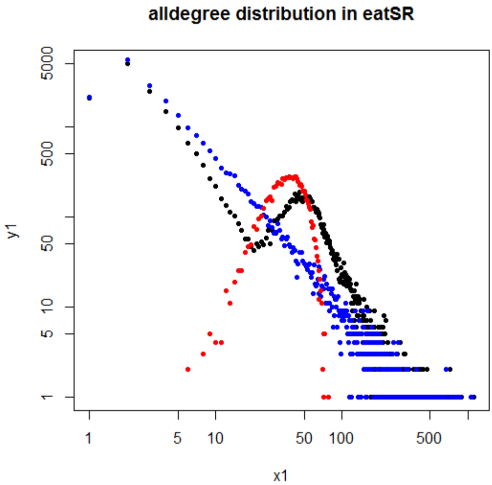
in/out-degree distributions





The in-degree distribution is ”scale-free”-like. The parameters 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



Papers by years / centrality network

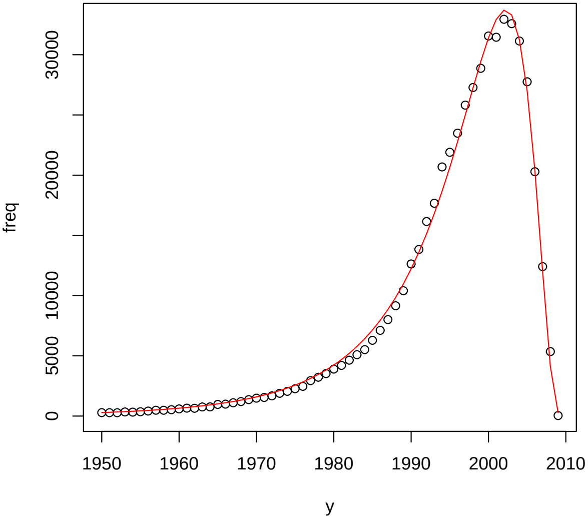
From the file `Year.clu` containing the year of publication of a paper we can get the distribution of *papers by years*. For the centrality network we get:

```
> setwd("C:/Users/Batagelj/work/Python/WoS/Central")
> years <- read.table(file="Year.clu",header=FALSE,skip=2)$V1
> t <- table(years)
> year <- as.integer(names(t))
> freq <- as.vector(t[1950<=year & year<=2009])
> y <- 1950:2009
> plot(y,freq)
> model <- nls(freq~c*dlnorm(2010-y,a,b),
+ start=list(c=350000,a=2,b=0.7))
> model
Nonlinear regression model
  model:  freq ~ c * dlnorm(2010 - y, a, b)
   data:  parent.frame()
        c         a         b
5.427e+05 2.491e+00 6.624e-01
 residual sum-of-squares: 20474181

Number of iterations to convergence: 7
Achieved convergence tolerance: 3.978e-06
> lines(y,predict(model,list(x=2010-y)),col='red')
```

It can be well approximated by the *lognormal distribution*, but also by the *generalized reciprocal power exponential curve* $c * (x + d)^{\frac{a}{b+x}}$.

Papers by years / centrality network



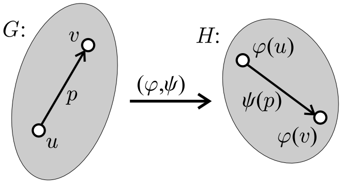
Homomorphisms of graphs

Functions (φ, ψ), $\varphi\colon \mathcal{V} \to \mathcal{V}'$ and $\psi\colon \mathcal{L} \to \mathcal{L}'$ determine a *weak homomorphism* of graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ in graph $\mathcal{H} = (\mathcal{V}', \mathcal{L}')$ iff:

$$\forall u, v \in \mathcal{V}\, \forall p \in \mathcal{L} : (p(u : v) \Rightarrow \psi(p)(\varphi(u) : \varphi(v)))$$

and they determine a *(strong) homomorphism* of graph $\mathcal{G}$ in graph $\mathcal{H}$ iff:

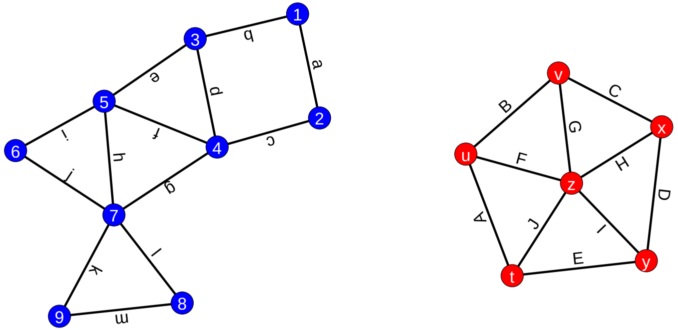
$$\forall u, v \in \mathcal{V}\, \forall p \in \mathcal{L} : (p(u, v) \Rightarrow \psi(p)(\varphi(u), \varphi(v)))$$



If φ and ψ are bijections and the condition hold in both direction we get an *isomorphism* of graphs $\mathcal{G}$ and $\mathcal{H}$. We denote the weak isomorphism by $\mathcal{G} \sim \mathcal{H}$; and the (strong) isomorphism by $\mathcal{G} \approx \mathcal{H}$. It holds $\approx \subset \sim$.

An *invariant* of graph is called each graph characteristic that has the same value for all isomorphic graphs.

Homomorphism

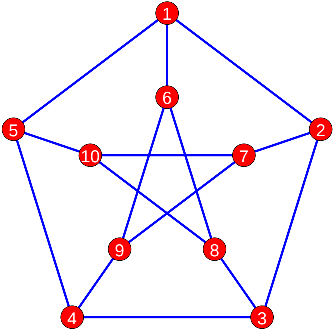


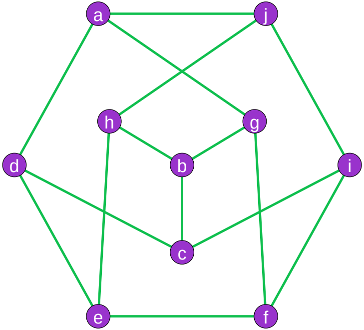
φ	1	2	3	4	5	6	7	8	9
	t	y	z	x	v	u	z	y	t

ψ	a	b	c	d	e	f	g	h	i	j	k	l	m
	E	J	D	H	G	C	H	G	B	F	J	I	E

homoEna.net

Isomorphic graphs





φ	1	2	3	4	5	6	7	8	9	10
	b	h	j	a	g	c	e	i	d	f

izoPet.net

Clusters, clusterings, partitions, hierarchies

A nonempty subset $C \subseteq \mathcal{V}$ is called a *cluster* (group). A nonempty set of clusters $\mathbf{C} = \{C_i\}$ forms a *clustering*.
Clustering $\mathbf{C} = \{C_i\}$ is a *partition* iff

$$\cup\mathbf{C} = \bigcup_i C_i = \mathcal{V} \quad \text{and} \quad i \neq j \Rightarrow C_i \cap C_j = \emptyset$$

Clustering $\mathbf{C} = \{C_i\}$ is a *hierarchy* iff

$$C_i \cap C_j \in \{\emptyset, C_i, C_j\}$$

Hierarchy $\mathbf{C} = \{C_i\}$ is *complete*, iff $\cup\mathbf{C} = \mathcal{V}$; and is *basic* if for all $v \in \cup\mathbf{C}$ also $\{v\} \in \mathbf{C}$.

Examples

Node set:

$$\mathcal{V} = \{a, b, c, d, e, f, g\}$$

Partition:

$$\mathbf{C} = \{\{a, b, e\}, \{c, g\}, \{d, f\}\}$$

Cluster, class:

$$C_2 = \{c, g\}$$

Hierarchy:

$$\begin{aligned}\mathbf{H} = \{&\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\},\\ &\{a, e\}, \{c, g\}, \{d, f\}, \{a, b, e\},\\ &\{c, d, f, g\}, \{a, b, c, d, e, f, g\}\}\end{aligned}$$

Draw / Partition

Size of networks

Pajek

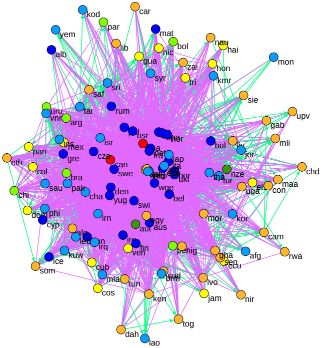
Statistics

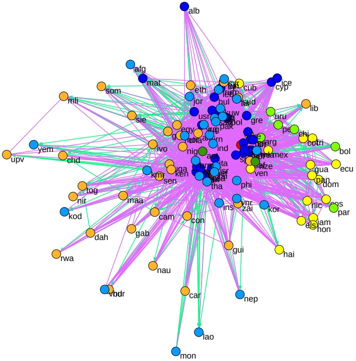
Morphisms

Partitions

Subgraphs

Cuts





```
Draw/Network + First Partition
Layout/Energy/Kamada-Kawai/Free
Layout/Energy/Fruchterman Reingold/2D
```

Contraction of cluster

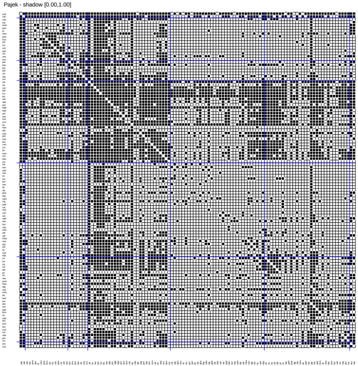
Contraction of cluster C is called a graph $\mathcal{G}/C$, in which all nodes of the cluster C are replaced by a single node, say c. More precisely:

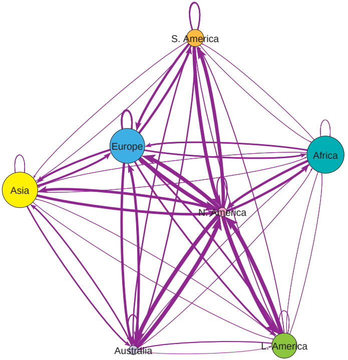
$\mathcal{G}/C = (\mathcal{V}', \mathcal{L}')$, where $\mathcal{V}' = (\mathcal{V} \setminus C) \cup \{c\}$ and $\mathcal{L}'$ consists of links from $\mathcal{L}$ that have both endnodes in $\mathcal{V} \setminus C$. Beside these it contains also a 'star' with the center c and: arc (v, c), if $\exists p \in \mathcal{L}, u \in C : p(v, u)$; or arc (c, v), if $\exists p \in \mathcal{L}, u \in C : p(u, v)$. There is a loop (c, c) in c if $\exists p \in \mathcal{L}, u, v \in C : p(u, v)$.

In a network over graph $\mathcal{G}$ we have also to specify how are determined the values/weights in the shrunk part of the network. Usually as the sum or maksimum/minimum of the original values.

`Operations/Network + Partition/Shrink Network`

Contracted clusters – international trade





Snyder and Kick's international trade. Matrix display of dense networks.

$$w(C_i, C_j) = \frac{n(C_i, C_j)}{n(C_i) \cdot n(C_j)}$$

Macros.

Computing the weights *w*

```
File/Pajek Project File/Read [SKtrade.paj]
Network/Create New Network/Transform/Remove/Loops [No]
Network/Create New Network/Transform/Edges -> Arcs [No]
Operations/Network+Partition/Shrink Network [1 0]

          1   2   3   4   5   6   7  Label
---------------------------------------------
   1.     2  30  13  56  42  45   4  #usa
   2.    30  74  25 196  20  37  12  #cub
   3.    12  28  33 124  16  36   5  #per
   4.    55 217 130 694 427 483  41  #uki
   5.    42   8  14 406 122 117  11  #mli
   6.    43  37  43 444 142 307  30  #irn
   7.     4   4   5  39   9  30   2  #aut

Partition/Make Permutation
[select partition (Sub)continents]
Operations/Partition+Permutation/
  Functional Composition Partition*Permutation
Partition/Count

count     2  15   7  29  33  30   2
```

. . . Computing the weights *w*

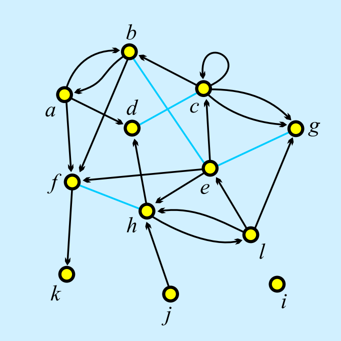
```
Partition/Copy to Vector
Vector/Create Constant Vector [7 1.0]
[select as second vector Copy of partition ...]
Vectors/Divide (First/Second)
Network/Create Vector/Get Loops
Vectors/Add (First+Second)
Operations/Network+Vector/Transform/Put Loops/as Arcs
[select vector Divide V? by ...]
Operations/Network+Vector/Vector#Network/input
Operations/Network+Vector/Vector#Network/output
```

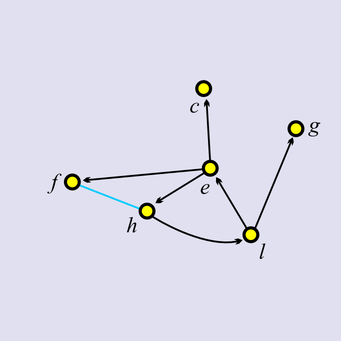
		1	2	3	4	5	6	7
#usa	1.	0.50	1.00	0.93	0.97	0.64	0.75	1.00
#cub	2.	1.00	0.33	0.24	0.45	0.04	0.08	0.40
#per	3.	0.86	0.27	0.67	0.61	0.07	0.17	0.36
#uki	4.	0.95	0.50	0.64	0.83	0.45	0.56	0.71
#mli	5.	0.64	0.02	0.06	0.42	0.11	0.12	0.17
#irn	6.	0.72	0.08	0.20	0.51	0.14	0.34	0.50
#aut	7.	1.00	0.13	0.36	0.67	0.14	0.50	0.50

Note: Set diagonal values to 1 ?

Macro `weights`.

Subgraph





A *subgraph* $\mathcal{H} = (\mathcal{V}', \mathcal{L}')$ of a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ is a graph which set of links is a subset of set of links of $\mathcal{G}$, $\mathcal{L}' \subseteq \mathcal{L}$, its node set is a subset of set of nodes of $\mathcal{G}$, $\mathcal{V}' \subseteq \mathcal{V}$, 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 $\mathcal{L}'$).

Subgraph in igraph

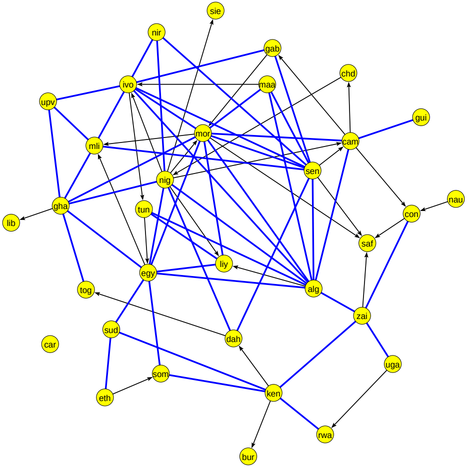
```
induced_subgraph(graph,vids,
    impl=c("auto","copy_and_delete","create_from_scratch"))

subgraph.edges(graph,eids,delete.vertices=TRUE)

delete_edges(graph, edges)

> Class <- read.graph("class.net",format="pajek")
> vertex_attr_names(Class)
[1] "id"    "name" "x"    "y"    "z"
> vertex_attr(Class)$shape <- NULL
> sex <- as.integer(substr(vertex_attr(Class)$id,1,1)=="m")
> F <- V(Class)[sex==0]
> Fclass <- induced_subgraph(Class,F)
> plot(Fclass)
> N <- E(Class)[F %--% F]
> N
+ 30/56 edges from 3a5cb23 (vertex names):
 [1] w07->w42 w09->w24 w09->w10 w10->w28 w24->w10 w28->w42 w42->
 [9] w12->w63 w09->w12 w07->w10 w07->w22 w07->w28 w10->w22 w22->
[17] w22->w28 w24->w42 w09->w63 w63->w12 w12->w09 w10->w07 w22->
[25] w22->w10 w24->w22 w42->w22 w28->w22 w42->w24 w63->w09
```

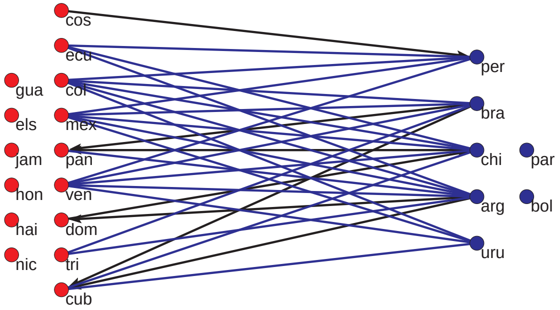
Cut-out – induced subgraph: Snyder and Kick Africa



Operations/Network + Partition/Extract

Subnetwork [6]

Cut-out: Snyder and Kick
Latin America : South America




```
Operations/Network + Partition/Extract Subnetwork  [3,4]
Operations/Network + Partition/Transform/Remove lines/
  Inside clusters   [3,4]
```

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

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

Cut-outs in igraph

```
extract_clusters <- function(N,atn,clus){
  C <- vertex_attr(N,atn); S <- V(N)[C %in% clus]
  return(induced_subgraph(N,S))
}
interlinks <- function(N,atn,c1,c2,col1="red",col2="blue"){
  S <- extract_clusters(N,atn,c(c1,c2))
  C <- vertex_attr(S,atn)
  C1 <- V(S)[C==c1]; C2 <- V(S)[C==c2]
  V(S)$color <- ifelse(C==c1,col1,col2)
  P <- E(S)[(C1 %--% C1)|(C2 %--% C2)]
  return(delete_edges(S,P))
}

> library(igraph); source("igraph+.R")
> SaK <- read.graph("./nets/SaKtrade.net",format="pajek")
> V(SaK)$sc <-  read_Pajek_clu("./nets/SaKtrade.clu",skip=7)
> Af <- extract_clusters(SaK,"sc",c(6))
> plot(Af)
> B <- interlinks(SaK,"sc",3,4,col1="yellow",col2="cyan")
> plot(B)
```

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 $\mathcal{N} = (\mathcal{V}, \mathcal{L}, p)$, $p : \mathcal{V} \to \mathbb{R}$, at selected level t is a subnetwork $\mathcal{N}(t) = (\mathcal{V}', \mathcal{L}(\mathcal{V}'), p)$, determined by the set

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

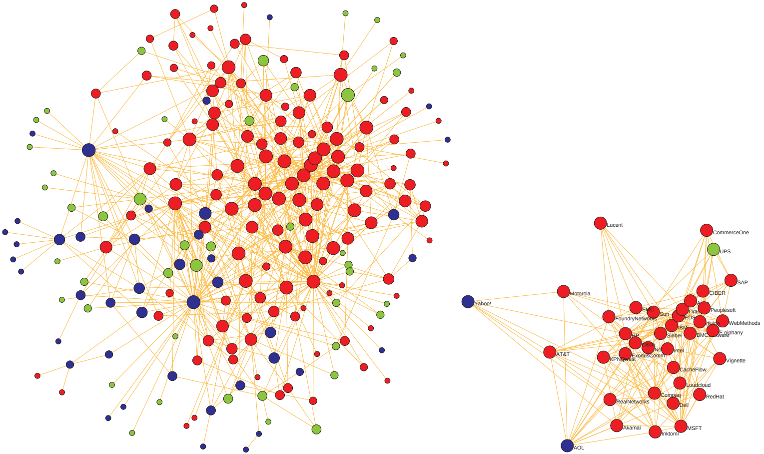
and $\mathcal{L}(\mathcal{V}')$ is the set of links from $\mathcal{L}$ that have both endnodes in $\mathcal{V}'$.

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

Node-cut: Krebs Internet Industries, core=6



Each node represents a company that competes in the Internet industry, 1998 do 2001. $n = 219$, $m = 631$. red – content, blue – infrastructure, green – commerce. Two companies are linked with an edge if they have announced a joint venture, strategic alliance or other partnership.

Subnetworks

V. Batagelj

Size of networks

Pajek

Statistics

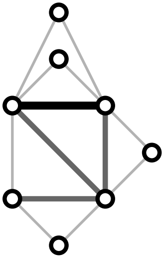
Morphisms

Partitions

Subgraphs

Cuts

Triangular network

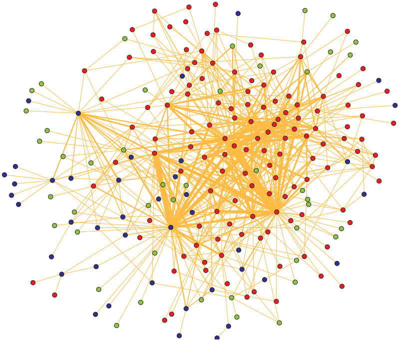


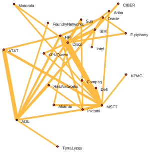
Let $\mathcal{G}$ be a simple undirected graph. A *triangular network* $\mathcal{N}_T(\mathcal{G}) = (\mathcal{V}, \mathcal{E}_T, w)$ determined by $\mathcal{G}$ is a subgraph $\mathcal{G}_T = (\mathcal{V}, \mathcal{E}_T)$ of $\mathcal{G}$ which set of edges $\mathcal{E}_T$ consists of all triangular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

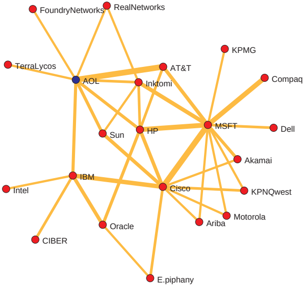
Triangular networks can be used to efficiently identify dense clique-like parts of a graph. If an edge e belongs to a k-clique in $\mathcal{G}$ then $w(e) \geq k - 2$.

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

Link-cut: Krebs Internet Industries, $w_3 \geq 5$







Overlap weight – definition

The (topological) *overlap weight* of an edge $e = (u : v) \in \mathcal{E}$ in an undirected simple graph $\mathbf{G} = (\mathcal{V}, \mathcal{E})$ is defined as

$$o(e) = \frac{t(e)}{(\deg(u) - 1) + (\deg(v) - 1) - t(e)}$$

$t(e) = w_3(e)$ is the *number of triangles* (cycles of length 3) to which the edge e belongs. In the case $\deg(u) = \deg(v) = 1$ we set $o(e) = 0$.

The overlap weight is essentially a Jaccard similarity index

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

for $X = N(u) \setminus \{v\}$ and $Y = N(v) \setminus \{u\}$ where $N(z)$ is the set of neighbors of a node z.
Denoting $\mu = \max_{e \in \mathcal{E}} t(e)$ and $M(e) = \max(\deg(u), \deg(v)) - 1$ we define a *corrected overlap weight* as

$$o'(e) = \frac{t(e)}{\mu + M(e) - t(e)}$$

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  [+10] [#10]
Vector/Make Partition/by Intervals/Selected Thresholds  [t]
Operations/Network + Partition/Extract Subnetwork  [2]
```

Link-cut: weighted network

```
Network/Info/Line values   [#10]
Network/Create New Network/Transform/Remove/Lines with Value/
  lower than   [t]
Network/Create Partition/Degree/All
Operations/Network + Partition/Extract Subnetwork  [1-*]
```

Cuts in igraph

```
vertex_cut <- function(N,atn,t){
  v <- vertex_attr(N,atn); vCut <- V(N)[v>=t]
  return(induced_subgraph(N,vCut))
}
edge_cut <- function(N,atn,t){
  w <- edge_attr(N,atn); eCut <- E(N)[w>=t]
  return(subgraph.edges(N,eCut))
}

> R <- read.graph("./nets/class.net",format="pajek")
> vertex_attr(R)$shape <- NULL
> V(R)$deg <- degree(R)
> Cut <- vertex_cut(R,"deg",8)
> plot(Cut,vertex.size=V(Cut)$deg*3)
> E(R)$rnd <- sample(1:10,ecount(R),replace=TRUE)
> Ec <- edge_cut(R,"rnd",9)
> plot(Ec,edge.width=E(Ec)$rnd)
```

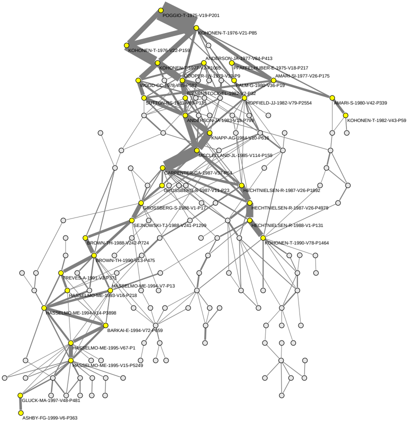
Simple analysis using cuts

We look at the components of $\mathcal{N}(t)$. 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 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.