Networks data analysis: An Introduction

Andrea Deghi

University of Siena

10th International Winter School on Inequality and Social Welfare Theory

Canazei – January 13, 2015

UNIVERSITÀ DI SIENA 1240

Outline

Main References:

- De Nooy, W., Mrvar, A. and Batagelj, V. (2005). "Exploratory Social Network Analysis with Pajek". Cambridge University Press.
- Kolaczyk, E.D. and Csàrdi, G. (2014). "Statistical Analysis of Network Data with R". Springer, New York.

UNIVERSITÀ DI SIENA 1240

Main analytical approaches

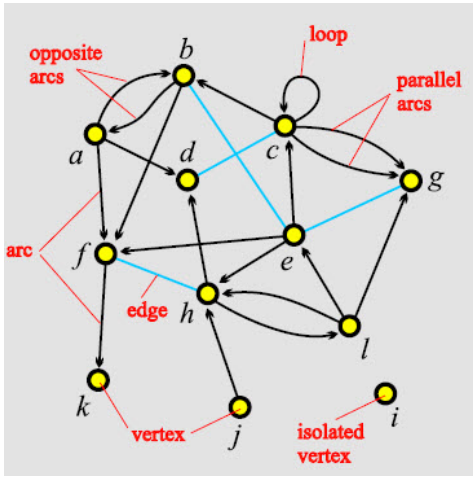
- Mapping topological characteristics
- Tracking the evolution of characteristics in time
- Modeling the generative mechanims underlying the observed structure and dynamics
- Predicting and testing outcomes of strategic interaction

UNIVERSITÀ DI SIENA 1240

Network Object

A **network** $N = (V, L, P, W)$ consists of:

- a **graph** $G = (V, L)$, where V is the set of vertices, A is the set of arcs, E is the set of edges, and $L = E \cup A$ is a set of lines. $\Longrightarrow$ $n = |V|$, $m = |L|$.
- **P vertex functions** / properties: p : V $\Longrightarrow$ A
- **W line value functions** / weights: w : L $\Longrightarrow$ B



Source: Nooy, W., Mrvar, A. and Batagelj, V. (2005). "Exploratory Social Network Analysis with Pajek

UNIVERSITÀ DI SIENA 1240

Resources for complex network analysis

Network Data

- NetWiki
- Stanford Large Network Dataset Collection
- KONECT

Network Analysis Software Tools

- Pajek is a GUI-based program.
- Gephi is an open-source software that offers a stand-alone GUI and a Java software package.
- Octave Toolbox for Matlab
- iGraph is a free software package that can be downloaded as a C library, an R package or a Python extension .

UNIVERSITÀ DI SIENA 1240

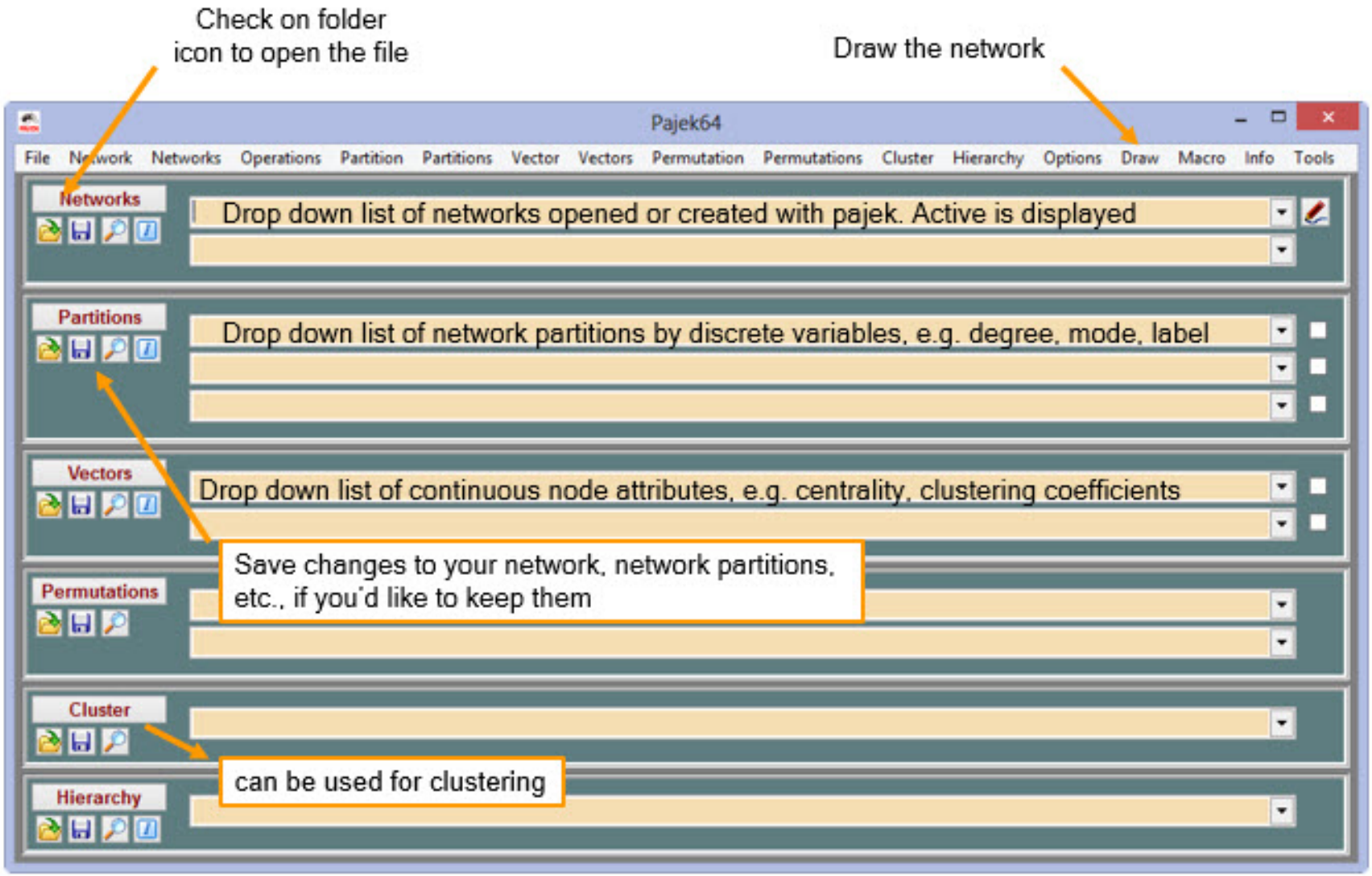
Pajek

Let's get started ...

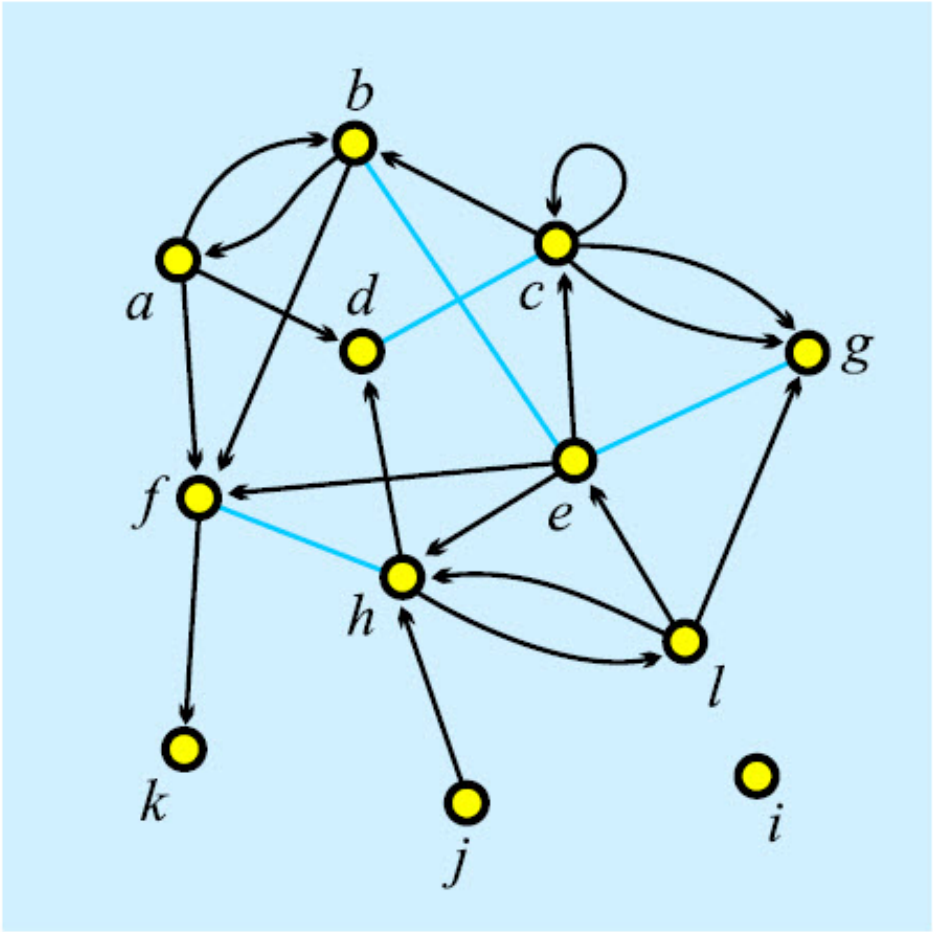
Pajek

UNIVERSITÀ DI SIENA 1240

Pajek main interface



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
```

Source: Nooy, W., Mrvar, A. and Batagelj, V. (2005). "Exploratory Social Network Analysis with Pajek

Representation of Properties

Properties of vertices and lines can be measured in different scales: numerical, ordinal and nominal. Tehy can be input as data or computed from the network.

In pajek numerical properties of vertices are represented by vectors, nominal properties by partitions or as labels of vertices. Numerical property can be displayed as size (width and height) of vertex (figure), as its coordinate; and a nominal property as color or shape of the figure; or as a vertex label (content, size and color). It is possible also to assign in Pajek numerical values to links. they can be displayed as value, thickness or grey level. Nominal values can be assigned as label, color or line pattern.

$\Longrightarrow$ See example in the shared folder

Vertex Properties: CLU, VEC, PER

All the three types of files have the same structure:

*Vertices n $\Longrightarrow$ n is the number of vertices

v_1 $\Longrightarrow$ vertex 1 has value v_1

...

v_n

CLUstering $\Longrightarrow$ partition vertices - nominal or ordinal data about vertices

$v_i \in$N: vertex i belongs to the cluster v_i

VECtor $\Longrightarrow$ numeric data about vertices

$v_i \in$R: the property has value v_i on vertex i

PERmutation $\Longrightarrow$ ordering of vertices

$v_i \in$N: vertex i is at the v_i-th position.

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

Pajek's Projek 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:

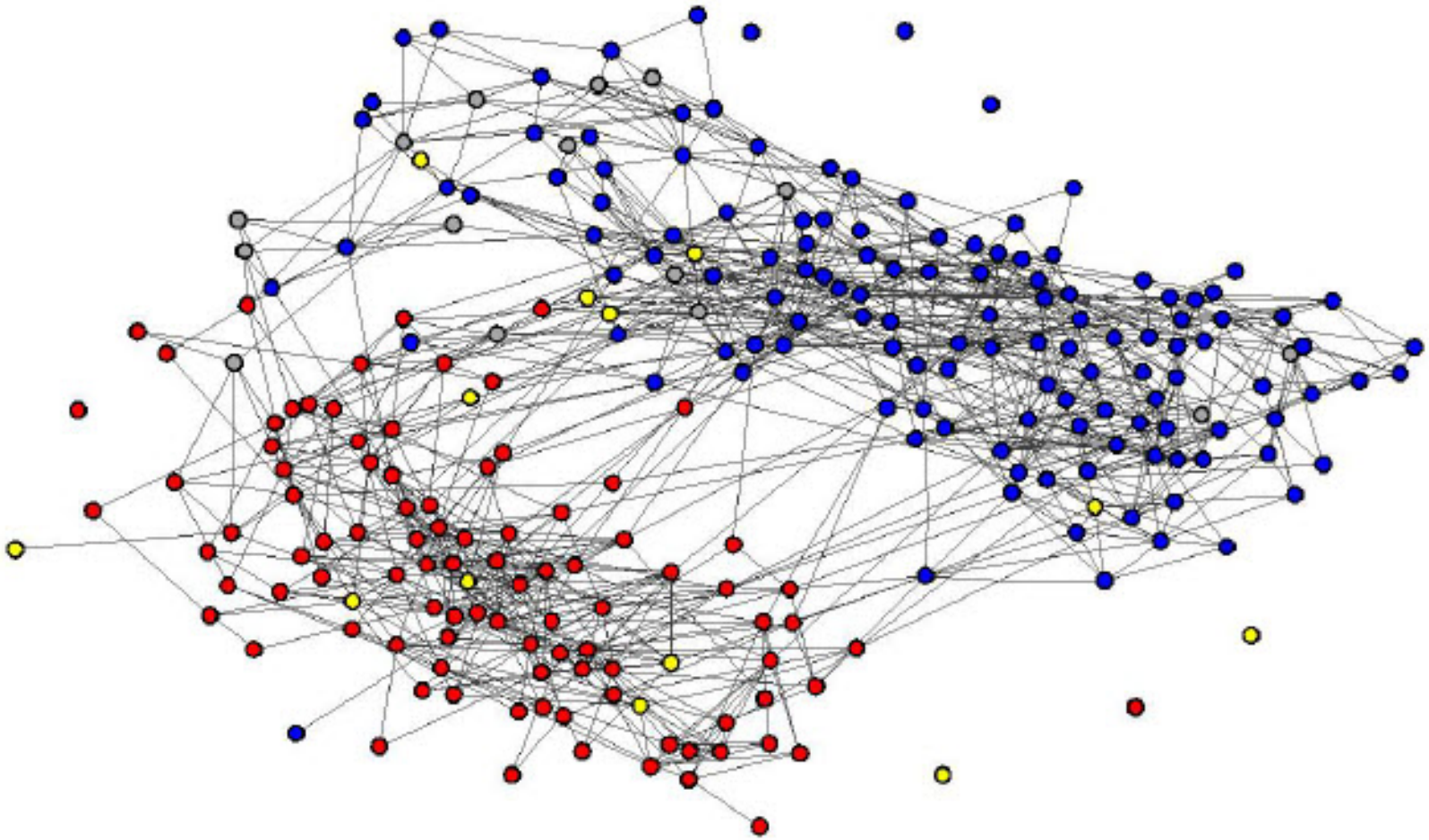
- read all data files in Pajek
- compute some additional data
- delete (dispose) some data
- save all as project file with: File→Pajek Project File→Save

Next time you wil be able to restore everything in the following way: File→Pajek Project File→ Read.

Check the example School (PDF/paj)

UNIVERSITÀ DI SIENA 1240

Display of Properties - School (AddHealth dataset)



Source: The National Longitudinal Study of Adolescent Health (Add Health)

UNIVERSITÀ DI SIENA 1240

Related Operations: Read/Save/Export Network

- **Read network**: Menu FILE → Network → Read: select file.net
  Clicking on the button 'i', INFO in the window Networks, Pajek already shows some basic information about the network
- **Save network**: Menu FILE → Network →
  Save
- **Draw simple network**: Menu DRAW → Network

  We have by default a circular layout, but there are also some predefined algorithms to position vertices in the space. If we go to the Menu LAYOUT, we find different visualization options.
- **To export the picture**: Menu EXPORT → 2D → EPS, SVG, JPEG, BITMAP . . . select your preferred format and save (SVG General saves it in html form, then you can browse the picture).

Find Components

- Menu **NETWORK** → Create Partition With this command we can compute different metrics that will result in a partition of the vertex set into subsets of vertices having certain characteristics.
- Menu **NETWORK**→Create Partition → Components → Weak, Strong . . . (choose the minimum size of components, 1, to obtain all the components).Then we have another network object displayed in the window Partitions.
- Click on the **VIEW shortcut button** in the window **Partitions** and examine the component structure (**Notice:** Components here are called clusters). At this stage, it is important to identify the **giant connected Component**.
- To highlight the components in the graph:
  DRAW→Network + First Partition

  To change the colours of the components: **OPTIONS**→Colours→Partition Colours→Select: colour, input number of partitions

UNIVERSITÀ DI SIENA 1240

Find Components

To Extract Components from the network

- **OPERATIONS**→Network + First partition (components)→Extract subnetwork: n. component

  Important: we may want to save the partition in a separate text file to later use it as input for statistical analysis: go to the **Save** button, by default Pajek save it with the .CLU extension

  Alternatively You can directly export the partition in the form of a vector to **R**:

  1. PARTITION→Copy to vector
  2. TOOLS→R or SPSS→Send to: . . . →Current network

To get the Average Path Length in the giant connected component

- **OPERATIONS**→Network + First partition (components)→Extract subnetwork: select cluster 1 (the giant component)

UNIVERSITÀ DI SIENA 1240

Two Groups of Indicators

1. Centrality and prominence of vertices
   - Degree Centrality
   - Eigenvector Centrality
   - Closeness Centrality
   - Betweenness Centrality
2. Cohesiveness and groups
   - Density
   - Neighborhood
   - Clustering Coefficient
   - Clique
   - k-Core

Centrality and prominence of vertices

- **Degree Centrality:**
  →**NETWORK**→Create Vector→Centrality
- **Eigenvector Centrality:**
  →**NETWORK**→Create Vector→Centrality →Hubs/Authorities:
  Select number of Hubs and Authorities to be identified
- **Closeness Centrality:**
  →**NETWORK**→Create Vector→Centrality→Closeness: ALL
- **Betweenness Centrality:**
  →**NETWORK**→Create Vector→Centrality→Betweennes

Notice: *We can draw the network with vertex size proportional to degree:*

→**DRAW**→Network + First vector

UNIVERSITÀ DI SIENA 1240

Cohesiveness and Groups

- **Neighbours:** NETWORK→Create partition→k-Neighbours: ALL →Select vertex and then the Maximum Distance (if=0, we obtain all neighbourhoods of the vertex selected)
- **Clustering coefficient:** Menu NETWORK →Create Vector→Clustering coefficient: CC1 (Watts-Strogatz Clustering Coefficient).
- **Clique:** Network→Create New Network→Complete Network→Undirected →Select number of vertices, e.g.3. We obtain a new network that is a clique of order 3. →Select the clique as first network in the window, and the original network as second in the window: Networks →Fragment (First in Second)→Find
- **K-cores:** With the original network in the first position: →Select→NETWORK→Create partition→ k-Core: All
  →To extract highest core only: Operations→Network+Partition→Extract SubNetwork

UNIVERSITÀ DI SIENA 1240

Create Random Network

Generate a random network in Pajek from scratch is easy and useful: many times we need to construct a random network against which to benchmark the properties of our actual network.

Have in mind the dimension of your original network: n. of vertices and n. of links . . . we want to create a random network of the same size (and eventually with the same properties) but with links placed at random.

NETWORK→Create random network→Total no. of Arcs Network without multiple lines: YES

If original network is directed we will have to transform arcs into undirected edges:

NETWORK→Create new network→Transform: Arcs→Edges Select All. Then **Create new network**: YES. Remove multiple lines: we can leave ZERO, since there should not be multiple lines.

$\Longrightarrow$ Recalculate network measures, compare and check the results!

UNIVERSITÀ DI SIENA 1240

R Programming

From Pajek to R

Suppose, that vectors v1, v2, v3 and networks n1, n2 were sent from Pajek to R. In the following some simple commands in R using these vectors and networks (matrices) are listed.

Vectors Operations

```
v1+v2
vsum ← v1+v2
a ← sqrt(v1*v2)
...
```

Matrix operations

```
t(n1)   # Transpose network
eigen(n1) # Eigenvalues/eigenvector
hubs <− eigen(n1 %*% t(n1)) $ vec[,1] # Hubs
auth <− eigen(t(n1) %*% n1) $ vec[,1] # Authorities
```

Basic Statistics (var - variance, cov - covariance, cor - correlation)

```
summary(v1)
var(v1)
cov(v1,v2)
cor(v1,v2)
```

UNIVERSITÀ DI SIENA 1240

Graphics

Plotting Charts

```
plot(v1)
plot(v1,v2)
boxplot(v1,v2)
hist(v1)
```

Exporting Graphics

```
pdf("c:/temp/test.pdf")   to pdf file
hist(v1)
dev.off()
```

```
postscript("c:/temp/test.ps")   to ps file
hist(v1)
dev.off()
```

UNIVERSITÀ DI SIENA 1240

Bivariate and Multivariate Analyisis

- **Cross-tabulation**
  table(v1,v2)
  tabl ← table(v1,v2) summary(tabl)) (to get chi-square test)
- **Comparing means (t-test)**
  t.test(v1,v2)
- **Comparing variances**
  var.test(v1, v2)
- **Regression**
  linm ← lm(v1 v2)
  linm ← lm(v1 v2 + v3) (with more variables)

Saving vector from R to Pajek input file

```
savevector ← function(v,direct)
  write(c(paste("*Vertices",length(v)),v),file=direct,ncolumns=1)
```

UNIVERSITÀ DI SIENA 1240

Statistics on Networks: Degree Distribution

In pajek we determine the degrees of vertices and submit them to R.

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

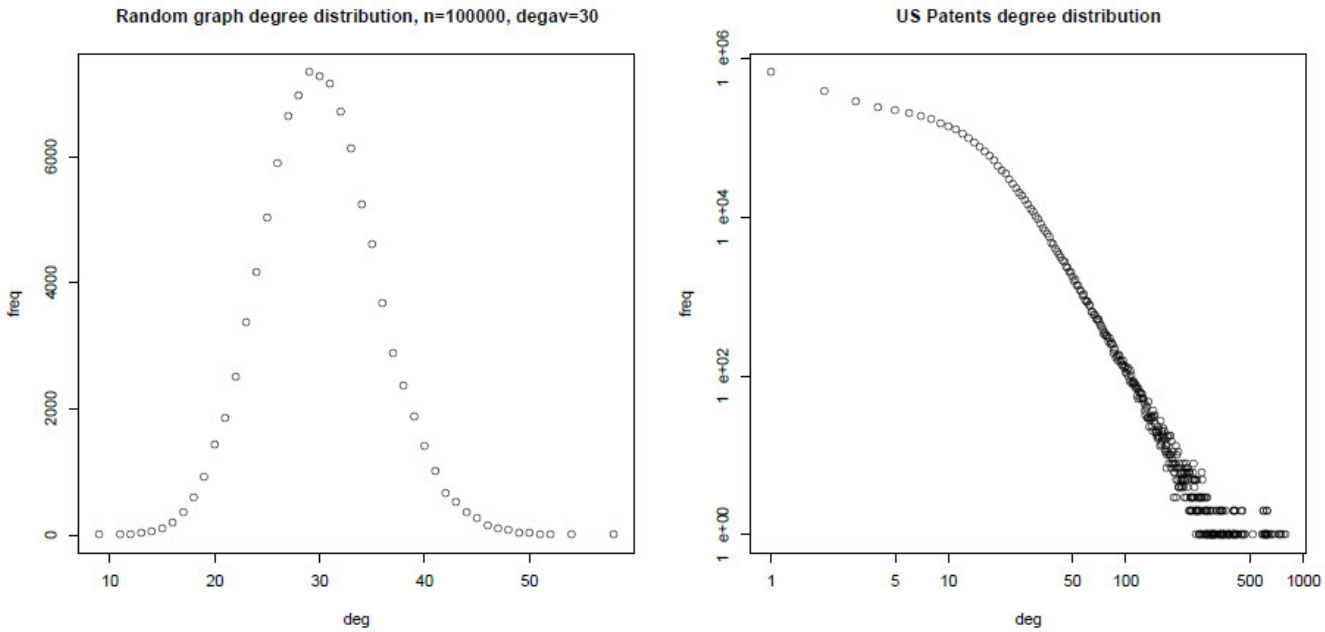
In R we determine their distribution and plot it!

```
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.
(Pay attention for the vertices of degree 0, make problems with log ='xy')

UNIVERSITÀ DI SIENA 1240

Degree Distribution



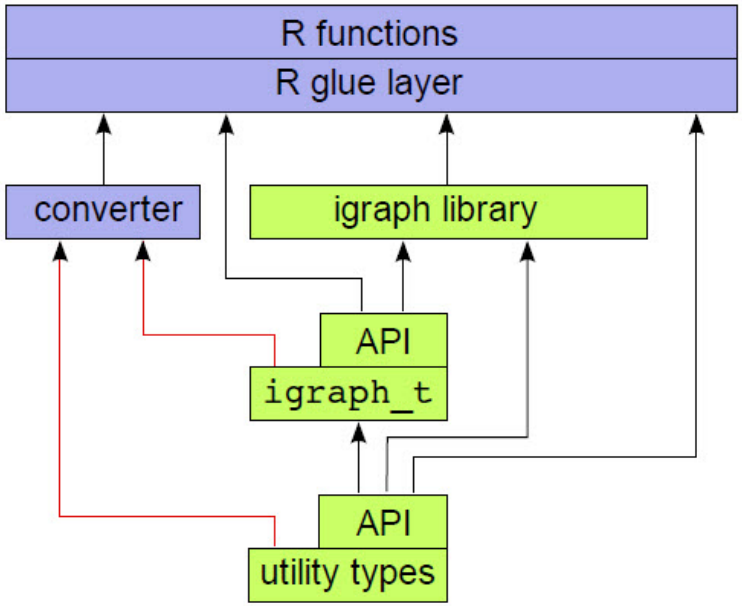
Real-life networs are usually not random in the Erdös and Renyi sense. The analyiss of their distributions gave a new view about the structure - Watts (Small worlds), Barabàsi (nd/networks, Linked)

UNIVERSITÀ DI SIENA 1240

The iGraph Package

- For classic graph theory and network science.
- Core functionality is implemented as a C library.
- High level interfaces from R and Python.
- http://igraph.org/redirect.html

The Architecture



UNIVERSITÀ DI SIENA 1240

The iGraph Package: Loading and Creating graphs

- Vertices are always numbered from zero (!)
- Numberign is continual, from 0 to [V]-1
- We have to "translate" vertex names to ids

Creating graph via vertex ids

```
> g <- graph( c(1,2,2,3,3,4,5,6), directed=FALSE ) # Vertices:6; Edges:4
```

Adding nodes and edges to an already existing graph, e.g.:

```
> g <- graph.empty() + vertices(letters[1:10], color="red")
> g <- g + vertices(letters[11:20], color="blue")
> g <- g + edges(sample(V(g), 30, replace=TRUE), color="green")
```

Naming edges, e.g.:

```
> V(gmis)$name
> g <- graph.ring(10)
> V(g)$name <- sample(letters, vcount(g))
```

Loading Graphs

```
> g <- read.graph("graph.txt", format="edgelist") # From txt
```

UNIVERSITÀ DI SIENA 1240

The iGraph Package: Visualization

Visualization

```
1 > g <- graph.tree(40, 4)
2 > plot(g, layout=layout.circle)
3 > plot(g, layout=l, vertex.color="cyan")
```

```
1 # Force directed layouts
2 > plot(g, layout=layout.fruchterman.reingold)
3 > plot(g, layout=layout.graphopt)
4 > plot(g, layout=layout.kamada.kawai)
```

```
1 # Interactive
2 > tkplot(g, layout=layout.kamada.kawai)
3 > l <- layout.kamada.kawai(g)
```

```
1 # 3D
2 > rglplot(g, layout=l)
```

```
1 # Visual properties
2 > plot(g, layout=l, vertex.color="cyan")
```

UNIVERSITÀ DI SIENA 1240

The iGraph Package: Vertex/Edges sets, attributes

- Assigning attributes: set/get.graph/vertex/edge.attribute.
- V(g) and E(g).
- Smart indexing, e.g. V(g)[color==”white”]
- Easy access of attributes:

```
1 > g <- erdos.renyi.game(100, 1/100)
2 > V(g)$color <- sample( c("red", "black"),
3 vcount(g), rep=TRUE)
4 > E(g)$color <- "grey"
5 > red <- V(g)[ color == "red" ]
6 > bl <- V(g)[ color == "black" ]
7 > E(g)[ red %--% red ]$color <- "red"
8 > E(g)[ bl %--% bl ]$color <- "black"
9 > plot(g, vertex.size=5, layout=
10          layout.fruchterman.reingold)
```

UNIVERSITÀ DI SIENA 1240

The iGraph Package: Generate Graphs

Graph generators

igraph implements many useful graph generators. Most used ones: Erdös and Renyi's model (ER), the Barabasi- Albert model (BA), and the Watts-Strogratz model (WS).

The following commands generate graphs using these models:

```
> er_graph <- erdos.renyi.game(100, 2/100)
> ws_graph <- watts.strogatz.game(1, 100, 4, 0.05)
> ba_graph <- barabasi.game(100)
```

```
# Generate random graph, fixed probability
> g <- erdos.renyi.game(20, 0.3)
> plot(g, layout=layout.fruchterman.reingold, vertex.label=NA, vertex.size=5)

# Generate random graph, fixed number of arcs
> g <-erdos.renyi.game(20, 15, type='gnm')

# Generate preferential attachment graph
> g <- barabasi.game(60, power=1, zero.appeal=1.3)
```

UNIVERSITÀ DI SIENA 1240

The iGraph Package: Measuring Graphs

Connected Component

> subcomponent(g, 4, mode="out") # Nodes reachable from, e.g node 4. >4
subcomponent(g, 4, mode="in") # Nodes who can reach node4
> clusters(g, mode="weak")
> myc <- clusters(g, mode="strong")
> mycolor <- c('green', 'yellow', 'red', 'skyblue')
> V(g)$color <- mycolor[myc$membership + 1]
> plot(g, layout=layout.fruchterman.reingold)

Minimum Spanning Tree algorithm

- **# Create the graph and assign random edge weights**
  > g <-erdos.renyi.game(12, 0.35)
  > E(g)$weight <- round(runif(length(E(g))),2) * 50 > plot(g, layout=layout.fruchterman.reingold, edge.label=E(g)$weight)
- **# Compute the minimum spanning tree**
  > mst <- minimum.spanning.tree(g)
  > plot(mst, layout=layout.reingold.tilford, edge.label=E(mst)$weight)

UNIVERSITÀ DI SIENA 1240

The iGraph Package: Basic Graph Algorithms

Centrality Measures

> degree(g)

> closeness(g)

> betweenness(g)

> transitivity(g, type="local") **# Local cluster coefficient**

> evcent(g)$vector **# Eigenvector centrality**

Eigenvector Centrality Vs Betweenes centrality

- **# Create a graph**

  > g <- barabasi.game(100, directed=F)

  > lay <- layout.fruchterman.reingold(g)

- **# Plot the eigevector and betweenness centrality**

  > plot(evcent(g)$vector, betweenness(g))

  > text(evcent(g)$vector, betweenness(g), 0:100, cex=0.6, pos=4)

  > V(g)[12]$color <- 'red'

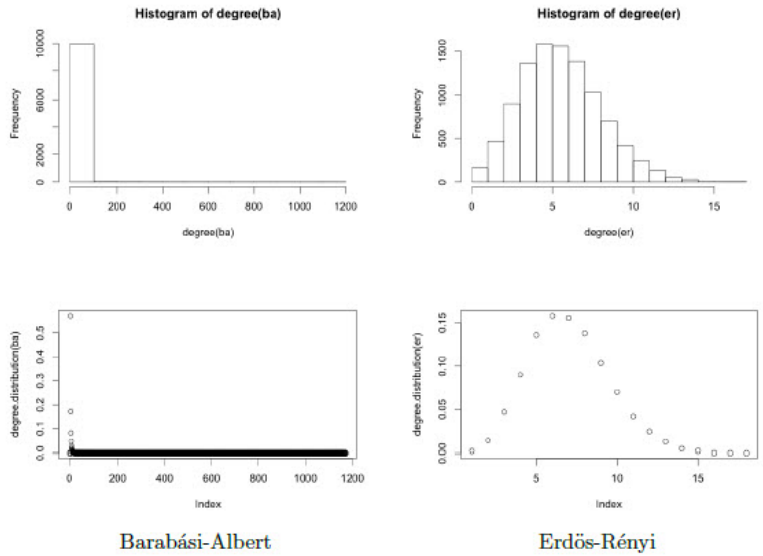
  > V(g)[8]$color <- 'green'

  > plot(g, layout=lay, vertex.size=8, vertex.label.cex=0.6)

The iGraph Package: Basic Graph Algorithms

For degree and distribution (Barabasi Network Vs Erdös and Renyi's Random Graph)

```
> g <- graph.ring(10)
> degree(g)
> ba <- barabasi.game(10000, m=3)
> p_hat <- ecount(ba)/ ((vcount(ba)-1)*vcount(ba)/2)
> er <- erdos.renyi.game(10000, p_hat)
> degree.distribution(er)
> hist(degree(er))
> hist(degree(ba))
> plot(degree.distribution(er))
> plot(degree.distribution(ba))
```



Barabási-Albert Erdös-Rényi

UNIVERSITÀ DI SIENA 1240

Thank you for the attention!