igraph – a package for network analysis

Gábor Csárdi
Gabor.Csardi@unil.ch

Department of Medical Genetics,
University of Lausanne, Lausanne, Switzerland

Outline

1. Why another graph package?

2. igraph architecture, data model and data representation

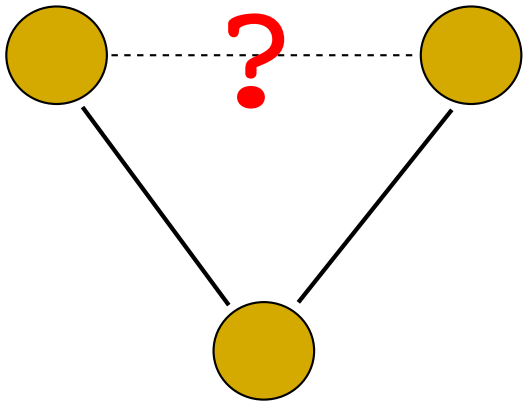
3. Manipulating graphs

4. Features and their time complexity

Why another graph package?

- graph is slow. RBGL is slow, too.

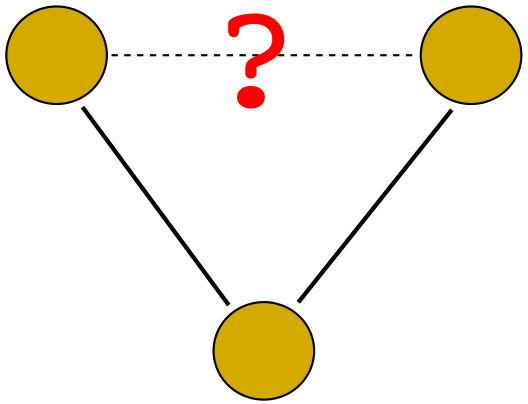
```
1  > ba2                                    # graph & RBGL
2   A graphNEL graph with undirected edges
3   Number of Nodes = 100000
4   Number of Edges = 199801
```



Why another graph package?

- graph is slow. RBGL is slow, too.

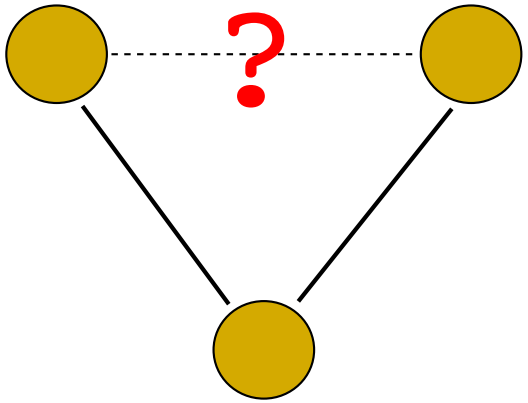
```
1  > ba2                                        # graph & RBGL
2   A graphNEL graph with undirected edges
3   Number of Nodes = 100000
4   Number of Edges = 199801
5  > system.time(RBGL::transitivity(ba2))
6       user  system elapsed
7      7.517   0.000   7.567
```



Why another graph package?

- graph is slow. RBGL is slow, too.

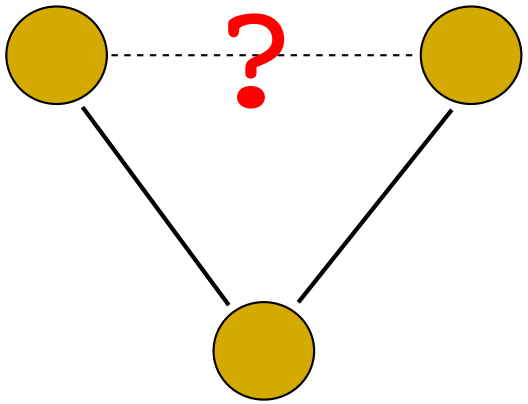
```
> ba2                                    # graph & RBGL
 A graphNEL graph with undirected edges
 Number of Nodes = 100000
 Number of Edges = 199801
> system.time(RBGL::transitivity(ba2))
    user  system elapsed
   7.517   0.000   7.567
> summary(ba)                            # igraph
 Vertices: 1e+05
 Edges: 199801
 Directed: FALSE
 No graph attributes.
 No vertex attributes.
 No edge attributes.
```



Why another graph package?

- graph is slow. RBGL is slow, too.

```
> ba2                                    # graph & RBGL
 A graphNEL graph with undirected edges
 Number of Nodes = 100000
 Number of Edges = 199801
> system.time(RBGL::transitivity(ba2))
    user  system elapsed
   7.517   0.000   7.567
> summary(ba)                            # igraph
 Vertices: 1e+05
 Edges: 199801
 Directed: FALSE
 No graph attributes.
 No vertex attributes.
 No edge attributes.
> system.time(igraph::transitivity(ba))
    user  system elapsed
   0.328   0.000   0.335
```



Why another graph package?

- sna is slow. network is slow, too.

```
> net2                                        # SNA & network
  Network attributes:
   vertices = 1e+05
   directed = TRUE
   hyper = FALSE
   loops = FALSE
   multiple = FALSE
   bipartite = FALSE
   total edges= 199801
     missing edges= 0
     non-missing edges= 199801
  ...
```

Why another graph package?

- sna is slow. network is slow, too.

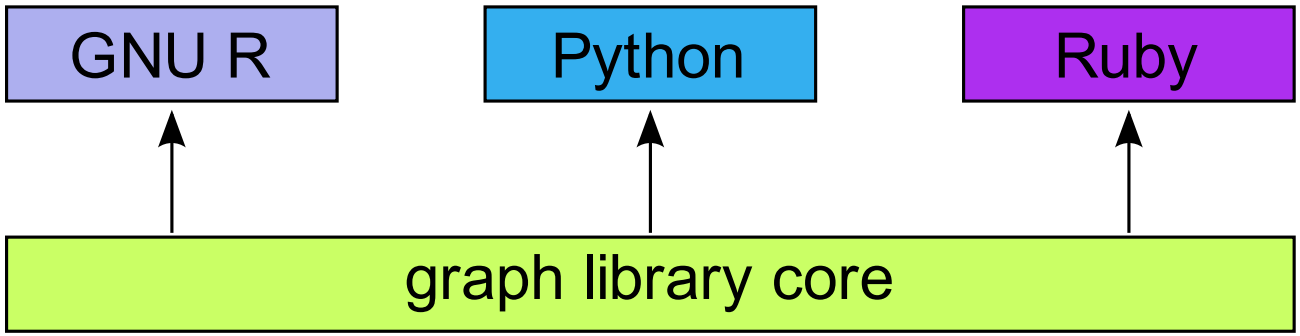
```
> net2                                          # SNA & network
  Network attributes:
   vertices = 1e+05
   directed = TRUE
   hyper = FALSE
   loops = FALSE
   multiple = FALSE
   bipartite = FALSE
   total edges= 199801
     missing edges= 0
     non-missing edges= 199801
  ...
> gtrans(net2)
 Error in matrix(0, nr = network.size(x), nc = network.size(x)) :
   too many elements specified
```

Why another graph package?

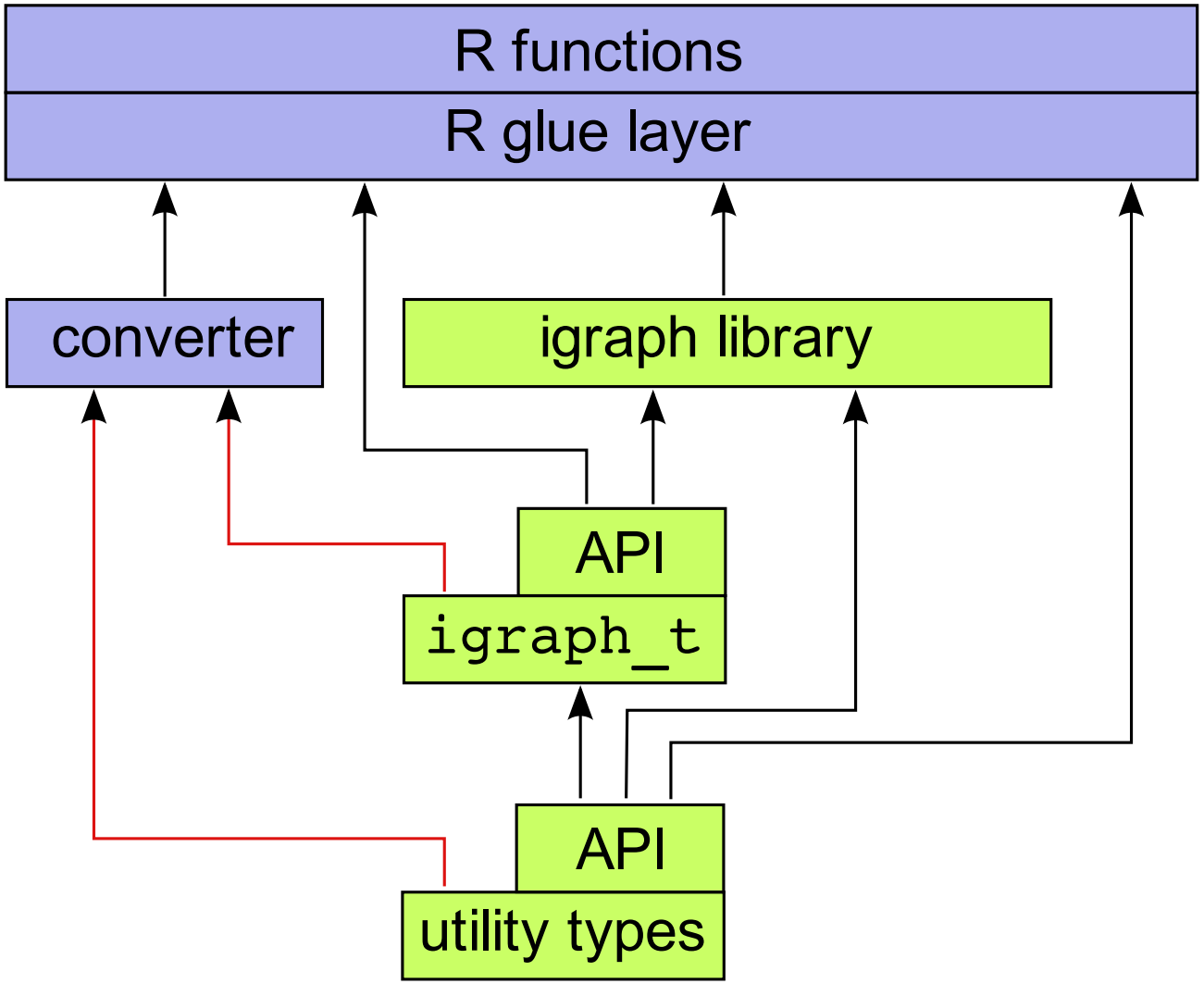
- `graph` is slow. `RBGL` is slow, too.
- `sna` is slow. `network` is slow, too.
- A generic solution was needed, i.e. a common C layer, that can be interfaced from C/C++, R, Python, etc.

Why another graph package?

- `graph` is slow. `RBGL` is slow, too.
- `sna` is slow. `network` is slow, too.
- A generic solution was needed, i.e. a common C layer, that can be interfaced from C/C++, R, Python, etc.



The igraph architecture



Dependencies

- Standard C/C++ libraries.

Dependencies

- Standard C/C++ libraries.
- `stats` package, this is part of `base`.

Dependencies

- Standard C/C++ libraries.
- `stats` package, this is part of `base`.
- Optional: `libxml2` library, for reading GraphML files (included in Windows builds).

Dependencies

- Standard C/C++ libraries.
- `stats` package, this is part of `base`.
- Optional: `libxml2` library, for reading GraphML files (included in Windows builds).
- Optional: GMP library, graph automorphisms (not included in Windows builds).

GMP

«Arithmetic without limitations»

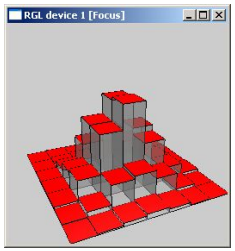
Dependencies

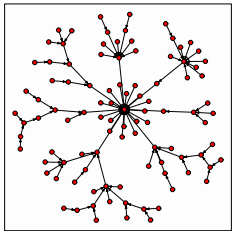
- Standard C/C++ libraries.
- `stats` package, this is part of `base`.
- Optional: `libxml2` library, for reading GraphML files (included in Windows builds).

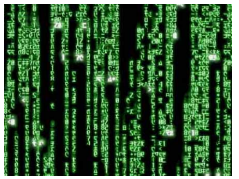
- Optional: GMP library, graph automorphisms (not included in Windows builds).

GMP
«Arithmetic without limitations»

- Suggested packages: `stats4`, `rgl`, `tcltk`, `RSQLite`, `digest`, `graph`, `Matrix`.

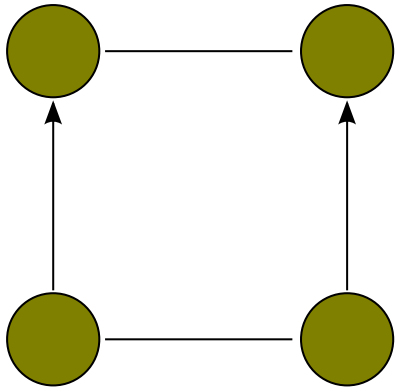






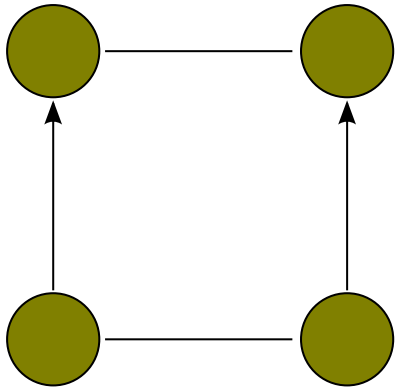
The igraph data model, what cannot be represented

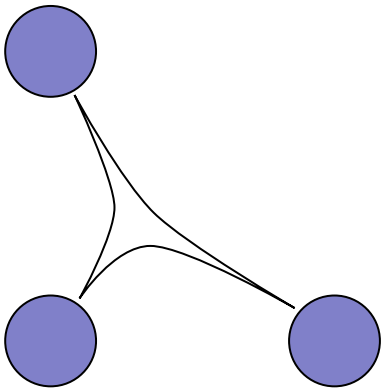
"Mixed" graphs, with undirected and directed edges.
You can "emulate" them via graph attributes.



The igraph data model, what cannot be represented

"Mixed" graphs, with undirected and directed edges.
You can "emulate" them via graph attributes.

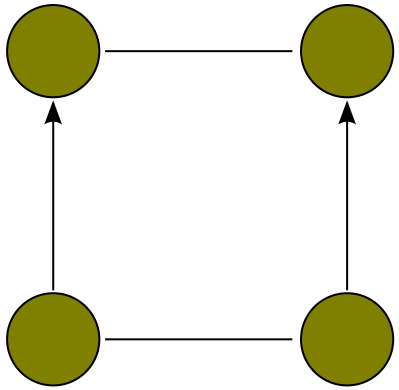


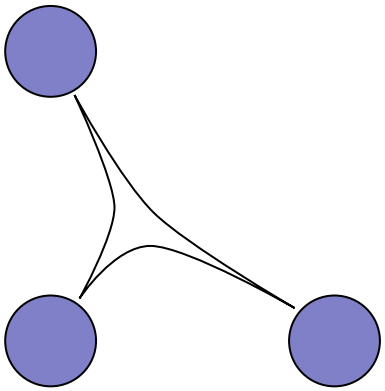


Hypergraphs. Perhaps see the `hypergraph` package.

The igraph data model, what cannot be represented

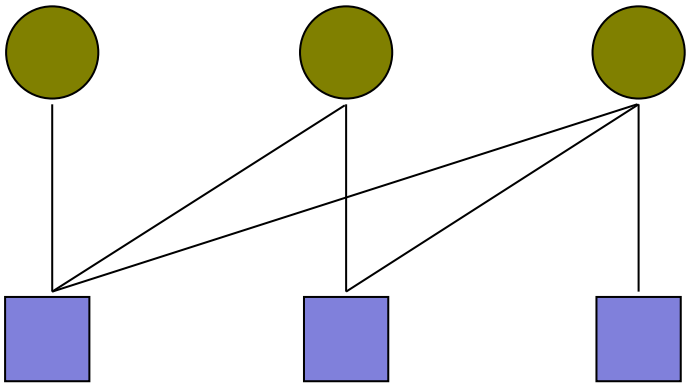
"Mixed" graphs, with undirected and directed edges.
You can "emulate" them via graph attributes.





Hypergraphs. Perhaps see the `hypergraph` package.

No direct support for bipartite (two-mode) graphs.
It is possible to handle them via graph attributes.



Graph representation, sparse graphs

Flat data structures, indexed edge lists. Easy to handle, good for many kind of questions.

Alice	Bob
Bob	Diana
Cecil	Diana
Alice	Esmeralda
Diana	Esmeralda
Cecil	Fabien
Esmeralda	Fabien
Bob	Gigi
Cecil	Gigi
Diana	Gigi
Alice	Helen
Bob	Helen
Diana	Helen
Esmeralda	Helen
Fabien	Helen
Gigi	Helen
Diana	Iannis
Esmeralda	Iannis
Alice	Jennifer
Helen	Jennifer

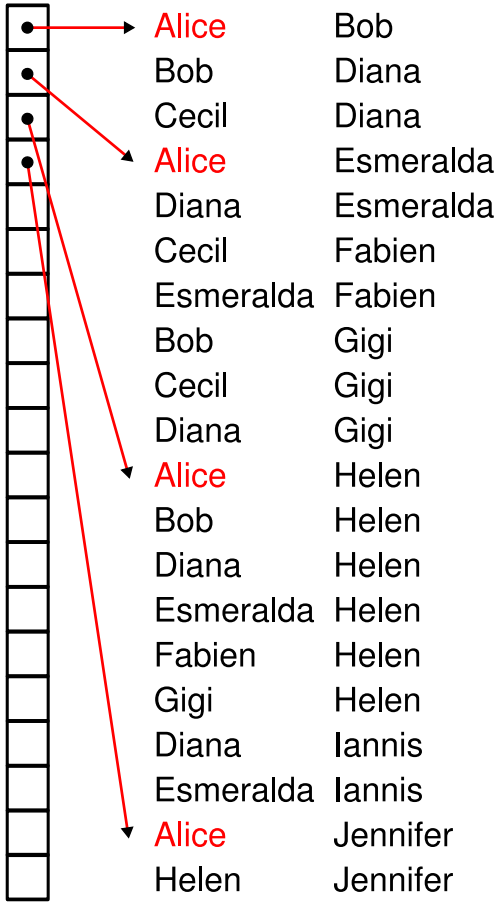
Graph representation, sparse graphs

Flat data structures, indexed edge lists. Easy to handle, good for many kind of questions.

Alice	Bob
Bob	Diana
Cecil	Diana
Alice	Esmeralda
Diana	Esmeralda
Cecil	Fabien
Esmeralda	Fabien
Bob	Gigi
Cecil	Gigi
Diana	Gigi
Alice	Helen
Bob	Helen
Diana	Helen
Esmeralda	Helen
Fabien	Helen
Gigi	Helen
Diana	Iannis
Esmeralda	Iannis
Alice	Jennifer
Helen	Jennifer

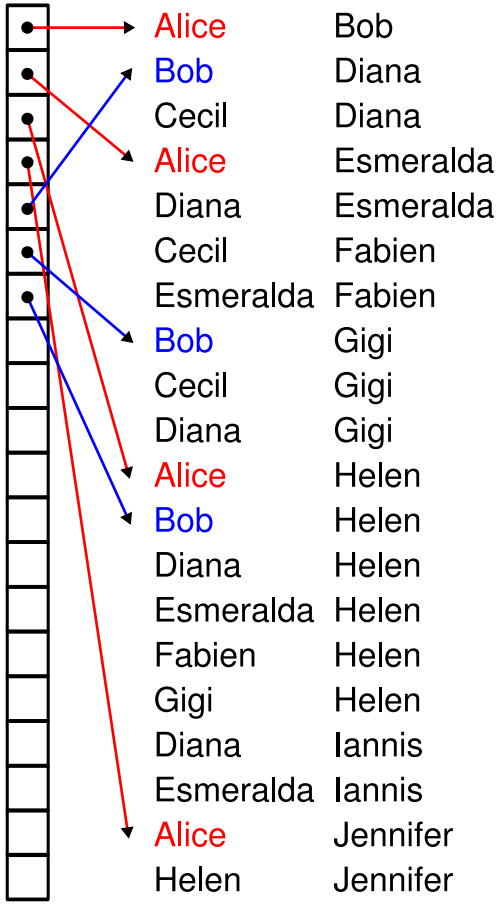
Graph representation, sparse graphs

Flat data structures, indexed edge lists. Easy to handle, good for many kind of questions.



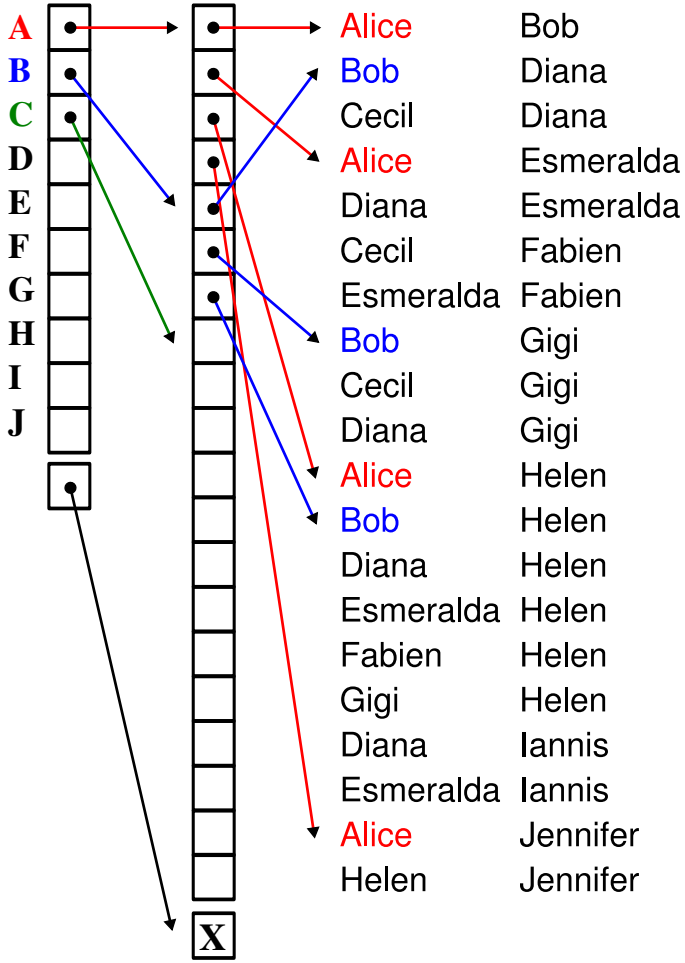
Graph representation, sparse graphs

Flat data structures, indexed edge lists. Easy to handle, good for many kind of questions.



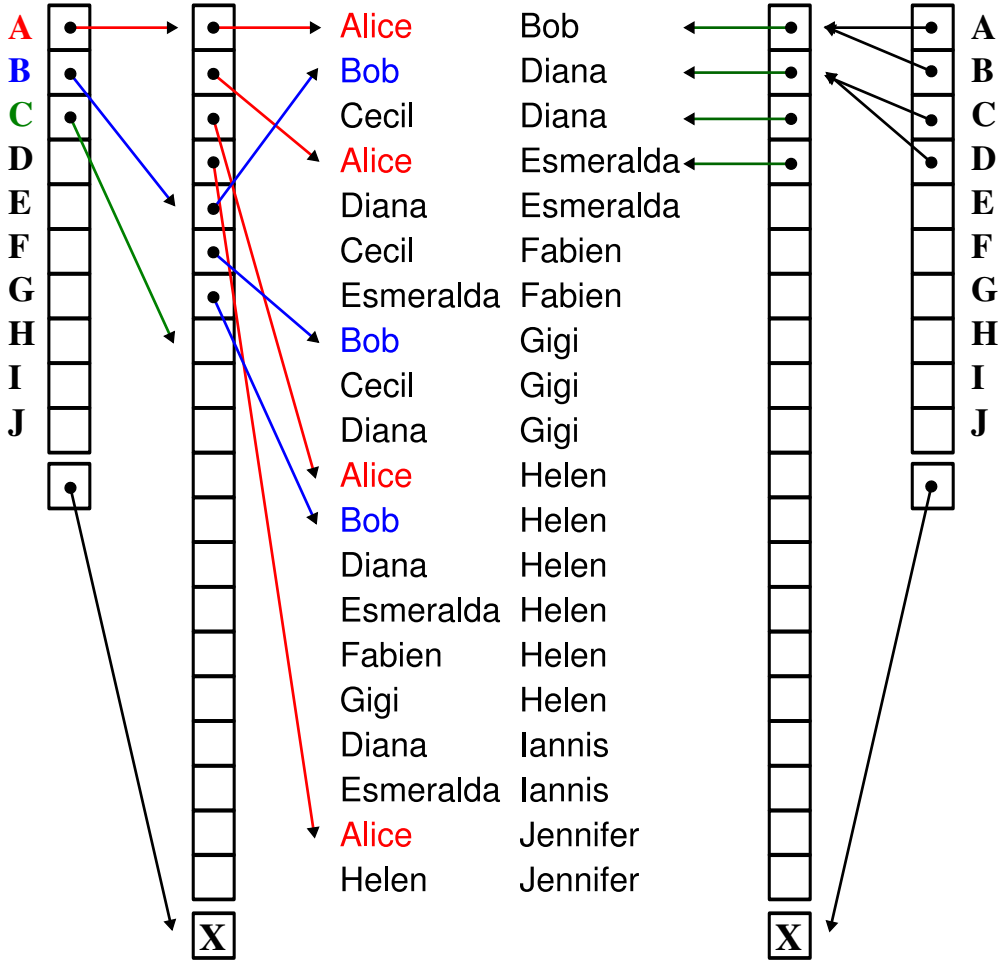
Graph representation, sparse graphs

Flat data structures, indexed edge lists. Easy to handle, good for many kind of questions.



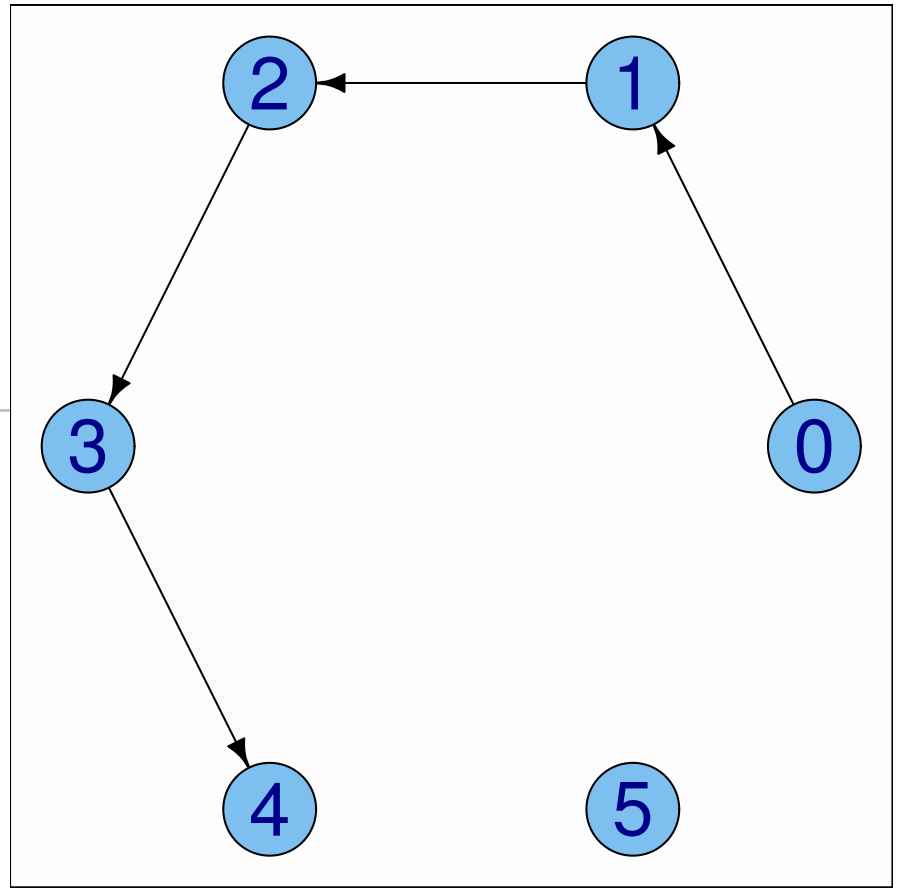
Graph representation, sparse graphs

Flat data structures, indexed edge lists. Easy to handle, good for many kind of questions.



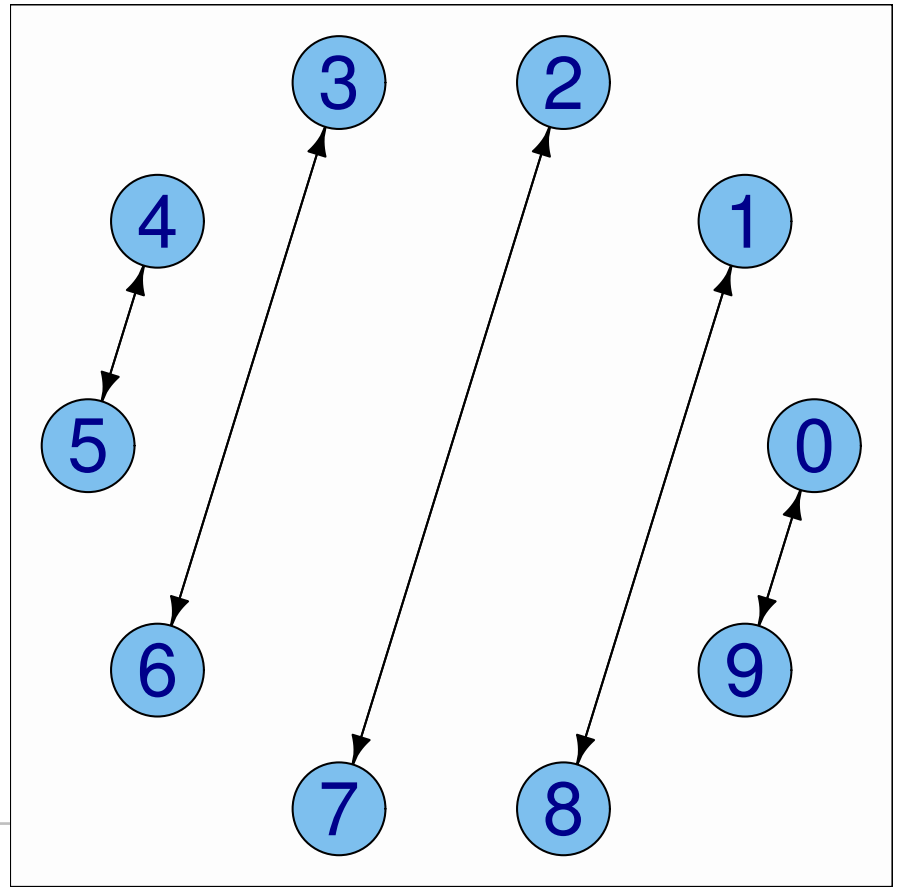
Creating graphs, via vertex ids

```
> g <- graph( c(0,1, 1,2, 2,3, 3,4), n=6, directed=TRUE )
> g
 Vertices: 6
 Edges: 4
 Directed: TRUE
 Edges:

 [0] 0 -> 1
 [1] 1 -> 2
 [2] 2 -> 3
 [3] 3 -> 4
```



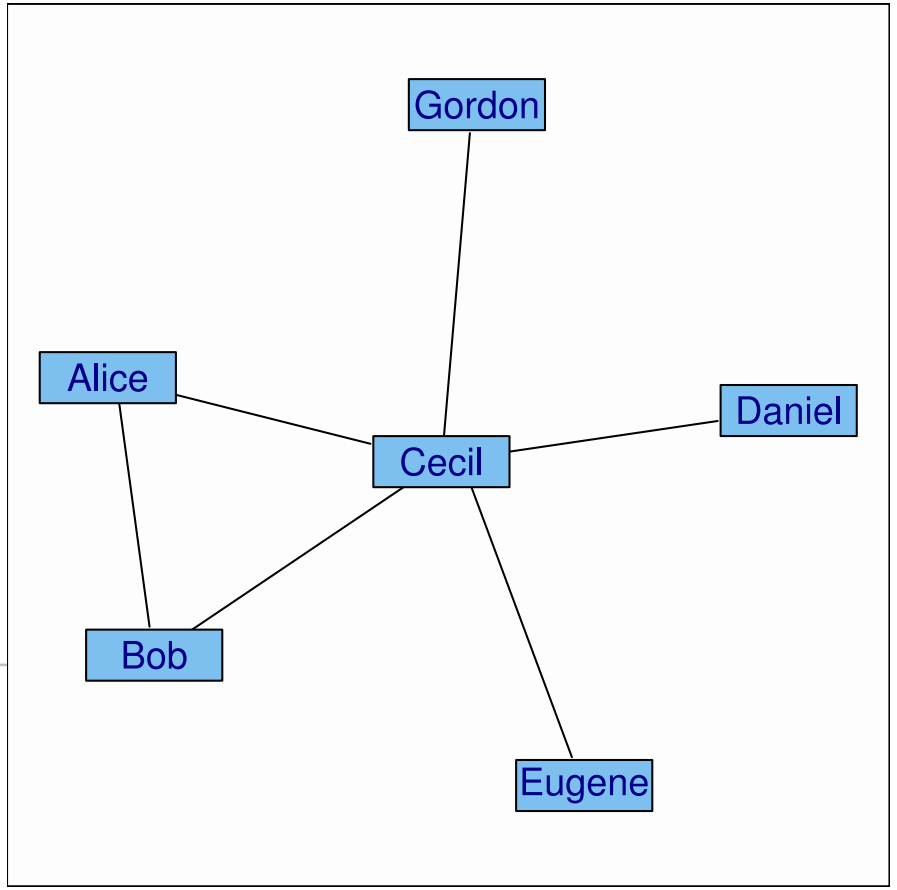
Creating graphs, via vertex ids

```
> el <- cbind(0:9, 9:0)
> g <- graph( t(el), directed=TRUE)
> g
 Vertices: 10
 Edges: 10
 Directed: TRUE
 Edges:

 [0] 0 -> 9
 [1] 1 -> 8
 [2] 2 -> 7
 [3] 3 -> 6
 [4] 4 -> 5
 [5] 5 -> 4
 [6] 6 -> 3
 [7] 7 -> 2
 [8] 8 -> 1
 [9] 9 -> 0
```



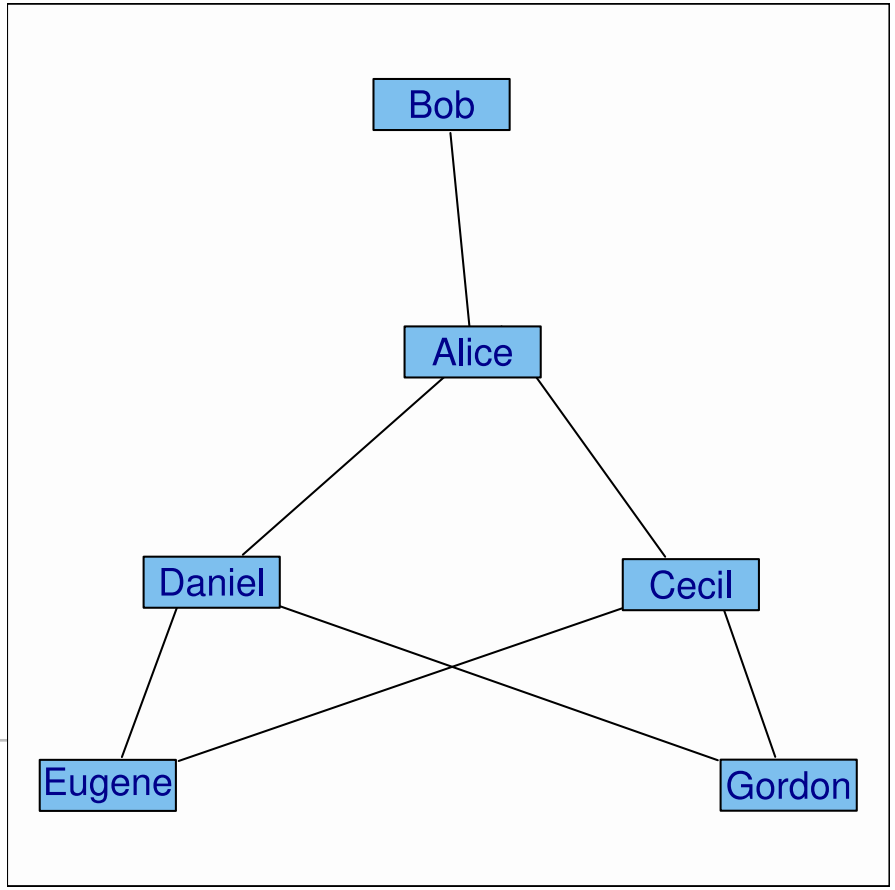
Creating graphs, graph.formula

```
# A simple undirected graph
> g <- graph.formula( Alice-Bob-Cecil-Alice,
      Daniel-Cecil-Eugene, Cecil-Gordon )
> g
 Vertices: 6
 Edges: 6
 Directed: FALSE
 Edges:

 [0] Alice  -- Bob
 [1] Bob    -- Cecil
 [2] Alice  -- Cecil
 [3] Cecil  -- Daniel
 [4] Cecil  -- Eugene
 [5] Cecil  -- Gordon
```



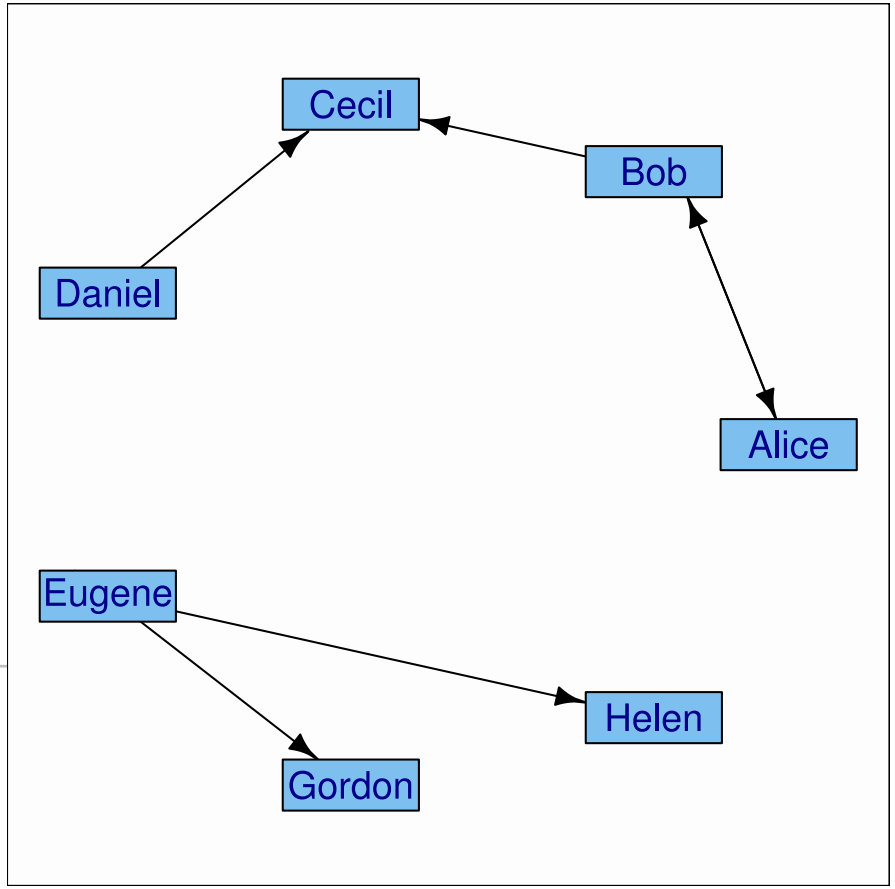
Creating graphs, graph.formula

```
# Another undirected graph, ":" notation
> g2 <- graph.formula( Alice-Bob:Cecil:Daniel,
        Cecil:Daniel-Eugene:Gordon )
> g2
 Vertices: 6
 Edges: 7
 Directed: FALSE
 Edges:

 [0] Alice  -- Bob
 [1] Alice  -- Cecil
 [2] Alice  -- Daniel
 [3] Cecil  -- Eugene
 [4] Cecil  -- Gordon
 [5] Daniel -- Eugene
 [6] Daniel -- Gordon
```



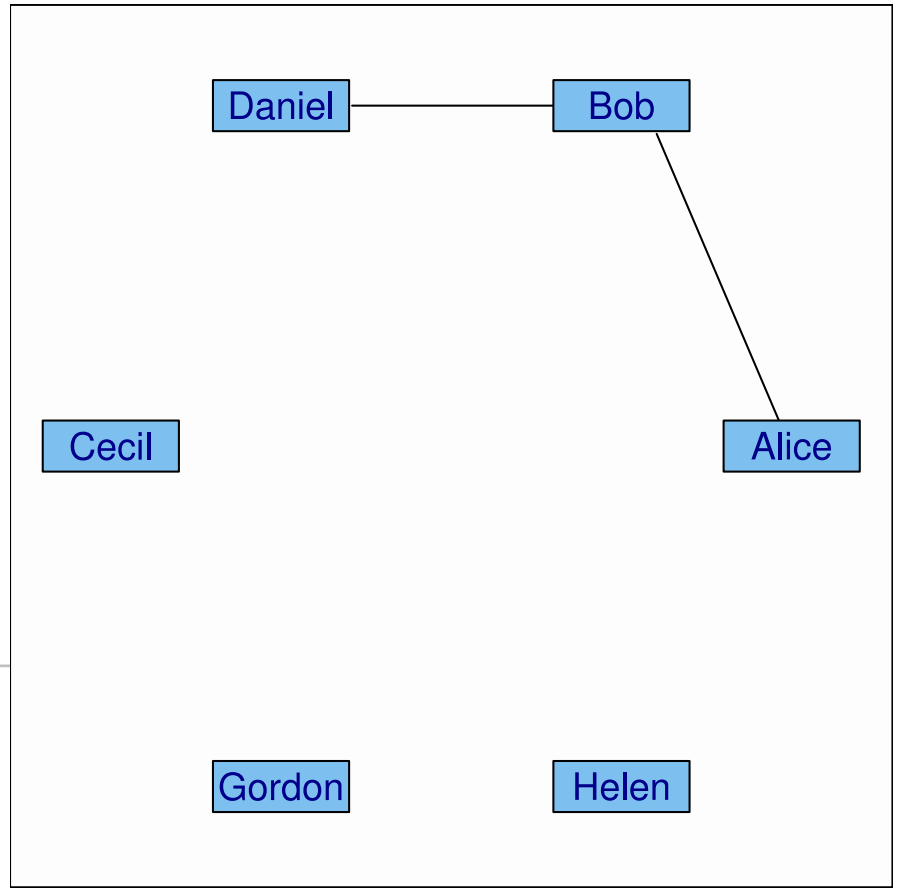
Creating graphs, graph.formula

```
# A directed graph
> g3 <- graph.formula( Alice +-+ Bob --+ Cecil
        +-- Daniel, Eugene --+ Gordon:Helen )
> g3
 Vertices: 7
 Edges: 6
 Directed: TRUE
 Edges:

 [0] Bob    -> Alice
 [1] Alice  -> Bob
 [2] Bob    -> Cecil
 [3] Daniel -> Cecil
 [4] Eugene -> Gordon
 [5] Eugene -> Helen
```



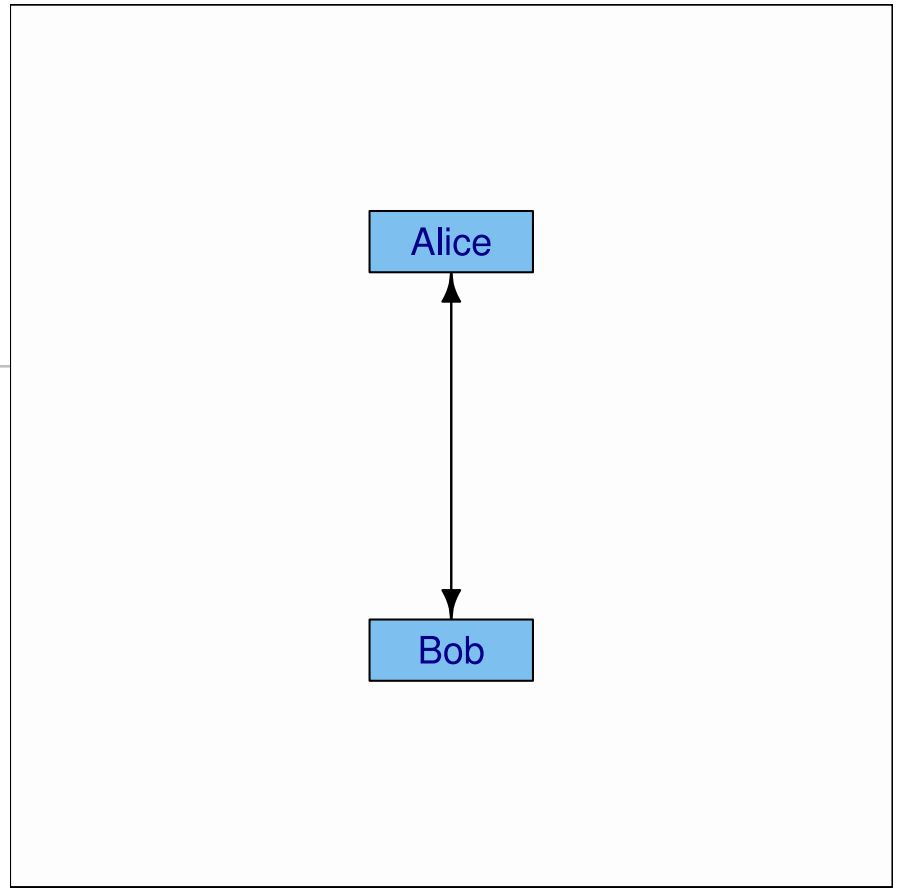
Creating graphs, graph.formula

```
# A graph with isolate vertices
> g4 <- graph.formula( Alice -- Bob -- Daniel,
        Cecil:Gordon, Helen )
> g4
 Vertices: 6
 Edges: 2
 Directed: FALSE
 Edges:

 [0] Alice  -- Bob
 [1] Bob    -- Daniel
> V(g4)
 Vertex sequence:
 [1] "Alice"  "Bob"    "Daniel"
 [4] "Cecil"  "Gordon" "Helen"
```



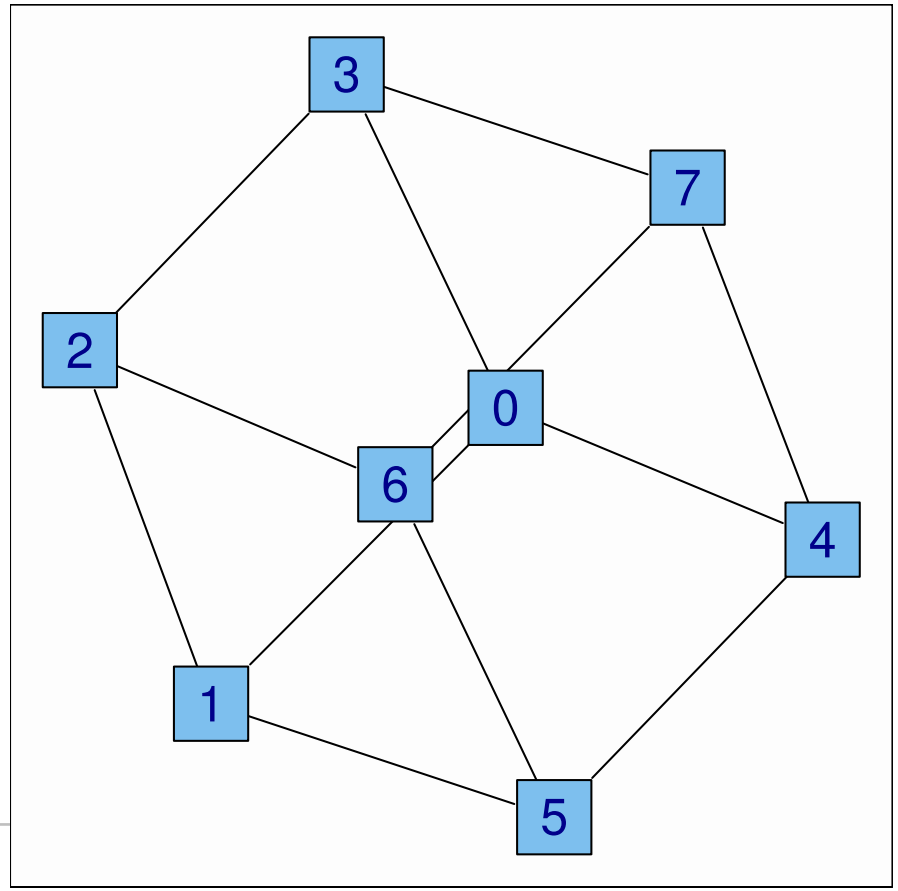
Creating graphs, graph.formula

```
1  # "Arrows" can be arbitrarily long
2  > g5 <- graph.formula( Alice +---------+ Bob )
3  > g5
4   Vertices: 2
5   Edges: 2
6   Directed: TRUE
7   Edges:
8
9   [0] Bob   -> Alice
10  [1] Alice -> Bob
```



Creating graphs, `graph.famous`

```
> graph.famous("Cubical")
 Vertices: 8
 Edges: 12
 Directed: FALSE
 Edges:

 [0]  0 -- 1
 [1]  1 -- 2
 [2]  2 -- 3
 [3]  0 -- 3
 [4]  4 -- 5
 [5]  5 -- 6
 [6]  6 -- 7
 [7]  4 -- 7
 [8]  0 -- 4
 [9]  1 -- 5
 [10] 2 -- 6
 [11] 3 -- 7
```



Creating graphs, graph.data.frame

```
> traits <- read.csv("traits.csv", head=F)
> traits
                    V1 V2 V3
 1      Alice Anderson 48  F
 2        Bob Bradford 33  M
 3        Cecil Connor 45  F
 4       David Daugher 34  M
 5  Esmeralda Escobar 21  F
 6        Frank Finley 36  M
 7          Gabi Garbo 44  F
 8          Helen Hunt 40  F
 9         Iris Irving 25  F
 10        James Jones 47  M
> colnames(traits) <- c("name", "age", "gender")
> traits[,1] <- sapply(strsplit(as.character(traits[,1]), " "), "[", 1)
```

Creating graphs, graph.data.frame

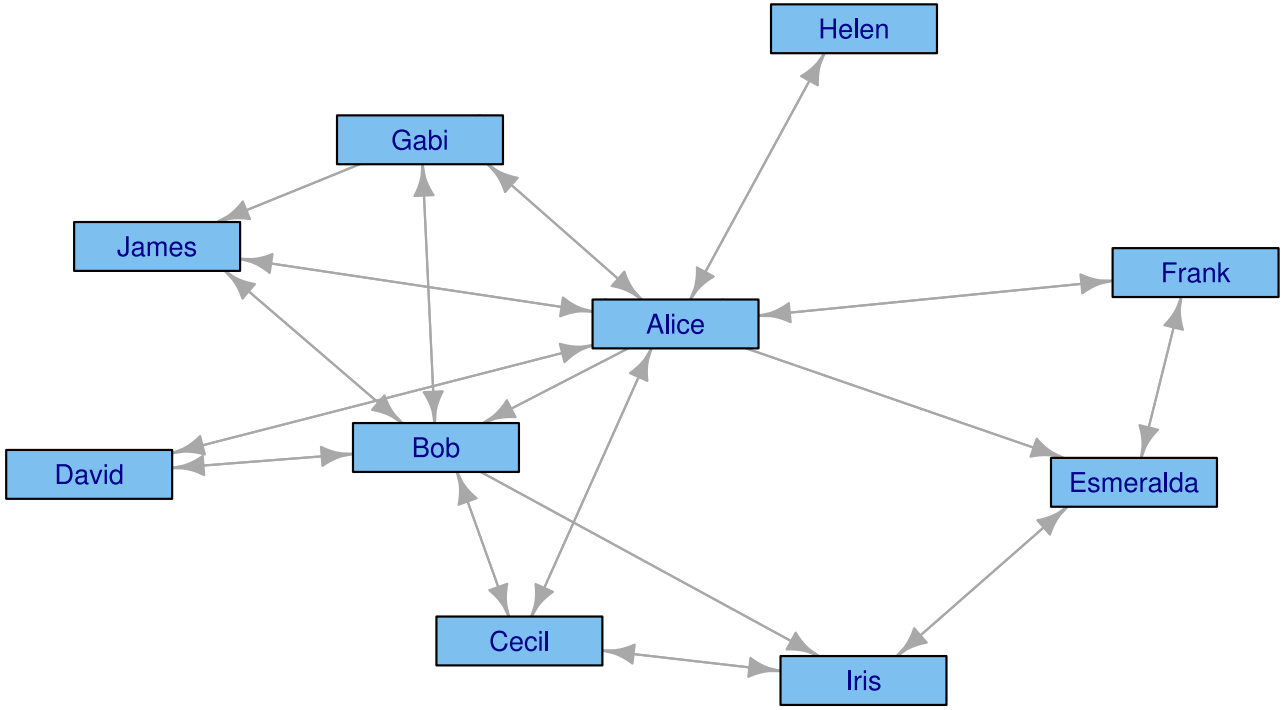
```
> relations <- read.csv("relations.csv", head=F)
> relations
            V1          V2 V3 V4 V5
 1          Bob       Alice  N  4  4
 2       Cecil         Bob  N  5  5
 3       Cecil       Alice  Y  5  5
 4       David       Alice  N  3  4
 5       David         Bob  N  4  2
 6  Esmeralda       Alice  Y  4  3
 7       Frank       Alice  N  3  2
 8       Frank  Esmeralda  N  4  4
 9        Gabi         Bob  Y  5  5
 10       Gabi       Alice  N  3  0
 11      Helen       Alice  N  4  1
 12       Iris       Cecil  N  0  1
 ...
> colnames(relations) <- c("from", "to", "same.room",
          "friendship", "advice")
```

Creating graphs, graph.data.frame

```
> orgnet <- graph.data.frame(relations, vertices=traits)
> summary(orgnet)
 Vertices: 10
 Edges: 34
 Directed: TRUE
 No graph attributes.
 Vertex attributes: name, age, gender.
 Edge attributes: same.room, friendship, advice.
```

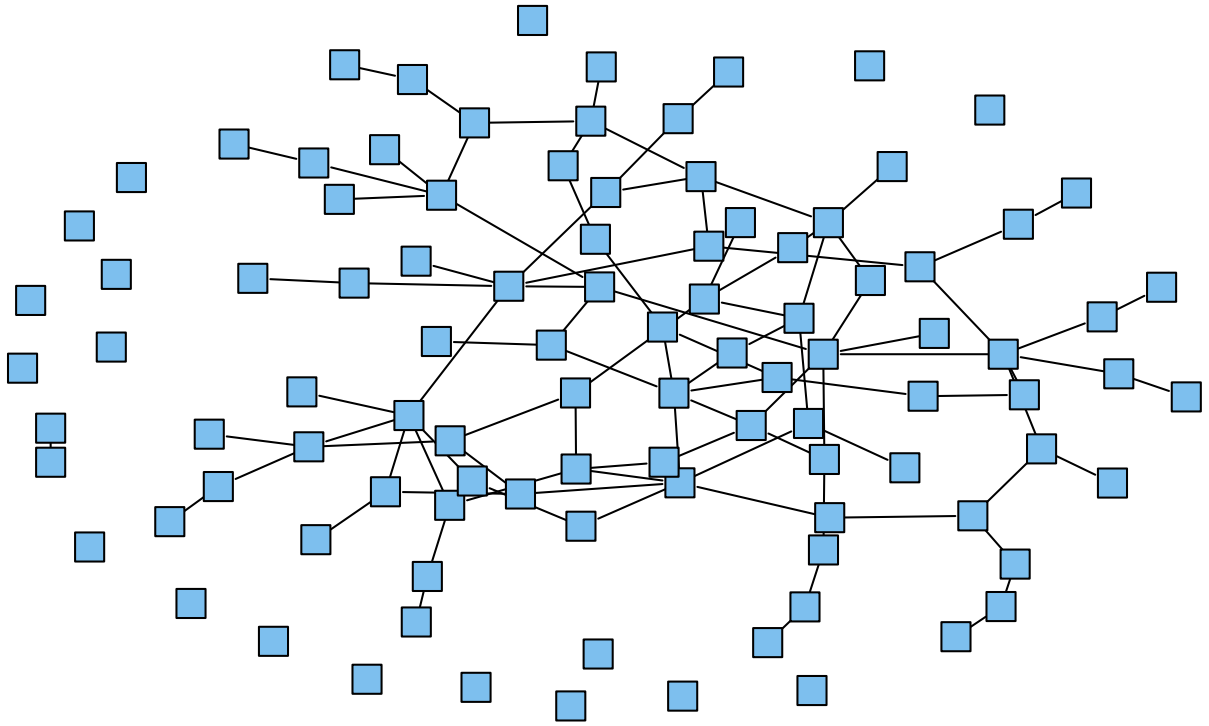
Creating graphs, graph.data.frame

```
> plot(orgnet, layout=layout.kamada.kawai, vertex.label=V(orgnet)$name,
    vertex.shape="rectangle", vertex.size=20, asp=FALSE)
```



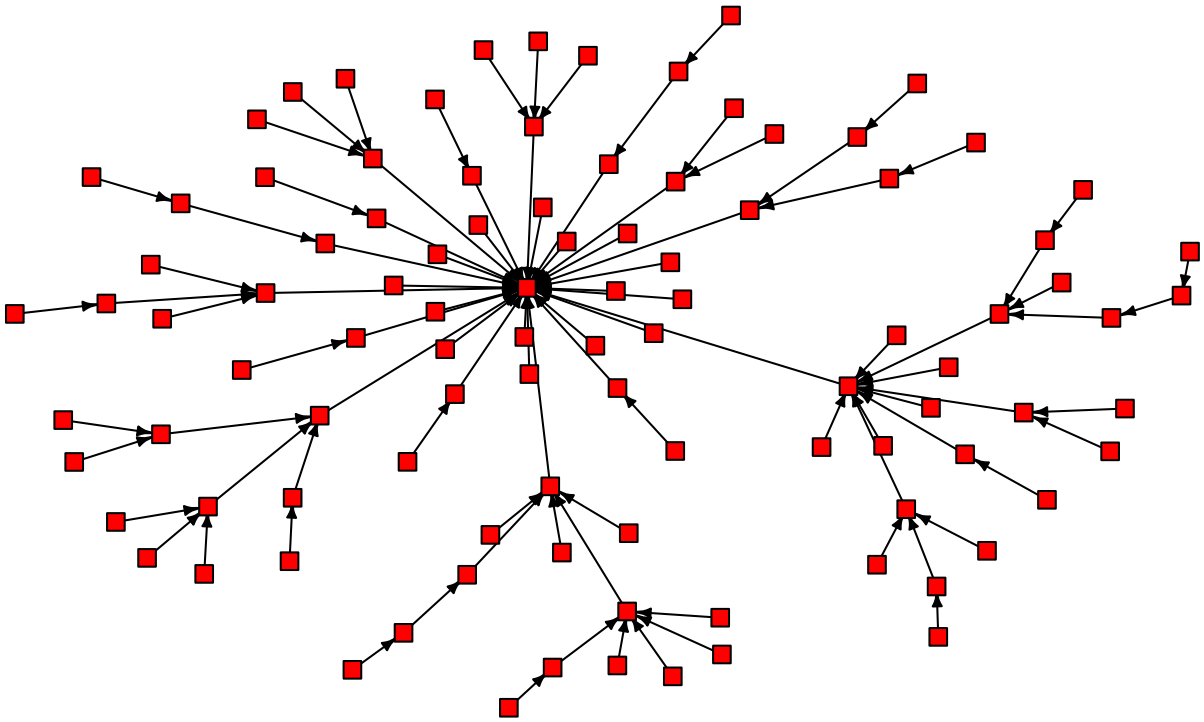
Creating graphs, random graphs

```
1 > er <- erdos.renyi.game(100, 100, type="gnm")
2 > plot(er, vertex.size=5, vertex.label=NA, asp=FALSE, vertex.shape="square",
3       layout=layout.fruchterman.reingold, edge.color="black")
```



Creating graphs, random graphs

```
1 > ba <- ba.game(100, power=1, m=1)
2 > plot(ba, vertex.size=3, vertex.label=NA, asp=FALSE, vertex.shape="square",
3       layout=layout.fruchterman.reingold, edge.color="black",
4       edge.arrow.size=0.5)
```



Meta data: graph/vertex/edge attributes

- Assigning attributes: `set/get.graph/vertex/edge.attribute`.

Meta data: graph/vertex/edge attributes

- Assigning attributes: `set/get.graph/vertex/edge.attribute`.
- `V(g)` and `E(g)`.

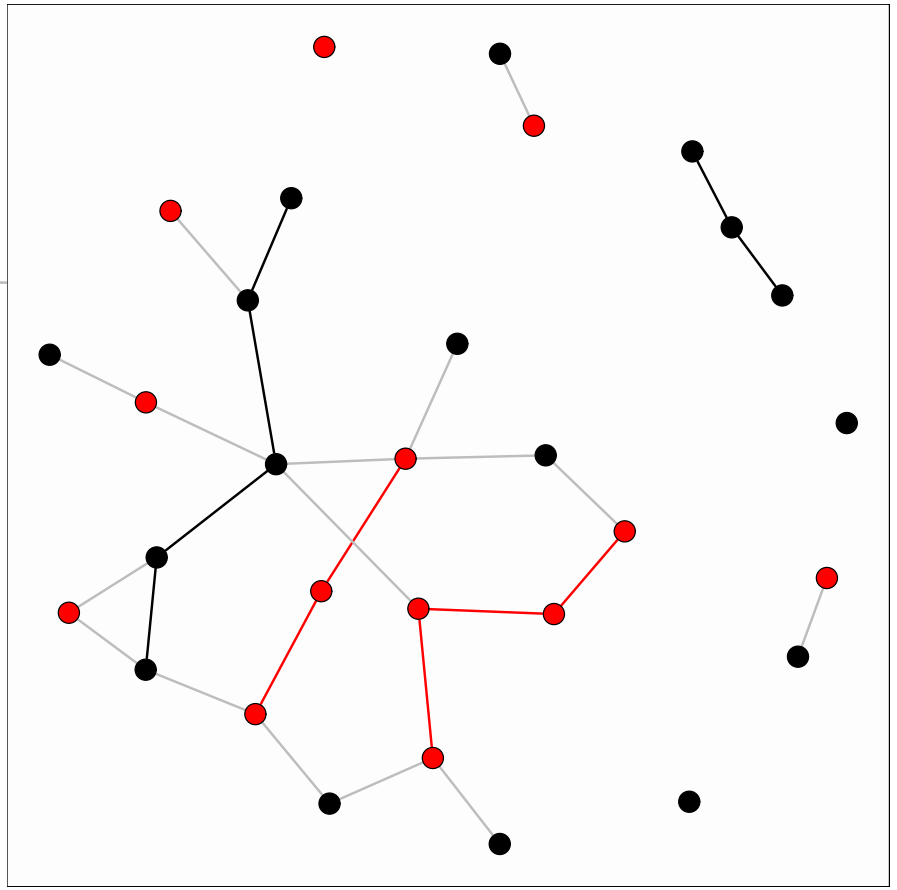
Meta data: graph/vertex/edge attributes

- Assigning attributes: `set/get.graph/vertex/edge.attribute`.
- `V(g)` and `E(g)`.
- Easy access of attributes:

```
> g <- erdos.renyi.game(30, 2/30)
> V(g)$color <- sample( c("red", "black"),
                        vcount(g), rep=TRUE)
> V(g)$color
 [1] "red"   "black" "red"   "black" "black" "black" "red"   "red"   "red"
[10] "black" "black" "black" "red"   "red"   "black" "red"   "black" "black"
[19] "red"   "red"   "black" "black" "red"   "black" "black" "red"   "black"
[28] "black" "black" "red"
> E(g)$color <- "grey"
```

Vertex/edge selection with attributes

```
1 > red <- V(g)[ color == "red" ]
2 > bl <- V(g)[ color == "black" ]
3 > E(g)[ red %--% red ]$color <- "red"
4 > E(g)[ bl  %--% bl ]$color <- "black"
5 > plot(g, vertex.size=5,
6        layout=layout.fruchterman.reingold,
7        vertex.label=NA)
```



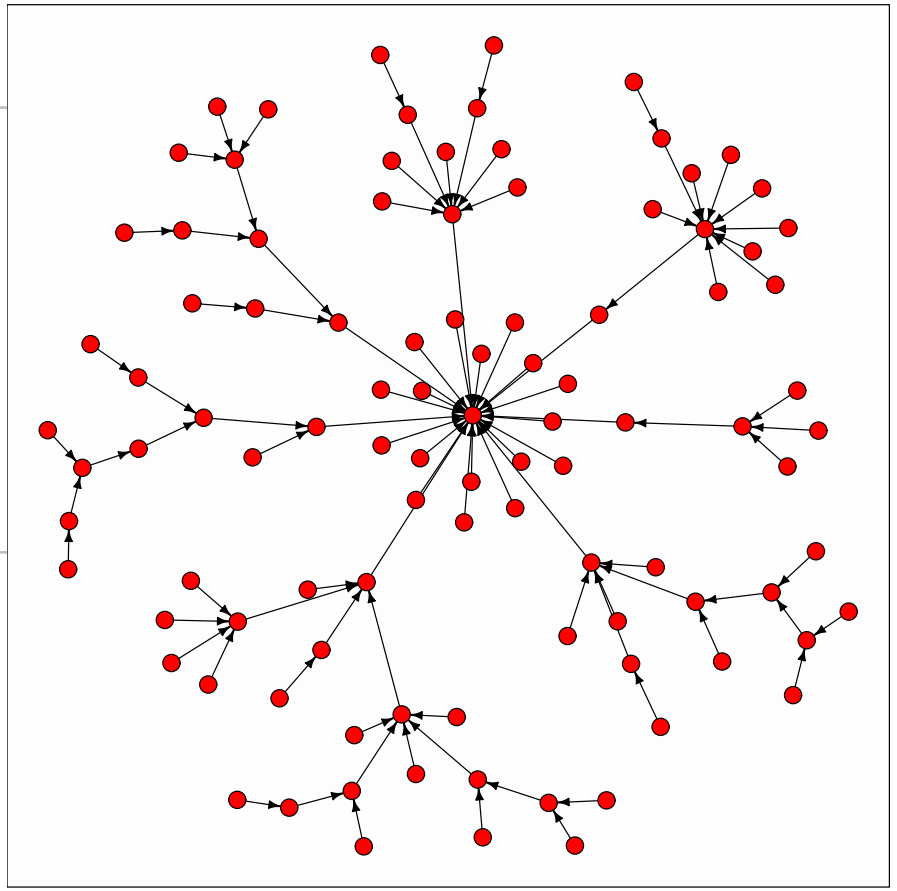
Visualizing graphs

- Three functions with (almost) identical interfaces.

Visualizing graphs

- Three functions with (almost) identical interfaces.
- `plot` Uses traditional R graphics, non-interactive, 2d. Publication quality plots in all formats R supports.

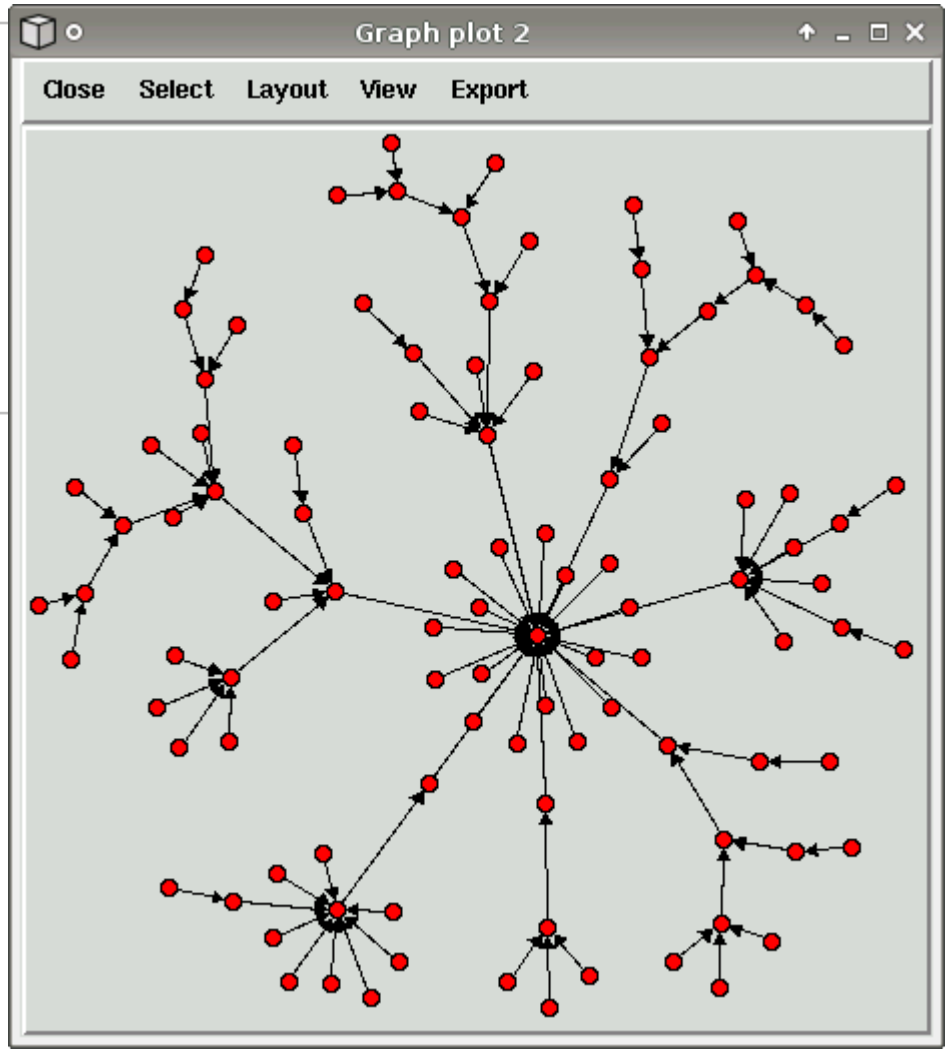
```
> g <- barabasi.game(100, m=1)
> igraph.par("plot.layout",
             layout.fruchterman.reingold)
> plot(g, vertex.size=4, vertex.label=NA,
       edge.arrow.size=0.7,
       edge.color="black",
       vertex.color="red", frame=TRUE)
```



Visualizing graphs

`tkplot` Uses Tcl/Tk via the `tcltk` package, interactive, 2d.

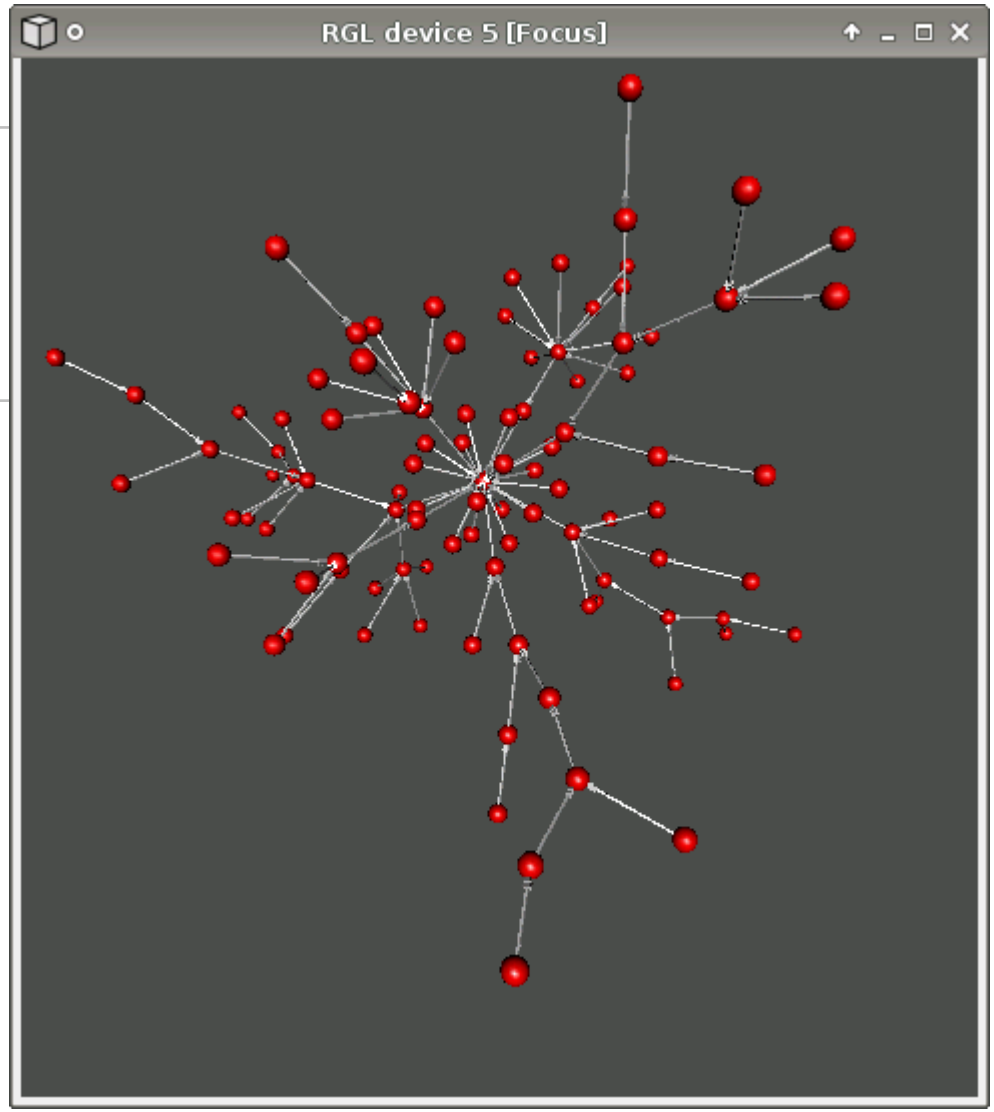
```
1 > id <- tkplot(g, vertex.size=4,
2          vertex.label=NA,
3          edge.color="black",
4          edge.arrow.size=0.7,
5          vertex.color="red")
6 > coords <- tkplot.getcoords(id)
```



Visualizing graphs

`rglplot` Needs the `rgl` package.

```
1 > co <- layout.kamada.kawai(g, dim=3)
2 > rglplot(g, vertex.size=5,
3             vertex.label=NA,
4             layout=co)
```



Working with a somewhat bigger graph

```
> vertices <- read.csv("http://cneurocvs.rmki.kfki.hu/igraph/judicial.csv")
> edges <- read.table("http://cneurocvs.rmki.kfki.hu/igraph/allcites.txt")
> jg <- graph.data.frame(edges, vertices=vertices, dir=TRUE)
> summary(jg)
 Vertices: 30288
 Edges: 216738
 Directed: TRUE
 No graph attributes.
 Vertex attributes: name, usid, parties, year, overruled, overruling,
   oxford, liihc, indeg, outdeg, hub, hubrank, auth, authrank, between, incent.
 No edge attributes.
```

Working with a somewhat bigger graph

```
> is.connected(jg)                    # Is it connected?
 [1] FALSE
```

Working with a somewhat bigger graph

```
> is.connected(jg)                    # Is it connected?
 [1] FALSE

> no.clusters(jg)                     # How many components?
 [1] 4881
```

Working with a somewhat bigger graph

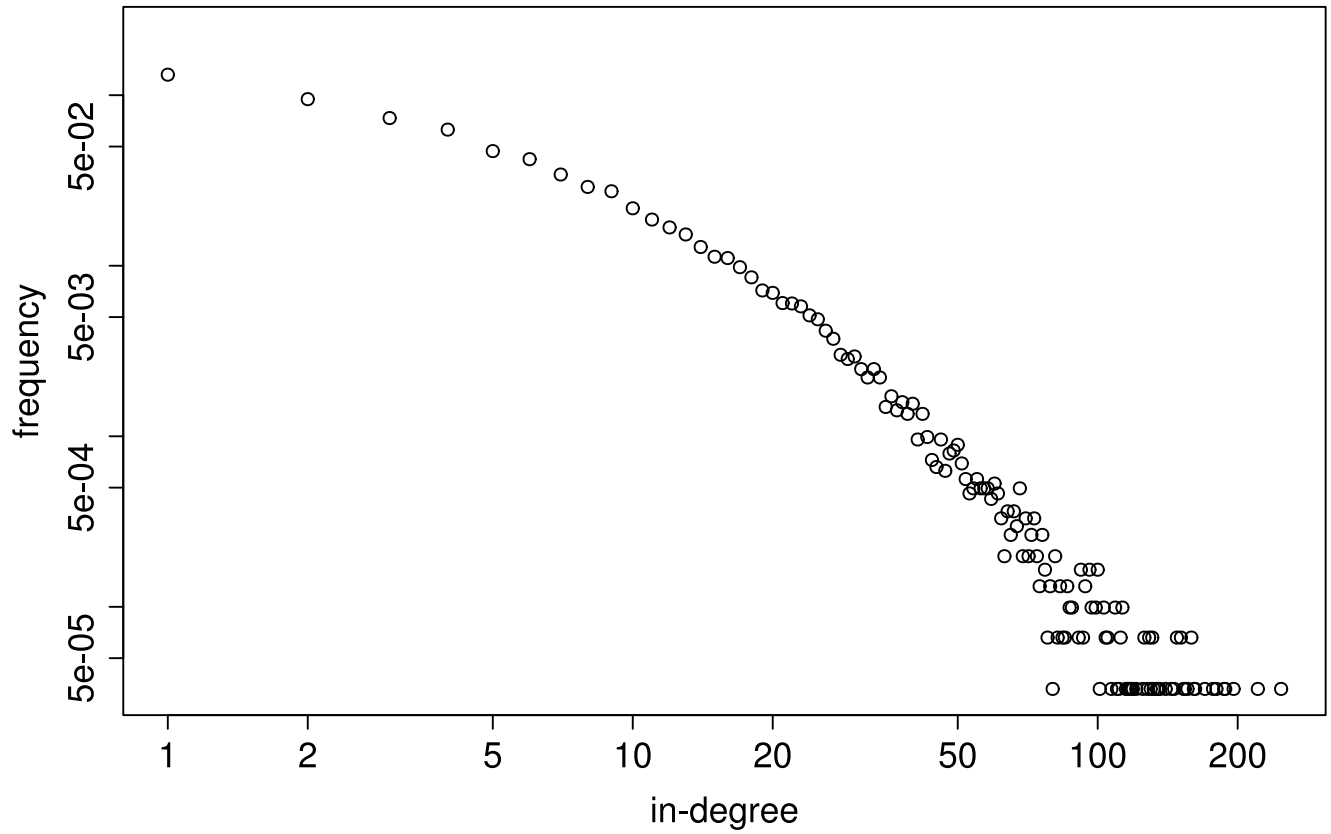
```
> is.connected(jg)                    # Is it connected?
 [1] FALSE

> no.clusters(jg)                     # How many components?
 [1] 4881

> table(clusters(jg)$csize)           # How big are these?

    1     3     4 25389
 4871     8     1     1
```

Working with a somewhat bigger graph

```
> is.connected(jg)                    # Is it connected?
 [1] FALSE

> no.clusters(jg)                     # How many components?
 [1] 4881

> table(clusters(jg)$csize)           # How big are these?

    1     3     4 25389
 4871     8     1     1

> max(degree(jg, mode="in"))          # Vertex degree
 [1] 248
> max(degree(jg, mode="out"))
 [1] 195
> max(degree(jg, mode="all"))
 [1] 313
```

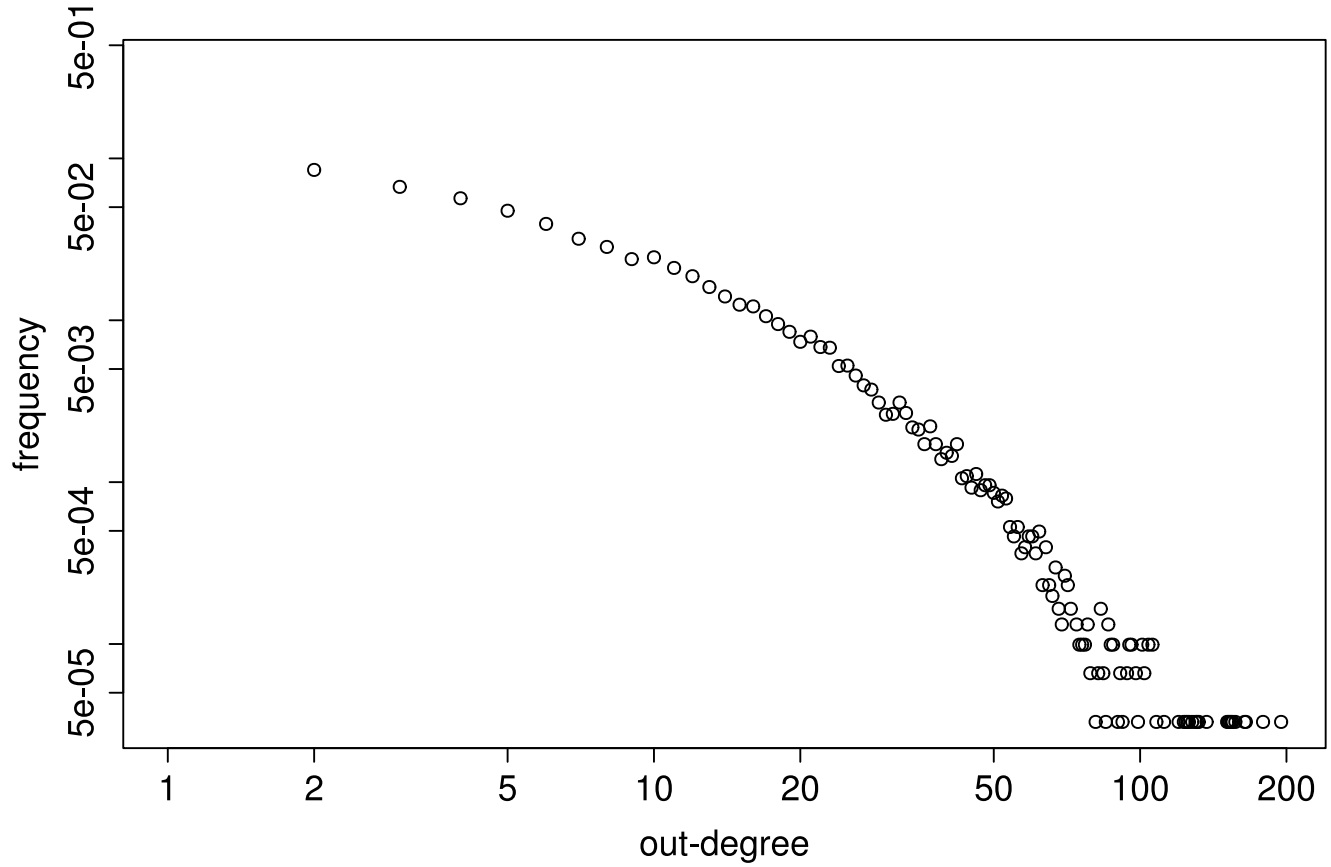
Working with a somewhat bigger graph

```
1 # In-degree distribution
2 > plot(degree.distribution(jg, mode="in"), log="xy")
```



Working with a somewhat bigger graph

```
1 # Out-degree distribution
2 plot(degree.distribution(jg, mode="out"), log="xy")
```



Working with a somewhat bigger graph

```
# Taking the largest component
> cl <- clusters(jg)
> jg2 <- subgraph(jg, which(cl$membership == which.max(cl$csize)-1)-1)
> summary(jg2)
 Vertices: 25389
 Edges: 216718
 Directed: TRUE
 No graph attributes.
 Vertex attributes: name, usid, parties, year, overruled, overruling,
    oxford, liihc, indeg, outdeg, hub, hubrank, auth, authrank,
    between, incent.
 No edge attributes.
```

Working with a somewhat bigger graph

```
1 > graph.density(jg2)                    # Density
2  [1] 0.0003362180
```

Working with a somewhat bigger graph

```
> graph.density(jg2)                    # Density
 [1] 0.0003362180

> transitivity(jg2)                     # Transitivity
 [1] 0.1260031
```

Working with a somewhat bigger graph

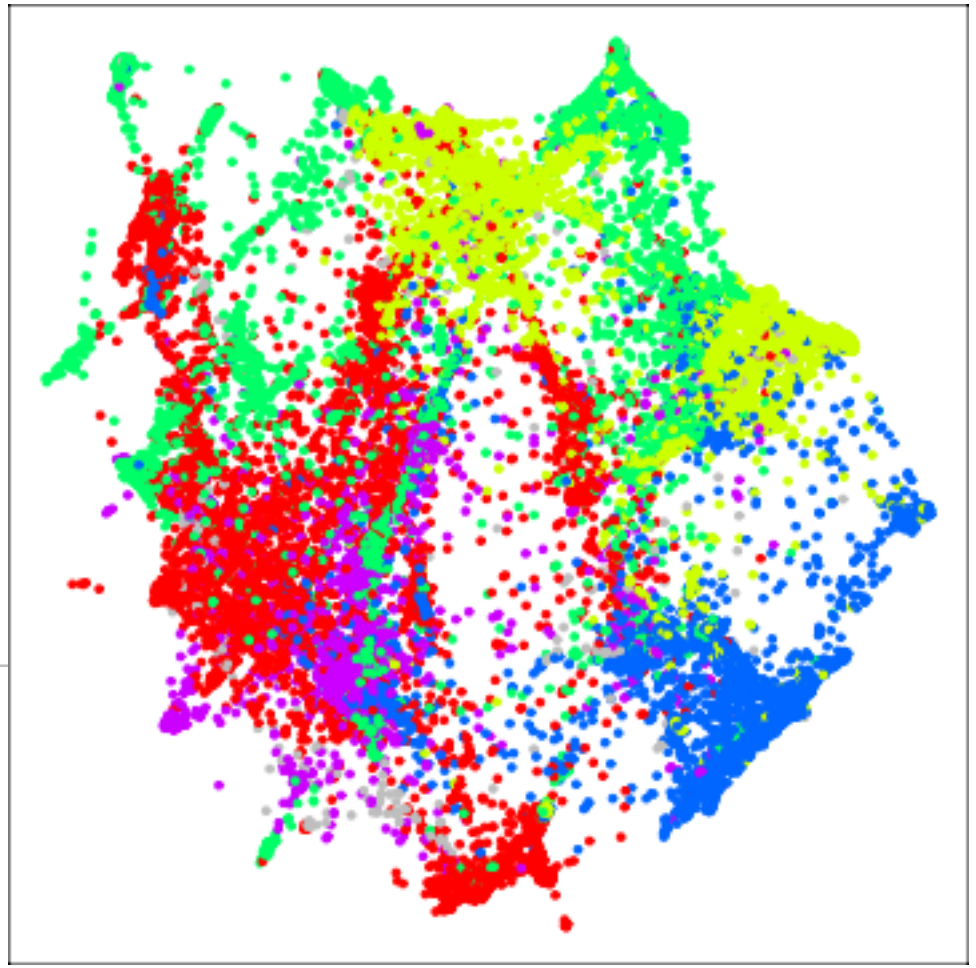
```
> graph.density(jg2)                                  # Density
 [1] 0.0003362180

> transitivity(jg2)                                   # Transitivity
 [1] 0.1260031

# Transitivity of a random graph of the same size
> g <- erdos.renyi.game(vcount(jg2), ecount(jg2), type="gnm")
> transitivity(g)
 [1] 0.00064649
```

Working with a somewhat bigger graph

```
> graph.density(jg2)                                  # Density
 [1] 0.0003362180

> transitivity(jg2)                                   # Transitivity
 [1] 0.1260031

# Transitivity of a random graph of the same size
> g <- erdos.renyi.game(vcount(jg2), ecount(jg2), type="gnm")
> transitivity(g)
 [1] 0.00064649

# Transitivity of a random graph with the same degrees
> g2 <- degree.sequence.game(degree(jg2,mode="all"), method="vl")
> transitivity(g2)
 [1] 0.004107072
```

Community structure detection

```
> fc <- fastgreedy.community(simplify(as.undirected(jg2)))
> memb <- community.to.membership(jg2,
          fc$merges,
          which.max(fc$modularity))
> lay <- layout.drl(jg2)
> jg3 <- graph.empty(n=vcount(jg2))
> colbar <- rainbow(5)
> col <- colbar[memb$membership+1]
> col[is.na(col)] <- "grey"
> plot(jg3, layout=lay, vertex.size=1,
       vertex.label=NA, asp=FALSE,
       vertex.color=col,
       vertex.frame.color=col)
```



Functionality, what can be calculated?

Fast (millions)	creating graphs (most of the time) • structural modification (add/delete edges/vertices) • subgraph • simplify • graph.decompose • degree • clusters • graph.density • is.simple, is.loop, is.multiple • articulation points and biconnected components • ARPACK stuff: page.rank, hub.score, authority.score, eigenvector centrality • transitivity • Burt's constraint • dyad & triad census, graph motifs • k-cores • MST • reciprocity • modularity • closeness and (edge) betweenness *estimation* • *shortest paths from one source* • *generating* $G_{n,p}$ *and* $G_{n,m}$ *graphs* • *generating PA graphs with various PA exponents* • *topological sort*
Slow (10000)	closeness • diameter • betweenness • all-pairs shortest paths, average path length • most layout generators •
Very slow (100)	cliques • cohesive blocks • edge/vertex connectivity • maximum flows and minimum cuts • power centrality • alpha centrality • (sub)graph isomorphism

Connection to other network/graph software

- `graph` package: `igraph.to.graphNEL`, `igraph.from.graphNEL`.

Connection to other network/graph software

- `graph` package: `igraph.to.graphNEL`, `igraph.from.graphNEL`.
- Sparse matrices (`Matrix` package), `get.adjacency` and `graph.adjacency` supports them.

Connection to other network/graph software

- `graph` package: `igraph.to.graphNEL`, `igraph.from.graphNEL`.
- Sparse matrices (`Matrix` package), `get.adjacency` and `graph.adjacency` supports them.
- `sna` and `network` R packages. Currently throught adjacency matrices. Use namespaces!

Connection to other network/graph software

- `graph` package: `igraph.to.graphNEL`, `igraph.from.graphNEL`.
- Sparse matrices (`Matrix` package), `get.adjacency` and `graph.adjacency` supports them.
- `sna` and `network` R packages. Currently throught adjacency matrices. Use namespaces!
- Pajek. `.net` file format is supported.

Connection to other network/graph software

- `graph` package: `igraph.to.graphNEL`, `igraph.from.graphNEL`.
- Sparse matrices (`Matrix` package), `get.adjacency` and `graph.adjacency` supports them.
- `sna` and `network` R packages. Currently throught adjacency matrices. Use namespaces!
- Pajek. `.net` file format is supported.
- Visone. Use GraphML format.

Connection to other network/graph software

- `graph` package: `igraph.to.graphNEL`, `igraph.from.graphNEL`.
- Sparse matrices (`Matrix` package), `get.adjacency` and `graph.adjacency` supports them.
- `sna` and `network` R packages. Currently throught adjacency matrices. Use namespaces!
- Pajek. `.net` file format is supported.
- Visone. Use GraphML format.
- Cytoscape. Use GML format.

Connection to other network/graph software

- `graph` package: `igraph.to.graphNEL`, `igraph.from.graphNEL`.
- Sparse matrices (`Matrix` package), `get.adjacency` and `graph.adjacency` supports them.
- `sna` and `network` R packages. Currently throught adjacency matrices. Use namespaces!
- Pajek. `.net` file format is supported.
- Visone. Use GraphML format.
- Cytoscape. Use GML format.
- GraphViz. igraph can write `.dot` files.

Connection to other network/graph software

- `graph` package: `igraph.to.graphNEL`, `igraph.from.graphNEL`.
- Sparse matrices (`Matrix` package), `get.adjacency` and `graph.adjacency` supports them.
- `sna` and `network` R packages. Currently throught adjacency matrices. Use namespaces!
- Pajek. `.net` file format is supported.
- Visone. Use GraphML format.
- Cytoscape. Use GML format.
- GraphViz. igraph can write `.dot` files.
- In general. The `GraphML` and `GML` file formats are fully supported, many programs can read/write these.

Acknowledgements

Tamás Nepusz

Peter McMahan, the BLISS, Walktrap, Spinglass, DrL projects

All the people who contributed code, sent bug reports, suggestions

The R project