Graphs in R

- Graph
- igraph-package
- GraphNEL
- KEGGgraph

Graph Package

- Is the base network manipulation library of R
- Later igraph library was made.
- Graph is still being developed ,for example graphNEL was added to it as a class later.
- igraph has most the common classes of graph in parallel
- For example:
  - In graph there is **addEdge(from, to, graph, weights)**
  - In igraph there is **add.edges(graph, edges, ..., attr=list())**

    to add edges to existing graphs

Graph library examples

- f <- c("a", "a", "b", "c", "d")
- t <- c("b", "c", "c", "d", "a")
- weight <- rep(1,length(f))
- df <- data.frame(from=f, to=t, weight= weight)
- g <- graphBAM(df)
- gd <- graphBAM(df, edgemode = "directed")
- nd <- nodes(g)
- w1 <- edgeWeights(g)
- w2 <- edgeWeights(g,"a")
- w3 <- edgeWeights(g,1)
- d1 <- edges(g)
- d2 <- edges(g,c("a", "b"))
- e1 <- edgeData(g)
- e2 <- edgeData(g, "a", "c",attr="weight")
- em <- edgeMatrix(g)
- id <- isDirected(g)
- sg <- subGraph(c("a","c","d"), g)
- ft <- extractFromTo(g)
- am <- as(g,"graphAM")
- nl <- as(g,"graphNEL")
- mt <- as(g,"matrix")
- k <- intersection(g,g)
- k <- union(g,g)
- g
- gd

igraph

- igraph is a library for network analysis.
- The main goals of the igraph library is to provide a set of data types and functions for:
  - Pain-free implementation of graph algorithms
  - fast handling of large graphs, with millions of vertices and edges
  - allowing rapid prototyping via high level languages like R

Creating graphs in igraph

- There are many functions in igraph for creating graphs, both deterministic and stochastic
- Stochastic graph constructors are called ‘games’ is igraph
- To create small graphs with a given structure the `graph.formula` function is easiest
- To create graphs from field data, `graph.edgelist`, `graph.data.frame` and `graph.adjacency` are probably the best choices.

Example

- g=graph.formula(A-B)
- Or g=graph.formula(A–----B)
- In directed case:
  - g2 =graph.formula(A -+ B -+ C)
  - g1 = graph.formula(b-+a , a-+c)

Enges and vertex IDs in igraph

- Vertices and edges have numerical vertex ids in igraph.
- Vertex ids are always consecutive and they start with zero. I.e. for a graph with ‘n’ vertices the vertex ids are between ‘0’ and ‘n-1’.
- If some operation changes the number of vertices in the graphs, e.g. a subgraph is created via `subgraph`,
- then the vertices are renumbered to satisfty this criteria.
- The same is true for the edges as well, edge ids are always between zero and ‘m-1’ where ‘m’ is the total number of edges in the graph.

Attributes

- In igraph it is possible to assign attributes to the vertices or edges of a graph, or to the graph itself.
- igraph provides flexible constructs for selecting a set of vertices or edges based on their attribute values.
- see get.vertex.attribute and iterators for details

Attribiutes continue

- Some vertex/edge/graph attributes are treated specially. One of them is the ‘name’ attribute.
- This is used for printing the graph instead of the numerical ids, if it exists. Vertex names can also be used
- to specify a vector or set of vertices, in all igraph functions E.g. **degree** has a ‘v’ argument that gives the vertices for which the degree is calculated. This argument can be given as character vector of vertex names as well
- Attribute values can be set to any R object, but note that storing the graph in some file formats might result the loss of complex attribute values. All attribute values are preserved if you use **save** and **load** to store/retrieve your graphs.

Visualization in igraph

- igraph provides three different ways for visualization. The first is the ‘plot.igraph’ function.
- (Actually you don’t need to write ‘plot.igraph’, ‘plot’ is enough. This function uses base R graphic and can be used with any R device.
- The second function is tkplot, which uses a Tk GUI for basic interactive graph manipulation.
- (Tk is quite resource hungry, so don’t try this for very large graphs.)
- The third way requires the rgl package and uses OpenGL. See the rglplot function for the details

example

- pdf("16sep.pdf")
- plot(g3, layout=layout.kamada.kawai, vertex.color="green")
- library(plotrix)
- dev.off()

File formats in igraphs

- igraph can handle various graph file formats, usually both for reading and writing. We suggest that
- you use the GraphML file format for your graphs, except if the graphs are too big. For big graphs a
- simpler format is recommended. See read.graph and write.graph for details

graphNEL

- It is a class that extendes graph package
- This is a class of graphs that are represented in terms of nodes and an edge list.
- This is a suitable representation for a graph with a large number of nodes and relatively few edges

graphNEL

- The edgeL is a named list of the same length as the node vector. The names are the names of the nodes.
- Each element of edgeL is itself a list. Each element of this (sub)list is a vector (all must be the same length)
- and each element represents an edge to another node.
- The sublist named edges holds index values into the node vector.
- And each such entry represents an edge from the node which has the same name as the component of edgeL to the node with index provided.

graphNEL

- Another component that is often used is named weights. It represents edge weights.
- The user can specify any other edge attributes (such as types etc). They are responsible for any special handling that these might require.
- For an undirected instance all edges are reciprocated (there is an edge from A to B and from B to A).
-

Conversion from graphNEL to igraph and back

- igraph.from.graphNEL(graphNEL, name = TRUE, weight = TRUE, unlist.attrs = TRUE)

- igraph.to.graphNEL(graph)

Example:

- GNEL1 <- igraph.to.graphNEL(g1)
- GNEL2 <- igraph.to.graphNEL(g2)

Mergin to graphnels:

- G = mergeGraphs(list(GNEL1, GNEL2))

KEGGgraph package

- Is an interface between KEGG pathway and graph object as well as a collection of tools to analyze, dissect and visualize these graphs
- KEGGgraph maintains the pathway topology and allows further analysis or dissection of pathway graphs

KEGGgraph Package cont

- This packege is used in corporation to graph objects usually those in GraphNel class
- The package requires KGML (KEGG XML) files, which can be downloaded from KEGG
- FTP site (ftp://ftp.genome.jp/pub/kegg/xml) without license permission for academic
- reference: http://watson.nci.nih.gov/bioc_mirror/packages/2.4/bioc/vignettes/KEGGgraph/inst/doc/KEGGgraph.pdf

KEGGgraph

- With this package we can :
- Read KGML file
  - > mapkKGML <- system.file("extdata/hsa04010.xml",
  - +
  - package="KEGGgraph")
- Parse it : this will convert the pathway into graph
  - > mapkG <- parseKGML2Graph(mapkKGML,expandGenes=TRUE)

KEGGgraph and GraphNEL in graph

We can merg them with graphNEL :

- graphs <- list(mapk=mapkG, wnt=wntG)

  merged <- mergeGraphs(graphs)

  merged

  A graphNEL graph with directed edges

  Number of Nodes = 386

  Number of Edges = 1628