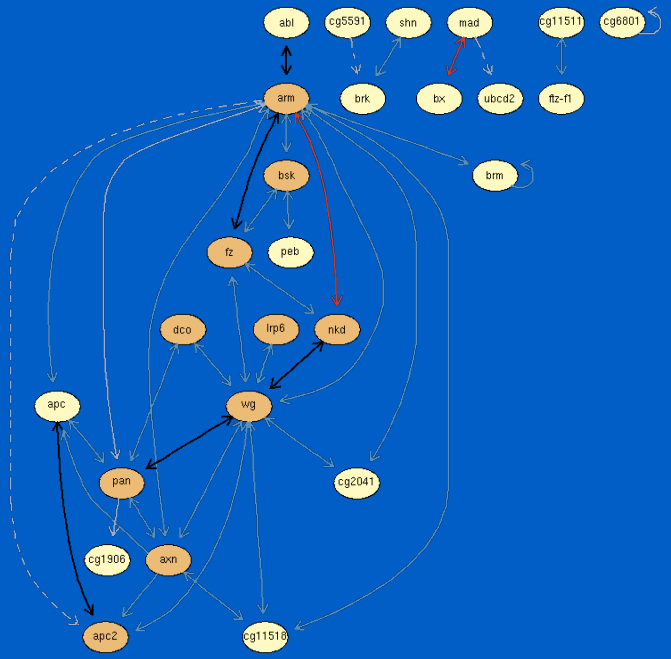
4/29/09

Explore biological graphs and networks using graph and Rgraphviz



Florian Hahne
Fred Hutchinson Cancer Research Center

FRED HUTCHINSON CANCER RESEARCH CENTER

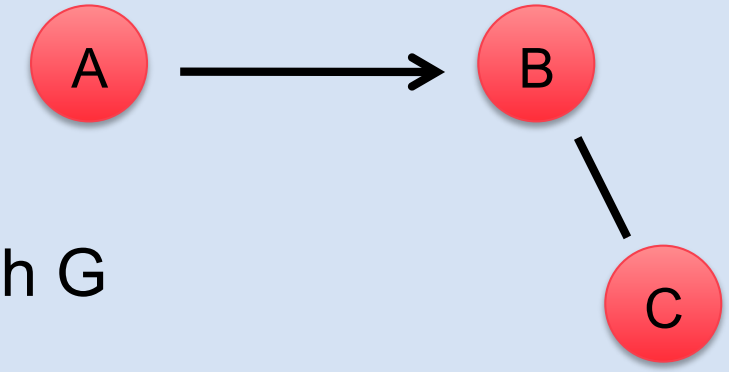
A LIFE OF SCIENCE

Overview

- Introduction to graphs
- Graphs in biology
- Bioconductor software for graphs
- How to create graphs
- Plotting graphs using the Rgraphviz package

Graphs: definition

• a graph is a collection of vertices (V) and edges (E) between the vertices



• G=(V,E) to denotes the graph G

• nodes represent entities

• edges represent relationships
  - binary or continuous (edge weights)
  - edge types
  - directed or undirected

Useful abstraction to talk about relationships/interactions

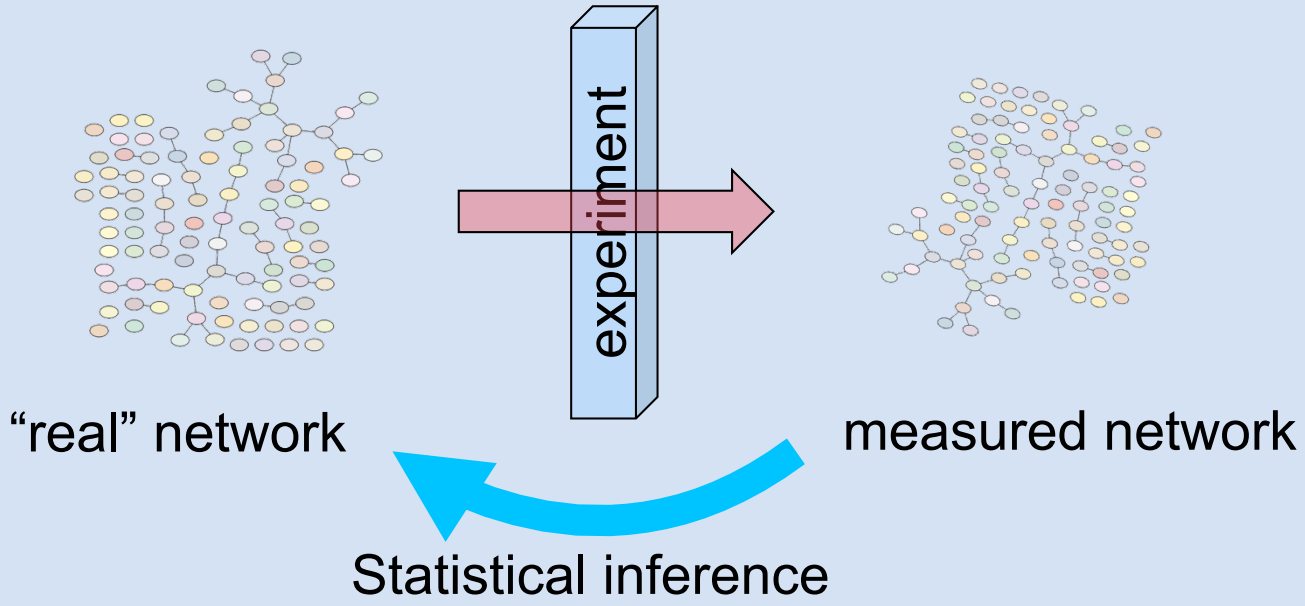
Graphs: paradigms

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE

- **social sciences:** social network analysis
- **communications industry:** telephone networks, computer networks
- **marketing:** relationships between people and the magazines they read, TV they watch, items they buy
- **biology:** pathways, co-citation, Gene Ontology, transcription factor, protein interactions

Graphs: applications

- **knowledge representation:** pathways, GO
- **exploratory data analysis:** mapping of gene expression data to a pathway graph
- **statistical inference:** comparing experimental measurements vs. true state of nature, random graphs, graph permutations



Uncertainty in biological graphs

- to date, the study of graphs has been primarily a mathematical study

- distinguish between the true, underlying property that you want to measure and the actual result of the measurement:

    1. False positive edges
    2. False negative edges
    3. Untested

Uncertainty is not usually considered in mainstream graph theory, but cannot be ignored in bioinformatics applications.

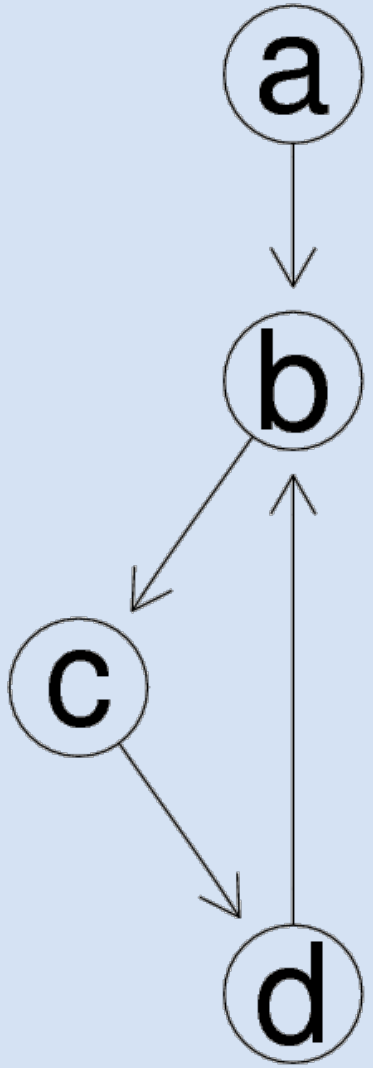
Graphs representations

- node and edge list
- from-to matrix
- adjacency matrix

```
> class(g)
[1] "graphNEL"

> nodes(g)
[1] "a" "b" "c" "d"

> edges(g)
$a
[1] "b"

$b
[1] "c"

$c
[1] "d"

$d
[1] "b"
```

```
> edgeMatrix(g)
  from to
[1,] "a" "b"
[2,] "b" "c"
[3,] "c" "d"
[4,] "d" "b"
```

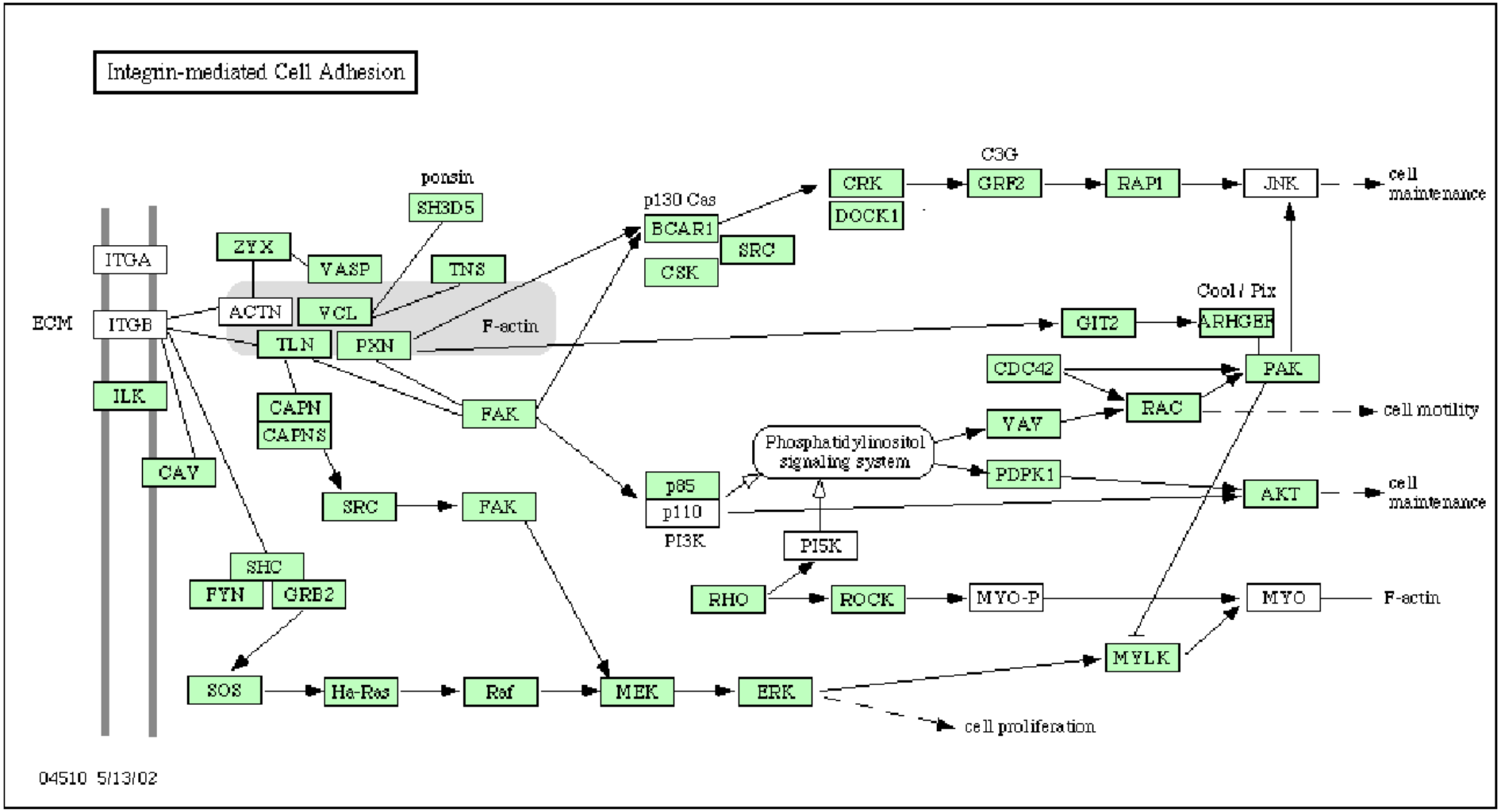
```
  a b c d
a 0 1 0 0
b 0 0 1 0
c 0 0 0 1
d 0 1 0 0
```



- **Performance and convenience considerations**
- **Coercion between representations**

Graph types: pathways

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE



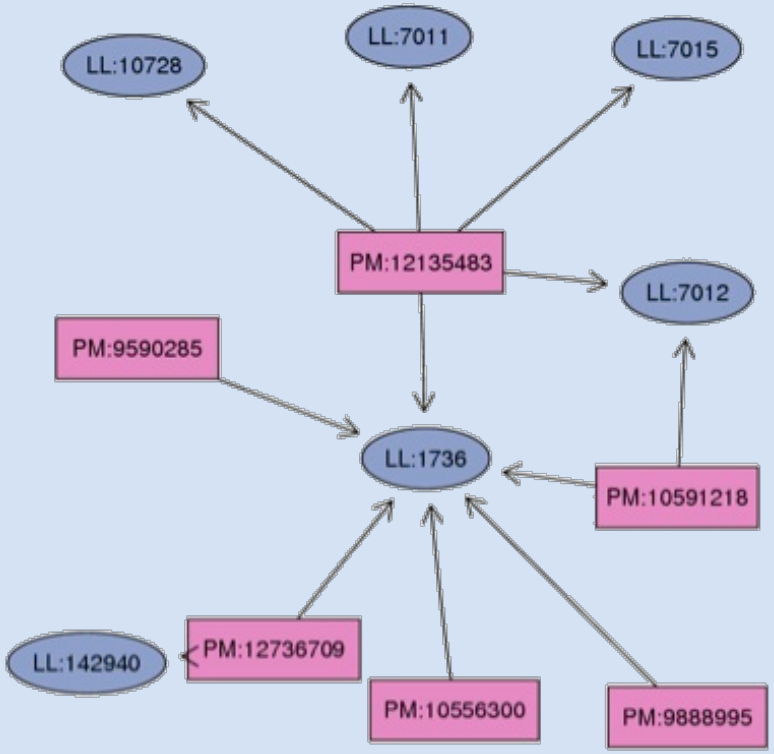
Graphs types: multigraph

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE

- in some situations we have a single set of nodes
  - genes in an organism
  - people of interest
  - airports
- and multiple relationships between them
  - co-regulated by transcription factors
  - flight connections
- these can be represented as multigraphs

Graphs types: bipartite graphs

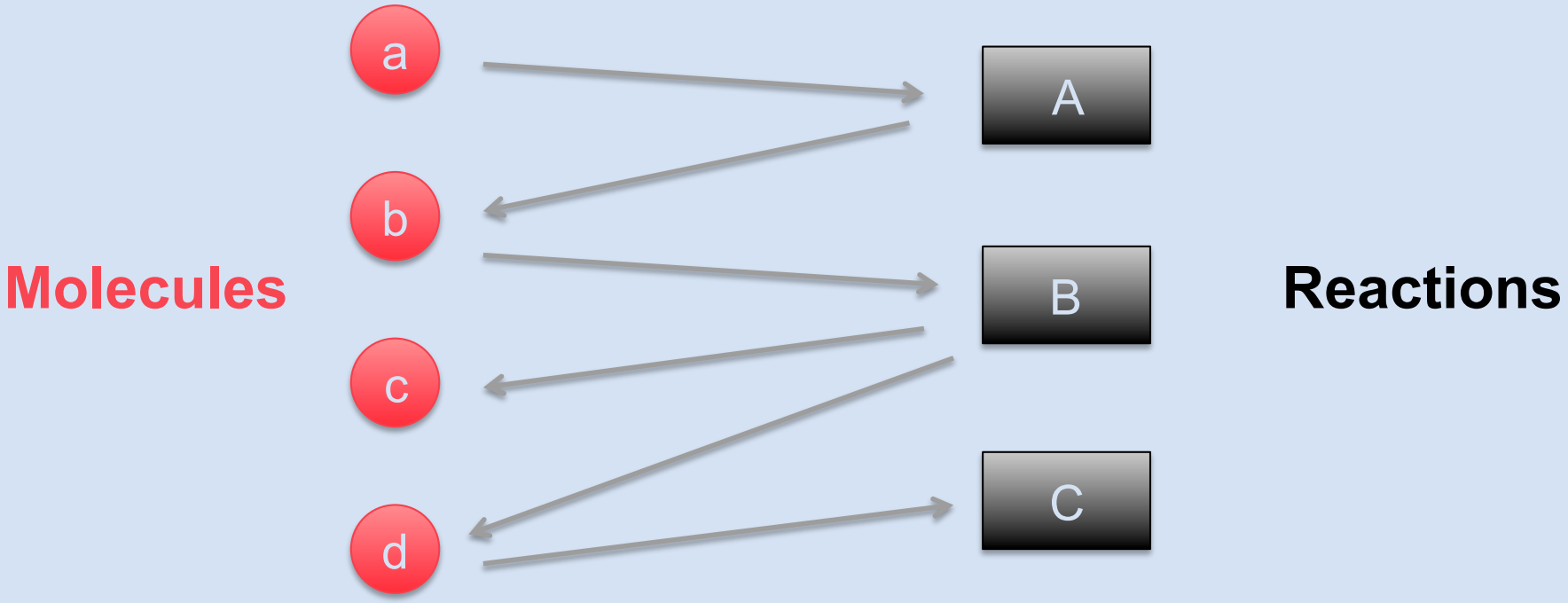
- A bipartite graph is a graph where the set of graph vertices can be decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent.
  - genes to papers
  - proteins to protein complexes
  - proteins/genes to pathways



Graphs types: directed bipartite graphs

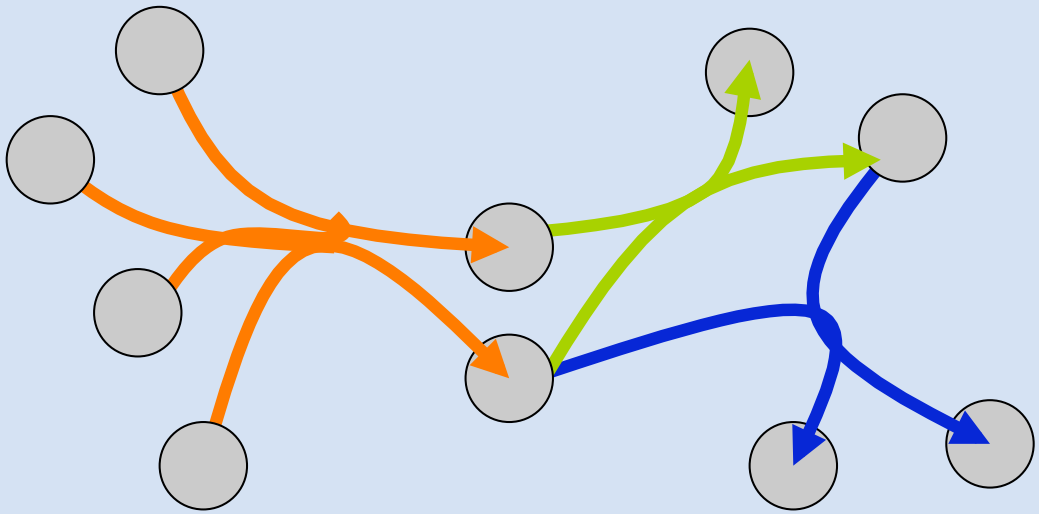
FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE

- one can also have directed edges in a bipartite graph
- such a graph may be very useful for representing chemical reactions, or metabolic reactions
- it can represent sequential aspects of a set of relationships



Graphs types: hypergraphs

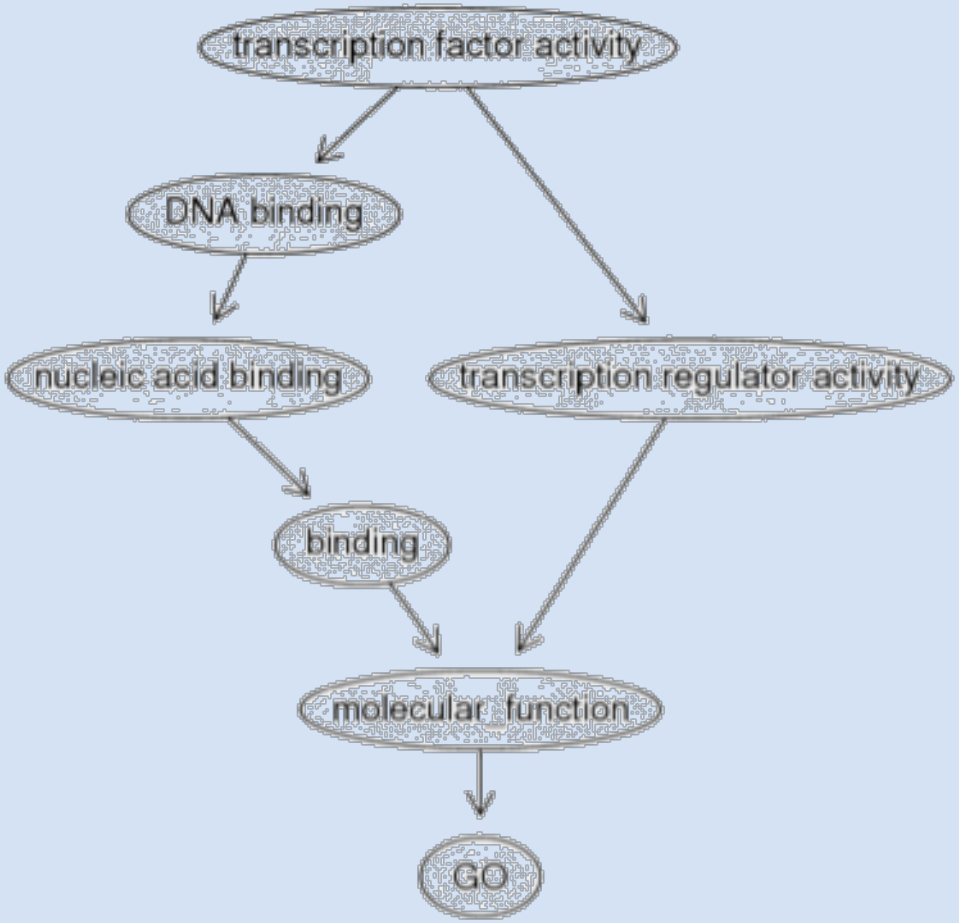
- sometime we want to represent many to many relationships
- this can be handled by considering hypergraphs
- set of nodes and set of hyperedges (which again is a set of nodes)
- e.g., protein complex interactions



Graphs types: directed acyclic graph (DAG)

- Useful for presenting hierarchies and partial orderings (e.g., in time, from general to special, from cause to effect)

GeneOntology:



Graph Software in BioConductor

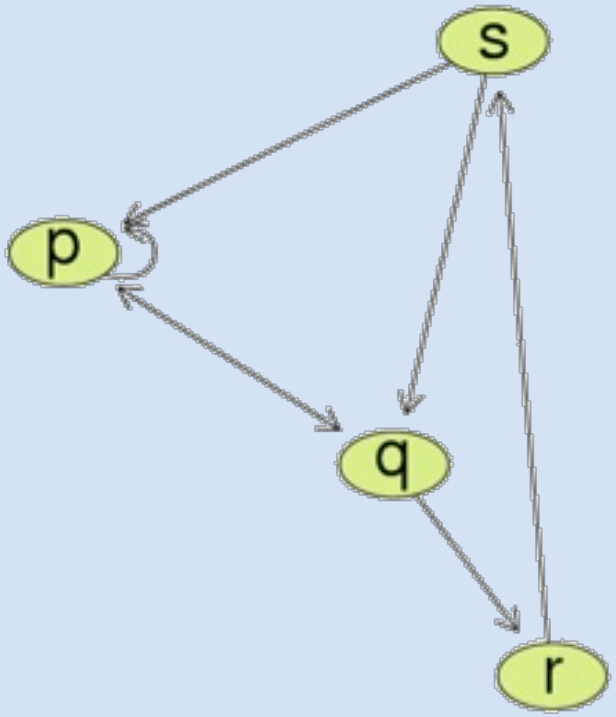
- graph: basic class definitions, coercion, basic operations (union, subgraph, etc.)
- RBGL: an interface to the BOOST graph library of algorithms (L. Long, ETH Lusanne)
- Rgraphviz: an interface to Graphviz for graph layout algorithms
- Many packages using this infrastrucutre

The graph package

- graph classes:
  - graph; clusterGraph, distGraph, (hyperGraph)
- operations:
  - nodes; edges; subgraph
  - random graph generation
  - serialization; GXL, tulip etc
- representations:
  - node and edgeList
  - adjacency matrix (sparse matrix)
  - node sets/edge sets
- generation of random graphs
  - various algorithms

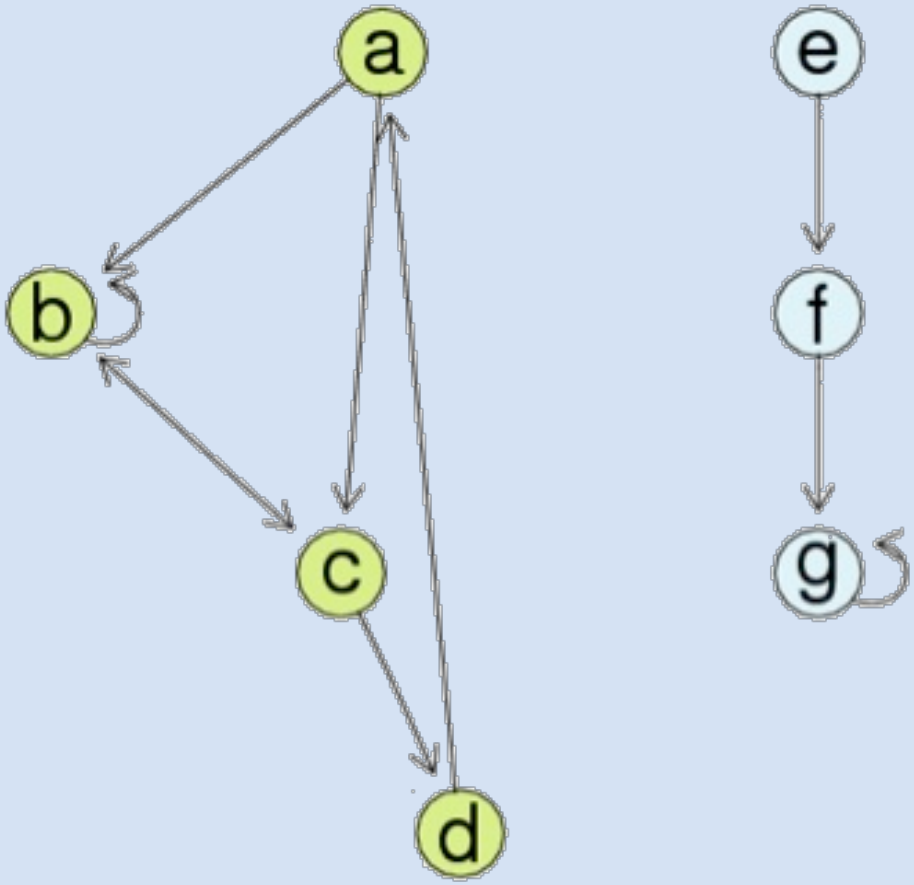
Interacting with graphs


```
> nodes(g)
[1] "s" "p" "q" "r"

> edges(g)
$s
[1] "p" "q"
$p
[1] "p" "q"
$q
[1] "p" "r"
$r
[1] "s"

> degree(g)
$inDegree
s p q r
1 3 2 1
$outDegree
s p q r
2 2 2 1
```



Interacting with graphs


```
> adj(g, c("b", "c"))
$b
[1] "b" "c"
$c
[1] "b" "d"

> acc(g, c("b", "c"))
$b
a c d
3 1 2

$c
a b d
2 1 1
```



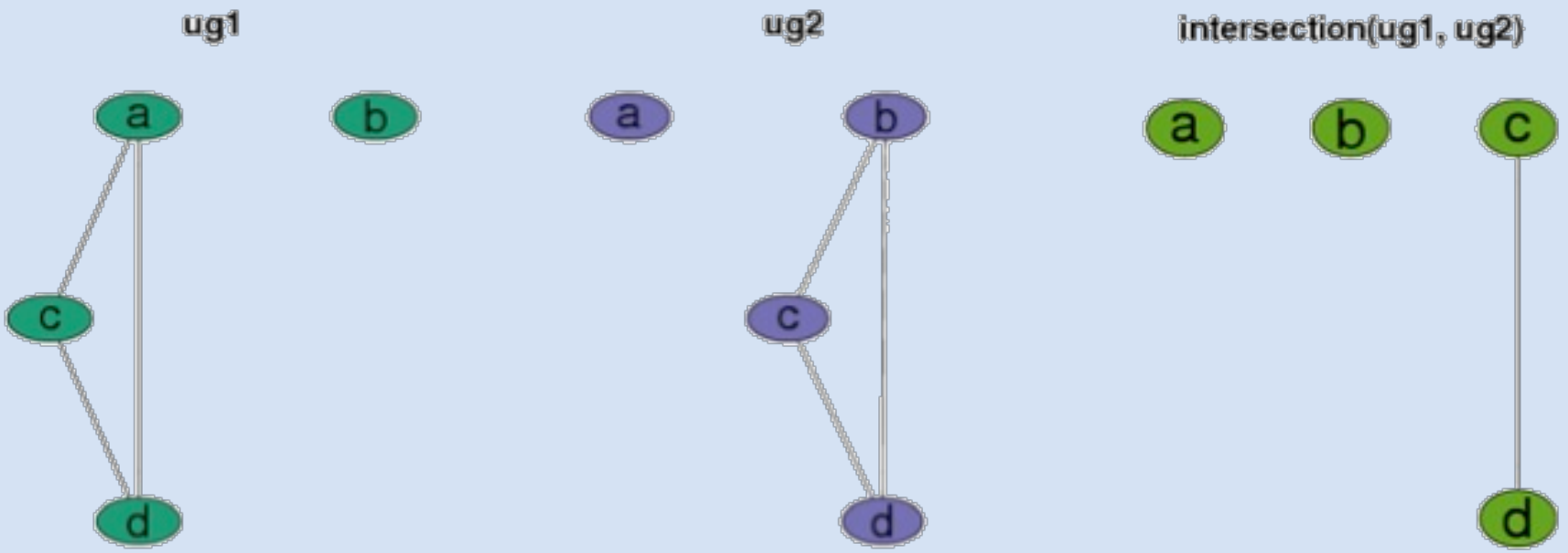
Graph manipulation


```
> g1 <- addNode("e", g)

> g2 <- removeNode("d", g)

> ## addEdge(from, to, graph, weights)
> g3 <- addEdge("e", "a", g1, pi/2)

> ## removeEdge(from, to, graph)
> g4 <- removeEdge("e", "a", g3)

> identical(g4, g1)
[1] TRUE
```

Intersection

- for any two graphs,
  - $G_1=(V, E_1)$ and
  - $G_2=(V, E_2)$

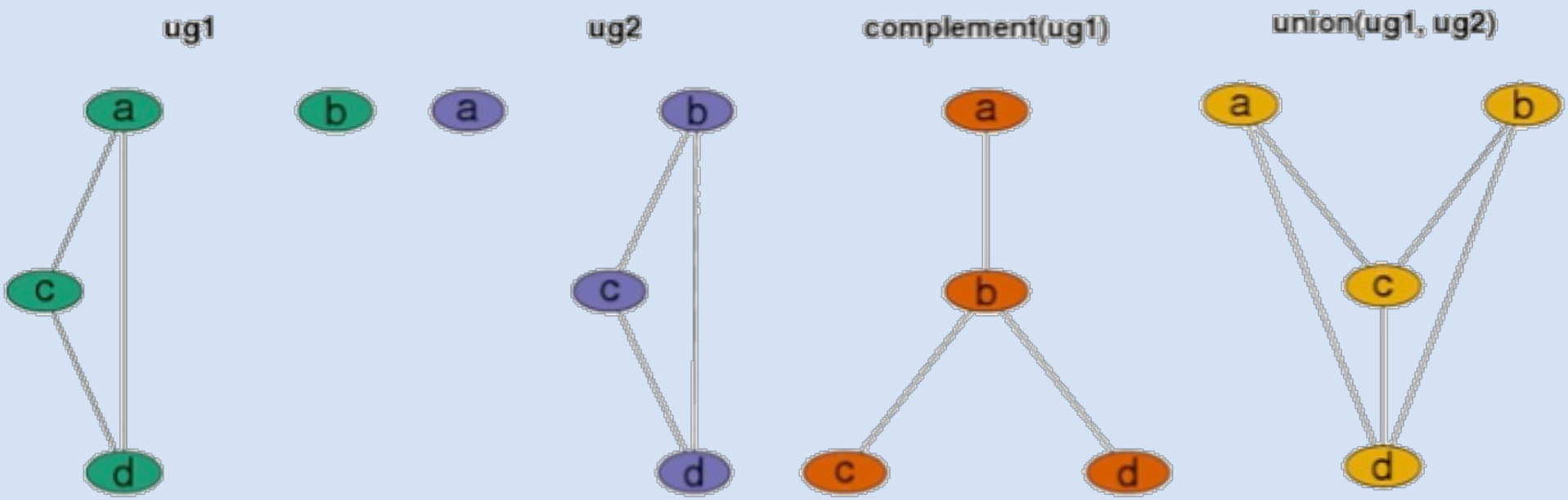
  defined on the same set of nodes (or vertices)
- define the ***intersection*** of G_1 and G_2 to be the graph, G = (V, E), where e is in E if and only if e is in E_1 and in E_2



Complement and Union

- for any graph G=(V,E), define the complement of the graph to be those edges in the complete graph defined on V that are not in E

- for any two graphs $G_1=(V,E_1)$ and $G_2=(V, E_2)$, defined on the same set of nodes, define their union to be G=(V, E), where e is in E if e is in either E_1 or E_2.



The RBGL package

- based on the BOOST graph library

- algorithms include:
  - shortest path (Dijkstra, Bellman-Ford etc.)
  - DFS and BFS
  - max-flow/min-cut algorithms
  - orderings
  - many more can be added

The RBGL package: connected components

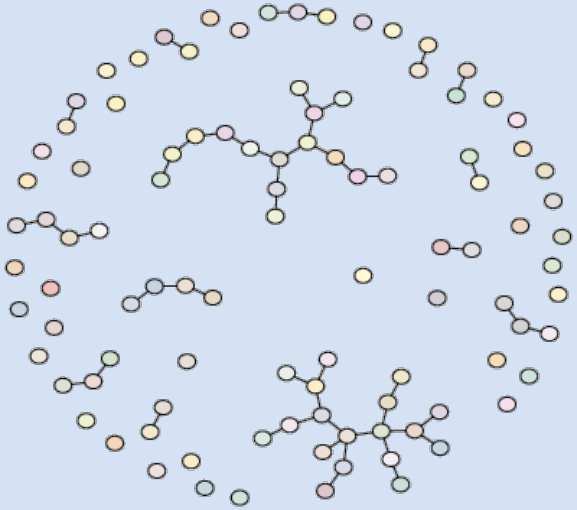
Connected components

```
cc = connComp(rg)
table(listLen(cc))
 1   2   3   4  15  18
36   7   3   2   1   1
```

Choose the largest component

```
wh = which.max(listLen(cc))
sg = subGraph(cc[[wh]], rg)
```

rg

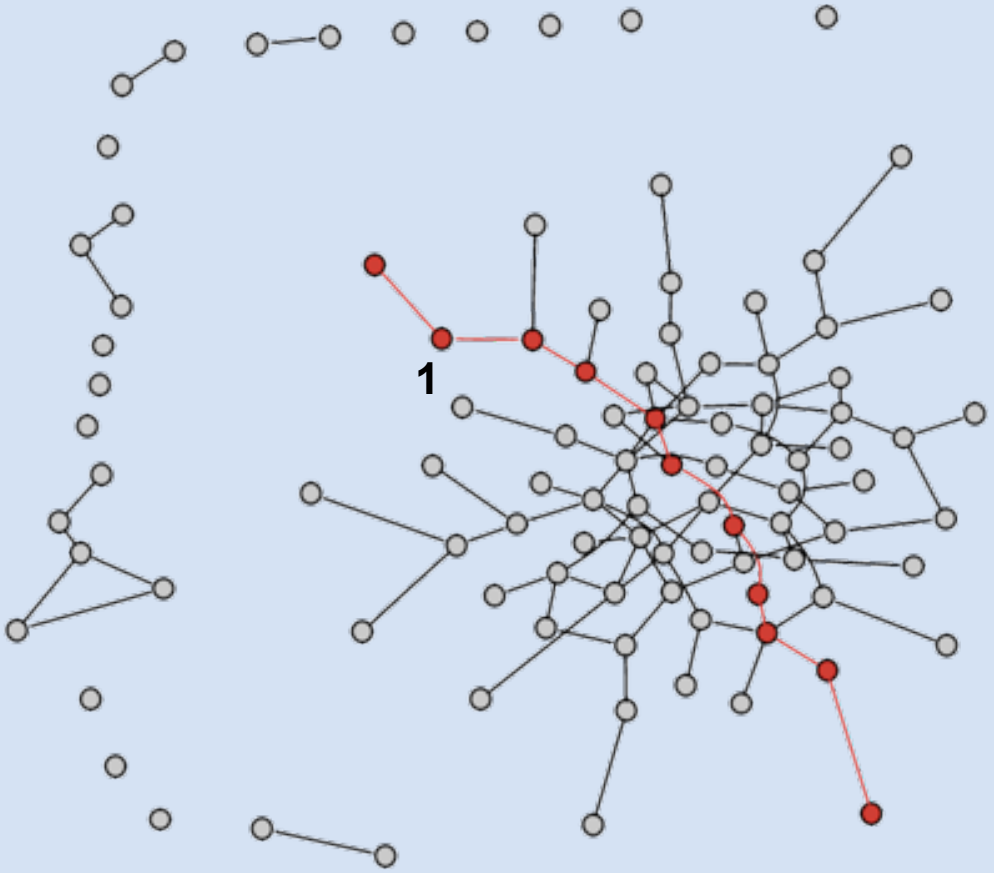


Depth first search

```
dfsres = dfs(sg, node = "N14")
nodes(sg)[dfsres$discovered]
[1] "N14" "N94" "N40" "N69" "N02" "N67" "N45" "N53" [9] "N28"
"N46
"
 "N51" "N64" "N07" "N19" "N37" "N35" [17] "N48" "N09"
```

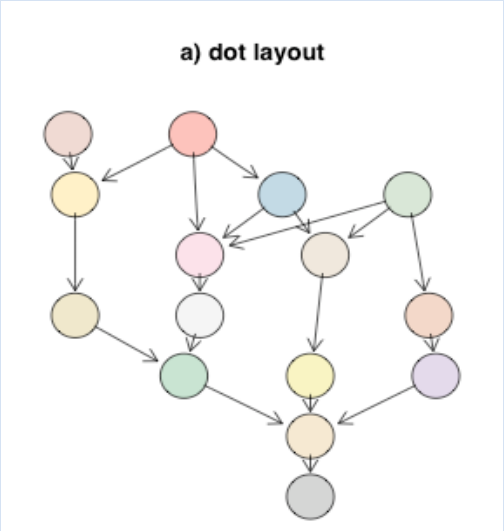
The RBGL package: shortest paths

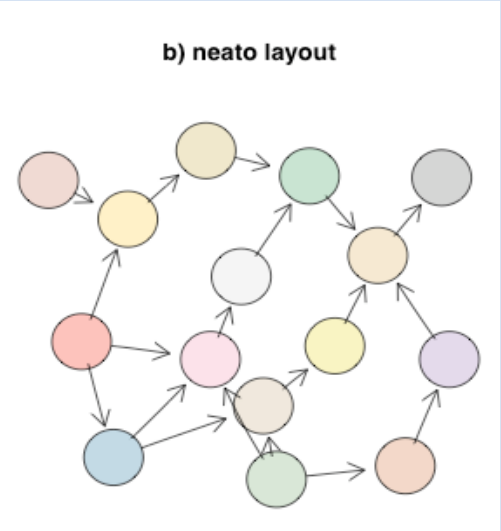

```
set.seed(123)
rg2 = randomEGraph(nodeNames, edges = 100)
fromNode = "N43"
toNode = "N81"
sp = sp.between(rg2,
    fromNode, toNode)

sp[[1]]$path
[1] "N43" "N08" "N88"
[4] "N73" "N50" "N89"
[7] "N64" "N93" "N32"
[10] "N12" "N81"

sp[[1]]$length
[1] 10
```

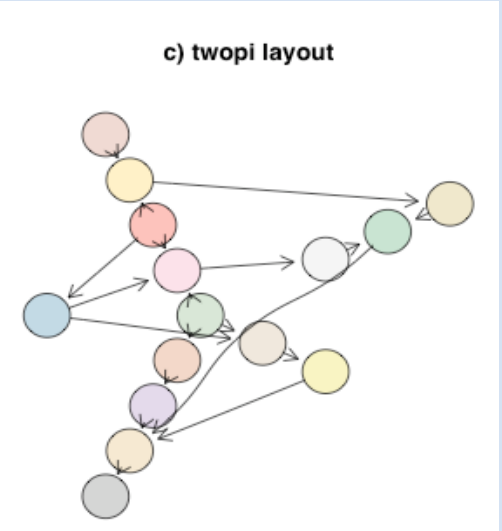


The Rgraphviz package

- an interface to Graphviz (www.graphviz.org)
- different layout algorithms
- graph rendering
- can handle multiple node shapes, edge designs, subgraphs

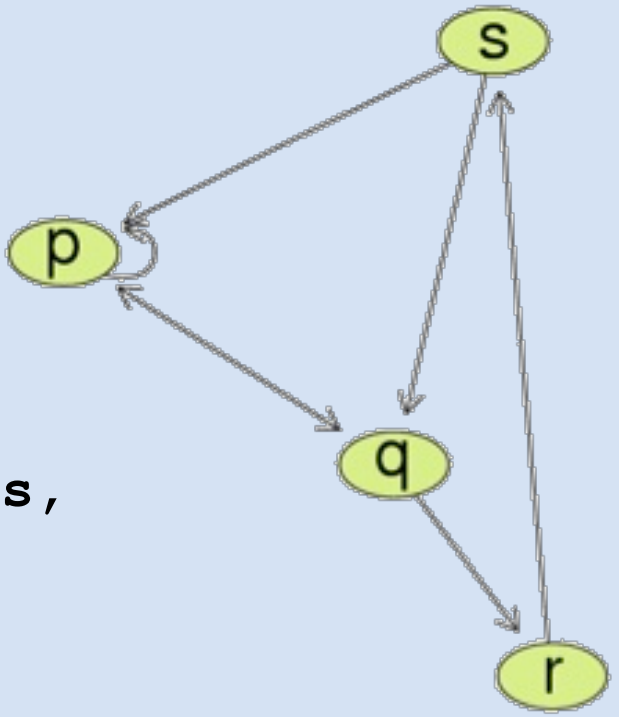






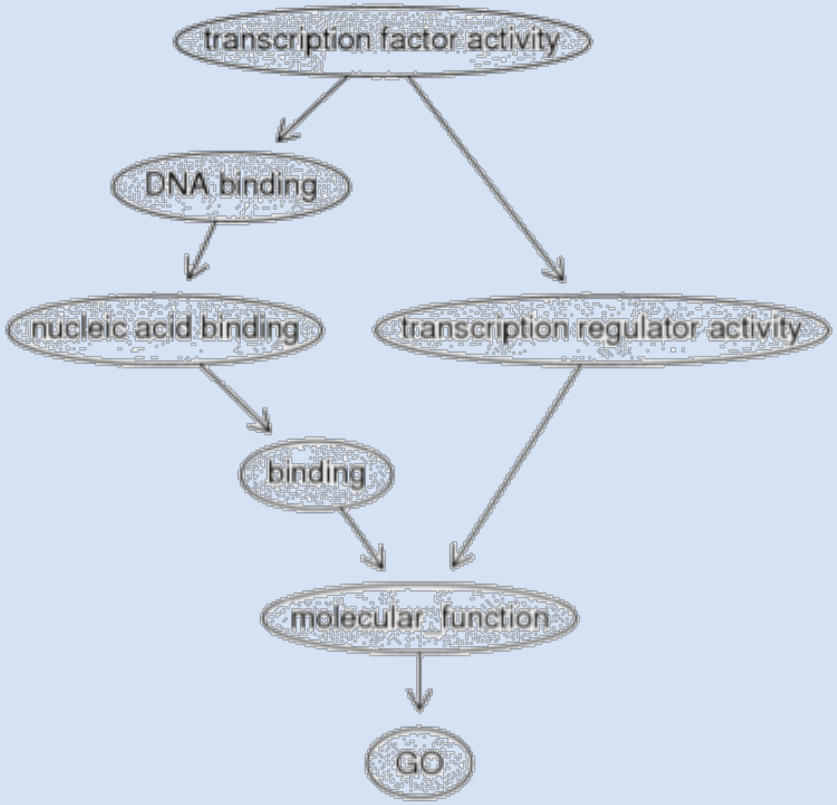
Creating graphs: manual


```
> library("graph")

> myNodes = c("s", "p", "q", "r")

> myEdges = list(
s = list(edges = c("p", "q")),
p = list(edges = c("p", "q")),
q = list(edges = c("p", "r")),
r = list(edges = c("s")))

> g = new("graphNEL", nodes = myNodes,
      edgeL = myEdges, edgemode =
      "directed")
```



Creating graphs: GO

goGraph function in the GOstats package



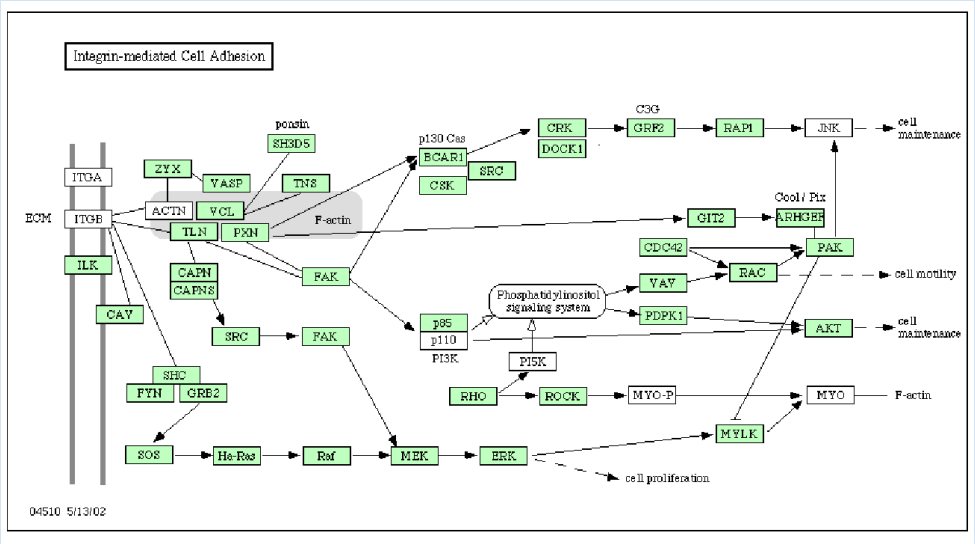

```
> tfG = GOGraph("GO:0003700", GOMFPARENTS)
```

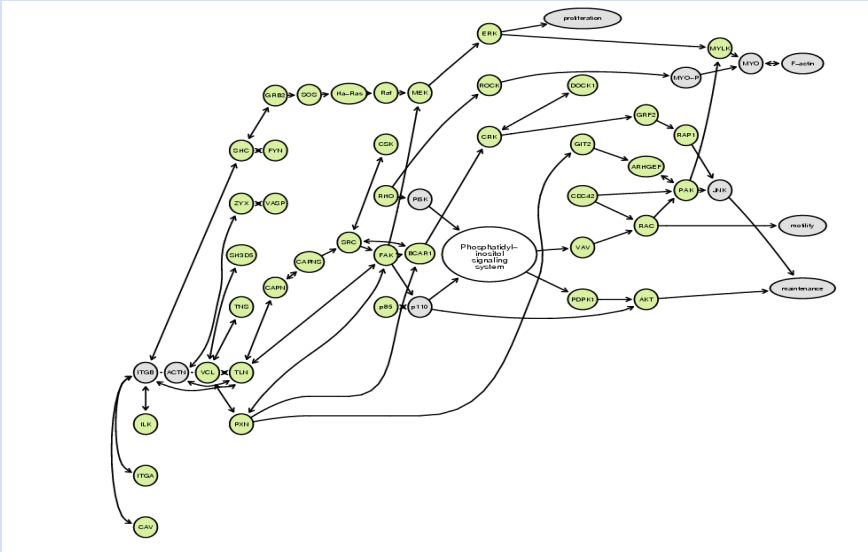
Creating graphs: KEGG

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE

KEGGgraph package:

- parsing of KEGG XML files (locally or from the KEGG webpage)
- KEGG-specific graph operations (merging, subsetting, identifier mapping
- visualization using Rgraphviz





Creating graphs: molecular interaction data

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE

RpsiXML package:

- Retrieve data from molecular interaction databases (PSI-MI XML2.5)
- Convert into R graph objects
- bait-to-prey information: *separateXMLDataByExpt()*
  →list of graph objects
- protein complex data: *buildPCHypergraph()*
  →list of hypergraphs
- transform interaction graphs from one species to another using the Inparanoid database: *graphConverter()*

Plotting graphs: the Rgraphviz package

- Plotting of graphs is a two-step process:

  1) layout ← Graphviz library

  2) rendering ← R's plotting facilities

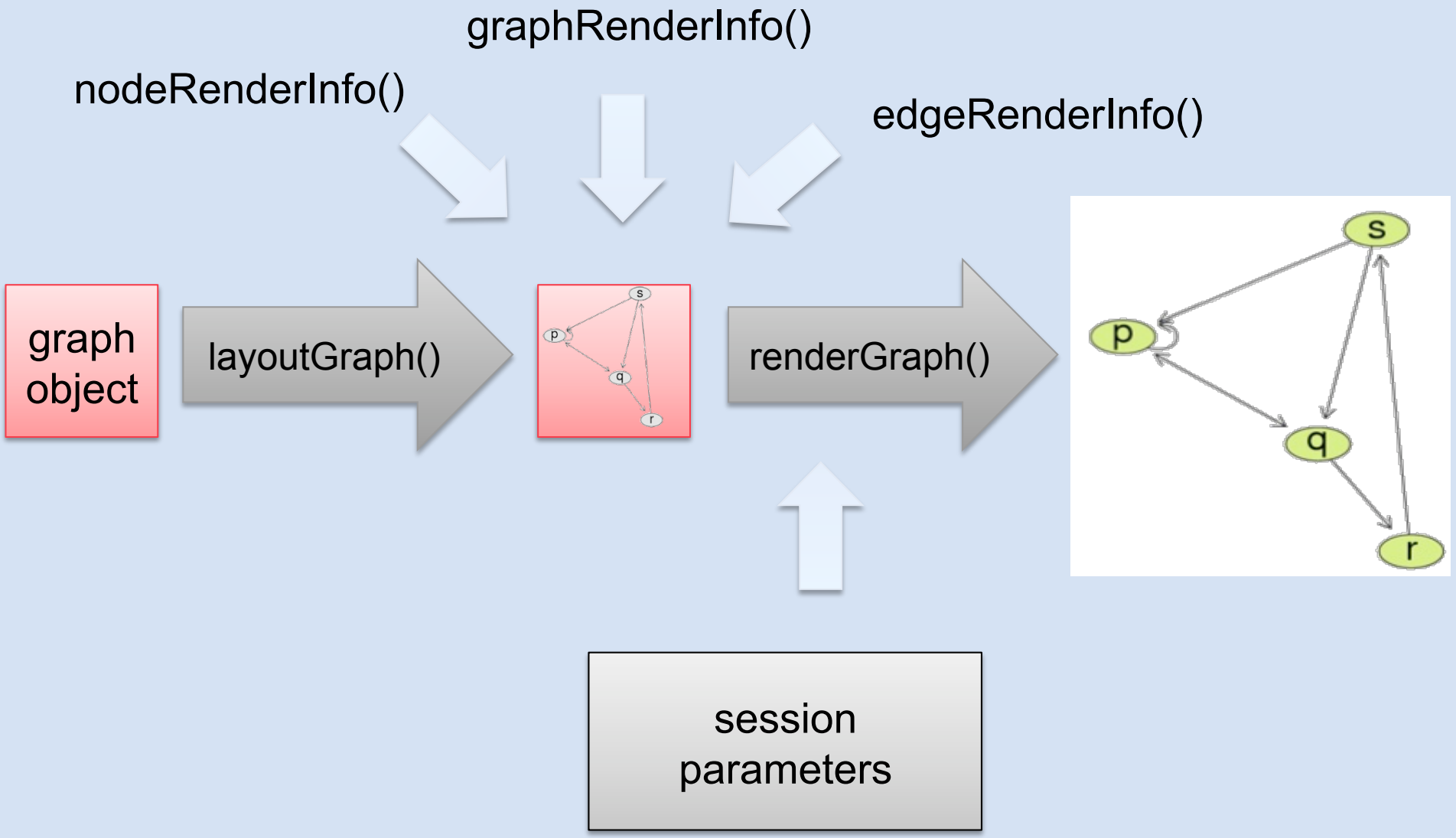
- The two steps are implemented in independent functions:

  - layoutGraph()

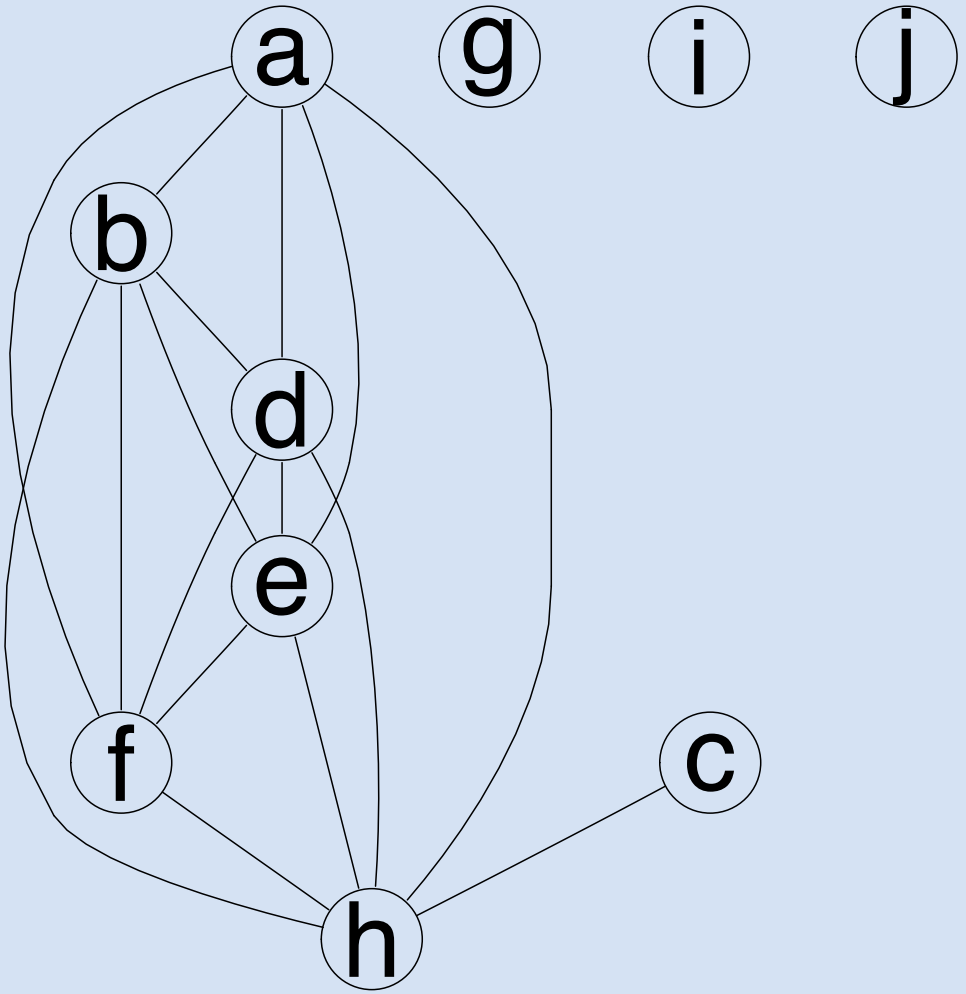
  - renderGraph()

Plotting graphs: the Rgraphviz package

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE

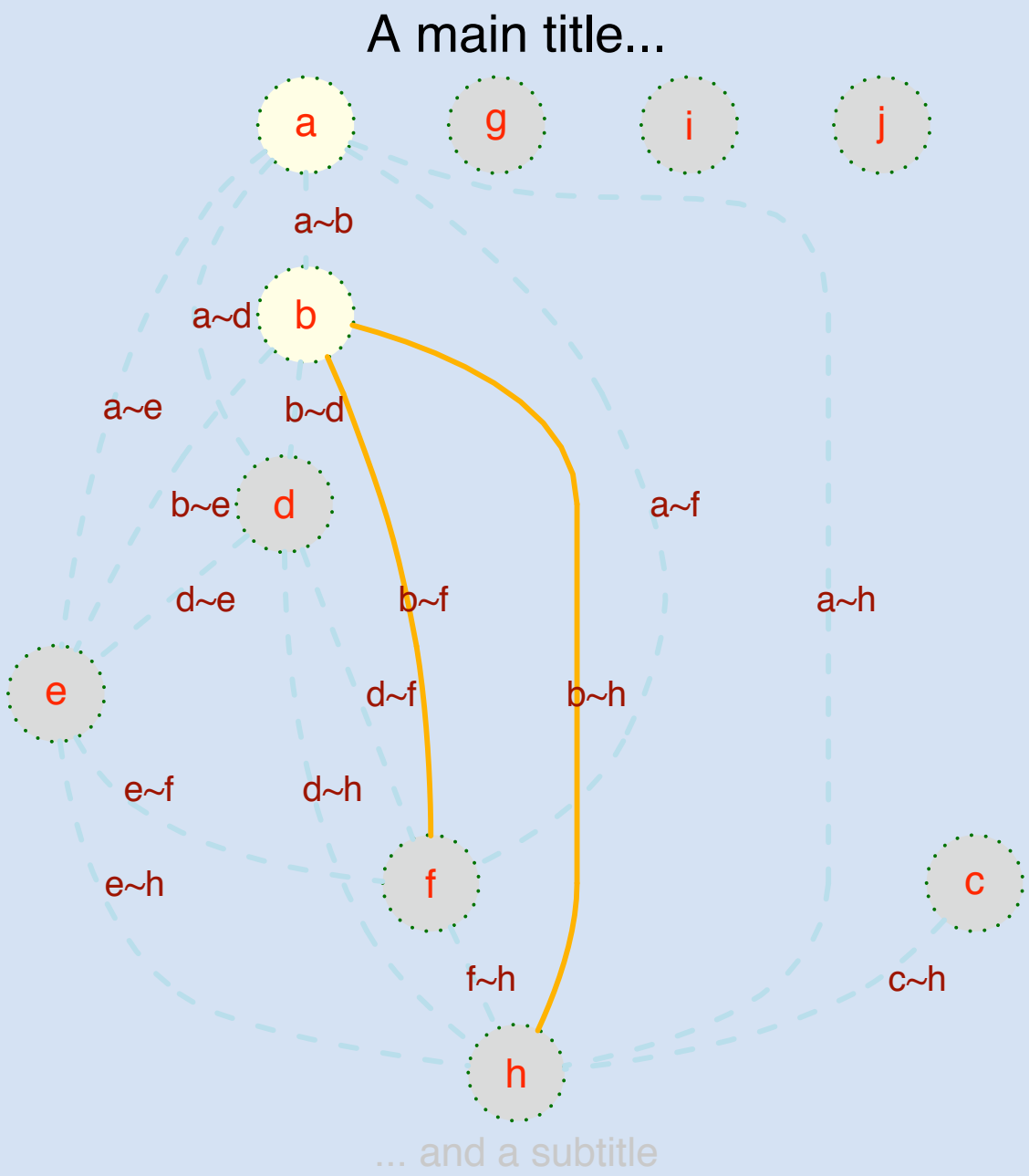


Plotting graphs: the Rgraphviz package



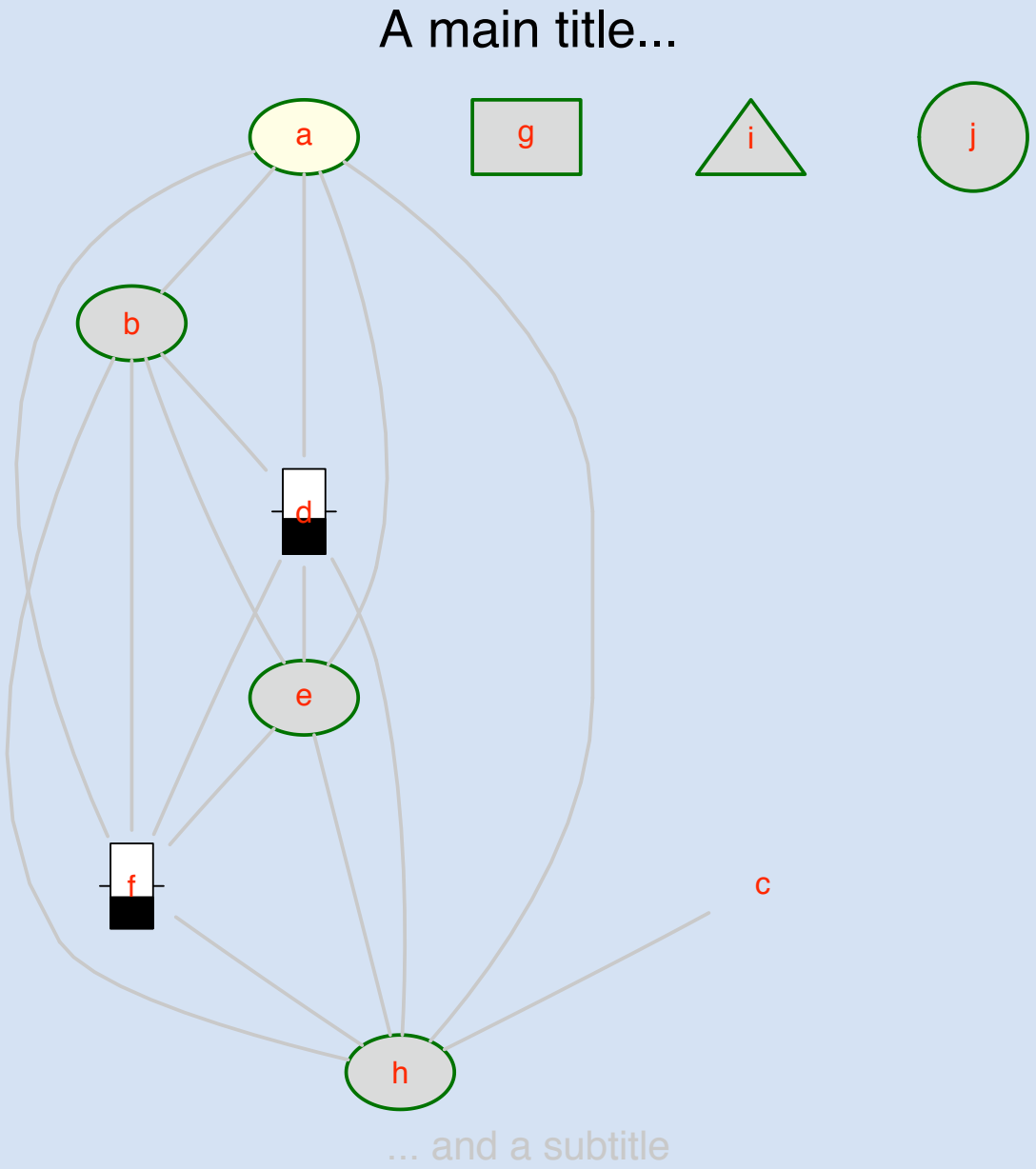
Plotting graphs: the Rgraphviz package

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE



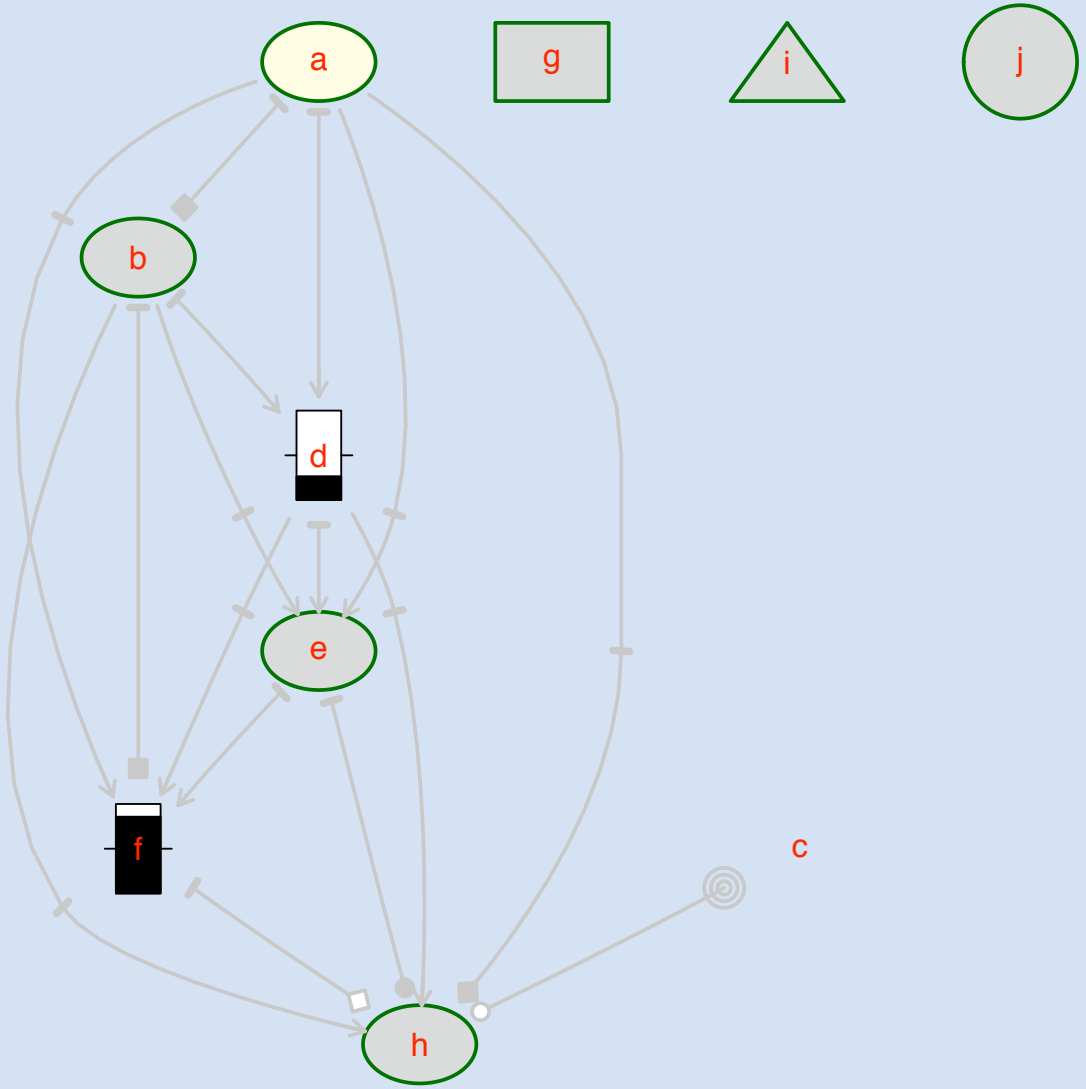
Plotting graphs: the Rgraphviz package

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE



Plotting graphs: the Rgraphviz package

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE



Acknowledgements

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE

- Seth Falcon
- Tony Chiang
- Vincent Carey
- Robert Gentleman
- Jeff Gentry
- Kasper Daniel Hansen
- Deepayan Sarkar
- Denise Scholtens
- Duncan Temple Lang
- David Zhang
- Elizabeth Whalen
- Li Long
- Wolfgang Huber
- Bioconductor developers