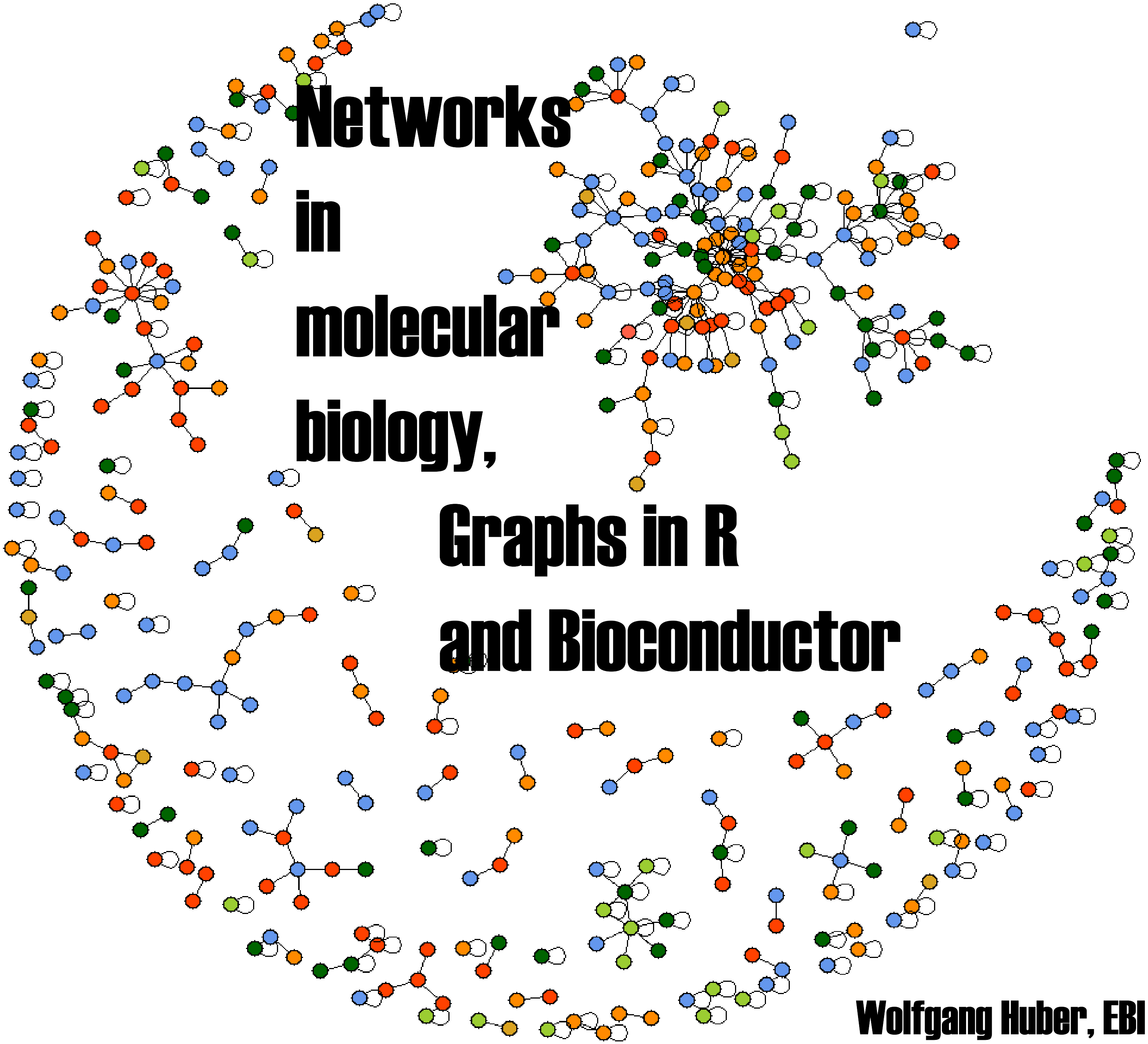
Networks in molecular biology, Graphs in R and Bioconductor



Wolfgang Huber, EBI / EMBL

Motivating examples

Regulatory network:

components = gene products

interactions = regulation of transcription, translation, phosphorylation...

Metabolic network:

components = metabolites, enzymes

interactions = chemical reactions

Physical interaction network:

components = molecules

interactions = binding to each other (e.g. complex)

Probabilistic network:

components = events

interactions = conditioning of each other's probabilities

Genetic interaction network:

components = genes

interactions = synthetic, epistatic, … phenotypes

Objectives

Representation of experimental data

a convenient way to represent and visualize experimental data

Map

(visual) tool to navigate through the world of gene products, proteins, domains, etc.

Predictive Model

complete description of causal connections that allows to predict and engineer the behavior of a biological system, like that of an electronic circuit

Definitions

Graph := set of nodes + set of edges

Edge := pair of nodes

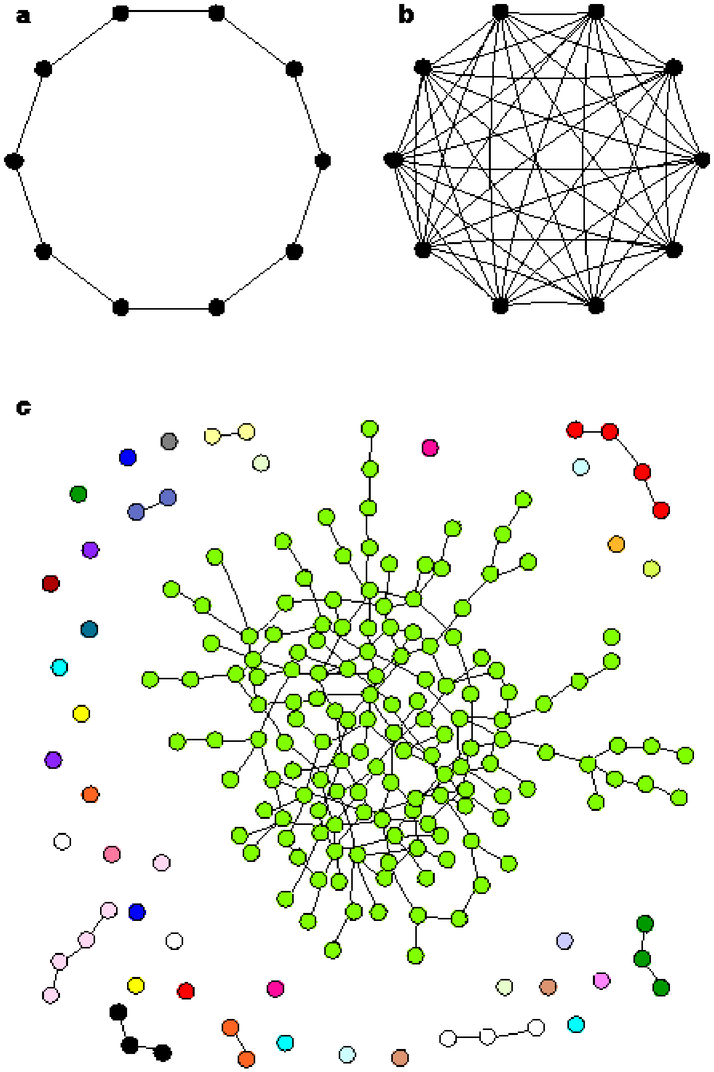
Edges can be

- directed
- undirected
- weighted, typed

special cases: cycles, acyclic graphs, trees

Network topologies

regular

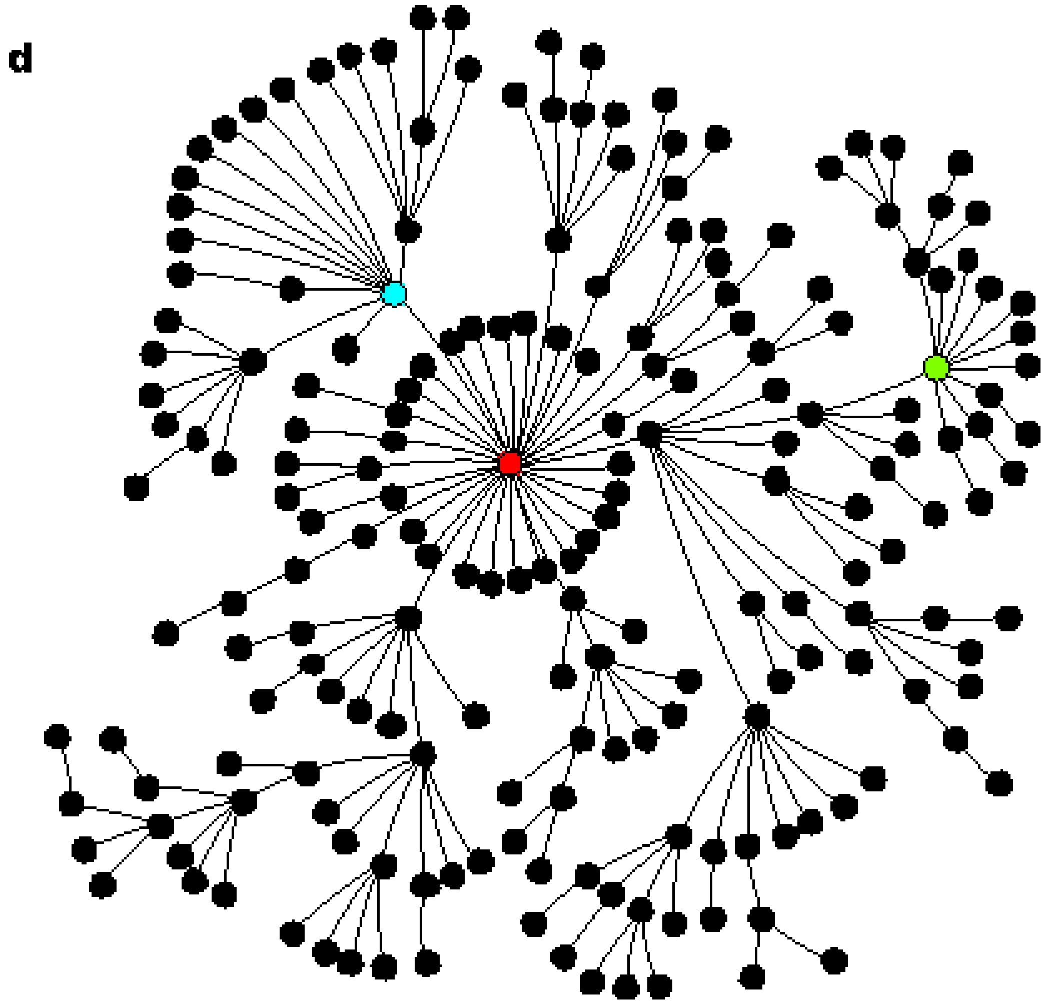


all-to-all

Random graph

(after "tidy" rearrangement of nodes)

Network topologies



Scale-free

Random Edge Graphs

n nodes, m edges

$p(i,j) = 1/m$

with high probability:

m < n/2: many disconnected components

m > n/2: one giant connected component: size ~ n.

(next biggest: size ~ log(n)).

degrees of separation: log(n).

Erdös and Rényi 1960

Some popular concepts:

Small worlds

Clustering

Degree distribution

Motifs

Small world networks

Typical path length („degrees of separation“) is short

many examples:

- communications
- epidemiology / infectious diseases
- metabolic networks
- scientific collaboration networks
- WWW
- company ownership in Germany
- „6 degrees of Kevin Bacon“

But not in

- regular networks, random edge graphs

Cliques and clustering coefficient

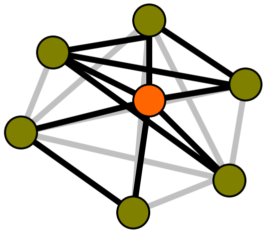
Clique: **every node connected to everyone else**

Clustering coefficient:

$$c = \frac{\text{no. edges between first-degree neighbors}}{\text{maximum possible number of such edges}}$$

Random network: E[c]=p

Real networks: c » p



Degree distributions

p(k) = proportion of nodes that have *k* edges

Random graph: p(k) = Poisson distribution with some parameter λ („scale“)

Many real networks: p(k) = power law,

$$p(k) \sim k^{-\gamma}$$

„scale-free“

In principle, there could be many other distributions: exponential, normal, …

Growth models for scale free networks

Start out with one node and continously add nodes, with preferential attachment to existing nodes, with probability ~ degree of target node.

$$\Rightarrow p(k) \sim k^{-3}$$

(Simon 1955; Barabási, Albert, Jeong 1999)

"The rich get richer"

Modifications to obtain $\gamma \neq 3$:

Through different rules for adding or rewiring of edges, can tune to obtain any kind of degree distribution

Real networks

- tend to have power-law scaling (truncated)
- are ‚small worlds' (like random networks)
- have a high clustering coefficient independent of network size (like lattices and unlike random networks)

Network motifs

:= pattern that occurs more often than in randomized networks

Intended implications

duplication: useful building blocks are reused by nature

there may be evolutionary pressure for convergence of network architectures

Network motifs

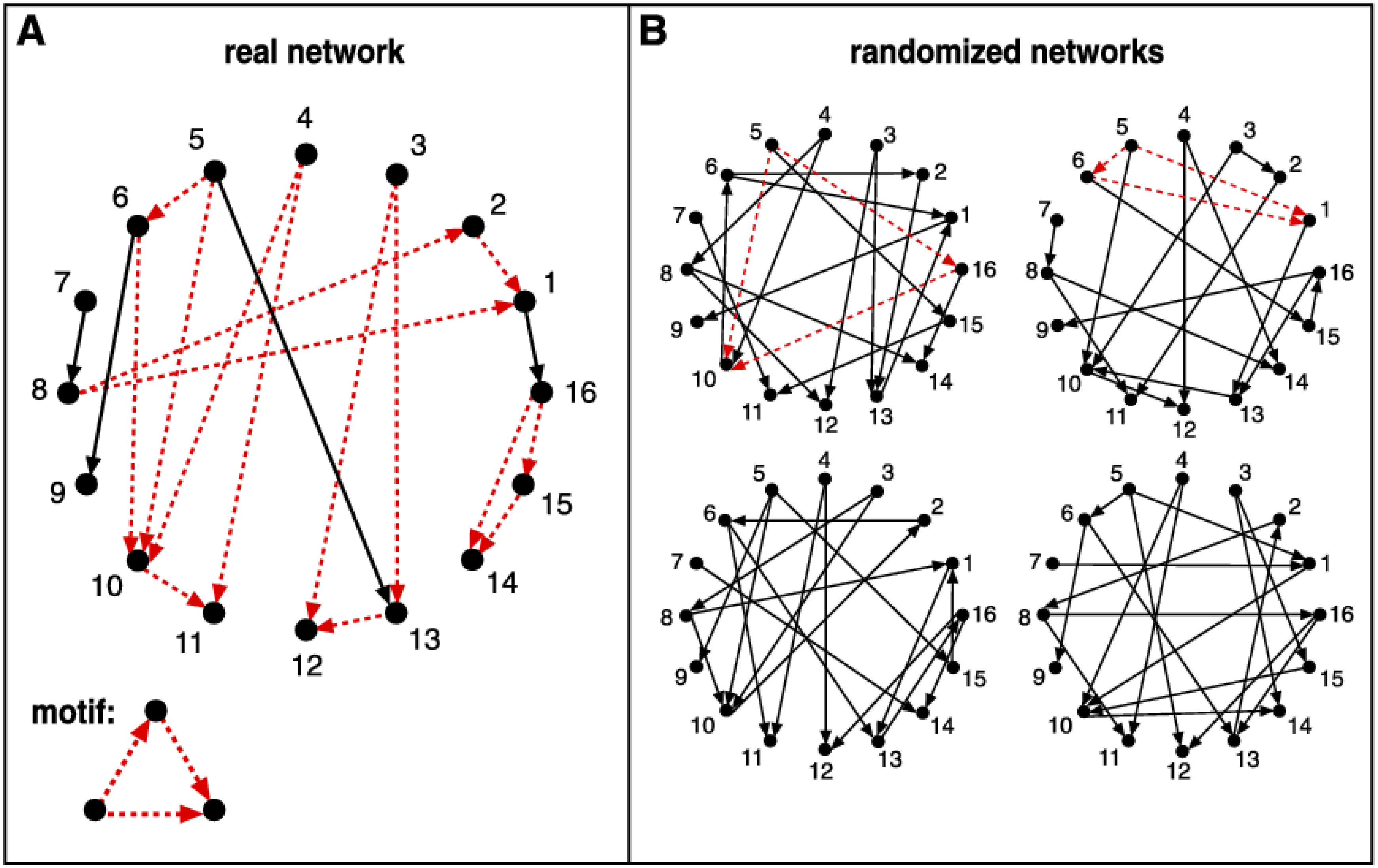
Starting point: graph with directed edges

Scan for n-node subgraphs (n=3,4) and count number of occurence

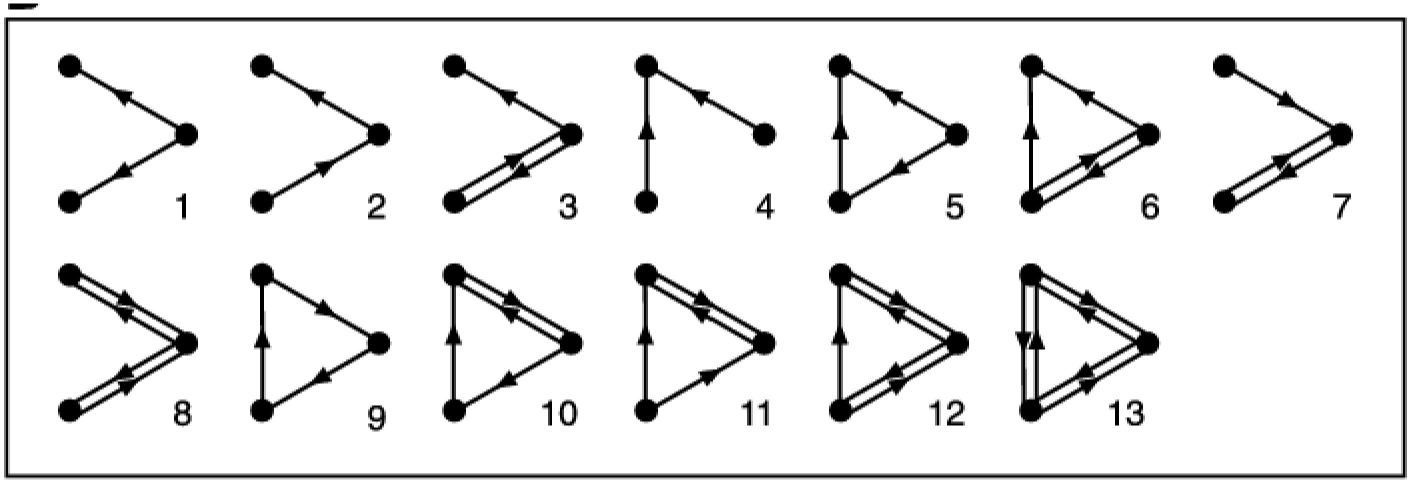
Compare to randomized networks

(randomization preserves in-, out- and in+out- degree of each node, and the frequencies of all (n-1)-subgraphs)

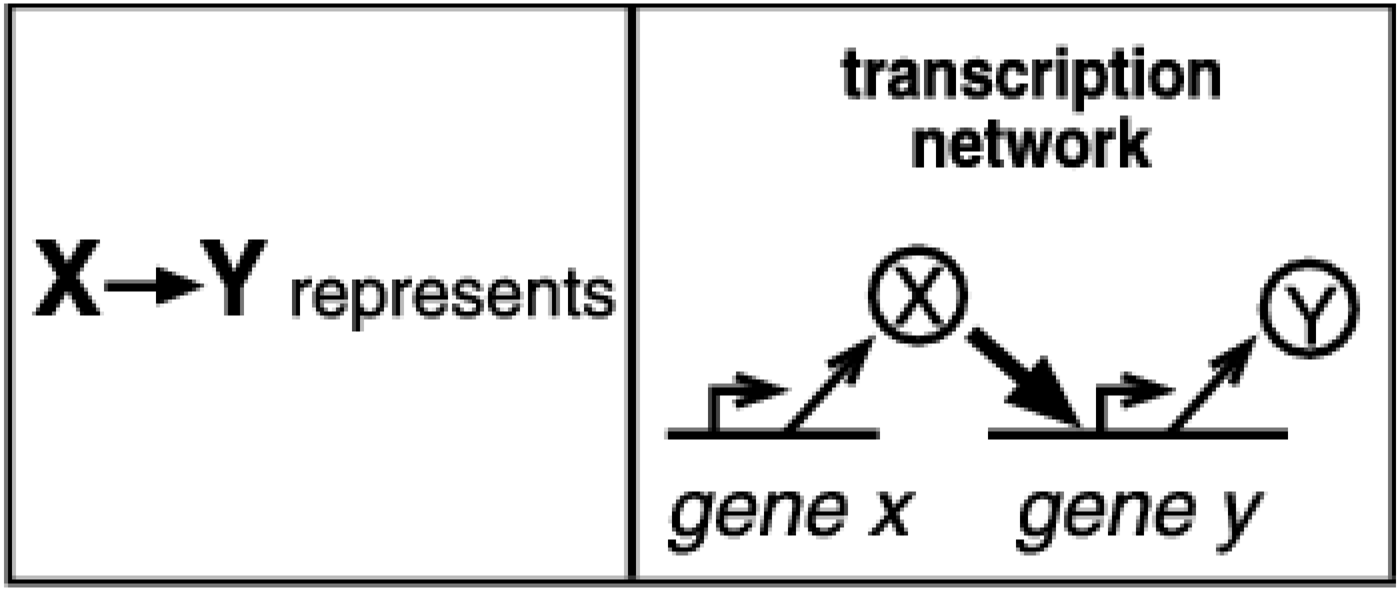
Schematic view of motif detection



All 3-node connected subgraphs



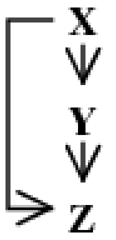
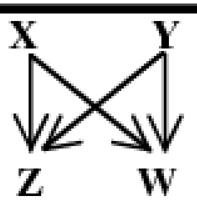
Transcription networks

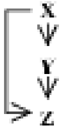
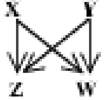
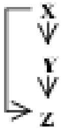
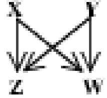
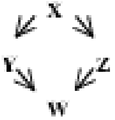
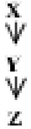
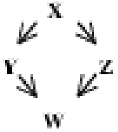
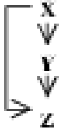
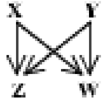
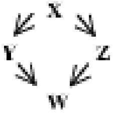
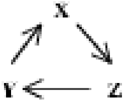
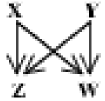
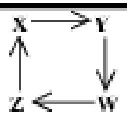
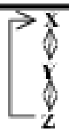
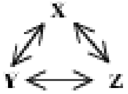
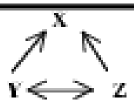


Nodes = transcription factors

Directed edge: X regulates transcription of Y

3- and 4-node motifs in transcription networks

Network	Nodes	Edges	N_{real}	$N_{\mathrm{rand}} \pm \mathrm{SD}$	Z score	N_{real}	$N_{\mathrm{rand}} \pm \mathrm{SD}$	Z score
Gene regulation (transcription)					**Feed-forward loop**			**Bi-fan**
E. coli	424	519	40	7 ± 3	10	203	47 ± 12	13
*S. cerevisiae**	685	1,052	70	11 ± 4	14	1812	300 ± 40	41

Network	Nodes	Edges	N_{real}	N_{rand} ± SD	Z score	N_{real}	N_{rand} ± SD	Z score	N_{real}	N_{rand} ± SD	Z score
Gene regulation (transcription)					**Feed-forward loop**			**Bi-fan**			
E. coli	424	519	40	7 ± 3	10	203	47 ± 12	13			
*S. cerevisiae**	685	1,052	70	11 ± 4	14	1812	300 ± 40	41			
Neurons					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
C. elegans†	252	509	125	90 ± 10	3.7	127	55 ± 13	5.3	227	35 ± 10	20
Food webs					**Three chain**			**Bi-parallel**			
Little Rock	92	984	3219	3120 ± 50	2.1	7295	2220 ± 210	25			
Ythan	83	391	1182	1020 ± 20	7.2	1357	230 ± 50	23			
St. Martin	42	205	469	450 ± 10	NS	382	130 ± 20	12			
Chesapeake	31	67	80	82 ± 4	NS	26	5 ± 2	8			
Coachella	29	243	279	235 ± 12	3.6	181	80 ± 20	5			
Skipwith	25	189	184	150 ± 7	5.5	397	80 ± 25	13			
B. Brook	25	104	181	130 ± 7	7.4	267	30 ± 7	32			
Electronic circuits (forward logic chips)					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
s15850	10,383	14,240	424	2 ± 2	285	1040	1 ± 1	1200	480	2 ± 1	335
s38584	20,717	34,204	413	10 ± 3	120	1739	6 ± 2	800	711	9 ± 2	320
s38417	23,843	33,661	612	3 ± 2	400	2404	1 ± 1	2550	531	2 ± 2	340
s9234	5,844	8,197	211	2 ± 1	140	754	1 ± 1	1050	209	1 ± 1	200
s13207	8,651	11,831	403	2 ± 1	225	4445	1 ± 1	4950	264	2 ± 1	200
Electronic circuits (digital fractional multipliers)					**Three-node feedback loop**			**Bi-fan**			**Four-node feedback loop**
s208	122	189	10	1 ± 1	9	4	1 ± 1	3.8	5	1 ± 1	5
s420	252	399	20	1 ± 1	18	10	1 ± 1	10	11	1 ± 1	11
s838‡	512	819	40	1 ± 1	38	22	1 ± 1	20	23	1 ± 1	25
World Wide Web					**Feedback with two mutual dyads**			**Fully connected triad**			**Uplinked mutual dyad**
nd.edu§	325,729	1.46e6	1.1e5	2e3 ± 1e2	800	6.8e6	5e4±4e2	15,000	1.2e6	1e4 ± 2e2	5000

System-size dependence

Extensive variable: proportional to system size. E.g. mass, diameter, number of molecules

Intensive variable: independent of system size. E.g. temperature, pressure, density, concentration

„Vanishing variable“: decreases with system size. E.g. Heat loss through radiation; in a city, probability to bump into one particular person

Alon et al.: In real networks, number of occurences of a motif is extensive. In randomized networks, it is non-extensive.

Examples

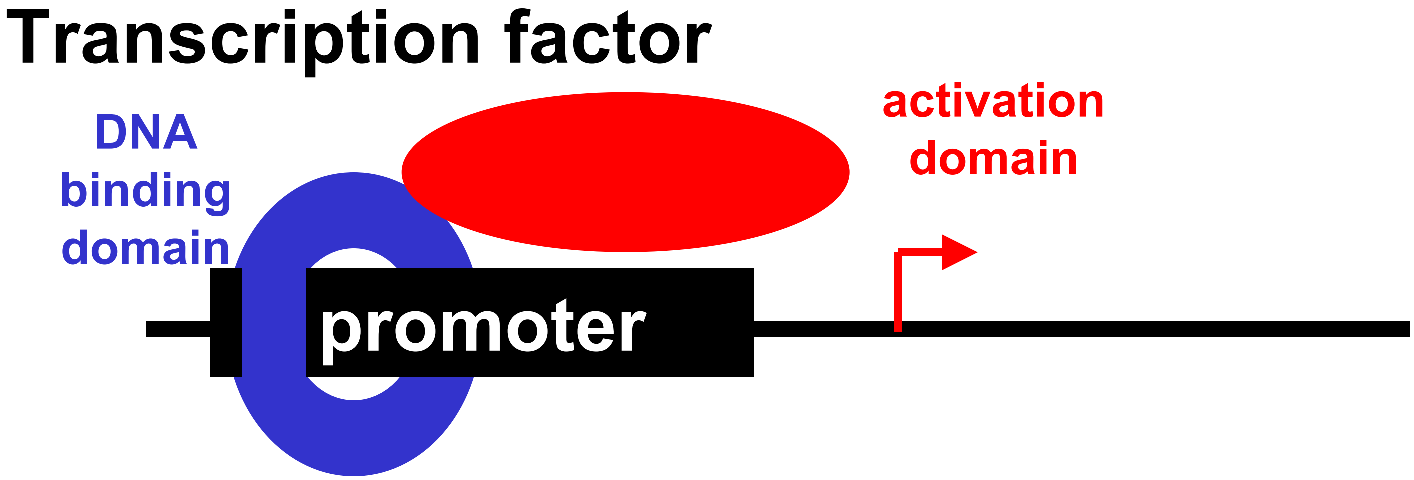
Protein interactions

(Yeast-2-Hybrid)

Genetic interactions

**(Rosetta Compendium,
Yeast synthetic lethal screen)**

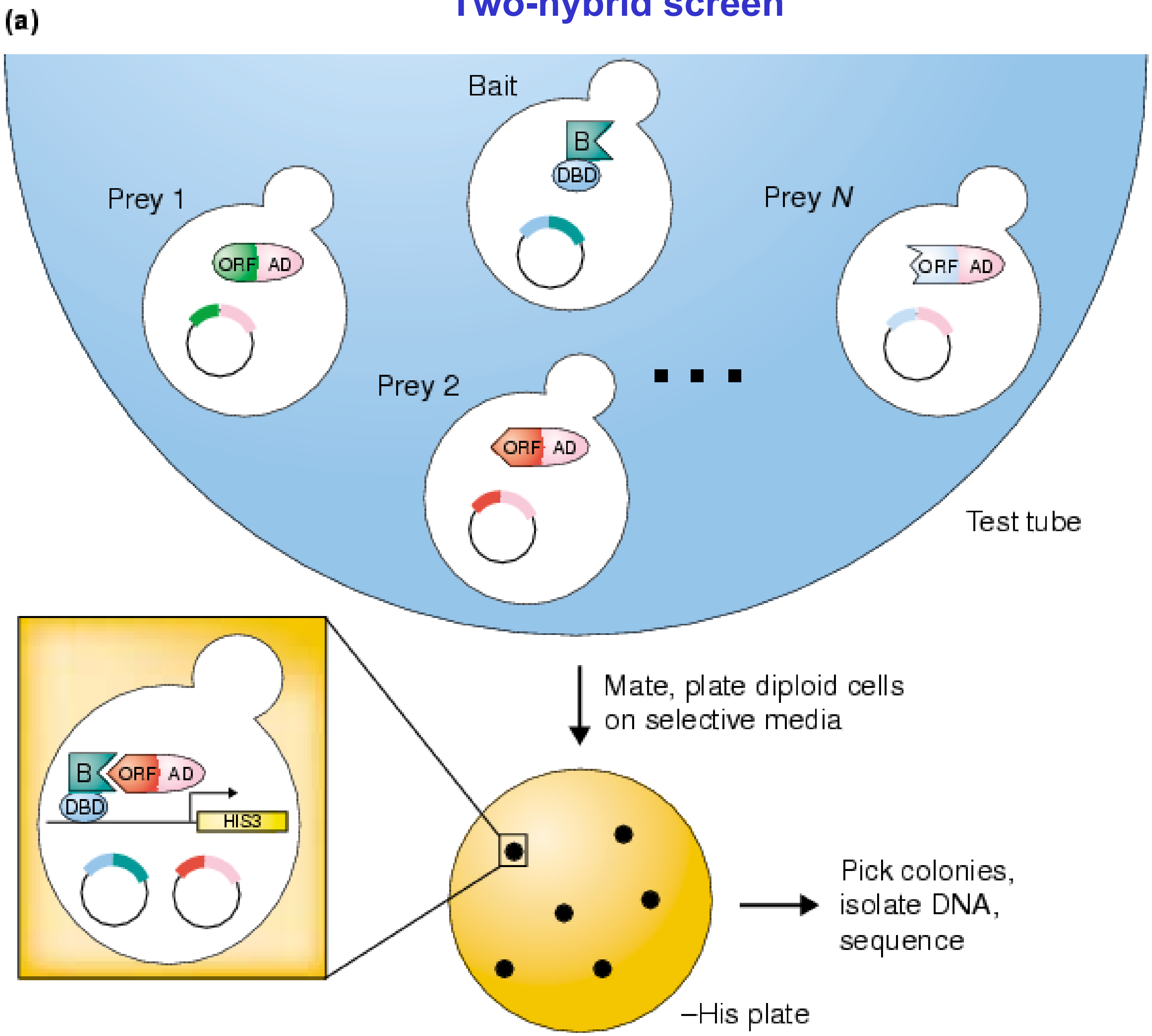
Two-hybrid screen



Idea:

„Make potential pairs of interacting proteins a transcription factor for a reporter gene“

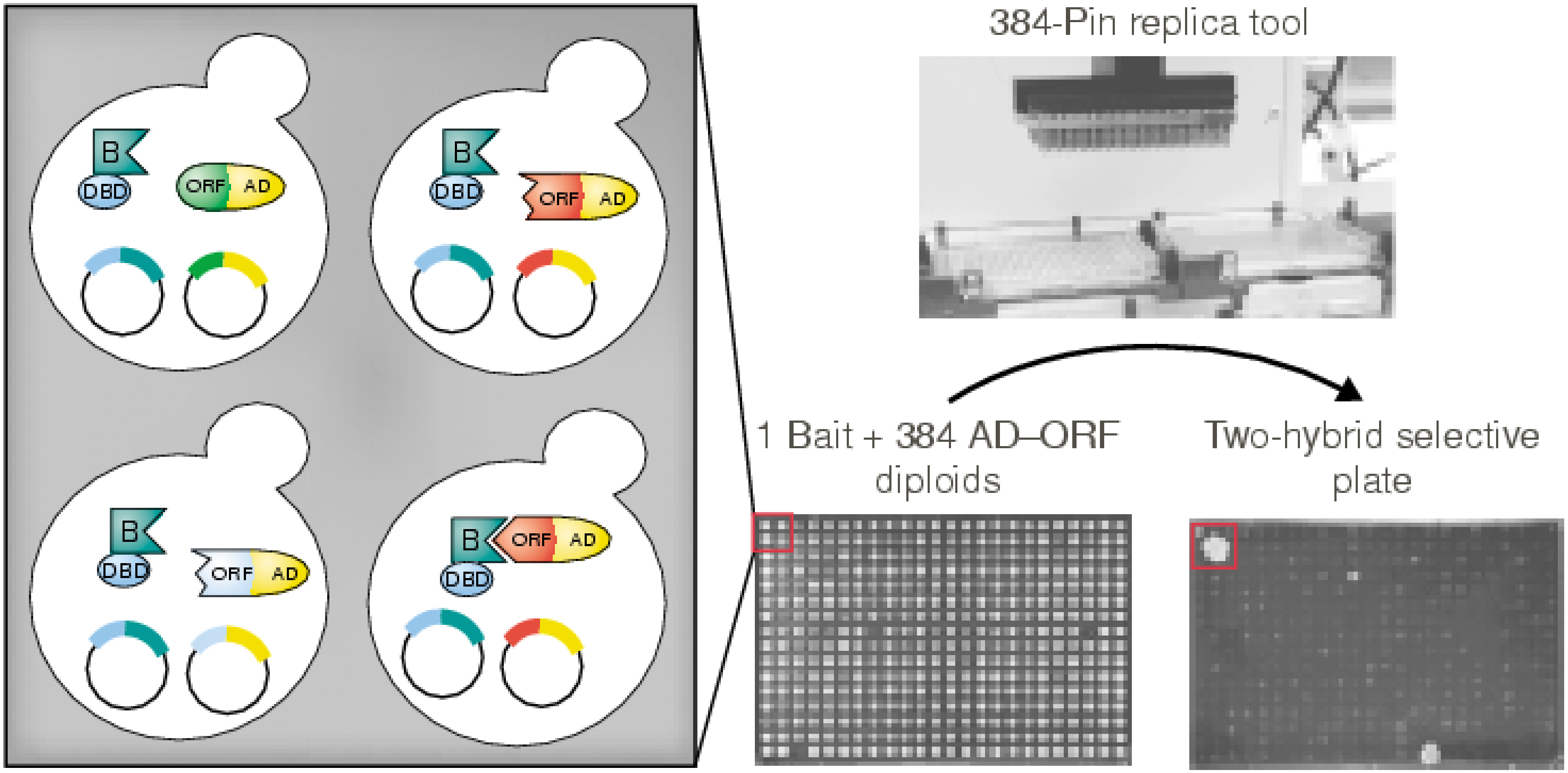
Two-hybrid screen



Mate, plate diploid cells on selective media

Pick colonies, isolate DNA, sequence

Two-hybrid arrays



Current Opinion in Chemical Biology

Colony array:

each colony expresses a defined pair of proteins

Table 2.

Two-hybrid array screens discussed in this paper.

Organism	Project	Proteins*	Assays*	Interactions*	Refs
Drosophila	Cell cycle proteins	13	45	19	[7]
C. elegans	Vulva development	29	841	8[†]	[9]
Mouse	Whole-genome pilot	~3500	$\sim 12\times10^6$	145	[15••]
HCV	Whole genome	10	~100	0/3[‡]	[16]
Vaccinia	Whole genome	266	~64 000	37[§]	[17•]
Yeast	One by one array	192	~1 150 000	281	[18••]
Yeast	Pool by pool	~6000	~36 000 000	4549/841[‡]	[19,20••]
Yeast	Cell polarity	68	~408 000	191[#]	[10]
Yeast	Proteasome	31	~186 000	55	[12]

Sensitivity, specificity and reproducibility

Specificity – false positives: **the experiment reports an interaction even though is really none**

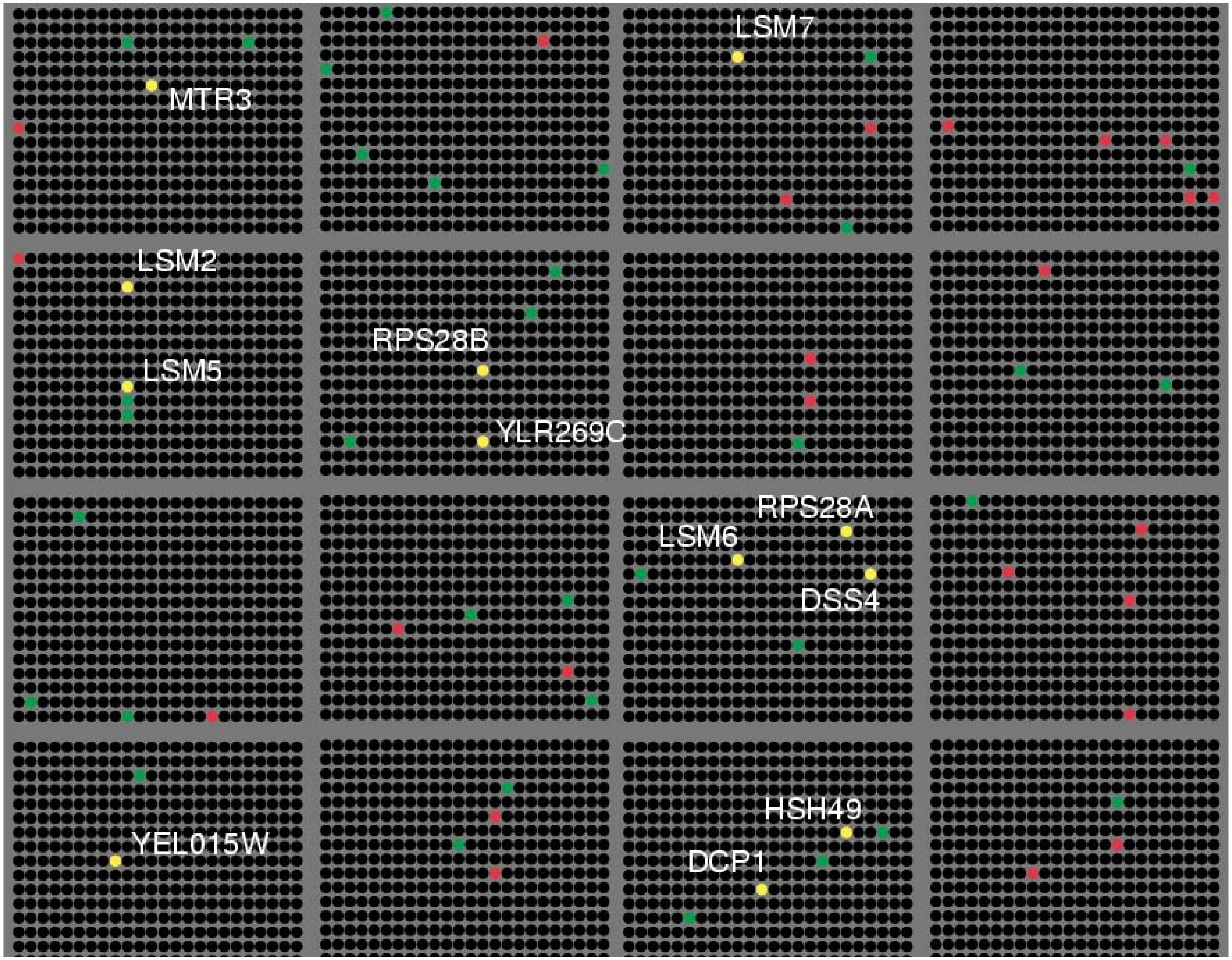
Sensitivity – false negatives: **the experiment reports no interaction even though is really one**

Problem: **what is the objective definition of an interaction?**

(Un)reproducibility: **the experiment reports different results when it is repeated**

„The molecular reasons for that are not really understood...“ (Uetz 2001)

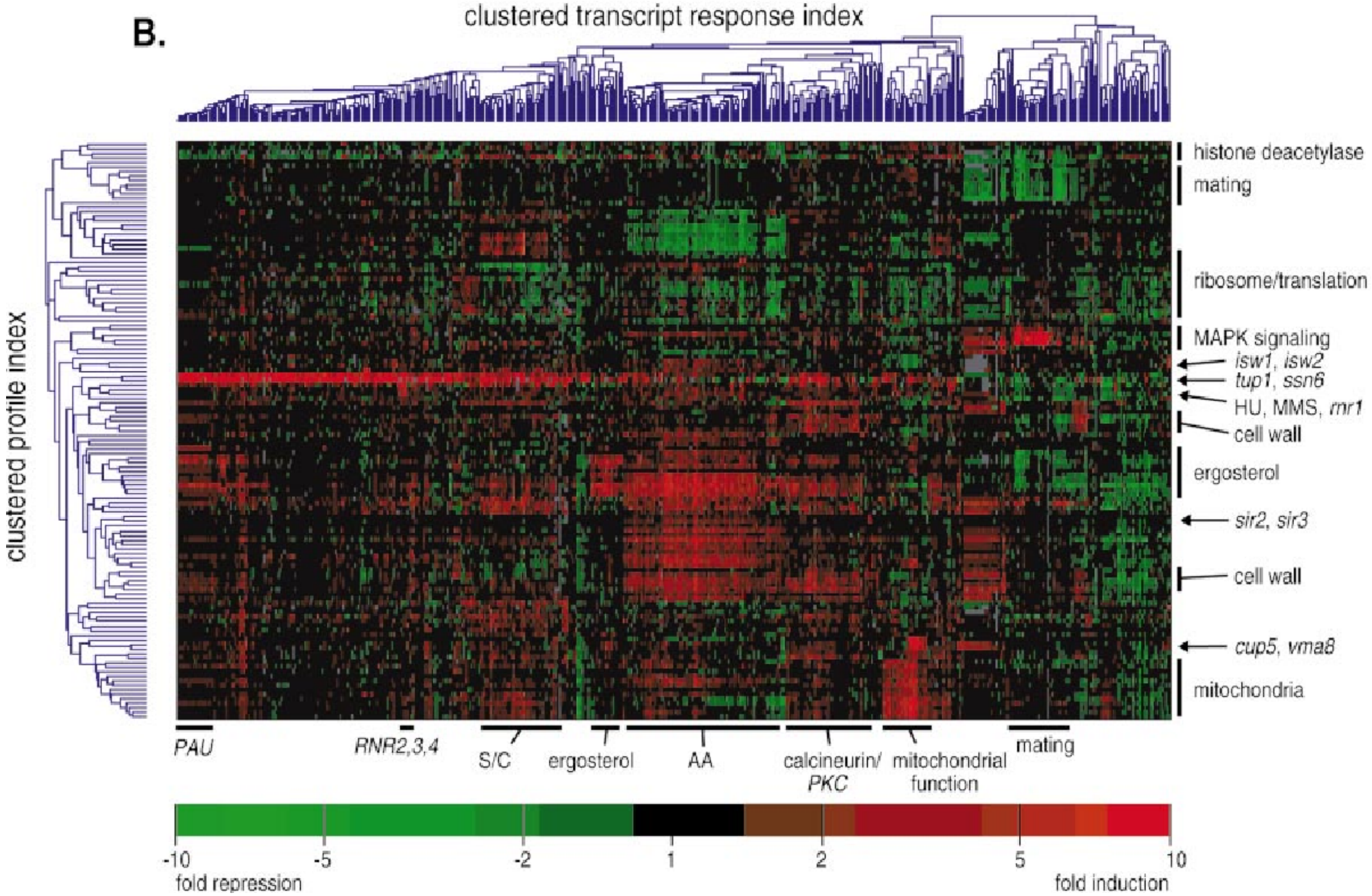
Reproducibility



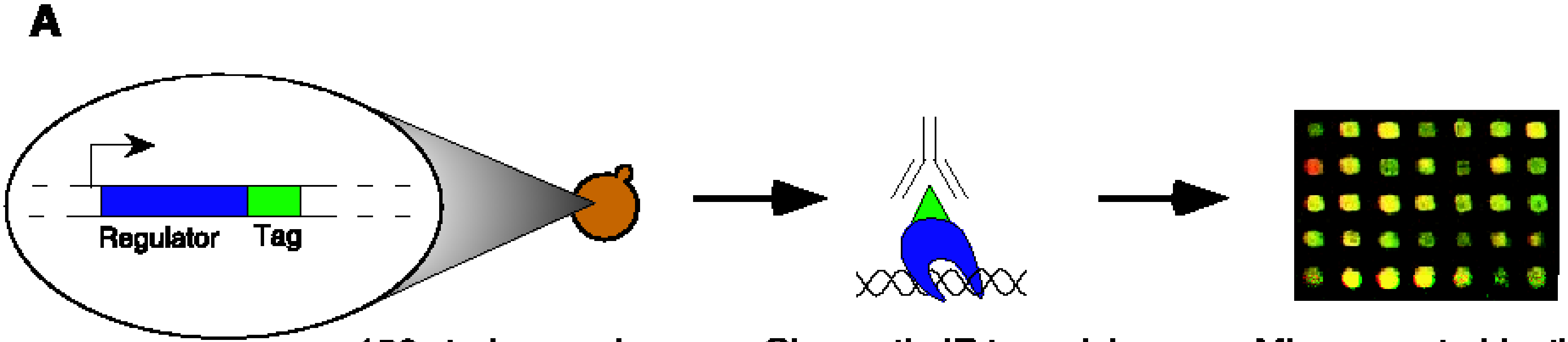
Rosetta compendium

568 transcript levels

300 mutations or chemical treatments



Transcriptional regulatory networks from "genome-wide location analysis"



106 strains, each with a tagged regulator

Chromatin IP to enrich promoters bound by regulator in vivo

Microarray to identify promoters bound by regulator in vivo

regulator := a transcription factor (TF) or a ligand of a TF
tag: c-myc epitope

106 microarrays
samples: enriched (tagged-regulator + DNA-promoter)
probes: cDNA of all promoter regions
spot intensity ~ affinity of a promotor to a certain regulator

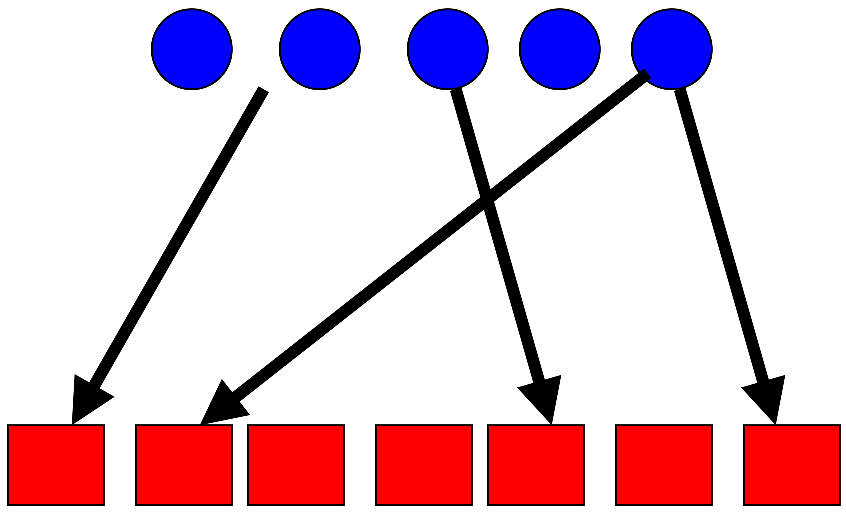
Transcriptional regulatory networks
bipartite graph

106 regulators (TFs)

6270 promoter regions

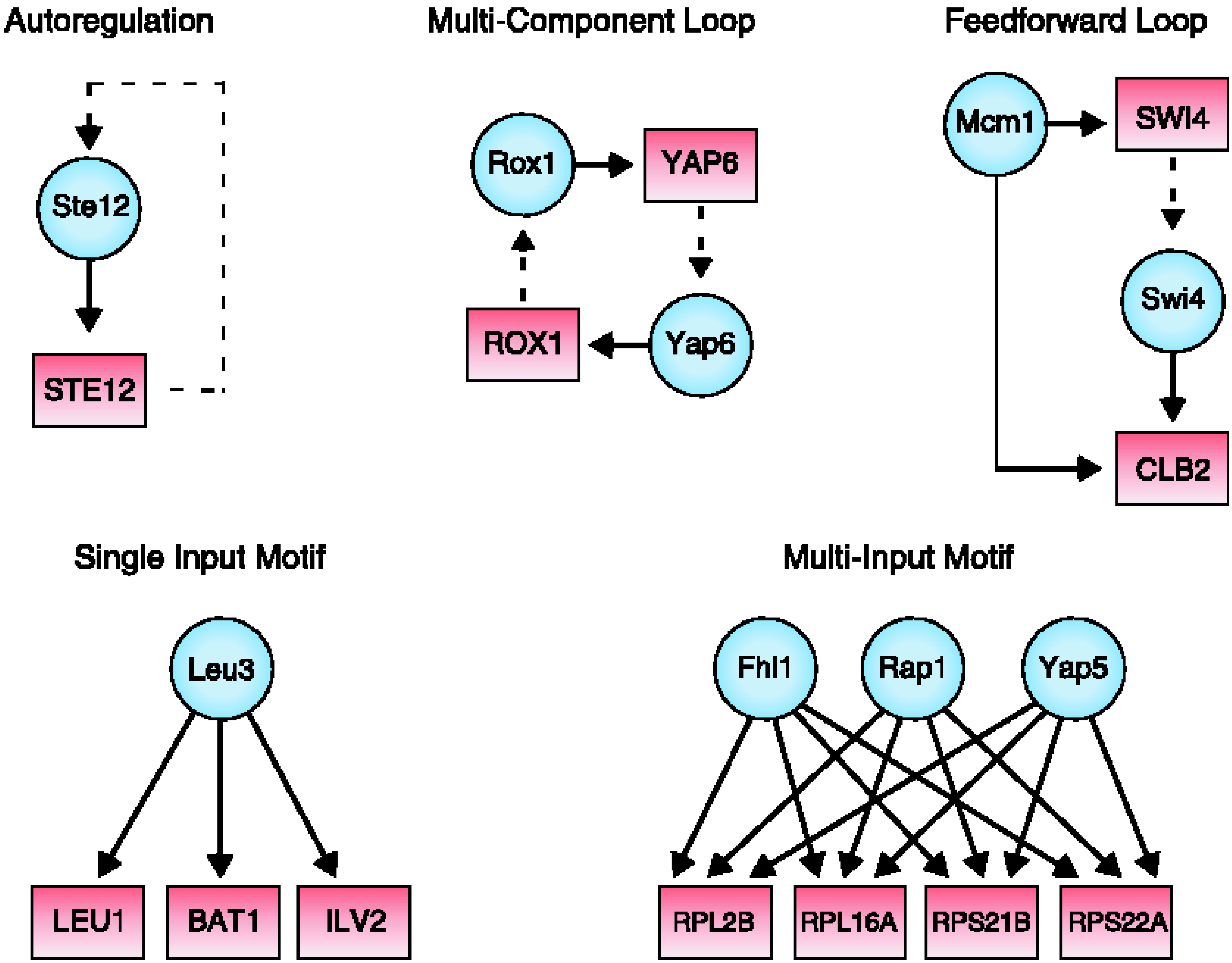
							1
		1					
			1				
						1	
				1			
	1						
						1	

regulators



promoters

Network motifs



Autoregulation

Multi-Component Loop

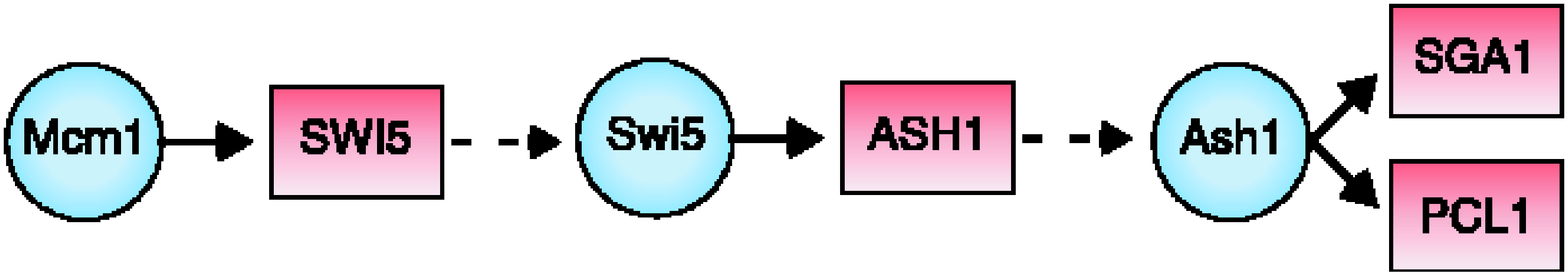
Feedforward Loop

Single Input Motif

Multi-Input Motif

Network motifs

Regulator Chain



Global Mapping of the Yeast Genetic Interaction Network

Amy Hin Yan Tong,…49 other people,
…Charles Boone

Science 303 (6 Feb 2004)

▶ Buffering and Genetic Variation

In yeast, ~73% of gene deletions are "non-essential"

(Glaever et al. Nature 418 (2002)

In Drosophila, ~95%

(Boutros et al. Science 303 (2004))

In Human, ca. 1 SNP / 1.5kB

Evolutionary pressure for robustness

Bilateral asymmetry is positively correlated with inbreeding

Most genetic variation is neutral to fitness, but may well affect quality of life

Probably mechanistic overlap between buffering of genetic, environmental and stochastic perturbations

▶ Models for Buffering

Comparison of single mutants to double mutants in otherwise isogenic genetic background

Synthetic Genetic Array (SGA) analysis (Tong, Science 2001): cross mutation in a "query" gene into a (genome-wide) array of viable mutants, and score for phenotype.

Tong 2004: 132 queries x 4700 mutants

▶ Buffering

A buffered by B

(i) molecular function of A can also be performed by B with sufficient efficiency

(ii) A and B part of a complex, with loss of A or B alone, complex can still function, but not with loss of both

(iii) A and B are in separate pathways, which can substitute each other's functions.

structural similarity -
physically interaction -
maybe, but neither is necessary.

▶ Selection of 132 queries

- actin-based cell polarity
- cell wall biosynthesis
- microtubule-based chromosome segregation
- DNA synthesis and repair

Reproducibility

Each screen 3 times: 3x132x4700 = 1.8 Mio measurements

25% of interactions observed only 1/3 times

4000 interactions amongst 1000 genes confirmed by tetrad or random spore analysis ("FP neglible")

FN rate: 17-41%

▶ Statistics

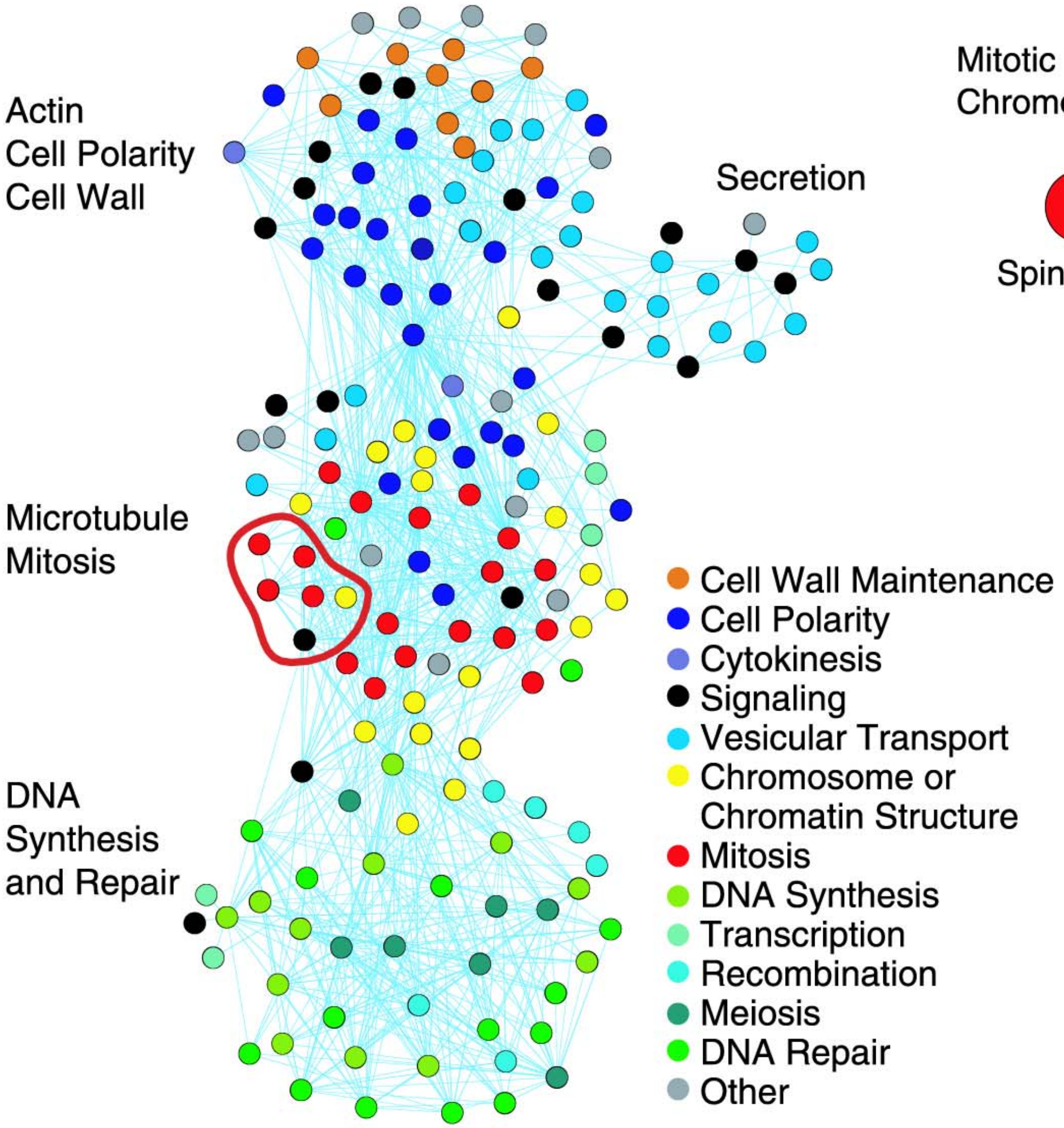
Hits per query gene:

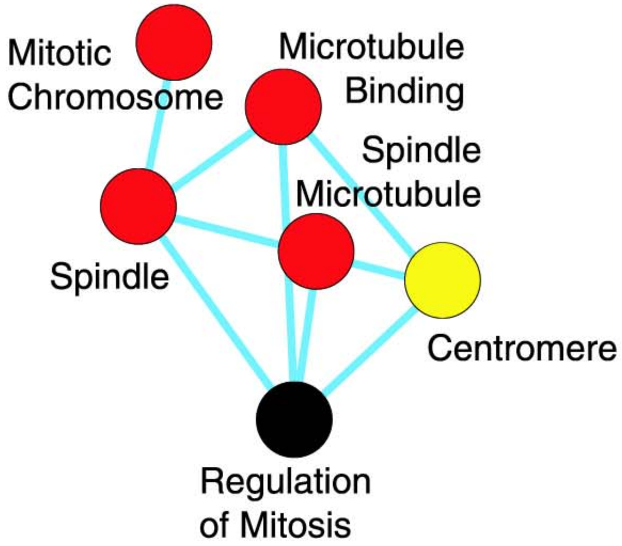
range 1...146, average 34 (!)   power-law (γ=-2)

Physical interactions: ~8

Dubious calculation: ~100,000 interactions

GO





Genes with same or "similar" GO category are more likely to interact

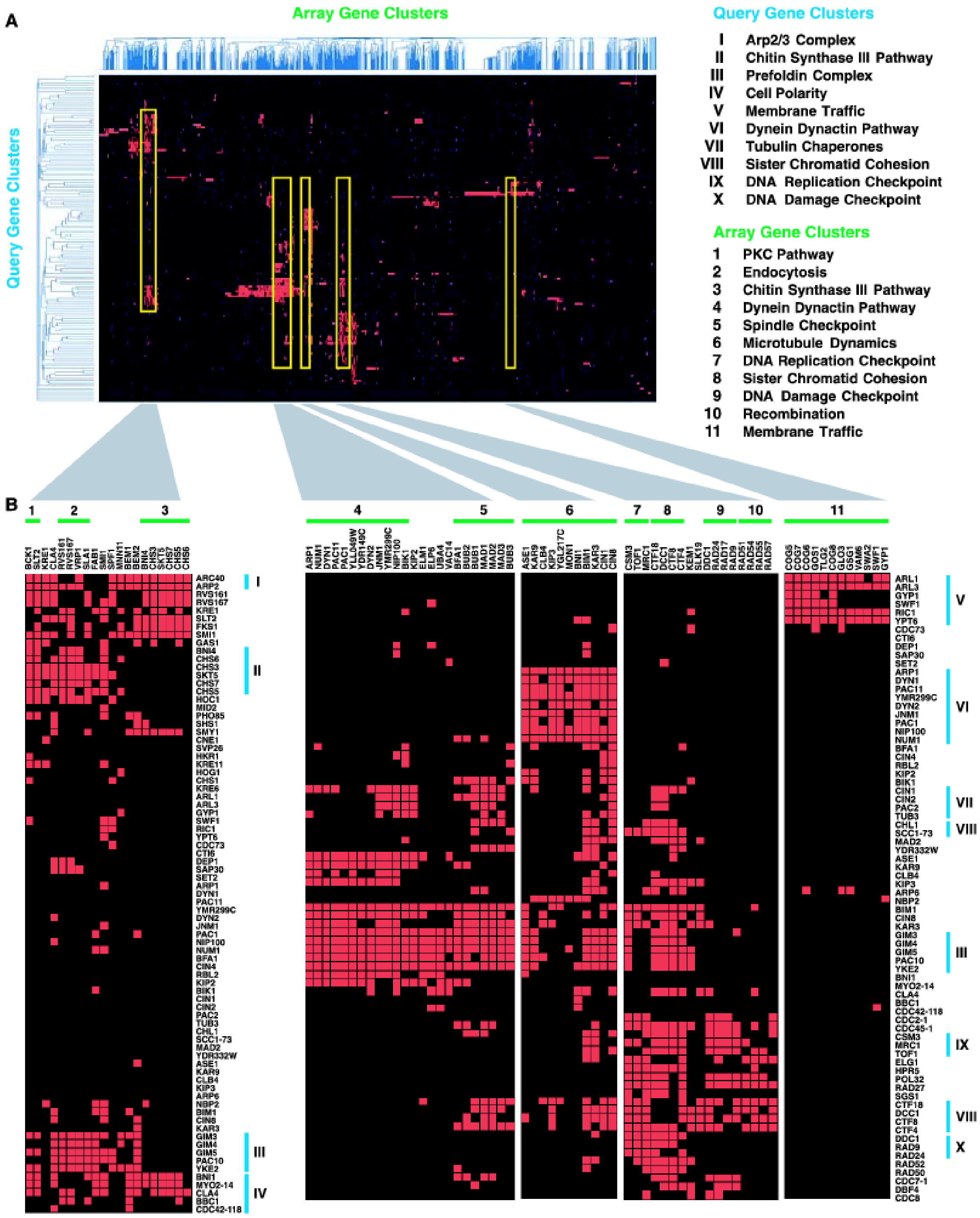
▶ Patterns

SGI more likely between genes
- with same mutant phenotype
- with same localization
- in same complex (but this explains only 1 % of IAs)
- that are homologous (but this explains only 2% of IAs)

Genes that have many common SGI partners tend to also physically interact:

30 / 4039 SGI pairs are also physically interacting
27 / 333 gene pairs with >=16 common SGI partners
factor: 11

Assignment of function to "new" genes



▶ Genetic interaction network

SGI more likely between genes

- with same mutant phenotype

- with same localization

- in same complex (but this explains only 1 % of IAs)

- that are homologous (but this explains only 2% of IAs)

A dense small world:

Average path-length 3.3 (like random graph)

High clustering coefficient (immediate SGI partners of a gene tend to also interact)

Literature

Exploring complex networks, Steven H Strogatz, Nature 410, 268 (2001)

Network Motifs: Simple Building Blocks of Complex Networks, R. Milo et al., Science 298, 824-827 (2002)

Two-hybrid arrays, P. Uetz, Current Opinion in Chemical Biology 6, 57-62 (2001)

Transcriptional Regulatory Networks in Saccharomyces Cerevisiae, TI Lee et al., Science 298, 799-804 (2002)

Functional organization of the yeast proteome by systematic analysis of protein complexes, AC Gavin et al., Nature 415, 141 (2002)

Functional discovery via a compendium of expression profiles, TR Hughes et al., Cell 102, 109-126 (2000)

Global Mapping of the Yeast Genetic Interaction Network, AHY Tong et al., Science 303 (2004)

- **Graphs with R and Bioconductor**

▶graph, RBGL, Rgraphviz

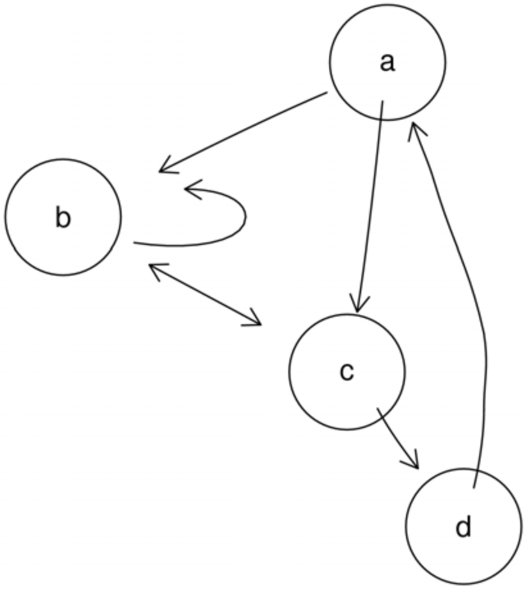
graph basic class definitions and functionality

RBGL interface to graph algorithms (e.g. shortest path, connectivity)

Rgraphviz rendering functionality
Different layout algorithms.
Node plotting, line type, color etc. can be controlled by the user.

▶ Creating our first graph

```
library(graph); library(Rgraphviz)

edges <- list(a=list(edges=2:3),
              b=list(edges=2:3),
              c=list(edges=c(2,4)),
              d=list(edges=1))

g <- new("graphNEL", nodes=letters[1:4], edgeL=edges,
            edgemode="directed")

plot(g)
```

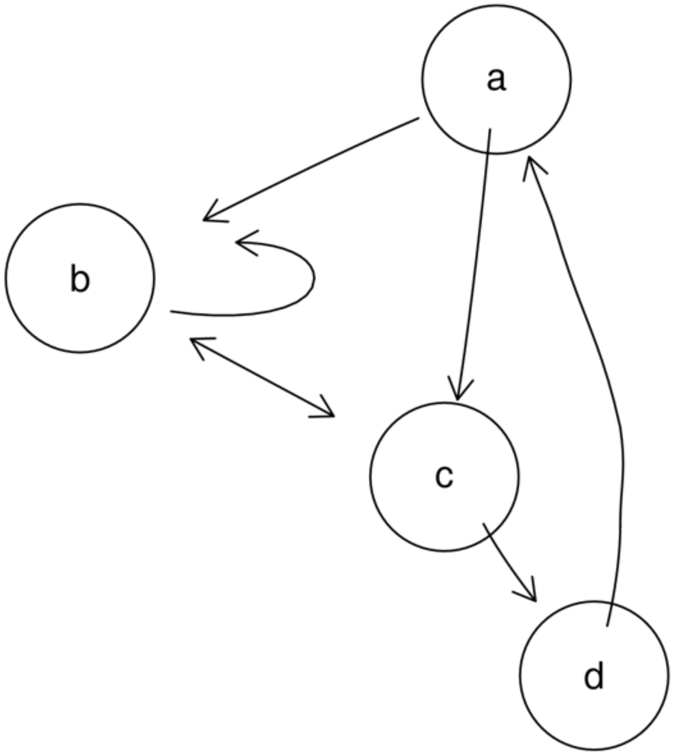


▶Querying nodes, edges, degree

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

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

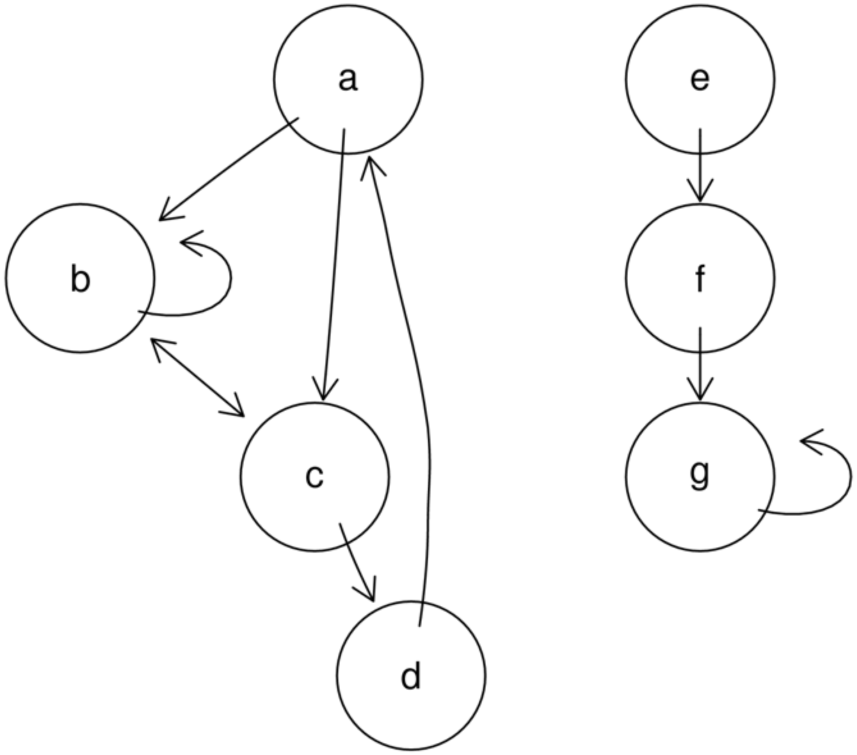
```
> degree(g)
$inDegree
a b c d
1 3 2 1
$outDegree
a b c d
2 2 2 1
```



▶Adjacent and accessible nodes

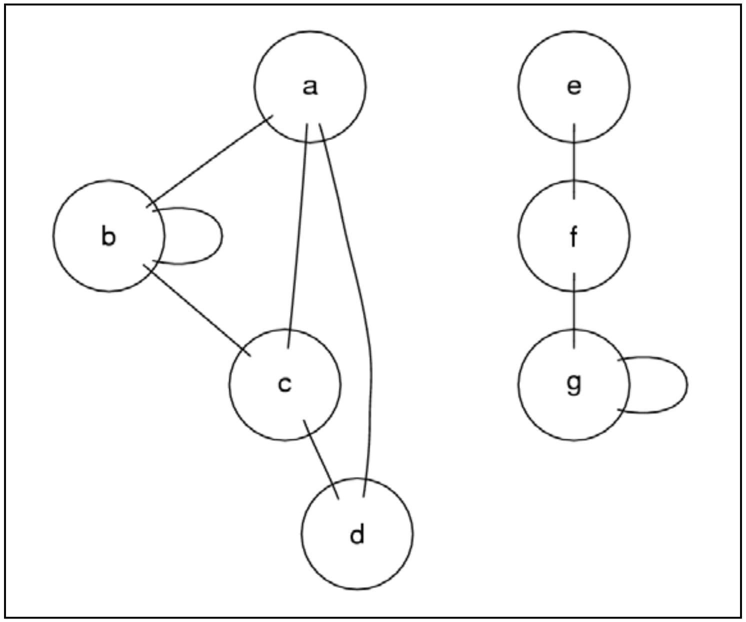
```
> adj(g, c("b", "c"))
$b
[1] "b" "c"
$c
[1] "b" "d"
```

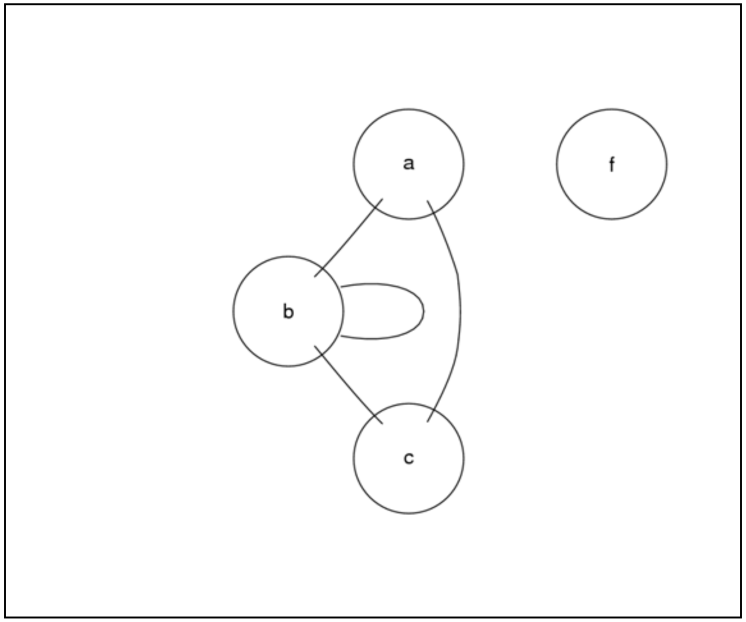
```
> acc(g, c("b", "c"))
$b
a c d
3 1 2

$c
a b d
2 1 1
```



▶Undirected graphs, subgraphs, boundary graph

```
> ug <- ugraph(g)
> plot(ug)


> sg <- subGraph(c("a", "b",
      "c", "f"), ug)
> plot(sg)
> boundary(sg, ug)
> $a
>[1] "d"
> $b
> character(0)
> $c
>[1] "d"
> $f
>[1] "e" "g"
```





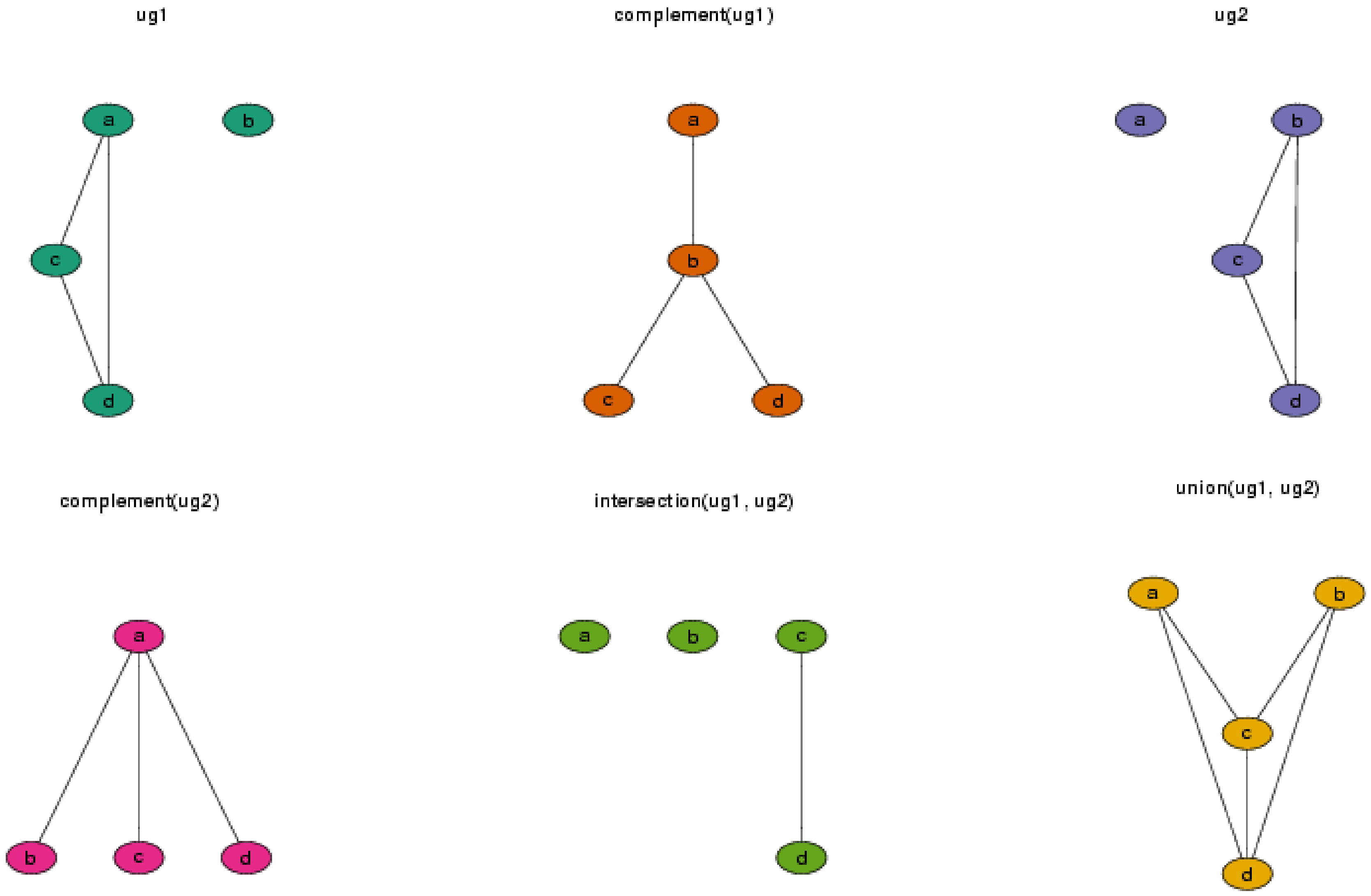
▶ Weighted graphs

```
> edges <- list(a=list(edges=2:3, weights=1:2),
+               b=list(edges=2:3, weights=c(0.5, 1)),
+               c=list(edges=c(2,4), weights=c(2:1)),
+               d=list(edges=1, weights=3))

> g <- new("graphNEL", nodes=letters[1:4],
edgeL=edges, edgemode="directed")

> edgeWeights(g)
$a
2 3
1 2
$b
  2   3
0.5 1.0
$c                    $d
2 4                   1
2 1                   3
```

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
```

▶ Graph algebra



Random graphs

Random edge graph: `randomEGraph(V, p, edges)`

`V:`	nodes
either `p:`	probability per edge
or `edges:`	number of edges

Random graph with latent factor: `randomGraph(V, M, p, weights=TRUE)`

`V:`	nodes
`M:`	latent factor
`p:`	probability

For each node, generate a logical vector of length `length(M)`, with P(TRUE)=`p`. Edges are between nodes that share >= 1 elements. Weights can be generated according to number of shared elements.

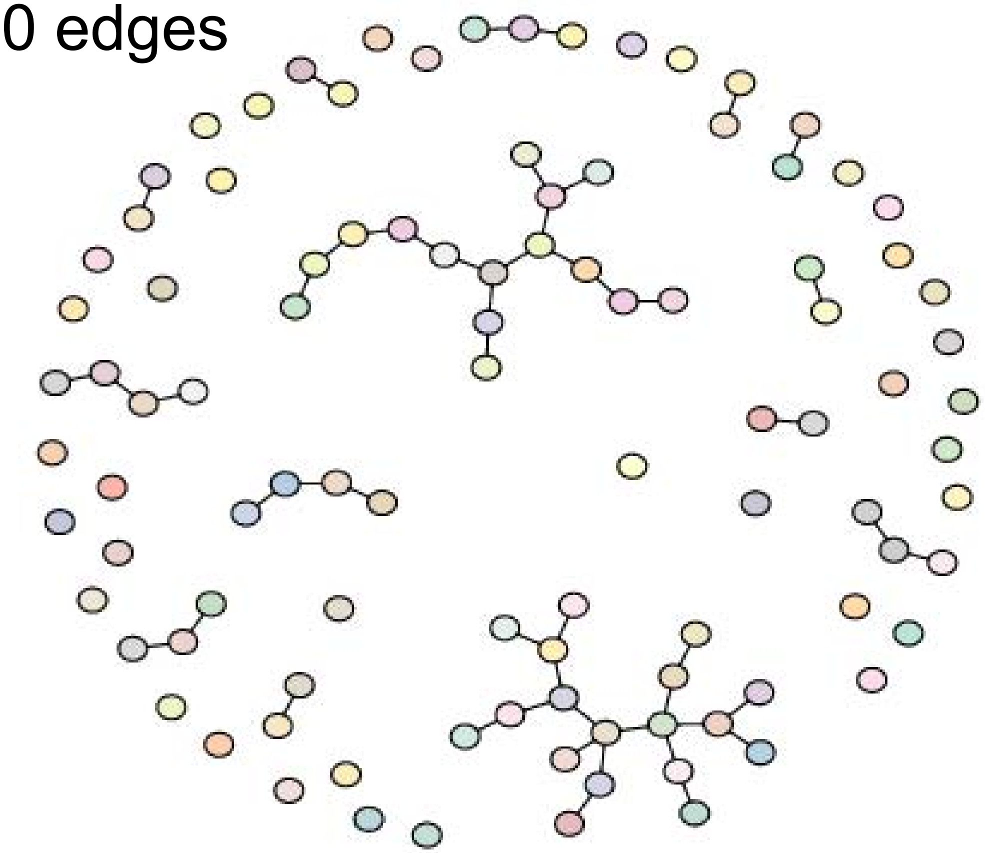
Random graph with predefined degree distribution:

`randomNodeGraph(nodeDegree)`

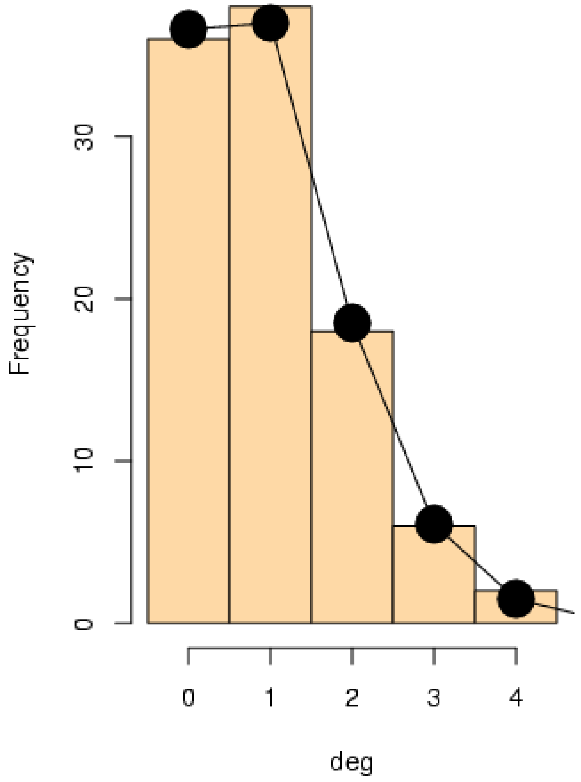
`nodeDegree:`	named integer vector sum(nodeDegree)%%2==0

▶ Random edge graph

100 nodes
50 edges



degree
distribution



▶ Graph representations

node-edge list: `graphNEL`

- list of nodes
- list of out-edges for each node

from-to matrix

adjacency matrix

adjacency matrix (sparse) `graphAM` (to come)

node list + edge list: `pNode, pEdge` (Rgraphviz)

- list of nodes
- list of edges (node pairs, possibly ordered)

`Ragraph:` representation of a laid out graph

Graph representations: from-to-matrix

```
> ft
     [,1] [,2]
[1,]    1    2
[2,]    2    3
[3,]    3    1
[4,]    4    4

> ftM2adjM(ft)
  1 2 3 4
1 0 1 0 0
2 0 0 1 0
3 1 0 0 0
4 0 0 0 1
```

▶ GXL: graph exchange language

```
<gxl>
 <graph edgemode="directed" id="G">
  <node id="A"/>
  <node id="B"/>
  <node id="C"/>
  …
  <edge id="e1" from="A" to="C">
   <attr name="weights">
    <int>1</int>
   </attr>
  </edge>
  <edge id="e2" from="B" to="D">
   <attr name="weights">
    <int>1</int>
   </attr>
  </edge>
  …
</graph>
</gxl>
```

from graph/GXL/kmstEx.gxl

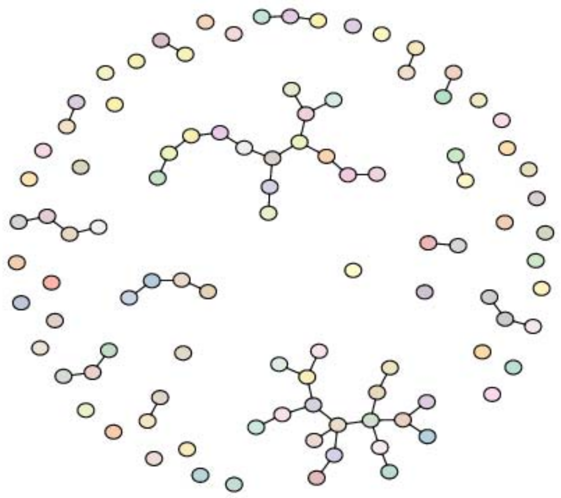
GXL (www.gupro.de/GXL) is "an XML sublanguage designed to be a standard exchange format for graphs". The graph package provides tools for im- and exporting graphs as GXL

▶RBGL: interface to the Boost Graph Library

Connected components

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

rg



Choose the largest component

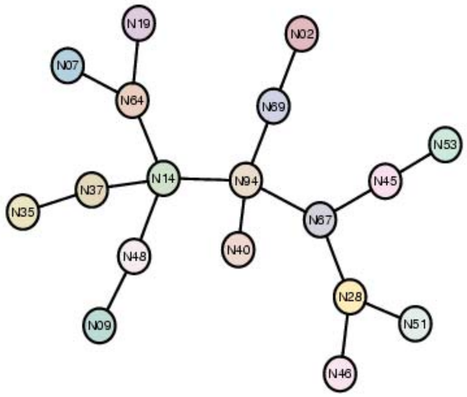
```
wh = which.max(listLen(cc))
sg = subGraph(cc[[wh]], rg)
```

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"
```

▶ depth / breadth first search

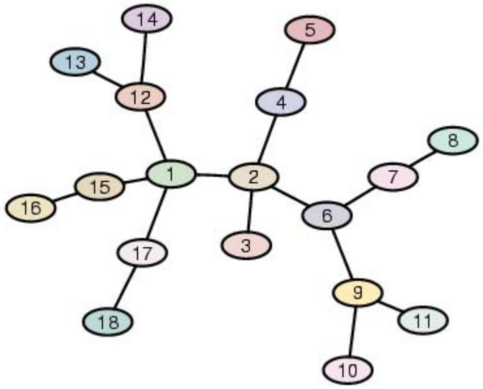
a connected subgraph



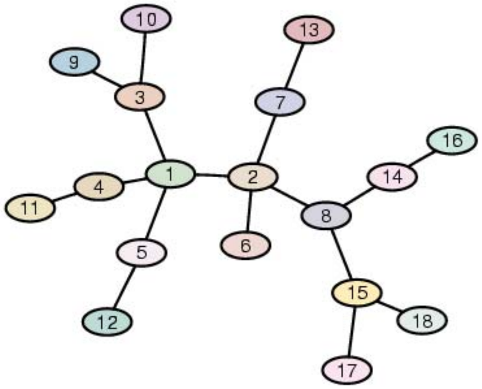
```
dfs(sg, "N14")
```

```
bfs(sg, "N14")
```

DFS

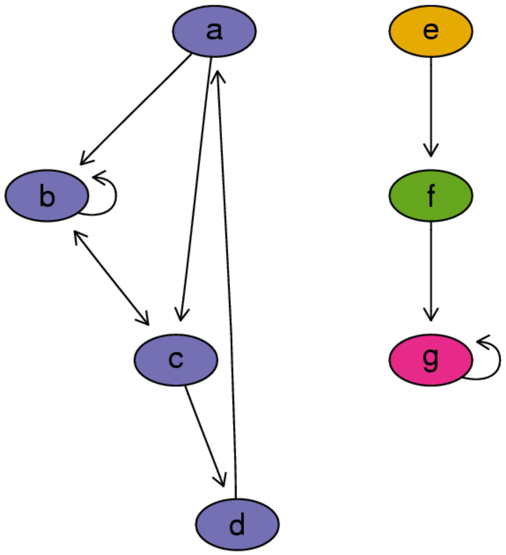


BFS

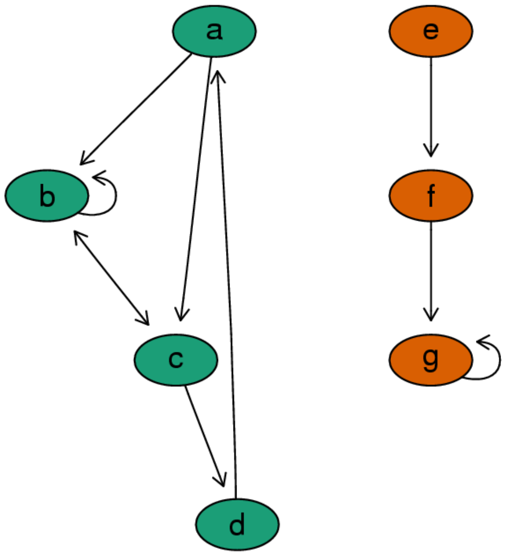


connected components

```
sc = strongComp(g2)

nattrs = makeNodeAttrs(g2,
             fillcolor="")

for(i in 1:length(sc))
  nattrs$fillcolor[sc[[i]]] =
              myColors[i]

plot(g2, "dot", nodeAttrs=nattrs)
```

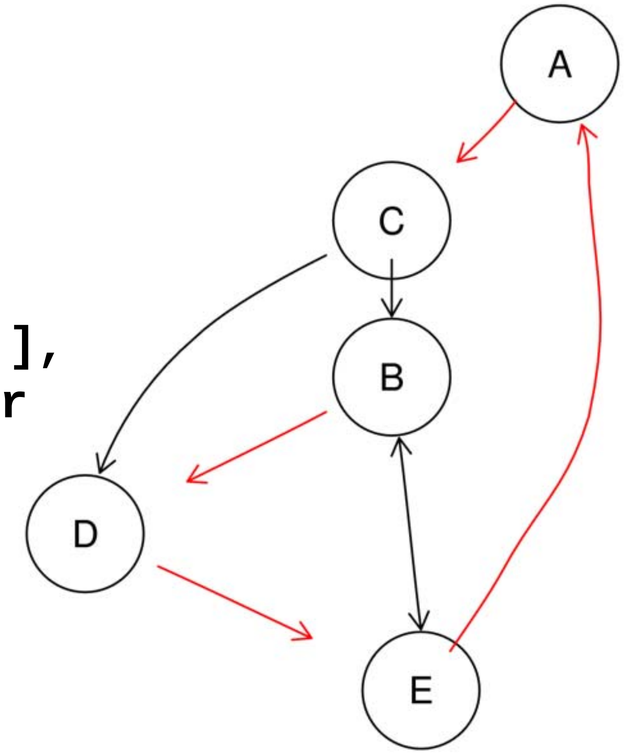


```
wc = connComp(g2)
```



▶minimal spanning tree

```
km <-
fromGXL(file(system.file("GXL/kmstEx
.gxl", package = "graph")))

ms <- mstree.kruskal(km)

e <- buildEdgeList(km)
n <- buildNodeList(km)

for(i in 1:ncol(ms$edgeList))

e[[paste(ms$nodes[ms$edgeList[,i]],
       collapse="~")]]@attrs$color
              <- "red"

z <- agopen(nodes=n, edges=e,
edgeMode="directed", name="")

plot(z)
```



shortest path algorithms

Different algorithms for different types of graphs

- o all edge weights the same
- o positive edge weights
- o real numbers

…and different settings of the problem

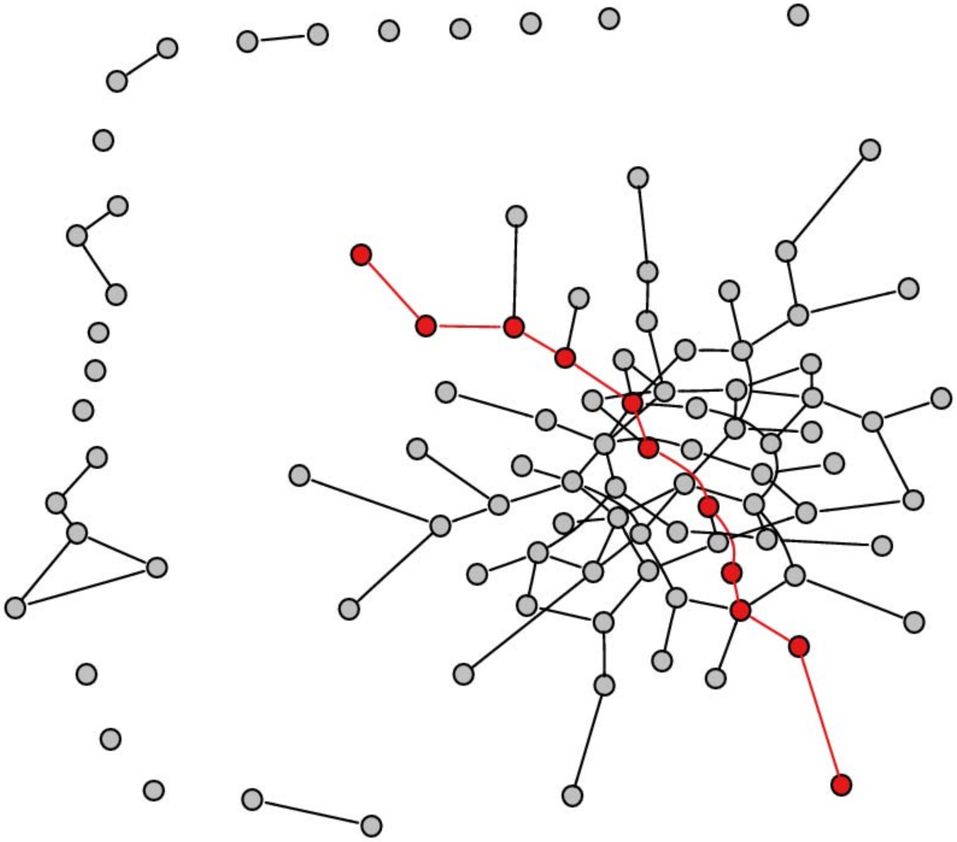
- o single pair
- o single source
- o single destination
- o all pairs

Functions

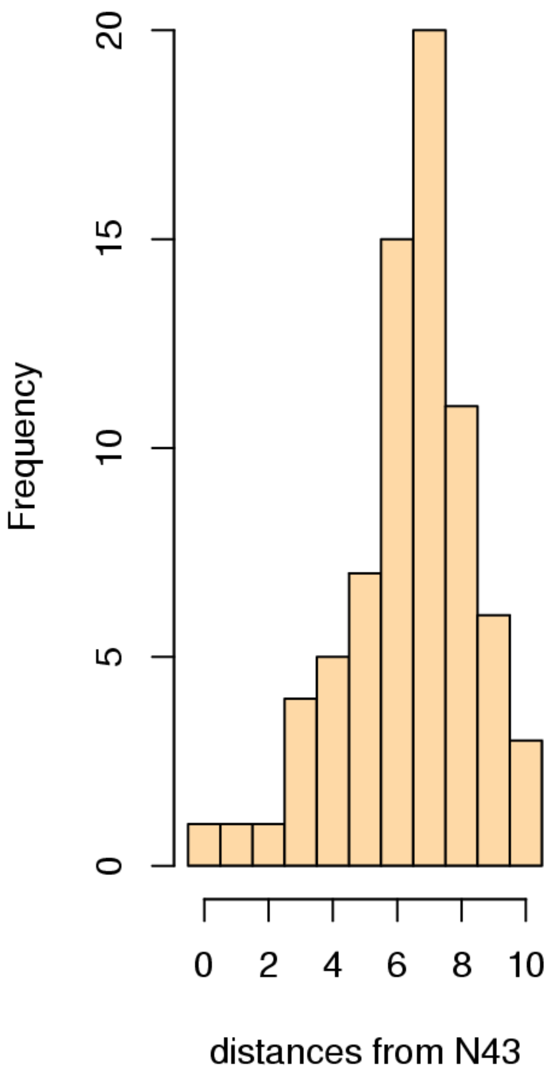
```
bfs
dijkstra.sp
sp.between
johnson.all.pairs.sp
```

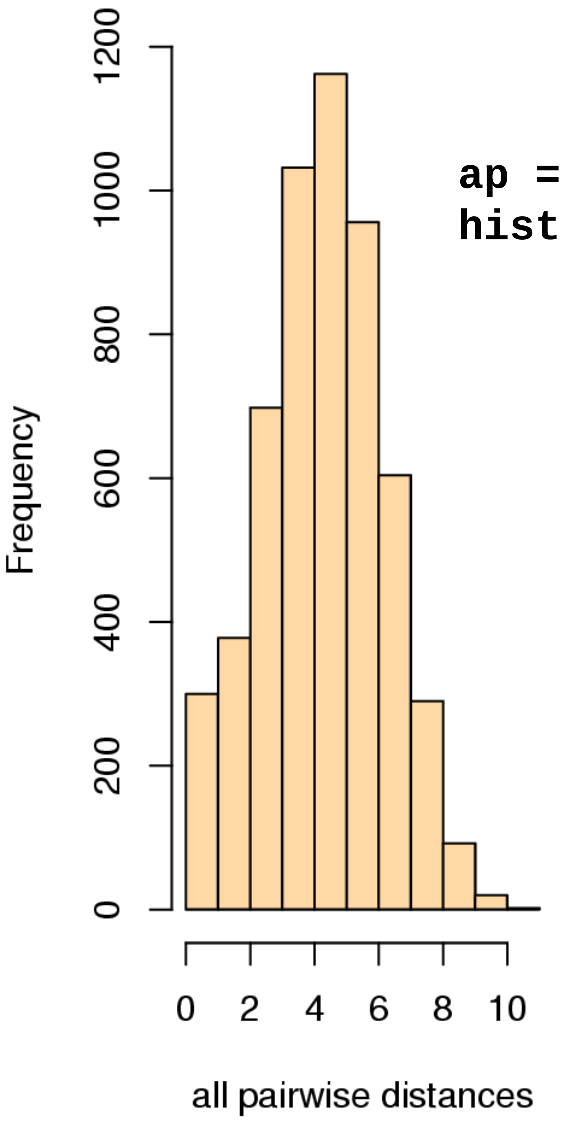
▶ shortest path

```
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
```



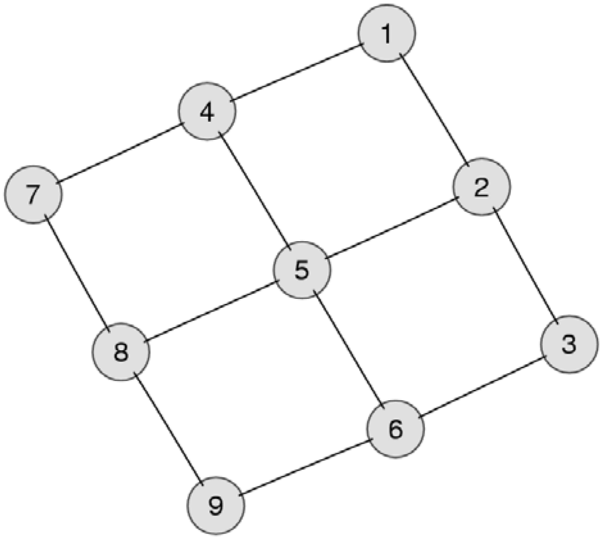
shortest path



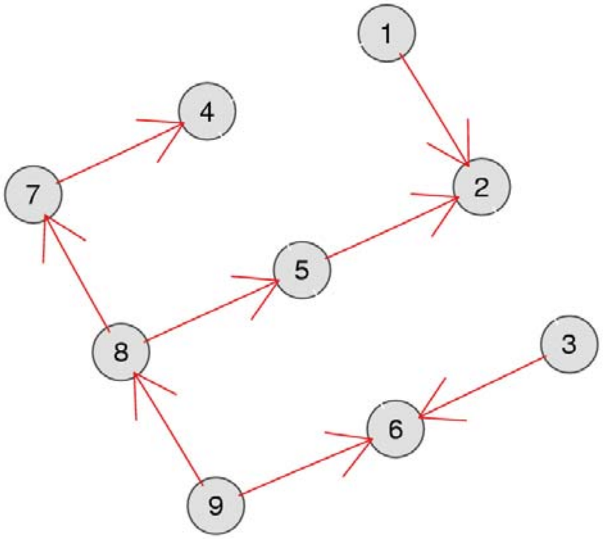


```
ap = johnson.all.pairs.sp(rg2)
hist(ap)
```

▶minimal spanning tree



```
gr
```

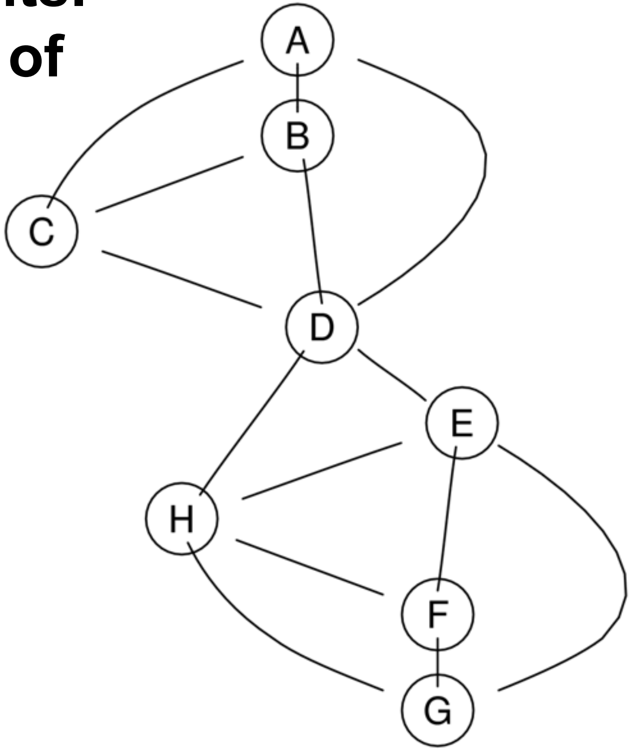


```
mst = mstree.kruskal(gr)
```

▶ connectivity

Consider graph g with single connected component.
Edge connectivity of g: minimum number of edges in g that can be cut to produce a graph with two components.
Minimum disconnecting set: the set of edges in this cut.

```
> edgeConnectivity(g)
$connectivity
[1] 2

$minDisconSet
$minDisconSet[[1]]
[1] "D" "E"

$minDisconSet[[2]]
[1] "D" "H"
```



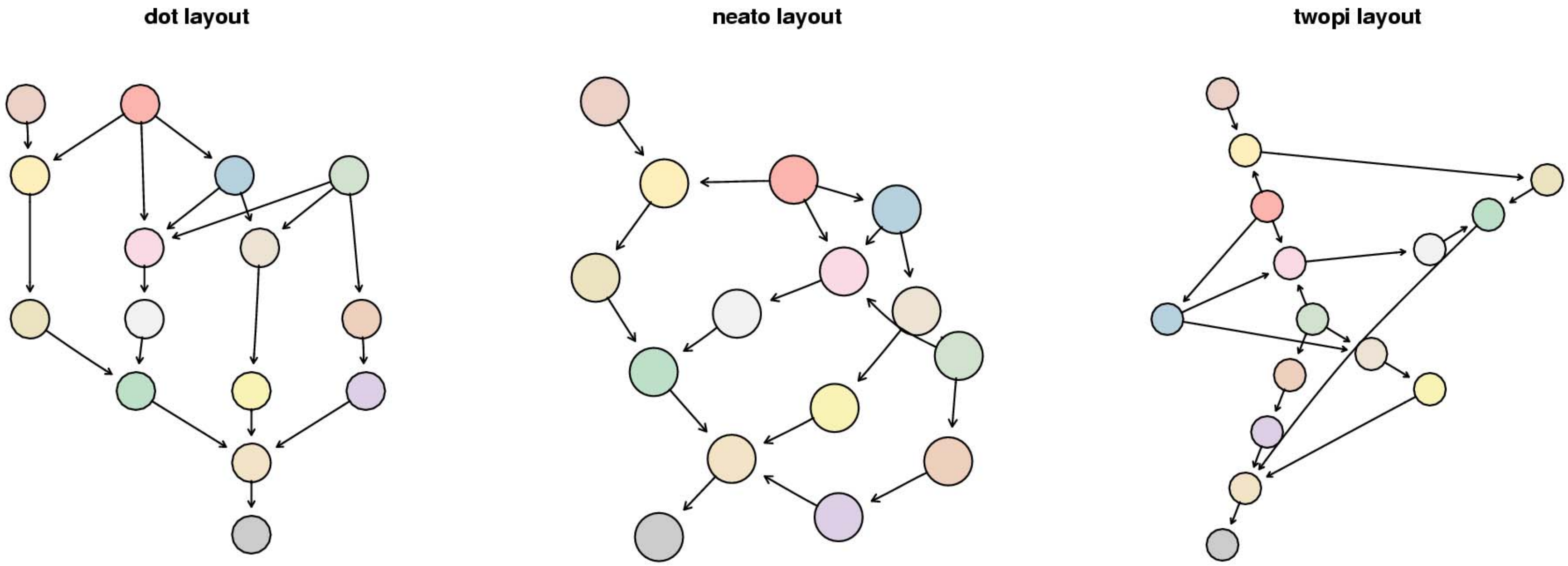
▶Rgraphviz: the different layout engines

dot: directed graphs. Works best on DAGs and other graphs that can be drawn as hierarchies.

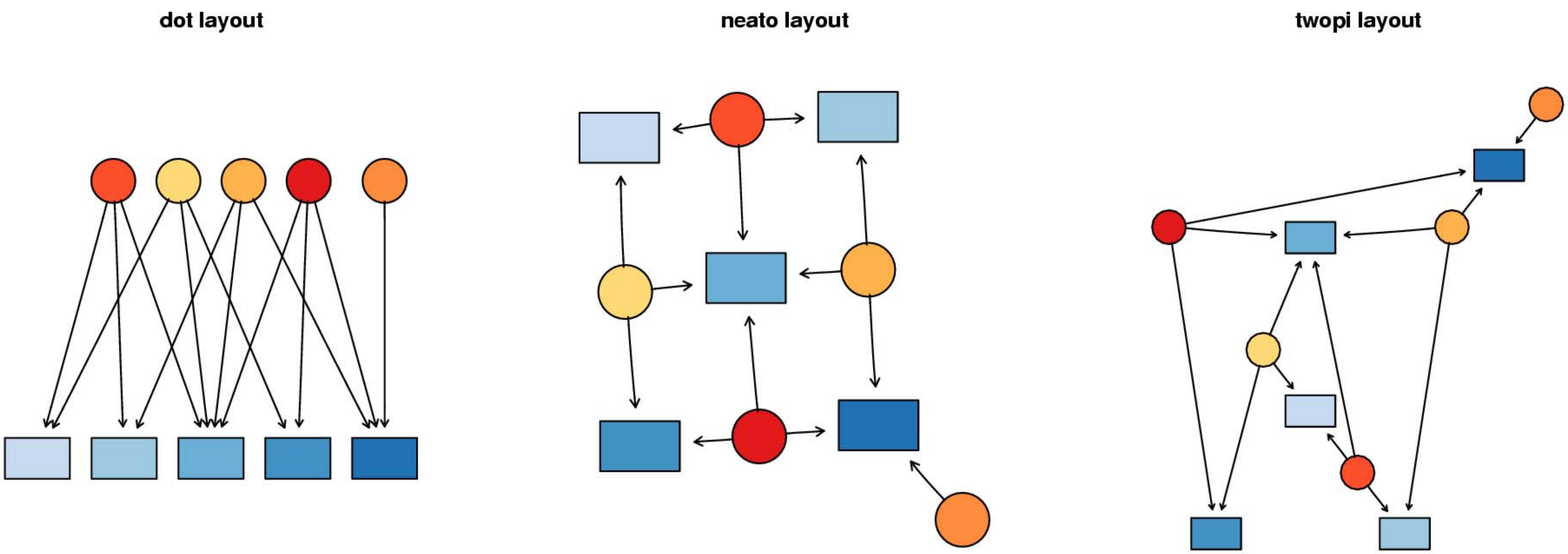
neato: undirected graphs using ’spring’ models

twopi: radial layout. One node (‘root’) chosen as the center. Remaining nodes on a sequence of concentric circles about the origin, with radial distance proportional to graph distance. Root can be specified or chosen heuristically.

Rgraphviz: the different layout engines



Rgraphviz: the different layout engines

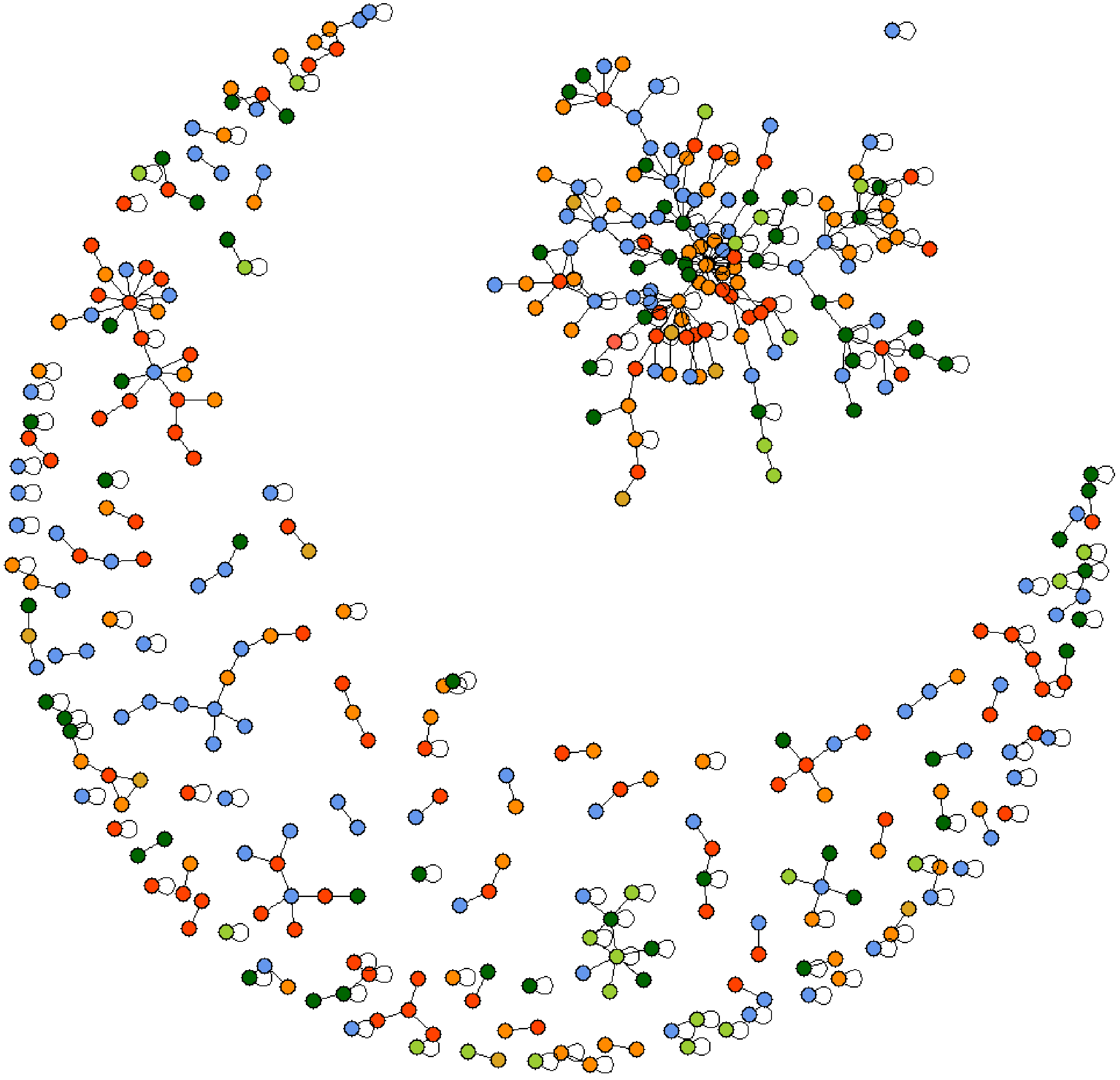


dot layout

neato layout

twopi layout

▶ domain combination graph



ImageMap

```
lg = agopen(g, …)

imageMap(lg,
  con=file("imca-frame1.html", open="w")
  tags= list(HREF   = href,
             TITLE  = title,
             TARGET = rep("frame2", length(AgNode(nag)))),
  imgname=fpng, width=imw, height=imh)
```

▶ Acknowledgements

R project: R-core team
www.r-project.org

Bioconductor project: Robert Gentleman, Vince Carey,
Jeff Gentry, and many others
www.bioconductor.org

graphviz project: Emden Gansner, Stephen North,
Yehuda Koren (AT&T Research)
www.graphviz.org

Boost graph library: Jeremy Siek, Lie-Quan Lee,
Andrew Lumsdaine, Indiana University
www.boost.org/libs/graph/doc

▶References

Can a biologist fix a radio? Y. Lazebnik, *Cancer Cell* 2:179 (2002)

Social Network Analysis, Methods and Applications. S. Wasserman and K. Faust, Cambridge University Press (1994)