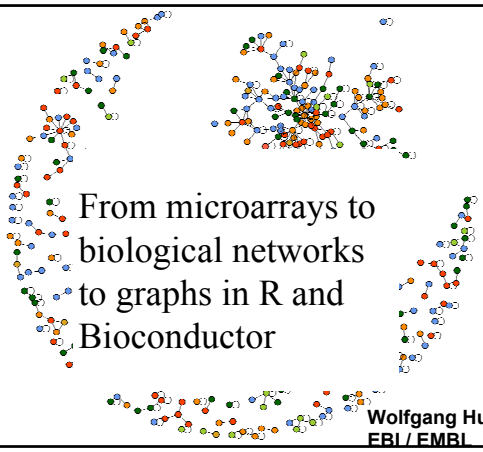


From microarrays to biological networks to graphs in R and Bioconductor

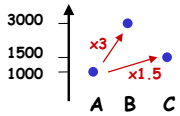
Wolfgang Huber, EBI / EMBL

recap "generalized" or "regularized" log-ratios

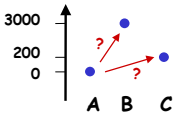
(on popular request)

What are the measurement units of gene expression?

We use *fold changes* to describe continuous changes in expression



But what if the gene is "off" (or below detection limit) in one condition?



ratios and fold changes

The idea of the log-ratio (base 2)

- **0: no change**
- **+1: up by factor of $2^1 = 2$**
- **+2: up by factor of $2^2 = 4$**
- **-1: down by factor of $2^{-1} = 1/2$**
- **-2: down by factor of $2^{-2} = 1/4$**

A unit for measuring changes in expression: assumes that a change from 1000 to 2000 units has a similar biological meaning to one from 5000 to 10000.

What about a change from 0 to 500?
- conceptually
- noise, measurement precision

Sources of variation

amount of RNA in the biopsy
efficiencies of
-RNA extraction
-reverse transcription
-labeling
-photodetection

PCR yield
DNA quality
spotting efficiency,
spot size
cross-/unspecific hybridization
stray signal

Systematic

- similar effect on many measurements
- corrections can be estimated from data

Stochastic

- too random to be explicitely accounted for
- "noise"

Calibration

Error model

A simple mathematical model

measured intensity = offset + gain × true abundance

$$y_{ik} = a_{ik} + b_{ik} x_k$$

$a_{ik} = a_i + \varepsilon_{ik}$

a_i per-sample offset

$\varepsilon_{ik} \sim N(0, b_i^2 s_1^2)$
"additive noise"

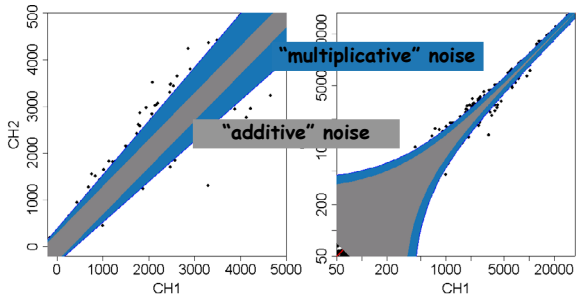
$b_{ik} = b_i \; b_k \; \exp(\eta_{ik})$

b_i per-sample normalization factor

b_k sequence-wise probe efficiency

$\eta_{ik} \sim N(0, s_2^2)$
"multiplicative noise"

The two-component model



B. Durbin, D. Rocke, JCB 2001

variance stabilization

X_u a family of random variables with $EX_u = u$, $\mathrm{Var}X_u = v(u)$.

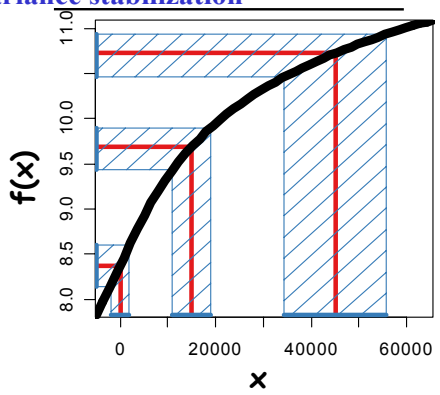
Define

$$f(x) = \int^{x} \frac{1}{\sqrt{v(u)}}\,du$$

$\Rightarrow$ var $f(X_u) \approx$ independent of u

derivation: linear approximation

variance stabilization



variance stabilizing transformations

$$f(x) = \int^{x} \frac{1}{\sqrt{v(u)}}\,du$$

1.) constant variance $\quad v(u) = \text{const} \Rightarrow f \propto u$

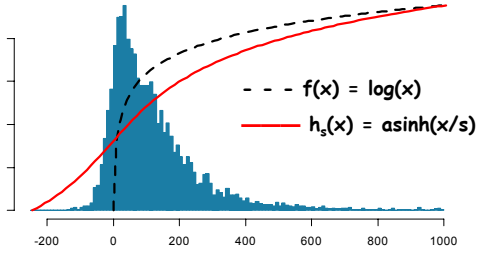
2.) const. coeff. of variation $\quad v(u) \propto u^2 \Rightarrow f \propto \log u$

3.) offset $\quad v(u) \propto (u + u_0)^2 \Rightarrow f \propto \log(u + u_0)$

4.) microarray

$$v(u) \propto (u + u_0)^2 + s^2 \Rightarrow f \propto \operatorname{arsinh}\frac{u + u_0}{s}$$

the "glog" transformation



$$\operatorname{arsinh}(x) = \log\left(x + \sqrt{x^2 + 1}\right)$$

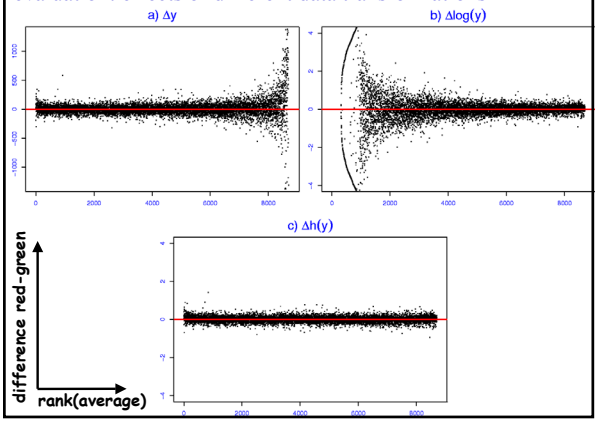
$$\lim_{x \to \infty} (\operatorname{arsinh} x - \log x - \log 2) = 0$$

P. Munson, 2001

D. Rocke & B. Durbin, ISMB 2002

W. Huber et al., ISMB 2002

evaluation: effects of different data transformations



from proof-of principle to a useful tool and building block: technical issues

$$\operatorname{arsinh}\frac{y_{ki} - a_i}{b_i} = \mu_k + \varepsilon_{ki}, \quad \varepsilon_{ki} : N(0, c^2)$$

- **choice of parameterization:** global background vs stratification by print-tip, sequence, etc.
- **estimation: maximum likelihood** is straightforward – but off-the shelf it is too sensitive to **outliers.**
- **robust** variant of ML, à la **Least Trimmed Sum of Squares** regression: works as long as <50% of genes are differentially transcribed
- **iterative numerical optimization** - likelihood is concave
- **software needs to produce correct results (or meaningful error message) with default settings for all sorts of arrays, labs, experiments**

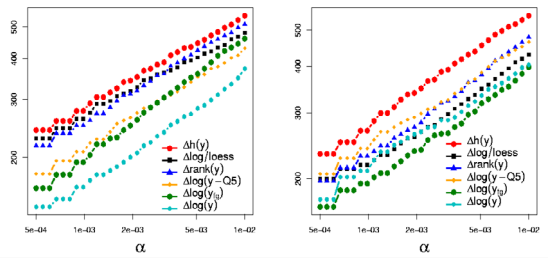
What is the bottomline?

Detecting differentially transcribed genes from cDNA array data

- **Data:** paired tumor/normal tissue from 19 kidney cancers, in color flip duplicates on 38 cDNA slides à 4000 genes.
- 6 different strategies for **normalization** and quantification of differential abundance
- Calculate for each gene & each method: *t*-statistics, **permutation-*p***
- For threshold α, **compare** the number of genes the different methods find, $\#\{p_i \mid p_i \le \alpha\}$

evaluation

one-sided test for up **one-sided test for down**

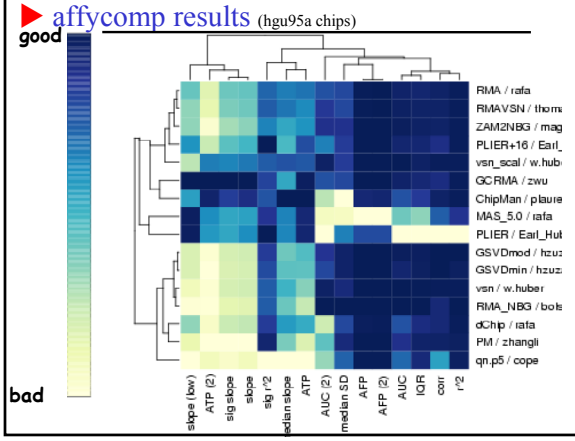


more accurate quantification of differential expression ⇒ higher sensitivity / specificity

Another evaluation: affycomp, a benchmark for Affymetrix genechip expression measures

- **Data:**
  Spike-in series: from **Affymetrix** 59 x HGU95A, 16 genes, 14 concentrations, complex background
  Dilution series: from **GeneLogic** 60 x HGU95Av2, liver & CNS cRNA in different proportions and amounts
- **Benchmark:**
  15 quality measures regarding
  -reproducibility
  -sensitivity
  -specificity
  Put together by Rafael Irizarry (Johns Hopkins)
  http://affycomp.biostat.jhsph.edu

affycomp results (hgu95a chips)



ROC curves

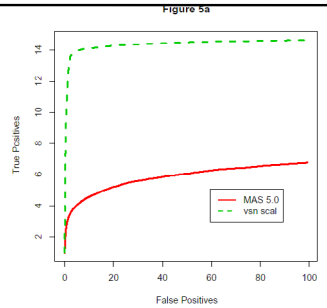


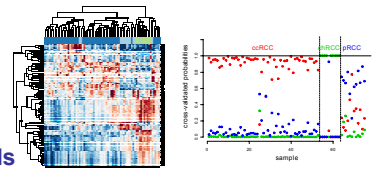
Figure 5a): A typical identification rule for differential expression filters genes with fold change exceeding a given threshold. This figure shows average ROC curves which offer a graphical representation of both specificity and sensitivity for such a detection rule.Average ROC curves based on comparisons with nominal fold changes ranging from 2 to 4096. b) As a) but with nominal fold changes equal to 2.

Availability

Package vsn in Bioconductor:

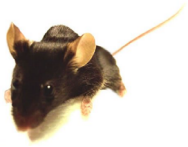
Preprocessing of two-color, Agilent, and Affymetrix arrays

Candidate gene sets from microarray studies:
dozens…hundreds



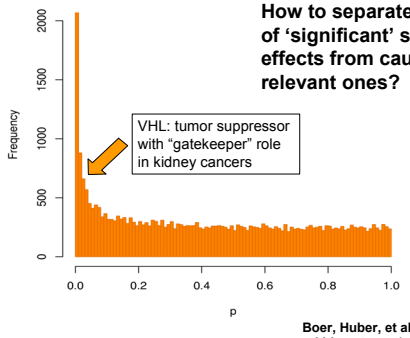
How to close the gap?

Capacity of detailed in-vivo functional studies:
one…few



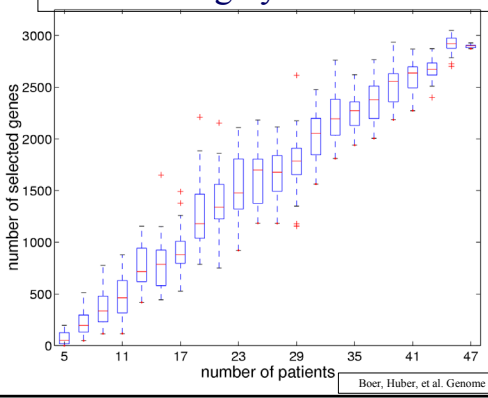
Drowning by numbers

How to separate a flood of 'significant' secondary effects from causally relevant ones?



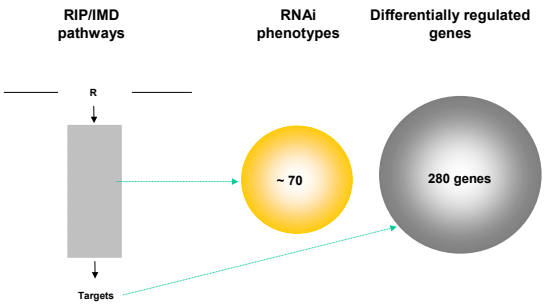
Boer, Huber, et al. Genome Res. 2001: kidney tumor/normal profiling study

Drowning by numbers



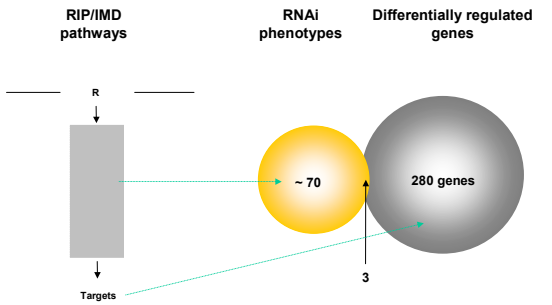
Boer, Huber, et al. Genome Res. 2001

Is differential expression a good predictor for 'signaling' function?



Michael Boutros

Most pathway targets are not required for pathway function



Michael Boutros

Buffering

- in yeast, ~73% of gene deletions are "non-essential" (Glaever et al. Nature 418 (2002))
- in Drosophila cell lines, only 5% show viability phenotype (Boutros et al. Science 303 (2004))
- association studies for most human genetic diseases have failed to produce single loci with high penetrance
- evolutionary pressure for robustness

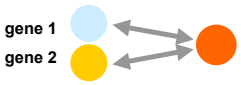
What are the implications for functional studies?

Need to:

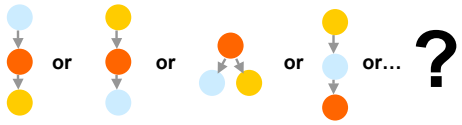
use combinatorial perturbations
observe multiple phenotypes with high sensitivity
understand gene-gene and gene-phenotype interactions in terms of graph-like models ("networks")

From association to intervention

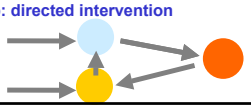
mRNA profiling studies: association of genes with diseases



the dilemma



the next step: directed intervention



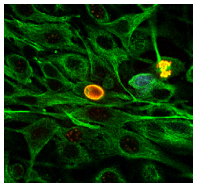
Interference/Perturbation tools

RNAi

+ genome wide
o specificity
- efficiency / monitoring?

Transfection (expression)

+ 100% specific
+ monitoring
- library size



Small compounds

...

Monitoring tools

Plate reader

96 or 384 well, 1…4 measurements per well

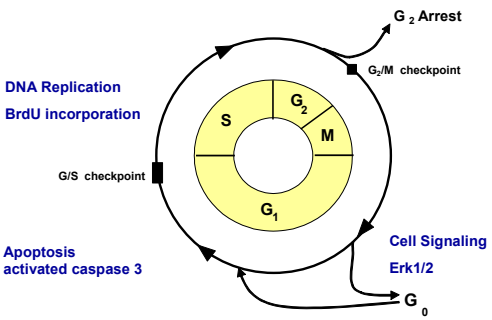
FACS

4…8 measurements per cell, thousands of cells per well

Automated Microscopy

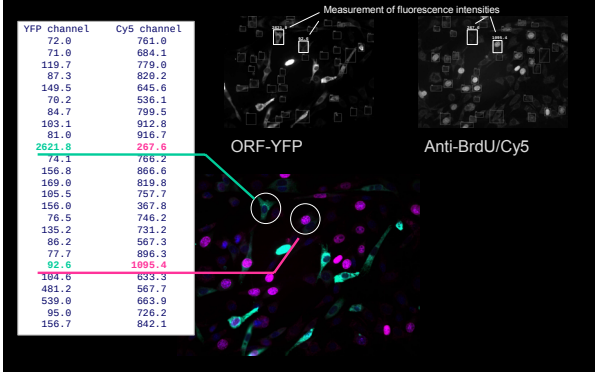
unlimited

Cell-Assays to Challenge the Cell-Cycle



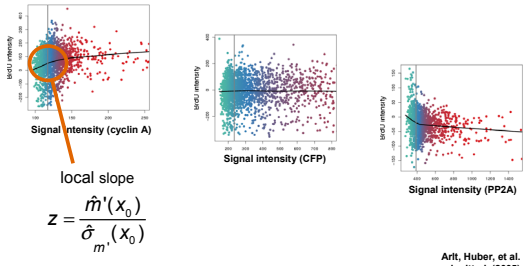
Dorit Arlt, DKFZ

Proliferation Assay



Local Regression analysis

… focus on small perturbations and weak phenotypes!

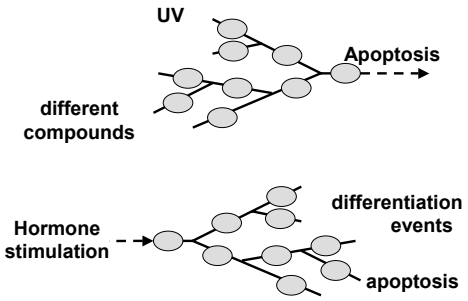


local slope

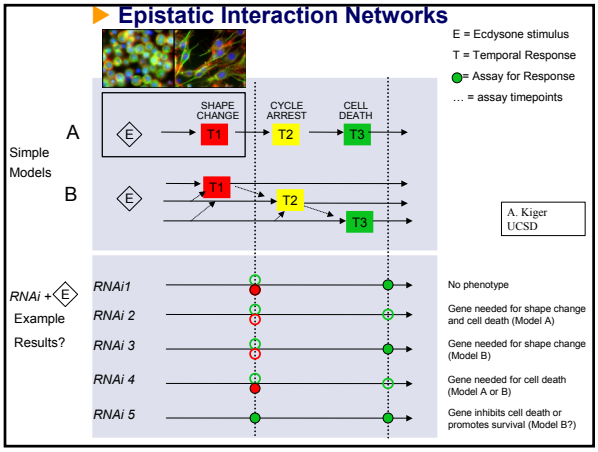
$$z = \frac{\hat{m}'(x_0)}{\hat{\sigma}_{m'}(x_0)}$$

Arlt, Huber, et al. submitted (2005)

Epistatic Interaction Networks



Epistatic Interaction Networks



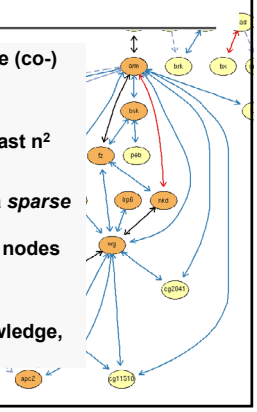
Graphical Models (a.k.a. Bayesian networks) to model genetic interactions

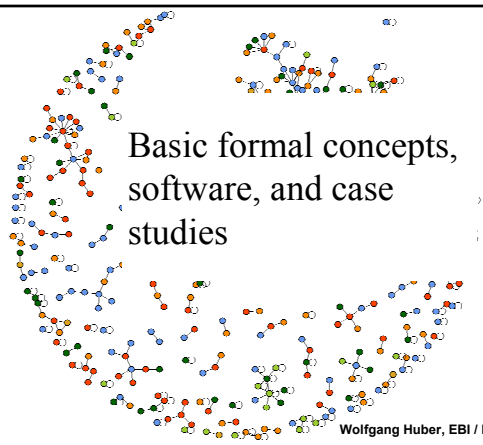
Probability distribution: models the (co-) occurence of a set of events

Problem: to infer just the pairwise correlation of n objects, need at least n^2 datapoints; for tripletts n^3, …

Solution: Graphical model - uses a *sparse* graph to construct this probability distribution (graph: a collection of nodes and edges)

Price: need to make simplifying assumptions - integrate prior knowledge, other data





Definitions

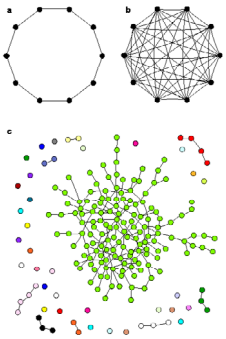
Graph := set of nodes + set of edges
Edge := pair of nodes

Edges can be
- directed
- undirected
- weighted, typed

special cases: cycles, acyclic graphs, trees

Network topologies

regular

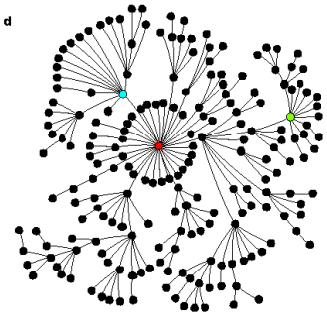


all-to-all

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

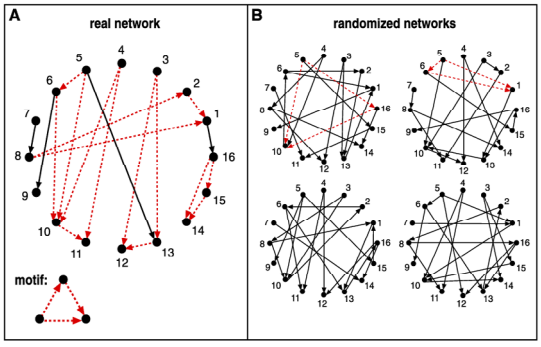
Starting point: graph with directed edges

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

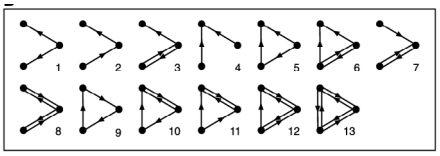
Compare to randomized networks

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

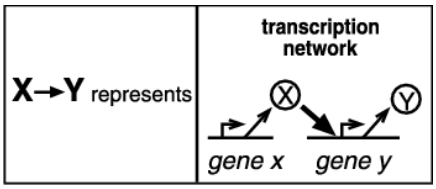
Schematic view of motif detection



All 3-node connected subgraphs



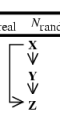
Transcription networks

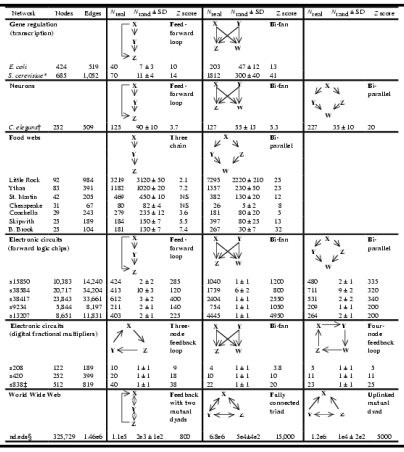


Nodes = transcription factors

Directed edge: X regulates transcription of Y

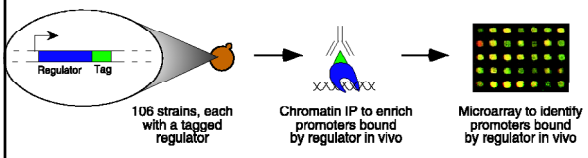
3- and 4-node motifs in transcription networks

Network	Nodes	Edges	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	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



Transcriptional regulatory networks from "genome-wide location analysis"

A

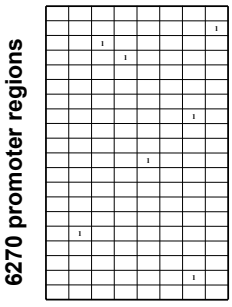


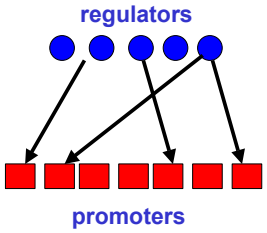
regulator := a transcription factor (TF) or a ligand of a TF
tag: c-myc epitope

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

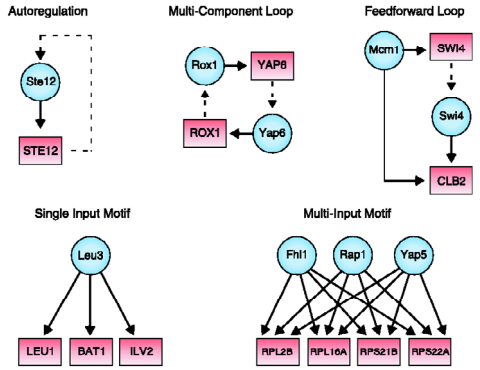
Transcriptional regulatory networks bipartite graph

106 regulators (TFs)

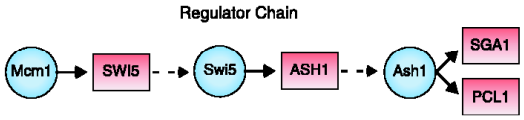




Network motifs



Network motifs



Graphs with R and Bioconductor

graph, RBGL, Rgraphviz

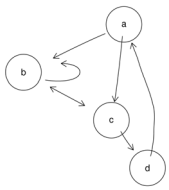
graph basic class definitions and functionality

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.

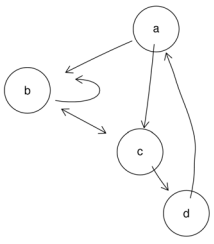
Creating our first graph

```
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)
```



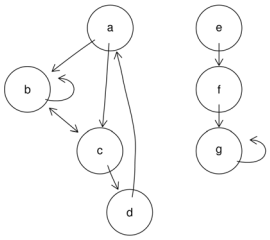
Querying nodes, edges, degree

```
> 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
```



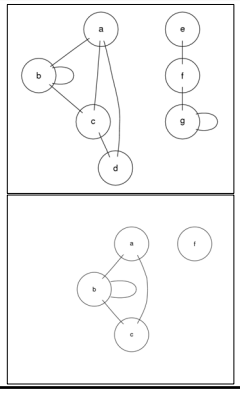
Adjacent and accessible nodes

```
> 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"
```



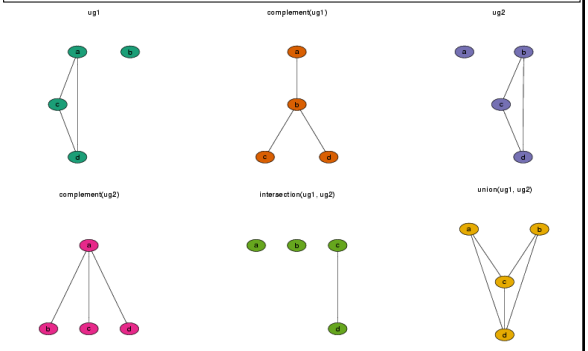
Weighted graphs

```
> 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

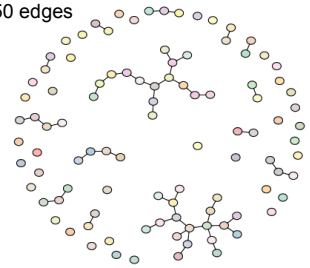
Random edge graph: `randomEGraph(V, p, edges)`
`V:` nodes
either `p:` probability per edge
or `edges:` number of edges

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.

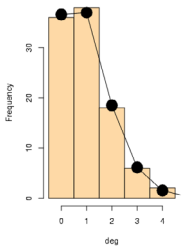
Random graph with predefined degree distribution:
`randomNodeGraph(nodeDegree)`
`nodeDegree:` named integer vector
sum(nodeDegree)%%2==0

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

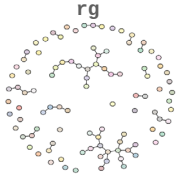
GXL (www.gupro.de/GXL) is "an XML sublanguage designed to be a standard exchange format for graphs". The graph package provides tools for im- and exporting graphs as GXL

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")`

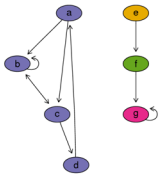
`bfs(sg, "N14")`

DFS

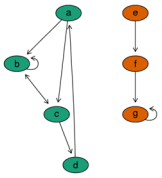
BFS

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)
```

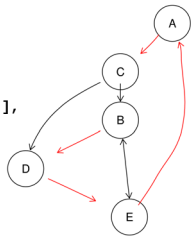



```
wc = connComp(g2)
```



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

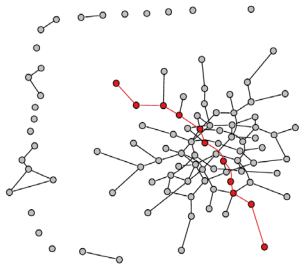
…and different settings of the problem
- single pair
- single source
- single destination
- all pairs

Functions

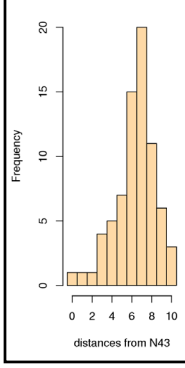
```
bfs
dijkstra.sp
sp.between
johnson.all.pairs.sp
```

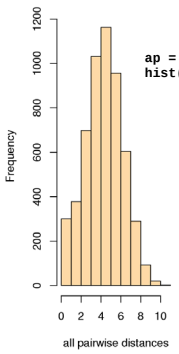
shortest path

```
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
```



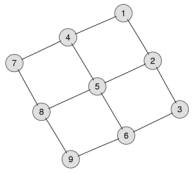
shortest path



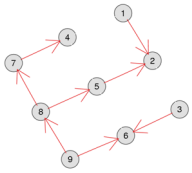



```
ap = johnson.all.pairs.sp(rg2)
hist(ap)
```

minimal spanning tree



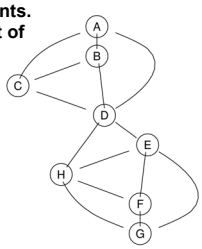
gr




```
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"
```

RBGL functions	Comments
Traversals	
bfs	BFS
dfs	DFS
Shortest paths	
dijkstra.sp	single-source, nonnegative weights
bellman.ford.sp	single source, general weights
dag.sp	single-source, DAG
johnson.all.pairs.sp	returns distance matrix
Minimal spanning trees	
mstree.kruskal	returns edge list and weights
prim.mstT	as above
Connectivity	
connectedComp	returns list of node-sets
strongComp	as above
edgeConnectivity	returns index and minimum disconnecting set
init.incremental.components	special processing for evolving graphs
incremental.components	
same.component	boolean in the incremental setting
Maximum flow algorithms	
edmunds.karp.max.flow	list of max flow, and edge specific flows
push.relabel.max.flow	
Vertex ordering	
tsort	topological sort
cuthill.mckee.ordering	reduces bandwidth
sloan.ordering	reduces wavefront
min.degree.ordering	heuristic
Other functions	
transitive.closure	returns from-to matrix
isomorphism	boolean
brandes.betweenness.centrality	indices and dominance measure
circle.layout	returns vertex coordinates
kamada.kawai.spring.layout	returns vertex coordinates

Table 21.1. Names of key functions in RBGL. Working examples for all functions are provided in the package manual pages.

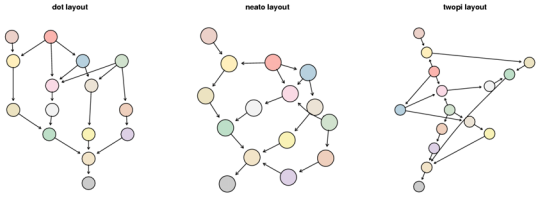
Rgraphviz: the different layout engines

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

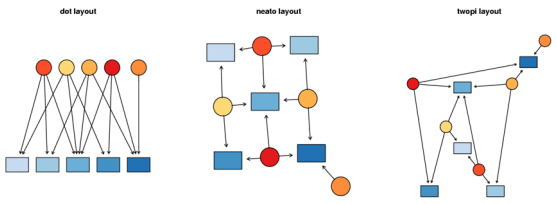
neato: undirected graphs using 'spring' models

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.

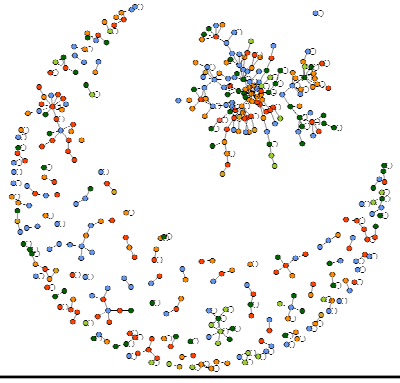
Rgraphviz: the different layout engines



Rgraphviz: the different layout engines



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)
```

Show drosophila interaction network example

Using GO to interprete gene lists

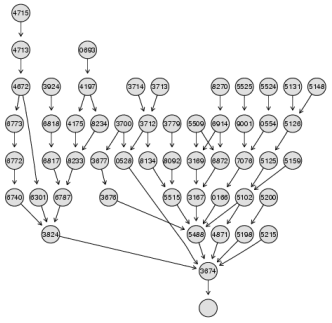
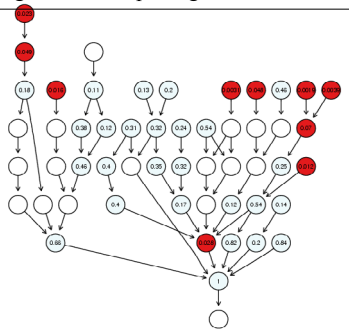


Figure 22.4. The induced GO graph for the selected genes, truncated GO identifiers are used as labels.

Using GO to interprete gene lists



Packages: Gostats, Rgraphviz

Figure 22.5. The induced GO graph colored according to unadjusted Hypergeometric *p*-values, whose values are given in the nodes.

A pathway graph

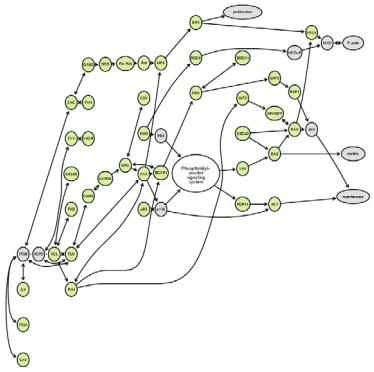


Figure 22.7. The integrin mediated cell adhesion network.

A pathway graph

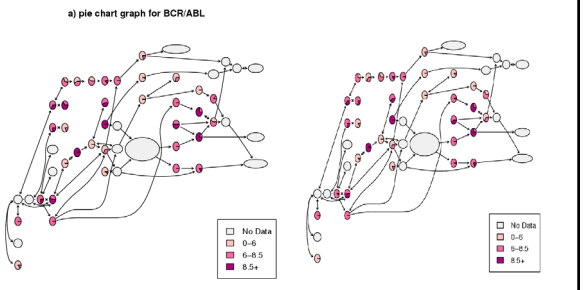
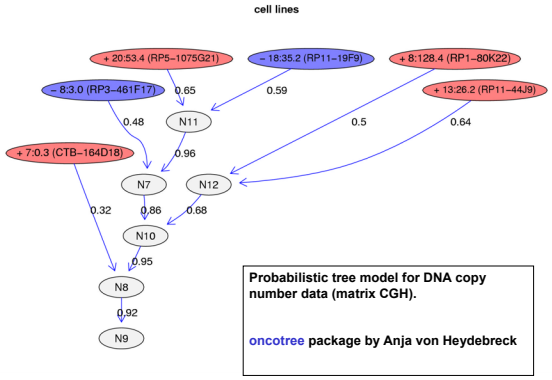


Figure 22.8. Pie chart graphs representing gene expression data for a) BCR/ABL samples, b) NEG samples.



Probabilistic tree model for DNA copy number data (matrix CGH).

oncotree package by Anja von Heydebreck

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)

Bioinformatics and Computational Biology Solutions using R and Bioconductor. R. Gentleman, V. Carey, W. Huber, R. Irizarry, S. Dudoit. Springer, available in summer 2005.