Inferring molecular regulatory networks with qpgraph

Robert Castelo
(robert.castelo@upf.edu)

Universitat Pompeu Fabra
Barcelona, Spain

BioC 2012 - Seattle, WA

Acknowledgments

Alberto Roverato, PhD
Università di Bologna, Italy

Inma Tur, MSc
Universitat Pompeu Fabra

Sonja Hänzelmann, MSc
Hospital del Mar Medical Research Institute
Barcelona, Spain

Bioconductor project

Funding
Spanish MINECO grant TIN2011-22826

Increasing interest in biological networks

OPEN ACCESS Freely available online

PLoS BIOLOGY

Stitching together Multiple Data Dimensions Reveals Interacting Metabolomic and Transcriptomic Networks That Modulate Cell Regulation

Jun Zhu[1*], Pavel Sova[2], Qiuwei Xu[3], Kenneth M. Dombek[2], Ethan Y. Xu[3], Heather Vu[3], Zhidong Tu[4], Rachel B. Brem[5], Roger E. Bumgarner[2], Eric E. Schadt[6*]

LETTER

doi:10.1038/nature09386

A *trans*-acting locus regulates an anti-viral expression network and type 1 diabetes risk

Matthias Heinig[1,2*], Enrico Petretto[3,4*], Chris Wallace[5], Leonardo Bottolo[3,4], Maxime Rotival[6], Han Lu[3], Yoyo Li[3], R
Sarah R. Langley[3], Anja Bauerfeind[1], Oliver Hummel[1], Young-Ae Lee[1,7], Svetlana Paskas[1], Carola Rintisch[1], Kathr
Jason Cooper[5], Rachel Buchan[3], Elizabeth E. Gray[8], Jason G. Cyster[8], Cardiogenics Consortium†, Jeanette Erdma

nature Vol 463|21 January 2010|doi:10.1038

Open Network Biology

Home | Authors | Reviewers | About thi

ARTICLES

The transcriptional network for mesenchymal transformation of brain tumours

Maria Stella Carro[1*†], Wei Keat Lim[2,3*†], Mariano Javier Alvarez[3,4*], Robert J. Bollo[8], Xudong Zhao[1], Evan Y. Snyder[9], Erik P. Sulman[10], Sandrine L. Anne[1†], Fiona Doetsch[5], Howard Colman[11], Anna Lasore, Ken Aldape[12], Andrea Califano[1,2,3,4] & Antonio Iavarone[1,5,7]

Network models of disease biolo

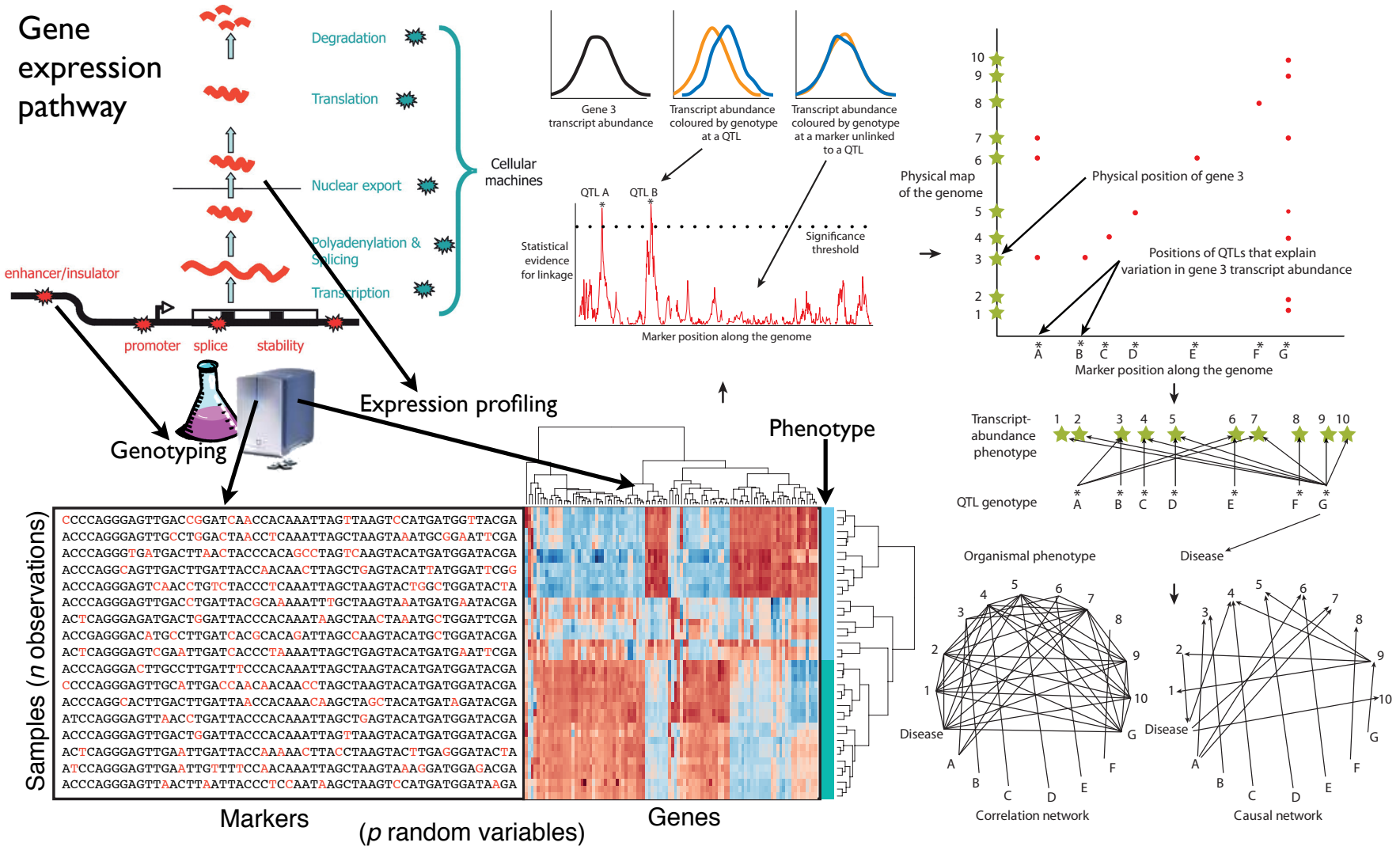
Open Network Biology publishes articles relating to predict systems linked to the corresponding coherent data sets up enabling a broad readership to evolve the models towards complex phenotypes. In addition to articles describing thes welcomes submissions of original research, software and r commentary, relevant to the emerging field of network biol

Types of biological networks by origin

- Built from literature: natural extension of functional annotations on genes to functional annotations on interactions. (not covered here)

- Built from high-throughput experimental data: natural extension of univariate, or bivariate, exploratory analyses (e.g., density estimation, clustering) to multivariate models (analogous to principal components analysis -PCA-). This is the goal of the package *qpgraph* described in this workshop.

Inferring molecular regulatory networks from high-throughput genomics data



Rockman, MV. Reverse engineering the genotype-phenotype map with natural genetic variation. *Nature*, 456:738-744, 2008.

Properties of biological networks

- Biological networks have been characterized in many different ways, demonstrating that they can follow higher-order organizational principles. Here we assume two simple properties of biological networks.

- **Sparseness**: the fraction of interactions present in a specific cellular state under study is much smaller than the total number of possible interactions.

- **High-dimension**: the number p of interacting entities (genes, proteins, SNPs, etc.) is very high and, in general, much larger than the number n of experimental samples available for estimating the network: the $p \gg n$ problem.

A motivating example with data from Escherichia coli

- Escherichia coli (E. coli) is the free-living organism for which a largest part of its transcriptional regulatory network is supported by some sort of experimental evidence.
- The database RegulonDB (Gama-Castro *et al.*, 2011) provides a curated set of transcription factor and target gene relationships that can be used as gold-standard for comparing different network inference approaches.
- We are going to use a microarray data set from Covert *et al.* (2004) with n=43 samples monitoring the response from E. coli during an oxygen shift.
- The experimental setup aimed at targeting the *a priori* most relevant part of the underlying regulatory network by using six strains with knockouts of key transcriptional regulators in the oxygen response: $\Delta appY$, Δfnr, $\Delta oxyR$, $\Delta soxS$ and the double knockout $\Delta arcA\Delta fnr$.

A motivating example with data from Escherichia coli

- Load the following packages:

```
> library(Biobase)
> library(Rgraphviz)
> library(qpgraph)
> library(org.EcK12.eg.db)
```

- Load and explore the E. coli data contained in the package:

```
> data(EcoliOxygen)
> ls()
```

```
[1] "filtered.regulon6.1"        "gds680.eset"
[3] "subset.filtered.regulon6.1" "subset.gds680.eset"
```

```
> gds680.eset
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 4205 features, 43 samples
  element names: exprs
protocolData: none
phenoData
  rowNames: GSM18235 GSM18236 ... GSM18289 (43 total)
  varLabels: Strain GrowthProtocol GenotypeVariation Description
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 15129285
Annotation: org.EcK12.eg.db
```

A motivating example with data from Escherichia coli

- We are going to focus on the subnetwork formed by the KO TFs and their target genes, as defined by RegulonDB:

```
> subset.gds680.eset
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 378 features, 43 samples
  element names: exprs
protocolData: none
phenoData
  rowNames: GSM18235 GSM18236 ... GSM18289 (43 total)
  varLabels: Strain GrowthProtocol GenotypeVariation Description
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 15129285
Annotation: org.EcK12.eg.db
```

```
> dim(subset.filtered.regulon6.1)
```

```
[1] 681   5
```

A motivating example with data from Escherichia coli

- Select the top 100 genes with largest variability through these expression data:

```
> IQRs <- esApply(subset.gds680.eset, 1, IQR)
> top100IQRgenes <- names(sort(IQRs, decreasing=TRUE))[1:100]
> eset100 <- subset.gds680.eset[top100IQRgenes, ]
> eset100
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 100 features, 43 samples
  element names: exprs
protocolData: none
phenoData
  rowNames: GSM18235 GSM18236 ... GSM18289 (43 total)
  varLabels: Strain GrowthProtocol GenotypeVariation Description
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 15129285
Annotation: org.EcK12.eg.db
```

- Build the corresponding subset of the RegulonDB gold-standard:

```
> maskTF <- subset.filtered.regulon6.1$EgID_TF %in% top100IQRgenes
> maskTG <- subset.filtered.regulon6.1$EgID_TG %in% top100IQRgenes
> regulon100 <- subset.filtered.regulon6.1[maskTF & maskTG, ]
> dim(regulon100)
```

```
[1] 128   5
```

A motivating example with data from Escherichia coli

- Estimate Pearson correlation coefficients (PCCs) between the all pairs of genes:

```
> pcc.est <- cor(t(exprs(eset100)))
> dim(pcc.est)
[1] 100 100
> pcc.est[1:5,1:5]
           948403     945316     945322     945507     945402
948403  1.0000000  0.6434955  0.6086180 -0.8740256  0.6391599
945316  0.6434955  1.0000000  0.9966337 -0.6577484  0.9971261
945322  0.6086180  0.9966337  1.0000000 -0.6221307  0.9941381
945507 -0.8740256 -0.6577484 -0.6221307  1.0000000 -0.6535775
945402  0.6391599  0.9971261  0.9941381 -0.6535775  1.0000000
```

- Using a high cutoff value on the absolute PCCs we can easily obtain a network of strongly correlated gene pairs using the *qpgraph* function `qpAnyGraph()`:

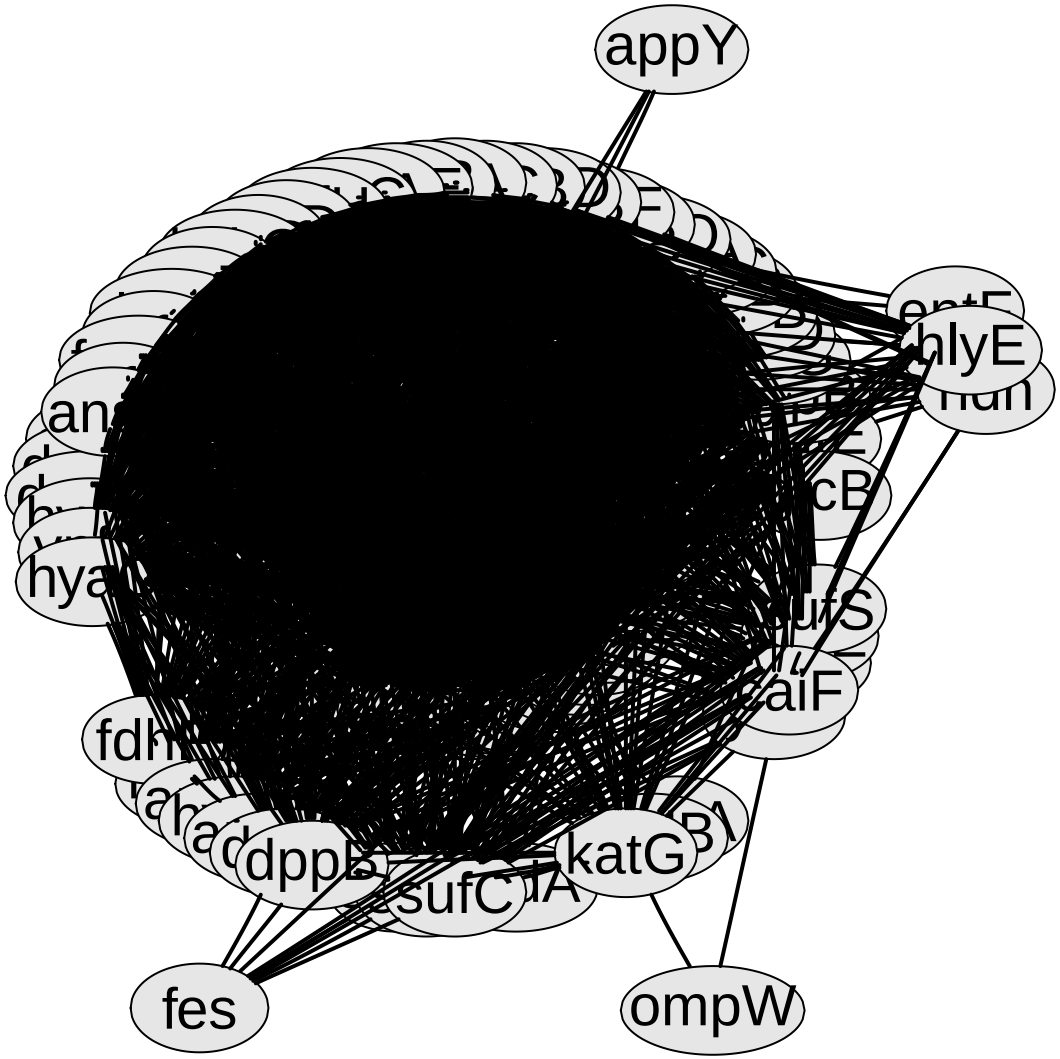
```
> pcc.g <- qpAnyGraph(abs(pcc.est), threshold=0.75, return.type="graphNEL")
> pcc.g
A graphNEL graph with undirected edges
Number of Nodes = 95
Number of Edges = 2043
```

- Plot the network with *Rgraphviz* and the *qpgraph* function `qpPlotNetwork()`:

```
> qpPlotNetwork(pcc.g, annotation="org.EcK12.eg.db")
```

A motivating example with data from Escherichia coli

Network built from all pairs of genes (i, j) such that their absolute PCCs $|\rho_{ij}| > 0.75$.



A motivating example with data from Escherichia coli

- A straightforward approach to remove non-interesting (because, e.g., we cannot interpret them), and possibly spuriously, correlated pairs of genes, is to discard edges where no TF is involved:

```
> TGgenes <- setdiff(featureNames(eset100), regulon100[, "EgID_TF"])
> allTGpairs <- as.matrix(expand.grid(list(TGgenes, TGgenes)))
> pcc.est[allTGpairs] <- NA
```

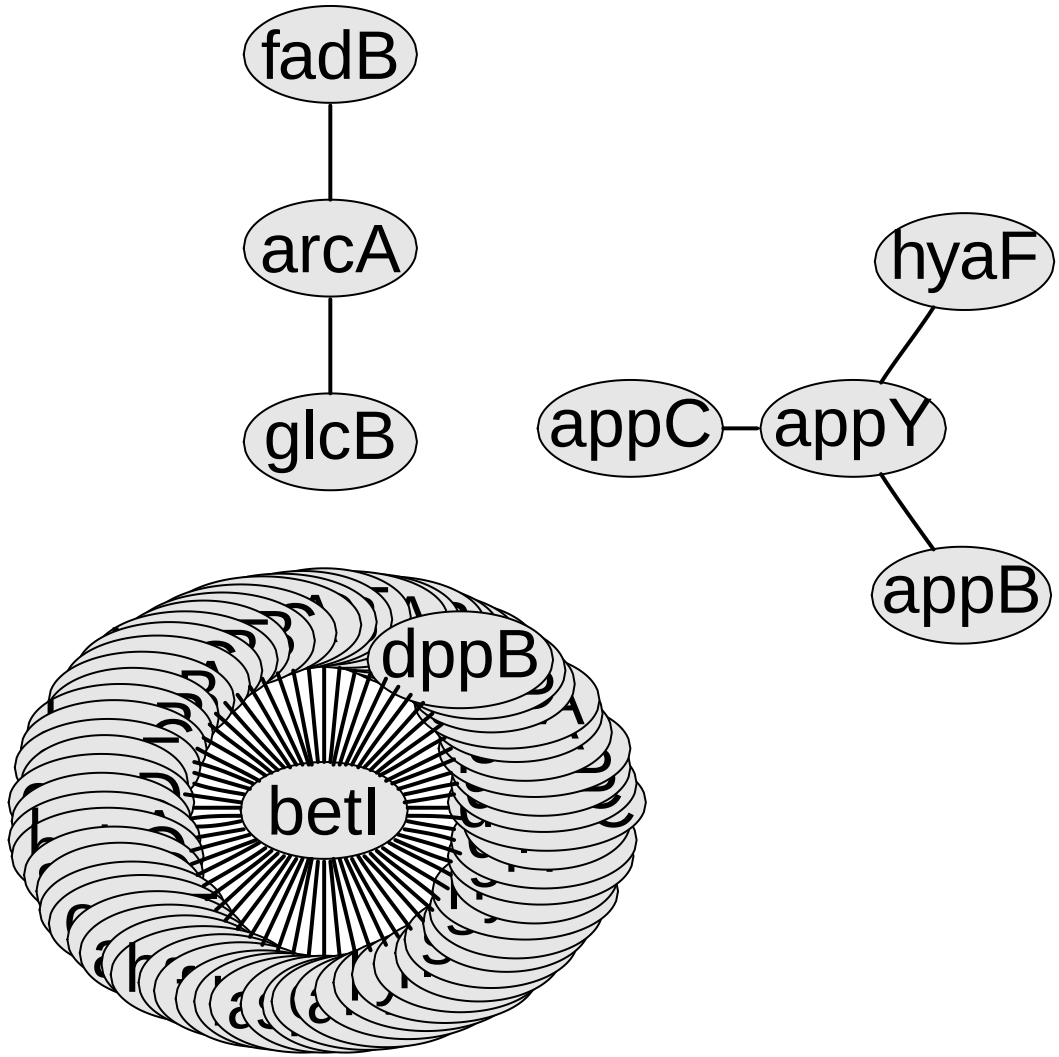
- Select the network using the same minimum cutoff value and plot it again:

```
> pcc.g <- qpAnyGraph(abs(pcc.est), threshold=0.75, return.type="graphNEL")
> pcc.g

A graphNEL graph with undirected edges
Number of Nodes = 74
Number of Edges = 71

> qpPlotNetwork(pcc.g, annotation="org.EcK12.eg.db")
```

- The *qpgraph* function `qpPCC()` calculates all pairwise PCCs and their *p*-values and returns them in a list with two *dspMatrix* objects that only store the upper triangle of these two matrices.

A motivating example with data from Escherichia coli

Network built from pairs of genes (i, j) with at least one TF such that their absolute PCCs $|\rho_{ij}| > 0.75$.



A motivating example with data from Escherichia coli

- Let's consider now building a network out of connecting genes by pure chance. This can be accomplished by simply generating pairwise correlations uniformly at random, as follows:

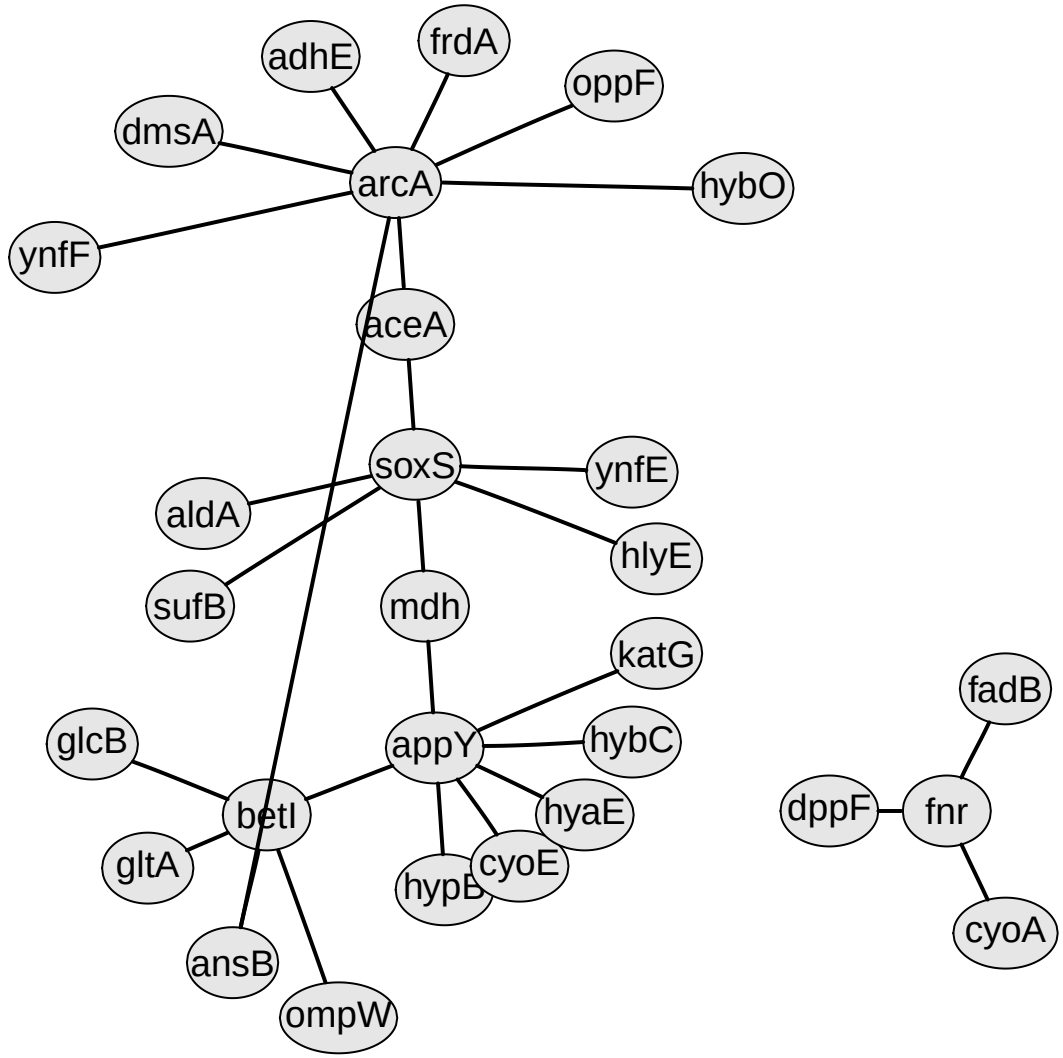
```
> set.seed(123)
> rnd.est <- matrix(runif(nrow(pcc.est)^2, min=-1, max=1),
+                   nrow=nrow(pcc.est), dimnames=dimnames(pcc.est))
> rnd.est <- (rnd.est + t(rnd.est)) / 2
> dim(rnd.est)
[1] 100 100
> rnd.est[1:5,1:5]
           948403      945316     945322      945507     945402
948403 -0.4248450  0.38829409 -0.3522971  0.66759267  0.9265216
945316  0.3882941 -0.33435292  0.4509720 -0.03609627 -0.3800301
945322 -0.3522971  0.45097197  0.2027315  0.29409561  0.3078829
945507  0.6675927 -0.03609627  0.2940956  0.45878130  0.2064337
945402  0.9265216 -0.38003013  0.3078829  0.20643369 -0.2091023
```

- Analogously to what we did before, we discard edges where no TF is involved, select a network with the same minimum cutoff and plot it:

```
> rnd.est[allTGpairs] <- NA
> rnd.g <- qpAnyGraph(abs(rnd.est), threshold=0.75, return.type="graphNEL")
> rnd.g
A graphNEL graph with undirected edges
Number of Nodes = 29
Number of Edges = 28
> qpPlotNetwork(rnd.g, annotation="org.EcK12.eg.db")
```

A motivating example with data from Escherichia coli

Network built from pairs of genes (i, j) with at least one TF such that *uniformly random* correlations $|\rho_{ij}| > 0.75$.



A motivating example with data from Escherichia coli

- Since we are analyzing E. coli data, we can use the RegulonDB gold-standard to assess the accuracy of each network inference method by means of precision-recall curves.
- For this purpose, the *qpgraph* package provides the function `qpPrecisionRecall()`:

```
> pcc.pr <- qpPrecisionRecall(abs(pcc.est), refGraph=regulon100[, 3:4])
> rnd.pr <- qpPrecisionRecall(abs(rnd.est), refGraph=regulon100[, 3:4])
```

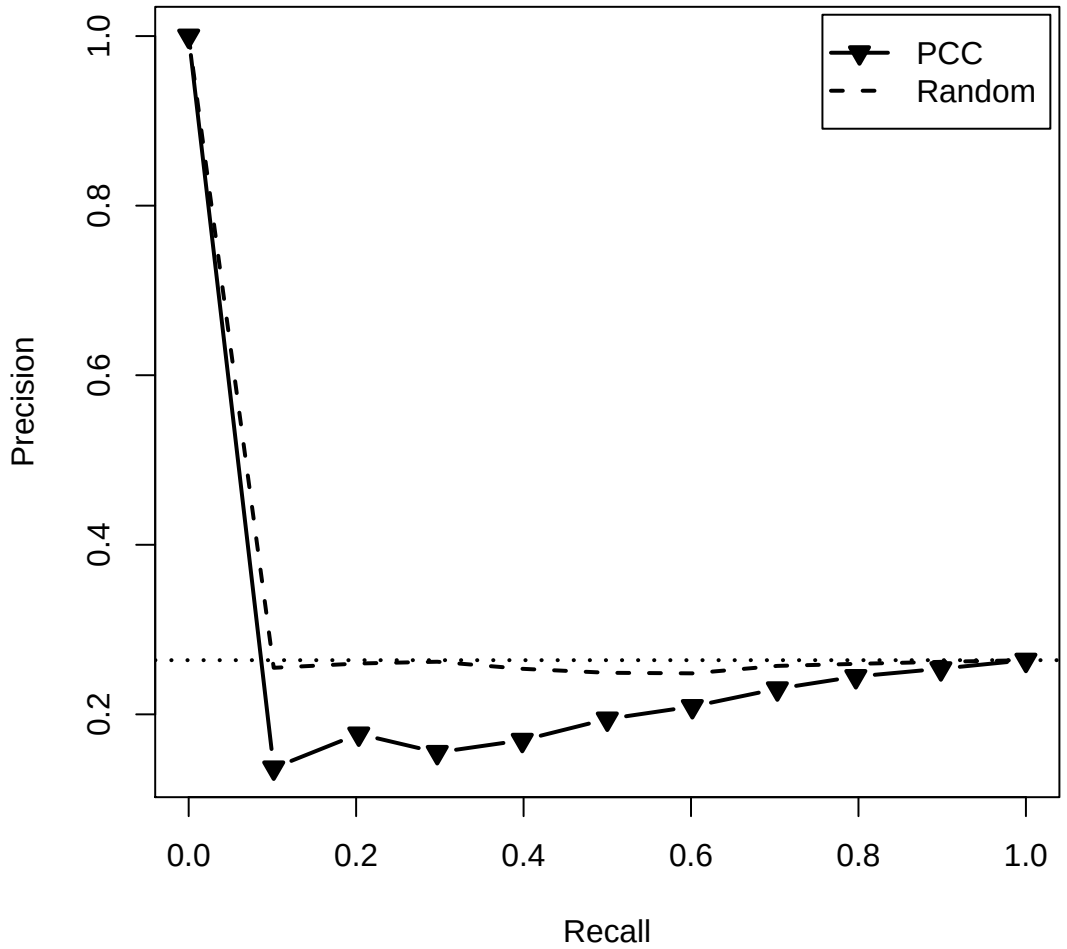
- We can plot the resulting curves as follows:

```
> plot(pcc.pr, type="b", lwd=2, pch=25, xlab="Recall", ylab="Precision", bg="black")
> lines(rnd.pr, type="l", lwd=2, lty=2, bg="black")
> legend("topright", c("PCC", "Random"), lwd=2, pch=c(25, -1),
+        lty=c(1, 2), pt.bg="black", inset=0.01)
> TFgenes <- setdiff(featureNames(eset100), TGgenes)
> nrow(regulon100)
[1] 128
> totalPossibleEdges <- length(TFgenes) * length(TGgenes) + choose(length(TFgenes), 2)
> totalPossibleEdges
[1] 485
> abline(h=nrow(regulon100) / totalPossibleEdges, lwd=2, lty=3)
```

where the last call to `abline()` draws a dotted line at the baseline performance obtained by predicting all possible edges as present.

A motivating example with data from Escherichia coli

- Assessment of performance with precision-recall curves: predicting edges uniformly at random works **in this case** better than using PCCs.



Undirected Gaussian graphical Markov models

- Assume that gene expression profiles form an independent and identically distributed (iid) *multivariate normal (Gaussian)* sample.
- Let $X_V = \{X_1, \ldots, X_p\}$ be a vector of continuous random variables representing genes such that

$$X_V \sim P(X_V) \equiv \mathcal{N}(\mu, \Sigma)\,.$$

where

  - μ is the p-dimensional mean vector parameter;
  - $\Sigma = \{\sigma_{ij}\}_{p\times p}$ is the covariance matrix;
  - $\Sigma^{-1} = K = \{\kappa_{ij}\}_{p\times p}$ is the concentration, also known as precision, matrix.

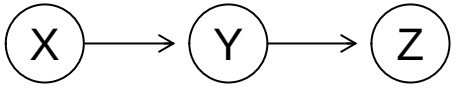
- Pearson and partial correlations can be calculated by scaling the covariance and concentration matrix, respectively, as follows:

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} \qquad \rho_{ij.R} = \frac{-\kappa_{ij}}{\sqrt{\kappa_{ii}\kappa_{jj}}}, R = V\backslash\{i,j\}\,.$$

Undirected Gaussian graphical Markov models

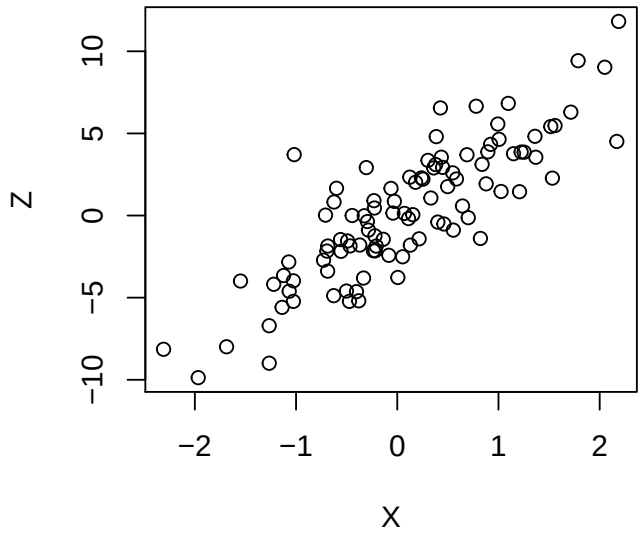
- Consider genes X, Y, Z where X, Y are transcription factors, X regulates Y and Y regulates Z, creating an indirect effect of X on Z:

```
> set.seed(123)
> X <- rnorm(100)
> Y <- X * 2 + rnorm(100)
> Z <- Y * 2 + rnorm(100)
```



- Plot the expression of X against Z and notice the high marginal (Pearson) correlation:

```
> cov2cor(cov(cbind(X, Y, Z)))
          X         Y         Z
X 1.0000000 0.8786993 0.8453189
Y 0.8786993 1.0000000 0.9725512
Z 0.8453189 0.9725512 1.0000000
> cor(X, Z)
[1] 0.8453189
> plot(X, Z)
```



- However, X and Z are only *marginally* dependent and, in fact, they are *conditionally independent given* Y.

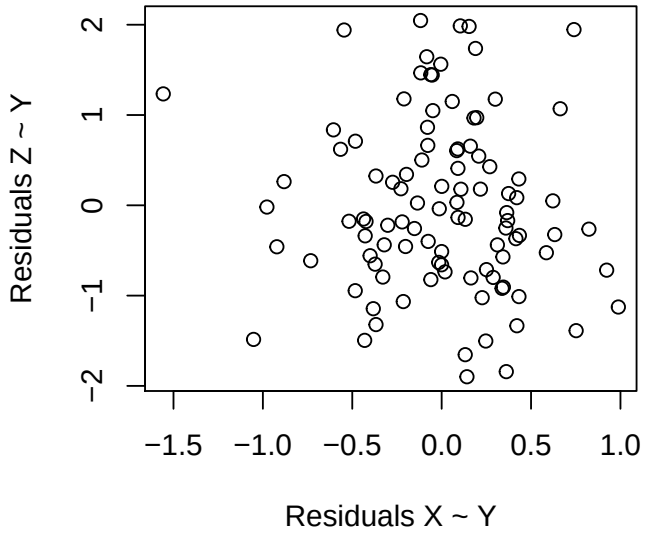
Undirected Gaussian graphical Markov models

- Partial correlations allow one to estimate the association between two variables adjusting for the remaining ones:

```
> -cov2cor(solve(cov(cbind(X,Y,Z))))
             X          Y           Z
X -1.00000000  0.4551541 -0.08337355
Y  0.45515415 -1.0000000  0.90090474
Z -0.08337355  0.9009047 -1.00000000
```

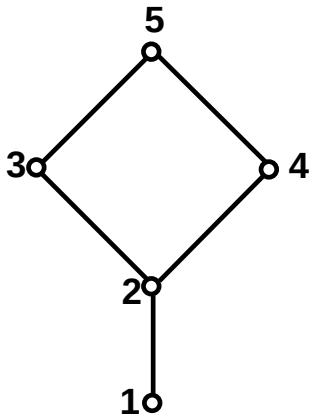
- They can be interpreted using a so-called partial regression plot:

```
> fitX <- lm(X ~ Y)
> fitZ <- lm(Z ~ Y)
> plot(resid(fitX), resid(fitZ),
+      xlab="Residuals X ~ Y",
+      ylab="Residuals Z ~ Y")
> cor.test(resid(fitX), resid(fitZ))
Pearson's product-moment correlation

data:  resid(fitX) and resid(fitZ)
t = -0.8282, df = 98, p-value = 0.4095
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.2752836  0.1149266
sample estimates:
        cor
-0.08337355
```



Undirected Gaussian graphical Markov models

- Let $G = (V, E)$ be an undirected graph with $V = \{1, \ldots, p\}$, a Gaussian graphical model can be described as follows:



$$\Sigma^{-1} = \begin{pmatrix} \kappa_{11} & \kappa_{12} & 0 & 0 & 0 \\ \kappa_{21} & \kappa_{22} & \kappa_{23} & \kappa_{24} & 0 \\ 0 & \kappa_{32} & \kappa_{33} & 0 & \kappa_{35} \\ 0 & \kappa_{42} & 0 & \kappa_{44} & \kappa_{45} \\ 0 & 0 & \kappa_{53} & \kappa_{54} & \kappa_{55} \end{pmatrix}$$

- A probability distribution $P(X_V)$ is undirected Markov w.r.t. G if

$$(i,j) \notin E \;\; \Rightarrow \;\; \kappa_{ij} = 0 \;\; \Leftrightarrow \;\; X_i \perp\!\!\!\perp X_j | X_V \backslash \{X_i, X_j\}$$

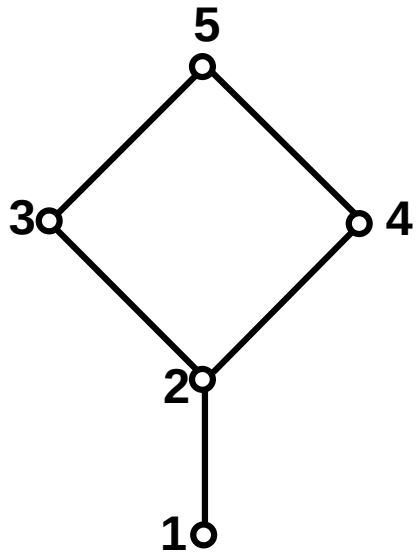
- These models are also known as covariance selection models (Dempster, 1972) or concentration graph models (Cox and Wermuth, 1996).
- Two vertices i and j are **separated** in G by a subset $S \subset V \backslash \{i,j\}$ iff every path between i and j intersects S, denoted hereafter by $i \perp_G j | S$.
- Global Markov property (Hammersley and Clifford, 1971):

$$i \perp_G j | S \Rightarrow X_i \perp\!\!\!\perp X_j | X_S \, .$$

Undirected Gaussian graphical Markov models

- Let's verify this by simulating a covariance matrix with the *qpgraph* function `qpG2Sigma()`:

```
> G <- matrix(c(0,1,0,0,0,1,0,1,1,0,0,1,0,0,1,0,1,0,0,1,0,0,1,1,0),
+             nrow=5, dimnames=list(1:5, 1:5))
> G
  1 2 3 4 5
1 0 1 0 0 0
2 1 0 1 1 0
3 0 1 0 0 1
4 0 1 0 0 1
5 0 0 1 1 0
> set.seed(123)
> Sigma <- qpG2Sigma(G, rho=0.5)
> round(solve(Sigma), digits=2)
      1     2     3     4     5
1  0.79 -0.56  0.00  0.00  0.00
2 -0.56  5.35 -2.77 -0.85  0.00
3  0.00 -2.77  2.65  0.00 -0.35
4  0.00 -0.85  0.00  1.38 -0.60
5  0.00  0.00 -0.35 -0.60  2.50
```



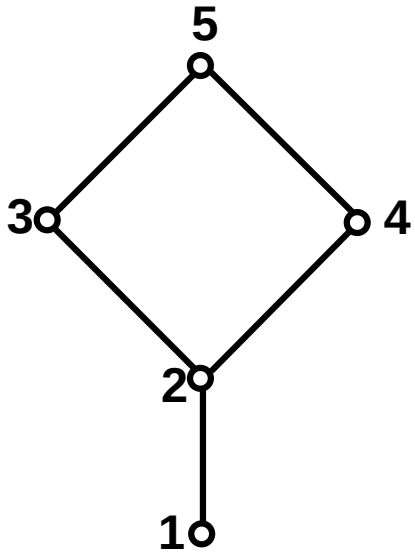
- Note that the mean marginal (Pearson) correlation between variables connected in G also approaches the nominal value `rho=0.5`:

```
> mean(cov2cor(as.matrix(Sigma))[upper.tri(as.matrix(Sigma)) & G])
[1] 0.5604867
```

Undirected Gaussian graphical Markov models

- Sample lots of observations ($n \gg p$) from this multivariate Gaussian distribution and try to infer the present and missing edges from the graph by estimating the pattern of zeroes in the concentration matrix:

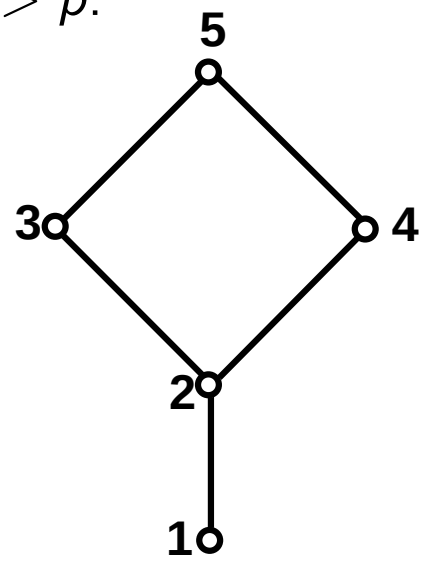
```
> library(mvtnorm)
> set.seed(123)
> X <- rmvnorm(n=50000, sigma=as.matrix(Sigma))
> dim(X)

[1] 50000     5

> S <- cov(X)
> round(solve(S), digits=2)

      [,1]  [,2]  [,3]  [,4]  [,5]
[1,]  0.79 -0.56  0.00  0.00 -0.01
[2,] -0.56  5.35 -2.78 -0.83  0.01
[3,]  0.00 -2.78  2.65  0.00 -0.36
[4,]  0.00 -0.83  0.00  1.37 -0.60
[5,] -0.01  0.01 -0.36 -0.60  2.49

> mean(cov2cor(S)[upper.tri(S) & G])

[1] 0.5598355
```



Undirected Gaussian graphical Markov models

- Now with fewer observations but still with $n > p$:

```
> set.seed(123)
> X <- rmvnorm(n=50, sigma=as.matrix(Sigma))
> dim(X)
[1] 50  5
> S <- cov(X)
> round(solve(S), digits=2)
      [,1]  [,2]  [,3]  [,4]  [,5]
[1,]  0.95 -0.59 -0.12  0.02  0.28
[2,] -0.59  5.85 -2.68 -0.76  0.28
[3,] -0.12 -2.68  2.58 -0.09 -0.42
[4,]  0.02 -0.76 -0.09  1.72 -1.16
[5,]  0.28  0.28 -0.42 -1.16  3.15
```



- Perform a hypothesis test $H_0 : \rho_{ij.R} = 0$ using the `qpCItest()` function:

```
> qpCItest(X, i=1, j=5, Q=2:4, long.dim.are.variables=FALSE)
Conditional independence test for continuous data using a t test for
zero partial regression coefficient

data:  1 and 5 given {2, 3, 4}
t = -1.0837, df = 45, p-value = 0.2843
alternative hypothesis: true partial regresion coefficient is not equal to 0
sample estimates:
      beta
-0.2903989
> coef(lm(X[,1] ~ X[,2]+X[,3]+X[,4]+X[,5]))
(Intercept)      X[, 2]      X[, 3]      X[, 4]      X[, 5]
 0.03864379  0.61982129  0.12406899 -0.01774059 -0.29039886
```

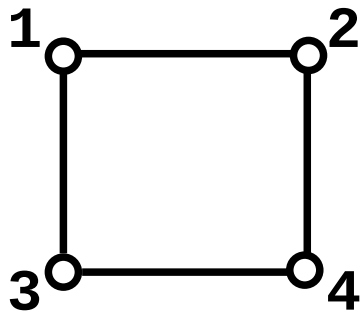
Undirected Gaussian graphical Markov models

- Sample fewer observations than random variables:

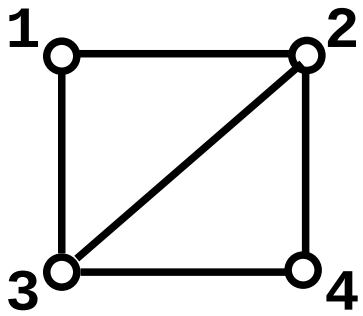
```
> set.seed(123)
> X <- rmvnorm(n=4, sigma=as.matrix(Sigma))
> dim(X)
[1] 4 5
> S <- cov(X)
> round(solve(S), digits=2)
Error in solve.default(S) :
  system is computationally singular: reciprocal condition number = 2.12472e-18
> qr(S)$rank
[1] 3
```

Undirected Gaussian graphical Markov models

- Conditions for the existence of the maximum likelihood estimate (MLE) of $K \equiv \Sigma^{-1}$:
  - to estimate $\hat{K} = S^{-1}$, the sample covariance matrix S must have full rank. This only happens if and only if $n > p$ (Dykstra, 1970).
  - $\hat{K}$ exists iff $w(G)$, the size of a maximum clique in G, is smaller than n ($w(G) < n$), with G being **decomposable** (Lauritzen, 1996):



non-decomposable graph



decomposable graph

  - The function `qpPAC()` enables this later approach:

```
> round(qpPAC(X, G, verbose=FALSE)$R, digits=2)

5 x 5 Matrix of class "dspMatrix"
     1     2    3     4    5
1 1.00  0.51 0.00  0.00 0.00
2 0.51  1.00 0.51 -0.51 0.00
3 0.00  0.51 1.00  0.00 0.46
4 0.00 -0.51 0.00  1.00 0.32
5 0.00  0.00 0.46  0.32 1.00
```

Undirected Gaussian graphical Markov models

- Main types of approaches to the problem of estimating a Gaussian graphical model from data with $p \gg n$:
  - Bayesian approaches with sparsity inducing priors (e.g., Dobra *et al.*, *J. Mult. Anal.*, 2004).
  - shrinkage estimate of the covariance matrix (e.g., Schäfer and Strimmer, *Stat. Appl. Genet. Mol. Biol.*, 2005).
  - dimension reduction (e.g., Segal *et al.*, *J. Mach. Learn. Res.*, 2005).
  - limited-order partial correlations (e.g., Castelo and Roverato *et al.*, *J. Mach. Learn. Res.*, 2006).
  - lasso estimate of the inverse covariance matrix (e.g., Friedman *et al.*, *Biostatistics*, 2008).

q-order partial correlation graphs – qp-graphs

- Instead of using *full*-order partial correlations, we can employ *limited*-order partial correlations by using subsets

$$Q \subseteq R = V \backslash \{i, j\}, \quad |Q| = q, \quad q < (n-2)\,.$$

- Limited-order partial correlations allow us to test $H_0 : \rho_{ij.Q} = 0$ with standard techniques, such that

$$\rho_{ij.Q} = 0 \iff X_i \perp\!\!\!\perp X_j | X_Q\,.$$

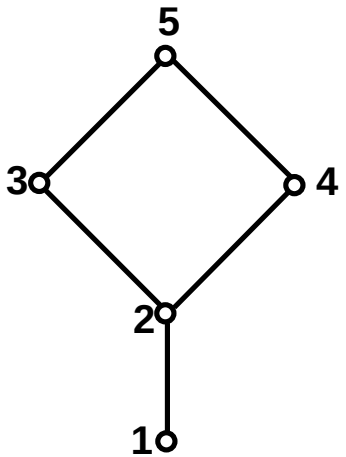
- The rationale behind is that if the underlying network G is sufficiently sparse, we can expect to identify many missing edges with $\rho_{ij.Q}$, i.e., accepting many tests $H_0 : \rho_{ij.Q} = 0$.

- From another perspective, we will work using marginal distributions of size $(q+2) < n$.

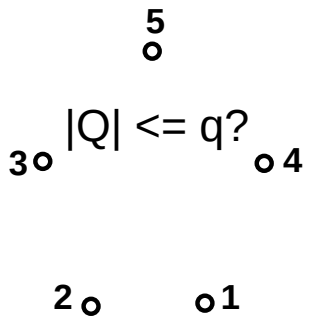
q-order partial correlation graphs – qp-graphs

- Definition of a q-order partial correlation graph, or qp-graph for short (Castelo and Roverato, 2006):

Underlying G representing full-order partial correlations



qp-graph $G^{(q)}$ representing q-order partial correlations

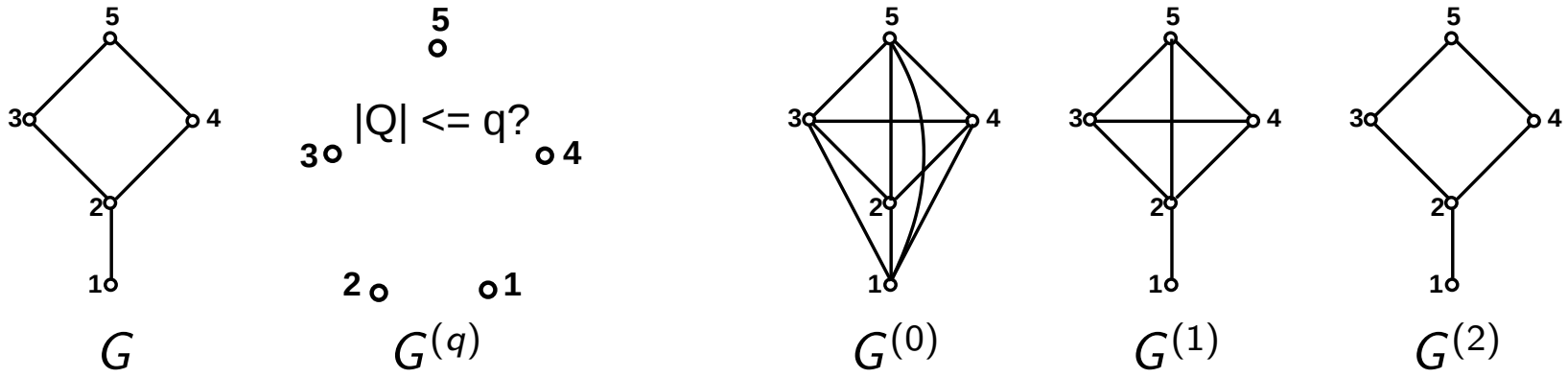


G associated to $P(X_V)$

$G^{(q)}$ associated to all marginal distributions $P_Q(X_V)$ of size $(q+2)$.

q-order partial correlation graphs – qp-graphs

- qp-graphs are an approximation to the underlying graph G:



- Assuming *faithfulness* of $P(X_V)$ to G (Castelo and Roverato, 2006):
  - If $r \leq q$ then $G^{(r)}$ will be always larger than $G^{(q)}$, it will have more edges.
  - $G^{(q)}$ is always going to be larger than G.
  - $P(X_V)$ is Markov w.r.t. $G^{(q)}$.

Learning qp-graphs with the non-rejection rate

- The *qpgraph* package implements a statistical procedure to learn qp-graphs using the so-called **non-rejection rate**, or NRR for short.
- The NRR is a measure of linear association between two variables (i, j) over all marginal distributions of size $(q + 2)$ defined by all subsets $\mathcal{Q}_{ij} = \{Q_k : Q_k \subseteq V \backslash \{i, j\}, |Q_k| = q\}$ with $|\mathcal{Q}_{ij}| = \binom{p-2}{q} = m$.
- Let T_{ij}^q be a binary random variable taking values $t_{ij}^{Q_1}, \ldots, t_{ij}^{Q_m}$ as follows. For every subset $Q \in \mathcal{Q}_{ij}$ of size q, test $H_0 : \rho_{ij.Q} = 0$ (i.e., $H_0 : X_i \perp\!\!\!\perp X_j | X_Q$) and, using a significance level α, decide:
  - (i) if H_0 is rejected then T_{ij}^q takes value 0;
  - (ii) if H_0 is accepted then T_{ij}^q takes value 1.
- The sample NRR equals the arithmetic mean value of non-rejections:

$$\mathrm{NRR}(i, j|q) := \frac{1}{m} \sum_{k=1}^{m} t_{ij}^{Q_k} .$$

- Since m can be very large, the estimation of the sample NRR is performed with a Monte Carlo method by sampling a limited number of subsets $Q \in \mathcal{Q}_{ij}$, e.g., 100, uniformly at random.

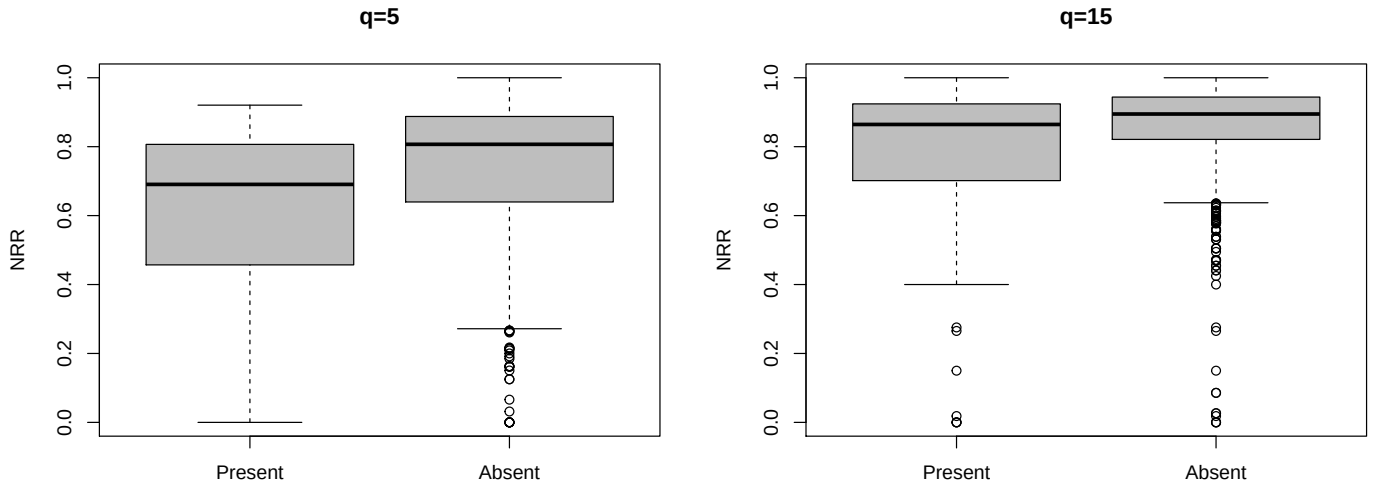
Learning qp-graphs with the non-rejection rate

- NRR values can be estimated with the function `qpNrr()`. Let's come back to the E. coli motivating example:

```
> nrr.q5 <- qpNrr(eset100, q=5, pairup.i=TFgenes, pairup.j=featureNames(eset100),
+                 verbose=FALSE)
> nrr.q15 <- qpNrr(eset100, q=15, pairup.i=TFgenes, pairup.j=featureNames(eset100),
+                  verbose=FALSE)
```

- As $n - q$ grows large, NRR values converge to 0 for $(i, j) \in G$. However, all NRR values increase as q approaches $n - 2$.

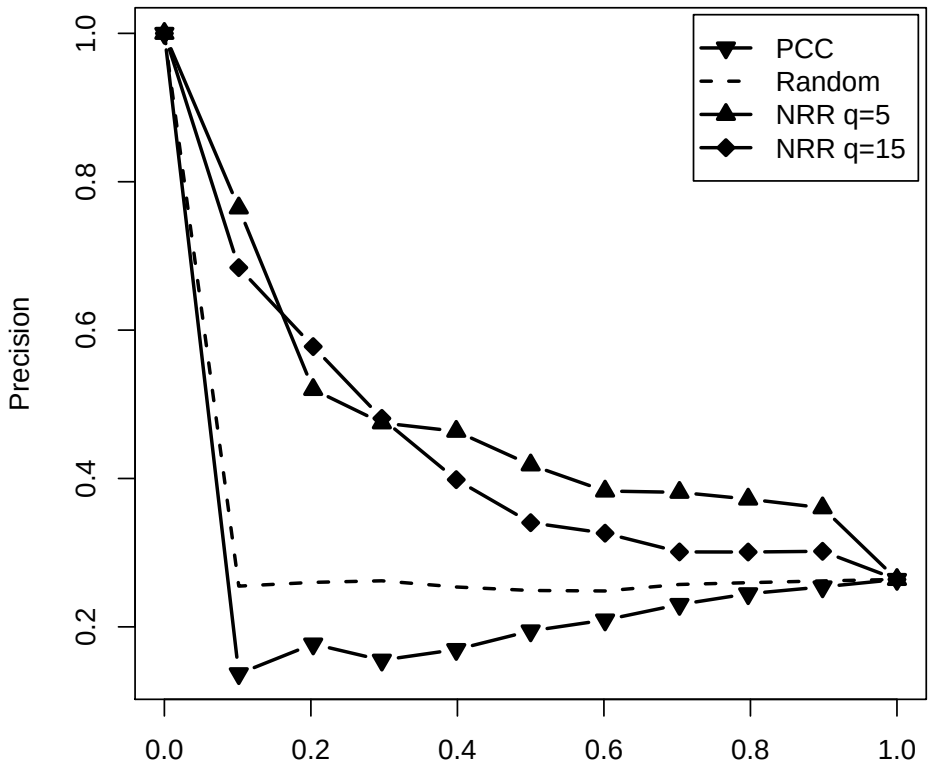
```
> mask <- matrix(FALSE, nrow=100, ncol=100, dimnames=dimnames(nrr.q15))
> mask[as.matrix(regulon100[, 3:4])] <- TRUE
> par(mfrow=c(1, 2))
> boxplot(list(Present=as.matrix(nrr.q5)[mask], Absent=as.matrix(nrr.q5)[!mask]),
+         main="q=5", col="grey", ylab="NRR")
> boxplot(list(Present=as.matrix(nrr.q15)[mask], Absent=as.matrix(nrr.q15)[!mask]),
+         main="q=15", col="grey", ylab="NRR")
```



Learning qp-graphs with the non-rejection rate

- Assess performance again with precision-recall curves:

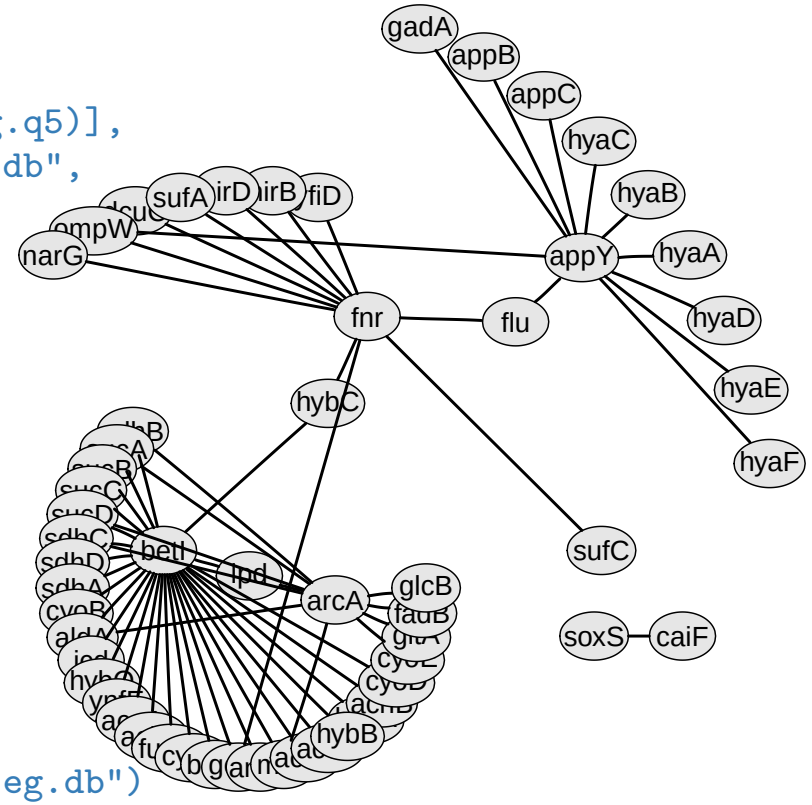
```
> nrr.q5.pr <- qpPrecisionRecall(nrr.q5, refGraph=regulon100[, 3:4], decreasing=FALSE)
> nrr.q15.pr <- qpPrecisionRecall(nrr.q15, refGraph=regulon100[, 3:4], decreasing=FALSE)
> par(mfrow=c(1, 1))
> plot(pcc.pr, type="b", lwd=2, pch=25, xlab="Recall", ylab="Precision", bg="black")
> lines(rnd.pr, type="l", lwd=2, lty=2, bg="black")
> lines(nrr.q5.pr, type="b", lwd=2, pch=24, bg="black")
> lines(nrr.q15.pr, type="b", lwd=2, pch=23, bg="black")
> legend("topright", c("PCC", "Random", "NRR q=5", "NRR q=15"), lwd=2,
+        pch=c(25, -1, 24, 23), lty=c(1, 2, 1), pt.bg="black", inset=0.01)
```



Learning qp-graphs with the non-rejection rate

- Select a network at certain precision level and explore its contents:

```
> nrr.thr.q5 <- qpPRscoreThreshold(nrr.q5.pr, level=0.5, recall.level=FALSE, max.score=0)
> nrr.g.q5 <- qpGraph(nrr.q5, threshold=nrr.thr.q5, return.type="graphNEL")
> nrr.g.q5

A graphNEL graph with undirected edges
Number of Nodes = 57
Number of Edges = 63

> qpTopPairs(nrr.q5[nodes(nrr.g.q5), nodes(nrr.g.q5)],
+            nrr.g.q5, annotation="org.EcK12.eg.db",
+            n=10)

        i      j iSymbol jSymbol          x
1  945908 947068     fnr    yfiD 0.00000000
2  944981 947376    betI    betB 0.00000000
3  945585 948797    appC    appY 0.00000000
4  946540 948797     flu    appY 0.00000000
5  947547 948797    appB    appY 0.00000000
6  948336 948874    fadB    arcA 0.00000000
7  948857 948874    glcB    arcA 0.03157895
8  945572 948797    hyaF    appY 0.06593407
9  945908 946540     fnr     flu 0.12500000
10 945908 947390     fnr    gcvT 0.15053763

> qpPlotNetwork(nrr.g.q5, annotation="org.EcK12.eg.db")
```



Learning qp-graphs with the non-rejection rate

- Formally, T_{ij}^q is a Bernoulli random variable and the NRR can be defined as its expectancy:

$$\mathrm{NRR}(i,j|q) := \mathrm{E}[T_{ij}^q] = \Pr(T_{ij}^q = 1)\,.$$

- This theoretical value can be described as follows, let

$$\Pr(T_{ij}^q = 1|Q) = \begin{cases} (1-\alpha) & \text{if } Q \text{ separates } i \text{ and } j \text{ in } G; \\ \beta_{ij.Q} & \text{otherwise;} \end{cases}$$

  where α and $\beta_{ij.Q}$ are the probability of the first and second type error of the test, respectively.

- Let $\mathcal{Q}_{ij}$ be the collection of all possible subsets Q of size q, then by the law of total probability,

$$\Pr(T_{ij}^q = 1) = \sum_{Q \in \mathcal{Q}_{ij}} \Pr(T_{ij}^q = 1|Q)\Pr(Q)\,,$$

Learning qp-graphs with the non-rejection rate

- Assuming that if $|\mathcal{Q}_{ij}| = m$ then $\Pr(Q) = 1/m$, it can be shown that:

$$\Pr(T_{ij}^q = 1) = \beta_{ij}^q(1 - \pi_{ij}^q) + (1 - \alpha)\pi_{ij}^q \,,$$

  where β_{ij}^q is the mean value of all the Type-II errors $\beta_{ij.Q}$ and π_{ij}^q is the fraction of subsets Q that separate i and j in G.

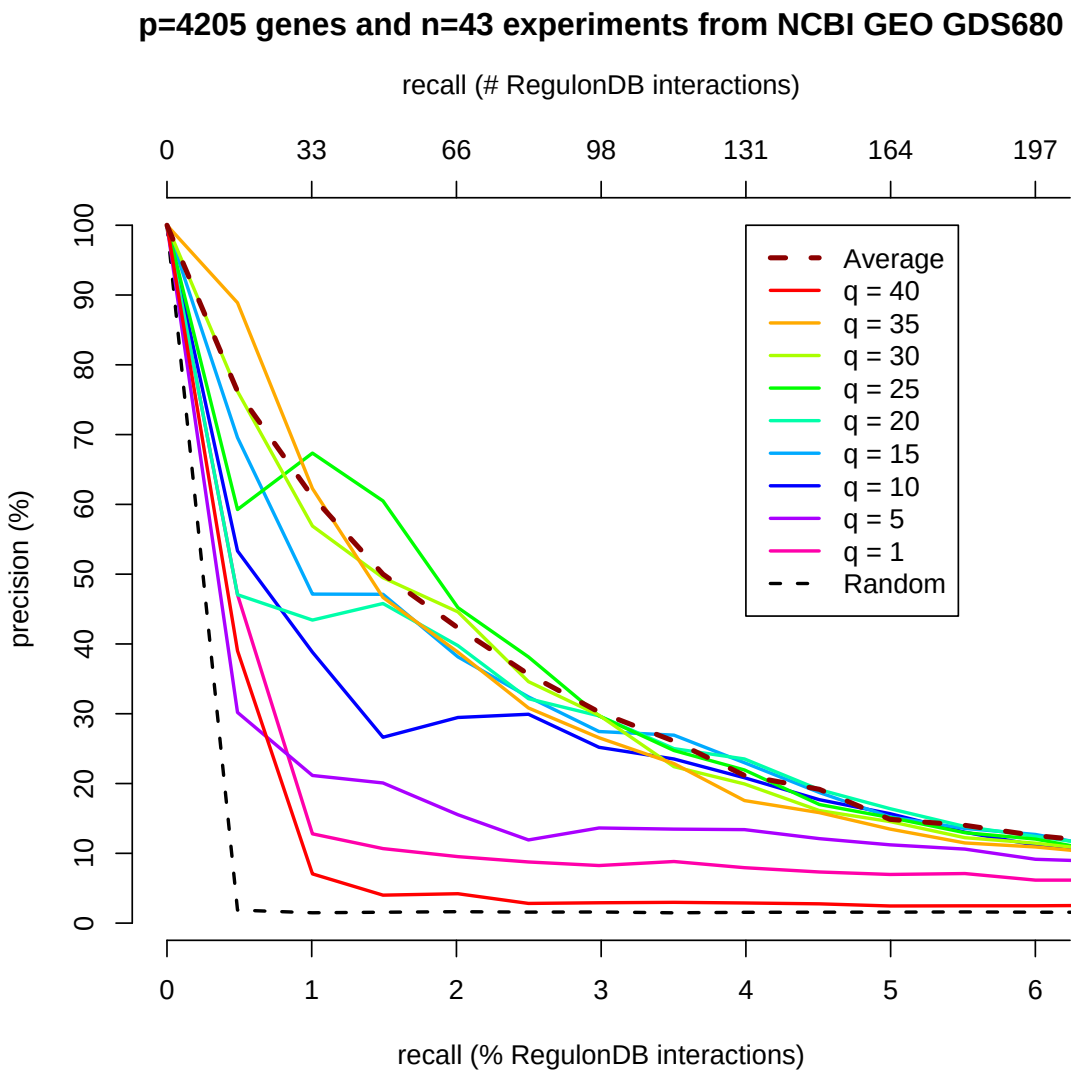
- The values β_{ij}^q and π_{ij} are *unkown* but it can be shown that $\beta_{ij}^q \leq (1 - \alpha)$ implies that the NRR is an upper bound to both values:

$$\mathrm{NRR}(i,j|q) := \beta_{ij}^q \leq \mathrm{NRR}(i,j|q) \quad \text{and} \quad \pi_{ij} \leq \frac{\mathrm{NRR}(i,j|q)}{1-\alpha} \,.$$

- Hence, a value of the non-rejection rate for a pair of genes that is close to zero implies both
  - (i) the probability that a subset of q genes separates the genes in the network is close to zero.
  - (ii) for those sets $Q \in \mathcal{Q}_{ij}$ with $\rho_{ij.Q} \neq 0$ then such association can be detected with high power.

Learning qp-graphs with the non-rejection rate

- A sensible option to avoid choosing a single q value is to average through different ones (Castelo and Roverato, 2009) with `qpAvgNrr()`:

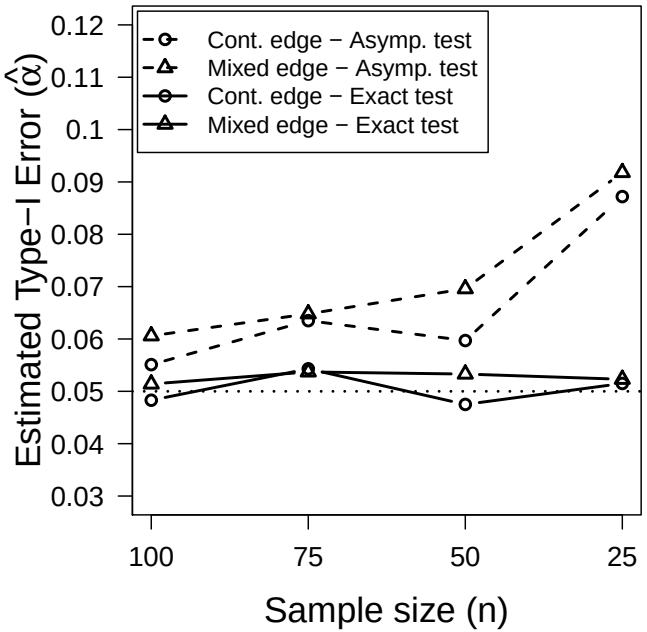


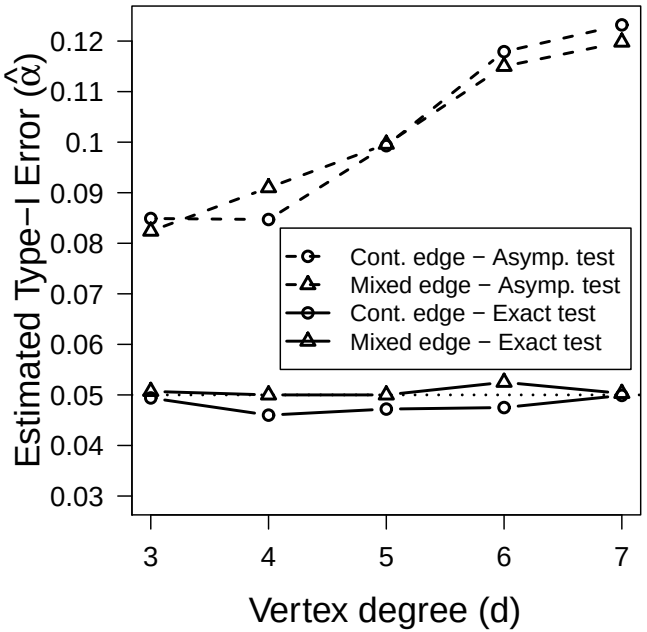
Learning qp-graphs from expression and genotype data

- We have extended the previous framework to mixed continous (expression) and discrete (genotype) data (Tur and Castelo, *in preparation*).
- The extension essentially consists of replacing the conditional independence test for continuous data by a proper one for mixed data.
- We have used mixed graphical model theory (Lauritzen, 1996; Edwards, 2000) to achieve that goal. Concretely, we will be using an *homogeneous* mixed graphical model (i.e., genotypes can affect mean expression levels but not their variance).
- To use it, if the input data is a matrix, one should specify the discrete variables with the argument `I` in calls to `qpCItest()`, `qpNrr()`, etc.; with `ExpressionSet` and `smlSet` objects as input, the software identifies automatically what features are discrete.

Learning qp-graphs from expression and genotype data

- Exploiting the fact that we perform a likelihood ratio test between a saturated and a constrained model which are both *decomposable*, one can use an exact test (Tur and Castelo, *in preparation*):





Learning qp-graphs from expression and genotype data

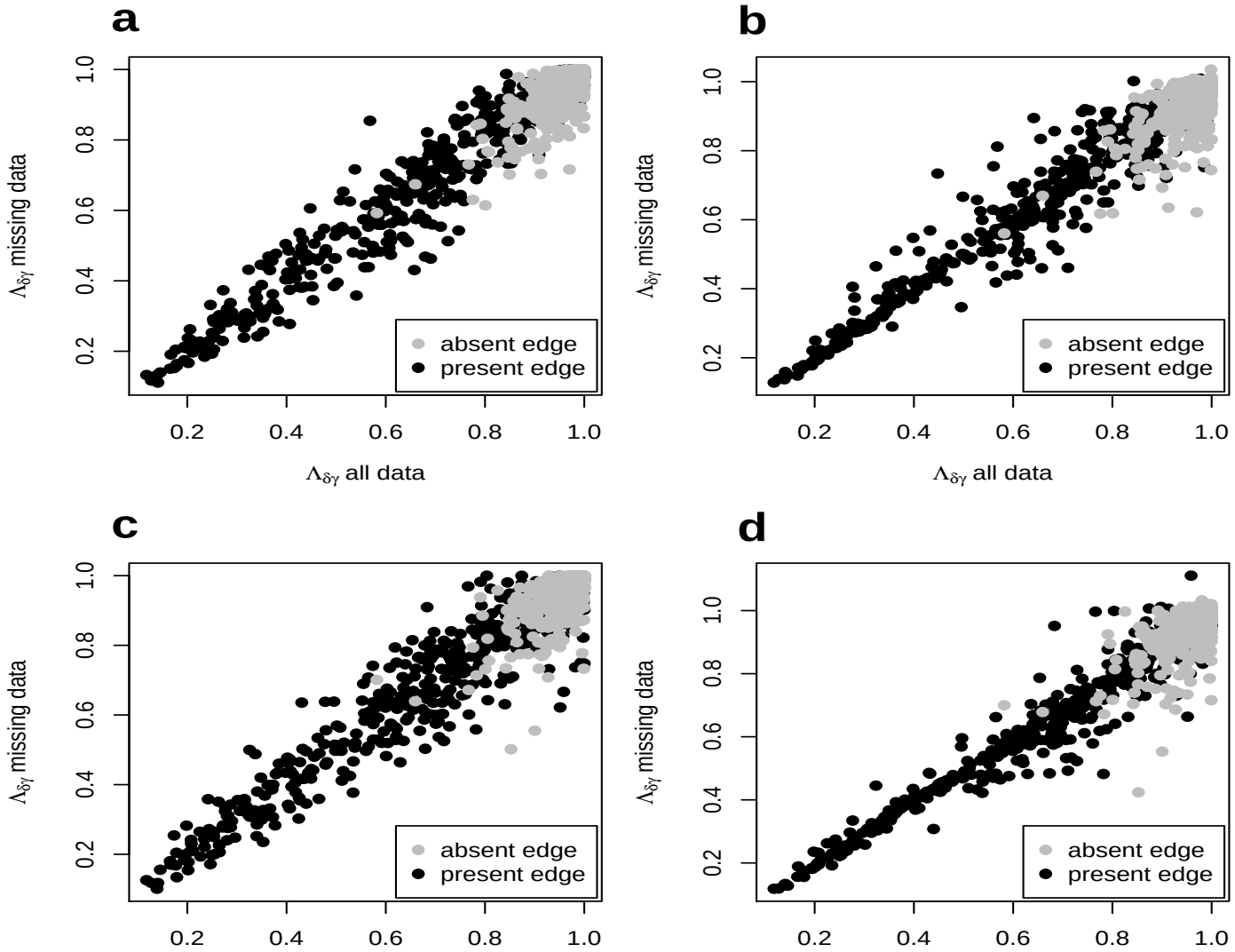
- Missing data occur frequently in genotype and clinical data. They are handled by default using a *complete-case analysis* strategy.

- In the forthcoming release, a maximum likelihood strategy based on the EM algorithm will be available through an argument `use=c("complete.obs", "em")`, similarly to the `cov()` base function.

- In our preliminary experiments, the EM algorithm provides more accurate estimates of the magnitude of the associations. However, the non-rejection rate seems to work quite robustly using complete-case analysis, which is much faster (Tur and Castelo, *in preparation*).

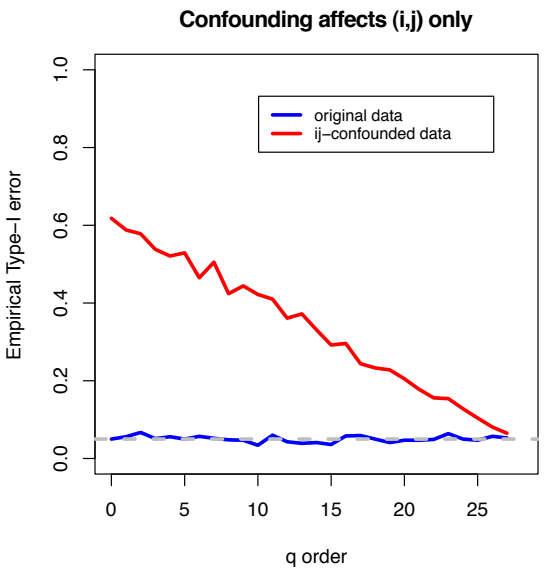
Learning qp-graphs from expression and genotype data

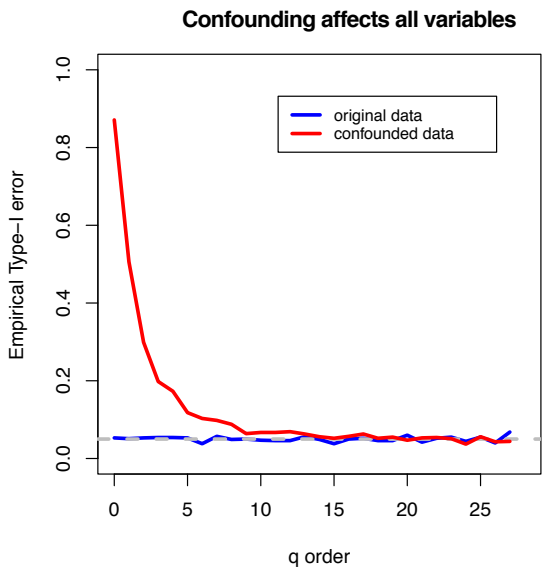
- Performance with simulated mixed data: (a,b) MCAR, (c,d) MAR, (a,c) complete-case analysis, (b,d) EM algorithm (Tur and Castelo, *in preparation*):

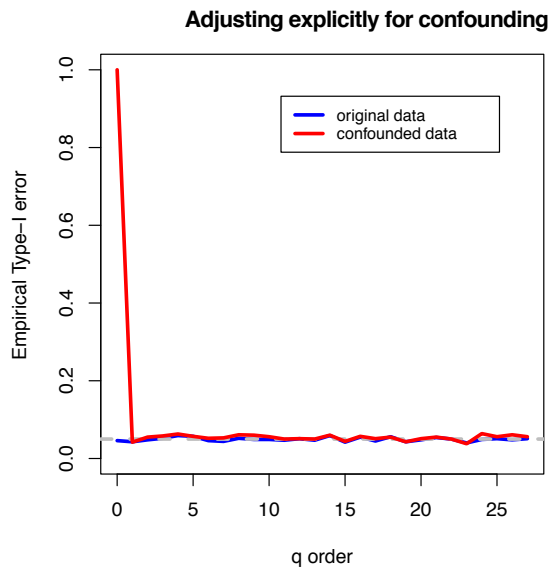


Adjusting for batch and other confounding effects

- Batch and other confounding effects may be implicitly (by conditioning on expression) or explicitly (via the `fix.Q` argument) adjusted within the calculations (Tur and Castelo, *in preparation*).







Mapping eQTL associations

- Inferring networks from eQTL data will be illustrated in this workshop with the following data:

```
> load("qpgraphWorkshop/YeastGG.RData")
> ls(pattern="sacCer3")
[1] "BremGGsacCer3chr3n4" "sacCer3chrLen"       "sacCer3genePos"
[4] "sacCer3markerPos"
```

- The matrix `BremGGsacCer3chr3n4` contains genotype and expression data from 112 segregants of an experimental cross between a lab and a wild strain of yeast (Brem *et al.*, 2005).

- This is a subset of the original data which contains expression profiles of about all yeast genes (6,216), but genotypes for only chromosomes III and IV (269 genetic markers).

Mapping eQTL associations

- Let's explore the data a little bit:

```
> dim(BremGGsacCer3chr3n4)
[1]  112 6485
> BremGGsacCer3chr3n4[1:5, 265:275]
       5913_at_x15 5914_at_x00 5916_at_x13 5917_at_x01 3357_at_x12     YDR407C
10_1_c           0           0           0           0           0  0.12011749
10_3_c           0           0           0           0           0 -0.07569854
10_4_d           1           1           1           1           1 -0.05744455
11_3_b           1           1           1           1           1 -0.14807044
9_6_d            0           0           0           0           0  0.12339946
           YDR180W     YAR050W     YKL129C     YOR328W    YJR138W
10_1_c  0.13858799 -0.53439312 -0.17761065 -0.08306616 -0.2144190
10_3_c -0.11020247  0.09272298  0.08288762  0.27866605 -0.1625249
10_4_d -0.13596410 -0.57002534 -0.38653009 -0.31692702 -0.2855860
11_3_b -0.04119522 -0.72399742 -0.33699962 -0.36677387 -0.4166205
9_6_d  -0.13085680  0.20014412  0.31184083  0.39644932  0.1210753
> sum(is.na(BremGGsacCer3chr3n4)) / (269*112)
[1] 0.02223845
```

Mapping eQTL associations

- Objects `sacCer3markerPos` and `sacCer3genePos` contain chromosomal locations of the markers and genes, respectively, and `sacCer3chrLen` the chromosome sequence lengths. All these physical genetic positions are based on the yeast assembly *sacCer3* at `http://genome.ucsc.edu`.

```
> head(sacCer3markerPos)
```

```
           chromosome position
6960_at_x06          3    14066
6929_at_x07          3    43867
6929_at_x06          3    43879
6893_at_x00          3    54436
6900_at_x13          3    64311
6906_at_x06          3    75021
```

```
> head(sacCer3genePos)
```

```
        chromosome position
YDR407C          4  1284069
YDR180W          4   821295
YAR050W          1   203403
YKL129C         11   196349
YOR328W         15   931803
YJR138W         10   684567
```

```
> sacCer3chrLen
```

```
   chrI   chrII  chrIII   chrIV    chrV   chrVI  chrVII chrVIII   chrIX    chrX
 230218  813184  316620 1531933  576874  270161 1090940  562643  439888  745751
  chrXI  chrXII chrXIII  chrXIV   chrXV  chrXVI
 666816 1078177  924431  784333 1091291  948066
```

Mapping eQTL associations

- Start by calculating all (un)conditional independence tests between genetic markers and gene expression profiles with the function `qpAllCItests()`:

```
> allci <- qpAllCItests(BremGGsacCer3chr3n4, I=rownames(sacCer3markerPos),
+                       pairup.i=rownames(sacCer3markerPos),
+                       pairup.j=rownames(sacCer3genePos), verbose=FALSE)
```

- This function returns by default only raw p-values of all performed tests, but the statistics and actual sample sizes per test can be also obtained with the argument `return.type`.

```
> allci$p.value[rownames(sacCer3markerPos)[1:5], rownames(sacCer3genePos)[1:5]]
```

```
5 x 5 Matrix of class "dgeMatrix"
               YDR407C   YDR180W    YAR050W   YKL129C    YOR328W
6960_at_x06 0.21019083 0.9546130 0.36074057 0.7179408 0.09387935
6929_at_x07 0.09233164 0.4617914 0.42947716 0.9163275 0.31350710
6929_at_x06 0.09233164 0.4617914 0.42947716 0.9163275 0.31350710
6893_at_x00 0.60941611 0.2715069 0.97658206 0.8547696 0.40867417
6900_at_x13 0.35657822 0.4148094 0.04218579 0.1277865 0.03522584
```

```
> allci$statistic
```

```
[1] NA
```

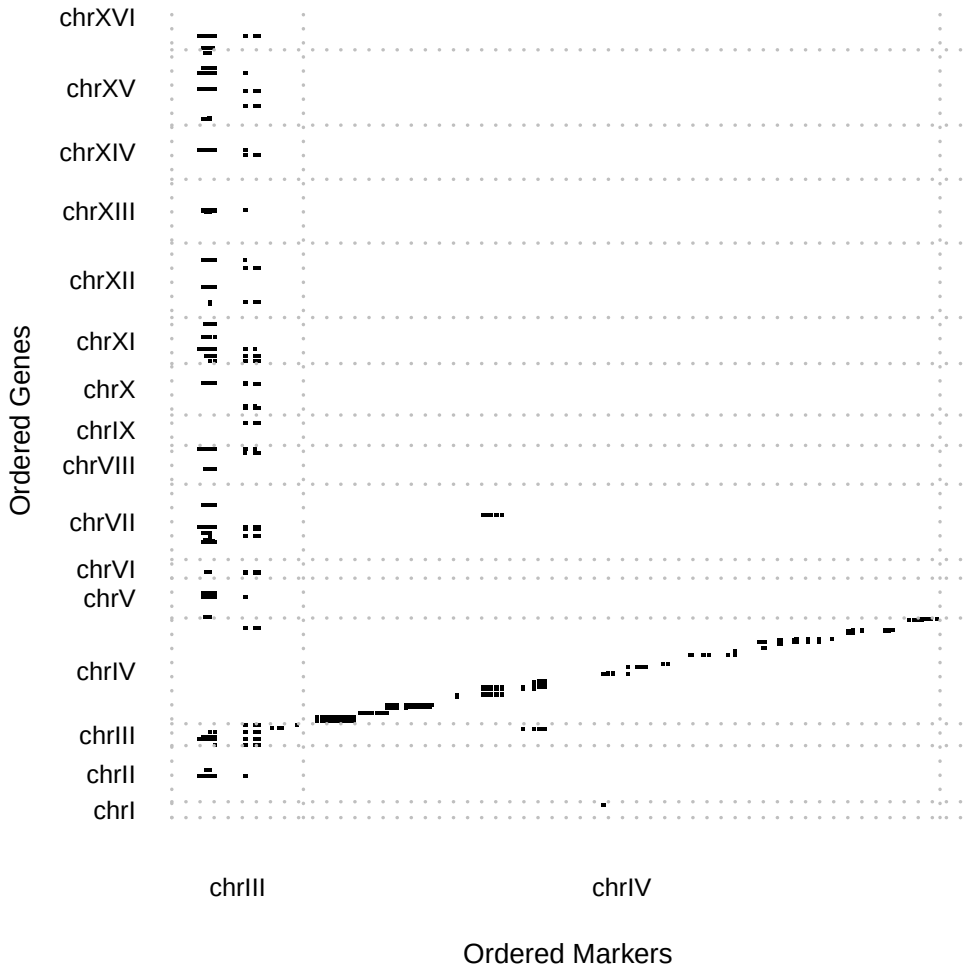
```
> allci$n
```

```
[1] NA
```

Mapping eQTL associations

- Plot a map of the significant associations with `qpPlotMap()`:

```
> par(mar=c(4, 5, 1, 1))
> sel.pairs <- qpPlotMap(allci$p.value, sacCer3markerPos, sacCer3genePos,
+                        sacCer3chrLen, ylab="", cex=2, p.value=0.01)
> mtext("Ordered Genes", 2, line=4)
```



Mapping eQTL associations

- Function `qpPlotMap()` returns pairs of genetic marker and gene meeting the adjusted *p*-value cutoff:

```
> dim(sel.pairs)
[1] 1985    4
> head(sel.pairs, n=2)
            i       j          pval      adjpval
1 6829_at_x01 YKL209C 1.024931e-54 1.698352e-48
2 6829_at_x02 YKL209C 4.819937e-54 7.986823e-48
```

- Explore genetic hotspots:

```
> markerlinks <- sapply(split(sel.pairs[,3], sel.pairs[,1]), length)
> head(sacCer3markerPos[names(sort(markerlinks, decreasing=TRUE)), ], n=10)
           chromosome position
6768_at_x05          3   175816
6768_at_x06          3   175810
2438_at_x03          3   177858
6768_at_x07          3   175807
6829_at_x01          3   201174
6829_at_x02          3   201175
6909_at_x10          3    92248
6909_at_x03          3    92014
2435_at_x00          3    90413
2435_at_x04          3    90677
```

- Take three hotspot markers at most chromosomal upstream positions:

```
> markerhotspots <- c("6768_at_x07", "6829_at_x01", "2435_at_x00")
```

Inferring eQTL networks

- We will use the previously selected genetic interactions and transcription factor annotations to restrict the network inference problem.
- Load transcriptional regulatory interactions documented in yeast from `http://www.yeastract.com`:

```
> yeastRegInt <- read.table("qpgraphWorkshop/yeastractRegTwoColTable.tsv",
+                           colClasses="character")
> head(yeastRegInt, n=2)
```

```
    V1      V2
1 Abf1 YKL112w
2 Abf1 YAL054c
```

- Extract transcription factor gene identifiers and select those present in our data. We need to load first the organism level package for yeast:

```
> library("org.Sc.sgd.db")
> tfIDs <- toupper(unique(yeastRegInt[, 1]))
> tfIDs <- unlist(mget(tfIDs, revmap(org.Sc.sgdGENENAME), ifnotfound=NA))
> tfIDs[is.na(tfIDs)] <- names(tfIDs)[is.na(tfIDs)]
> tfIDs <- tfIDs[tfIDs %in% rownames(sacCer3genePos)]
> tfIDs <- intersect(tfIDs, sel.pairs$j)
> length(tfIDs)
```

```
[1] 5
```

Inferring eQTL networks

- Estimate NRRs with $q = 20$ (ideally we would use something like $q = \{25, 50, 75, 100\}$ and average them using `qpAvgNrr()`):

```
> nrr <- qpNrr(BremGGsacCer3chr3n4, q=20, I=rownames(sacCer3markerPos),
+              pairup.i=c(markerhotspots, tfIDs), pairup.j=unique(sel.pairs$j),
+              restrict.Q=rownames(sacCer3genePos), verbose=FALSE)
> dim(nrr)
```

```
[1] 6485 6485
```

- Subset the resulting matrix to the involved features to gain speed:

```
> nrr <- nrr[c(markerhotspots, tfIDs, unique(sel.pairs$j)),
+            c(markerhotspots, tfIDs, unique(sel.pairs$j))]
> dim(nrr)
```

```
[1] 129 129
```

- Select the qp-graph with strongest associations (NRR=0). These are marker-gene and gene-gene pairs which rejected every of the default $nTest = 100$ conditional independence tests on randomly selected conditioning subsets of $q = 20$ genes.

```
> g <- qpGraph(nrr, threshold=0, return.type="graphNEL")
> g
```

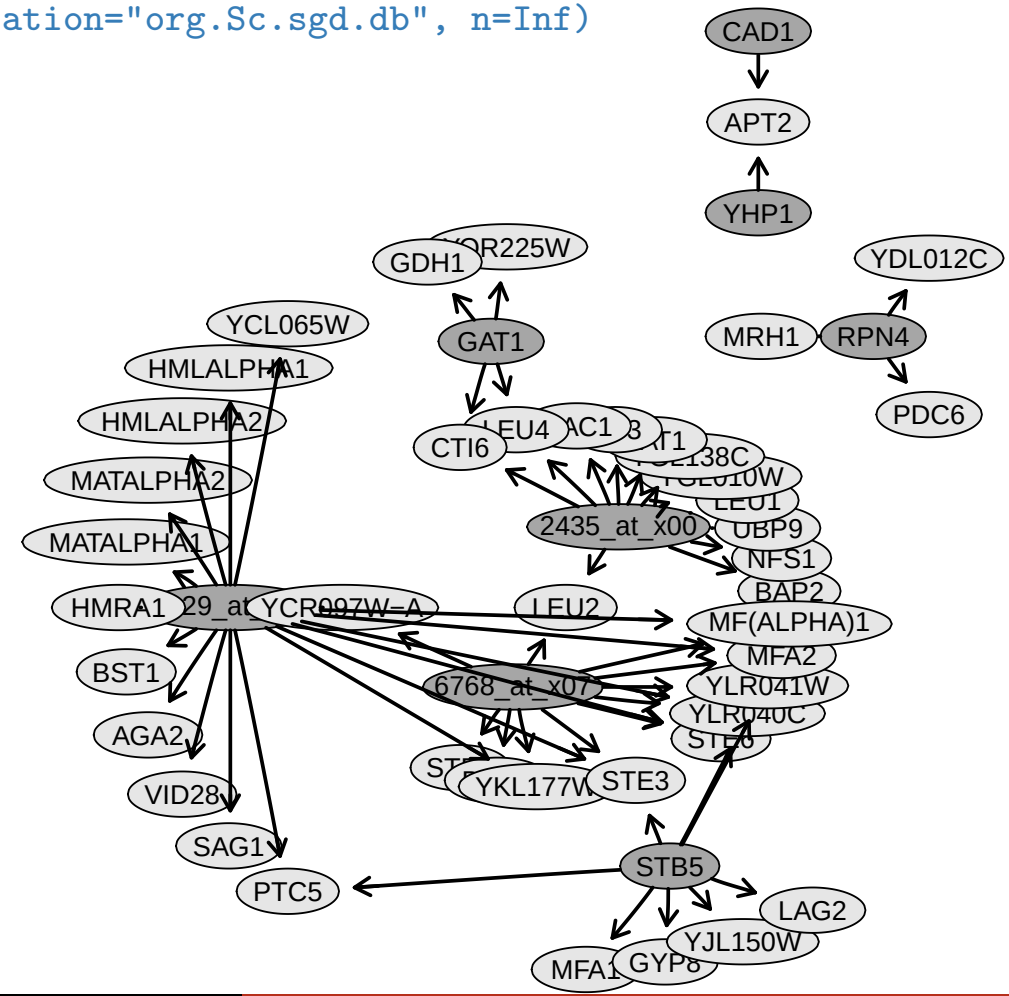
```
A graphNEL graph with undirected edges
Number of Nodes = 51
Number of Edges = 58
```

Inferring eQTL networks

- Plot the network and explore genetic interactions with 2435_at_x00:

```
> qpPlotNetwork(g, pairup.i=c(markerhotspots, tfIDs),
+                pairup.j=unique(sel.pairs$j), annotation="org.Sc.sgd.db")
> chr3hs <- qpTopPairs(refGraph=g, pairup.i="2435_at_x00", pairup.j=nodes(g),
+                      annotation="org.Sc.sgd.db", n=Inf)
> chr3hs[, 3:4]
```

	iSymbol	jSymbol
1	2435_at_x00	BAP2
2	2435_at_x00	NFS1
3	2435_at_x00	LEU2
4	2435_at_x00	UBP9
5	2435_at_x00	LEU1
6	2435_at_x00	YGL010W
7	2435_at_x00	YGL138C
8	2435_at_x00	BAT1
9	2435_at_x00	ILV3
10	2435_at_x00	OAC1
11	2435_at_x00	LEU4
12	2435_at_x00	CTI6



Inferring eQTL networks

- Examine some of the interactions around hotspot 2435_at_x00. We build a *GRanges* object with the positions of the hotspot and the interacting genes and export it into a BED file.

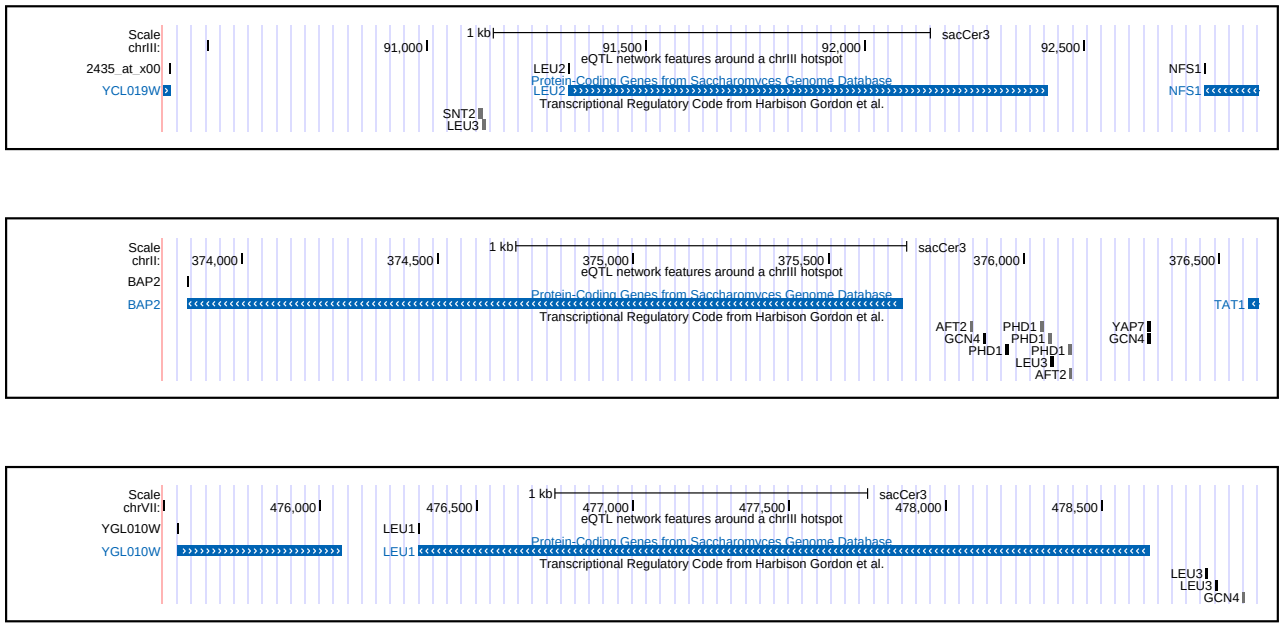
```
> library(rtracklayer)
> chr <- c(paste0("chr", as.roman(sacCer3markerPos[unique(chr3hs$i), 1])),
+          paste0("chr", as.roman(sacCer3genePos[unique(chr3hs$j), 1])))
> rng <- IRanges(c(sacCer3markerPos[unique(chr3hs$i), 2],
+                  sacCer3genePos[unique(chr3hs$j), 2]), width=1,
+                names=c(unique(chr3hs$iSymbol), unique(chr3hs$jSymbol)))
> gr <- GRangesForUCSCGenome("sacCer3", chr, rng, strand="+")
> head(gr)
```

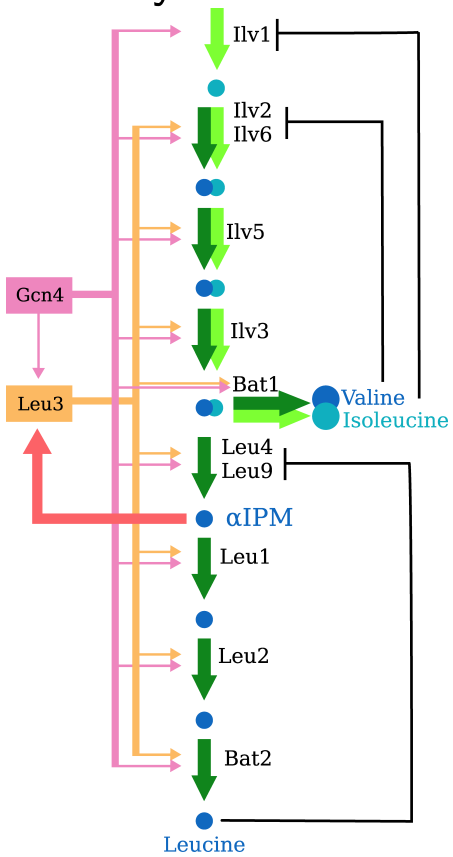
```
GRanges with 6 ranges and 0 elementMetadata cols:
             seqnames           ranges strand
                <Rle>        <IRanges>  <Rle>
  2435_at_x00   chrIII [ 90413,  90413]      +
         BAP2    chrII [373861, 373861]      +
         NFS1   chrIII [ 92777,  92777]      +
         LEU2   chrIII [ 91324,  91324]      +
         UBP9     chrV [355466, 355466]      +
         LEU1   chrVII [476313, 476313]      +
  ---
  seqlengths:
      chrI   chrII  chrIII   chrIV   chrIX ...  chrXIV   chrXV  chrXVI    chrM
    230218 813184  316620 1531933  439888 ...  784333 1091291  948066   85779
```

```
> export(gr, "chr3hsint.bed")
```

Inferring eQTL networks

- The examination of the genomic context at the UCSC genome browser, previously uploading the exported BED file, produces hits pointing to *cis* and *trans* associations involved in the leucine biosynthesis pathway:





Chin *et al.*,

PLos Biol, 2008, Fig. 1

Integrating phenotype information

- The integration of phenotypic information into the network inference can be done by using *ExpressionSet* and *smlSet* objects for the input data.

- Let us consider again the E. coli data set stored as an *ExpressionSet* object in `eset100` and particularly its phenotypic information:

```
> head(pData(eset100)[, 1:3])
```

```
          Strain GrowthProtocol GenotypeVariation
GSM18235 mutant        aerobic              appY
GSM18236 mutant        aerobic              appY
GSM18237 mutant        aerobic              appY
GSM18246 mutant        aerobic              arcA
GSM18247 mutant        aerobic              arcA
GSM18248 mutant        aerobic              arcA
```

- Estimate again NRR values this time including the phenotypic variable `Strain` and adjusting for `GrowthProtocol`:

```
> nrr.q5.st.gp <- qpNrr(eset100, q=5, pairup.i=c(TFgenes, "Strain"),
+                       pairup.j=featureNames(eset100),
+                       fix.Q="GrowthProtocol", verbose=FALSE)
```

Integrating phenotype information

- Calculate the precision-recall curve and select a network at a nominal 50% precision.

```
> nrr.q5.st.gp.pr <- qpPrecisionRecall(nrr.q5.st.gp, refGraph=regulon100[, 3:4],
+                                      pairup.i=TFgenes, pairup.j=featureNames(eset100),
+                                      decreasing=FALSE)
> nrr.thr.q5.st.gp <- qpPRscoreThreshold(nrr.q5.st.gp.pr, level=0.5,
+                                        recall.level=FALSE, max.score=0)
> nrr.g.q5.st.gp <- qpGraph(nrr.q5.st.gp, threshold=nrr.thr.q5.st.gp,
+                           return.type="graphNEL")
> nrr.g.q5.st.gp
```

```
A graphNEL graph with undirected edges
Number of Nodes = 57
Number of Edges = 59
```

- Perform now a differential expression analysis using *limma* to search for genes changing between the mutant and the wild strain adjusting for the growth protocol:

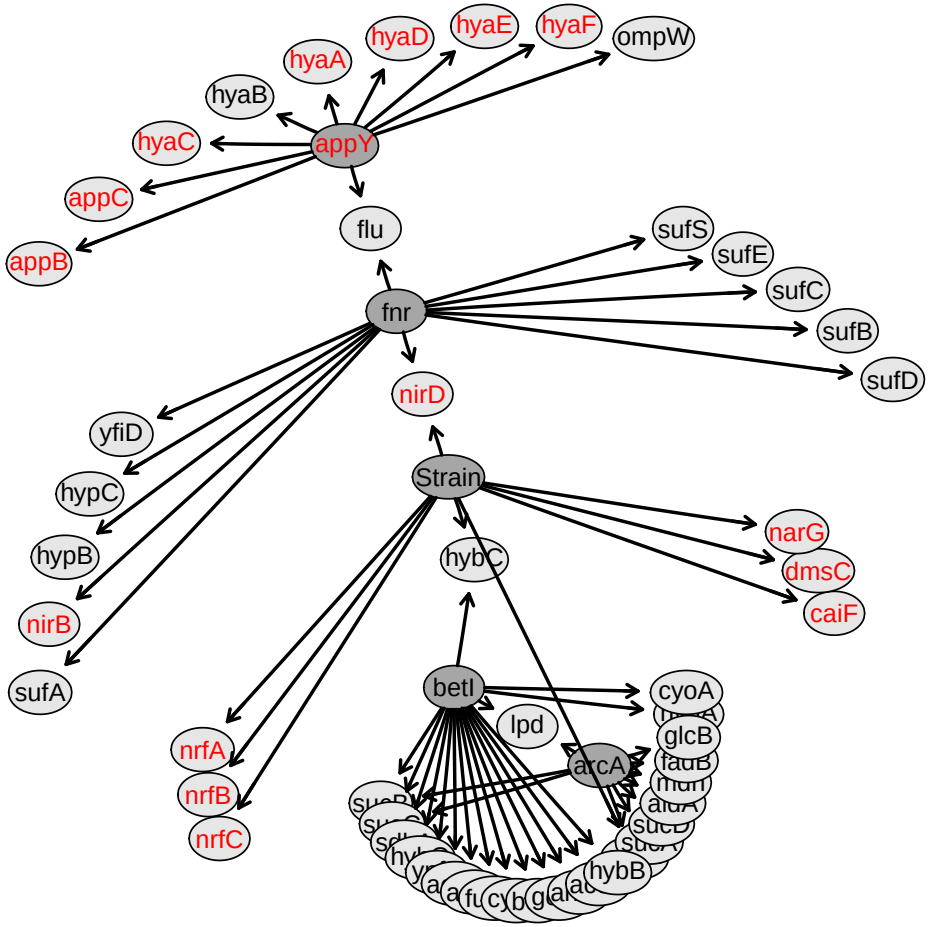
```
> library(limma)
> design <- model.matrix(~ factor(Strain) + factor(GrowthProtocol), data=eset100)
> fit <- lmFit(eset100, design)
> fit <- eBayes(fit)
> deGenes <- topTable(fit, coef=2, p.value=0.1, n=Inf)$ID
> length(deGenes)
```

```
[1] 22
```

Integrating phenotype information

- Plot the resulting network highlighting the previously found differentially expressed genes:

```
> qpPlotNetwork(nrr.g.q5.st.gp, pairup.i=c(TFgenes, "Strain"),
+               pairup.j=nodes(nrr.g.q5.st.gp),
+               highlight=deGenes, annotation="org.EcK12.eg.db")
```



Functionality we have not seen

- A meta-analysis approach to identify common interactions with the function `qpGenNrr()` (Roverato and Castelo, 2012).
- A model-based approach to the Selection of networks using partial correlation estimation after dimension reduction with the NRR.
- Performing calculations in parallel with MPI via *snow*/*parallel* and the argument `clusterSize`.
- Integrating genotype and phenotype information through *smlSet* objects. It works essentially the same as when integrating phenotypic information from *ExpressionSet* objects but the dimension of genotype data poses computational challenges that *qpgraph* is not addressing yet.

Concluding remarks

- The *qpgraph* package implements a principled statistical methodology for inferring networks from high-throughput genomics data.
- This methodology is based on graphical model theory that leads to powerful and exact tests of conditional independence for pure continuous and mixed, continuous and discrete, data.
- Although it is computationally demanding, the operations through the interacting pairs are independent allowing for parallel execution with MPI and nearly linear speed-ups.
- It currently lacks the implementation of appropriate data structures for a more efficient and easy use of the methods that the package provides. Improvements in this direction should appear in the next releases.
- In the twitter account `@robertclab` we will be posting when a release update with a bugfix of *qpgraph* is available via `biocLite()`.

References

- Brem *et al.* The landscape of genetic complexity across 5,700 gene expression traits in yeast. *PNAS*, 102:1572–1577, 2005.
- Castelo and Roverato. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621–2650, 2006.
- Castelo and Roverato. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *J. Comput. Biol.*, 16:213–227.
- Covert *et al.* Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429:92–96, 2004.
- Dobra *et al.* Sparse graphical models for exploring gene expression data. *J. Mult. Anal.*, 90:196–212, 2004.
- Dykstra *et al.* Establishing the positive definiteness of the sample covariance matrix. *Ann. Math. Statist.*, 41:2153–2154, 1970.

References

- Edwards. *Introduction to graphical modelling*, Springer, 2000.
- Friedman, Hastie and Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441, 2008.
- Gama-Castro *et al.* RegulonDB version 7.0: transcriptional regulation of *Escherichia coli* K-12 integrated within genetic sensory response units. *Nucl. Acids Res.*, 39(suppl 1):D98–D105, 2011.
- Lauritzen. *Graphical Models*, Oxford University Press, 1996.
- Roverato and Castelo. Learning undirected graphical models from multiple datasets with the generalized non-rejection rate. *Int. J. Approx. Reas., in press*.
- Schäfer and Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Stat. Appl. Genet. Mol. Biol.*, 4:art32, 2005.
- Segal *et al.* Learning module networks. *J. Mach. Learn. Res.*, 6:557–588, 2005.

Session information

```
> toLatex(sessionInfo())
```

- R version 2.15.0 (2012-03-30), `x86_64-apple-darwin9.8.0`
- Locale: `C/UTF-8/C/C/C/C`
- Base packages: base, datasets, grDevices, graphics, grid, methods, stats, utils
- Other packages: AnnotationDbi 1.19.28, Biobase 2.17.6, BiocGenerics 0.3.0, DBI 0.2-5, GenomicRanges 1.9.39, IRanges 1.15.24, RSQLite 0.11.1, Rgraphviz 1.35.2, graph 1.35.1, limma 3.13.13, mvtnorm 0.9-9992, org.EcK12.eg.db 2.7.1, org.Sc.sgd.db 2.7.1, qpgraph 1.13.30, rtracklayer 1.17.13
- Loaded via a namespace (and not attached): BSgenome 1.25.3, Biostrings 2.25.8, GGBase 3.19.6, Matrix 1.0-6, RCurl 1.91-1, Rsamtools 1.9.24, XML 3.9-4, annotate 1.35.3, bitops 1.0-4.1, genefilter 1.39.0, lattice 0.20-6, snpStats 1.7.3, splines 2.15.0, stats4 2.15.0, survival 2.36-14, tools 2.15.0, xtable 1.7-0, zlibbioc 1.3.0