Inferring Causal Phenotype Networks from Segregating Populations

Elias Chaibub Neto
chaibub@stat.wisc.edu

Statistics Department, University of Wisconsin - Madison

July 15, 2008

Overview

- Introduction
- Description of the approach
  - PC algorithm.
  - QDG algorithm.
- Remarks
- Performance on simulations.
- Real data example.
- Future work.

Introduction

- Our objective is to learn metabolic pathways from data.
- We represent these pathways by directed networks composed by transcripts, metabolites and clinical traits.
- These phenotypes are quantitative in nature, and can be analyzed using quantitative genetics techniques.

Introduction

- In particular, we use Quantitative Trait Loci (QTL) mapping methods to identify genomic regions affecting the phenotypes.

- Since variations in the genotypes (QTLs) cause variations in the phenotypes, but not the other way around, we can unambiguously determine the causal direction

$$\text{QTL} \Rightarrow \text{phenotype}$$

- Knowing that a QTL causally affects a phenotype will allow us to infer causal direction between phenotypes.

Introduction

- We assume that a set of QTLs associated with the phenotypes has been previously determined.

- We assume linear relationships between phenotypes and between phenotypes and QTLs.

Introduction

Our procedure is composed of two parts:

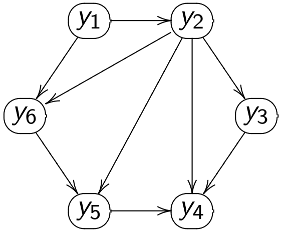
1. First we infer the skeleton of the causal model (phenotype network) using the PC-algorithm.
2. Orient the edges in the skeleton using the QDG algorithm.

PC algorithm

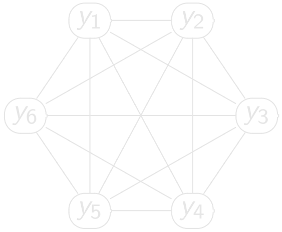
- Causal discovery algorithm developed by Spirtes et al 1993.
- It is composed of two parts:
  1. Infers the skeleton of the causal model.
  2. Partially orient the graph (orient some but not all edges).
- We are only interested in the first part (the "PC skeleton algorithm"). We do **not** use the PC algorithm to edge orientation (we use the QDG algorithm instead).

Step 1 (PC skeleton algorithm)

Suppose the true network describing the causal relationships between six transcripts is



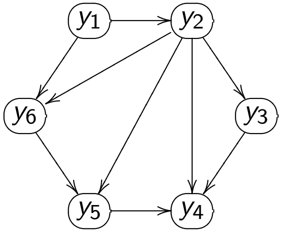
The PC-algorithm starts with the complete undirected graph



and progressively eliminates edges based on conditional independence tests.

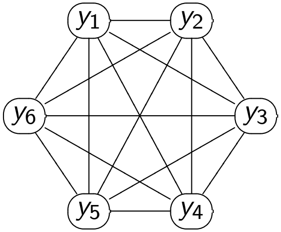
Step 1 (PC skeleton algorithm)

Suppose the true network describing the causal relationships between six transcripts is



$$\begin{tikzcd} & y_1 \arrow[r] \arrow[dl] & y_2 \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\ y_6 \arrow[dr] & & & y_3 \arrow[dl] \\ & y_5 \arrow[r] & y_4 & \end{tikzcd}$$

The PC-algorithm starts with the complete undirected graph



and progressively eliminates edges based on conditional independence tests.

Step 1 (PC skeleton algorithm)

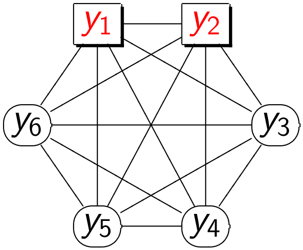
The algorithm performs several rounds of conditional independence tests of increasing order.

It starts with all zero order tests, then performs all first order, second order . . .

- Notation: $\perp\!\!\!\perp \equiv$ independence. We read $i \perp\!\!\!\perp j \mid k$ as *i is conditionally independent from j given k*.
- Remark: in the Gaussian case zero partial correlation implies conditional independence, thus

$$i \perp\!\!\!\perp j \mid k \;\; \Leftrightarrow \;\; cor(i, j \mid k) = 0 \;\; \Rightarrow \;\; \text{drop } (i, j) \text{ edge}$$

Example (order 0)

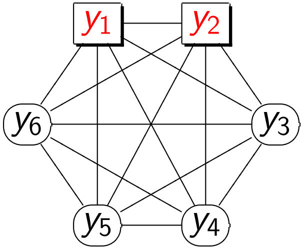


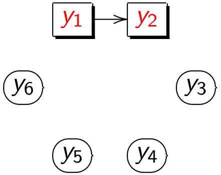
$$1 \perp\!\!\!\perp 2$$

vs

$$1 \not\!\perp\!\!\!\perp 2$$

Example (order 0)





direct effect of y_1 on y_2

$$1 \perp\!\!\!\perp 2$$

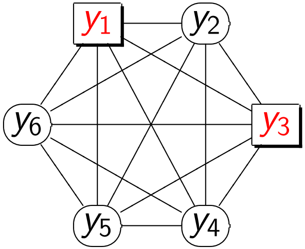
vs

$$1 \not\perp\!\!\!\perp 2$$

$$1 \not\perp\!\!\!\perp 2$$

keep edge and move to next one

Example (order 0)

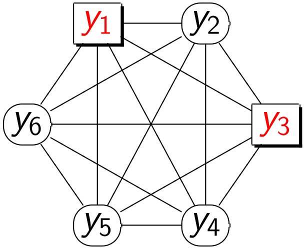


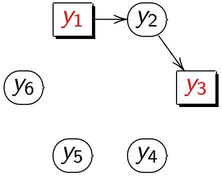
$$1 \perp\!\!\!\perp 3$$

vs

$$1 \not\perp\!\!\!\perp 3$$

Example (order 0)





indirect effect of y_1 on y_3

$$1 \perp\!\!\!\perp 3$$

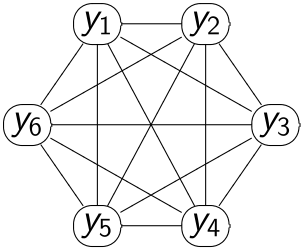
vs

$$1 \not\perp\!\!\!\perp 3$$

$$1 \not\perp\!\!\!\perp 3$$

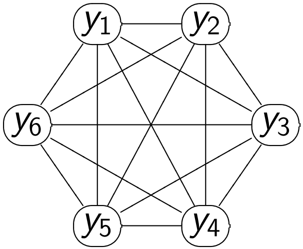
keep edge and move to next one

Example (order 0)



After all zero order conditional independence tests.

Example (order 0)



After all zero order conditional independence tests.

The algorithm then moves to first order conditional independence tests.

Example (order 1)

For any edge (i, j) the algorithm tests whether

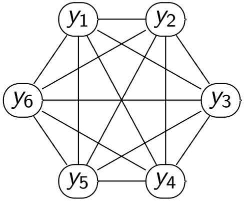
$$i \perp\!\!\!\perp j \mid k$$

for all

$$k \in A(i) \setminus j$$

where $A(i)$ represent the set of nodes adjacent to node i.

For example,



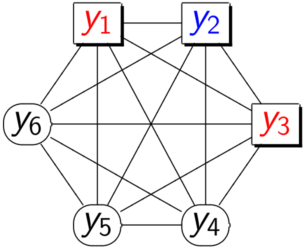
$$A(1) \setminus 2 = \{3, 4, 5, 6\}$$

and the algorithm tests whether

$$1 \perp\!\!\!\perp 2 \mid 3 \qquad 1 \perp\!\!\!\perp 2 \mid 4$$

$$1 \perp\!\!\!\perp 2 \mid 5 \qquad 1 \perp\!\!\!\perp 2 \mid 6$$

Example (order 1)



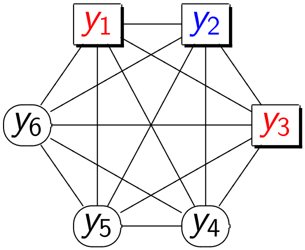
$$A(1) \setminus 2 = \{2, 4, 5, 6\}$$

$$1 \perp\!\!\!\perp 3 \mid 2$$

vs

$$1 \not\perp\!\!\!\perp 3 \mid 2$$

Example (order 1)

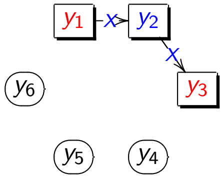


$$A(1) \setminus 2 = \{2, 4, 5, 6\}$$

$$1 \perp\!\!\!\perp 3 \mid 2$$

vs

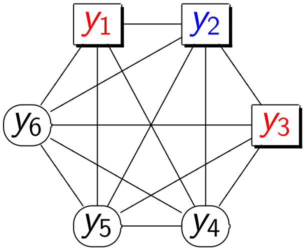
$$1 \not\perp\!\!\!\perp 3 \mid 2$$



y_2 d-separates y_1 from y_3

$$1 \perp\!\!\!\perp 3 \mid 2$$

Example (order 1)

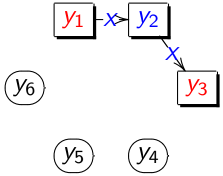


$$A(1) \setminus 2 = \{2, 4, 5, 6\}$$

$$1 \perp\!\!\!\perp 3 \mid 2$$

vs

$$1 \not\perp\!\!\!\perp 3 \mid 2$$



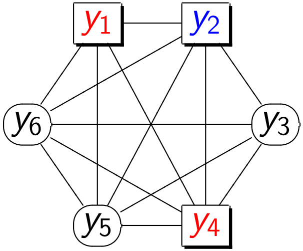
y_2 d-separates y_1 from y_3

$$1 \perp\!\!\!\perp 3 \mid 2$$

drop edge

move to next edge

Example (order 1)



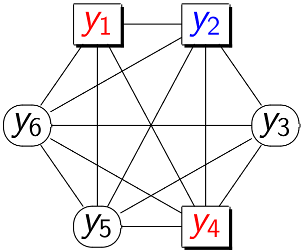
$$A(1) \setminus 4 = \{2, 5, 6\}$$

$$1 \perp\!\!\!\perp 4 \mid 2$$

vs

$$1 \not\!\perp\!\!\!\perp 4 \mid 2$$

Example (order 1)

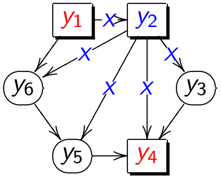


$$A(1) \setminus 4 = \{2, 5, 6\}$$

$$1 \perp\!\!\!\perp 4 \mid 2$$

vs

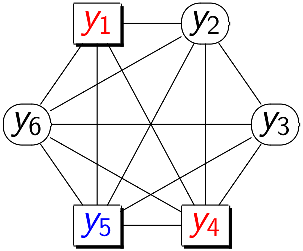
$$1 \not\perp\!\!\!\perp 4 \mid 2$$



$$1 \not\perp\!\!\!\perp 4 \mid 2$$

keep edge
move to next conditioning set

Example (order 1)



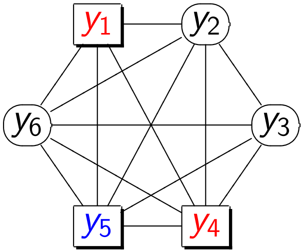
$$A(1) \setminus 4 = \{2, 5, 6\}$$

$$1 \perp\!\!\!\perp 4 \mid 5$$

vs

$$1 \not\!\perp\!\!\!\perp 4 \mid 5$$

Example (order 1)

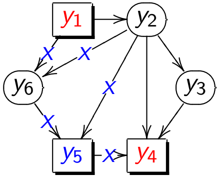


$$A(1) \setminus 4 = \{2, 5, 6\}$$

$$1 \perp\!\!\!\perp 4 \mid 5$$

vs

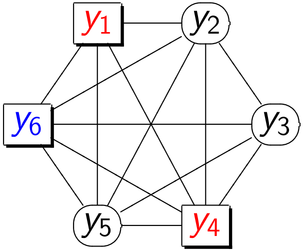
$$1 \not\perp\!\!\!\perp 4 \mid 5$$



$$1 \not\perp\!\!\!\perp 4 \mid 5$$

keep edge
move to next conditioning set

Example (order 1)



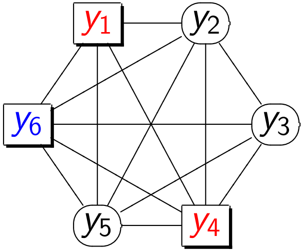
$$A(1) \setminus 4 = \{2, 5, 6\}$$

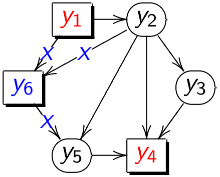
$$1 \perp\!\!\!\perp 4 \mid 6$$

vs

$$1 \not\!\perp\!\!\!\perp 4 \mid 6$$

Example (order 1)





$$A(1) \setminus 4 = \{2, 5, 6\}$$

$$1 \not\perp\!\!\!\perp 4 \mid 6$$

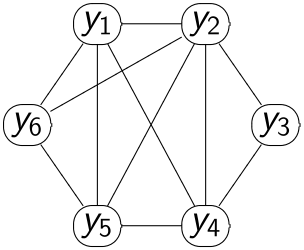
$$1 \perp\!\!\!\perp 4 \mid 6$$

vs

$$1 \not\perp\!\!\!\perp 4 \mid 6$$

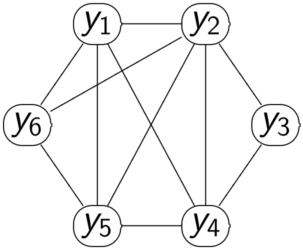
keep edge
move to next edge

Example (order 1)



After all first order conditional independence tests.

Example (order 1)



After all first order conditional independence tests.

The algorithm then moves to second order conditional independence tests.

Example (order 2)

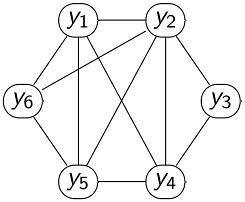
For any edge (i, j) the algorithm tests whether

$$i \perp\!\!\!\perp j \mid k, l.$$

for all

$$(k, l) \in A(i) \setminus j$$

For example,



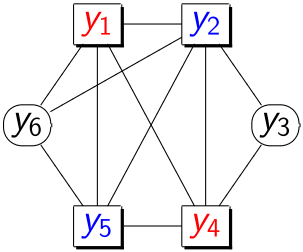
$$A(1) \setminus 2 = \{4, 5, 6\}$$

and the algorithm tests whether

$$1 \perp\!\!\!\perp 2 \mid 4, 5 \qquad 1 \perp\!\!\!\perp 2 \mid 4, 6$$

$$1 \perp\!\!\!\perp 2 \mid 5, 6$$

Example (order 2)



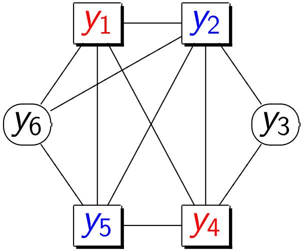
$$A(1) \setminus 4 = \{2, 5, 6\}$$

$$1 \perp\!\!\!\perp 4 \mid 2, 5$$

vs

$$1 \not\!\perp\!\!\!\perp 4 \mid 2, 5$$

Example (order 2)

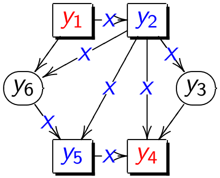


$A(1) \setminus 4 = \{2, 5, 6\}$

$1 \perp\!\!\!\perp 4 \mid 2, 5$

vs

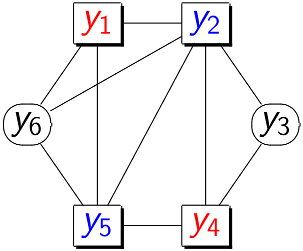
$1 \not\perp\!\!\!\perp 4 \mid 2, 5$

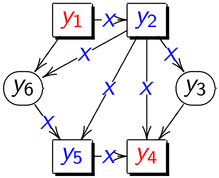


(y_2, y_5) d-separate y_1 from y_4

$1 \perp\!\!\!\perp 4 \mid 2, 5$

Example (order 2)





(y_2, y_5) d-separate y_1 from y_4

$$A(1) \setminus 4 = \{2, 5, 6\}$$

$$1 \perp\!\!\!\perp 4 \mid 2, 5$$

vs

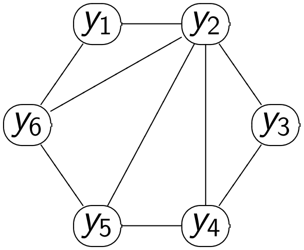
$$1 \not\!\perp\!\!\!\perp 4 \mid 2, 5$$

$$1 \perp\!\!\!\perp 4 \mid 2, 5$$

drop edge

move to next edge

Example (order 2)



After all second order conditional independence tests.

The algorithm than moves to third order, fourth order . . .

It stops when for each pair (i, j) the cardinality of

$$A(i) \setminus j$$

is smaller then the order of the algorithm.

Edge orientation

Consider two traits y_1 and y_2. Our problem is to decide between models:

$$M_1 : \quad \begin{tikzcd} y_1 \arrow[r] & y_2 \end{tikzcd} \qquad M2 : \quad \begin{tikzcd} y_1 & y_2 \arrow[l] \end{tikzcd}$$

Problem: the above models are likelihood equivalent,

$$f(y_1)f(y_2 \mid y_1) = f(y_1, y_2) = f(y_2)f(y_1 \mid y_2) \; .$$

Edge orientation

However, models

$$\begin{tikzcd}
q_{11} \arrow[dr] & & & q_{21} \arrow[dl] \\
\vdots & y_1 \arrow[r] & y_2 & \vdots \\
q_{1k} \arrow[ur] & & & q_{2l} \arrow[ul]
\end{tikzcd}
\qquad
\begin{tikzcd}
q_{11} \arrow[dr] & & & q_{21} \arrow[dl] \\
\vdots & y_1 & y_2 \arrow[l] & \vdots \\
q_{1k} \arrow[ur] & & & q_{2l} \arrow[ul]
\end{tikzcd}$$

are *not* likelihood equivalent because

$$f(\mathbf{q}_1)f(y_1 \mid \mathbf{q}_1)f(y_2 \mid y_1, \mathbf{q}_2)f(\mathbf{q}_2)$$
$$\neq$$
$$f(\mathbf{q}_2)f(y_2 \mid \mathbf{q}_2)f(y_1 \mid y_2, \mathbf{q}_1)f(\mathbf{q}_1)$$

Edge orientation

We perform model selection using a direction LOD score

$$LOD = \log_{10} \left\{ \frac{\prod_{i=1}^{n} f(y_{1i} \mid \mathbf{q}_{1i}) f(y_{2i} \mid y_{1i}, \mathbf{q}_{2i})}{\prod_{i=1}^{n} f(y_{2i} \mid \mathbf{q}_{2i}) f(y_{1i} \mid y_{2i}, \mathbf{q}_{1i})} \right\}$$

where $f()$ represents the predictive density, that is, the sampling model with parameters replaced by the corresponding maximum likelihood estimates.

QDG algorithm

QDG stands for QTL-directed dependency graph.

The QDG algorithm is composed of 7 steps:

1. Get the causal skeleton (with the PC skeleton algorithm).
2. Use QTLs to orient the edges in the skeleton.
3. Choose a random ordering of edges, and
4. Recompute orientations incorporating causal phenotypes in the models (update the causal model according to changes in directions).
5. Repeat 4 iteratively until no more edges change direction (the resulting graph is one solution).
6. Repeat steps 3, 4, and 5 many times and store all different solutions.
7. Score all solutions and select the graph with best score.

QDG algorithm

QDG stands for QTL-directed dependency graph.
The QDG algorithm is composed of 7 steps:

1. Get the causal skeleton (with the PC skeleton algorithm).
2. Use QTLs to orient the edges in the skeleton.
3. Choose a random ordering of edges, and
4. Recompute orientations incorporating causal phenotypes in the models (update the causal model according to changes in directions).
5. Repeat 4 iteratively until no more edges change direction (the resulting graph is one solution).
6. Repeat steps 3, 4, and 5 many times and store all different solutions.
7. Score all solutions and select the graph with best score.

QDG algorithm

QDG stands for QTL-directed dependency graph.
The QDG algorithm is composed of 7 steps:

1. Get the causal skeleton (with the PC skeleton algorithm).
2. Use QTLs to orient the edges in the skeleton.
3. Choose a random ordering of edges, and
4. Recompute orientations incorporating causal phenotypes in the models (update the causal model according to changes in directions).
5. Repeat 4 iteratively until no more edges change direction (the resulting graph is one solution).
6. Repeat steps 3, 4, and 5 many times and store all different solutions.
7. Score all solutions and select the graph with best score.

QDG algorithm

QDG stands for QTL-directed dependency graph.
The QDG algorithm is composed of 7 steps:

1. Get the causal skeleton (with the PC skeleton algorithm).
2. Use QTLs to orient the edges in the skeleton.
3. Choose a random ordering of edges, and
4. Recompute orientations incorporating causal phenotypes in the models (update the causal model according to changes in directions).
5. Repeat 4 iteratively until no more edges change direction (the resulting graph is one solution).
6. Repeat steps 3, 4, and 5 many times and store all different solutions.
7. Score all solutions and select the graph with best score.

QDG algorithm

QDG stands for QTL-directed dependency graph.
The QDG algorithm is composed of 7 steps:

1. Get the causal skeleton (with the PC skeleton algorithm).
2. Use QTLs to orient the edges in the skeleton.
3. Choose a random ordering of edges, and
4. Recompute orientations incorporating causal phenotypes in the models (update the causal model according to changes in directions).
5. Repeat 4 iteratively until no more edges change direction (the resulting graph is one solution).
6. Repeat steps 3, 4, and 5 many times and store all different solutions.
7. Score all solutions and select the graph with best score.

QDG algorithm

QDG stands for QTL-directed dependency graph.
The QDG algorithm is composed of 7 steps:

1. Get the causal skeleton (with the PC skeleton algorithm).
2. Use QTLs to orient the edges in the skeleton.
3. Choose a random ordering of edges, and
4. Recompute orientations incorporating causal phenotypes in the models (update the causal model according to changes in directions).
5. Repeat 4 iteratively until no more edges change direction (the resulting graph is one solution).
6. Repeat steps 3, 4, and 5 many times and store all different solutions.
7. Score all solutions and select the graph with best score.

QDG algorithm

QDG stands for QTL-directed dependency graph.
The QDG algorithm is composed of 7 steps:

1. Get the causal skeleton (with the PC skeleton algorithm).
2. Use QTLs to orient the edges in the skeleton.
3. Choose a random ordering of edges, and
4. Recompute orientations incorporating causal phenotypes in the models (update the causal model according to changes in directions).
5. Repeat 4 iteratively until no more edges change direction (the resulting graph is one solution).
6. Repeat steps 3, 4, and 5 many times and store all different solutions.
7. Score all solutions and select the graph with best score.

QDG algorithm

QDG stands for QTL-directed dependency graph.
The QDG algorithm is composed of 7 steps:

1. Get the causal skeleton (with the PC skeleton algorithm).
2. Use QTLs to orient the edges in the skeleton.
3. Choose a random ordering of edges, and
4. Recompute orientations incorporating causal phenotypes in the models (update the causal model according to changes in directions).
5. Repeat 4 iteratively until no more edges change direction (the resulting graph is one solution).
6. Repeat steps 3, 4, and 5 many times and store all different solutions.
7. Score all solutions and select the graph with best score.

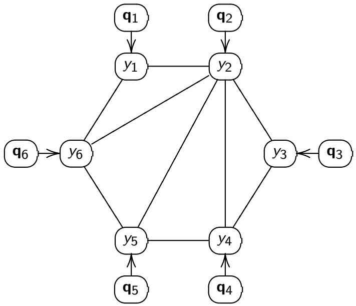
QDG algorithm

QDG stands for QTL-directed dependency graph.
The QDG algorithm is composed of 7 steps:

1. Get the causal skeleton (with the PC skeleton algorithm).
2. Use QTLs to orient the edges in the skeleton.
3. Choose a random ordering of edges, and
4. Recompute orientations incorporating causal phenotypes in the models (update the causal model according to changes in directions).
5. Repeat 4 iteratively until no more edges change direction (the resulting graph is one solution).
6. Repeat steps 3, 4, and 5 many times and store all different solutions.
7. Score all solutions and select the graph with best score.

Step 2

Now suppose that for each transcript we have a set of e-QTLs



Given the QTLs we can distinguish causal direction:

$$\begin{tikzcd} q_1 \arrow[r] & y_1 \arrow[r] & y_2 & q_2 \arrow[l] \end{tikzcd}$$

$$\begin{tikzcd} q_1 \arrow[r] & y_1 & y_2 \arrow[l] & q_2 \arrow[l] \end{tikzcd}$$

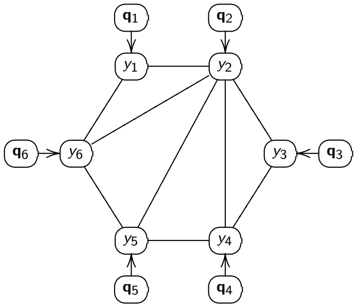
$$\vdots$$

$$\begin{tikzcd} q_5 \arrow[r] & y_5 \arrow[r] & y_6 & q_6 \arrow[l] \end{tikzcd}$$

$$\begin{tikzcd} q_5 \arrow[r] & y_5 & y_6 \arrow[l] & q_6 \arrow[l] \end{tikzcd}$$

Step 2

Now suppose that for each transcript we have a set of e-QTLs



Given the QTLs we can distinguish causal direction:

$$\begin{tikzcd}\mathbf{q}_1 \arrow[r] & y_1 \arrow[r] & y_2 & \mathbf{q}_2 \arrow[l]\end{tikzcd}$$

$$\begin{tikzcd}\mathbf{q}_1 \arrow[r] & y_1 & y_2 \arrow[l] & \mathbf{q}_2 \arrow[l]\end{tikzcd}$$

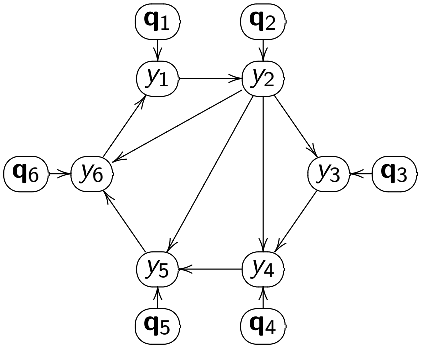
$$\vdots$$

$$\begin{tikzcd}\mathbf{q}_5 \arrow[r] & y_5 \arrow[r] & y_6 & \mathbf{q}_6 \arrow[l]\end{tikzcd}$$

$$\begin{tikzcd}\mathbf{q}_5 \arrow[r] & y_5 & y_6 \arrow[l] & \mathbf{q}_6 \arrow[l]\end{tikzcd}$$

Steps 2 and 3

First estimate of the causal model (DG_0)



(using only QTLs to infer causal direction)

In step 3 we randomly choose an ordering of all edges in DG_0. Say,

$y_2 - y_3$ $\quad$ $y_1 - y_6$ $\quad$ $y_2 - y_5$

$y_1 - y_2$ $\quad$ $y_3 - y_4$ $\quad$ $y_2 - y_6$

$y_2 - y_4$ $\quad$ $y_5 - y_6$ $\quad$ $y_4 - y_5$

In step 4 we recompute the directions including other transcripts as covariates in the models (following the above ordering).

Steps 2 and 3

First estimate of the causal model (DG_0)

$$\begin{tikzcd}
 & & \mathbf{q}_1 \arrow[d] & \mathbf{q}_2 \arrow[d] & & \\
 & & y_1 \arrow[r] & y_2 \arrow[dll] \arrow[ddl] \arrow[dd] \arrow[dr] & & \\
\mathbf{q}_6 \arrow[r] & y_6 \arrow[ur] & & & y_3 \arrow[dl] & \mathbf{q}_3 \arrow[l] \\
 & & y_5 \arrow[ul] & y_4 \arrow[l] & & \\
 & & \mathbf{q}_5 \arrow[u] & \mathbf{q}_4 \arrow[u] & &
\end{tikzcd}$$

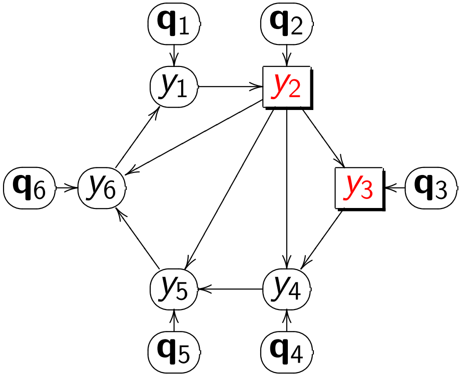
(using only QTLs to infer causal direction)

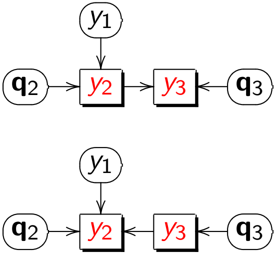
In step 3 we randomly choose an ordering of all edges in DG_0. Say,

$$\begin{tikzcd}
y_2 \arrow[r, dash] & y_3 & y_1 \arrow[r, dash] & y_6 & y_2 \arrow[r, dash] & y_5 \\
y_1 \arrow[r, dash] & y_2 & y_3 \arrow[r, dash] & y_4 & y_2 \arrow[r, dash] & y_6 \\
y_2 \arrow[r, dash] & y_4 & y_5 \arrow[r, dash] & y_6 & y_4 \arrow[r, dash] & y_5
\end{tikzcd}$$

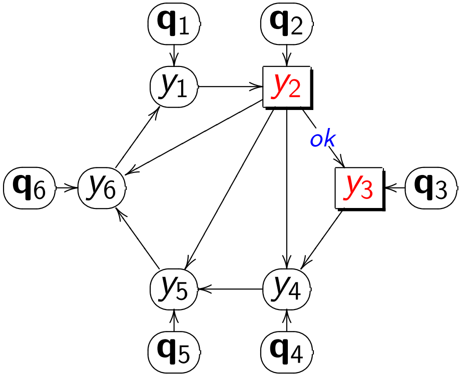
In step 4 we recompute the directions including other transcripts as covariates in the models (following the above ordering).

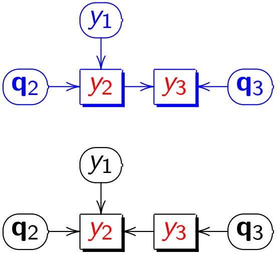
Steps 4 and 5 (first iteration)



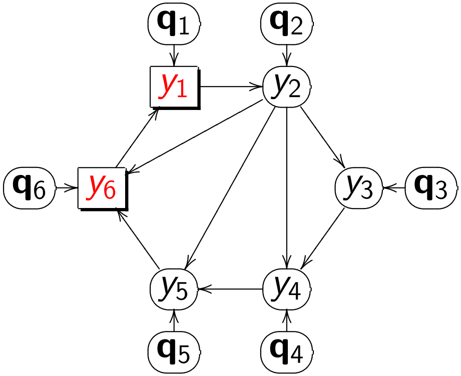


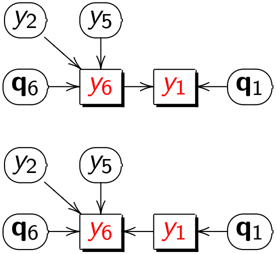
Steps 4 and 5 (first iteration)



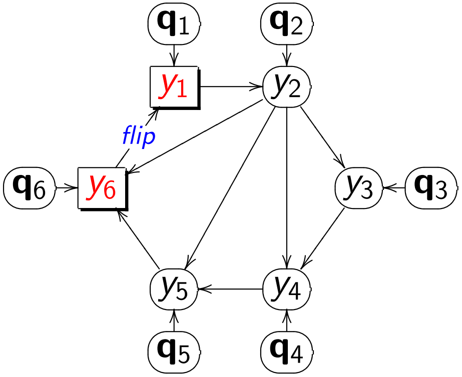


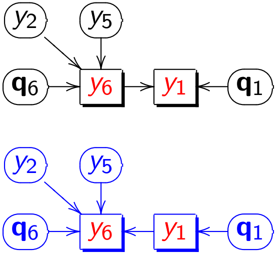
Steps 4 and 5 (first iteration)



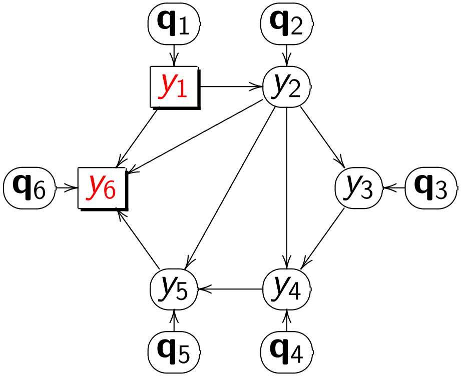


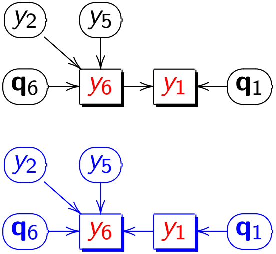
Steps 4 and 5 (first iteration)



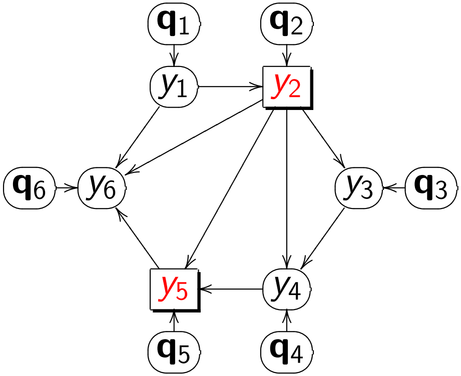


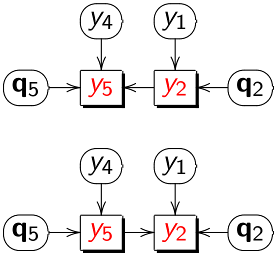
Steps 4 and 5 (first iteration)



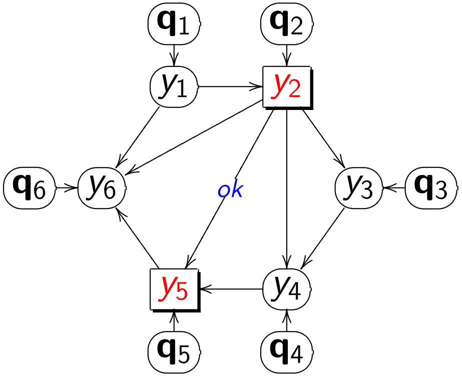


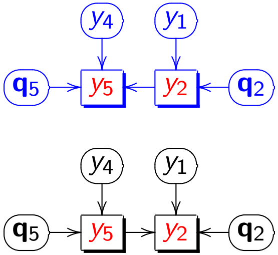
Steps 4 and 5 (first iteration)





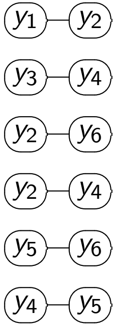
Steps 4 and 5 (first iteration)





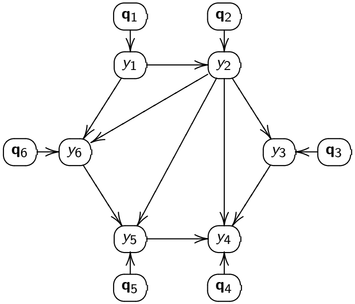
Steps 4 and 5 (first iteration)

And so on until the algorithm recheck the directions for all remaining ordered edges.


$$\begin{tikzcd}
y_1 \arrow[r, dash] & y_2 \\
y_3 \arrow[r, dash] & y_4 \\
y_2 \arrow[r, dash] & y_6 \\
y_2 \arrow[r, dash] & y_4 \\
y_5 \arrow[r, dash] & y_6 \\
y_4 \arrow[r, dash] & y_5
\end{tikzcd}$$

Steps 4 and 5 (first iteration)

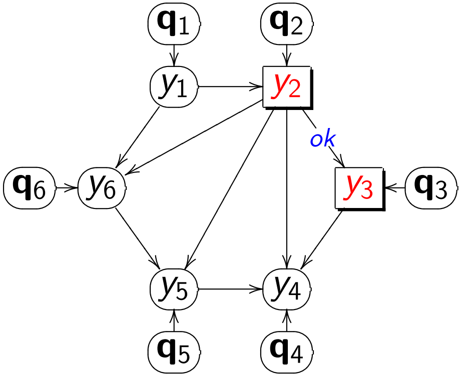
Suppose the updated causal model after the first iteration (DG_1) is

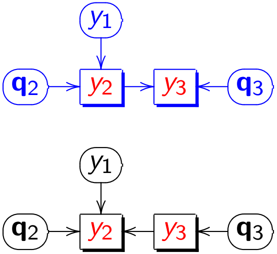


$$\begin{tikzcd}
 & \mathbf{q}_1 \arrow[d] & \mathbf{q}_2 \arrow[d] & & \\
 & y_1 \arrow[r] \arrow[dl] & y_2 \arrow[dll] \arrow[ddl] \arrow[dd] \arrow[dr] & & \\
\mathbf{q}_6 \arrow[r] & y_6 \arrow[d] & & y_3 \arrow[dl] & \mathbf{q}_3 \arrow[l] \\
 & y_5 \arrow[r] & y_4 & & \\
 & \mathbf{q}_5 \arrow[u] & \mathbf{q}_4 \arrow[u] & &
\end{tikzcd}$$

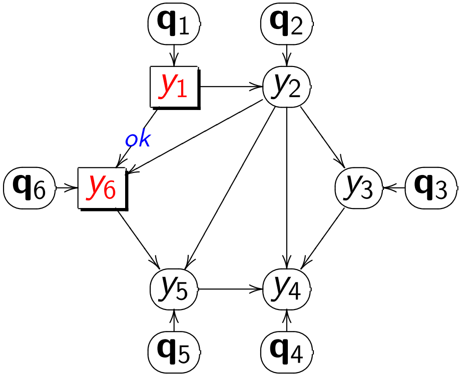
Since some arrows changed direction ($DG_1 \neq DG_0$), the algorithm goes for another round of re-computations.

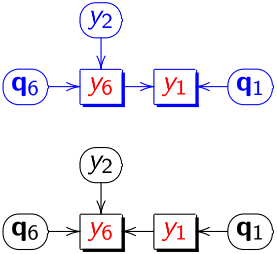
Steps 4 and 5 (second iteration)



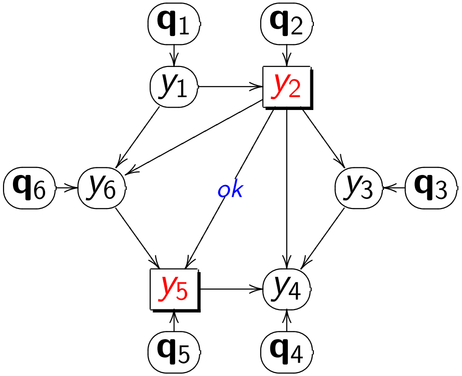


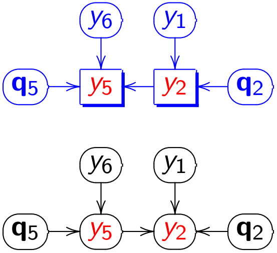
Steps 4 and 5 (second iteration)





Steps 4 and 5 (second iteration)





Steps 4 and 5 (second iteration)

And so on . . .

If no further arrows change direction, the algorithm converged to a solution.

Steps 6 and 7

Different random orderings (step 3) can result in different solutions.

- Step 6: repeat Steps 3 to 5 many times and store all different solutions.
- Step 7: score all solutions and select the graph with best score (maximized log-likelihood or BIC).

Steps 6 and 7

Different random orderings (step 3) can result in different solutions.

- Step 6: repeat Steps 3 to 5 many times and store all different solutions.
- Step 7: score all solutions and select the graph with best score (maximized log-likelihood or BIC).

Steps 6 and 7

Different random orderings (step 3) can result in different solutions.

- Step 6: repeat Steps 3 to 5 many times and store all different solutions.
- Step 7: score all solutions and select the graph with best score (maximized log-likelihood or BIC).

Sparsity assumption

The PC skeleton algorithm and QDG algorithm perform well in sparse graphs.

Directing edges without QTLs

- In general we need to have at least one QTL per pair of phenotypes to infer causal direction.

- In some situations, however, we may be able to infer causal direction for a pair of phenotypes without QTLs. Eg.

$$\begin{tikzcd} q_1 \arrow[d] & & \\ y_1 \arrow[r] & y_2 \arrow[r, dash, "?"] & y_3 \end{tikzcd} \qquad \begin{tikzcd} y_1 \arrow[r] & y_2 \arrow[r] & y_3 \\ y_1 \arrow[r] & y_2 & y_3 \arrow[l] \end{tikzcd}$$

since $f(y_1)\, f(y_2 \mid y_1)\, f(y_3 \mid y_2) \neq f(y_1)\, f(y_2 \mid y_1, y_3)\, f(y_3)$.

- So both QTLs and phenotypes play important roles in the orientation process.

Directing edges without QTLs

- In general we need to have at least one QTL per pair of phenotypes to infer causal direction.
- In some situations, however, we may be able to infer causal direction for a pair of phenotypes without QTLs. Eg.

$$\begin{tikzcd} q_1 \arrow[d] & & \\ y_1 \arrow[r] & y_2 \arrow[r, dash, "?"] & y_3 \end{tikzcd}
\qquad
\begin{tikzcd} y_1 \arrow[r] & y_2 \arrow[r] & y_3 \\ y_1 \arrow[r] & y_2 & y_3 \arrow[l] \end{tikzcd}$$

since $f(y_1)\, f(y_2 \mid y_1)\, f(y_3 \mid y_2) \neq f(y_1)\, f(y_2 \mid y_1, y_3)\, f(y_3)$.

- So both QTLs and phenotypes play important roles in the orientation process.

Directing edges without QTLs

- In general we need to have at least one QTL per pair of phenotypes to infer causal direction.

- In some situations, however, we may be able to infer causal direction for a pair of phenotypes without QTLs. Eg.

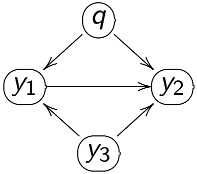
$$\begin{tikzcd}
q_1 \arrow[d] & & \\
y_1 \arrow[r] & y_2 \arrow[r, dash, "?"] & y_3
\end{tikzcd}
\qquad
\begin{tikzcd}
y_1 \arrow[r] & y_2 \arrow[r] & y_3 \\
y_1 \arrow[r] & y_2 & y_3 \arrow[l]
\end{tikzcd}$$

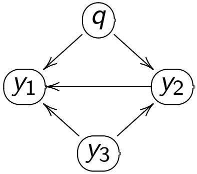
since $f(y_1)\, f(y_2 \mid y_1)\, f(y_3 \mid y_2) \neq f(y_1)\, f(y_2 \mid y_1, y_3)\, f(y_3)$.

- So both QTLs and phenotypes play important roles in the orientation process.

Unresolvable situation

- We cannot infer direction when the phenotypes have exactly same set of QTLs and causal phenotypes


$$\begin{tikzcd} & q \arrow[dl] \arrow[dr] & \\ y_1 \arrow[rr] & & y_2 \\ & y_3 \arrow[ul] \arrow[ur] & \end{tikzcd}$$


$$\begin{tikzcd} & q \arrow[dl] \arrow[dr] & \\ y_1 & & y_2 \arrow[ll] \\ & y_3 \arrow[ul] \arrow[ur] & \end{tikzcd}$$

since

$$f(y_1 \mid y_3, q)\, f(y_2 \mid y_1, y_3, q) \;=\; f(y_1 \mid y_2, y_3, q)\, f(y_2 \mid y_3, q)$$

Reducing graph space

The QDG algorithm drastically reduces the number of graphs that need to be scored.

1. The maximum number of graphs is 2^k models, where k is the number of edges in the skeleton.

2. The number of solutions of the QDG algorithm is generally much smaller than 2^k.

Cyclic networks

- Cycles are a common feature of biological networks (homeostatic mechanisms).
- The PC skeleton algorithm assumes an acyclic causal graph, and cycles may lead to spurious edges. E.g.

$$\begin{tikzcd}
1 \arrow[r] & 2 \arrow[d, bend left] \\
4 \arrow[r] & 3 \arrow[u, bend left]
\end{tikzcd}
\qquad
\begin{tikzcd}
1 \arrow[r, dash] \arrow[dr, dash] & 2 \arrow[d, dash] \arrow[dl, dash] \\
4 \arrow[r, dash] & 3
\end{tikzcd}$$

$$\begin{array}{lll}
1 \not\perp\!\!\!\perp 3 & 1 \not\perp\!\!\!\perp 3 \mid 2 & 1 \not\perp\!\!\!\perp 3 \mid 2,4 \\
2 \not\perp\!\!\!\perp 4 & 2 \not\perp\!\!\!\perp 4 \mid 3 & 2 \not\perp\!\!\!\perp 4 \mid 1,3
\end{array}$$

Cyclic networks

- Cycles are a common feature of biological networks (homeostatic mechanisms).
- The PC skeleton algorithm assumes an acyclic causal graph, and cycles may lead to spurious edges. E.g.

$$\begin{tikzcd} \boxed{1} \arrow[r] & \boxed{2} \arrow[d, bend left] \\ \boxed{4} \arrow[r] & \boxed{3} \arrow[u, bend left] \end{tikzcd}$$

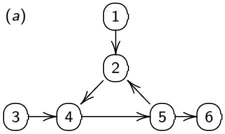
$$\begin{tikzcd} \boxed{1} \arrow[r, dash] \arrow[dr, dash] & \boxed{2} \arrow[d, dash] \arrow[dl, dash] \\ \boxed{4} \arrow[r, dash] & \boxed{3} \end{tikzcd}$$

$1 \not\perp\!\!\!\perp 3 \quad 1 \not\perp\!\!\!\perp 3 \mid 2 \quad 1 \not\perp\!\!\!\perp 3 \mid 2, 4$

$2 \not\perp\!\!\!\perp 4 \quad 2 \not\perp\!\!\!\perp 4 \mid 3 \quad 2 \not\perp\!\!\!\perp 4 \mid 1, 3$

Cyclic networks

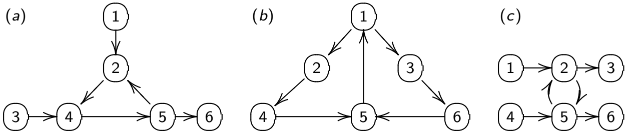
- Our simulations showed good performance with toy cyclic graphs, though.



- The spurious edges in graph (c) were detected at low rates.
- QDG approach cannot detect reciprocal interactions. In graph (c) it orients the edge (2)—(5) in the direction with higher strength.

Cyclic networks

- Our simulations showed good performance with toy cyclic graphs, though.

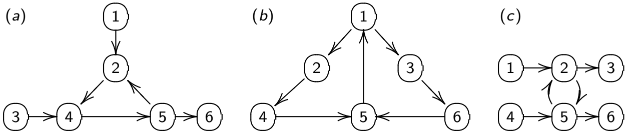


- The spurious edges in graph (c) were detected at low rates.

- QDG approach cannot detect reciprocal interactions. In graph (c) it orients the edge $\textcircled{2}$—$\textcircled{5}$ in the direction with higher strength.

Cyclic networks

- Our simulations showed good performance with toy cyclic graphs, though.



- The spurious edges in graph (c) were detected at low rates.

- QDG approach cannot detect reciprocal interactions. In graph (c) it orients the edge $\begin{tikzcd}2 \arrow[r, no head] & 5\end{tikzcd}$ in the direction with higher strength.

Unique graph instead of an equivalence class

Two DAGs are Markov equivalent iff they have the same skeleton and the same set of v-structures. For example

$$\begin{tikzcd}
y_1 \arrow[dr] & & & & & & y_3 \arrow[dl] & & & \\
& y_2 \arrow[dr] & & & & y_2 \arrow[dl] & & & y_2 \arrow[dl] \arrow[dr] & \\
& & y_3 & & y_1 & & & y_1 & & y_3
\end{tikzcd}$$

The three graphs have the same skeleton, $\begin{tikzcd} y_1 \arrow[r, dash] & y_2 \arrow[r, dash] & y_3 \end{tikzcd}$, and the same set of v-structures (none).

The graphs will also be likelihood equivalent if we assume a linear regression with Gaussian errors.

Unique graph instead of an equivalence class

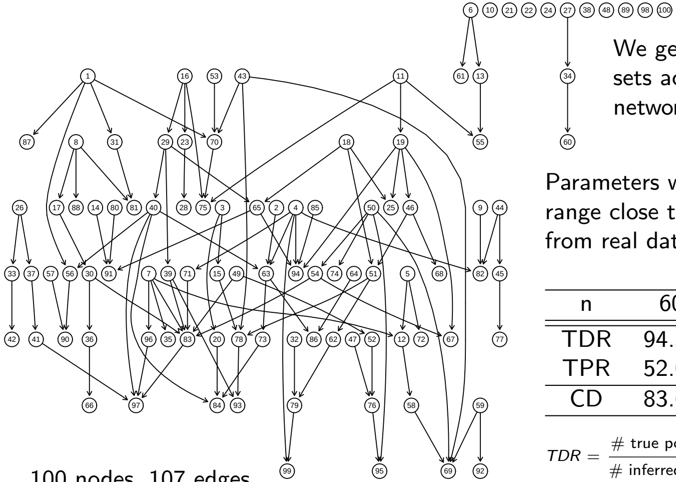
$$\begin{tikzcd}
q_1 \arrow[d] & q_2 \arrow[d] & q_3 \arrow[d] \\
y_1 \arrow[r] & y_2 \arrow[r] & y_3
\end{tikzcd}
\qquad
\begin{tikzcd}
q_1 \arrow[d] & q_2 \arrow[d] & q_3 \arrow[d] \\
y_1 & y_2 \arrow[l] & y_3 \arrow[l]
\end{tikzcd}
\qquad
\begin{tikzcd}
q_1 \arrow[d] & q_2 \arrow[d] & q_3 \arrow[d] \\
y_1 & y_2 \arrow[l] \arrow[r] & y_3
\end{tikzcd}$$

Same skeleton, but different sets of v-structures

$$\begin{tikzcd}
y_1 \arrow[dr] & & q_2 \arrow[dl] \\
& y_2 &
\end{tikzcd}
\qquad
\begin{tikzcd}
q_2 \arrow[dr] & & y_3 \arrow[dl] \\
& y_2 &
\end{tikzcd}
\qquad
\begin{tikzcd}
y_2 \arrow[dr] & & q_3 \arrow[dl] \\
& y_3 &
\end{tikzcd}$$

$$\begin{tikzcd}
y_2 \arrow[dr] & & q_3 \arrow[dl] \\
& y_3 &
\end{tikzcd}
\qquad
\begin{tikzcd}
q_1 \arrow[dr] & & y_2 \arrow[dl] \\
& y_1 &
\end{tikzcd}
\qquad
\begin{tikzcd}
q_1 \arrow[dr] & & y_2 \arrow[dl] \\
& y_1 &
\end{tikzcd}$$

Simulations



We generated 100 data sets according to this network.

Parameters were chosen in a range close to values estimated from real data.

n	60	300	500
TDR	94.53	95.18	91.22
TPR	52.07	87.33	93.64
CD	83.65	98.58	99.63

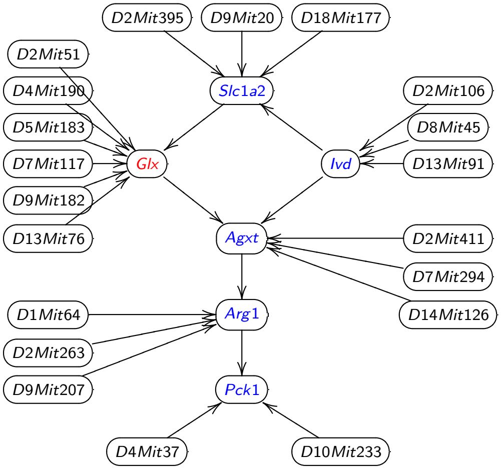
$$TDR = \frac{\#\text{ true positives}}{\#\text{ inferred edges}}, \quad TPR = \frac{\#\text{ true positives}}{\#\text{ true edges}}$$

CD: correct direction

100 nodes, 107 edges

2 or 3 QTLs per phenotype (not shown)

Real data example



- We constructed a network from metabolites and transcripts involved in liver metabolism.
- We validated this network with *in vitro experiments* (Ferrara et al 2008). Four out of six predictions were confirmed.

Software and references

The *qdg* R package is available at CRAN.

References:

- Chaibub Neto et al 2008. Inferring causal phenotype networks from segregating populations. Genetics 179: 1089-1100.
- Ferrara et al 2008. Genetic networks of liver metabolism revealed by integration of metabolic and transcriptomic profiling. PLoS Genetics 4: e1000034.
- Spirtes et al 1993. Causation, prediction and search. MIT press.

Acknowledgements

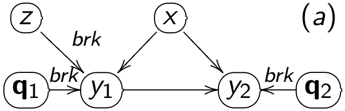
Co-authors:

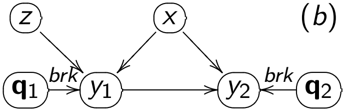
- Alan D. Attie
- Brian S. Yandell
- Christine T. Ferrara

Funding:

- CNPq Brazil
- NIH grants DK66369, DK58037 and DK06639

Permutation p-values





- To break the connections (brk) that affect direction of an edge, we permute the corresponding pair of nodes (and their common covariates) as a block.
- In panel (a) we permute (y_1, y_2, x) as a block breaking the connections with z, $\mathbf{q}_1$ and $\mathbf{q}_2$;
- In panel (b) we incorrectly keep z in the permutation block.

Direct versus indirect effects of a common QTL

(a)

$$\begin{tikzcd}
\mathbf{q}_1 \arrow[d] & q \arrow[dl] \arrow[dr] & \mathbf{q}_2 \arrow[d] \\
y_1 \arrow[rr, dash] & & y_2
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
\mathbf{q}_1 \arrow[d] & q \arrow[dl] & \mathbf{q}_2 \arrow[d] \\
y_1 \arrow[rr] & & y_2
\end{tikzcd}$$

(c)

$$\begin{tikzcd}
\mathbf{q}_1 \arrow[d] & q \arrow[dr] & \mathbf{q}_2 \arrow[d] \\
y_1 & & y_2 \arrow[ll]
\end{tikzcd}$$

- A strong QTL directly affecting an upstream trait may also be (incorrectly) detected as a QTL for a downstream phenotype.
- To resolve this situation we apply a generalization of Schadt et al. 2005 allowing for multiple QTLs.
- Model (a) supports both traits being directly affected by the common QTL q. Model (b) implies that q directly affects y_1 but should not be included as a QTL of phenotype y_2. Model (c) supports the reverse situation.