Belief Propagation in Genotype-Phenotype Networks using the `geneNetBP` package

Janhavi Moharil
University at Buffalo

`geneNetBP` version 2.0.0 as of 2016-05-04

Contents

1 Introduction

The `geneNetBP` package leverages belief propagation methods in genotype-phenotype networks inferred from Quantitative Trait Loci (QTL) data. The network structure can be perturbed by absorbing phenotypic evidence and the system-wide effects on the network are quantified in a nodewise manner. The package implements methods specifically to fit Conditional Gaussian Bayesian Network (CG-BN) or Discrete Bayesian Network to QTL data, absorb phenotype evidence and quantify and visualize the changes in network beliefs. For detailed description of methods, refer to our SAGMB publication. To cite "geneNetBP", use:

Janhavi Moharil, Paul May, Daniel P. Gaile, Rachael Hageman Blair (2016). "Belief Propagation in Genotype-Phenotype Networks.", Stat Appl Genet Mol Biol, 15(1):39-53.

For belief propagation in CG-BN, the package makes extensive use of the package `RHugin` that provides an R interface for the Hugin Decision Engine, a commercial software for building and infering Bayesian belief networks. Note that `RHugin` is currently not available on CRAN and is hosted on R-Forge. `geneNetBP` requires both Hugin and RHugin to be installed. `RHugin` can be downloaded from `http://rhugin.r-forge.r-project.org`. The Hugin Decision Engine can be downloaded from `http://www.hugin.com`. Detailed installation instructions of the `geneNetBP` package and package dependencies are available on the geneNetBP project homepage. Note that `RHugin` is required for the functioning of CG-BN implementation of `geneNetBP`. The package `RHugin` will not automatically load upon loading `geneNetBP` package. Use `library(RHugin)` or `require(RHugin) to load RHugin before using geneNetBP`.

For belief propagation in discrete bayesian networks where both the genotypes and phenotypes are categorical data, the structural learning in `geneNetBP` version 2.0.0 is implemented using the package `bnlearn` while belief propagation is implemented using the package `gRain`. Both the packages are available on CRAN. `HuginLite` can also be used

to infer networks from discrete data, however the demo version is restricted to 50 states and 500 cases. For larger datasets, the discrete bayesian network learning and inference using functions that implement `bnlearn` and `gRain` is recommended.

2 Datasets

There are 4 datasets provided with this package.

2.1 mouse

The *Mus Musculus* Kidney eQTL data (`mouse`) was obtained from a F2 inner-cross between inbred MRL/MpJ and SM/J strains of mice [1]. The original data consists of 33,872 gene expression traits for 173 males. After linkage analysis and filtering based on location and significance of QTL, the data consists of 14 genes and their SNP markers corresponding to their QTL. Thus the final dataset `mouse` is a data frame of 173 observations of 19 variables (5 genotypes - SNP markers and 14 genes - normalized gene expression values).

Load the dataset and view the first 3 observations:

```
> data(mouse,package="geneNetBP")
> head(mouse,n=3)
```

```
  Qchr4 Qchr17 Qchr15 Qchr11 Qchr2    Cyp4a31     Slc5a9    Slc6a9      Hmgcl
1     2      3      2      2     2 -0.8581591 -1.1433976 2.1143808 -0.3683079
3     3      2      2      2     2  0.2622828  0.3683079 0.6476036  0.1155036
4     2      2      2      3     1  0.1445610  0.6476036 0.3993231  0.5951785
      Ptp4a2        Ak2     Zbtb8a       Stx12    Trspap1       Mecr      Wdtc1
1  1.2006550  0.4149740  0.5443409  0.02881581 -1.0144987 -0.4625623 -0.3224307
3 -0.2177984  0.8581591 -1.0389014  0.66547438 -1.6851791 -0.7582926  0.9906857
4 -0.1737411 -0.8172368  1.0389014 -0.81723682 -0.7972819 -0.2030698  0.5611245
       Atpif1     Rbbp4       Tlr12
1 -1.14339756 1.3644887 -0.52770925
3 -1.32881793 1.2308184 -0.83752265
4  0.08654337 0.3683079 -0.01440641
```

There are 3 possible genotype states MM (homozygous) denoted by 1, H (heterozygous) by 2 and SS (homozygous) by 3. The genotypes are categorical variables and hence first 5 columns in the data frame `mouse` have to be of class factor while the phenotypes are continuous variables with 14 columns in data frame `mouse` of class numeric.

2.2 hdl

The *Mus Musculus* HDL QTL data (`hdl`) was obtained from a F2 inner-cross between inbred MRL/MpJ and SM/J strains of mice [**?**]. The original data consists of 33,872 gene expression traits for 280 males and females. After linkage analysis and filtering based on location and significance of QTL, the data consists of 10 phenotypes (9 genes and HDL level) and their 5 SNP markers corresponding to their QTL. Thus the final dataset `hdl` is a data frame of 280 observations of 15 variables (5 SNP markers and 10 phenotypes (9 normalized gene expression and HDL levels).

Load the dataset and view the first 3 observations:

```
> data(hdl,package="geneNetBP")
> head(hdl,n=3)
```

```
  c1 c2 c4 c7 c12        HDL     Pla2g4a      Nr1i3     Cyp2b10     Ppap2a
1  2  1  2  1   3 -0.1601137  0.67171243 -0.5748821  0.96978138  0.6606545
2  3  3  3  2   2 -0.8365833 -0.75159139  1.1396864 -0.05760458  0.4145857
3  3  1  2  2   1 -1.1655010 -0.08424431 -0.5233845 -0.30585203 -1.0132221
       Kdsr       Degs1      Neu1       Spgl1     Apoa2
1 0.3762732  0.4049526 0.6940793  1.35105303 1.1396864
2 1.9645187 -0.9011827 0.7398946 -1.37345382 2.1935392
3 0.3667830  0.6065198 1.4451709 -0.02657516 0.7873186
```

Note that there are 3 possible genotype states MM (homozygous) denoted by 1, H (heterozygous) by 2 and SS (homozygous) by 3.

2.3 toy

The `toy` is a simulated eQTL dataset from the network shown below, of 500 observations, 3 genotypes (Q1,Q2,Q3) each having 2 possible states and 6 phenotypes, X1-X6.

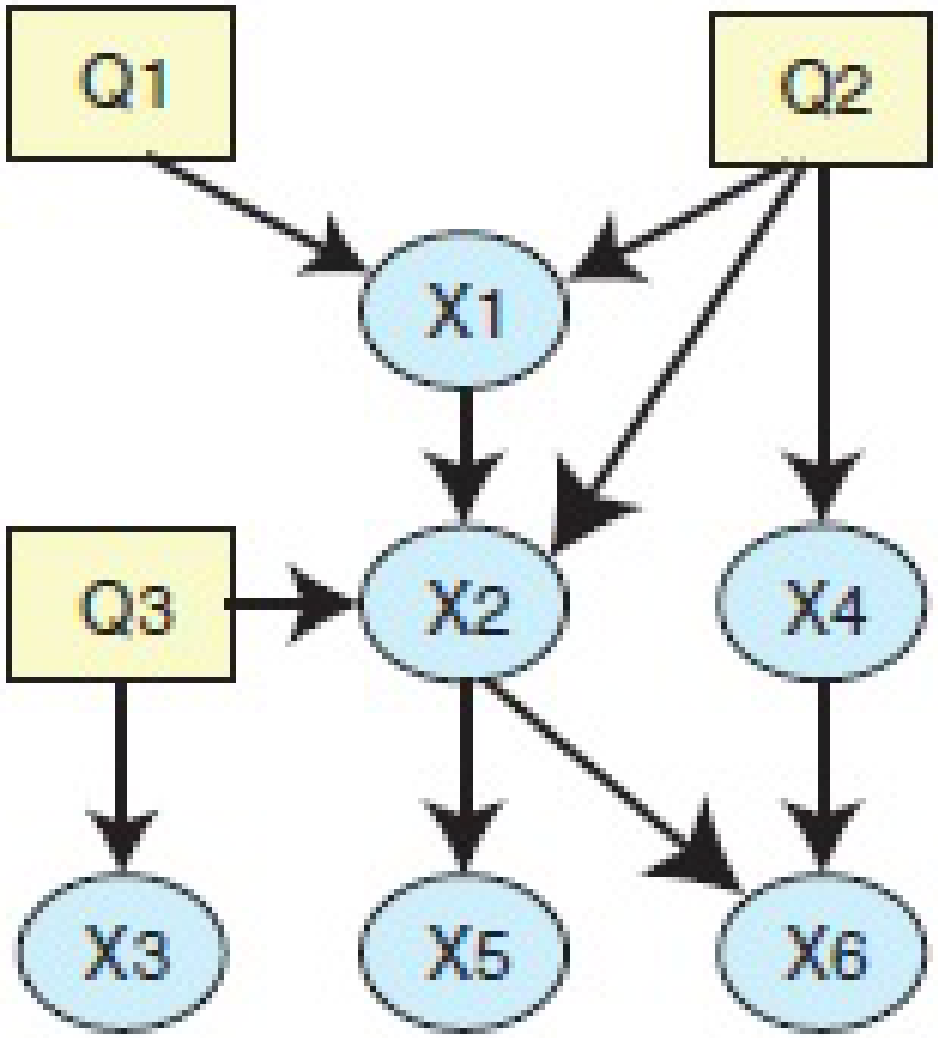

Figure 1. Toy network example.

2.4 yeast

The `yeast` dataset is a subset of the widely studied yeast expression dataset comprising of 112 F1 segregants from a cross between BY4716 and RM11-1a strains of *Saccharomyces Cerevisiae* [2, 3]. The original dataset consists of expression values reported as log2(sample/ BY reference) for 6216 genes. The data can be accessed in Gene Expression Omnibus (GEO) by accession number (GSE1990). After linkage analysis and filtering based on location and significance of QTL, a final set of 25 genes and their corresponding

12 SNP markers were identified and included in the yeast dataset.

Thus the final dataset `yeast` is a data frame of 112 observations of 25 variables (9 SNP markers and 16 genes - normalized gene expression values).

Load the dataset and view the first 3 observations:

```
> data(yeast,package="geneNetBP")
> head(yeast,n=3)
```

Note that there are 2 possible genotype states denoted by 1 and 2. The genotypes are categorical variables and hence all genotype columns in data frame `yeast` have to be of class factor. The phenotypes are continuous variables and phenotype columns in data frame `yeast` of class numeric.

3 Infering network structure

3.1 Fit CG-BN to QTL data

3.1.1 Model

The graphical model is represented by a Directed Acyclic Graph (DAG). The nodes in the graph represent the model variables, which may be discrete (QTL) or continuous (phenotypes). The phenotypes (e.g., metabolites, gene-expression, or clinical traits etc) are assumed to be continuous and follow a normal distribution. The data consists of n phenotypes (X) and m genotypes at Single Nucleotide Polymorphism (SNP) markers and is defined as: $D = \{X_1, \ldots, X_n, Q_1, \ldots, Q_m\}$.

Model Assumptions:

1. Discrete variables precede the continuous variables.
2. No relationships between discrete variables (no edges between them).

Local relationships between continuous child nodes and parents are described using Homogeneous Conditional Gaussian Models (HCGM). The conditional distribution for a phenotype $Y = X_j$ with discrete parent Q_i with genotype states (g) and continuous parent X_i ($i \neq j$) is modeled as:

$$P\left(Y \mid Q_i = g, X_i = x_i\right) = N\left(\alpha(g) + \beta(g)^T x_i, \gamma(g)\right), \tag{1}$$

where the mean is a regression that depends on both discrete and continuous parents, but the variance depends only on the discrete parents (genotype states). The parameters

of the CG-BN and subsequently the marginal distributions are inferred from the data under the constraints of the topology and the Markov condition using the PC-algorithm in `RHugin` package.

3.1.2 Mouse kidney eQTL Example

We will use the function `fit.gnbp` to learn the structure of a genotype-phenotype network from `mouse` dataset. This function uses the PC algorithm and the EM algorithm implemented in the `RHugin` package to learn the network structure and and the conditional probability tables for each node in the network. You will need both HuginLite and `RHugin` installed. Please refer to Section 1 for installation instructions.

We will first load the `mouse` dataset and extract the genotype and phenotype data.The first five columns are genotype (categorical) and the remaining 14 columns are phenotypes (continuous).

```
> data(mouse,package=geneNetBP)
> mousegeno<-mouse[,1:5]
> mousepheno<-mouse[,6:19]
```

The simplest example of fitting a CG-BN to mouse QTL data is as follows. This example uses default parameters.

```
>data(mouse)
> fit.gnbp(mousegeno,mousepheno)

$gp
A Hugin domain: there are 19 nodes and 17 edges


$gp_nodes
node      class     levels type
[1,] "Cyp4a31" "numeric" "0"    "pheno"
[2,] "Slc5a9"  "numeric" "0"    "pheno"
[3,] "Slc6a9"  "numeric" "0"    "pheno"
[4,] "Hmgcl"   "numeric" "0"    "pheno"
[5,] "Ptp4a2"  "numeric" "0"    "pheno"
[6,] "Ak2"     "numeric" "0"    "pheno"
[7,] "Zbtb8a"  "numeric" "0"    "pheno"
[8,] "Stx12"   "numeric" "0"    "pheno"
```

```
[9,] "Trspap1" "numeric" "0"    "pheno"
[10,] "Mecr"    "numeric" "0"    "pheno"
[11,] "Wdtc1"   "numeric" "0"    "pheno"
[12,] "Atpif1"  "numeric" "0"    "pheno"
[13,] "Rbbp4"   "numeric" "0"    "pheno"
[14,] "Tlr12"   "numeric" "0"    "pheno"
[15,] "Qchr4"   "factor"  "3"    "geno"
[16,] "Qchr17"  "factor"  "3"    "geno"
[17,] "Qchr15"  "factor"  "3"    "geno"
[18,] "Qchr11"  "factor"  "3"    "geno"
[19,] "Qchr2"   "factor"  "3"    "geno"

$gp_flag
[1] "cg"

attr(,"class")
[1] "gpfit"
```

The learnt network structure is returned as RHugin domain in the first element `gp` of the list. RHugin domain is an external pointer and hence cannot be saved in R workspace. The `RHugin` package provides functions `read.rhd` and `write.rhd` for loading and saving Hugin domains. The domains that are not saved will be lost when quitting R. The use of assignment operator such as <- or = will only return the pointer.Refer to the RHugin help manual for more information.The other elements in the list are for internal use with other functions.

The inferred network structure is very sensitive to the significance level (specified as `alpha`) and hence it is recommended to try out different values of the argument `alpha`. Note that the argument `alpha` is for use with RHugin package i.e. the function `fit.gnbp` will pass on `alpha` to RHugin functions.For example,

```
> fit.gnbp(mousegeno,mousepheno,alpha = 0.1)

$gp
A Hugin domain: there are 19 nodes and 31 edges


$gp_nodes
node      class     levels type
[1,] "Cyp4a31" "numeric" "0"    "pheno"
[2,] "Slc5a9"  "numeric" "0"    "pheno"
```

```
[3,] "Slc6a9"  "numeric" "0"    "pheno"
[4,] "Hmgcl"   "numeric" "0"    "pheno"
[5,] "Ptp4a2"  "numeric" "0"    "pheno"
[6,] "Ak2"     "numeric" "0"    "pheno"
[7,] "Zbtb8a"  "numeric" "0"    "pheno"
[8,] "Stx12"   "numeric" "0"    "pheno"
[9,] "Trspap1" "numeric" "0"    "pheno"
[10,] "Mecr"    "numeric" "0"    "pheno"
[11,] "Wdtc1"   "numeric" "0"    "pheno"
[12,] "Atpif1"  "numeric" "0"    "pheno"
[13,] "Rbbp4"   "numeric" "0"    "pheno"
[14,] "Tlr12"   "numeric" "0"    "pheno"
[15,] "Qchr4"   "factor"  "3"    "geno"
[16,] "Qchr17"  "factor"  "3"    "geno"
[17,] "Qchr15"  "factor"  "3"    "geno"
[18,] "Qchr11"  "factor"  "3"    "geno"
[19,] "Qchr2"   "factor"  "3"    "geno"

$gp_flag
[1] "cg"

attr(,"class")
[1] "gpfit"
```

The inferred network structure can be visualized by the generic plot method for objects of class "gpfit".

```
> network<-fit.gnbp(mousegeno,mousepheno,alpha = 0.1)
> ## plot method for graph objects
> plot(network)
```

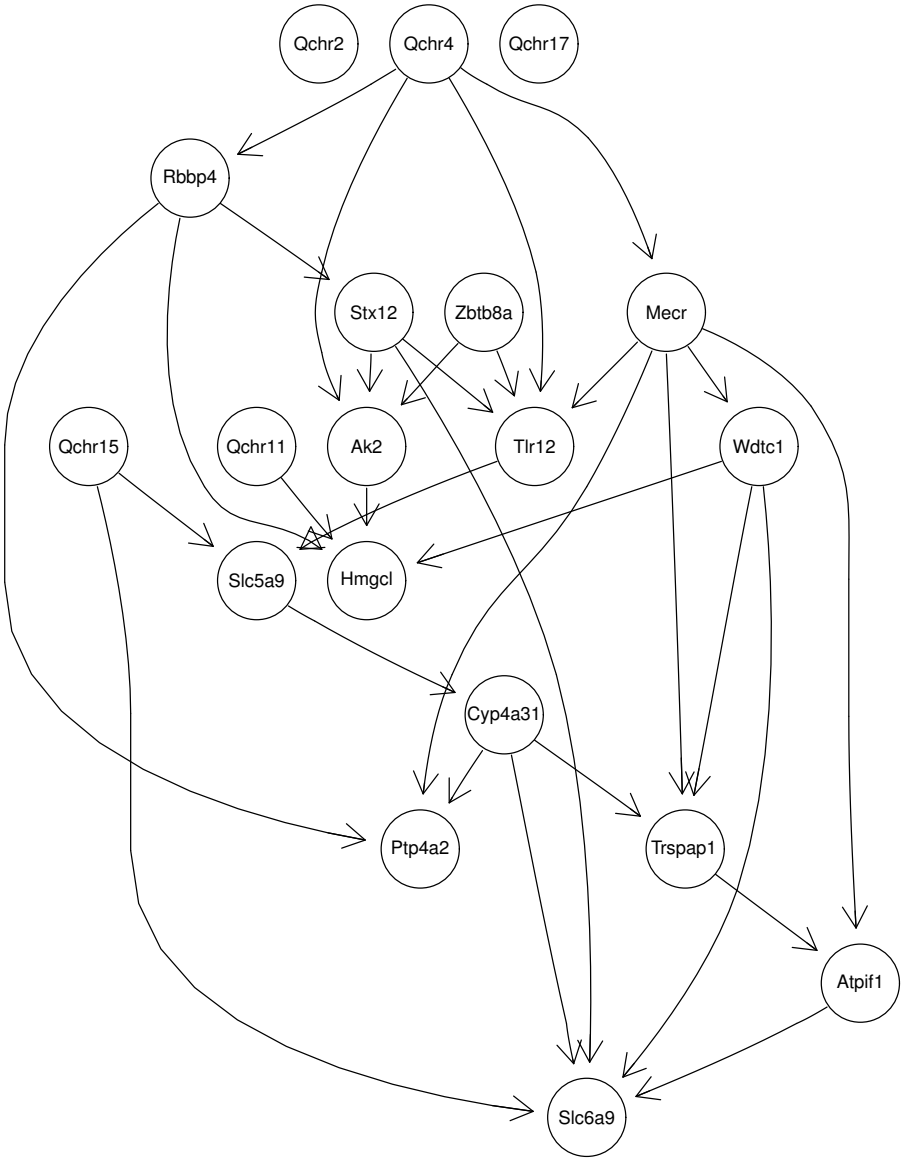

Figure 2. Conditional Gaussian network learnt from mouse kidney eQTL data

The genotypes are represented by boxes and the phenotypes are represented by elliptical nodes. Notice that the network now has 31 edges.Also, Qchr17 and Qchr2 are not included in the network. Any additional domain knowledge can be provided through a list of constraints. More help about the structure of the constraints list can be found in RHugin documentation.

3.2 Fit a Discrete Bayesian Network to QTL data

3.2.1 Model

The model for a Discrete Bayesian Network is very similar to the CG-BN. The graphical model is still represented by a Directed Acyclic Graph (DAG). The QTL variables are discrete. The **phenotypes** however are also **discrete** and not continuous unlike in the CG-BN representation. The data consists of n phenotypes (X) and m genotypes at Single Nucleotide Polymorphism (SNP) markers and is defined as: $D = \{X_1, \ldots, X_n, Q_1, \ldots, Q_m\}$.

Model Assumptions are restated as: 1. Genotypes precede the phenotypes.
2. No relationships between genotypes (no edges between them).

3.2.2 Yeast Example

A discrete bayesian network can be learnt using 2 approaches. First one is by using `fit.gnbp` that implements the PC algorithm as described above.

We will first load the `yeast` dataset and extract the genotype and phenotype data as before.The first 9 columns are genotypes (categorical) and the next 16 columns are phenotypes (continuous).

```
> data(yeast,package="geneNetBP")
> yeastgeno<-yeast[,1:9]
> yeastpheno<-yeast[,10:25]
```

Since the phenotypes are continuous, we will discretize the data around the median.

```
> yeastpheno_dis<-yeastpheno
> for (i in 1:dim(yeastpheno)[2])
+ {
+   yeastpheno_dis[which(yeastpheno[,i]>=median(yeastpheno[,i])),i]<-"1"
+   yeastpheno_dis[which(yeastpheno[,i]<median(yeastpheno[,i])),i]<-"-1"
+   yeastpheno_dis[,i]<-as.factor(yeastpheno_dis[,i])
+ }
```

A discrete bayesian network can be learnt using `fit.gnbp` by setting `type = "db"`.

```
> yeast.gnbp<-fit.gnbp(yeastgeno, yeastpheno_dis,type="db",alpha=0.1)
```

The RHugin pointer to the inferred network structure is returned in the variable `gp` of the list. It consists of 25 nodes and 14 edges. Here is a plot of the network structure.

```
> plot(yeast.gnbp)
```

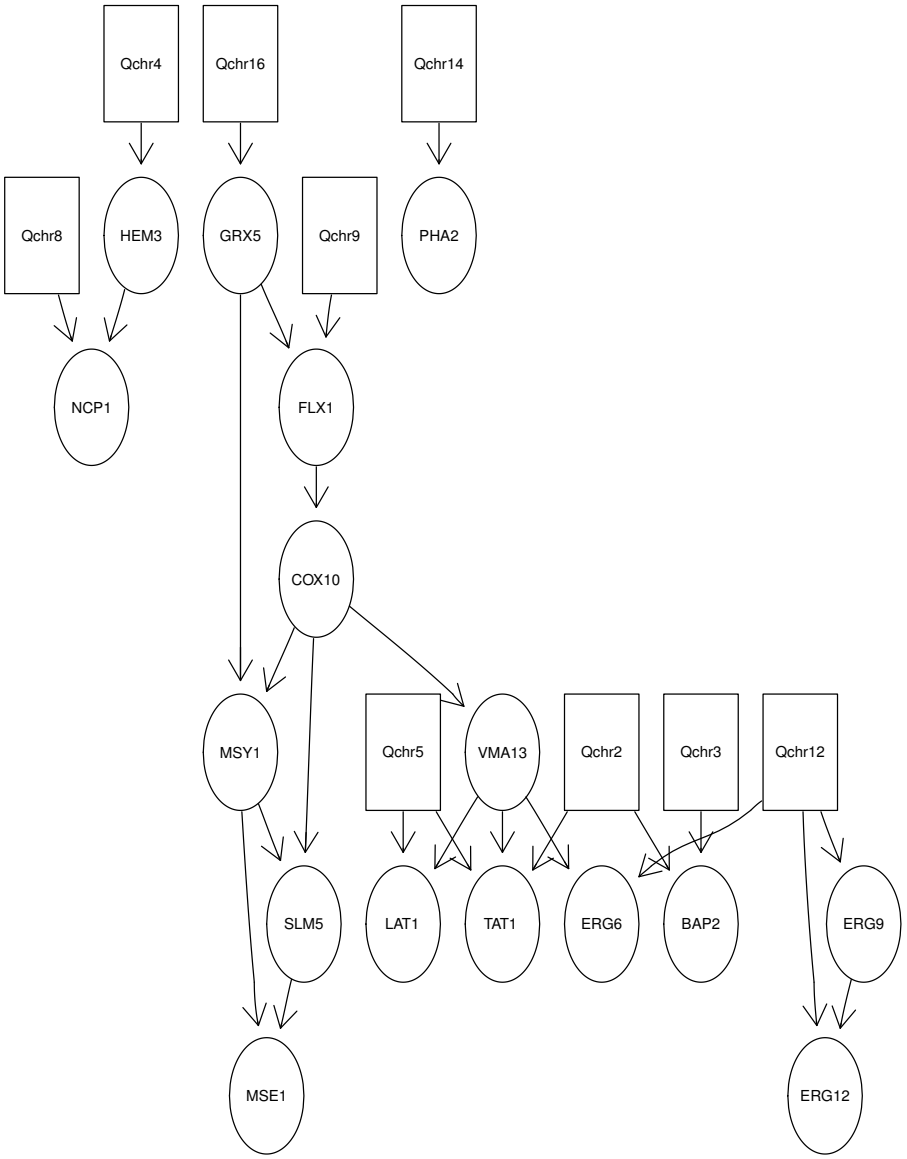

Figure 3. Discrete Bayesian Network learnt from Yeast data using (`fit.gnbp`)

The second approach to infer the network structure is by using `fit.dbn` that can implement several score-based and constraint-based learning methods from the package `bnlearn`. The default method is *Hill-Climbing* (`method = "hc"`).

```
> yeast.dbn<-fit.dbn(yeastgeno,yeastpheno_dis)

> plot(yeast.dbn)
```

`fit.dbn` return an object of class `dbnfit` which is a list several elements. The inferred network is returned as a "bn.fit" object in `dbn` variable of the list.

To choose a different method, specify `method`. For example, to fit the network by *Grow-Shrink* method,

```
> yeast.dbn<-fit.dbn(yeastgeno,yeastpheno_dis,method="gs")
> plot(yeast.dbn)
```

3.3 Extracting Conditional Probability Distributions

In both CG-BN and discrete Bayesian networks, there are conditional probability tables associated with each node in the network. The marginal distributions are returned in the second element `marginal` of the list in both `dbnfit` and `gpfit` objects.

In addition, the conditional distributions can also be accessed by using the package specific functions. For example, `get.marginal` from the package `RHugin` can be used to compute the marginal distributions in an RHugin domain. Another useful function is `get.table` to extract the CPT, experience or fading table associated with any node in an RHugin domain. Refer to `RHugin` manual for more help on these functions.

4 Absorbing evidence and Network Comparison

4.1 Conditional Gaussian Bayesian Networks

4.1.1 Belief propagation

In a CG-BN, new evidence can be entered by setting phenotypes in the network to a particular value, $X_i = x_i^*$. The evidence can pertain to a single node or multiple nodes in the network.

Through message passing, the probability distributions are updated (called as beliefs) after taking into account new evidence. Updated beliefs for discrete nodes (genotypes) are simply updated estimated frequencies under the new evidence. For continuous nodes (phenotypes), the updated beliefs are in terms of revised parameters for the Gaussian distribution. The original and absorbed network are compared node-wise by quantifying the change in marginals.

4.1.2 Jeffrey's Signed Information (JSI)

A symmetric version of the Kullback-Leibler information, known as Jeffrey's information is calculated to compare the marginal belief in the original network $X_i^0 \sim N(\mu_0, \sigma_0^2)$ to the absorbed network $X_i^{\text{abs}} \sim N(\mu_{\text{abs}}, \sigma_{\text{abs}}^2)$. Jeffrey's information, which is computed for all continuous unabsorbed nodes in the network, is given as:

$$J\left(X_i^0, X_i^{\text{abs}}\right) = I^{\text{KL}}\left(X_i^0, X_i^{\text{abs}}\right) + I^{\text{KL}}\left(X_i^{\text{abs}}, X_i^0\right)$$

where

$$I^{\text{KL}}\left(X_i^0, X_i^{\text{abs}}\right) = \frac{1}{2}\left\{\frac{(\mu_0 - \mu_{\text{abs}})^2}{\sigma_0^2} + \frac{\sigma_0^2}{\sigma_{\text{abs}}^2} - \log\left(\frac{\sigma_0^2}{\sigma_{\text{abs}}^2}\right) - 1\right\}.$$

For ease of interpretation, the signed Jeffrey's information

$$\text{sign}(\mu_0 - \mu_{\text{abs}}) \cdot \text{J}\left(\text{X}_\text{i}^0, \text{X}_\text{i}^{\text{abs}}\right)$$

is used to demonstrate the direction of change after the absorption of evidence.

The changes in belief are measured only for the nodes that are d-connected (conditionally dependent) to the entered evidence. Nodes that are d-separated from absorbed evidence are not influenced, and, consequently, do not change beliefs.

4.1.3 Mouse Kidney eQTL Example

We know that the marginal mean of the node `Tlr12` is -0.99 (see `marginal` in section 3.1.2) and we wish to enter this new information in the mouse network and compute the updated states of other nodes. New evidence for single or multiple nodes can be entered using the function `absorb.gnbp` which absorbs evidence and propagates the beliefs. The input to `absorb.gnbp` is an object of class `gpfit`, that is the output returned by the function `fit.gnbp`.
The function `absorb.gnbp` uses the `RHugin` package to absorb the evidence in the specified nodes and update the beliefs of all nodes and then calculates Jeffrey's signed information for all d-connected nodes. The following example illustrates how to absorb evidence in a genotype-phenotype network.

1. Absorb a single evidence for a single node

```
> mouse.cgbn<-fit.gnbp(mousegeno,mousepheno,alpha=0.1)
> ## Absorb evidence
> absorb.gnbp(mouse.cgbn,node="Tlr12",evidence=matrix(-0.99))
```

```
$gp
A Hugin domain: there are 19 nodes and 31 edges


$gp_flag
[1] "cg"

$gp_nodes
node      class     levels type
[1,] "Cyp4a31" "numeric" "0"    "pheno"
[2,] "Slc5a9"  "numeric" "0"    "pheno"
[3,] "Slc6a9"  "numeric" "0"    "pheno"
[4,] "Hmgcl"   "numeric" "0"    "pheno"
[5,] "Ptp4a2"  "numeric" "0"    "pheno"
[6,] "Ak2"     "numeric" "0"    "pheno"
[7,] "Zbtb8a"  "numeric" "0"    "pheno"
[8,] "Stx12"   "numeric" "0"    "pheno"
[9,] "Trspap1" "numeric" "0"    "pheno"
[10,] "Mecr"    "numeric" "0"    "pheno"
[11,] "Wdtc1"   "numeric" "0"    "pheno"
[12,] "Atpif1"  "numeric" "0"    "pheno"
[13,] "Rbbp4"   "numeric" "0"    "pheno"
[14,] "Tlr12"   "numeric" "0"    "pheno"
[15,] "Qchr4"   "factor"  "3"    "geno"
[16,] "Qchr17"  "factor"  "3"    "geno"
[17,] "Qchr15"  "factor"  "3"    "geno"
[18,] "Qchr11"  "factor"  "3"    "geno"
[19,] "Qchr2"   "factor"  "3"    "geno"

$evidence
[,1]
[1,] -0.99

$node
[1] "Tlr12"

$marginal
$marginal$pheno
$marginal$pheno$mean
[,1]
Rbbp4    2.317482e-17
```

```
Atpif1   2.190113e-03
Wdtc1    2.514671e-17
Mecr    -1.551256e-16
Trspap1  4.239712e-03
Stx12    4.433032e-17
Zbtb8a  -2.003327e-17
Ak2     -7.153821e-03
Ptp4a2   3.519799e-03
Hmgcl   -7.136515e-03
Slc6a9  -1.957688e-02
Slc5a9   2.471620e-02
Cyp4a31  1.914642e-02

$marginal$pheno$var
[,1]
Rbbp4   0.9557443
Atpif1  0.9027874
Wdtc1   0.9574396
Mecr    0.9550281
Trspap1 0.8530483
Stx12   0.9575380
Zbtb8a  0.9551227
Ak2     0.7696464
Ptp4a2  0.8550665
Hmgcl   0.8509102
Slc6a9  0.7939058
Slc5a9  0.8538129
Cyp4a31 0.8965621


$marginal$geno
$marginal$geno$freq
state1    state2   state3
Qchr4 0.2312139 0.4682081 0.300578



$belief
$belief$pheno
$belief$pheno$mean
[,1]
```

```
Rbbp4    0.8776457
Atpif1  -0.6538109
Wdtc1    0.6669131
Mecr    -0.8791569
Trspap1 -0.6613503
Stx12    0.8676931
Zbtb8a  -0.1222389
Ak2      0.6720433
Ptp4a2  -0.6969352
Hmgcl    0.6855139
Slc6a9   0.5667517
Slc5a9  -0.6510656
Cyp4a31 -0.5043484

$belief$pheno$var
[,1]
Rbbp4   0.4859803
Atpif1  0.6226163
Wdtc1   0.6627283
Mecr    0.4428854
Trspap1 0.5679888
Stx12   0.4933635
Zbtb8a  0.8083572
Ak2     0.5327134
Ptp4a2  0.5448964
Hmgcl   0.5628789
Slc6a9  0.5718937
Slc5a9  0.5254673
Cyp4a31 0.6995273


$belief$geno
$belief$geno$state1
[,1]
Qchr4 0.007944801

$belief$geno$state2
[,1]
Qchr4 0.2152284

$belief$geno$state3
```

```
[,1]
Qchr4 0.7768268



$JSI
[,1]
Rbbp4    0.71650239
Atpif1  -0.32687548
Wdtc1    0.31813768
Mecr    -0.79365404
Trspap1 -0.36674950
Stx12    0.69209864
Zbtb8a  -0.01550701
Ak2      0.40056441
Ptp4a2  -0.42017671
Hmgcl    0.39734466
Slc6a9   0.28567820
Slc5a9  -0.41106696
Cyp4a31 -0.18983139

$FC
NULL

attr(,"class")
[1] "gnbp"
```

Note that the function `absorb.gnbp` requires the argument `evidence` to be of class matrix. If only a single value of evidence is to be entered, this can be done by simply using the function matrix(), as above.

`absorb.gnbp` returns an object of class "gnbp" which is a list of several variables.

The Jeffrey's signed information is returned as a matrix `JSI` that gives the quantified comparison of beliefs of the continuous nodes (phenotypes) before and after evidence absorption. Note that since we absorbed only a single value of evidence, `JSI` is a column vector.In addition to Jeffrey's signed information, the marginal distributions (mean and variance for continuous nodes in and genotype frequencies for SNP markers) before evidence absorption and the updated beliefs (after evidence absorption) are also returned.

Since Qchr15 is `d`-separated when evidence is absorbed in `Tlr12`, it's marginal distribution is not affected and hence the beliefs are not calculated. Qchr4, on the other hand

is d-connected and a list returns the updated frequencies of all 3 genotype states of the SNP marker Qchr15.

2. Absorb a sequence of evidence for a single node

```
> mouse.cgbn<-fit.gnbp(mousegeno,mousepheno,alpha=0.1)
> ##Absorb evidence
> absorb.gnbp(mouse.cgbn,node="Tlr12",evidence=t(matrix(c(2.5,3,3.5,4))))
```

A function gen.evidence is useful to generate evidence for a node based on it's marginal distribution. This is particularly useful when network perturbation to assess the network behaviour is of interest.

To generate a spectrum of evidence for Tlr12 within ± 2 standard deviations of it's marginal distribution, we input the inferred network to gen.evidence

```
> mouse.cgbn<-fit.gnbp(mousegeno,mousepheno,alpha = 0.1)
> ##Generate evidence
> ev<-gen.evidence(mouse.cgbn,node="Tlr12",std=2,length.out=20)
> ##absorb evidence
> absorb.gnbp(mouse.cgbn,node="Tlr12",evidence=ev)
```

Note that JSI will now be a matrix whose number of rows are the d-connected phenotype nodes to Tlr12 and the number of columns is the length of evidence absorbed in Tlr12.

When a sequence of evidence is absorbed for a single node in the network, absorb.gnbp also plots the JSI of the d-connected nodes vs the evidence absorbed.

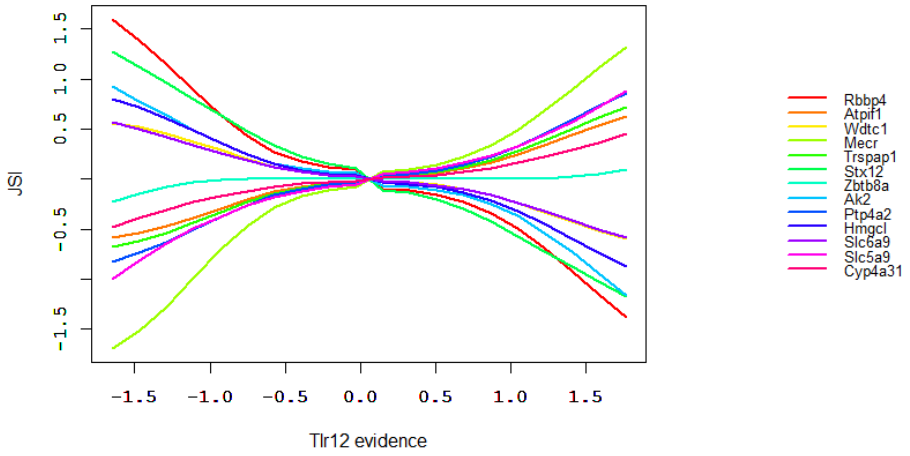

Figure 4. Plot produced by `absorb.gnbp`

4.2 Discrete Bayesian Networks

4.2.1 Belief propagation and Fold change (FC)

In Discrete Bayesian Networks, phenotype nodes are represented as $\{X_1, X_2, \ldots, X_p\}$. A node X_i, has the states, $X_i \in \{s_1, s_2, \ldots, s_n\}$, where $\sum_{k=1}^{n} s_k = 1$. Let $X_i^0 = \{s_1^0, s_2^0, \ldots s_n^0\}$ denote the states of X_i in the initial network, and $X_i^{pert} \in \{s_1^{pert}, s_2^{pert}, \ldots s_n^{pert}\}$ denotes the states of X_i in the perturbed network.

The node-wise change in marginals is quantified simply by the Fold Change (FC) as a measure of effect size for the state of maximal probability in the perturbed network. Let I^* be an indicator for the state of X_i^{pert} with maximum probability. That is, $I^* = 1$ if $s_k^{pert} = \max P(s_k^{pert})$, and 0 otherwise. The node-wise change in marginals is:

$$FC(X_i) = I^* \cdot \frac{P(s_k^{pert})}{P(s_k^0)}.$$

Note that $FC \in [0, 1)$ when the node is inhibited, $FC = 1$ when the node stays the same, and $FC > 1$ when the node is activated.

4.2.2 Yeast example

Consider the yeast network inferred in section 3.2. The phenotypes are discrete variables with 2 states (1,-1) in `Yeast` dataset. Like fit methods, there are two ways to absorb the

phenotypic evidence in discrete bayesian networks, the function `absorb.dbn` that implements `gRain` or `absorb.gnbp` that implements `RHugin`.

absorb.dbn

```
> ## Fit the network
> yeast.dbn<-fit.dbn(yeastgeno,yeastpheno_dis)
> ##Absorb evidence
> yeast.dbn.abs<-absorb.dbn(yeast.dbn,"COX10",matrix(c("1","-1"),ncol=2))
```

absorb.gnbp

Like previously mentions, `absorb.gnbp` can be used with objects of class `gpfit` that are output from `fit.gnbp`.

```
> ## Fit the network
> yeast.gnbp<-fit.gnbp(yeastgeno,yeastpheno_dis,type="db",alpha=0.1)
> ##Absorb evidence
> yeast.gnbp.abs<-absorb.gnbp(yeast.gnbp,"COX10",matrix(c("1")))
```

5 Visualizing network changes

A generic plot method for plotting the genotype-phenotype network in which evidence has been absorbed and propagated is available. The plot method will convert network into an object of class "graphNEL" by using Rgraphviz package. The argument nodeAttrs to plot method for graph objects in Rgraphviz package is then used to customize the plot.

5.1 A complete example of CG-BN

For CG-BN, a generic plot method `plot.gnbp` will be called for objects of class "gnbp". A complete example that fits a CG-BN, absorbs evidence and plots the network:

```
> network<-fit.gnbp(mousegeno,mousepheno,alpha=0.1)
> network<-absorb.gnbp(network,node="Tlr12",evidence=matrix(-0.99))
> plot(x=network)
```

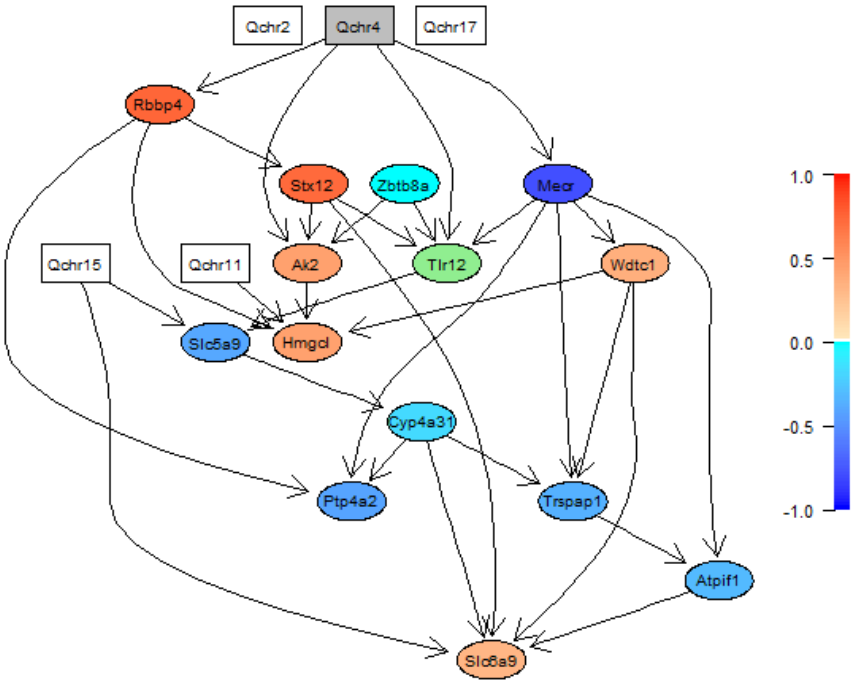

Figure 5. Evidence absorption in single node

The plot method will draw the network with Jeffrey's signed information mapped onto it by a colormap. There is an option to plot beliefs (updated marginal means) which can be entered through the argument `y` (see help for `plot.gnbp`).

The `d`-separated nodes are white while the colored nodes are `d`-connected, with the color indicating the strength and direction of change . By default, the continuous nodes are of shape "ellipse" and a "box" shape is used for discrete nodes. The node for which evidence is absorbed is colored green (default color).

5.1.1 Plot options in plot.gnbp

Colormap options such as end colors for the positive and negative gradients and the resolution of the colormap can be customized. The resolution of the colormap can be specified by `col.length`. The argument `col.palette` can be used to specify the end colors.

```
> col.palette<-list(pos_high="darkgreen", pos_low= "palegreen2",
                    neg_high="wheat1", neg_low = "red",
```

```
                    dsep_col="white",qtl_col="grey",node_abs_col="yellow")
> plot(x=network,col.palette=col.palette)
```

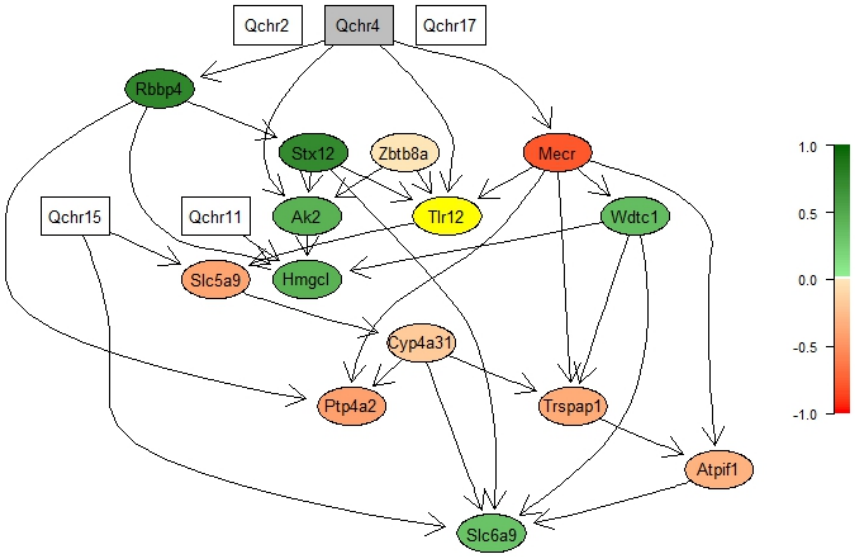

Figure 6. Mouse network with custom color palette

The plot method will always map the JSI or beliefs onto the network for a single piece of evidence. Incase a spectrum of evidence is absorbed for a single/multiple node(s), then the evidence for which we wish to visualize the network changes can be chosen by specifying the corresponding column number of JSI or belief matrix through the argument `ncol`.

For example if we absorbed a sequence of evidence for `Tlr12` and we wish to visualize the belief changes for evidence = 1.767, we can do this as follows.

```
> network<-fit.gnbp(mousegeno,mousepheno,alpha = 0.1)
> ##Generate evidence
> evidence<-gen.evidence(network,node="Tlr12",std=2,length.out=20)
> network<-absorb.gnbp(network,node="Tlr12",evidence=evidence)
> plot(x=network,y="belief",ncol=20)
```

5.2 Plotting Discrete Bayesian Networks

For discrete bayesian networks, a generic plot method is available for both "gnbp" and "dbn" objects.

6 Specifying Additional Biological Information

If additional biological information is known such as known or forbidden interactions between variables or the network hierarchy, it can be incorporated into the learning process. Such information can be provided in the `constraints` option of `fit.gnbp` and as `whitelist` and/or `blacklist` in `fit.dbn`. An example illustrating this:

Fit a CG-BN to `hdl` data using`fit.gnbp`. To do this, load the data.

```
> ## load data
> data(hdl)
> ##get the genotype and phenotype data
> hdlgeno<-hdl[,1:5]
> hdlpheno<-hdl[,6:15]
```

This dataset has 5 SNP markers and 10 phenotypes that include HDL levels and 9 genes. Since HDL levels is the observed characteristic, the genes should precede HDL in the network. In other words, HDL should be downstream and it cannot be a parent of rest of the variables in the network. This information can be included in the learning process by providing a list of constraints that defines the edges **from** HDL **to** genes as forbidden.

```
> ## create an empty vector for the blacklist
> blackL<-c()
> # fill in the forbidden edges. For example :"HDL"->"Nr1i3"
> for(i in 2:dim(hdlpheno_dis)[2])
+ blackL=rbind(blackL,cbind(colnames(hdlpheno_dis)[1],colnames(hdlpheno_dis)[i]))
> ## Form a list
> directed.forbidden <- vector("list", nrow(blackL))
> for (i in 1:nrow(blackL))
+   directed.forbidden[[i]] <- blackL[i,]
> constraints<-list(directed=list(forbidden=directed.forbidden,
+                                 required=NULL), undirected=NULL)
> ## Fit a CG-BN
> hdl.cgbn<-fit.gnbp(hdlgeno,hdlpheno_dis,constraints=constraints,alpha=0.08)
> ## Plot the network
> plot(hdl.cgbn)
```

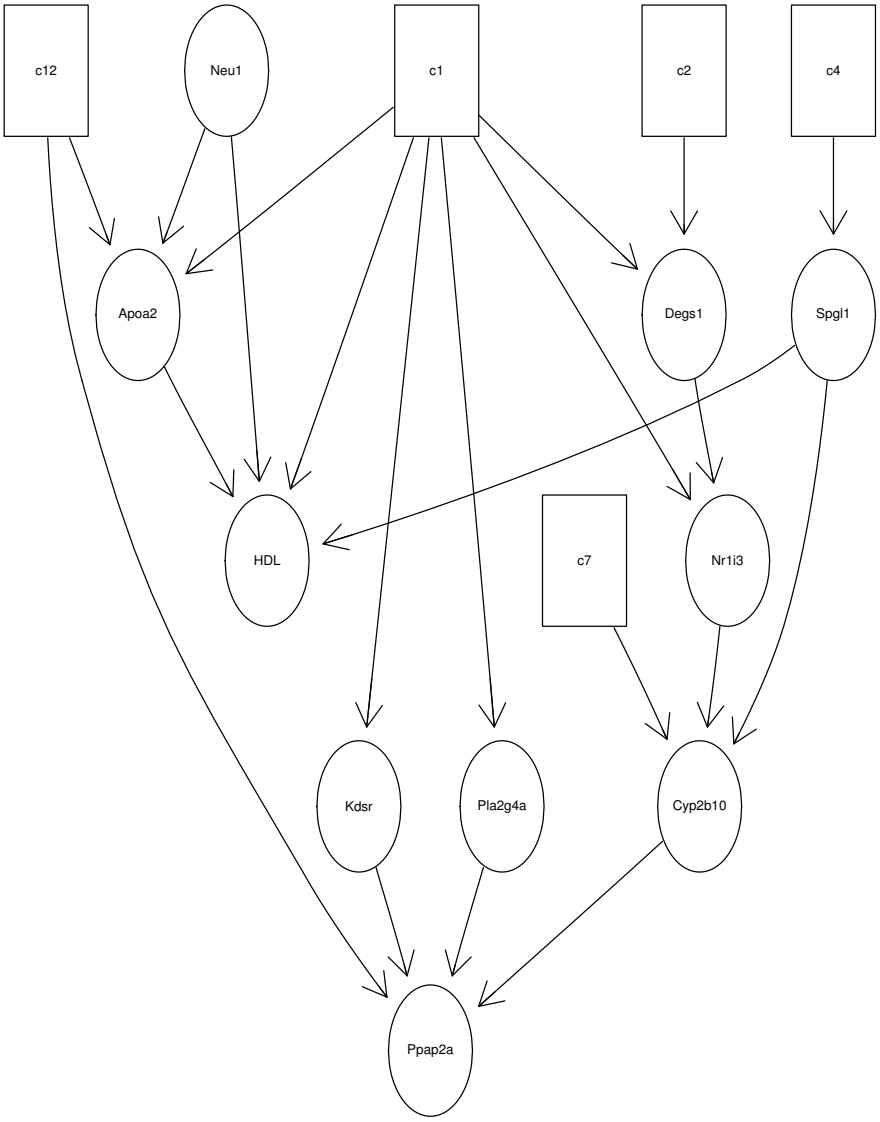

Figure 7. HDL network

As desired, HDL is downstream of the genes in the network. For `fit.dbn`, the `whitelist` option can be used to specify the required edges while the `blacklist` option can be used to specify forbidden edges.

7 Belief propagation in known networks

Belief propagation can be implemented in known genotype-phenotype networks. If the network structure is known apriori from a knowledge database, then learning step can be skipped in `fit.gnbp` by setting `learn = FALSE`. The conditional probabilities will still need to be learnt. This section demonstrates how to specify known networks and subsequent belief propagation.

7.1 Specifying graphNEL objects

One way to specify a known network is to provide the graph structure directly as an input (a graphNEL object) to the fit methods. The following example illustrates how to input a known graph structure using the `fit.dbn` function. Consider a discrete bayesian network implementation of HDL network. Assume that the HDL network structure is known. We will input the structure that we learnt in the previous section to `fit.dbn`.

```
> ## Convert the RHugindomain to a graphNEL object
> bngraph<-RHugin::as.graph.RHuginDomain(hdl.cgbn$gp)
```

For discrete bayesian network, discretize the phenotypes around median. To fit the network parameters, set `learn==FALSE` and specify the graph.

```
> ## discretize the data around median
> hdlpheno_dis<-hdlpheno
> for (i in 1:dim(hdlpheno)[2])
+ {
+   hdlpheno_dis[which(hdlpheno[,i]>=median(hdlpheno[,i])),i]<-"1"
+   hdlpheno_dis[which(hdlpheno[,i]<median(hdlpheno[,i])),i]<-"-1"
+   hdlpheno_dis[,i]<-as.factor(hdlpheno_dis[,i])
+ }
> ## fit dbn
> hdl.dbn<-fit.dbn(hdlgeno,hdlpheno_dis,graph=bngraph,learn ="FALSE")
```

7.2 Specifying edges, Toy example

The second way to specify an existing network is by providing a data frame of edges. Consider the `toy` network in section 2. The interactions between the variables in the `toy` dataset are known. To specify the toy network as a CG-BN and learn the conditional probabilities:

First create a data frame of known edges from parent to child.

```
>   ## Load the toy dataset
>   data(toy)
> ## Create a matrix of edges ("from (parent)", "to (child)")
> edgelist=data.frame(matrix(NA,ncol=2,nrow=10))
> edgelist[1,]<-cbind("Q1","X1")
> edgelist[2,]<-cbind("Q2","X1")
> edgelist[3,]<-cbind("Q2","X2")
> edgelist[4,]<-cbind("Q2","X4")
> edgelist[5,]<-cbind("X1","X2")
> edgelist[6,]<-cbind("Q3","X2")
> edgelist[7,]<-cbind("Q3","X3")
> edgelist[8,]<-cbind("X2","X5")
> edgelist[9,]<-cbind("X2","X6")
> edgelist[10,]<-cbind("X4","X6")
> ## label the columns
> colnames(edgelist)<-c("from","to")
```

In `fit.gnbp` provide the edgelist by setting `graph=edgelist` and set `learn = FALSE`. This will skip the learning and only conditional probabilities will be calculated for each node in the network based on the given network structure and data. Absorbing evidence and propagating the beliefs subsequently is then straightforward.

```
> ## Specify the network and learn conditional probabilities
> toy.cgbn<-fit.gnbp(toygeno,toypheno,learn=FALSE,graph=edgelist)
> ##Generate evidence
> evidence<-gen.evidence(toy.cgbn,node="X2",std=2,length.out=20)
> toy.cgbn.abs<-absorb.gnbp(toy.cgbn,node="X2",evidence=evidence)
> plot(x=toy.cgbn.abs,y="JSI",ncol=17,fontsize = 5)
```

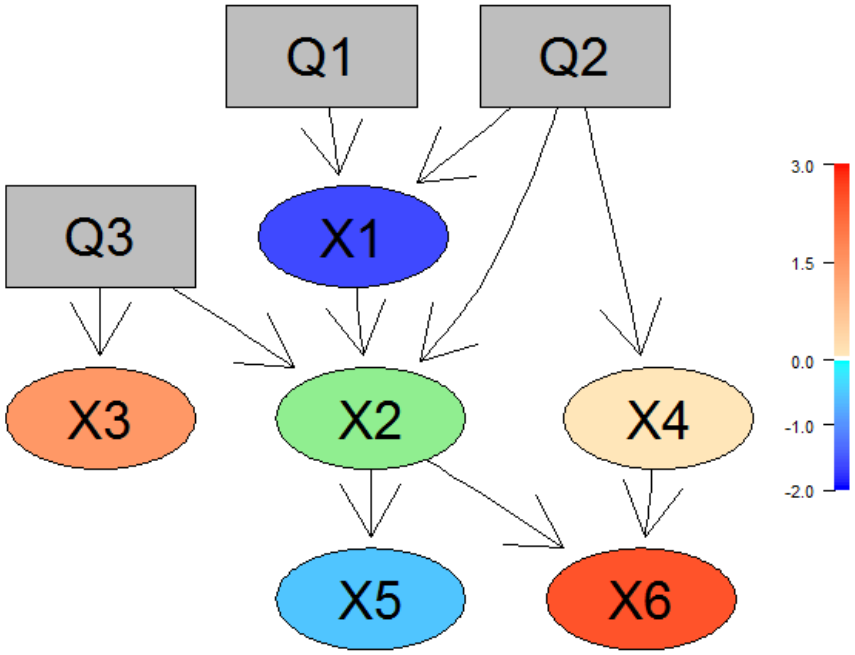

Figure 8. Belief propagation in known network

References

1. Hageman RS, Leduc MS, Caputo CR, Tsaih SW, Churchill GA, et al. (2011) Uncovering genes and regulatory pathways related to urinary albumin excretion. Journal of the American Society of Nephrology 22: 73–81.

2. Brem R, Kruglyak L (2005) The landscape of genetic complexity across 5,700 gene expression traits in yeast. Proc Natl Acad Sci 102: 1572-1577.

3. Brem R, Storey J, Whittle J, Kruglyak L (2005) Genetic interactions between polymorphisms that affect gene expression in yeast. Nature 436: 701-703.