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

Janhavi Moharil
University at Buffalo

`geneNetBP` version 1.0.0 as of 2016-03-17

Contents

1 Introduction

The `geneNetBP` package implements methods to predict system-wide changes in beliefs after absorbing evidence in probabilistic graphical models. The package includes functions to fit Conditional Gaussian Bayesian Network (CG-BN) to specifically genotype-phenotype or Quantitative Trait Loci (QTL) data, absorb evidence in these networks and quantify and visualize the changes in network beliefs.

The package makes extensive use of `RHugin` package 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 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 project homepage on R-Forge, `http://genenetbp.r-forge.r-project.org`.

Please note that `RHugin` is required for proper functioning of `geneNetBP`. The package `RHugin` will not automatically load upon loading `geneNetBP` package. Please use `library(RHugin)` or `require(RHugin) to load RHugin before using geneNetBP`.

2 Datasets

There are 3 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 has 2 variables `mousegeno`, data frame of 173 observations (genotype) of 5 variables (SNP markers) and `mousepheno`, data frame of 173 observations (normalized gene expression) of 14 variables (genes).

Load the dataset and view the first 3 observations:

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

```
  Qchr4 Qchr17 Qchr15 Qchr11 Qchr2
1     2      3      2      2     2
2     1      3      2   <NA>     2
3     3      2      2      2     2
```

```
> head(mousepheno,n=3)
```

```
     Cyp4a31     Slc5a9     Slc6a9      Hmgcl     Ptp4a2        Ak2     Zbtb8a
1 -0.8581591 -1.1433976  2.1143808 -0.3683079  1.2006550  0.4149740  0.5443409
2  1.8186456  1.7480246 -1.7480246 -1.5763614  1.8186456 -1.0639390  1.0144987
3  0.2622828  0.3683079  0.6476036  0.1155036 -0.2177984  0.8581591 -1.0389014
        Stx12   Trspap1       Mecr      Wdtc1     Atpif1     Rbbp4      Tlr12
1  0.02881581 -1.014499 -0.4625623 -0.3224307 -1.1433976  1.364489 -0.5277093
2 -1.23081837  1.483540  2.2736256 -1.0144987  0.7018726 -1.995604  0.8581591
3  0.66547438 -1.685179 -0.7582926  0.9906857 -1.3288179  1.230818 -0.8375227
```

Note that 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 all columns in data frame `mousegeno` have to be of class factor while the phenotypes are continuous variables with all columns in data frame `mousepheno` of class numeric.

2.2 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 (`toygeno`) and 6 phenotypes, X1-X6 (`toypheno`).

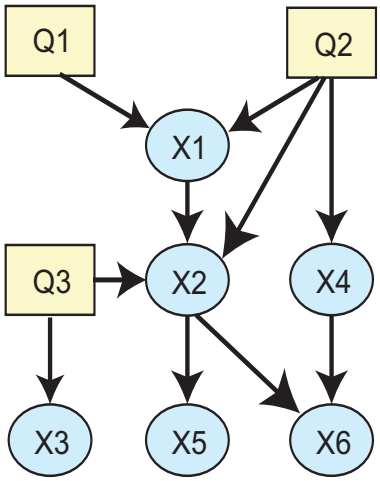

Figure 1. Toy network example.

2.3 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 has 2 variables `yeastgeno`, data frame of 112 observations (genotype) of 9 variables (SNP markers) and `yeastpheno`, data frame of 112 observations (normalized gene expression) of 16 variables (genes).

Load the dataset and view the first 3 observations:

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

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

3 Fit CG-BN to QTL data

3.1 Model

The graphical model is represented as 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.

3.2 Mouse 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.

3.2.1 fit.gnbp

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

```
> 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. An 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 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 RHugin domain, however it has minimal graphic capabilities. Refer to RHugin manual for more help on the plot method. We will plot the network using `Rgraphviz` package that has

several ways of rendering customized graphs. Good news is that the RHugin package has a function to coerce the RHugin domain into a graph object of class "graphNEL".

```
> network<-fit.gnbp(mousegeno,mousepheno,alpha = 0.1)
> ##convert the RHugin domain to a graph object
> BNgraph<-as.graph.RHuginDomain(network$gp)
> ##set node font size
> attrs<-list()
> attrs$node$fontsize<-30
> ## plot method for graph objects
> plot(BNgraph,attrs=attrs)
```

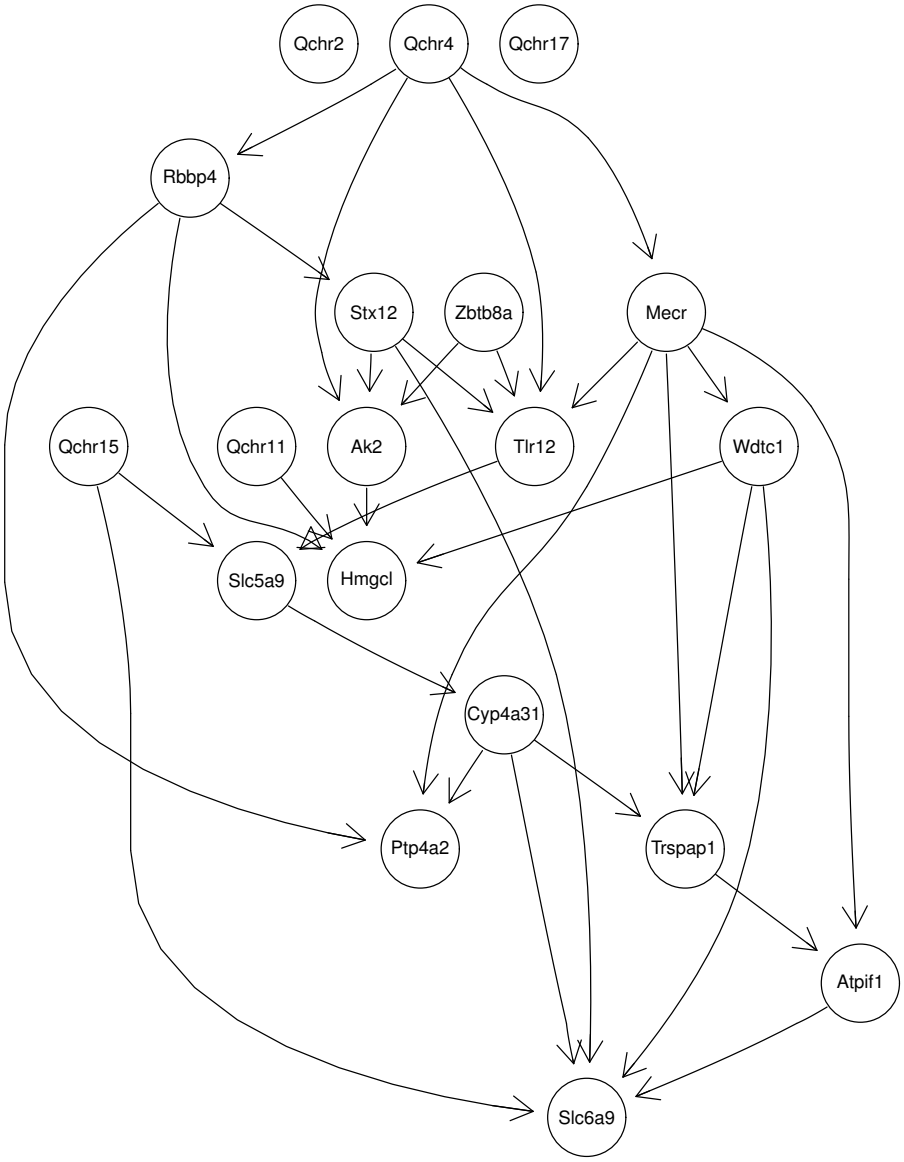

Figure 2. Conditional Gaussian network learnt from mouse QTL data

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.

4 Absorbing evidence and network comparison

4.1 Evidence Absorption and Belief Propagation

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.

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

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

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.

4.2 Mouse Example

Suppose we know the marginal mean of one of the nodes `Tlr12` is -0.99 and we wish to enter this new information in the mouse network and see 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.

4.2.1 absorb.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 after fitting a network to QTL data using geneNetBP package.

1. Absorb a single evidence for a single node

```
> network<-fit.gnbp(mousegeno,mousepheno,alpha=0.1)
> ## Absorb evidence
> absorb.gnbp(network,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

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

4.2.2 gen.evidence

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`

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

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.

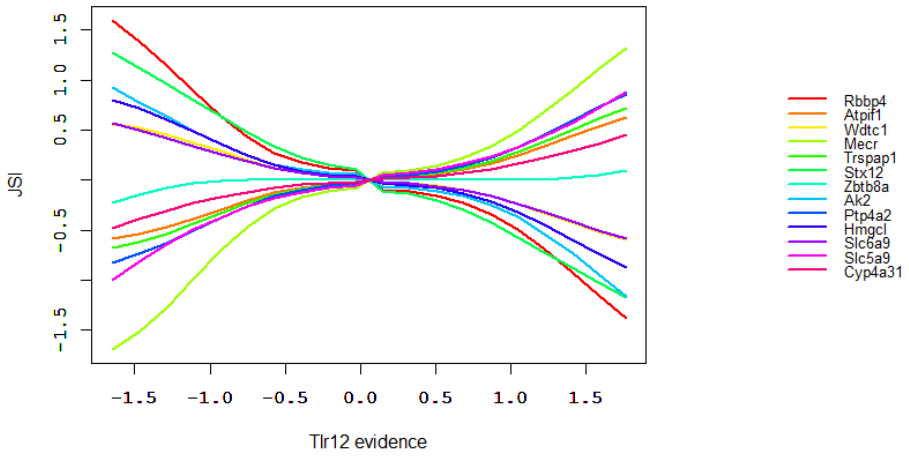

Figure 3. Plot produced by `absorb.gnbp`

5 Visualizing network changes

A generic plot method for plotting the genotype-phenotype network in which evidence has been absorbed and propagated is available. It is important to note that the input to this plot method is an object of class "gnbp". If a RHugin domain is input to plot, the corresponding plot method for RHugin domain will be used. The plot method will convert the RHugin domain into an object of class "graphNEL" by using Rgraphviz package as mentioned previously. The argument nodeAttrs to plot method for graph objects in Rgraphviz package is then used to customize the plot.

5.1 A complete example

A complete example that fits a network, 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)
```

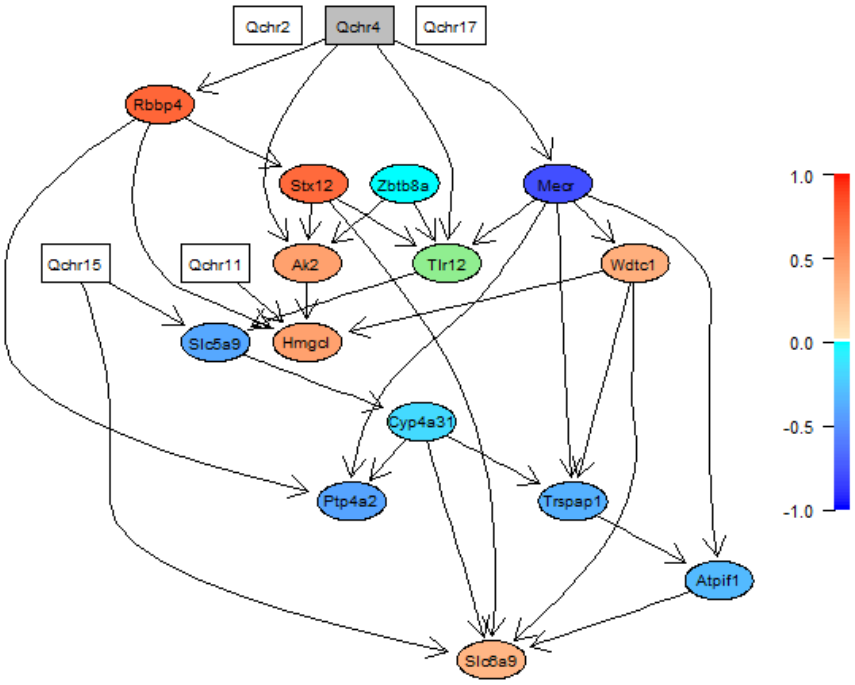

Figure 4. Evidence absorption in single node

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).

5.2 Plot options

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)
```

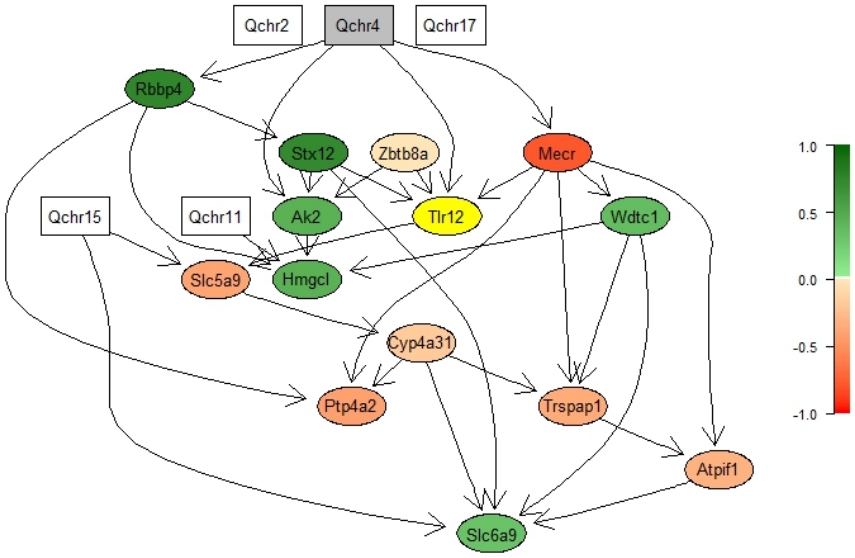

Figure 5. Mouse network with custom color palette

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 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)
```

6 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 seting `learn = FALSE`. The conditional probabilities will still need to be learnt. This section demonstrates how to specify known networks and subsequent belief propagation in a simulated toy example.

First create a list of known edges from parent to child.

```
>   ## Load the toy dataset
>   data(toy)
> ## Create a list of edges ("from (parent)", "to (child)")
> edgelist=list()
> 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")
```

In `fit.gnbp` provide the 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.

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

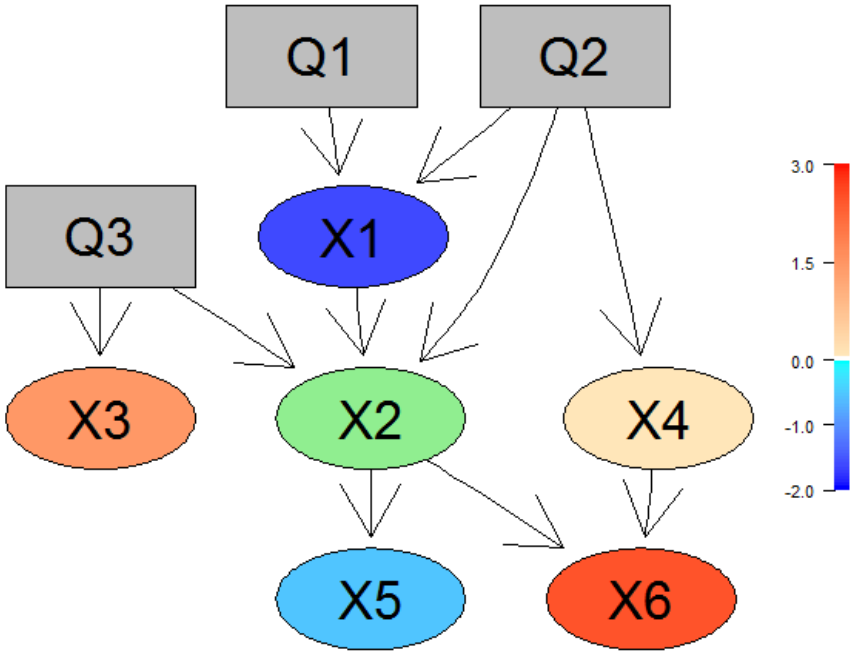

Figure 6. Belief propagation in known network

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.