QuaternaryProd

Carl Tony Fakhry, Ping Chen and Kourosh Zarringhalam

2017-10-30

A signed causal graph for gene regulation is a directed graph where the edges are signed and the signs indicate the direction of regulation of the target genes (the signs are either (+) or (-)). **QuaternaryProd** is a package for computing the Quaternary Dot Product Scoring Statistic [2] (or simply the Quaternary Statistic) of signed causal graphs for gene regulation. The Quaternary Dot Product Scoring Statistic is a generalization of the Ternary Dot Product Scoring Statistic [1] which allows for ambiguities to arise in the causal graph. Ambiguities arise when a regulator can affect a target gene in two different ways or if the direction of regulation is unknown. We will first provide some background, and then we will apply the statistic to STRINGdb [3] which is a publicly available biological network.

Introduction

The Quaternary Dot Product Scoring Statistic [2] is a goodness of fit test for evaluating the performance of regulation predictions made by a signed and directed causal network on a given gene expression data set. Given a regulator s in a causal graph, let q_p, q_m and q_r denote the number of target genes which are upregulated (+), downregulated (-) and regulated (r) by the regulator s respectively. Regulated relations occur when a regulator regulates a target gene without knowing the direction of regulation or if an ambiguity in direction of regulation occurs. An ambiguity can occur if a regulator, according to a given network, shares both (+) and (-) relations with the same target gene. Moreover, let q_z denote the set of target genes in the causal network which do not share a relation with s i.e which are not affected by s. Next, suppose we are presented with new gene expression data. Let n_p, n_m and n_z denote the number of genes which are upregulated, downregulated and are unregulated in the gene expression data respectively. For the regulator s, we can tabulate the predictions of the network vs. the gene expression data:

	Observed +	Observed −	Observed 0	Total
Predicted +	n_{pp}	n_{pm}	n_{pz}	q_p
Predicted −	n_{mp}	n_{mm}	n_{mz}	q_m
Predicted r	n_{rp}	n_{rm}	n_{rz}	q_r
Predicted 0	n_{zp}	n_{zm}	n_{zz}	q_z
Total	n_p	n_m	n_z	T

Table 1: Tabulation of predictions from network edges vs. observations from experimental results.

n_{pp} denotes the number of target genes which s is predicted to upregulate by the network and were indeed upregulated in the gene expression data. n_{pm} denotes the number of target genes which s is predicted to upregulate and were downregulated in the gene expression data. n_{pz} denotes the number of target genes which s is predicted to upregulate and were not expressed in the gene expression data. Similar interpretation follows for all other entries of the table. The probability of a tabulation table follows the Quaternary Dot Product distribution which is given by:

$$P(\text{Table}) = \frac{\binom{q_p}{n_{pp},n_{pm},n_{pz}}\binom{q_m}{n_{mp},n_{mm},n_{mz}}\binom{q_z}{n_{zp},n_{zm},n_{zz}}\binom{q_r}{n_{rp},n_{rm},n_{rz}}}{\binom{T}{n_p,n_m,n_z}}. \tag{1}$$

Note, since the predictions by the network and the experimental values are fixed, then the table has 6 degrees of freedom n_{pp}, n_{mm}, n_{rp}, n_{rm}, n_{mp} and n_{pm}. The score S to measure the goodness of fit is given by:

$$S(\text{Table}) = n_{pp} + n_{mm} + n_{rp} + n_{rm} - (n_{mp} + n_{pm}) \tag{2}$$

which is the sum of the good predictions (i.e n_{pp}, n_{mm}, n_{rp} and n_{rm}) minus the bad predictions (i.e n_{mp} and n_{pm}). To compute the probability of a score, we sum the probabilites of all tables with score S as follows:

$$P(S) = \sum_{P(\text{Table})=S} P(\text{Table}). \tag{3}$$

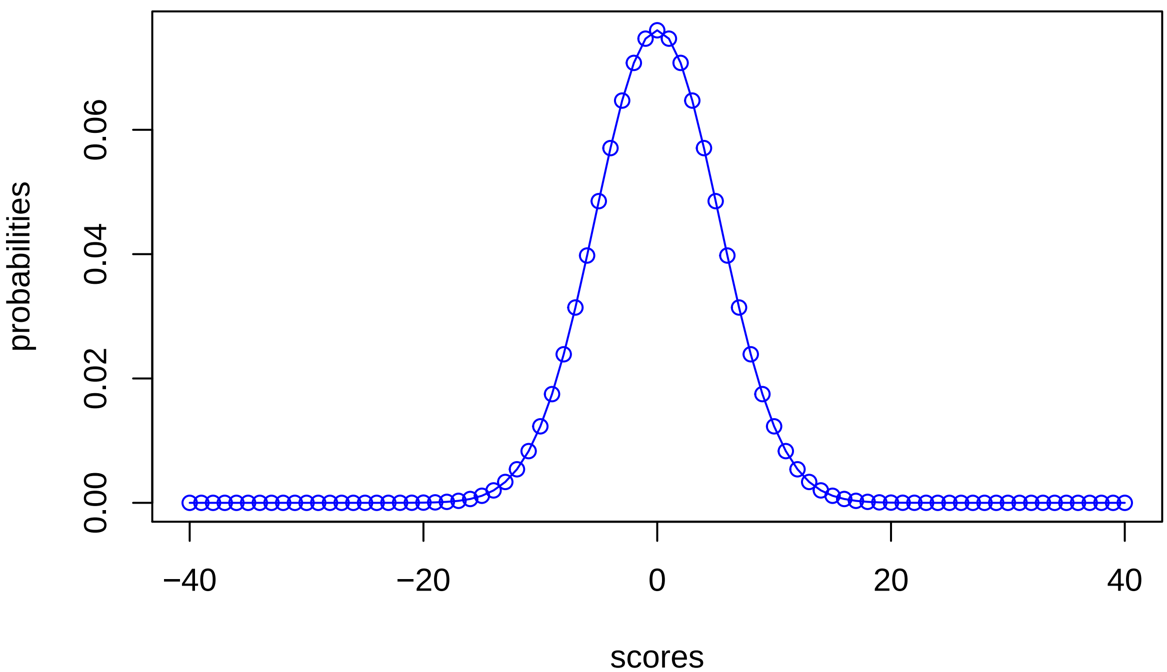
Functionality

QuaternaryProd provides different functions for computing the probability of a score, probability mass function, p-value of a score and the domain of the Quaternary Dot Product Scoring Statistic. The probability mass function can be computed if given the margins of the table.

```
library(QuaternaryProd)

# Compute the probability mass function
pmf <- QP_Pmf(q_p = 20, q_m = 20, q_z = 20, q_r = 0, n_p = 20, n_m = 20, n_z = 20)

# Plot the mass function
plot(names(pmf), pmf, col="blue", xlab = "scores", ylab = "probabilities")
lines(names(pmf), pmf, col = "blue")
```



The package contains optimized functions for computing the p-value of a score. To compute the p-value of score we can use the following:

```
# Get the p-value of score 5
pval <- QP_Pvalue(score = 5, q_p = 20, q_m = 20, q_z = 20, q_r = 0,
                                                  n_p = 20, n_m = 20, n_z = 20)
pval
```

```
## [1] 0.1948157
```

```r
# Compue the p-value only if it is statistically significant otherwise
# return -1
pval <- QP_SigPvalue(score = 5, q_p = 20, q_m = 20, q_z = 20, q_r = 0,
                                                    n_p = 20, n_m = 20, n_z = 20)
pval
```

```
## [1] -1
```

If the user is only interested in obtaining statistically significant p-values, then `QP_SigPvalue` is optimized for this purpose. In either case, the user is advised to compute the p-value of a score using the previous two functions which will be faster than computing the entire probability mass function and then computing the p-value. Finally, it is possible to also compute the probabilities of scores individually using `QP_Probability` and the support of the distribution using `QP_Support`. Since this package is written for the benefit of bioinformaticians, we will provide an example on how to apply this statistic to a publicly available network. One bioinformatic application is to test how well protein-protein causal networks can predict the regulators in gene expression data. In the last section of this Vignette, we present an example for computing this statistic over the STRINGdb network.

Functionality for working with the Homo sapien causal network from STRINGdb

Here we provide functionality for using **QuaternaryProd** with the STRINGdb Homo Sapien causal network (version 10) provided under the creative commons license.

Compute Pvalues Over the Network

Given new gene expression data, we can compute the scores and p-values for all regulators in the STRINGdb Homo Sapien causal network using the specialized `RunCRE_HSAStringDB` function. We use the gene expression data sets that were used in [1]. The data sets contain the c-Myc and E2F3 expression signatures. Note that the results may differ from those reported in [2] since the network was parsed differently.

Load Gene Expression Data

First, we load all the data sets. The gene expression data sets must have the following columns: 1- `entrez` column corresponding to the entrez id of the gene, 2- `pvalue` column corresponding to the pvalue of the gene, 3- `fc` column corresponding to the fold change of the gene. After we load the data sets, we make sure that there are no duplicated entrez ids in the data sets.

```r
library(QuaternaryProd)

# Get gene expression data
e2f3 <- system.file("extdata", "e2f3_sig.txt",
                             package = "QuaternaryProd")
e2f3 <- read.table(e2f3, sep = "\t",
                             header = TRUE, stringsAsFactors = FALSE)
myc <- system.file("extdata", "myc_sig.txt",
                             package = "QuaternaryProd")
myc <- read.table(myc, sep = "\t",
                             header = TRUE, stringsAsFactors = FALSE)

# Rename column names appropriately
# and remove duplicated entrez ids in the gene expression data
names(e2f3) <- c("entrez", "pvalue", "fc")
```

```r
e2f3 <- e2f3[!duplicated(e2f3$entrez),]

names(myc) <- c("entrez", "pvalue", "fc")
myc <- myc[!duplicated(myc$entrez),]
```

Compute the Quaternary Dot Product Scoring Statistic over STRINGdb

We can now compute the Quaternary Dot Product Scoring Statistic over STRINGdb using the following:

```r
# Compute the Quaternary Dot Product Scoring Statistic for only statistically
# significant regulators
quaternary_results <- RunCRE_HSAStringDB(e2f3, method = "Quaternary",
                                 fc.thresh = log2(1.3), pval.thresh = 0.05,
                                 only.significant.pvalues = TRUE,
                                 significance.level = 0.05)
```

```
## 137 rows from gene_expression_data removed due
##  to entrez ids being unrepsented in StringDB entities!
```

```r
quaternary_results[1:4, c("uid","symbol","regulation","pvalue")]
```

```
##                    uid    symbol regulation       pvalue
## 1 9606.ENSP00000345571      E2F1         up 7.810196e-09
## 2 9606.ENSP00000244741    CDKN1A       down 2.454842e-07
## 3 9606.ENSP00000362592     RBBP4       down 3.084143e-05
## 4 9606.ENSP00000379140 No-Symbol         up 5.875947e-05
```

`RunCRE_HSAStringDB` returns a data frame containing all the regulators of the String causal network. Computing the p-value of the Quaternary Dot Product Scoring Statistic has computational time complexity $\mathcal{O}(kn^3)$ (see [2]). To improve performance, `RunCRE_HSAStringDB` has an optional argument `only.significant.pvalues` which can be set to `TRUE` so that only the p-values of the statistically significant regulators are computed. P-values which are not statistically significant (i.e p-values greater than `signficance.level`) are not reported, in which case the p-value is set to a value of -1. If the user wishes to compute p-values for regulators which are not statistically significant then the user should set the parameter `only.significant.pvalues = FALSE`. The regulators are ordered in increasing order of the p-values (Note: details on the columns of the data frame returned can be found in the help page for `RunCRE_HSAStringDB`). Finally, we see that this approach retrieves the signal regulator E2F1.

Compute the Ternary Dot Product Scoring Statistic and the Enrichment test over STRINGdb

To compute the Ternary Dot Product Statistic over STRINGdb we can use the following:

```r
ternary_results <- RunCRE_HSAStringDB(myc, method = "Ternary",
                                      fc.thresh = log2(1.3), pval.thresh = 0.05,
                                      only.significant.pvalues = TRUE,
                                      significance.level = 0.05)
```

```
## 108 rows from gene_expression_data removed due
##  to entrez ids being unrepsented in StringDB entities!
```

```r
ternary_results[1:4, c("uid","symbol","regulation","pvalue")]
```

```
##                    uid    symbol regulation       pvalue
## 1 9606.ENSP00000367207       MYC         up 3.511516e-06
## 2 9606.ENSP00000351490       MAX         up 3.097577e-05
```

```
## 3 9606.ENSP00000313199 No-Symbol         up 4.226215e-05
## 4 9606.ENSP00000258962     SRSF1         up 2.867982e-04
```

We see that this method retrieves MYC as a significant regulator. To compute the Enrichment test over STRINGdb we can use the following:

```r
enrichment_results <- RunCRE_HSAStringDB(myc, method = "Enrichment",
                                         fc.thresh = log2(1.3), pval.thresh = 0.05,
                                         only.significant.pvalues = TRUE,
                                         significance.level = 0.05)
```

```
## 108 rows from gene_expression_data removed due
##  to entrez ids being unrepsented in StringDB entities!
```

```r
enrichment_results[1:10, c("uid","symbol","regulation","pvalue")]
```

```
##                      uid symbol regulation       pvalue
## 1  9606.ENSP00000351490    MAX         up 2.916784e-06
## 2  9606.ENSP00000351490    MAX       down 2.916784e-06
## 3  9606.ENSP00000355249   E2F2         up 2.216447e-05
## 4  9606.ENSP00000355249   E2F2       down 2.216447e-05
## 5  9606.ENSP00000256996   DDB2         up 8.026163e-05
## 6  9606.ENSP00000256996   DDB2       down 8.026163e-05
## 7  9606.ENSP00000425561  EIF4E         up 2.868088e-04
## 8  9606.ENSP00000425561  EIF4E       down 2.868088e-04
## 9  9606.ENSP00000367207    MYC         up 3.108668e-04
## 10 9606.ENSP00000367207    MYC       down 3.108668e-04
```

We see that the Enrichment method also retrieves MYC as a significant regulator although not as significant as in the case of the Ternary Dot Product Scoring Statistic.

References

[1] Chindelevitch et al. (2012). Assessing statistical significance in causal graphs. BMC Bioinformatics, Volume 3, Issue 1, 2012, Page 35.

[2] Carl Tony Fakhry, Parul Choudhary, Alex Gutteridge, Ben Sidders, Ping Chen, Daniel Ziemek, and Kourosh Zarringhalam. Interpreting transcriptional changes using causal graphs: new methods and their practical utility on public networks. BMC Bioinformatics, 17:318, 2016. ISSN 1471-2105. doi: 10.1186/s12859-016-1181-8.

[3] Franceschini, A (2013). STRING v9.1: protein-protein interaction networks, with increased coverage and integration. In:'Nucleic Acids Res. 2013 Jan;41(Database issue):D808-15. doi: 10.1093/nar/gks1094. Epub 2012 Nov 29'.