QuaternaryProd

Carl Tony Fakhry, Ping Chen and Kourosh Zarringhalam

2017-04-24

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

Introduction

The Quaternary Dot Product Scoring Statistic [2] is a goodness of fit test for examining how well the predictions of a signed and directed causal graph predict on newly realized experimental data. Given a source node s in a signed causal graph, let q_p, q_m and q_r denote the number of target nodes which are increased, decreased and regulated by the source node respectively. Similarly, let q_z denote the set of target nodes in the causal network which do not share a relation with s i.e which are not affected by s. Regulated relations occur when a source node regulates a target node without knowing the direction of causality or if an ambiguity in direction of causality occurs. An ambiguity can occur if a source node, according to a given network, shares both increase and decrease relations with the same target node. Next, Suppose we run some experiments on entities which are target nodes in the network. Let n_p, n_m and n_z denote the set of values which are increased, decreased and remain unchanged in the experimental values respectively. For the source node s, we can tabulate the predictions from the network vs. the experimental values:

	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 nodes which s is predicted to increase by the network and were indeed increased in experimental values; n_{pm} the number of target nodes which s is predicted to increase and were decreased in experimental values; n_{pz} is the number of target nodes which s is predicted to increase and were unchanged in experimental values. Similar interpretation follows for all other entries of the table. The probability of a 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:

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

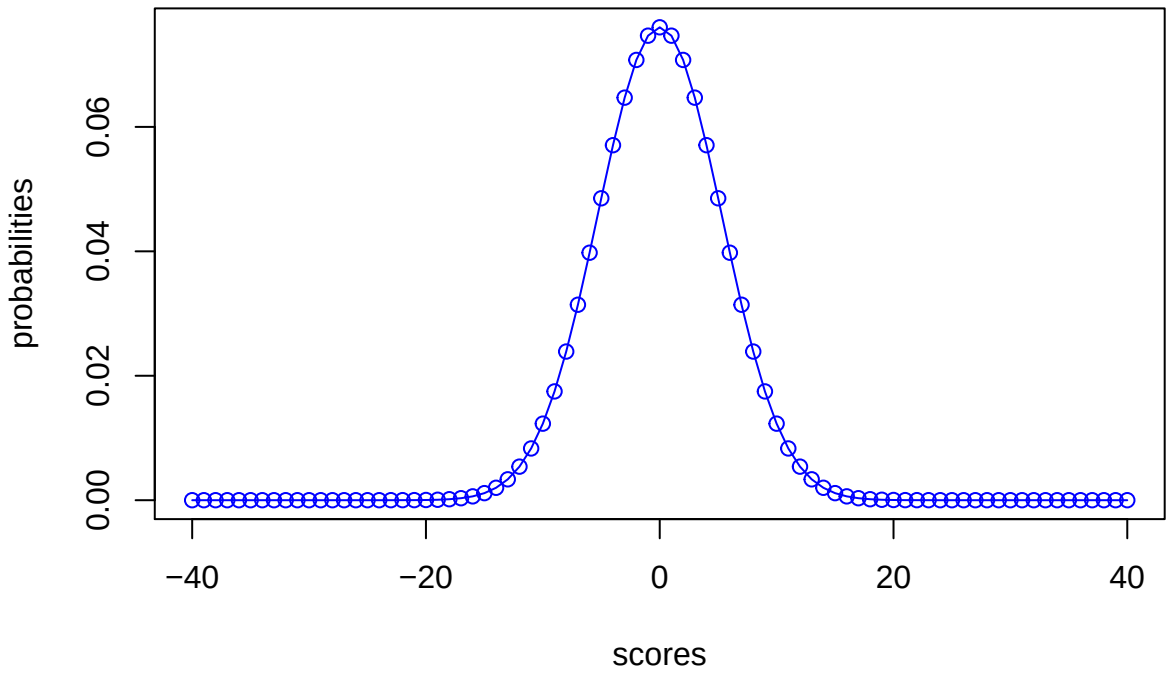
Functionality

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
```

```
# 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 to 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 results in gene expression data. In the last section of this Vignette, we present an example of computing this statistic over the Stringdb network and given gene expression data.

Edges from STRINGdb

You can use the **STRINGdb** package to interact with the String database. The current release of **STRINGdb** only allows querying of neighbors in the network without the direction of action. For the purposes of the Quaternary Dot Product Scoring Statistic, it is necessary to have the direction of action. It is possible to obtain the signed network by downloading it directly from String-db network source, selecting the species of interest, and downloading the protein actions network. Here, we present an example for working with the freely available Homo Sapien protein-protein interaction network from STRINGdb. The network is in tab seperated format which is straightforward to work with. It is possible to get a larger version of network with more relations from STRINGdb by signing a license agreement with the authors of STRINGdb.

Parse File

In this section, we provide an example of one possible way to parse the Homo Sapien protein actions network and prepare it to be used with our package. First, we need to upload the network which is attached to **QuaternaryProd** for convenience.

```
library(QuaternaryProd)
library(readr)
library(org.Hs.eg.db)
library(dplyr)
library(stringr)
library(fdrtool)

# Get the full file name containing the STRINGdb relations
ff <- system.file("extdata", "9606.protein.actions.v10.txt.gz", package="QuaternaryProd")
all_rels <- read_tsv(gzfile(ff), col_names = TRUE)
```

Next, we filter out the important columns and important relations. We remove all rows which do not have a relation *activation*, *inhibition* and *expression*. Moreover, we also consider reverse causality for any relation which has a *direction* value equal to 0.

```
# Set new names for columns
names(all_rels) <- c("srcuid", "trguid", "mode", "action", "direction","score")
Rels <- all_rels[, c("srcuid", "trguid", "mode", "direction")]
```

```
# Get all rows with causal relations
Rels <- Rels[Rels$mode %in% c("activation", "inhibition","expression"),]

# Get causal relations where direction is not specified, and consider reversed
# direction of causality as a valid causal relation
Bidirectional <- Rels[Rels$direction == 0 , c("trguid", "srcuid", "mode", "direction")]
names(Bidirectional) <- c("srcuid", "trguid", "mode", "direction")
Rels <- unique(bind_rows(Rels, Bidirectional))
Rels$direction <- NULL

# Rename activation as increases, inhibition as decreases, expression
# as regulates
Rels$mode <- sub("activation", "increases", Rels$mode)
Rels$mode <- sub("inhibition", "decreases", Rels$mode)
Rels$mode <- sub("expression", "regulates", Rels$mode)
Rels <- unique(Rels)

# Get a subset of the network: Skip this step if you want the p-values
# of the scores corresponding to the source nodes computed over the
# entire network.
Rels <- Rels[sample(1:nrow(Rels), 40000, replace=FALSE),]
```

Third, we extract the protein entities from the network, and we map them to their respective genes. Note, the entities could have been possibly a drug or compound, but we are working with this protein interactions network for the purpose of providing a nontrivial example.

```
# Get all unique protein ensemble ids in the causal network
allEns <- unique(c(Rels$srcuid, Rels$trguid))

# Map ensemble protein ids to entrez gene ids
map <- org.Hs.egENSEMBLPROT2EG
id <- unlist(mget(sub("9606.","",allEns), map, ifnotfound=NA))
id[is.na(id)] <- "-1"
uid <- paste("9606.", names(id), sep="")

# Function to map entrez ids to gene symbols
map <- org.Hs.egSYMBOL
symbol <- unlist(mget(id, map, ifnotfound=NA))
symbol[is.na(symbol)] <- "-1"

# Create data frame of STRINGdb protein Id, entrez id and gene symbol and type of entity
Ents <- data_frame(uid, id, symbol, type="protein")
Ents <- Ents[Ents$uid %in% allEns,]

# Remove ensemble ids in entities with duplicated entrez id
Ents <- Ents[!duplicated(Ents$id),]

# Add mRNAs to entities
uid <- paste("mRNA_", Ents$uid, sep = "")
mRNAs <- data_frame(uid=uid, id=Ents$id, symbol=Ents$symbol, type="mRNA")
Ents <- bind_rows(Ents, mRNAs)
```

Finally, we filter unique relations in the network, and remove source proteins which do not have more than 10 children in the network.

```
# Get all unique relations
Rels$trguid <- paste("mRNA_", Rels$trguid, sep="")
Rels <- Rels[Rels$srcuid %in% Ents$uid & Rels$trguid %in% Ents$uid,]
Rels <- unique(Rels)

# Leave source proteins which contain at least 10 edges
sufficientRels <- group_by(Rels, srcuid) %>% summarise(count=n())
sufficientRels <- sufficientRels %>% filter(count > 10)
Rels <- Rels %>% filter(srcuid %in% sufficientRels$srcuid)
```

Compute Pvalues Over the Network

Given new gene expression data, we can compute the scores and p-values for all source nodes in the network. *BioQCREtoNet* is a specialized function for this purpose.

```
# Gene expression data
evidence1 <- system.file("extdata", "e2f3_sig.txt", package = "QuaternaryProd")
evidence1 <- read.table(evidence1, sep = "\t", header = TRUE, stringsAsFactors = FALSE)
evidence2 <- system.file("extdata", "myc_sig.txt", package = "QuaternaryProd")
evidence2 <- read.table(evidence2, sep = "\t", header = TRUE, stringsAsFactors = FALSE)
evidence3 <- system.file("extdata", "ras_sig.txt", package = "QuaternaryProd")
evidence3 <- read.table(evidence3, sep = "\t", header = TRUE, stringsAsFactors = FALSE)

# Remove duplicated entrez ids in evidence and rename column names appropriately
names(evidence1) <- c("entrez", "pvalue", "fc")
evidence1 <- evidence1[!duplicated(evidence1$entrez),]

names(evidence2) <- c("entrez", "pvalue", "fc")
evidence2 <- evidence2[!duplicated(evidence2$entrez),]

names(evidence3) <- c("entrez", "pvalue", "fc")
evidence3 <- evidence3[!duplicated(evidence3$entrez),]

# Run Quaternary CRE for entire Knowledge base on new evidence
# which computes the statistic for each of the source proteins

CRE_results <- BioQCREtoNet(Rels, evidence1, Ents, is.Logfc = TRUE)
```

```
## [1] "217  rows from evidence removed due to entrez ids being unrepsented in entities!"
```

```
# Get FDR corrected p-values
CRE_results$pvalue <- fdrtool(CRE_results$pvalue, "pvalue", FALSE,
                                                    FALSE, FALSE, "fndr")$q
```

```
## Warning in fdrtool(CRE_results$pvalue, "pvalue", FALSE, FALSE, FALSE,
## "fndr"): There may be too few input test statistics for reliable FDR
## calculations!
```

```
head(CRE_results[order(CRE_results$pvalue), c("uid","name","pvalue")])
```

```
##                     uid   name pvalue
## 1 9606.ENSP00000272233   RHOB      1
## 2 9606.ENSP00000339136   ERAS      1
## 3 9606.ENSP00000267205   RHOF      1
## 4 9606.ENSP00000258411 WNT10A      1
```

```
## 5 9606.ENSP00000298532 SNAPC4      1
## 6 9606.ENSP00000272164  WNT9A      1
```

BioQCREtoNet returns a data frame containing all the source nodes of the causal network, all of which had their respective score p-value computed. The source nodes are ordered in increasing order (Note: details on the columns of the data frame returned can be found in the help page for *BioQCREtoNet*).

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'.