MarkRank Tutorial

Ling-Yun Wu

2017-04-28

Contents

1 MarkRank algorithm

MarkRank is a novel proposed network-based model for identifying the cooperative biomarkers. MarkRank uses the gene cooperation network to explicitly evaluate the gene cooperative effects. MarkRank suggests that explicit modeling of gene cooperative effects can greatly improve biomarker identification for complex diseases, especially for diseases with high heterogeneity. This tutorial could help the user to execute the markrank function compiled in the Corbi package.

We imported the Corbi and other required packages:

```r
rm(list=ls(all=TRUE))
library(Corbi)
library(Matrix)
```

```
##
## Attaching package: 'Matrix'

## The following object is masked from 'package:Corbi':
##
##     nnzero
```

```r
options(scipen=0)
```

2 MarkRank example

The inputs of markrank function is an expression dataset of known label (disease and normal) samples and an adjacent matrix of biological network. We use the simulation datasets to show the usage of markrank function.

2.1 Generate the model inputs

First, we load in a small network using another function read_net compiled in Corbi.

```
sub_info <- read_net("network.txt")
```

This network contains 100 genes. Then we set the number of preset differential expression nodes.

```
size <- 10
```

We randomly extract a connected subnetwork with the preset size from the loaded network. Here we use the function search_net to implement.

```
source("search_net.R")
preset  <- search_net(sub_info, node_size = size, ori_name = TRUE)
de_list <- as.character(unique(as.vector(preset)))
```

The preset differentially expression genes are:

```
de_list
```

```
##  [1] "92" "49" "94" "57" "61" "87" "69" "80" "68" "43"
```

Now we simulate the expression matrix. The sample number is set as

```
sam_num <- 50
```

The number of disease samples and normal samples are equal.

```
half <- 25
```

The code of simulating the expression dataset is as follows. We up-regulated the expression values of preset differentially expression gene set. The detailed description of this process can be found in the Supplementary Materials in our manuscript.

```
library(matrixcalc)
library(MASS)
l <- sub_info$size
p <- length(de_list)
dataset <- matrix(0, sam_num, l, dimnames = list(paste("sample", 1:sam_num, sep=""), sub_info$node))
vars  <- 1
sigma <- matrix(0)
while(!is.positive.definite(sigma)){
  vars  <- vars + 1
  sigma <- as.matrix(as(sub_info$matrix,'dgCMatrix'))
  sigma[which(sigma == 1)] <- rnorm(length(which(sigma == 1)), 4, 1)
  sigma[which(sigma == 0)] <- rnorm(length(which(sigma == 0)), 2, 1)
  diag(sigma) <- rnorm(l, vars, 1)
  sigma <- (sigma + t(sigma))/2
}
multi_mean <- rnorm(l, 5, 1)
dataset <- mvrnorm(sam_num, multi_mean, sigma)
dataset[1:half, de_list] <- dataset[1:half, de_list] * rnorm(half*p, 2, 0.1)
```

The final simulated microarray expression matrix contains 50 samples and 100 genes. The number of preset marker genes is 10.

```
dim(dataset)
```

```
## [1]  50 100
```

The sample label is

```
label <- c(rep(0, half), rep(1, half))
```

The adjacent matrix of the network is

```
adj_matrix <- as.matrix(sub_info$matrix)
adj_matrix <- adj_matrix[colnames(dataset), colnames(dataset)]
adj_matrix <- adj_matrix + t(adj_matrix)
```

2.2 Run the markrank function

With the above simulated variables as model inputs, we now execute the markrank function to test whether MarkRank could prioritize the preset genes. We use the default parameter combination as alpha=0.8 and lambda=0.2 to run the markrank.

```
time1 <- system.time(
  result1 <- markrank(dataset, label, adj_matrix, alpha=0.8, lambda=0.2, trace=TRUE)
  )
```

```
## [1] "Computing discriminative potential network ..."
## [1] 10
## [1] 20
## [1] 30
## [1] 40
## [1] 50
## [1] 60
## [1] 70
## [1] 80
## [1] 90
```

The output variable result contains the following model outputs:

```
names(result1)
```

```
## [1] "score"        "steps"        "NET2"         "initial_pars"
## [5] "dis"
```

The scores of top 10 markrank genes are:

```
s1 <- sort(result1$score, decreasing=TRUE)
s1[1:10]
```

```
##         80         43         61         69         49         57 
## 0.11986202 0.06877236 0.05621261 0.05256604 0.04522431 0.04387191 
##         87         94         92         68 
## 0.04201298 0.03803157 0.02114340 0.02011342
```

The scores of pre-set differential expression genes are:

```
result1$score[de_list]
```

```
##         92         49         94         57         61         87 
## 0.02114340 0.04522431 0.03803157 0.04387191 0.05621261 0.04201298 
##         69         80         68         43 
## 0.05256604 0.11986202 0.02011342 0.06877236
```

The false discovery genes are:

```
setdiff(names(s1[1:10]), de_list)
```

```
## character(0)
```

The iteration steps in the random walk iteration is

```
result1$steps
```

```
## [1] 114
```

The user could find the input parameters by using the following code:

```
result1$initial_pars
```

```
## $alpha
## [1] 0.8
## 
## $lambda
## [1] 0.2
## 
## $eps
## [1] 1e-10
```

3 Tune the model parameters

The computation of gene cooperation network (G2) is separated from the random walk iteration. The computation result of G2 is stored in

```
NET2 <- result1$NET2
```

Using the parameter Given_NET2, we could tune the parameters without the repeated computation of G2 For example, we use the alpha=0.8 and lambda=0.5 to compute the related result:

```
time2 <- system.time(
  result2 <- markrank(dataset, label, adj_matrix, lambda=0.5, trace=FALSE, Given_NET2=NET2)
  )
```

The running time of two models is:

```
time1
```

```
##    user  system elapsed
##    2.23    0.00    2.16
```

```
time2
```

```
##    user  system elapsed
##    0.13    0.00    0.13
```

The running time of result2 is far less than result1, because the result2 just contains the random walk iteration step. Now the new scores of top 10 markrank genes are:

```
s2 <- sort(result2$score, decreasing=TRUE)
s2[1:10]
```

```
##         80         49         43         69         94         57
## 0.07669075 0.06690943 0.04427958 0.03959001 0.03829175 0.03754659
##         61         87         40         68
## 0.03493058 0.02957750 0.02762423 0.02163369
```

The scores of pre-set differential expression genes are:

```
result2$score[de_list]
```

```
##         92         49         94         57         61         87
## 0.01925936 0.06690943 0.03829175 0.03754659 0.03493058 0.02957750
##         69         80         68         43
## 0.03959001 0.07669075 0.02163369 0.04427958
```

The false discovery genes are:

```
setdiff(names(s2[1:10]), de_list)
```

```
## [1] "40"
```

4 The simplified version for G2 construction

By using the input parameter d, markrank could reduce the computation time on the construction of G2. Only the gene pairs whose shortest distances in the PPI network are less than d participate in G2 construction. For example, we could run

```r
time3 <- system.time(
  result3 <- markrank(dataset, label, adj_matrix, trace=F, d=2)
  )
```

The running time of two models is:

```r
time1
```

```
##    user  system elapsed 
##    2.23    0.00    2.16
```

```r
time3
```

```
##    user  system elapsed 
##    0.68    0.00    0.69
```

In this situation, the distance information of each gene pair can be found in output variable dis. For example, the distance matrix of gene 1 to 10 is:

```r
result3$dis[1:10,1:10]
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    0    1    2    3    2    3    3    3    3     4
##  [2,]    1    0    1    2    1    2    2    2    2     3
##  [3,]    2    1    0    1    1    2    2    2    2     3
##  [4,]    3    2    1    0    1    2    2    2    2     3
##  [5,]    2    1    1    1    0    1    1    1    1     2
##  [6,]    3    2    2    2    1    0    2    1    1     2
##  [7,]    3    2    2    2    1    2    0    2    1     3
##  [8,]    3    2    2    2    1    1    2    0    1     1
##  [9,]    3    2    2    2    1    1    1    1    0     2
## [10,]    4    3    3    3    2    2    3    1    2     0
```

The user should balance the calculation depth with computation time to achieve a acceptable result.