Computing pairwise distances between different biological states

Yang Cao, Lu Han, Fei Li, Xiaochen Bo

March 30, 2012

Contents

1 Introduction **1**

2 Getting Started **1**
2.1 Data 1
2.2 Aggregating 2
2.3 Similarity Measuring 3

3 Implementation Details **4**
3.1 Aggregating 4
3.2 Similarity Measuring 5

1 Introduction

The *GeneExpressionSignature* package utilizes gene expression profile to measure the similarity between different biological states. The similarity metric implemented here is mentioned in [Iorio et al., 2010]. A further description of the measurement methods based on gene expression signature can be found in Lamb[Lamb et al., 2006], Hu[Hu and Agarwal, 2009] and Iorio[Iorio et al., 2010].

2 Getting Started

The basic analysis process based on gene expression signature can be divided into the following steps: data preprocessing, aggregating, and similarity measuring. This package includes the functions used in aggregating and similarity measuring.

2.1 Data

Once the expression data is properly preprocessed, the Protype Ranked List (PRL) can be obtained by the relative ranking of the gene expression fold-change ratio. Here, one or several PRLs may be needed to describe a responsive state.

As an example, we load the PRLs from sample data first. This *PRLs* is composed of 10 columns corresponding to six compounds, so the PRLs should

be a $22283 * 10$ matrix, where 22283 is the length of gene expression profile. The values are given as rank scores.

```
> library(GeneExpressionSignature)
> PRLs <- as.matrix(read.table(system.file("extdata/example_PRLs.txt",
+ package="GeneExpressionSignature")))
> states <- read.table(system.file("extdata/example_states.txt",
+ package="GeneExpressionSignature"))
> PRLs[c(1:10),c(1:3)]
```

```
      V1    V2    V3
1   6432 12201  5085
2   4817 15753 17671
3  21798 17152 21366
4   3435  1669  2628
5   7647  2382  4808
6   7512  1645  2037
7   4652 15256 15503
8   1815  1525  1549
9   9960  8702 11778
10 13283  7567 11566
```

```
> states
```

```
                V1
1         metformin
2         metformin
3         metformin
4         metformin
5        phenformin
6  phenyl_biguanide
7     valproic_acid
8         metformin
9         estradiol
10  alpha-estradiol
```

2.2 Aggregating

If in the cases that multiple PRLs are assigned to one single state, aggregating process should be performed before similarity measuring. For instance, the metformin state corresponds multiple PRLs, Function *krubor* aggregates these PRLs into one single PRL representing the metformin state.

```
> PRL <- krubor(PRLs[,c(1:4,8)])
> PRL[c(1:10)]
```

```
 [1]  2145  8961 18114  1198  4524  7568 20783   183  6197 17545
```

In most cases, we deal with many PRLs of multiple states in one time. These states have one or more corresponding PRLs. That is really boring to process them one by one. To avoid unnecessary inconvenience, Function *aggregate* aggregates the PRLs all together. The only thing we must do is to wrap the

PRLs and the corresponding states with a ExpressionSet object, then call the *aggregate* function.

```
> library(Biobase)
> rownames(states)=colnames(PRLs)
> phenodata=new("AnnotatedDataFrame", data = states)
> exampleSet=new("ExpressionSet", exprs=PRLs, phenoData=phenodata)
> summary(exampleSet)

       Length         Class          Mode
            1 ExpressionSet            S4

> aggregatedSet=aggregate(exampleSet)
> summary(aggregatedSet)

       Length         Class          Mode
            1 ExpressionSet            S4
```

2.3 Similarity Measuring

After that, we obtain the aggregated PRLs with one PRL for one state. The Once ranked lists was aggregated, users can use function *distances* to compute the distances among different biological states. It computes distances according to function *quickenrichmentscore*, users will obtain a distance-matrix by this function. This function has two arguments, the first argument is the PRL obtained by function *aggregate*, and the second argument is qlen value, which is the length of signature. We take qlen for 250 in our example. We will get a $6 * 6$ matrix corresponding to six compounds, the $m * n$ element represents the distance between mth and nth states.

```
> d  <- distances(aggregatedSet,250)
> ES <- d[[1]]
> exprs(ES)

             1           2          3         4          5           6
1  0.000000000  0.585185494  0.1213122 0.1634067 -0.1544331 -0.007530704
2  0.532347116  1.000000000 -0.1205890 0.1961283  0.1856957  0.186878500
3  0.223519811 -0.174526029  1.0000000 0.4715567  0.4627624  0.407694458
4 -0.009746562 -0.004706486  0.3415942 1.0000000  0.6527843  0.612994508
5  0.009272909  0.148695230  0.3452670 0.6665296  1.0000000  0.648401943
6 -0.006462397 -0.016785277  0.3229811 0.5752355  0.5923120  1.000000000

> distance <- d[[2]]
> exprs(distance)

          1         2         3         4         5         6
1 1.0000000 0.4412337 0.8275840 0.9231699 1.0725801 1.0069966
2 0.4412337 0.0000000 1.1475575 0.9042891 0.8328045 0.9149534
3 0.8275840 1.1475575 0.0000000 0.5934245 0.5959853 0.6346622
4 0.9231699 0.9042891 0.5934245 0.0000000 0.3403431 0.4058850
5 1.0725801 0.8328045 0.5959853 0.3403431 0.0000000 0.3796430
6 1.0069966 0.9149534 0.6346622 0.4058850 0.3796430 0.0000000
```

3 Implementation Details

Finally, using the GSEA algorithms (*distances*) to compute the distances among biological states.

3.1 Aggregating

In order to get a ranked list of genes for each treatment by aggregation, the distance of these lists must be calculated first, as follows: for the ranked lists with the same biological state, a measure of the distance between two ranked lists is computed using *Spearman* algorithm, function *FootruleMatrix* compute distances between any two ranked lists, and create a $n * n$ matrix, where n is the number of ranked lists. Next, merge the two or more ranked lists treated with the same biological state using *Borda merging* algorithm. The function *BMRankMerging*, merging two or more selected ranked lists into a new one ranked list, implements the *Borda merging* algorithm. After all, function *krubor* aggregate all ranked lists into one list to get a single ranked List with *Kruskal* algorithm.

According to the *Kruskal* algorithm method[Cormen et al., 1990], the aggregating algorithm searches for the two ranked lists with the smallest Spearman's Footrule distance first, and then merges them using the Borda Merging method, obtaining the new ranked list. Finally, the new list replace the two merged lists. This process restarts until only one list remains.

For convenience, users can obtain a ranked list for each state by the function *aggregate* directly, which uses Sprearman, BordaMerging, and Kruskal algorithms to aggregate the ranked lists obtained with the same biological state by calling *FootruleMatrix*, *BMRankMerging* and *krubor* functions. The functions used in aggregation is described as below:

1) Function *FootruleMatrix* computes the pairwise distances between two ranked lists, *PRLs* is composed of ten samples (preprocessed gene expression profiles), so the result should be a 10-10 matrix, m-n element in the matrix represents the distance between mth and nth sample.

```
> SMDM <- FootruleMatrix(PRLs,1)
> SMDM
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
 [1,] 0.0000000 0.7266919 0.6306522 0.6255262 0.9197052 0.9039212 0.8971226
 [2,] 0.7266919 0.0000000 0.6934784 0.8170153 0.8759146 0.9999552 1.0000000
 [3,] 0.6306522 0.6934784 0.0000000 0.6512889 0.8690472 0.8932847 0.9267207
 [4,] 0.6255262 0.8170153 0.6512889 0.0000000 0.9109997 0.8316454 0.8765561
 [5,] 0.9197052 0.8759146 0.8690472 0.9109997 0.0000000 0.5796371 0.6239008
 [6,] 0.9039212 0.9999552 0.8932847 0.8316454 0.5796371 0.0000000 0.5718923
 [7,] 0.8971226 1.0000000 0.9267207 0.8765561 0.6239008 0.5718923 0.0000000
 [8,] 0.8862017 0.8496384 0.9141791 0.9425495 0.6039091 0.6601715 0.6405672
 [9,] 0.9184146 0.9137780 0.9091172 0.9138256 0.8655967 0.8985621 0.8868729
[10,] 0.8955524 0.8032911 0.8746648 0.9404120 0.8752885 0.9364788 0.9219843
          [,8]      [,9]     [,10]
 [1,] 0.8862017 0.9184146 0.8955524
 [2,] 0.8496384 0.9137780 0.8032911
 [3,] 0.9141791 0.9091172 0.8746648
```

```
 [4,] 0.9425495 0.9138256 0.9404120
 [5,] 0.6039091 0.8655967 0.8752885
 [6,] 0.6601715 0.8985621 0.9364788
 [7,] 0.6405672 0.8868729 0.9219843
 [8,] 0.0000000 0.9125154 0.8757162
 [9,] 0.9125154 0.0000000 0.6226474
[10,] 0.8757162 0.6226474 0.0000000
```

2) Function *BMRankMerging* merges the two or more ranked lists, the order of merging is determined by the results of the FootruleMatrix.

```
> outranking=BMRankMerging(PRLs[,c(1,2)])
```

3) Function *krubor* aggregates all the input lists corresponding to the same biological state into one list.

```
> PRL <- krubor(PRLs[,c(1:3)])
```

4) Function *aggregate* aggregates the ranked lists treated with the same compounds to a new ranked list.

```
> aggregate(exampleSet)
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 22283 features, 6 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 1 2 ... 6 (6 total)
  varLabels: phenodata
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

3.2 Similarity Measuring

Once a list had been obtained by aggregating ranked lists treated with the same biological state, we adopted a nonparametric, rank-based method called Gene Set Enrichment Analysis (GSEA)[Subramanian et al., 2005] to compute the pairwise distances between biological states.

```
> PRL <- aggregate(exampleSet)
> d  <- distances(PRL,250)
> ES <- d[[1]]
> exprs(ES)
```

```
             1           2          3         4          5           6
1  0.000000000  0.585185494  0.1213122 0.1634067 -0.1544331 -0.007530704
2  0.532347116  1.000000000 -0.1205890 0.1961283  0.1856957  0.186878500
3  0.223519811 -0.174526029  1.0000000 0.4715567  0.4627624  0.407694458
4 -0.009746562 -0.004706486  0.3415942 1.0000000  0.6527843  0.612994508
5  0.009272909  0.148695230  0.3452670 0.6665296  1.0000000  0.648401943
6 -0.006462397 -0.016785277  0.3229811 0.5752355  0.5923120  1.000000000
```

```
> distance <- d[[2]]
> exprs(distance)
```

```
          1         2         3         4         5         6
1 1.0000000 0.4412337 0.8275840 0.9231699 1.0725801 1.0069966
2 0.4412337 0.0000000 1.1475575 0.9042891 0.8328045 0.9149534
3 0.8275840 1.1475575 0.0000000 0.5934245 0.5959853 0.6346622
4 0.9231699 0.9042891 0.5934245 0.0000000 0.3403431 0.4058850
5 1.0725801 0.8328045 0.5959853 0.3403431 0.0000000 0.3796430
6 1.0069966 0.9149534 0.6346622 0.4058850 0.3796430 0.0000000
```

Another function *integratePRL* is designed for adding new ranked list into existing data set without recalculation. The previous ES matrix and distance matrix are used as arguments.

```
> newPRL <- PRL[,2]
> d <- integratePRL(ES,PRL,newPRL,250)
> newES <- d[[2]]
> newdistance <- d[[3]]
> exprs(newES)
```

```
              1           2          3         4          5           6
1   0.000000000  0.585185494  0.1213122 0.1634067 -0.1544331 -0.007530704
2   0.532347116  1.000000000 -0.1205890 0.1961283  0.1856957  0.186878500
3   0.223519811 -0.174526029  1.0000000 0.4715567  0.4627624  0.407694458
4  -0.009746562 -0.004706486  0.3415942 1.0000000  0.6527843  0.612994508
5   0.009272909  0.148695230  0.3452670 0.6665296  1.0000000  0.648401943
6  -0.006462397 -0.016785277  0.3229811 0.5752355  0.5923120  1.000000000
21  0.532347116  1.000000000 -0.1205890 0.1961283  0.1856957  0.186878500
             21
1   0.585185494
2   1.000000000
3  -0.174526029
4  -0.004706486
5   0.148695230
6  -0.016785277
21  1.000000000
```

```
> exprs(newdistance)
```

```
           1         2         3         4         5         6         21
1  1.0000000 0.4412337 0.8275840 0.9231699 1.0725801 1.0069966 0.4412337
2  0.4412337 0.0000000 1.1475575 0.9042891 0.8328045 0.9149534 0.0000000
3  0.8275840 1.1475575 0.0000000 0.5934245 0.5959853 0.6346622 1.1475575
4  0.9231699 0.9042891 0.5934245 0.0000000 0.3403431 0.4058850 0.9042891
5  1.0725801 0.8328045 0.5959853 0.3403431 0.0000000 0.3796430 0.8328045
6  1.0069966 0.9149534 0.6346622 0.4058850 0.3796430 0.0000000 0.9149534
21 0.4412337 0.0000000 1.1475575 0.9042891 0.8328045 0.9149534 0.0000000
```

Session Information

The version number of R and packages loaded for generating the vignette were:

```
R version 2.15.0 (2012-03-30)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] GeneExpressionSignature_1.2.0 Biobase_2.16.0
[3] BiocGenerics_0.2.0

loaded via a namespace (and not attached):
[1] tools_2.15.0
```

References

T.T. Cormen, C.E. Leiserson, and R.L. Rivest. Introduction to algorithms. 1990.

G. Hu and P. Agarwal. Human disease-drug network based on genomic expression profiles. *PLoS One*, 4(8):e6536, 2009.

F. Iorio, R. Bosotti, E. Scacheri, V. Belcastro, P. Mithbaokar, R. Ferriero, L. Murino, R. Tagliaferri, N. Brunetti-Pierri, A. Isacchi, et al. Discovery of drug mode of action and drug repositioning from transcriptional responses. *Proceedings of the National Academy of Sciences*, 107(33):14621, 2010.

J. Lamb, E.D. Crawford, D. Peck, J.W. Modell, I.C. Blat, M.J. Wrobel, J. Lerner, J.P. Brunet, A. Subramanian, K.N. Ross, et al. The connectivity map: using gene-expression signatures to connect small molecules, genes, and disease. *science*, 313(5795):1929, 2006.

A. Subramanian, P. Tamayo, V.K. Mootha, S. Mukherjee, B.L. Ebert, M.A. Gillette, A. Paulovich, S.L. Pomeroy, T.R. Golub, E.S. Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15545, 2005.