GeneExpressionSignature: Computing pairwise distances between different biological states

Yang Cao, Lu Han, Fei Li, Xiaochen Bo

October 17, 2016

Contents

1 Introduction

The *GeneExpressionSignature* package utilizes gene expression profiles to measure the similarity between different biological states. It provides two algorithms for similarity measurement: the GSEA algorithm which is mentioned in (**?**) and the PGSEA algorithm in PGSEA R package. A further description of the measurement methods based on gene expression signature can be found in Lamb(**?**), Hu(**?**) and Iorio(**?**).

This manual is a brief introduction to structure, functions and usage of GeneExpressionSignature package. It shows how the biological similarity is determined through a series of calculation steps and how that information can be used for further cluster analysis.

The current version of GeneExpressionSignature can be used only with data coming from the same platform, examples are on the HG-U133A platform.

2 Getting Started

A complete analysis procedure accepts a set of gene expression profiles representing different biological states as input, and generates a similarity matrix as output. It can be divided into three steps: 1)data ranking, 2)rank merging, and 3)similarity measuring.
First, we load the package by entering the following command in your R session:

```
> library(GeneExpressionSignature)
```

2.1 Data Ranking

Gene expression profiles should be properly preprocessed before analysis as prerequisite, including background correction, normalization and summarization. Instead of the exact values, ranks of gene expression levels are used in the following procedure. A ranked list of genes was obtained first by sorting the microarray probe-set identifiers according to the different expression values (count or ratio). It should be noticed that there is no standard methods for data preprocessing, and there is a function *getRLs* which takes the method in C-MAP for data preprocessing just for reference. We can obtain ranked lists matrix by calling *getRLs*.

Your experimental data could be used for analysing, or users can download gene-expression profiles from the GEO database with R package *GEOquery*. Users can see the doc in the package GEOquery for more details.

As an example, we download data from GEO database with package GEOquery. Then combined the treatment expression values to form a treatment matrix as well as the control expression values.

```
> # If you have network access
> #GSM118720 <- getGEO('GSM118720')
> # GSM118721 <- getGEO('GSM118721')
> if (require(GEOquery)){
+   #treatment gene-expression profiles
+   GSM118720 <- getGEO(filename=system.file("extdata/GSM118720.soft",package=
+   "GeneExpressionSignature"))
+   #control gene-expression profiles
+   GSM118721 <- getGEO(filename=system.file("extdata/GSM118721.soft",package=
+   "GeneExpressionSignature"))
+   #data ranking according to the different expression values
+   control <- as.matrix(as.numeric(Table(GSM118721)[,2]))
+   treatment <- as.matrix(as.numeric(Table(GSM118720)[,2]))
+   ranked_list <-getRLs(control,treatment)}
```

2.2 Rank Merging

By rank merging, multiple ranked lists are merging into a single ranked list, referred as prototype ranked list (PRL), representing certain kind of biological state. This procedure is mainly performed before similarity measuring, and applied to specific situations that occur when multiple ranked list are assigned to one single biological state with different cell types or experimental condition.

However, two different cases should be considered: 1) all ranked list with the same biological state are treated eaually important; 2) each individual ranked lists has its own ranked weights. This package provides two commonly employed algorithms: one utilizes the Kruskal algorithm proposed by (**?**) for the former case and another takes the average ranking technique a simple but ranther useful method. Function *RankMering* is provided for aggregating the ranked lists into one or many PRLs according their phenotypic data. All the things that we need to do is construct a ExpressionSet object as input, with ranked lists as assay data and corresponding biological states as phenotypic data.

For convenience, ranking data stored as ExpressionSet class in 'eset' object as input data, with ranked lists (obtained by calling *getRLs*) as assay data and corresponding biological states as phenotypic data. As an example, we start from loading cultured the exampleSet data, a subset of C-MAP Lamb(**?**) as sample data, which is a large reference catalogue of gene expression data from cultured human cells perturbed with many chemicals and genetic reagents. The sub dataset is composed of 50 paired gene expression profiles involving 22283 genes. This profiles are obtained from cells treated 15 compounds respectively, the values of which already converted to rank orders.

```
> data(exampleSet)
> show(exampleSet)
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 22283 features, 50 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 1 2 ... 50 (50 total)
  varLabels: state
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

```
> exprs(exampleSet)[c(1:10),c(1:3)]
```

```
       1     2     3
1  11264 14408 13919
2  12746 12365  3080
3   8267  5630 13060
4   2193 16694 16084
5   9556  6044  8294
6    279  5120  4826
7  15381 10225 10883
8   9452 10777 13359
9   6149  6213  6800
10  4943 12760  3444
```

```
> levels(as(phenoData(exampleSet),"data.frame")[,1])
```

```
 [1] "alsterpaullone" "azacitidine"    "camptothecin"   "chrysin"
 [5] "daunorubicin"   "doxorubicin"    "ellipticine"    "etacrynic_acid"
 [9] "fisetin"        "harmine"        "luteolin"       "mitoxantrone"
[13] "parthenolide"   "staurosporine"  "thiostrepton"
```

Rank merging process will generate a mergingSet of 15 PRLs from 50 paired expression profiles with each PRL corresponding one of 15 compounds respectively.

```
> MergingSet <- RankMerging(exampleSet,"Spearman",weighted=TRUE)
> show(MergingSet)
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 22283 features, 15 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: alsterpaullone azacitidine ... thiostrepton (15 total)
  varLabels: state
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

2.3 Similarity Measuring

One single combined PRL for a state was obtained after rank merging procedure. These PRLs are used to measure the similarity of the gene signature across different biological states by scoring functions *ScoreGSEA* and *ScorePGSEA*. Not all the genes are involved in similarity measuring, but only a subset of genes called gene signature whose combined expression pattern is uniquely characteristic of the biological state. Generally the genes used as gene signatures in the similarity scoring procedure are predefined by priori knowledge. Iorio(**?**) proposed an "optimal signature" approach by taking the most up-regulated genes and the most down-regulated genes as gene signature.The size of gene signatures need to be considered, which is taken as another parameter besides the PRLs in similarity measuring. In most cases, the default size of gene signature is 250 for genome-wide expression profile.

Suppose N is the number of PRLs (also same as the number of biological states), an N x N distance matrix is generated by similarity measurement. For mergingSet, we will get a 15 x 15 matrix corresponding to the similarity distances between these compounds.

```
> ds <- ScoreGSEA(MergingSet,250,"avg")
> ds[1:5,1:5]
```

```
               alsterpaullone azacitidine camptothecin   chrysin daunorubicin
alsterpaullone      0.0000000   0.6176992    0.4669311 0.6896005    0.5288110
azacitidine         0.6176992   0.0000000    0.6125031 0.8515960    0.6413233
camptothecin        0.4669311   0.6125031    0.0000000 0.7897938    0.5372661
chrysin             0.6896005   0.8515960    0.7897938 0.0000000    0.7443612
daunorubicin        0.5288110   0.6413233    0.5372661 0.7443612    0.0000000
```

2.4 Signature Distance

As we mentioned above, four algorithms implemented as functions *getRLs*, *RankMerging*, *ScoreGSEA* and *ScorePGSEA*, one is for data preprocessing, one called Iorio algorithm is for rank merging, the other two algorithms called GSEA and PGSEA are for similarity measuring. Moreover, function *SignatureDistance* is provided to serve as a single entry and easy access point to rank merging and similarity measuring, which runs through the including rank merging and scoring, and is recommended to use in most cases. Data ranking is not integration into this funciton for no standard methods for data preprocessing and gene-expression data types is uncertain. Furthermore, there is no effective method to integrate data from different platforms. Function *getRLs* which takes the method in C-MAP for data preprocessing just for reference.

```
> SignatureDistance(exampleSet,SignatureLength=250,MergingDistance="Spearman",ScoringMetho
```

```
               alsterpaullone azacitidine camptothecin   chrysin daunorubicin
alsterpaullone      0.0000000   0.6176992    0.4669311 0.6896005    0.5288110
azacitidine         0.6176992   0.0000000    0.6125031 0.8515960    0.6413233
camptothecin        0.4669311   0.6125031    0.0000000 0.7897938    0.5372661
chrysin             0.6896005   0.8515960    0.7897938 0.0000000    0.7443612
daunorubicin        0.5288110   0.6413233    0.5372661 0.7443612    0.0000000
```

```
doxorubicin          0.4449537   0.6223770    0.5590938 0.8152383    0.4805674
ellipticine          0.6147176   0.6958627    0.6060621 0.8399921    0.6013995
etacrynic_acid       0.9546259   0.9625380    0.9150898 0.8846840    0.9174653
fisetin              0.6191321   0.7401457    0.7258576 0.9056164    0.7204894
harmine              0.7381854   0.9011707    0.8082408 0.6264392    0.7486181
luteolin             0.6601723   0.8357249    0.6559045 0.4627429    0.6882700
mitoxantrone         0.5351687   0.6326825    0.6586904 0.8367069    0.5450045
parthenolide         0.9183664   0.8581793    0.8943531 0.8865647    0.8487120
staurosporine        0.6984201   0.6982204    0.7120952 0.9037265    0.7625792
thiostrepton         0.9258501   0.8624173    0.8523135 0.9235488    0.8449146
               doxorubicin ellipticine etacrynic_acid   fisetin   harmine
alsterpaullone   0.4449537   0.6147176      0.9546259 0.6191321 0.7381854
azacitidine      0.6223770   0.6958627      0.9625380 0.7401457 0.9011707
camptothecin     0.5590938   0.6060621      0.9150898 0.7258576 0.8082408
chrysin          0.8152383   0.8399921      0.8846840 0.9056164 0.6264392
daunorubicin     0.4805674   0.6013995      0.9174653 0.7204894 0.7486181
doxorubicin      0.0000000   0.5737439      0.9590589 0.6130763 0.8146797
ellipticine      0.5737439   0.0000000      0.9130729 0.8065379 0.6967980
etacrynic_acid   0.9590589   0.9130729      0.0000000 0.9558315 0.9745611
fisetin          0.6130763   0.8065379      0.9558315 0.0000000 0.9077328
harmine          0.8146797   0.6967980      0.9745611 0.9077328 0.0000000
luteolin         0.7326149   0.7415050      0.8413584 0.8872799 0.5954765
mitoxantrone     0.4266442   0.6073227      0.9880406 0.7107602 0.8455741
parthenolide     0.9058101   0.8260994      0.6586242 0.9955009 0.9361217
staurosporine    0.6729108   0.7785568      0.9677592 0.7667645 0.9525701
thiostrepton     0.8801426   0.8887309      0.7367122 0.9806517 0.9818760
                luteolin mitoxantrone parthenolide staurosporine thiostrepton
alsterpaullone 0.6601723    0.5351687    0.9183664     0.6984201    0.9258501
azacitidine    0.8357249    0.6326825    0.8581793     0.6982204    0.8624173
camptothecin   0.6559045    0.6586904    0.8943531     0.7120952    0.8523135
chrysin        0.4627429    0.8367069    0.8865647     0.9037265    0.9235488
daunorubicin   0.6882700    0.5450045    0.8487120     0.7625792    0.8449146
doxorubicin    0.7326149    0.4266442    0.9058101     0.6729108    0.8801426
ellipticine    0.7415050    0.6073227    0.8260994     0.7785568    0.8887309
etacrynic_acid 0.8413584    0.9880406    0.6586242     0.9677592    0.7367122
fisetin        0.8872799    0.7107602    0.9955009     0.7667645    0.9806517
harmine        0.5954765    0.8455741    0.9361217     0.9525701    0.9818760
luteolin       0.0000000    0.7964905    0.8139365     0.9017486    0.8642047
mitoxantrone   0.7964905    0.0000000    0.9181195     0.7469812    0.8984673
parthenolide   0.8139365    0.9181195    0.0000000     0.9864220    0.6180173
staurosporine  0.9017486    0.7469812    0.9864220     0.0000000    0.9521839
thiostrepton   0.8642047    0.8984673    0.6180173     0.9521839    0.0000000
```

```
> ds[1:5,1:5]
```

```
               alsterpaullone azacitidine camptothecin   chrysin daunorubicin
alsterpaullone      0.0000000   0.6176992    0.4669311 0.6896005    0.5288110
azacitidine         0.6176992   0.0000000    0.6125031 0.8515960    0.6413233
camptothecin        0.4669311   0.6125031    0.0000000 0.7897938    0.5372661
chrysin             0.6896005   0.8515960    0.7897938 0.0000000    0.7443612
daunorubicin        0.5288110   0.6413233    0.5372661 0.7443612    0.0000000
```

3 Implementation Details

3.1 Adaptively Weighted Rank Merging

The Iorio's rank merging algorithm utilizes Kruskal algorithm (**?**) to merge the ranked lists which corresponding to a same biological state. The distance of these ranked lists must be calculated first, a measure of the distance between two ranked lists is computed using *Spearman* algorithm or *Kendall tau* algorithm. It should be noticed that is rank merging with Kendall tau distance is time consuming, so we recommend selecting the Spearman distance. Next, merge the two or more ranked lists with the same biological state using *Borda merging* algorithm.

According to the *Kruskal* algorithm method (**?**), this rank merging 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 a new ranked list. Finally, the new list replaces the two unmerged lists. This process won't terminate until only one list remains.

For convenience, users can directly obtain a PRL for each state by the function *Iorio.RankMerging*, which uses Sprearman, BordaMerging, and Kruskal algorithms to aggregate the ranked lists obtained with the same biological state. For instance, we will merge the sample data which with 50 samples into 15 samples.

```
> MergingSet <- RankMerging(exampleSet,"Spearman",weighted=TRUE)
> show(MergingSet)

ExpressionSet (storageMode: lockedEnvironment)
assayData: 22283 features, 15 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: alsterpaullone azacitidine ... thiostrepton (15 total)
  varLabels: state
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

3.2 Eqully Weighted Rank Merging

A simple but rather useful method for this problem is the average ranking technique. The technique is a two step process when we are under the assumption that importance is equally weighted for each ranked list. First step is to calculate average rank for each ranked list and then the second step is to construct their final rankings.

3.3 Similarity Measuring

Once ranked lists with same biological states are merged to one single PRL, Gene Set Enrichment Analysis (GSEA) and Parametric Gene Set Enrichment Analysis (PGSEA) are adopted to measure the similarity among these PRLs.

GSEA algorithm(?) is a nonparametric, rank-based method for similarity measuring to determine whether a priori defined set of genes shows statistically significant, concordant differences between two biological states. PGSEA algorithm is a modified gene set enrichment analysis method based on a parametric statistical analysis model, and we use the functions in R package PGSEA for similarity measuring. Both of these two functions gives the corresponding p value, function ScoreGSEA calcutes the empirical p values from Monte Carlo Procedures(?).

```
> ds <- ScoreGSEA(MergingSet,250,"avg")
> ds[1:5,1:5]
```

```
               alsterpaullone azacitidine camptothecin   chrysin daunorubicin
alsterpaullone      0.0000000   0.6176992    0.4669311 0.6896005    0.5288110
azacitidine         0.6176992   0.0000000    0.6125031 0.8515960    0.6413233
camptothecin        0.4669311   0.6125031    0.0000000 0.7897938    0.5372661
chrysin             0.6896005   0.8515960    0.7897938 0.0000000    0.7443612
daunorubicin        0.5288110   0.6413233    0.5372661 0.7443612    0.0000000
```

```
> ds <- ScorePGSEA(MergingSet,250,"avg")
> ds[1:5,1:5]
```

```
               alsterpaullone azacitidine camptothecin   chrysin daunorubicin
alsterpaullone      0.0000000   0.5477505    0.4136914 0.6340643    0.4182223
azacitidine         0.5477505   0.0000000    0.5646674 0.8402056    0.5478599
camptothecin        0.4136914   0.5646674    0.0000000 0.7438953    0.4305080
chrysin             0.6340643   0.8402056    0.7438953 0.0000000    0.7067854
daunorubicin        0.4182223   0.5478599    0.4305080 0.7067854    0.0000000
```

4 Futher Analysis

To illustrate how to use GeneExpressionSignature in analysis of gene expression signatures, affinity propagation clustering can be used to group these biological states by the similarity of gene signature. Affinity propagation cluster algorithm iteratively searches for optimal clustering by maximizing an objective function called net similarity. Here, we use function in R apcluster package to classify the 15 biological states into 3 groups. In this step, R package apcluster should also be installed on your computer.

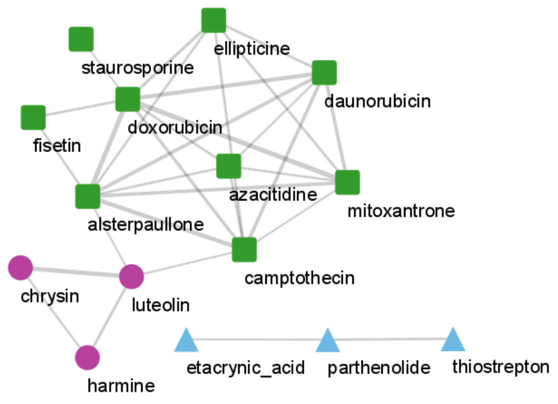
```
> if (require(apcluster)){
+   library(apcluster)
+   clusterResult <- apcluster(1-ds)
+   show(clusterResult)
+ }
```

```
APResult object

Number of samples     =  15
Number of iterations  =  122
Input preference      =  0.2561047
Sum of similarities   =  6.432731
Sum of preferences    =  0.768314
Net similarity        =  7.201045
Number of clusters    =  3

Exemplars:
   doxorubicin luteolin parthenolide
Clusters:
   Cluster 1, exemplar doxorubicin:
      alsterpaullone azacitidine camptothecin daunorubicin doxorubicin
      ellipticine fisetin mitoxantrone staurosporine
   Cluster 2, exemplar luteolin:
      chrysin harmine luteolin
   Cluster 3, exemplar parthenolide:
      etacrynic_acid parthenolide thiostrepton
```

Cytoscape is used to visualize the result of clustering. In the network, nodes denotes different compounds (cell states treated with different compounds), and the edge means the similarity distance between these two compounds is lower than a threshold, which is 0.68 here. Different colors denote different groups, as the classification of compounds. We note that the largest group is numbered 9 nodes, and the other two consist of 3 nodes for each group.



Session Information

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

```
R version 3.3.1 (2016-06-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] apcluster_1.4.3                GEOquery_2.40.0
 [3] GeneExpressionSignature_1.20.0 PGSEA_1.48.0
 [5] annaffy_1.46.0                 KEGG.db_3.2.3
 [7] GO.db_3.4.0                    AnnotationDbi_1.36.0
 [9] IRanges_2.8.0                  S4Vectors_0.12.0
[11] Biobase_2.34.0                 BiocGenerics_0.20.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.7     lattice_0.20-34 XML_3.98-1.4    bitops_1.0-6
 [5] grid_3.3.1      R6_2.2.0        DBI_0.5-1       RSQLite_1.0.0
 [9] httr_1.2.1      Matrix_1.2-7.1  tools_3.3.1     RCurl_1.95-4.8
```