RegSeq Tutorial

Duanchen Sun, Xiaopeng Huang

2018-09-12

Contents

1 Introduction

RegSeq package contains the proposed RegSeq algorithm (function `Reg_Finding`), which is designed to perform the regulon analysis for RNA-Seq data, especially in small sample scenario. RegSeq can identify the key regulons that are most responsible for the difference in two groups samples. In this tutorial, we use several examples to help users executing RegSeq in real applications.

We first load the required package.

```
library("RegSeq")
options(scipen = 0)
```

2 RegSeq implementation example

2.1 Input data preparation

There are two necessary input data sources of RegSeq. The first one is a gene expression signature and the second one is a regulon network. The gene expression signature is a gene-wise vector that each gene has a score, which represents distinctive phenotypes or treatments of comparing two groups of samples. Actually, any computational method that generates a quantitative measurement of difference between the groups is applicable for obtaining this vector. Here, we use the function `get_signature` to compute the expression signature.

We load the processed esophageal carcinoma (ESCA) raw counts data from TCGA.

```
load('ESCA_expression.RData')
```

In this expression matrix, each row is a gene and each column is a sample. The dimensions of ESCA dataset are

```
dim(exprs)
```

```
## [1] 14788    16
```

which indicate there are totally 14788 genes and 16 samples. Notably, this data only keep the paired samples that diagnosed with 'normal' and 'tumor'. The detailed information of these samples are:

```
colnames(exprs)
```

```
##  [1] "TCGA-L5-A4OG-01A-11R-A260-31" "TCGA-IC-A6RE-01A-11R-A336-31"
##  [3] "TCGA-L5-A4OJ-01A-11R-A260-31" "TCGA-V5-AASX-01A-11R-A38D-31"
##  [5] "TCGA-L5-A4OO-01A-11R-A260-31" "TCGA-L5-A43C-01A-11R-A24K-31"
##  [7] "TCGA-IC-A6RF-01A-13R-A336-31" "TCGA-V5-A7RE-01A-11R-A354-31"
##  [9] "TCGA-L5-A4OG-11A-12R-A260-31" "TCGA-IC-A6RE-11A-12R-A336-31"
## [11] "TCGA-L5-A4OJ-11A-12R-A260-31" "TCGA-V5-AASX-11A-11R-A38D-31"
## [13] "TCGA-L5-A4OO-11A-12R-A260-31" "TCGA-L5-A43C-11A-11R-A24K-31"
## [15] "TCGA-IC-A6RF-11A-21R-A336-31" "TCGA-V5-A7RE-11A-11R-A354-31"
```

which show that the first 8 samples are normal samples and the latter 8 samples are matched tumor samples.

Given an above RNA-Seq raw counts data with different sample labels, we can obtain the gene signature by using the compiled function `get_signature`. We set the following function parameters:

```
p <- ncol(exprs)/2
rawCounts <- exprs
groupA <- 1:p
groupB <- (p+1):(2*p)
```

Then, the signature can be got by:

```
signature <- get_signature(rawCounts, groupA, groupB)
```

```
## converting counts to integer mode
```

The scores of first 5 genes are:

```
signature[1:5]
```

```
##    TSPAN6      DPM1     SCYL3  C1orf112       FGR
## -1.420484  1.312739 -1.786783  3.508417  1.611835
```

The positive score represents highly expressed in `groupA` and negative score represents highly expressed in `groupB`, respectively. `get_signature` directly uses the differential expression analysis results in DESeq2 package. Again, users can perform any method (e.g. fold change, Student's t-test, Mann-Whitney U test) to obtain the signature and directly use the derived signature as model input.

The second part of model input is a regulon network. `Reg_Finding` use the list object of class `regulon`, where each element represent a transcriptional regulator (transcription factor) and contains two vectors: (1) a named numeric vector indicating the mode of regulation for each target gene, whose ID is indicated by the names attribute of the vector. (2) a numeric vector indicating the confidence score for the TF-target interaction. The regulon network can be generated from networks reverse engineered with the ARACNe algorithm. We now load the regulon network built from ESCA dataset.

```
load('ESCA_regulon.RData')
```

This regulon network contains the following number of transcription factors:

```
length(regulon)
```

```
## [1] 5951
```

2.2 Perform regulon analysis via RegSeq

The regulon analysis can be achieved by using the folloing code:

```
set.seed(11)
result <- Reg_Finding(signature, regulon, Permutation = 10, sort = TRUE)
```

```
## [1] "STEP1: Regulon preprocessing start..."
## [1] "STEP2: Run absolute gene permuting GSEA..."
## [1] 69
## [1] "STEP3: Run reverse positive/negative target gene permuting GSEA...."
## [1] "Algorithm finished!!!"
```

In the implementation trial, the seed can obtain a reproducible result. The algorithm can automatically print out the current process. The final output is saved in a `data.frame` table with the following elements:

```
head(result)
```

```
##          Positive Negative Target Reg_Score      FDR Direction
## TIMELESS      111       16    127  4.291371 0.000000 ACTIVATED
## MYBL2         126       35    161  4.175523 0.001500 ACTIVATED
## HELLS         102       38    140  3.715183 0.001750 ACTIVATED
## ASF1B          98       50    148  3.785127 0.002000 ACTIVATED
## SPAG5          32        5     37  3.313683 0.002250 ACTIVATED
## UBE2C          67       21     88  3.588315 0.002571 ACTIVATED
```

In this result, each row is a transcription factor and the rows are sorted in `FDR` ascending order, due to the parameter `sort = TRUE`. The column names `Positive` and `Negative` are the number of positive and negative target genes of current transcription factor, respectively. `Target` is the total number of target genes. `Reg_Score` is the regulon score computed by RegSeq and `FDR` represents the FDR value of absolute-based GSEA. The final condition of regulon is shown in `Direction`, which is a indicator to show whether the significant regulon is being activated (`ACTIVATED`) or repressed (`REPRESSED`).

Besides, the users can set other parameters to meet different needs. For example, we want find the significant regulon (FDR cutoff < 0.05) with size range from 100 to 300 and save the final result to a file named 'result.txt'. This request can be achieved by using the following code:

```
set.seed(123)
result <- Reg_Finding(signature, regulon, 100, 300, cutoff = 0.05, outfile = 'result.txt')
```

```
## [1] "STEP1: Regulon preprocessing start..."
## [1] "STEP2: Run absolute gene permuting GSEA..."
## [1] 144
## [1] "STEP3: Run reverse positive/negative target gene permuting GSEA...."
## [1] "Algorithm finished!!!"
```

```
head(result)
```

```
##       Positive Negative Target Reg_Score FDR Direction
## FOXN3       40       61    101 -3.316262   0 REPRESSED
## E2F7        79       37    116  2.982476   0 ACTIVATED
## FOXM1      176       52    228  3.809022   0 ACTIVATED
## ZHX3        72      107    179 -3.474108   0 REPRESSED
## AFF4        65       95    160 -3.521238   0 REPRESSED
## ATAD2      110       52    162  3.630932   0 ACTIVATED
```

3 Reference

Duanchen Sun and Zheng Xia (2018): RegSeq: A novel GSEA-based algorithm to perform the regulon analysis on RNA-Seq data with small sample replicates.