1 Introduction

This article introduces usage of the LMGene package. LMGene has been developed for analysis of microarray data using a linear model and glog data transformation in the R statistical package.

2 Data preparation

LMGene takes objects of class ExpressionSet, which is the standard data structure of the Biobase package. Therefore, the user who already has data that is of class ExpressionSet can jump to further steps, such as g-log transformation or looking for differentially expressed genes. Otherwise, the user needs to generate new objects of class ExpressionSet. For more detail, please see the vignette, ‘An Introduction to Biobase and ExpressionSets’ in the Biobase package.

**Note: ExpressionSet.** In this package, an object of class ExpressionSet must produce proper data using the commands exprs(object) and phenoData(object).

**Example.** LMGene includes sample array data which is of class ExpressionSet. Let’s take a look this sample data.

1. First, load the necessary packages in your R session.

   ```r
   > library(LMGene)
   > library(Biobase)
   > library(tools)
   ```

2. Load the sample ExpressionSet class data in the package LMGene.

   ```r
   > data(sample.eS)
   ```
3. View the data structure of the sample data and the details of `exprs` and `phenoData` slots in the data.

```r
> slotNames(sample.eS)

[1] "experimentData"  "assayData"      "phenoData"
[4] "featureData"     "annotation"     "protocolData"
[7] ".__classVersion__" 

> dim(exprs(sample.eS))

[1] 613 32

> exprs(sample.eS)[1:3,]

     p1d0  p1d1  p1d2  p1d3  p2d0  p2d1  p2d2  p2d3  p3d0  p3d1  p3d2  p3d3  p4d0  p4d1  p4d2
p1d0 216 149 169 113 193 172 167 168 151 179 142 156 156 160 214
p1d1 334 311 187 135 514 471 219 394 438 438 329 427 429 574 419
p1d2 398 367 351 239 712 523 356 629 474 438 532 427 429 574 419

     p4d3  p5d0  p5d1  p5d2  p5d3  p6d0  p6d1  p6d2  p6d3  p7d0  p7d1  p7d2  p7d3  p8d0  p8d1
p4d3 195 165 144 185 162 246 227 173 151 796 378 177 278 183 285
p5d0 450 293 285 390 428 645 631 324 343 852 451 259 379 259 386
p5d1 564 438 321 519 488 824 579 416 489 1046 501 375 388 373 509
p5d2 275 202
p5d3 361 333
p6d0 468 436

> phenoData(sample.eS)

An object of class 'AnnotatedDataFrame'
  sampleNames: p1d0 p1d1 ... p8d3 (32 total)
  varLabels: patient dose
  varMetadata: labelDescription

> slotNames(phenoData(sample.eS))

[1] "varMetadata"  "data"      "dimLabels"
[4] ".__classVersion__"

Data generation. If you don’t have `ExpressionSet` class data, you need to make some. `LMGene` provides a function that can generate an object of class `ExpressionSet`, assuming that there are array data of `matrix` class and experimental data of `list` class.

1. The package includes sample array and experimental/phenotype data, `sample.mat` and `vlist`.

   > data(sample.mat)
   > dim(sample.mat)
2. Generate ExpressionSet class data using `neweS` function.

```r
> test.eS <- neweS(sample.mat, vlist)
> class(test.eS)

[1] "ExpressionSet"
attr("package")
[1] "Biobase"
```

3 **G-log transformation**

1. **Estimating parameters for g-log transformation.** In LMGene, the linear model is not intended to be applied to the raw data, but to transformed and normalized data. Many people use a log transform. LMGene uses a log-like transform involving two parameters. We estimate the parameters \( \lambda \) and \( \alpha \) of the generalized log transform \( \log (y - \alpha + \sqrt{(y - \alpha)^2 + \lambda}) = \sinh^{-1}(\frac{y-\alpha}{\lambda}) + \log(\lambda) \) using the function `tranest` as follows:

```r
> tranpar <- tranest(sample.eS)
> tranpar

$lambda
[1] 726.6187

$alpha
[1] 56.02754
```

The optional parameter `ngen` controls how many genes are used in the estimation. The default is all of them (up to 100,000), but this option allows the use of less. A typical call using this parameter would be

```r
> tranpar <- tranest(sample.eS, 100)
> tranpar
```
In this case, 100 genes are chosen at random and used to estimate the transformation parameter. The function returns a list containing values for lambda and alpha.

2. **G-log transformation.** Using the obtained two parameters, the g-log transformed expression set can be calculated as follows.

```r
> trsample.eS <- transeS(sample.eS, tranpar$lambda, tranpar$alpha)
> exprs(sample.eS)[1:3,1:8]

  p1d0  p1d1  p1d2  p1d3  p2d0  p2d1  p2d2  p2d3
 g1  216   149   169   113  193  172  167  168
 g2  334   311   187   135  514  471  219  394
 g3  398   367   351   239  712  523  356  629

> exprs(trsample.eS)[1:3,1:8]

  p1d0  p1d1  p1d2  p1d3  p2d0  p2d1  p2d2  p2d3
 g1 6.851409 5.368760 5.538283 4.967871 5.710219 5.561440 5.522547 5.530446
 g2 6.368455 6.286422 5.669895 5.230755 6.848893 6.753307 5.868450 6.555376
 g3 6.566680 6.475542 6.425047 5.975234 7.199441 6.867794 6.441102 7.067093

3. Tranest options: multiple alpha, lowessnorm, model

Rather than using a single alpha for all samples, we can estimate a separate alpha for each sample. This allows for differences in chips, in sample concentration, or exposure conditions.

```r
> tranparmult <- tranest(sample.eS, mult=TRUE)
> tranparmult

$lambda
 [1] 689.2819

$alpha
 [1] 69.67146 37.02711 54.13904 69.35728 60.33270 60.75301 71.72965
 [8] 64.55506 58.63427 65.73625 48.40173 59.43778 76.34568 78.81046
[15] 82.20326 96.19938 77.60070 79.48089 73.63257 73.41650 33.86029
[22] 69.26448 55.75460 54.29840 139.89493 91.36521 46.46158 59.02056
[29] 73.60255 89.48728 57.13387 64.98866

For vector alphas, transeS uses exactly the same syntax:

```r
> trsample.eS <- transeS(sample.eS, tranparmult$lambda, tranparmult$alpha)
> exprs(trsample.eS)[1:3,1:8]
```
It’s also possible to estimate the parameters using the more accurate lowess normalization (as opposed to uniform normalization):

```r
> tranparmult <- tranest(sample.eS, ngenes=100, mult=TRUE, lowessnorm=TRUE)
> tranparmult

$lambda
[1] 446.9194

$alpha
[1] 82.33815 55.65733 55.65038 62.58256 63.39246 63.27436 78.50829
[8] 59.90486 52.14531 64.32567 64.18845 67.62663 68.78720 69.00023
[22] 90.79049 64.51033 59.03890 180.89441 108.24910 57.45316 71.89433
[29] 55.46476 85.35130 56.31470 60.94888
```

One may also specify a model other than the default no-interaction model. For example, if we think that the interaction of variables in vlist is important, we can add interaction to the model:

```r
> tranpar <- tranest(sample.eS, model='patient + dose + patient:dose')
> tranpar

$lambda
[1] 860.0836

$alpha
[1] 55.68625
```

The model is always specified in the same way as the right-hand side of an `lm` model. In the example above, we set the parameters to minimize the mean squared error for a regression of transformed gene expression against patient, log dose, and their interaction.

Be very careful of using interactions between factor variables. If you do not have enough replicates, you can easily overfit the data and have no degrees of freedom left for error. Naturally, it’s possible to use `mult`, `lowessnorm`, and `model` all together.

## 4 Finding differentially expressed genes

1. **Transformation and Normalization.** Before finding differentially expressed genes, the array data needs to be transformed and normalized.
> trsample.eS <- transeS (sample.eS, tranparmult$lambda, tranparmult$alpha)
> ntrsample.eS <- lnormeS (trsample.eS)

2. Finding differentially expressed genes The LMGene routine computes significant probes/genes by calculating gene-by-gene p-values for each factor in the model and adjusting for the specified false discovery rate (FDR). A typical call would be

> sigprobes <- LMGene(ntrsample.eS)

There is an optional argument, level, which is the FDR (default 5 percent). A call using this optional parameter would look like

> sigprobes <- LMGene(ntrsample.eS,level=.01)

The result is a list whose components have the names of the effects in the model. The values are the significant genes for the test of that effect or else the message "No significant genes". As with tranest, it’s possible to specify a more complex model to LMGene:

> sigprobes <- LMGene(ntrsample.eS, model='patient+dose+patient:dose')
> sigprobes

$patient
[1] "g2" "g3" "g4" "g9" "g10" "g15" "g43" "g54" "g56" "g84"
[11] "g85" "g86" "g88" "g93" "g123" "g155" "g176" "g178" "g179" "g277"
[21] "g304" "g305" "g310" "g336" "g375" "g399" "g405" "g406" "g407" "g408"
[31] "g409" "g411" "g412" "g413" "g414" "g415" "g423" "g461" "g462" "g463"
[41] "g477" "g485" "g503" "g520" "g528" "g544" "g566" "g607" "g612"

$dose
[1] "No significant genes"

`patient:dose`
[1] "No significant genes"

References


