LPE test for microarray data with a small number of replicates

Nitin Jain <nitin.jain@pfizer.com>,
Michael O Connell <moconnell@insightful.com>,
and Jae K. Lee <jaeklee@virginia.edu>

October 17, 2016

Contents

1 Introduction

The *LPE* package describes local-pooled-error (LPE) test for identifying significant differentially expressed genes in microarray experiments. Local pooled error test is especially useful when the number of replicates is low (2-3) [**?**]. LPE estimation is based on pooling errors within genes and between replicate arrays for genes in which expression values are similar. This is motivated by the observation that errors between duplicates vary as a function of the average gene expression intensity and by the fact that many gene expression studies are implemented with a limited number of replicated arrays [**?**].

LPE library is primarily used for analyzing data between two conditions. To use it for paired data, see *LPEP* library. For using LPE in multiple conditions, use *HEM* library.

1.1 Mouse Immune Response Study dataset

Step by step analysis is presented in Section **??** using data from a 6-chip (Affymetrix munie GeneChip, MG-U74Av2) oligonucleotide microarray study of a mouse immune response study. Three replicates of Affymetrix oligonucleotide chips per condition (Naive and Activated) were used. Mouse immune response study was conducted by Dr. Klaus Ley, Univeristy of Virginia.

Details of methodology and application of Local Pooled Error (LPE) test can be obtained from the LPE paper, published in Bioinformatics [**?**], and detailed description of `Rank-invariant resampling based FDR method` can be obtained from FDR paper, published in BMC Bioinformatics [**?**].

2 Analyzing data set using LPE library

2.1 Check installed version of *LPE*

First, make sure that the version of LPE library you are using is at least 1.6.0

```
> packageDescription("LPE")
```

If you have an older version, download the latest version from Bioconductor.

2.2 Load the library

First, set the seed to 0 (to have reproducible results as shown below):

```
> set.seed(0)
```

Load the LPE library:

```
> library(LPE)
```

2.3 Load the data set

Load the data set 'Ley' (built in LPE package), check its dimensions and see the dataset. For illustration purposes, we will use only a small subset of Ley data (1000 rows) in the subsequent examples. Replicates of Naive condition are named as c1, c2, c3 and those of Activated condition are named as t1, t2 and t3 respectively.

```
> data(Ley)
> dim(Ley)

[1] 12488     7

> head(Ley)

                ID    c1    c2    c3     t1     t2     t3
1  AFFX-MurIL2_at  16.0  14.1  19.3 2782.7 2861.3 2540.2
2 AFFX-MurIL10_at  22.7   6.9  28.2   18.6   12.7    7.5
3  AFFX-MurIL4_at  33.9  17.1  23.9   24.9   25.2   24.9
4  AFFX-MurFAS_at 151.0 133.6 134.1  160.6  188.2  156.4
5  AFFX-BioB-5_at 246.9 182.9 273.6  178.0  184.8  146.6
6  AFFX-BioB-M_at 644.8 398.8 577.3  364.9  355.7  349.7

> Ley.subset <- Ley[seq(1000),]
```

2.4 Normalization of data

Do the pre-processing (or normalization) of the data. Since the data obtained here is in MAS5 format, we will use data.type=MAS5. (Note that LPE does not require users to normalize the gene-expression data using the `preprocess` function. Users can very well use other methods, such as `RMA` or any other method of their choice.)

The `preprocess` function does IQR normalization (so that inter-quartile ranges on all chips are set to their widest range), thresholding (making the intensity values lower than 1.0 to 1.0), log based 2 transformation and LOWESS normalization (if LOWESS is set to TRUE). Note that this preprocess is a simple constant-scale and location-normalization step.

```
> Ley.normalized <- Ley.subset
> Ley.normalized[,2:7] <- preprocess(Ley.subset[,2:7], data.type = "MAS5")
> Ley.normalized[1:3,]
```

```
               ID       c1       c2       c3        t1        t2        t3
1  AFFX-MurIL2_at 4.304733 4.076621 4.560498 11.442270 11.611246 11.385874
2 AFFX-MurIL10_at 4.809354 3.045594 5.107593  4.217231  3.795548  2.982039
3  AFFX-MurIL4_at 5.387947 4.354922 4.868908  4.638074  4.784143  4.713222
```

Remove the Affymetrix control spots (whose ID begins with 'AFFX'):

```
> Ley.final <- Ley.normalized[substring(Ley.normalized$ID,1,4) !="AFFX",]
> dim(Ley.final)
```

```
[1] 934   7
```

```
> Ley.final[1:3,]
```

```
           ID        c1        c2        c3        t1        t2        t3
67   92539_at 12.245451 12.410680 12.570798 12.048521 12.255754 11.981926
68 92540_f_at  9.194694  9.262375  8.920783 11.356672 11.504808 11.415221
69   92541_at  6.488617  6.337949  6.379552  5.153805  5.388064  5.982039
```

2.5 Obtain baseline error distribution

Calculate the baseline error distribution of Naive condition, which returns a data.frame of A vs M for selected number of bins (= 1/q), where q = quantile.

```
> var.Naive <- baseOlig.error(Ley.final[,2:4],q=0.01)
> dim(var.Naive)
```

```
[1] 934   2
```

```
> var.Naive[1:3,]
```

```
           A      var.M
67 12.410680 0.01618317
68  9.194694 0.07100961
69  6.379552 0.24239947
```

Similarly calculate the base-line distribution of Activated condition:

```
> var.Activated <- baseOlig.error(Ley.final[,5:7], q=0.01)
> dim(var.Activated)

[1] 934   2

> var.Activated[1:3,]

           A      var.M
67 12.048521 0.01405070
68 11.415221 0.01563467
69  5.388064 0.36761319
```

2.6 Calculate z-statistics for each gene

Calculate the lpe variance estimates as described above. The function `lpe` takes the first two arguments as the replicated data, next two arguments as the baseline distribution of the replicates calculated from the `baseOlig.error` function, and Gene IDs as probe.set.name.

```
> lpe.val <- data.frame(lpe(Ley.final[,5:7], Ley.final[,2:4],
+                           var.Activated, var.Naive,
+                           probe.set.name=Ley.final$ID)
+                       )
> lpe.val <- round(lpe.val, digits=2)
> dim (lpe.val)

[1] 934  13

> lpe.val[1:3,]

            x.t1  x.t2  x.t3 median.1 std.dev.1  y.c1  y.c2  y.c3 median.2
92539_at   12.05 12.26 11.98    12.05      0.12 12.25 12.41 12.57    12.41
92540_f_at 11.36 11.50 11.42    11.42      0.13  9.19  9.26  8.92     9.19
92541_at    5.15  5.39  5.98     5.39      0.61  6.49  6.34  6.38     6.38
           std.dev.2 median.diff pooled.std.dev z.stats
92539_at        0.13       -0.36           0.13   -2.88
92540_f_at      0.27        2.22           0.21   10.43
92541_at        0.49       -0.99           0.57   -1.75
```

2.7 FDR correction

Various FDR correction methods are supported in LPE: `BH` (Benjamini-Hochberg), `BY` (Benjamini-Yekutieli), `mix.all` (does FDR adjustment similar to that of SAM) or `resamp` (Rank invariant resampling based FDR correction - recommended method). For the sake of completion, there is an option "Bonferroni" to get Bonferroni adjusted p-values also. Note that BH and BY methods are adopted from `multtest` package.

```
> fdr.BH <- fdr.adjust(lpe.val, adjp="BH")
> dim(fdr.BH)

[1] 934   2

> round(fdr.BH[1:4, ],2)

           FDR z.real
93086_at     0  28.47
94224_s_at   0  20.46
92830_s_at   0  20.02
94063_at     0  19.03
```

Resampling based FDR adjustment takes a while to run, and returns the critical z-values and corresponding FDR. Users can decide from the table that which z.critical values to select (from the lpe results, here in lpe.val) to obtain the target fdr.

```
> fdr.resamp <- fdr.adjust(lpe.val, adjp="resamp", iterations=2)

iteration number 1 is in progress
iteration number 1 finished
iteration number 2 is in progress
iteration number 2 finished
Computing FDR...

> fdr.resamp

      target.fdr z.critical
 [1,]      0.001       3.58
 [2,]      0.010       2.92
 [3,]      0.020       2.37
 [4,]      0.030       2.10
 [5,]      0.040       1.97
 [6,]      0.050       1.77
 [7,]      0.060       1.68
 [8,]      0.070       1.61
 [9,]      0.080       1.53
```

```
[10,]      0.090       1.47
[11,]      0.100       1.40
[12,]      0.150       1.07
[13,]      0.200       0.87
[14,]      0.500       0.00
```

Note that above table may differ slightly due to generation of 'NULL distribution' by resampling. For each target.fdr, we can note critical z-value, above which all genes are considered significant. For example, in the above table, to obtain all the genes with FDR less than or equal to 5%, identify the `z.critical` corresponding to 5%, (=1.77), and subselect all the genes obtained from LPE-method (section **??**) for which absolute value of `z.statistics` is greater than 1.77.

Finally, here is an example of Bonferroni correction (sorted in order of significance):

```
> Bonferroni.adjp <- fdr.adjust(lpe.val, adjp="Bonferroni")
> head(Bonferroni.adjp)

                     FDR z.real
93086_at   1.161547e-183  28.47
94224_s_at  3.310302e-87  20.46
92830_s_at  1.298127e-83  20.02
94063_at    7.768063e-79  19.03
94304_at    1.626826e-64  17.97
92642_at    5.485022e-54  16.24
```

3 Discussion

Using our LPE approach, the sensitivity of detecting subtle expression changes can be dramatically increased and differential gene expression patterns can be identified with both small false-positive and small false-negative error rates. This is because, in contrast to the individual gene's error variance, the local pooled error variance can be estimated very accurately.

Acknowledgments. We wish to acknowledge the following colleagues: P. Aboyoun, J. Betcher, D Clarkson, J. Gibson, A. Hoering, S. Kaluzny, L. Kannapel, D. Kinsey, P. McKinnis, D. Stanford, S. Vega and H. Yan.

References

[1] Jain et. al. Local-pooled-error test for identifying differentially expressed genes with a small number of replicated microarrays, *Bioinformatics*, 2003, Vol 19, No. 15, pp: 1945-1951.

[2] Jain et. al. Rank-invariant resampling based estimation of false discovery rate for analysis of small sample microarray data, *BMC Bioinformatics*, 2005, Vol 6, 187.

[3] Chen et. al. Ratio-based decisions and the quantitative analysis of cDNA microarray images, *Biomedical Optics*, 1997, Vol 2, pp: 364-374.