http://bibs.snu.ac.kr/software/LPEseq/LPEseq_tutorial.pdf

Analysing no replicate RNA-seq data with LPEseq package

Gim, Jungsoo

BioInformatics and BioStatistics (BIBS) Lab,
Seoul National University,
Seoul, Korea (republic of)

iedenkim@gmail.com

2015-04-01 (version 0.99.3)

1. Introduction

The *LPEseq* is an R package for performing differential expression (DE) test with RNA sequencing data. Briefly, *LPEseq* extends local pooled error method, which was developed for microarray data analysis, to sequencing data even with non-replicated sample in each condition. A number of methods are available for both count-based and FPKM-based RNA-Seq data. Among these methods, few (for example, EdgeR and DESeq) can deal with no replicate data, but not accurately. *LPEseq* was designed for the RNA-Seq data with a small number of replicates, especially with non-replicate in each class. Also *LPEseq* can be equally applied both count-base and FPKM-based (non-count values) input data. This brief vignette is written for the users who want to use the *LPEseq* for their DE analysis. An extended documentation about the method can be found in our original manuscript (http://bibs.snu.ac.kr/software/LPEseq).

2. Installation

The source code and a package of *LPEseq* are freely available from our website (It will be soon available from Bioconductor). You can use it by loading source code in our website,

```
> source("http://bibs.snu.ac.kr/software/LPEseq/LPEseq.R")
```

```
> install.packages("local_folder/LPEseq_version.tar.gz", repos=NULL,
type="source")
```

3. What's in *LPEseq*

The *LPEseq* package comes with a number of functions to perform a differential expression test with or without replicates. The main functions are `LPEseq.normalise()` and `LPEseq.test()`, which are designed for running a normalization across the samples and a whole differential expression test, respectively. All the functions that start with the same as the package are newly developed in our method while others are from original *LPE* package. The most of the functions are described in the manual available in our web-site.

4. Quick Example

A. Input Data: generation

The *LPEseq* package starts its analysis with read counts. Therefore you have to equip yourself with RNA-Seq read count data sets on your hand first. The package expects count data in the form of a matrix (or a vector) of integer values. But it is not limited to *non-count* data, for example, *FPKM* values generated using *Cufflinks*. If you are not familiar with generating count table, please visit web-sites to learn how to obtain such a data. Good references are *GenomicRanges* in Bioconductor, *htseq-count* script written in Python framework, the well-known software and etc. In this *LPEseq* tutorial, you do not need count dataset right away. Using `generateData()`, you can generate simulated datasets and learn how to use *LPEseq* library without real dataset.

Now you are ready to generate the simulation datasets by typing

```
> set.seed(5)
> simData <- generateData(n.rep=3, n.deg=1000)
```

Let us take a look at the data generated using `generateData()` function.

```
> head(simData)
```

	condition1.1	condition1.2	condition1.3	condition2.1	condition2.2	condition2.3	DEG
gene_1	1299	541	1109	0	0	0	1
gene_2	0	0	0	0	0	0	0
gene_3	0	0	0	0	0	0	0
gene_4	26	39	41	27	36	31	0
gene_5	0	0	0	0	0	0	0
gene_6	0	0	0	0	0	0	0

The output of generated data consists of 20000 genes with 6 samples (three replicate per each condition) and differential expression index in the last column (denoted by DEG). DEGs are indexed with 1 otherwise 0.

If you have your own data, you can directly read the data with `read.table()`.Once you loaded your own data, the following analysis procedure is the same.

```
> yourData <- read.table("your_data.txt", header = , sep = , …)
```

To visualize mean and variance of the data, *LPEseq* provides `AVplot()` function.

```
> par(mfrow=c(1,2))
> AVplot(simData[,1:3])
> AVplot(simData[,1:3], logged=F)
```

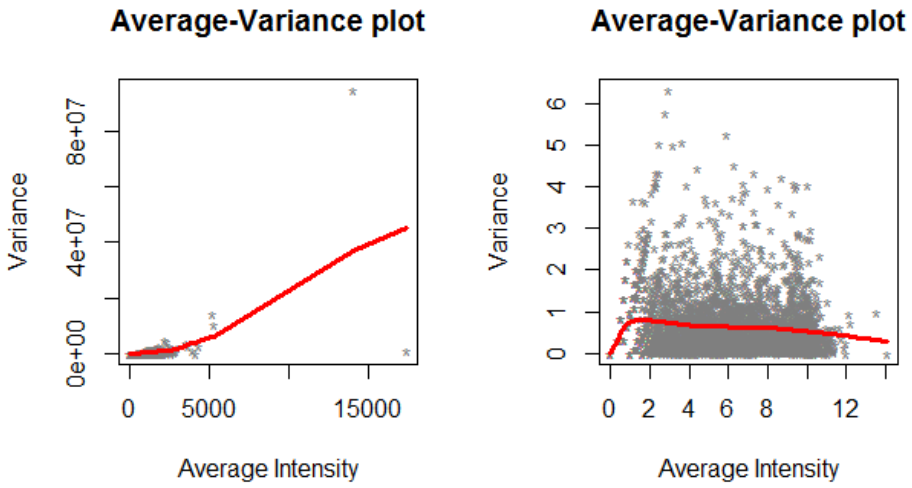

Figure 1 Variance versus mean intensity plot with original intensity (left) and normalized intensity (right)

B. Normalization

As the first step of analysis, we need to remove the effect of sequencing depth. *LPEseq* follows the similar idea of DESeq. *LPEseq* divides each column of the count table by the size factor for this column. By doing so, the count values are brought to a common comparable scale. *LPEseq* adds pseudo-count value 1 to all the values in the data and take log-2 transformation. Because of sparse properties in average bins of RNA-Seq in raw scale, it is recommended to log-2 transform the original data. *LPEseq* does this by typing

```
> simData.norm <- LPEseq.normalise(simData[,-7])
```

Note that DEG index is removed. If your own data consists of original count values, exactly the same script will do,

```
> youData.norm <- LPEseq.normalise(yourData)
```

But when your data includes normalized count values, such *RPKM* or *FPKM*, just take log-transformation to your data for further analyses. We recommend using log-transformed data for *LPEseq* method.

```
> youData.norm <- log(yourData, base = 2)
```

C. Testing Differential Expression

LPEseq provides simple one-step procedure to perform differential expression test. Unlike other methods, *LPEseq* is applicable to experiments without replicates. By simply providing expression matrix (or vector) per each condition as arguments of `LPEseq.test()` function, *LPEseq* automatically performs appropriate differential expression test, if the input data is properly given. Replicates are essential to interpret biological experiments. Nevertheless, experiments without any replicates per each condition are frequently undertaken, and *LPEseq* can deal with them. The followings are the R scripts for differential expression test with or without replicates in each condition, respectively.

```
> sim.result <- LPEseq.test(simData.norm[,1:3], simData.norm[,4:6])
```

```
> sim.result.norep <- LPEseq.test(simData.norm[,1], simData.norm[,4])
```

The result of LPEseq.test includes average values in each condition, pooled standard deviation, Z type statistics, nominal p-value and adjusted p-value (with Benjamini-Hochberg multiple testing correction).

```
> head(sim.result)
```

	mu.x	mu.y	pooled.std.dev	z.stats	p.value	q.value
gene_1	10.149991	0.000000	1.020765	-9.9435183	0.0000000	0.0000000
gene_2	0.000000	0.000000	1.273648	0.0000000	NA	NA
gene_3	0.000000	0.000000	1.273648	0.0000000	NA	NA
gene_4	5.299016	4.973103	0.804995	-0.4048625	0.6855786	0.9063719
gene_5	0.000000	0.000000	1.273648	0.0000000	NA	NA
gene_6	0.000000	0.000000	1.273648	0.0000000	NA	NA

```
> head(sim.result.norep)
```

	mu.x	mu.y	pooled.std.dev	z.stats	p.value	q.value
gene_1	10.353526	0.000000	1.715027	-6.03694650	1.570578e-09	2.843468e-08
gene_2	0.000000	0.000000	2.185038	0.00000000	NA	NA
gene_3	0.000000	0.000000	2.185038	0.00000000	NA	NA
gene_4	4.763784	4.847599	1.082377	0.07743624	9.382765e-01	9.961941e-01
gene_5	0.000000	0.000000	2.185038	0.00000000	NA	NA
gene_6	0.000000	0.000000	2.185038	0.00000000	NA	NA

To save the output to a file, use the `write.table()` function.

```
> write.table(sim.result, file="result_file.txt", quote=F, sep="\t")
```

5. How LPEseq works?

Since `LPEseq.test()` directly performs differential expression calling without additional steps, it would be worth describing how `LPEseq.test()` works. *LPEseq* first counts the number of input data column. If the number of column is larger than 1, *LPEseq* estimates LPE variance curve exactly the same way in original LPE method using `lpe.var()`. However, if the number of column is equal to 1, then *LPEseq* estimates LPE variance curve after performing the outlier-removing step using `LPEseq.var()`.

```
> sim.var <- lpe.var(simData.norm[,1:2], n.bin=100, df=10)
> sim.var.norep <- LPEseq.var(simData.norm[,1:2], n.bin=100, df=10, d=3,
fudge.factor=1)
> head(sim.var)
```

	A	var.M
gene_1	9.706107	0.5613424
gene_2	0.000000	3.1211790
gene_3	0.000000	3.1211790
gene_4	5.031400	0.5846971
gene_5	0.000000	3.1211790
gene_6	0.000000	3.1211790

```
> names(sim.var.norep)
```

```
[1] "x"       "y"       "w"       "yin"     "data"    "lev"     "cv.crit" "pen.crit" "crit"
[10] "df"      "spar"    "lambda"  "iparms"  "fit"     "call"
> par(mfrow=c(1,2))
> plot(sim.var)
> plot(sim.var.norep$x, sim.var.norep$y)
```

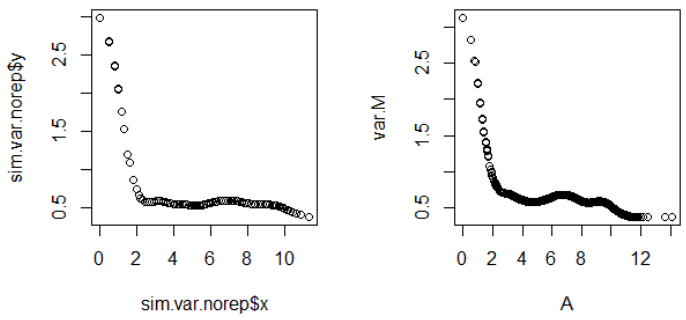

Figure 2 Comparison of LPE curve without replicates (left) and with replicates (right)

As can be seen in Fig. 2, LPE curve can be obtained from RNA-Seq data both with replicates and without replicates. But when working without any replicates, `LPEseq.var()` conducts an extra outlier-removing step. By removing these (possibly differentially expressed) genes or transcripts, the remaining genes can be thought as in the same condition. For this purpose *LPEseq* needs the extra argument, *d*, the expression difference between two different conditions. The default value is 1.2 (in log-2 transformed scale). Thus any genes or transcript whose expression difference between conditions is larger than 1.2 is depicted as outliers and is removed for evaluating LPE curve. This can be changed according to the data type. If the data is thought to be largely varied, like biological replicates do, the value can be larger. For the technical replicates, it is recommended to use 0.5 (please see supplementary note).

Once LPE curve is obtained, it is straightforward to estimate gene-specific variance from the curve.

```
> LPEseq.predict.var(5.342, sim.var.norep)
[1] 0.5418984
```

Then p-value of differential expression can be obtained from z-type statistics as described in the manuscript

```
> var.x <- LPEseq.predict.var(5.342, sim.var.norep)
> var.y <- LPEseq.predict.var(6.012, sim.var.norep)
> std.dev <- sqrt(var.x + var.y)
> z.stats <- (6.012-5.342)/std.dev
> p.val <- as.numeric(2*(1-pnorm(abs(z.stats))))
> p.val
[1] 0.5253698
```

6. Session Info

```
> sessionInfo()

R version 3.0.1 (2013-05-16)
Platform: i386-w64-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=Korean_Korea.949  LC_CTYPE=Korean_Korea.949    LC_MONETARY=Korean_Korea.949
[4] LC_NUMERIC=C                 LC_TIME=Korean_Korea.949

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