intSEQ: Differential Analysis of RNA-seq Data with Integrated Likelihood Method

Yilun Zhang
David Rocke

August 8, 2018

1 Introduction

This vignette is intended to give a brief introduction how to conduct differential analysis with RNA-seq data by means of the intSEQ R package. For further details of the methods, please consult [first paper], [second paper].

Consider the case we have RNA-seq count data arranged in a matrix that each columns describe a sample, and each row describe a genomic event (gene/exon/isoform). We denote the count in ith row and of jth group as Y_{ij}. Assume we have a predictor variable X, can be treatment/control group, dose of radiation, etc, which may or may or affect the level of expression across samples. We want to determine if it exist any pattern between the predictor X and different samples in each row.

The negative binomial distribution can be seen as Poisson-gamma mixture. If $Y \sim$Poisson(λ), and λ is a random variable following Gamma distribution with shape parameter $1/\theta$ and rate $\frac{1}{\mu\theta}$. Then Y has negative binomial distribution with mean parameter $\mu = E(Y)$ and dispersion parameter θ. The variance function of negative binomial distribution is

$$V(\mu) = \mu + \theta\mu^2$$

Many exsiting methods models RNA-seq data with negative binomial model. However, many of them suffers from false postive problem, that is, there are more than expected rejections occur. [first paper] suggests it may due to unaccounted variability of estimated dispersion parameters. The integrated likelihood method integrate the likelihood over the support of dispersion parameter rather than using a point estimator of the dispersion to control the false positive. It seems outperform other methods that also controls the false postive in term of higher power when the sample size is moderated.

2 Prepare the data

2.1 Read the data

The users should arrange data into a numeric matrix with rows corresponds to genomic event (such as genes), and columns corresponds to samples. We also accept object of class "DGEList" in edgeR.

We use the Montgomery and Pickrell data as an example. The data set is RNA-Seq of RNA from lymphoblastoid cell lines from 129 individuals from the HapMap project. There were 60 from the above-referenced CEU subset and 69 from the Yoruba people in Ibadan, Nigeria (YRI). 52,580 unique genes of which 8,124 had a count of at least 129 across the 129 samples. We choose 10 CEU and 10 YRI individuals to conduct differential analysis.

```
library(intSEQ)
data("count.data")
data("condition")
```

```
count = count.data[ , c(1:10, 61:70)]
cond = condition[c(1:10, 61:70)]
count[1:10,]
```

```
##                 NA06985 NA06986 NA06994 NA07000 NA07037 NA07051 NA07346 NA07347 NA07357
## ENSG00000000419      11      16      10      19       8      24      14      19      20
## ENSG00000000457      28      22      17      21      18      15      21      13      18
## ENSG00000000460       0       7       2       4       6       3       8       5       2
## ENSG00000000938       8      28      20      10      17       4      19      35      16
## ENSG00000001036      19      25      13      22      10      16      12      12      23
## ENSG00000001167     112     107      89      95      75      87      80      73      82
## ENSG00000001497      10       9      13       4      18      10      10       7       4
## ENSG00000001561      83      62      52      69      18      44      30      50     108
## ENSG00000002016       1       1       1       1       0       0       0       0       0
## ENSG00000002330      69      82      51      69     100      88      34      47      65
##                 NA10847 NA18486 NA18498 NA18499 NA18501 NA18502 NA18504 NA18505 NA18507
## ENSG00000000419       6      22     105      40      55      67      37      88     127
## ENSG00000000457      15      22     100     107      53      72      38      98      69
## ENSG00000000460       2       5      23      10      18      15       8      11      16
## ENSG00000000938       9      36      70      41      33      59      29      22      71
## ENSG00000001036       4      29      79      33      31      29      21      42      62
## ENSG00000001167     108     301     351     344     176     340     170     238     247
## ENSG00000001497      10      32      22       4       8      15       5      15      16
## ENSG00000001561      31      52     419     173     137     127      72     267     247
## ENSG00000002016       1       3       8       6       6       5       4       3       4
## ENSG00000002330      41      48      81      65      19      59      57      44      79
##                 NA18508 NA18510
## ENSG00000000419      70      43
## ENSG00000000457      66      43
## ENSG00000000460      18       7
## ENSG00000000938      12      43
## ENSG00000001036      41      25
## ENSG00000001167     226     227
## ENSG00000001497      19      15
## ENSG00000001561     154      81
## ENSG00000002016       6       3
## ENSG00000002330      71      79
```

```
cond
```

```
##  [1] CEU CEU CEU CEU CEU CEU CEU CEU CEU CEU YRI YRI YRI YRI YRI YRI YRI YRI YRI YRI
## Levels: CEU YRI
```

2.2 Filtering

Since almost all existing methods have very low power for low expressed gene. We recommend filter out those genes with small average row count.

```
count = count[rowMeans(count) >= 1,]
```

3 intSEQ Approach

3.1 Normalization

The intSEQ function has a built-in normalization method "TMM". The users can use this method by setting "normalize=TRUE". Please don't normalize the count data yourself. Any non-integer input of count.data will cause error.

3.2 Integrated Likelihood Ratio Test

The intSEQ function will estimate the Cox and Reid's estimator of dispersion, then fit the negative binomial model. After that, the likelihood will be integrated over the dispersion:

$$IL(\mu) = L(Y|\mu) = \int L(Y|\mu,\theta)\pi(\theta|\mu)d\theta$$

Then the integrated likelihood will be treated as likelihood function. The integrated likelihood ratio statistics is compared with a χ^2 distribution.

```
res=intSEQ(count, cond)
```

Alternatively, we can use DGEList

```
library(edgeR)
dge=DGEList(counts=count, group=cond)
res2=intSEQ(dge)
```

3.3 Display the Results

There is a function named *show* to display the result. The users can choose to sort the genes from p values small to large, sort the log fold change from large to small by changing argument *sortby*.

```
show(res, sortby = "pvalue", shownum = 10)
```

```
##                      logFC   logCPM          FDR    intLR    intPValue ordinaryLR
## ENSG00000185246  2.1449454 24.27938 1.768360e-07 44.80496 2.176711e-11   170.9681
## ENSG00000079134  1.4882846 24.43664 4.880124e-07 41.46271 1.201409e-10   142.6679
## ENSG00000165792  0.9596607 25.82333 5.462896e-07 40.44981 2.017318e-10   177.3485
## ENSG00000084733  0.9930815 24.46722 5.511677e-07 39.75877 2.873479e-10   129.6687
## ENSG00000108465  1.3800198 24.73898 5.511677e-07 39.43466 3.392219e-10   138.4683
## ENSG00000135617 -2.7759272 21.06611 1.077335e-06 37.05008 1.151341e-09   132.4831
## ENSG00000178971  1.2574490 26.23903 1.077335e-06 36.75513 1.339375e-09   118.0295
## ENSG00000167136 -1.8795301 24.45529 1.077335e-06 36.75472 1.339659e-09   114.4304
## ENSG00000181472  1.0825896 25.51406 1.077335e-06 36.64482 1.417352e-09   126.5957
## ENSG00000188986 -0.8065574 27.67979 1.077335e-06 36.35038 1.648474e-09   115.4995
##                 ordinaryPValue
## ENSG00000185246   4.546990e-39
## ENSG00000079134   6.947704e-33
## ENSG00000165792   1.838165e-40
## ENSG00000084733   4.842153e-30
## ENSG00000108465   5.756722e-32
## ENSG00000135617   1.172981e-30
## ENSG00000178971   1.708275e-27
## ENSG00000167136   1.048858e-26
## ENSG00000181472   2.277463e-29
## ENSG00000188986   6.117582e-27
```

```
show(res, sortby = "LFC", shownum = 10)
```

```
##                     logFC   logCPM          FDR     intLR    intPValue ordinaryLR
## ENSG00000144214 5.247848 19.93416 7.873251e-06 28.818967 7.946906e-08   41.94483
## ENSG00000174171 5.228006 19.75586 1.952697e-03 13.640377 2.213730e-04   19.37467
## ENSG00000166523 5.081534 19.65238 1.226811e-02  9.146696 2.491676e-03   11.87244
## ENSG00000171217 4.937806 19.71183 6.605082e-05 22.792236 1.804934e-06   30.60231
## ENSG00000175265 4.836845 22.20792 1.416953e-05 27.295553 1.746126e-07   56.69613
## ENSG00000173110 4.827374 21.83189 3.738208e-03 12.026254 5.245639e-04   14.96935
## ENSG00000204666 4.659077 19.46568 4.994931e-03 11.318585 7.673515e-04   14.77077
## ENSG00000171747 4.484295 19.39571 3.794746e-03 11.980381 5.376358e-04   14.73600
## ENSG00000117616 4.398118 20.09435 2.690687e-04 18.936665 1.351305e-05   33.98861
## ENSG00000183578 4.342930 19.98446 9.153824e-03  9.882494 1.668586e-03   14.16119
##                 ordinaryPValue
## ENSG00000144214   9.388517e-11
## ENSG00000174171   1.074225e-05
## ENSG00000166523   5.697087e-04
## ENSG00000171217   3.167162e-08
## ENSG00000175265   5.086367e-14
## ENSG00000173110   1.092718e-04
## ENSG00000204666   1.214025e-04
## ENSG00000171747   1.236622e-04
## ENSG00000117616   5.543564e-09
## ENSG00000183578   1.677962e-04
```

4 Choose the Best Methods

We found the performance of methods fluctuant as the sample size differs. We suggest the strategy to simulate synthetic negative binomial distributed data many times and apply each methods to the simulated data. Then compare their null performance and power. The simulated data uses the means and dispersions estimated from the real data. If user set "null = TRUE", it will simulate two groups of RNA-seq data with same expression level each row and will run FPR analysis. Otherwise, it will use the means of two groups and then conduct power analysis.

4.1 Null performance

Suppose we want to repeat the study in previous section for 10 times with same sample size. More number of simulations are always recommended.

```
set.seed(100)
simu.res.null <- simuComp(res, nsamp=10, ntime = 10, null=TRUE)
```

```
## ----------
```

The FPR performance can be shown by *summary*, the FPR should be controlled under levels.

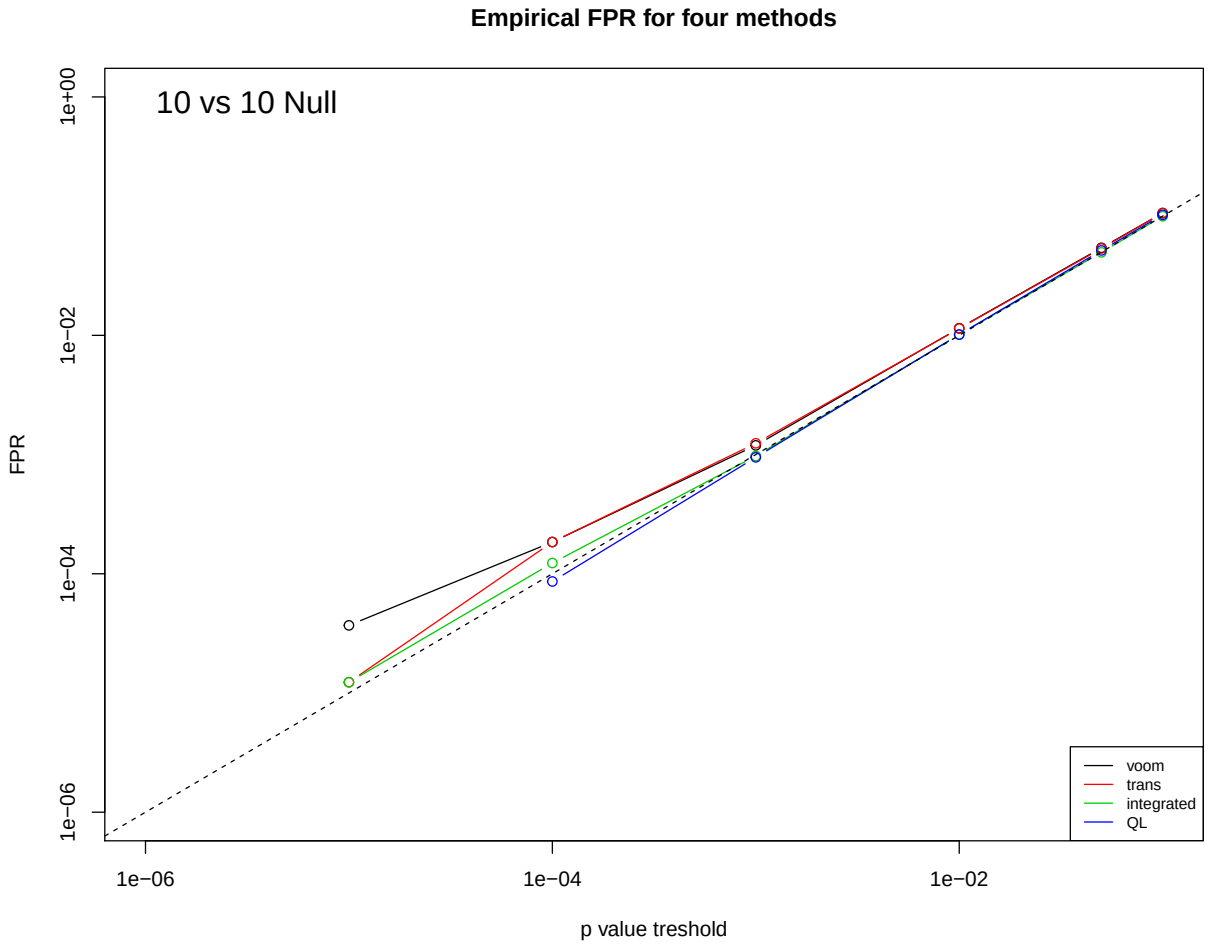
```
summary(simu.res.null)
```

```
## The four methods has FPR of:
##
##             treshold = 1e-06 treshold = 1e-05 treshold = 1e-04 treshold = 0.001
## limma.voom                 0            4e-05          0.00018          0.00119
## limma.trans                0            1e-05          0.00018          0.00124
```

```
## IntSEQ                     0            1e-05          0.00012          0.00097
## edgeR.Quasi                0            0e+00          0.00009          0.00095
##             treshold = 0.01 treshold = 0.05 treshold = 0.1
## limma.voom          0.01148         0.05423        0.10624
## limma.trans         0.01141         0.05357        0.10586
## IntSEQ              0.01014         0.04968        0.09984
## edgeR.Quasi         0.01016         0.05158        0.10254
```

To plot the levels against FPR:

```
plotComp(simu.res.null, text = " 10 vs 10 Null")
```



The four method all maintained the expected size.

4.2 Power Analysis

This is similar with the null case, except setting the "null = FALSE":

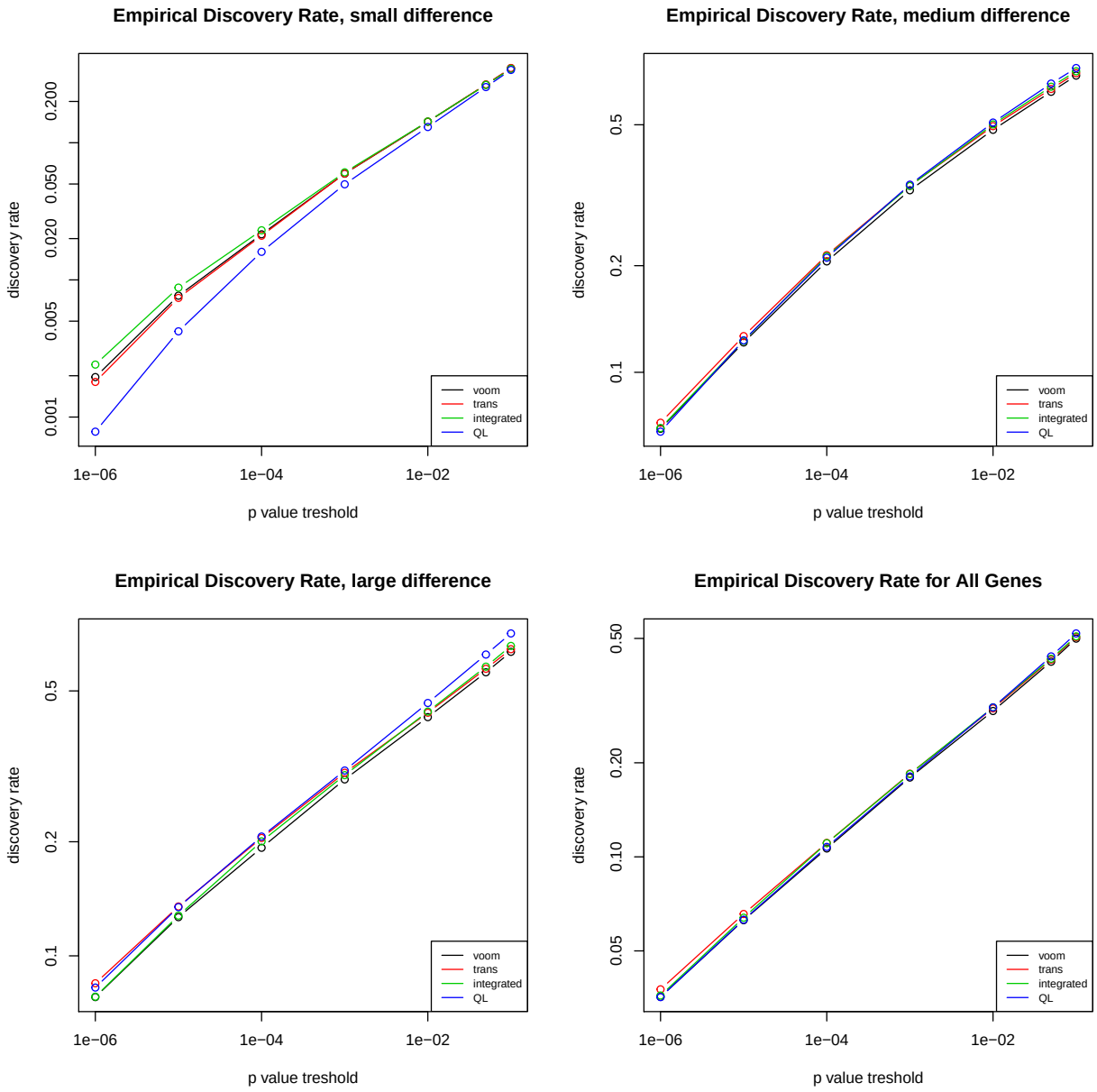
```
set.seed(100)
simu.res.full <- simuComp(res, nsamp=10, ntime = 10, null=FALSE)

## ----------
```

The summary function first will display the discovery rate of four method under FDR level equals to the argument "fdrlevel" passed to *simuComp*. Then we categorize genes into different groups with respect to their log fold change. The users can choose "small" to display power of genes that are diminutively different, "medium" and "large" for medium and large log fold change genes, and "all" for all genes.

```
summary(simu.res.full, difflevel = "small")

## The number of genes in each category are:
##  small medium  large
##   4602   2408   1751
## Under FDR threshold 0.1 the four method has average discovery rate of:
##  limma.voom limma.trans      IntSEQ edgeR.Quasi
##   0.3713811   0.3784958   0.3778065   0.3816716
##
## ------------------------------------------------------------------
##
## For those genes with small difference the four method has average discovery rate of:
##
##             treshold = 1e-06 treshold = 1e-05 treshold = 1e-04 treshold = 0.001
## limma.voom            0.00196          0.00767          0.02149          0.05947
## limma.trans           0.00180          0.00739          0.02099          0.05963
## IntSEQ                0.00241          0.00878          0.02299          0.06082
## edgeR.Quasi           0.00078          0.00422          0.01601          0.04976
##             treshold = 0.01 treshold = 0.05 treshold = 0.1
## limma.voom          0.14220         0.26471        0.34911
## limma.trans         0.14246         0.26615        0.35035
## IntSEQ              0.14276         0.26406        0.34585
## edgeR.Quasi         0.13020         0.25458        0.33946

summary(simu.res.full, difflevel = "all")

## Under FDR threshold 0.1 the four method has average discovery rate of:
##  limma.voom limma.trans      IntSEQ edgeR.Quasi
##   0.3713811   0.3784958   0.3778065   0.3816716
##
## ------------------------------------------------------------------
##
## For those genes with all difference the four method has average discovery rate of:
##
##             treshold = 1e-06 treshold = 1e-05 treshold = 1e-04 treshold = 0.001
## limma.voom            0.03562          0.06269          0.10635          0.17939
## limma.trans           0.03767          0.06563          0.11080          0.18458
## IntSEQ                0.03599          0.06386          0.11058          0.18411
## edgeR.Quasi           0.03558          0.06292          0.10757          0.18085
##             treshold = 0.01 treshold = 0.05 treshold = 0.1
## limma.voom          0.29290         0.42124        0.49928
## limma.trans         0.29856         0.42760        0.50499
## IntSEQ              0.30086         0.43051        0.50765
## edgeR.Quasi         0.30067         0.43800        0.51872
```

There will be four plot showing discovery rate for small, medium, large group genes.

```
par(mfrow=c(2,2))
plotComp(simu.res.full, text = "")
```



We recommend the users to use the method that has highest power with FPR controlled.

5 Session Information

```
sessionInfo()

## R version 3.4.3 (2017-11-30)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
```

```
## [1] LC_COLLATE=Chinese (Simplified)_China.936  LC_CTYPE=Chinese (Simplified)_China.936
## [3] LC_MONETARY=Chinese (Simplified)_China.936 LC_NUMERIC=C
## [5] LC_TIME=Chinese (Simplified)_China.936
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] intSEQ_0.99.0 edgeR_3.20.8  limma_3.34.8  knitr_1.19
##
## loaded via a namespace (and not attached):
##  [1] locfit_1.5-9.1     Rcpp_0.12.15       lattice_0.20-35    matrixStats_0.53.1
##  [5] grid_3.4.3         magrittr_1.5       evaluate_0.10.1    highr_0.6
##  [9] stringi_1.1.6      splines_3.4.3      tools_3.4.3        stringr_1.2.0
## [13] compiler_3.4.3     fastGHQuad_0.2
```