MDSeq: Gene expression mean and variability analysis for RNA-seq counts

Di Ran and Z. John Daye

March 3, 2017

Introduction

This guide provides an overview of the R package MDSeq for differential mean and dispersion analysis of RNA-seq count data. The package can also be adopted for the analysis of read counts from other genomic platforms. The MDSeq implements a mean-dispersion model, based on the coefficient of dispersion, for efficient analysis of gene expression mean and variability. The MDSeq utilizes a novel reparametrization of the negative binomial to provide flexible generalized linear models (GLMs) that can incorporate treatment effects and additional covariates on both the mean and dispersion. Simultaneous analysis of differential dispersion along with the mean is a unique feature of the MDSeq, compared with current packages that only focus on the analysis of expression means. The MDSeq provides comprehensive features to account for technical excess zeros often seen in RNA-seq data due to technical variability, identify outliers with computational efficiency, and perform hypothesis tests at biologically interesting levels based on user-specified log fold-change thresholds. Several assistant functions were implemented to help users in filtering and normalization of raw read counts. Options are provided that allow users to incorporate normalization factors as re-scaled counts or offsets in the mean-dispersion GLMs. Additionally, the software allows for parallel processing with multiple threads for efficient computations.

Further details can be found in Ran and Daye (2017).

Installation

Install MDSeq from local source with

```
install.packages("MDSeq_1.0.5.tar.gz", repos=NULL, type="source")
```

Install MDSeq from GitHub with

```
library(devtools)
install_github("zjdaye/MDSeq")
```

Getting started

We start the R session with loading the MDSeq package as follows:

```
library(MDSeq)
```

We illustrate the analysis by using a sample RNA-seq dataset, which contains 200 samples with expression counts of 49722 genes. These 200 samples come from two treatment groups - half of which are from the case group while the other half are from the control group. In addition, each sample has two demographic variables, which are considered to be covariates related to both the expected mean and dispersion. First, we load the sample dataset.

```
data(sampleData)

# raw count table
dat <- sample.exprs
dim(dat)
[1] 49722   200

dat[1:6, 1:6]
      sample1 sample2 sample3 sample4 sample5 sample6
gene1      33      72      94     106      64      73
gene2       0       0       0       0       0       3
gene3     951     969    1126     795     977     802
gene4       0       0       0       0       0       0
gene5       0       6       0      10       0       0
gene6       0       0       0       0       0       0
```

In the raw count table, each row represents one gene and each column represents one sample. For example, the number "33" in upper-left corner is the read count of gene1 from sample1.

The MDSeq is based on flexible GLMs. Users can specify covariates related to either the mean or dispersion, or both. In this example, there are two covariates, *X1* and *X2*, that are assumed to have association with both the mean and dispersion. Both covariates in this example are categorical. Note that continuous covariates can be included in the GLMs in the same manner.

```
# covariates
X <- sample.pheno[,c("X1","X2")]
head(X)
        X1 X2
sample1  2  2
sample2  3  1
sample3  1  1
sample4  1  1
sample5  3  1
sample6  2  1

# group information
group <- sample.pheno$group
summary(factor(group))
 -1   1
100 100
```

Filtering and Normalization

We could filter out lowly expressed genes that provide little information for gene expression analysis. The function *filter.counts()* is provided in the MDSeq to filter lowly expressed genes. By default, it will remove genes whose mean counts per million (cpm) is less than 0.05. cpm is calculated using the edgeR package (Robinson, McCarthy, and Smyth 2010). In this example, we applied a more stringent threshold with mean cpm greater than 0.1. As a result, 21992 high-quality genes were obtained from the original 49722 genes. We highly recommend users to remove lowly expressed genes for robust analysis of gene expression mean and variability. Different criteria or other methods can also be applied. For example, one could filter out more genes by setting a higher cutoff point of cpm; or one could include more genes by setting a smaller cutoff point.

Since the MDSeq evaluates relative differences between treatments, results may be effected by technical biases due to compositional differences among libraries, and technical biasness should be taken into account by normalization. In this example, we computed the TMM normalization factors (Robinson, McCarthy, and Smyth 2010). The normalization factors can be incorporated, in the MDSeq, as either normalized counts or offsets in the mean-dispersion GLMs. Offsets can be applied in the function *MDSeq()* by inputing normalization factors with the *offsets* parameter. The MDSeq also provides the function *normalize.counts()* which can calculate normalized counts based on different normalization methods, such as "upperquartile" (Bullard et al. 2010) and "RLE" (Anders and Huber 2010). User could also apply more complex normalization approaches such as the "cqn", which considers adjustment for both gene length and GC content. This optional method depends on the implement in R package "cqn" (Hansen, Irizarry, and Wu 2012).

```
# lowly expressed genes were filtered by mean cpm value across all samples
dat.filtered <- filter.counts(dat, mean.cpm.cutoff = 0.1)
dim(dat.filtered)
[1] 21992   200

# inlcuding normalization factor as an offset in mean-dispersion GLM
# alternatively, normalized counts can be used as input
# require(edgeR)
cnf <- calcNormFactors(dat.filtered, method="TMM")
libsize <- colSums(dat.filtered)              #normalization factor
rellibsize <- libsize/exp(mean(log(libsize))) #relative library size
nf <- cnf * rellibsize                        #normalization factor including library size
```

Design matrix and constrast

The MDSeq allows users to specify the design matrix using various contrast settings. Function *get.model.matrix()* allows one to generate the design matrix and form the contrast. For example, in the analysis of the sample dataset, the goal is to make a comparison between two treatments or groups, which is the most common and simple situation in biological research. It is necessary to form a contrast between the case and control groups and assign it to each sample within its group in the design matrix. First of all, treatment/group indicator needs to be in factor formats with predefined levels and labels. User could use the default contrast matrix or provide any proper contrast matrix. The default considers the contrast setting to make comparison such as "$-1 * control + 1 * case$". It can also be extended to multi-group comparisons, greater than or equal to 3 groups. User can compare any of the treatment groups using different contrast settings. For example, a three-groups comparison can use a sum to zero contrast (default in MDSeq) as follows.

```
get.model.matrix(factor(rep(1:3,each=4)))
$mean
   (Intercept) group1 group2
1            1      1      0
2            1      1      0
3            1      1      0
4            1      1      0
5            1      0      1
6            1      0      1
7            1      0      1
8            1      0      1
9            1     -1     -1
10           1     -1     -1
11           1     -1     -1
```

```
12           1     -1     -1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$group
  [,1] [,2]
1    1    0
2    0    1
3   -1   -1

$dispersion
...
```

One could also use a more traditional GLM approach with treatment contrast setting. The previous three-levels treatment example can be modified in this more traditional way.

```
get.model.matrix(factor(rep(1:3,each=4)), contrast.type = "contr.treatment")

$mean
...

$dispersion
   (Intercept) group2 group3
1            1      0      0
2            1      0      0
3            1      0      0
4            1      0      0
5            1      1      0
6            1      1      0
7            1      1      0
8            1      1      0
9            1      0      1
10           1      0      1
11           1      0      1
12           1      0      1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$group
  2 3
1 0 0
2 1 0
3 0 1
```

Other types of contrasts are also allowed. Please check R function *model.matrix()*.

Design matrix and contrast used in this illustration.

```
# treatment group assignment
group <- factor(sample.pheno$group, labels = c("Control", "Case"))
table(group)
group
Control    Case
    100     100
```

```
# make design matrix with proper contrast setting
groups <- get.model.matrix(group)
groups
$mean
    (Intercept) group1
1             1      1
2             1      1
3             1      1
4             1      1
...         ...    ...
...         ...    ...
197           1     -1
198           1     -1
199           1     -1
200           1     -1
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$group
        [,1]
Control    1
Case      -1

$dispersion
    (Intercept) group1
1             1      1
2             1      1
3             1      1
4             1      1
...         ...    ...
...         ...    ...
197           1     -1
198           1     -1
199           1     -1
200           1     -1
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$group
        [,1]
Control    1
Case      -1
```

In this example, we can apply two possible comparisons "case vs. control" or "control vs. case". They have the same effect sizes but differing signs. In the three-groups example, one can make comparisons between any pairs of groups. For example, one can test "2 vs. 1" or "3 vs. 2". MDSeq provides the function *extract.ZIMD()* to perform any pairwise comparisons (see Section "Differential mean and dispersion analysis" below).

Checking and removing outliers

Outlier detection is an important step in RNA-seq studies to allow robust model estimation and inference. The MDSeq provides a computationally efficient procedure to detect outliers that are influential for statistical

inference on a given set of parameters of interest (Ran and Daye 2016). The function *remove.outlier()* will remove outliers and provide a summary of outlier detection. A count matrix of RNA-seq counts will be outputted, in which outliers are set to NA. A summary indicates the status (label=0 for successful detection) and numbers of outliers found at each gene. Outliers will be ignored in ensuing analyses.

After outlier detection, outliers within a gene were replaced by *NA*. The mean-dispersion GLM utilizes the observed Fisher information to compute the variance-covariance matrix. In some rare scenarios, the observed Fisher information may be non-invertible. An error is flagged in these scenarios, even though an estimate may still be available. We recommend removing those genes with nonzero error flags from downstream analysis. As a consequence, we removed 126 genes and finally got 21866 genes in which outliers were replaced by *NA*.

```
# Check outliers using parallel process with 4 threads
# For the sake of simplicity, we demonstrate outlier detection
# by using the first 100 genes.

dat.checked <- remove.outlier(dat.filtered[1:100, ], X=X, U=X,
                              contrast = groups, offsets = nf,
                              mc.cores = 4)
4 threads are using!
Total time elapsed:9.5 seconds

head(dat.checked$outlier)
       status num.outliers
gene1       0            3
gene3       0            0
gene5       2           NA
gene7       0            0
gene9       0            4
gene10      0            2

# status of outlier checking
table(dat.checked$outlier$status)
 0  2
99  1

# frequency distribtuion of outliers
table(dat.checked$outlier$num.outliers)
 0  1  2  3  4  5  6  7  9 18
28 11 10 10 18 10  6  4  1  1

# remove genes with status flag other than 0
counts <- dat.checked$count[dat.checked$outliers$status==0,]
dim(counts)
[1] 99   200
```

Differential mean and dispersion analysis

Differential mean and dispersion analysis can be performed simultaneously using the main function *MDSeq()*, which allows parallel processing with multiple threads. For experiments with many genes, one can take advantage of parallelized computation. For instance, an example data with 200 samples and around 20,000 genes will cost less than 30 minutes with 4 threads on a lab desktop. MDSeq requires at least several inputs, such as expression count matrix, optional covariates of the mean(X) and dispersion(U), design matrix with proper contrast setting. By default, this function will simultaneously test zero-inflation using a likelihood

ratio test. One can also assess the goodness-of-fit of zero-inflation using function *gof.ZI()*. MDSeq will return an "ZIMD" class object. In this example, estimations are stored in *fit$Dat*, which contains not only the estimations of GLM coefficients but also the results for testing zero-inflation. For example, one can find "s" the estimated proportion of excess zeros, "ZI.pval" p-value of likelihood ratio test, and "ZI" zero-inflation indicator.

```
# using parallel process with 4 threads
fit <- MDSeq(counts, X=X, U=X, contrast = groups, offsets = nf, mc.cores = 4)
4 threads are using!
Total time elapsed:9.4 seconds

# simultaneously test zero-inflation
head(fit$Dat[c("s","ZI.pval", "ZI")], 20)
                s      ZI.pval ZI
gene1  0.00000000 1.087272e-01  0
gene3  0.00000000 1.000000e+00  0
gene7  0.00000000 1.000000e+00  0
gene9  0.00000000 1.000000e+00  0
gene10 0.05020535 1.399395e-05  1
gene16 0.00000000 1.000000e+00  0
gene24 0.00000000 1.000000e+00  0
gene26 0.00000000 1.000000e+00  0
gene28 0.00000000 1.000000e+00  0
gene30 0.00000000 1.000000e+00  0
gene36 0.10616926 2.621651e-02  1
gene37 0.00000000 1.000000e+00  0
gene41 0.00000000 2.053253e-01  0
gene42 0.00000000 1.299335e-01  0
gene50 0.09224318 2.233739e-02  1
gene53 0.00000000 1.000000e+00  0
gene55 0.00000000 1.000000e+00  0
gene56 0.04319981 1.222741e-07  1
gene59 0.02411948 1.083405e-02  1
gene61 0.00000000 1.000000e+00  0
```

Testing with a given |log2|-fold-change

It is often of interest to identify genes with differential changes beyond a given threshold level that could more readily allow for experimental replication and biological interpretations. Traditional methods of statistical inference often focus on evaluating the compliant hypothesis that any change in differential expressions $Ha : |log2FC| \neq 0$ may occur. This would often result in the selection of a large proportion of genes that are only mildly differentially expressed and cannot be replicated experimentally, especially in RNA-seq studies with moderate to large numbers of samples. Therefore, we provided inequality hypothesis tests $Ha : |log2FC| > \tau$ for both differential mean and dispersion that evaluate whether absolute log-fold changes are above a given threshold level τ. We applied a rigorous development based on one-sided hypothesis tests within restricted parameter spaces and union-intersection principle (Ran and Daye 2016). The function *extract.ZIMD()* is provided in the MDSeq that incorporates inequality hypothesis tests. One can specify any |log2|-fold-change threshold "log2FC.threshold", which will be applied on both mean and dispersion testing. In addition. this function offers the comparison between any possible single pair. One can test any comparison, such as "2 vs. 1" or "3 vs. 2" in a three-levels treatment, for example. To make a comparison "2 vs. 1", one needs to specify "A='2' " and "B='1' " in the "compare" argument.

```
# given log2-fold-change threshold = 1
result <- extract.ZIMD(fit, compare = list(A="Case", B="Control"), log2FC.threshold = 1)
head(result)
```

```
       CasevsControl.mean.log2FC.1 Statistics.mean  Pvalue.mean     FDR.mean
gene1                   -2.1899929       101.59256 3.410050e-24 8.739282e-23
gene3                    0.2019807         0.00000 1.000000e+00 1.000000e+00
gene7                    0.1346394         0.00000 1.000000e+00 1.000000e+00
gene9                    1.7598168        49.96804 7.813551e-13 1.334972e-11
gene10                  -1.4337538        11.08457 4.352445e-04 6.029435e-03
gene16                  -1.7313686       343.09159 6.769509e-77 2.669063e-75
       CasevsControl.dispersion.log2FC.1 Statistics.dispersion Pvalue.dispersion
gene1                         -0.6000018             0.0000000      1.000000e+00
gene3                         -0.3092432             0.0000000      1.000000e+00
gene7                          1.1305998             0.1807466      3.353662e-01
gene9                          2.4893128            23.9299494      4.995269e-07
gene10                        -1.8636851             6.1933917      6.411410e-03
gene16                        -1.4732579             2.3939073      6.090442e-02
       FDR.dispersion
gene1    1.000000e+00
gene3    1.000000e+00
gene7    1.000000e+00
gene9    2.560377e-05
gene10   2.053897e-01
gene16   1.000000e+00
```

Session Information

```
sessionInfo()
```

```
R version 3.3.1 (2016-06-21)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 7 x64 (build 7601) Service Pack 1

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

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

other attached packages:
[1] quadprog_1.5-5 gtools_3.5.0   edgeR_3.14.0   limma_3.28.17  MDSeq_1.0.5

loaded via a namespace (and not attached):
[1] tools_3.3.1
```

References

Anders, Simon, and Wolfgang Huber. 2010. "Differential Expression Analysis for Sequence Count Data." *Genome Biology* 11 (10). BioMed Central: 1.

Bullard, James H, Elizabeth Purdom, Kasper D Hansen, and Sandrine Dudoit. 2010. "Evaluation of Statistical Methods for Normalization and Differential Expression in MRNA-Seq Experiments." *BMC Bioinformatics* 11 (1). BioMed Central: 1.

Hansen, Kasper D, Rafael A Irizarry, and Zhijin Wu. 2012. "Removing Technical Variability in RNA-Seq Data Using Conditional Quantile Normalization." *Biostatistics* 13 (2). Biometrika Trust: 204–16.

Ran, Di, and John Z Daye. 2016. "Gene Expression Variability in Large-Scale RNA-Seq Studies with the MDSeq." *Submitted*.

Robinson, Mark D, Davis J McCarthy, and Gordon K Smyth. 2010. "EdgeR: A Bioconductor Package for Differential Expression Analysis of Digital Gene Expression Data." *Bioinformatics* 26 (1). Oxford Univ Press: 139–40.