Differential analyses with DSS

Hao Wu

Department of Biostatistics and Bioinformatics
Emory University
Atlanta, GA 303022

hao.wu@emory.edu

November 2, 2014

Contents

Abstract

This vignette introduces the use of the Bioconductor package DSS (Dispersion Shrinkage for Sequencing data), which is designed for differential analysis based on high-throughput sequencing data. It performs differential expression analyses for RNA-seq, and differential methylation analyses for bisulfite sequencing (BS-seq) data. The core of DSS is a new procedure to estimate and shrink gene- or CpG site-specific dispersions, then conduct Wald tests for differential expression/methylation. Compared with existing methods, DSS provides excellent statistical and computational performance.

1 Introduction

Recent advances in high-throughput sequencing technology have revolutionized genomic research. For example, RNA-seq is a new technology for measuring the abundance of RNA products. Compared to gene expression microarrays, it provides a better dynamic range and lower signal-to-noise ratio. Bisulfite sequencing (BS-seq) is a new technology for measuring DNA methylation. Compared to capture-based methods such as MeDIP-seq, it provides single-base resolution and eliminates biases associated with CpG density.

Fundamental questions for RNA-seq or BS-seq data analyses are whether gene expression regulation or DNA methylation dynamics vary under different biological contexts. Identifying sites or regions exhibiting differential expression (DE) or differential methylation (DM) are thus key tasks in functional genomics research.

RNA- or BS-seq experiments typically have a limited number of biological replicates due to cost constraints. This can lead to unstable estimation of within group variance, and subsequently undesirable results from hypothesis testing. Variance shrinkage methods have been widely used in DE analyses based on microarray data. The methods are typically based on a Bayesian hierarchical model, with a prior imposed on the gene-specific variances to provide a basis for information sharing across all genes/CpG sites. In these models, shrinkage is achieved for variance estimation. Using shrunk variance in hypothesis tests has been shown to provide better results.

A distinct feature of RNA-seq or BS-seq data is that the measurements are in the form of counts. These data are often assumed to be from the Poisson (for RNA-seq) or Binomial (for BS-seq) distributions. Unlike continuous distributions such as the Gaussian distribution, the variances depend on means in these discrete distributions. This implies that the sample variances do not account for biological variation between replicates, and shrinkage cannot be applied on variances directly.

In contrast, we assume that our count data come from the Gamma-Poisson (RNA-seq) or Beta-Binomial (BS-seq) distribution. These distributions can be parameterized by a mean and an over dispersion parameter. The over dispersion parameters, which represent the biological variation for replicates within a treatment group, play a central role in the differential analyses.

Here we present a new DE/DM detection algorithm, where shrinkage is performed on the dispersion parameters. We first impose a log-normal prior on the dispersions, and then combine data from all genes/CpG sites to shrink dispersions through a penalized likelihood approach. Finally, we construct Wald tests to test each gene/site for differential expression/methylation. Our results show that the new method provides excellent performance compared to existing methods, especially when the overall dispersion level is high or the number of replicates is small.

Currently DSS only supports comparison of expression or methylation from two treatment groups. Methods for more advanced study designs are under development and will be implemented soon.

2 Using `DSS` for differential expression analysis

2.1 Single factor experiment

Required inputs for DSS are (1) gene expression values as a matrix of integers, rows are for genes and columns are for samples; and (2) a vector representing experimental designs. The length of the design vector must match the number of columns of input counts. Optionally, normalization factors or additional annotation for genes can be supplied.

The basic data container in the package is `SeqCountSet` class, which is directly inherited from `ExpressionSet` class defined in `Biobase`. An object of the class contains all necessary information for a DE analysis: gene expression values, experimental designs, and additional annotations.

A typical DE analysis contains the following simple steps.

1. Create a `SeqCountSet` object using `newSeqCountSet`.

2. Estimate normalization factor using `estNormFactors`.

3. Estimate and shrink gene-wise dispersion using `estDispersion`

4. Two-group comparison using `waldTest`.

The usage of DSS is demonstrated in the simple simulation below.

1. First load in the library, and make a `SeqCountSet` object from some counts for 2000 genes and 6 samples.

```
> library(DSS)
> counts1=matrix(rnbinom(300, mu=10, size=10), ncol=3)
> counts2=matrix(rnbinom(300, mu=50, size=10), ncol=3)
> X1=cbind(counts1, counts2) ## these are 100 DE genes
> X2=matrix(rnbinom(11400, mu=10, size=10), ncol=6)
> X=rbind(X1,X2)
> designs=c(0,0,0,1,1,1)
> seqData=newSeqCountSet(X, designs)
> seqData
```

```
SeqCountSet (storageMode: lockedEnvironment)
assayData: 2000 features, 6 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 1 2 ... 6 (6 total)
  varLabels: designs
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

2. Estimate normalization factor.

```
> seqData=estNormFactors(seqData)
```

3. Estimate and shrink gene-wise dispersions

```
> seqData=estDispersion(seqData)
```

4. With the normalization factors and dispersions ready, the two-group comparison can be conducted via a Wald test:

```
> result=waldTest(seqData, 0, 1)
> head(result,5)
```

```
   geneIndex      muA      muB       lfc   difExpr     stats         pval
81        81 6.319184 57.16484 -2.134908 -50.84566 -5.622639 1.880625e-08
40        40 6.310148 52.92943 -2.059948 -46.61929 -5.515765 3.472668e-08
71        71 9.301676 58.20707 -1.790007 -48.90539 -5.289130 1.228993e-07
67        67 7.972264 52.10124 -1.825942 -44.12897 -5.277026 1.312971e-07
70        70 7.643482 50.44998 -1.833626 -42.80649 -5.245439 1.559105e-07
      local.fdr          fdr
81 4.053204e-05 4.053204e-05
40 4.755051e-05 4.487629e-05
71 7.974217e-05 6.319303e-05
67 8.182227e-05 6.319303e-05
70 8.824126e-05 6.319303e-05
```

2.2 Multifactor experiment

DSS provides functionalities for dispersion shrinkage for multifactor experimental designs. Downstream model fitting (through genearlized linear model) and hypothesis testing can be performed using other packages such as edgeR, with the dispersions estimated from DSS.

Below is an example, based a simple simulation, to illustrate the DE analysis of a crossed design.

1. First simulate data for a 2x2 crossed experiments. Note the counts are randomly generated.

```
> library(DSS)
> library(edgeR)
> counts=matrix(rpois(800, 10), ncol=8)
> design=data.frame(gender=c(rep("M",4), rep("F",4)), strain=rep(c("WT", "Mutant"),4))
> X=model.matrix(~gender+strain, data=design)
```

2. make SeqCountSet, then estimate size factors and dispersion

```
> seqData=newSeqCountSet(counts, as.data.frame(X))
> seqData=estNormFactors(seqData)
> seqData=estDispersion(seqData)
```

3. Using edgeR's function to do glm model fitting, but plugging in the estimated size factors and dispersion from DSS.

```
> fit.edgeR <- glmFit(counts, X, lib.size=normalizationFactor(seqData),
+               dispersion=dispersion(seqData))
```

4. Using edgeR's function to do hypothesis testing on the second parameter of the model (gender).

```
> lrt.edgeR <- glmLRT(fit.edgeR, coef=2)
> head(lrt.edgeR$table)
```

```
        logFC   logCPM          LR    PValue
1  0.10824075 21.02530 0.069594530 0.7919281
```

```
2 -0.44323598 21.36571 1.385675284 0.2391368
3  0.02464308 21.12189 0.003816895 0.9507372
4  0.24331915 21.15193 0.371876333 0.5419833
5 -0.15460358 21.41923 0.173358419 0.6771442
6  0.36023559 21.27215 0.873939976 0.3498669
```

3 Using DSS for differential methylation analysis

For BS-seq experiments, after sequence alignment and proper processing, the BS-seq data can be summarized with following information for each C position (mostly CpG sites, with the occasional CH): chromosome number, genomic coordinate, total number of reads covering the position, and number of reads showing methylation at this position. For a sample, this information needs to be saved in a simple text file, with each row representing a CpG site.

DML detection using DSS starts from several such text files. A typical DML detection contains two simple steps. Below we will use files distributed with the package to illustrate the usage of the package.

1. Load in library. Read in text files and create BSseq objects for two conditions. This step requires bsseq Bioconductor package. BSseq class is defined in that package.

```
> library(DSS)
> require(bsseq)
> path <- file.path(system.file(package="DSS"), "extdata")
> dat1.1 <- read.table(file.path(path, "cond1_1.txt"), header=TRUE)
> dat1.2 <- read.table(file.path(path, "cond1_2.txt"), header=TRUE)
> dat2.1 <- read.table(file.path(path, "cond2_1.txt"), header=TRUE)
> dat2.2 <- read.table(file.path(path, "cond2_2.txt"), header=TRUE)
> BS1 <- makeBSseqData( list(dat1.1, dat1.2), paste("cond1",1:2,sep=".") )
> BS2 <- makeBSseqData( list(dat2.1, dat2.2), paste("cond2",1:2,sep=".") )
> BS1

An object of type 'BSseq' with
  28913 methylation loci
  2 samples
has not been smoothed
```

2. Detect DML by calling callDML function.

```
> dmls <- callDML(BS1, BS2)
> head(dmls)

       chr     pos       mu1       mu2         diff    diff.se        stat
6528 chr18 3014904 0.5056818 0.4696970  0.035984848 0.23057082  0.15606853
6576 chr18 3031032 0.3400000 0.1590909  0.180909091 0.11940500  1.51508807
6577 chr18 3031044 0.3445946 0.3409091  0.003685504 0.13275390  0.02776192
```

```
6578 chr18 3031065 0.4353448 0.3723404  0.063004402 0.10426725  0.60425879
6579 chr18 3031069 0.3000000 0.5370370 -0.237037037 0.14105165 -1.68049812
6580 chr18 3031082 0.3506787 0.3950000 -0.044321267 0.07838165 -0.56545463
           pval       fdr
6528 0.87597900 1.0000000
6576 0.12975010 0.8169241
6577 0.97785203 1.0000000
6578 0.54567160 1.0000000
6579 0.09286044 0.6882329
6580 0.57176458 1.0000000
```

3. Based on the DML results, detect DMR by calling `callDMR` function.

```
> dmrs <- callDMR(dmls, p.threshold=0.001)
> head(dmrs)
```

```
      chr    start      end length nCG meanMethy1 meanMethy2 diff.Methy
77  chr18 32719368 32719501    134  13  0.9701046 0.39415782  0.5759467
102 chr18 36447232 36447337    106  10  0.8870297 0.06843556  0.8185942
124 chr18 38115895 38115970     76   4  0.6443536 0.12561822  0.5187354
```

For differentially methylated loci (DML) detection, a hypothesis test is conducted at each CpG site, and the tests are performed independently. The differentially methylated region (DMR) detection is based on the DML test results. Regions will CpG sites being statistically significant are detected as DMRs. Nearby DMRs are merged into longer ones. Some restrictions including the minimum length, minimum number of CpG sites, etc. are applied. Note that this function doesn't consider the spatial correlation among nearby CpG sites, which is an important aspect in the whole genome BS-seq data. This will be in the future development plan.

4 Session Info

```
> sessionInfo()
```

```
R version 3.1.2 (2014-10-31)
Platform: x86_64-apple-darwin10.8.0 (64-bit)

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
 [1] stats4    splines   parallel  stats     graphics  grDevices utils
 [8] datasets  methods   base

other attached packages:
```

```
 [1] bsseq_1.2.0          matrixStats_0.10.3   GenomicRanges_1.18.1
 [4] GenomeInfoDb_1.2.2   IRanges_2.0.0        S4Vectors_0.4.0
 [7] edgeR_3.8.2          limma_3.22.1         DSS_2.4.1
[10] Biobase_2.26.0       BiocGenerics_0.12.0

loaded via a namespace (and not attached):
 [1] colorspace_1.2-4  grid_3.1.2        lattice_0.20-29   locfit_1.5-9.1
 [5] munsell_0.4.2     plyr_1.8.1        R.methodsS3_1.6.1 Rcpp_0.11.3
 [9] scales_0.2.4      tools_3.1.2       XVector_0.6.0
```