derfinder: Differential expression analysis of RNA-seq data at base-pair resolution

Alyssa C. Frazee Sarven Sabunciyan Kasper D. Hansen Rafael A. Irizarry
Jeffrey T. Leek

February 13, 2013

1 Overview

DER Finder is a new method for performing differential expression analysis using RNA-seq data. Rather than testing previously-annotated features for differential expression or assembling transcripts from scratch and testing the assembled transcripts, DER Finder calculates coverage for each sample at every base pair in the genome, models expression at each base pair as a function of a covariate of interest (e.g. disease status) plus any confounders, and segments the genome into regions based on the results of these models. Regions that are candidates for being differentially expressed between conditions are then tested for differential expression. Therefore, the output of DER Finder is simply a table of genomic regions with average expression fold changes between the conditions, plus associated test statistics and p-values. Significantly differentially expressed regions are referred to as DERs.

A manuscript, with the same title as this vignette, which contains details about DER Finder is under revision in *Biostatistics*.

2 Before using derfinder: Preprocessing

Most RNA-seq experiments produce FASTQ files of reads as output. Before performing downstream statistical analysis with *derfinder*, the FASTQ files must go through two preprocessing steps: alignment and coverage calculation.

Reads must be aligned to the reference genome in a way that appropriately handles junction reads, or reads that span multiple exons. A well-established aligner that handles junction reads is TopHat [1]. Each sample's reads can be aligned with TopHat from the command line as follows:

```
tophat -o <OUTPUT DIRECTORY> <INDEX> sample.fastq
```

Users must specify an <OUTPUT DIRECTORY>, where all TopHat output files will be deposited, and a Bowtie <INDEX>. Bowtie indexes may be downloaded from the TopHat website [2]. The TopHat manual [3] also details other options available for this alignment step.

After alignment, the number of reads overlapping each base-pair in the genome must be determined. We provide a python script, `countReads.py`, that produces base-pair-level coverage for each chromosome for each sample. The script can be used as follows:

```
python countReads.py --file accepted_hits.bam --output counts.txt --kmer 101 --chrom 'Y'
```

In the example command, `accepted_hits.bam` is the alignment file produced by TopHat, `counts.txt` is the output file (a two-column, tab-delimited text file, where the first column is genomic position and the second column is coverage), 101 is the read length, and `Y` is the chromosome. (Chromosome names must be specified the same way they are in the Bowtie index used in alignment).

Finally, the `counts.txt` files from each sample must be appropriately merged to create a base-pair-by-sample coverage matrix. In this vignette, we will assume that the final merged file is called `allcounts.txt`. A very small (1000bp) `allcounts.txt` is provided with *derfinder*, for illustrative purposes.

3 Database Creation

To begin using *derfinder*, we dump the base-pair-by-sample count matrix into a SQLite database in order to avoid reading this (usually very large) matrix into R's memory. After placing the `allcounts.txt` file in your working directory, this can be done from within R using the `makeDb` function:

```
> library(derfinder)
> makeDb(dbfile = "allcounts.db", textfile = "allcounts.txt", tablename = "chrY")
```

```
[1] "Wrote database file allcounts.db containing table chrY"
```

The resulting database, `allcounts.db`, contains a table called "chrY" which contains a filtered version of `allcounts.txt`, meaning that all rows without at least one coverage value of 5 or greater have not been put into the database. Users can explore the data using the *Genominator* package [4]:

```
> library(Genominator)
> dat = ExpData(dbFilename = "allcounts.db", tablename = "chrY")
> head(dat)
```

```
       pos orbFrontalF1 orbFrontalF11 orbFrontalF2 orbFrontalF23 orbFrontalF3
1 15016742            0             2            0             4            0
2 15016743            0             2            0             1            0
3 15016744            0             2            0             0            0
4 15016745            0             2            1             5            0
5 15016746            0             2            0             3            0
6 15016747            0             2            1             2            0
  orbFrontalF32 orbFrontalF33 orbFrontalF40 orbFrontalF42 orbFrontalF43 orbFrontalF47
1             0             2             0             0             0             0
2             0             1             5             0             0             0
3             0             5             3             0             0             0
4             0             1             5             0             0             0
5             0             3             1             0             0             0
6             0             5             0             0             0             0
  orbFrontalF53 orbFrontalF55 orbFrontalF56 orbFrontalF58
1            10             1             2             0
2            10             0             2             0
3            10             2             2             0
4            10             1             2             0
5            10             1             1             0
6            10             3             3             0
```

```
> getColnames(dat)
```

```
 [1] "pos"           "orbFrontalF1"  "orbFrontalF11" "orbFrontalF2"  "orbFrontalF23"
 [6] "orbFrontalF3"  "orbFrontalF32" "orbFrontalF33" "orbFrontalF40" "orbFrontalF42"
[11] "orbFrontalF43" "orbFrontalF47" "orbFrontalF53" "orbFrontalF55" "orbFrontalF56"
[16] "orbFrontalF58"
```

```
> length(dat[,1]$pos)
```

```
[1] 407
```

4 Statistical Analysis at Base-Pair Level

DER Finder analyzes differential expression by calculating a test statistic from a linear model fit at each base-pair. The example shown here is a simple case: the model contains only two covariates, sex (the covariate of interest) and library size, defined as the median nonzero count for each sample. The statistic of interest from this model is a moderated t statistic, defined as the fitted coefficient for sex divided by its posterior standard error. To stabilize the estimate of the coefficient variance across the base-pairs, the estimated variances are shrunk toward their prior. This methodology - both efficient fitting of millions of linear models and the moderated t-statistic shrinkage approach - has been described and implemented in the *limma* Bioconductor package [5,6]. DER Finder makes use of *limma* utilities to do model fitting:

```
> ## define the covariate:
> sex = c(1,1,0,0,1,1,0,0,1,1,1,1,0,0,1)
> ## first pass at linear model:
> limma.input = getLimmaInput(dbfile = "allcounts.db", tablename = "chrY", group = sex, nonzero = TRUE)
> ## extract genomic positions at which model was fit:
> pos = limma.input$pos
> ## calculate moderated t-stats and fold changes
> ## allowing for trend on prior variances across bps
> tstats = getTstats(fit = limma.input$ebobject, trend=TRUE)
> tt = tstats$tt
> logfchange = tstats$logfchange
```

5 Hidden Markov Model

Based on the moderated t statistics from the linear models, we would like to analyze the differential expression signal at each base-pair and group contiguous base-pairs with similar signals into regions. This segmentation is accomplished with a Hidden Markov Model (HMM), and in this example, the HMM has four hidden states: not expressed, equally expressed, overexpressed in cases, and underexpressed in cases. The output will be a list of regions, including each region's average t-statistic and fold-change.

```
> myparams = getParams(tt)
> regions = getRegions(method = "HMM", chromosome = "Y", pos = pos, tstats = tt,
+   stateprobs = myparams$stateprobs, params = myparams$params,
+   includet = TRUE, includefchange = TRUE, fchange = logfchange)
> head(regions$states)

  chr    start      end state length    mean.t mean.fold.change
1   Y 15016742 15016856     2    115 0.2615291         1.061176
2   Y 15016857 15016957     3    101 4.2888343         1.926871
3   Y 15016958 15016990     2     33 1.4512825         1.204365
4   Y 15016991 15017067     1     77 0.0000000         1.000000
5   Y 15017068 15017225     2    158 0.9366970         1.037204

> ders = subset(regions$states, state==3|state==4)
> head(ders)

  chr    start      end state length   mean.t mean.fold.change
2   Y 15016857 15016957     3    101 4.288834         1.926871
```

In this small example, the base-pairs analyzed were grouped into five regions: three regions we classify as equally expressed between men and women, one region of no expression, and one 101-bp region that appears to be overexpressed in men.

6 Statistical Significance

We can attach statistical significance to regions using a permutation test. Because this example is so small, we run the test using 1000 permutations, but in analyzing a whole chromosome, we recommend using far fewer (e.g., 10).

```
> pvals = get.pvals(regions = regions$states, dbfile = "allcounts.db",
+   tablename = "chrY", num.perms = 1000, group = sex, est.params = myparams,
+   chromosome = "Y")
> # takes several minutes
> reg.withp = data.frame(regions$states, pvals = pvals)
> reg.withp

  chr    start      end state length    mean.t mean.fold.change        pvals
1   Y 15016742 15016856     2    115 0.2615291         1.061176           NA
2   Y 15016857 15016957     3    101 4.2888343         1.926871 0.0007955449
3   Y 15016958 15016990     2     33 1.4512825         1.204365           NA
4   Y 15016991 15017067     1     77 0.0000000         1.000000           NA
5   Y 15017068 15017225     2    158 0.9366970         1.037204           NA
```

The discovered region is highly significant, since its permutation p-value was 0.0008. In this case, the test was only performed on a single region, so no multiple-testing adjustment is needed. However, in most analyses, several genomic regions will be tested for significance. We recommend converting the p-values from these experiments to q-values [7] to determine regions' significance; using a cutoff of any specific q-value controls the false discovery rate at that level. P-values can be converted to q-values using R's built-in `p.adjust`:

```
> q = p.adjust(reg.withp$pvals, method="fdr")
```

7 Annotating Regions

We can use the `getAnnotation` function to download a data frame with information about annotated exons, and we can use this exon information to find the closest exon and/or gene to our region of interest:

```
> exons = getAnnotation("hg19","knownGene")
```

```
TranscriptDb object:
| Db type: TranscriptDb
| Supporting package: GenomicFeatures
| Data source: UCSC
| Genome: hg19
| Genus and Species: Homo sapiens
| UCSC Table: knownGene
| Resource URL: http://genome.ucsc.edu/
| Type of Gene ID: Entrez Gene ID
| Full dataset: yes
| miRBase build ID: NA
| transcript_nrow: 80922
| exon_nrow: 286852
| cds_nrow: 235842
| Db created by: GenomicFeatures package from Bioconductor
| Creation time: 2013-02-13 14:17:34 -0500 (Wed, 13 Feb 2013)
| GenomicFeatures version at creation time: 1.10.1
| RSQLite version at creation time: 0.11.2
| DBSCHEMAVERSION: 1.0
[1] "Labeling exons by gene..."
```

```
> Yexons = subset(exons,chr=="chrY")
> getExons(region = c("chrY",15016857,15016957), annotation = Yexons)
```

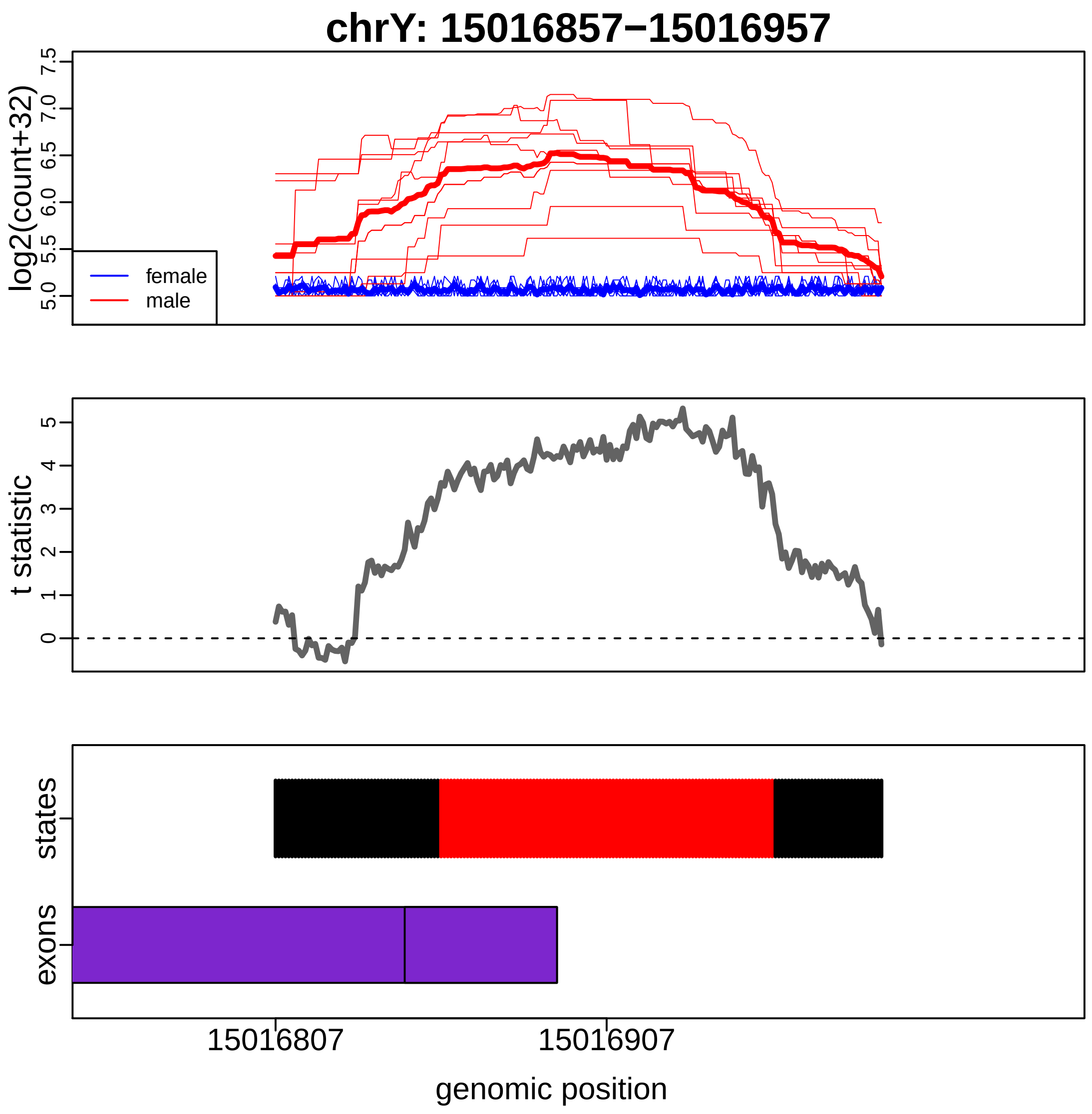
```
[1] "region overlaps annotated exon(s)"
$region
[1] "chrY"     "15016857" "15016957"

$closestExons
       gene  chr    start      end width strand exon_id
245743 8653 chrY 15016699 15016892   194      +  273990
245744 8653 chrY 15016846 15016892    47      +  273991
```

8 Visualizing Results

DER Finder also enables users to make plots of regions of interest, showing the raw data (top panel), moderated t statistics for each base-pair (middle panel), and HMM state predictions and locations of exons overlapping the region (bottom panel).

```
> plotRegion(regions, ind=2, tstats=tt, pos=pos, annotation=Yexons,
+   counts="allcounts.db", group=ifelse(sex==1,"male","female"),
+   tabname="chrY", chromosome="chrY", scalefac=32, ylim=c(4.8, 7.5))
```



9 References

[1] Trapnell C, Pachter L, and Salzberg S (2009). "Tophat: discovering splice junctions with RNA-Seq". *Bioinformatics* 25(9): 1105-1111.

[2] `http://tophat.cbcb.umd.edu/igenomes.html`

[3] `http://tophat.cbcb.umd.edu/manual.html`

[4] Bullard J, Purdom E, Hansen KD, and Dudoit S (2010). "Evaluation of Statistical Methods for Normalization and Differential Expression in mRNA-seq Experiments". *BMC Bioinformatics*, 11(94). R Package version 1.12.0.

[5] Smyth, GK et al (2004). "Linear models and empirical bayes methods for assessing differential expression in microarray experiments". *Statistical Applications in Genetics and Molecular Biology*, 3(1): 3.

[6] Smyth, GK (2005). "Limma: linear models for microarray data". Bioinformatics and Computational Biology Solutions using R and Bioconductor: 397-420.

[7] Storey, JD (2002). "A direct approach to false discovery rates". *Journal of the Royal Statistical Society, Series B (Methodological)*, 64 (3): 479-498.