Data preprocessing and creation of the data objects pasillaGenes and pasillaExons

Alejandro Reyes

April 4, 2013

Abstract

This vignette describes the steps that were followed for the generation of the data objects contained in the package *pasilla*.

Contents

1 Downloading the files

We used the RNA-Seq data from the publication by Brooks et al.˜ [1]. The experiment investigated the effect of siRNA knock-down of pasilla, a gene that is known to bind to mRNA in the spliceosome, and which is thought to be involved in the regulation of splicing. The data set contains 3 biological replicates of the knockdown as well as 4 biological replicates for the untreated control. Data files are publicly available in the NCBI Gene Expression Omnibus under the accession GSE18508[1]. The read sequences in FASTQ format were extracted from the NCBI short read archive file (.sra files), using the sra toolkit[2].

2 Read alignment and filtering

The reads in the FASTQ files were aligned using tophat version 1.2.0 with default parameters against the reference Drosophila melanogaster genome. Table˜1 summarizes the read number and alignment statistics.

[1]`http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE18508`

[2]`http://www.ncbi.nlm.nih.gov/books/NBK47540/#SRA_Download_Guid_B.5_Converting_SRA_for`

	file	type	number of lanes	total number of reads	exon counts
1	treated1fb	single-read	5	35158667	15679615
2	treated2fb	paired-end	2	12242535 (x2)	15620018
3	treated3fb	paired-end	2	12443664 (x2)	12733865
4	untreated1fb	single-read	2	17812866	14924838
5	untreated2fb	single-read	6	34284521	20764558
6	untreated3fb	paired-end	2	10542625 (x2)	10283129
7	untreated4fb	paired-end	2	12214974 (x 2)	11653031

Table 1: Read numbers and alignment statistics. The column *exon counts* refers to the number of reads that could be uniquely aligned to an exon.

The reference genome fasta files were obtained from the Ensembl ftp server[3]. We ran `bowtie-build` to index the fasta file. For more information on this procedure see the bowtie webpage[4]. The indexed form is required by bowtie, and thus tophat.

```
wget ftp://ftp.ensembl.org/pub/release-62/fasta/drosophila_melanogaster/ \
dna/Drosophila_melanogaster.BDGP5.25.62.dna_rm.toplevel.fa.gz

gunzip Drosophila_melanogaster.BDGP5.25.62.dna_rm.toplevel.fa.gz
bowtie-build Drosophila_melanogaster.BDGP5.25.62.dna_rm.toplevel.fa \
    d_melanogaster_BDGP5.25.62
```

We generated the alignment BAM file using tophat. For the single-reads data:

```
tophat bowtie_index reads1.fastq,reads2.fastq,...,readsN.fastq
```

For the paired-end data:

```
tophat -r inner-fragment-size bowtie_index \
    reads1_1.fastq,reads2_1.fastq,...,readsN_1.fastq \
    reads1_2.fastq,reads2_2.fastq,...,readsN_2.fastq
```

More information on tophat is provided on its webpage[5]. The SAM alignment files from which *pasilla* was generated are available at `http://www-huber.embl.de/pub/DEXSeq/analysis/brooksetal/bam/`.

3 Exon count files

To generate the per-exon read counts, we first needed to define the exonic regions. To this end, we downloaded the file Drosophila_melanogaster.BDGP5.25.62.gtf.gz from Ensembl[6]. The script `dexseq_prepare_annotation.py` contained in the *DEXSeq* package was used to extract the exons of the transcripts from the file, define new non-overlapping exonic regions and reformat it to create the file Dmel.BDGP5.25.62.DEXSeq.chr.gff contained in pasilla/extdata. For example, for this file we ran:

```
wget ftp://ftp.ensembl.org/pub/release-62/gtf/ \
drosophila_melanogaster/Drosophila_melanogaster.BDGP5.25.62.gtf.gz
```

[3]`http://www.ensembl.org/info/data/ftp/index.html`
[4]`http://bowtie-bio.sourceforge.net/tutorial.shtml`
[5]`http://tophat.cbcb.umd.edu/tutorial.html`
[6]`ftp://ftp.ensembl.org/pub/release-62/gtf/drosophila_melanogaster`

```
gunzip Drosophila_melanogaster.BDGP5.25.62.gtf.gz
python dexseq_prepare_annotation.py Drosophila_melanogaster.BDGP5.25.62.gtf \
    Dmel.BDGP5.25.62.DEXSeq.chr.gff
```

To count the reads that fell into each non-overlapping exonic part, the script `dexseq_count.py`, which is also contained in the *DEXSeq* package, was used. It took the alignment results in the form of a SAM file (sorted by position in the case of a paired end data) and the `gtf` file Dmel.BDGP5.25.62.DEXSeq.chr.gff and returned one file for each biological replicate with the exon counts. For example, for the file treated1.bam, which contained single-end alignments, we ran:

```
samtools index treated1.bam
samtools view treated1.bam > treated1.sam
python dexseq_count.py Dmel.BDGP5.25.62.DEXSeq.chr.gff \
    treated1.sam treated1fb.txt
```

For the file treated2.bam, which contained paired-end alignments:

```
samtools index treated2.bam
samtools view treated2.bam > treated2.sam
sort -k1,1 -k2,2n treated2.sam > treated2_sorted.sam
python dexseq_count.py -p yes Dmel.BDGP5.25.62.DEXSeq.chr.gff \
    treated2_sorted.sam treated2fb.txt
```

The output of the two HTSeq python scripts is provided in the *pasilla* package:

```
> library("pasilla")

> inDir = system.file("extdata", package="pasilla", mustWork=TRUE)
> dir(inDir)

 [1] "Dmel.BDGP5.25.62.DEXSeq.chr.gff" "geneIDsinsubset.txt"
 [3] "pasilla_gene_counts.tsv"         "treated1fb.txt"
 [5] "treated2fb.txt"                  "treated3fb.txt"
 [7] "untreated1fb.txt"                "untreated2fb.txt"
 [9] "untreated3fb.txt"                "untreated4fb.txt"
```

The Python scripts are built upon the HTSeq library[7].

4 Creation of the *ExonCountSet* `pasillaExons`

To create an *ExonCountSet* object, we started with a data frame `samples` that contained the sample annotations, as in Table~1.

```
> head(samples)

             condition        type
treated1fb     treated single-read
treated2fb     treated  paired-end
treated3fb     treated  paired-end
```

[7]`http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html`

```
untreated1fb untreated single-read
untreated2fb untreated single-read
untreated3fb untreated  paired-end
```

We also needed the annotation file with the per exon annotation.

```
> annotationfile = file.path(inDir, "Dmel.BDGP5.25.62.DEXSeq.chr.gff")
```

With these, we could call the function `read.HTSeqCounts` to construct the object `ecs`.

```
> library("DEXSeq")
> ecs = read.HTSeqCounts(countfiles = file.path(inDir, paste(rownames(samples), "txt", sep=".")),
+            design = samples,
+            flattenedfile = annotationfile)
> sampleNames(ecs) = rownames(samples)
```

We only wanted to work with data from a subset of genes, which was defined in the following file.

```
> genesforsubset = readLines(file.path(inDir, "geneIDsinsubset.txt"))
> pasillaExons = subsetByGenes(ecs, genes=genesforsubset)
```

We added the experiment data:

```
> expdata = new("MIAME",
+    name="pasilla knockdown",
+    lab="Genetics and Developmental Biology, University of Connecticut Health Center",
+    contact="Dr. Brenton Graveley",
+    title="modENCODE Drosophila pasilla RNA Binding Protein RNAi knockdown RNA-Seq Studies",
+    url="http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE18508",
+    abstract="RNA-seq of 3 biological replicates of from the Drosophila melanogaster
+       S2-DRSC cells that have been RNAi depleted of mRNAs encoding pasilla, a mRNA binding
+       protein and 4 biological replicates of the the untreated cell line.")
>    pubMedIds(expdata) <- "20921232"
> experimentData(pasillaExons) <- expdata
```

5 Creation of the *CountDataSet* `pasillaGenes`

The *CountDataSet* class is analogous to the *ExonCountSet* class; the latter is specifically designed to store exon level counts, while the *CountDataSet* class is useful more generally for whatever one wishes to count (e. g. ChIP peaks, gene levels counts). We made use of the function `geneCountTable` from the package *DEXSeq* to get a data frame containing the number of reads falling on each of the genes. We used the function `newCountDataSet` to create the object `pasillaGenes`.

```
> library("DESeq")
> genetable = geneCountTable(ecs)
> pasillaGenes = newCountDataSet(genetable,
+   conditions = samples)
> experimentData(pasillaGenes) = expdata
```

We saved the objects in the data directory of the package:

```
> save(pasillaExons, file=file.path("..", "..", "data", "pasillaExons.RData"))
> save(pasillaGenes, file=file.path("..", "..", "data", "pasillaGenes.RData"))
> toLatex(sessionInfo())
```

- R version 3.0.0 (2013-04-03), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=C`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase~2.20.0, BiocGenerics~0.6.0, DESeq~1.12.0, DEXSeq~1.6.0, lattice~0.20-15, locfit~1.5-9, pasilla~0.2.16, xtable~1.7-1
- Loaded via a namespace (and not attached): AnnotationDbi~1.22.0, Biostrings~2.28.0, DBI~0.2-5, GenomicRanges~1.12.0, IRanges~1.18.0, RColorBrewer~1.0-5, RCurl~1.95-4.1, RSQLite~0.11.2, Rsamtools~1.12.0, XML~3.96-1.1, annotate~1.38.0, biomaRt~2.16.0, bitops~1.0-5, genefilter~1.42.0, geneplotter~1.38.0, grid~3.0.0, hwriter~1.3, splines~3.0.0, statmod~1.4.17, stats4~3.0.0, stringr~0.6.2, survival~2.37-4, tools~3.0.0, zlibbioc~1.6.0

Table 2: The output of `sessionInfo` on the build system after running this vignette.

References

[1] A.~N. Brooks, L.~Yang, M.~O. Duff, K.~D. Hansen, J.~W. Park, S.~Dudoit, S.~E. Brenner, and B.~R. Graveley. Conservation of an RNA regulatory map between Drosophila and mammals. *Genome Research*, pages 193–202, October 2010.