Generation of transcript counts from pasilla dataset with kallisto

Malgorzata Nowicka[*], Mark Robinson

October 19, 2016

This vignette describes version 1.2.0 of the *PasillaTranscriptExpr* package.

Contents

1 Description of pasilla dataset

The *pasilla* dataset was produced by Brooks et al. [1]. The aim of their study was to identify exons that are regulated by pasilla protein, the Drosophila melanogaster ortholog of mammalian NOVA1 and NOVA2 (well studied splicing factors). In their RNA-seq experiment, the libraries were prepared from 7 biologically independent samples: 4 control samples and 3 samples in which pasilla was knocked-down. The libraries were sequenced on the Illumina Genome Analyzer II using single-end and paired-end sequencing and different read lengths. The RNA-seq data can be downloaded from the NCBI's Gene Expression Omnibus (GEO) under the accession number GSE18508.

[*] gosia.nowicka@uzh.ch

2 Required software

This work-flow can be run on a Unix-like operating system, i.e., Linux or MacOS X with bash shell. All commands, including the one that could be run from terminal window, are run from within *R* using `system()` function. The downloaded and generated files will be saved in the current working directory.

Brooks et al. deposited their data in the Short Read Archive. In order to convert SRA data into fastq format, you need to install the *SRA toolkit* available on http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software.

For the transcript quantification, we use *kallisto* version 0.42.1 [2], which is an extremely fast program that quantifies abundances of transcripts. *kallisto* is based on the novel idea of pseudoalignment to rapidly determine the compatibility of reads with transcripts, without the need for alignment. Thus it works directly on fastq files. The quantification is available in transcripts per million (TPM) and in expected counts. In this package, we make available the expected counts. *kallisto* can be downloaded from http://pachterlab.github.io/kallisto/.

3 Downloading the pasilla data

We use an automated process to download the SRA files that correspond to 4 control (Untreated) samples and 3 pasilla knocked-down (CG8144_RNAi) samples. All the information about the pasilla assay can be found in the metadata file *SraRunInfo.csv*, which can be downloaded from http://www.ncbi.nlm.nih.gov/sra?term=SRP001537 under *Send to:* → *File* → *RunInfo* → *Create File*. The same file is also available within this package in the `extdata` directory.

```
library(PasillaTranscriptExpr)

data_dir <- system.file("extdata", package = "PasillaTranscriptExpr")

sri <- read.table(paste0(data_dir, "/SraRunInfo.csv"), stringsAsFactors = FALSE,
  sep = ",", header = TRUE)
keep <- grep("CG8144|Untreated-", sri$LibraryName)
sri <- sri[keep, ]
```

```
sra_files <- basename(sri$download_path)

for(i in 1:nrow(sri))
  download.file(sri$download_path[i], sra_files[i])
```

To convert the SRA files to fastq format, we use the *fastq-dump* command from the *SRA toolkit*. Then, we compress the fastq files.

```
cmd <- paste0("fastq-dump -O ./ --split-3 ", sra_files)

for(i in 1:length(cmd))
  system(cmd[i])

system("gzip *.fastq")
```

4 Downloading the reference genome

To run *kallisto*, you need to download a FASTA formatted file of target sequences:

```
system("wget ftp://ftp.ensembl.org/pub/release-70/fasta/drosophila_melanogaster/cdna/Drosophila_m
system("gunzip Drosophila_melanogaster.BDGP5.70.cdna.all.fa.gz")
```

The output produced by *kallisto* contains only the transcript IDs. To add the corresponding gene IDs, we need to download the gene model annotation in GTF format:

```
system("wget ftp://ftp.ensembl.org/pub/release-70/gtf/drosophila_melanogaster/Drosophila_melanoga
system("gunzip Drosophila_melanogaster.BDGP5.70.gtf.gz")
```

5 Transcript quantification with kallisto

We create a metadata file where each row corresponds to a collection of information needed for a single call of *kallisto*. The *pasilla* data consists of paired-end and single-end samples. When you run *kallisto* on single-end reads, you have to specify an `-l` option which defines the average fragment length. It can be found in `sri$avgLength`. There is one sample (GSM461179) which was sequenced using different read lengths. Therefore, for this sample, we do the transcript quantification for each read length separately and we add the resulting transcript counts in another step.

```
sri$LibraryName <- gsub("S2_DRSC_","",sri$LibraryName)
metadata <- unique(sri[,c("LibraryName", "LibraryLayout", "SampleName",
  "avgLength")])

for(i in seq_len(nrow(metadata))){
  indx <- sri$LibraryName == metadata$LibraryName[i]

  if(metadata$LibraryLayout[i] == "PAIRED"){
    metadata$fastq[i] <- paste0(sri$Run[indx], "_1.fastq.gz ",
      sri$Run[indx], "_2.fastq.gz", collapse = " ")
  }else{
    metadata$fastq[i] <- paste0(sri$Run[indx], ".fastq.gz", collapse = " ")
  }
}

metadata$condition <- ifelse(grepl("CG8144_RNAi", metadata$LibraryName),
  "KD", "CTL")
metadata$UniqueName <- paste0(1:nrow(metadata), "_", metadata$SampleName)
```

In the first step of *kallisto* work-flow, we build an index with *kallisto index*:

```
cDNA_fasta <- "Drosophila_melanogaster.BDGP5.70.cdna.all.fa"
index <- "Drosophila_melanogaster.BDGP5.70.cdna.all.idx"

cmd <- paste("kallisto index -i", index, cDNA_fasta, sep = " ")
cmd
```

```
## [1] "kallisto index -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx Drosophila_melanogaster.B
```

```
system(cmd)
```

The quantification is done with *kallisto quant* command:

```
out_dir <- metadata$UniqueName

cmd <- paste("kallisto quant -i", index, "-o", out_dir, "-b 0 -t 5",
  ifelse(metadata$LibraryLayout == "SINGLE",
    paste("--single -l", metadata$avgLength), ""), metadata$fastq)
cmd
## [1] "kallisto quant -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx -o 1_GSM461176 -b 0 -t 5
## [2] "kallisto quant -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx -o 2_GSM461177 -b 0 -t 5
## [3] "kallisto quant -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx -o 3_GSM461178 -b 0 -t 5
## [4] "kallisto quant -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx -o 4_GSM461179 -b 0 -t 5
## [5] "kallisto quant -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx -o 5_GSM461179 -b 0 -t 5
## [6] "kallisto quant -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx -o 6_GSM461179 -b 0 -t 5
## [7] "kallisto quant -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx -o 7_GSM461180 -b 0 -t 5
## [8] "kallisto quant -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx -o 8_GSM461181 -b 0 -t 5
## [9] "kallisto quant -i Drosophila_melanogaster.BDGP5.70.cdna.all.idx -o 9_GSM461182 -b 0 -t 5
```

```
for(i in 1:length(cmd))
  system(cmd[i])
```

We want to add the gene information and merge the expected transcript counts from different samples into one table.

```
library(rtracklayer)
```

```
gtf_dir <- "Drosophila_melanogaster.BDGP5.70.gtf"

gtf <- import(gtf_dir)

gt <- unique(mcols(gtf)[, c("gene_id", "transcript_id")])
rownames(gt) <- gt$transcript_id
```

```
samples <- unique(metadata$SampleName)

counts_list <- lapply(1:length(samples), function(i){
  indx <- which(metadata$SampleName == samples[i])

  if(length(indx) == 1){
    abundance <- read.table(file.path(metadata$UniqueName[indx],
      "abundance.txt"), header = TRUE, sep = "\t", as.is = TRUE)
  }else{
    abundance <- lapply(indx, function(j){
      abundance_tmp <- read.table(file.path(metadata$UniqueName[j],
        "abundance.txt"), header = TRUE, sep = "\t", as.is = TRUE)
      abundance_tmp <- abundance_tmp[, c("target_id", "est_counts")]
      abundance_tmp
    })
    abundance <- Reduce(function(...) merge(..., by = "target_id", all = TRUE,
      sort = FALSE), abundance)
    est_counts <- rowSums(abundance[, -1])
    abundance <- data.frame(target_id = abundance$target_id,
      est_counts = est_counts, stringsAsFactors = FALSE)
  }

  counts <- data.frame(abundance$target_id, abundance$est_counts,
    stringsAsFactors = FALSE)
  colnames(counts) <- c("feature_id", samples[i])
  return(counts)
})

counts <- Reduce(function(...) merge(..., by = "feature_id", all = TRUE,
  sort = FALSE), counts_list)

### Add gene IDs
counts$gene_id <- gt[counts$feature_id, "gene_id"]
```

At the end, we keep only the unique samples in our metadata file.

```
metadata <- unique(metadata[, c("LibraryName", "LibraryLayout", "SampleName",
  "condition")])
metadata
##       LibraryName LibraryLayout SampleName condition
## 156   Untreated-1        SINGLE  GSM461176       CTL
## 162   Untreated-3        PAIRED  GSM461177       CTL
## 164   Untreated-4        PAIRED  GSM461178       CTL
## 166 CG8144_RNAi-1        SINGLE  GSM461179        KD
## 172 CG8144_RNAi-3        PAIRED  GSM461180        KD
## 174 CG8144_RNAi-4        PAIRED  GSM461181        KD
## 176   Untreated-6        SINGLE  GSM461182       CTL
```

```
write.table(metadata, "metadata.txt", quote = FALSE, sep = "\t",
  row.names = FALSE)
```

```
### Final counts with columns sorted as in metadata
counts <- counts[, c("feature_id", "gene_id", metadata$SampleName)]
write.table(counts, "counts.txt", quote = FALSE, sep = "\t", row.names = FALSE)
```

APPENDIX

A Session information

```
sessionInfo()
## R version 3.3.1 (2016-06-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.1 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets  methods
## [9] base
##
## other attached packages:
## [1] rtracklayer_1.34.0          GenomicRanges_1.26.0        GenomeInfoDb_1.10.0
## [4] IRanges_2.8.0               S4Vectors_0.12.0            BiocGenerics_0.20.0
## [7] PasillaTranscriptExpr_1.2.0 knitr_1.14
##
## loaded via a namespace (and not attached):
##  [1] XVector_0.14.0             magrittr_1.5               zlibbioc_1.20.0
##  [4] GenomicAlignments_1.10.0   BiocParallel_1.8.0         lattice_0.20-34
##  [7] stringr_1.1.0              highr_0.6                  tools_3.3.1
## [10] grid_3.3.1                 SummarizedExperiment_1.4.0 Biobase_2.34.0
## [13] digest_0.6.10              Matrix_1.2-7.1             formatR_1.4
## [16] bitops_1.0-6               codetools_0.2-15           RCurl_1.95-4.8
## [19] evaluate_0.10              stringi_1.1.2              Biostrings_2.42.0
## [22] Rsamtools_1.26.0           XML_3.98-1.4               BiocStyle_2.2.0
```

B References

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*, vol. 21, no. 2, pp. 193–202, 2011.

[2] N. L. Bray, H. Pimentel, P. Melsted, and L. Pachter, "Near-optimal RNA-Seq quantification.".