Process Single Cell RNA-Seq reads using *scuff*

Zhe Wang

2017-10-18

`scuff` is a tool for processing single cell RNA-seq fastq reads. It does demultiplexing, alignment, umi filtering, and transcript counting in an automated fashion and outputs expression table. This vignette provides a brief introduction to the `scuff` package by walking through the demultiplexing, alignment, and UMI-counting of a built-in example dataset.

Quick Start

```
# run scuff on example dataset
# NOTE: Requires index and TxDb objects for reference genome.
# For generation of these files, please refer to Stepwise Tutorial.
expr <- scuff(exampleannot,
              examplebc,
              "GRCh38_MT",
              "GRCh38_MT.gtf")
```

Stepwise Tutorial

Load Example Dataset

The `scuff` package contains a example single cell RNA-seq dataset stored as `ShortReadQ` objects. They can be accessed via variable `examplefastq` and exported to local directory as fastq.gz files.

```
library(scuff)

# write fastg.gz files to working directory
for (i in seq_len(length(examplefastq))) {
  writeFastq(examplefastq[[i]], names(examplefastq)[i], mode = "w")
}
```

Demultiplex and Assign Cell Specific Reads

Now the fastq files are saved in current working directory and are ready to be demultiplexed. `scuff` package provides built-in corresponding sample annotating table `exampleannot` and barcodes `examplebc` for processing the example dataset. In the example fastq files, the barcode sequence of each read starts at base 6 and ends at base 11. The UMI sequence starts at base 1 and ends at base 5. They can be set via `bc.pos` and `umi.pos` parameters. By default, reads with any nucleotide in the barcode and UMI sequences with sequencing quality lower than 10 (Phred score) will be excluded. The following command demultiplexes the example fastq reads and trims reads longer than 50 nucleotides. The command returns a summary `data.table` containing the sample barcodes, filenames, number of reads, ID, dir, etc.

```
demultiplex.res <- demultiplex(
  exampleannot,
  examplebc,
```

```
  bc.pos = c(6, 11),
  umi.pos = c(1, 5),
  keep = 50,
  bc.qual = 10,
  out.dir = "../Demultiplex",
  cores = 2,
  verbose = TRUE,
  overwrite = TRUE
  )
```

Alignment

By default, the cell specific fastq files are stored in `../Demultiplex` folder. `align.rsubread` is a wrapper function to `align` in `Rsubread` package. It aligns the reads to reference genome index and outputs sequence alignment map files. For simplicity, the built-in mitochondrial DNA sequence from GRCh38 reference assembly `GRCh38_MT` will be used for mapping the reads. First, a `Rsubread` index for the reference sequence needs to be generated.

```
ref <- "Homo_sapiens.GRCh38.dna.chromosome.MT.fa"
index <- "GRCh38_MT"

library(data.table)
# write reference sequence to "./Homo_sapiens.GRCh38.dna.chromosome.MT.fa".
fwrite(GRCh38_MT, ref, quote = FALSE, sep = "\t")

# NOTE: Rsubread package does not support Windows environment.
library(Rsubread)
# Create index for GRCh38_MT. For details, please refer to Rsubread user manual.
buildindex(
  basename = index,
  reference = ref,
  indexSplit = FALSE,
  memory = 16000
  )
```

The following command maps the fastq files to `GRCh38_MT` and returns the directories of the generated sequence alignment map files.

```
# get the directories to fastq files
fastq.dir <- demultiplex.res[!(is.na(cell_num)), dir]

# map the reads
align.res <- align.rsubread(
  fastq.dir,
  index,
  format = "BAM",
  out.dir = "../Alignment",
  cores = 16,
  summary.prefix = "alignment",
  overwrite = TRUE,
  verbose = TRUE
  )
```

UMI Filtering and Generation of Expression Table

Example GTF file `GRCh38_MT_gtf` will be used for feature counting. Currently, `scuff` applies the union counting mode of the HTSeq Python package. The following command generates the expression table for the example dataset.

```
gtf.file <- "GRCh38_MT.gtf"
fwrite(GRCh38_MT_gtf, gtf.file, sep = "\t", col.names = FALSE)

expr = count.umi(
  align.res,
  gtf.file,
  format = "BAM",
  out.dir = "../Count",
  cores = 2,
  output.prefix = "countUMI",
  verbose = TRUE
  )
```