Supplementary Material

Zheng Wei, Wei Zhang, Huan Fang, Yanda Li and Xiaowo Wang

2018-02-15

Contents

1 Package Installation and Loading

1.1 Download and Installation

The following code will download and install the current version of esATAC from Bioconductor (version>=3.6).

```
source("http://www.bioconductor.org/biocLite.R")
biocLite("esATAC")
```

```
## Bioconductor version 3.6 (BiocInstaller 1.28.0), ?biocLite for help
```

For more detailed installation instruction on Linux, Windows and MAC OS, visit: https://github.com/wzthu/esATAC/wiki/esATAC-Installation-Tutorial.

1.2 Loading

Just like other R packages, esATAC has to be loaded each time before using the package.

```
library(esATAC)
```

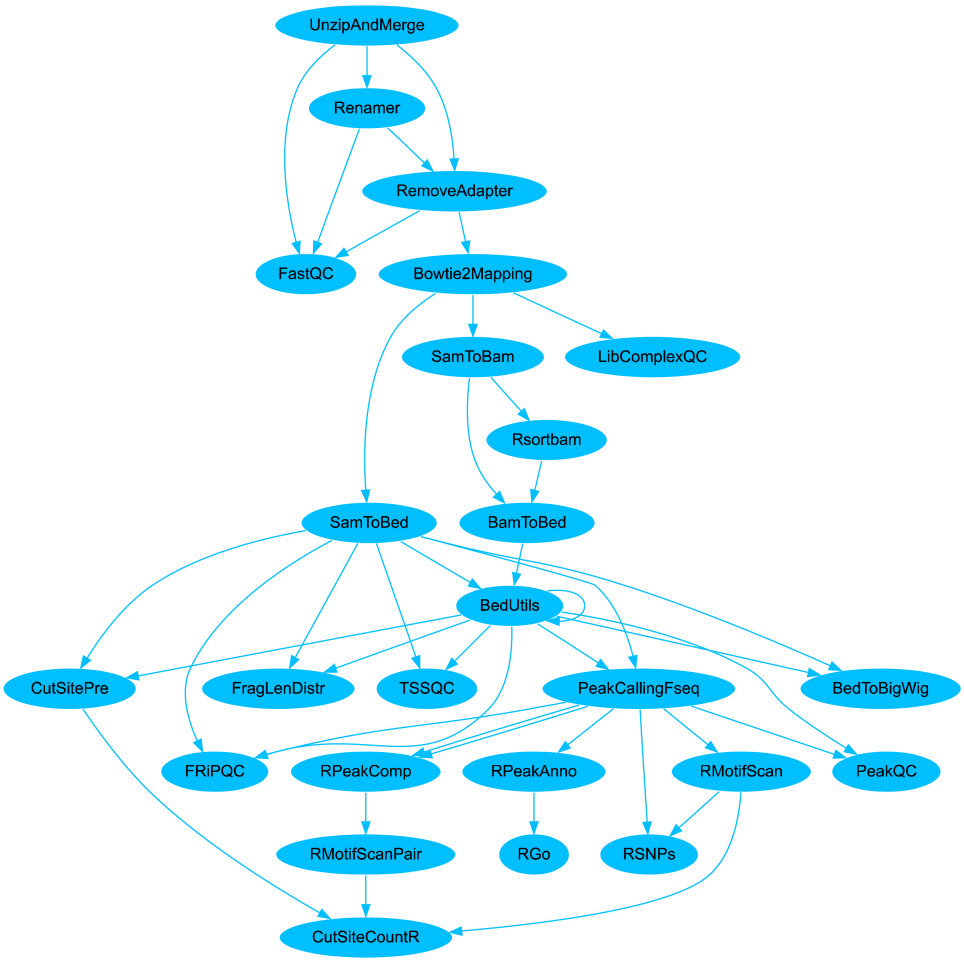

Figure S1: Pipeline flowchart

2 Flowchart and Functional Summary

2.1 Flowchart

Relationship of functional modules in esATAC can be visualized. Users can print flowchart like this:

```
printMap()
```

The result is shown in Figure S1. Both the preset pipeline and customized pipeline follow this flowchart. Parameters can be passed from upstream module to downstream module seamlessly if there is an edge between them. Following the flowchart, users may know the available downstream operation modules.

Documents for each module can be obtained easily by querying the module's name. For example,if users want to query functions related to "SamToBed" in the flowchart, the command may be like this:

```
?SamToBed
```

For more detail about the package, visit package home page: http://bioconductor.org/packages/esATAC/

2.2 Functional Summary

The entire workflow can be mainly divided into two parts, raw data processing and statistical analysis. Either part contains several modules and each module can be used individually. Preset pipelines are implemented with these modules (see Section 3). Advanced users are also able to build their own pipeline with them (see Section 4). Here we introduce essential module functions in these two parts. For the parameter detail of the modules, please see the manual from the home page (http://bioconductor.org/packages/esATAC/).

In the raw data processing part, esATAC can directly handle ATAC-seq raw data in FASTQ format (gzip/bzip2 compressed or uncompressed) and merge several samples if necessary. It wraps AdapterRemoval (Schubert, et al., 2016) for adapter detecting trimming. Bowtie2 (Langmead and Salzberg, 2012) are integrated for reads alignment and generate a SAM file for alignment results. SAM, BAM and BED are the most commonly used format for saving reads information. esATAC offers functions to convert file from SAM to BAM, BAM to BED, SAM to BED. What's more, esATAC will sort the mapped reads, remove duplicates, shift reads for Tn5 insertion (Buenrostro, et al., 2013) and generate intensity profile in BigWig format for genome browser visualization.

In the statistical analysis part, esATAC provides a comprehensive analyzing procedure for mapped ATAC-seq reads. It identifies open chromatin peak regions using F-seq (Boyle, et al., 2008) which specializes in seeking genome-wide profiling of open chromatin regions with high sensitivity (Koohy, et al., 2014). The peaks are annotated according to their genomic coordinate (Yu, et al., 2015). Genes around peaks and their significant gene ontology (GO) terms are reported (Yu, et al., 2012). esATAC has integrated known TF motifs in JASPAR database (Mathelier, et al., 2016) to find potential TF binding sites in the peak regions, and generate TF footprinting plots. Users can provide their own PFM (position frequency matrix) or PWM (position weight matrix) to the program as well. For human data, esATAC can also report peaks overlapped with DNase Hypersensitive Sites (DHS) identified in ENCODE project (Thurman, et al., 2012), and peaks associated with known disease SNPs from ClinVar database (Landrum, et al., 2014).

esATAC provides multiple level QC functions (sequence QC, library QC and functional annotation QC). It calls QuasR (Gaidatzis, et al., 2015) to generate raw sequencing reads quality report. It performs fragment length QC analysis, providing that typical ATAC-seq fragment length distribution has a clear periodicity caused by nucleosome protection and the pitch of the DNA helix. Other QC methods adopted by ENCODE consortium like proportion of peaks within blacklist regions (Dunham, et al., 2012), nonredundant fraction, PCR bottlenecking coefficients and fraction of reads in peaks (Landt, et al., 2012) have been also integrated.

3 One Command Line Execution for Preset Pipelines

esATAC provides an easy-to-use entry, user only need to provide ATAC-seq sequencing files (FASTQ format), and assign the spaces and genome assembly, it will do everything for user. esATAC will download the genome sequence and annotation files, build bowtie2 index, mapping the reads, do the quality control analysis, find peak regions, perform GO analysis and motif enrichment analysis, etc. automatically. Finally, users will get a report file in HTML format included to quality control and statistical analysis results. We just provide some preliminary examples here. More comprehensive examples about advance usage, multiple replicates and results of representative datasets are shown on our example websites https://wzthu.github.io/esATAC/example.

3.1 For Case-control Analysis

Here, we show a simple runnable example for case-control analysis. Case and control test sample are both paired-end data. Each of them contains two gzipped FASTQ files and has been wrapped in the package. The test file paths can be obtained like this: `system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz")`. They are under package installation directory "your-r-package-directory/esATAC/extdata/".

```
library(esATAC)

conclusion <-
atacPipe2(
  case=list(fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
            fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz")),
  control=list(fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.2.fq.bz2"),
               fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.2.fq.bz2")),
  genome = "hg19",
  motifPWM = getMotifPWM(motif.file = system.file("extdata", "CTCF.txt", package="esATAC"),
                         is.PWM = FALSE))
```

3.2 For Single Sample Analysis

Here, we show an simple runnable example for single sample analysis. We just use case data in "Case-control Analysis" section.

```
library(esATAC)

conclusion <-
  atacPipe(
    fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
    fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz"),
    genome = "hg19",
    motifPWM = getMotifPWM(motif.file = system.file("extdata", "CTCF.txt", package="esATAC"),
                           is.PWM = FALSE))
```

To test on real data from GEO (accession number GSE47753, Buenrostro, et al., 2013), we downloaded SRR891268.sra, SRR891269.sra, SRR891270.sra and SRR891271.sra. They are 4 replicates of ATAC-seq paired end sequencing data from human GM12878 cell line containing nearly 400M paired end reads in total.

And then, we used NCBI SRA Toolkit to extract FASTQ files with command like `fastq-dump --split-3 SRR8912XX.sra`. Eight FASTQ files will be generated (SRR891268_1.fastq, SRR891268_2.fastq, SRR891269_1.fastq, SRR891269_2.fastq, SRR891270_1.fastq, SRR891270_2.fastq, SRR891271_1.fastq, SRR891271_2.fastq) under current working directory.

The scripts above can be modified like this:

```
options(java.parameters = "-Xmx8000m")
library(esATAC)

conclusion <-
  atacRepsPipe(
    fastqInput1 = list("SRR891268_1.fastq", "SRR891269_1.fastq",
                       "SRR891270_1.fastq", "SRR891271_1.fastq"),
    fastqInput2 = list("SRR891268_2.fastq", "SRR891269_2.fastq",
                       "SRR891270_2.fastq", "SRR891271_2.fastq"),
    genome = "hg19")
```

Max memory size for java is recommended to be set to 8000MB in this example or rJava will use the system java default parameter for f-seq. By default, esATAC will perform footprint analysis for all the vertebrate TF motif PWM in JASPAR database. It may take ~2 days to run the pipeline depending on computer performance. To speed up the process, users can configure the multi-thread parameter like `threads=8` depending on hardware environment. See Section 6: Speed up for more details.

4 Customized Pipeline

All sub-modules are available for recombining new whole pipeline or sub-pipeline easily and flexibly. They are also able to be called individually. We just show some functions and their combinations from the package. For detail, the users can read the package vignettes and manuals.

4.1 Overview for a Customized Pipeline

Example: a simplified `atacPipe` from single sample (FASTQ) to library quality control and peak calling

```
library(esATAC)
library(magrittr)


dir.create("./esATAC_pipeline")
dir.create("./esATAC_pipeline/refdir")
dir.create("./esATAC_pipeline/result")

#configure reference path, result path, the max number of threads, genome
options(atacConf=setConfigure("refdir","esATAC_pipeline/refdir"))
options(atacConf=setConfigure("tmpdir","esATAC_pipeline/result"))
options(atacConf=setConfigure("threads",2))
options(atacConf=setConfigure("genome","hg19"))

#raw reads
fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz")
fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz")

#pipeline
atacUnzipAndMerge(fastqInput1 = fastqInput1,fastqInput2 = fastqInput2) %T>%
atacQCReport %>%
atacRenamer %>%
atacRemoveAdapter %>%
atacBowtie2Mapping %T>%
atacLibComplexQC %>%
atacSamToBed(maxFragLen = 2000) %T>%
atacBedToBigWig %T>%
atacFragLenDistr %>%
atacBedUtils(maxFragLen = 100, chrFilterList = NULL) %>%
atacPeakCalling
```

4.2 Details for Customized Pipeline

4.2.1 Configuration

There are 4 environment parameters (“refdir”,“tmpdir”,“threads”,“genome”) in this package. None of them is required because users can also pass them to the function as arguments based on needs. For convenience, by configuring them in advance, these arguments for functions will not need to be set repeatedly. Users can focus more on data flow pipeline implementation rather than dependency. These are the details for the parameters:

“genome”, the genome like “hg19”,“mm10”, etc.

“refdir”, the directory for genome reference and annotation data storage.

"tmpdir", the directory for intermediate files and result storage. Default: "./", current working directory

"threads", the max number of threads allowed to be created. Default: 1

```
dir.create("./esATAC_pipeline")
dir.create("./esATAC_pipeline/refdir")
dir.create("./esATAC_pipeline/result")

#configure reference path, result path, the max number of threads, genome
options(atacConf=setConfigure("refdir","esATAC_pipeline/refdir"))
options(atacConf=setConfigure("tmpdir","esATAC_pipeline/result"))
options(atacConf=setConfigure("threads",2))
options(atacConf=setConfigure("genome","hg19"))
```

In this example, we create folder "./esATAC_pipeline/refdir" for "refdir" and "./esATAC_pipeline/result" for "tmpdir". Two threads are allowed. Genome reference and annotations will be installed under "./esATAC_pipeline/refdir" if they are not existing or complete.

4.2.2 Recombining Sub-modules Functions

Users may find the modules and paths from the dataflow graph easily. In this example, users only need to call the blue modules in Figure S2. By using pipe operators in R (%>% and %T>%), parameters are passed seamlessly from upstream modules to downstream modules.

```
#raw reads
fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz")
fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz")

#pipeline
atacUnzipAndMerge(fastqInput1 = fastqInput1,fastqInput2 = fastqInput2) %T>%
atacQCReport %>%
atacRenamer %>%
atacRemoveAdapter %>%
atacBowtie2Mapping %T>%
atacLibComplexQC %>%
atacSamToBed(maxFragLen = 2000) %T>%
atacBedToBigWig %T>%
atacFragLenDistr %>%
atacBedUtils(maxFragLen = 100, chrFilterList = NULL) %>%
atacPeakCalling
```

4.3 Integrate Other Tools with esATAC

Users may integrate other tools from any intermediate stages easily. For example, if users want to try other programs to find open chromatin peak regions, they can directly take the mapped reads in standard BED or BAM format generated by esATAC, provide them to other peak calling programs, and return the identified peaks in BED format back to esATAC for annotation analysis.

5 Resume Analysis

To run the whole pipeline for a typical ATAC-seq data set of human sample may take ~2 days on a personal computer with single thread. If R has been stopped during the process under any circumstance, users can

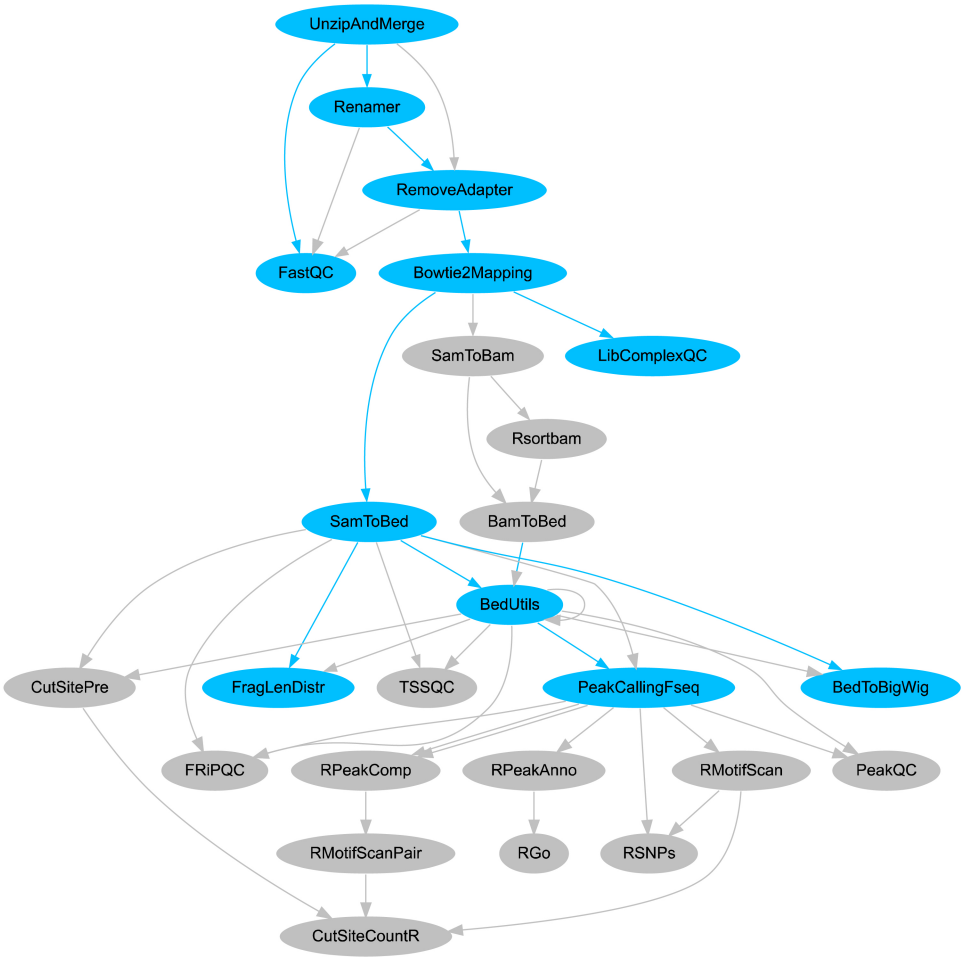

Figure S2: Sub-pipeline flowchart

simply resume the analysis by rerun the same R scripts. The program will automatically check and skip the steps that have been finished and continue the analysis.

6 Speed up

6.1 Program Configuration

In order to to fit minimum memory requirement (8GB), the default value of "threads" for the preset pipeline is set to 2 and the customized pipeline is set to 1. If the computer contains more CPU cores and memory, increasing the number of threads may greatly accelerate the time-consuming modules like bowtie2 index building, bowtie2 read alignments and motif scanning. But in some modules like motif scanning, one more threads will consume about 2GB more memory for human and mouse data. Users can adjust the trade-off between speed and memory depending on the computer to maximize the performance. For detail, the users can read the package vignettes and manuals.

6.2 Using Preset Motif Data

Scanning motif is a time-consuming step in ATAC-seq data analysis. If users use the pre-scaned data, the whole pipeline will be faster. Of course, once the data is saved, it can be used many times without redoing motif scan every times. We use "scanGenomeMotif" function to scan motif information in the genome. Users can use these data to plot footprint.
We provide 4 datasets (hg19, hg38, mm9, mm10) for esATAC users. Vertebrata motifs are from *JASPAR2016 database*. Users could download *here* or generate by themselves. The following is the tutorial about how to use esATAC to build preset motif information data for hg19 and use this data to run ATAC-seq data analysis pipeline.

6.2.1 How to Generate Preset Motif Data of hg19

```
# get motif PWM from JASPAR2016 and change motif ID to motif name
library(JASPAR2016)
library(TFBSTools)
library(esATAC)
library(BSgenome.Hsapiens.UCSC.hg19)

opts <- list()
opts["tax_group"] <- "vertebrates"  # using vertebrates
pwm <- getMatrixSet(JASPAR2016, opts)
pwm <- TFBSTools::toPWM(pwm)  # convert PFM to PWM
names(pwm) <- TFBSTools::name(pwm)
pwm <- lapply(X = pwm, FUN = TFBSTools::as.matrix)
names(pwm) <- gsub(pattern = "[^a-zA-Z0-9]", replacement = "", x = names(pwm), perl = TRUE)
# remove special characters
# scan motif position of the genome and save as a rds file
scanGenomeMotif(motifPWM = pwm, refgenome = BSgenome.Hsapiens.UCSC.hg19,
                min.score = "90%", n.core = 12, output = "JASPAR_verteb_hg19.rds")
```

The output of the above program can be loaded directly by the pipeline in esATAC. This will save lots of time when running the ATAC-seq pipeline. Of course, if you want to use another PWM and genome, just change the parameters in "scanGenomeMotif". For more information, please see *esATAC Reference Manual*.

6.2.2 How to Use Preset Data When Analysis ATAC-seq Data

Now, we have the motif data. Users can use this data in ATAC-seq analysis just modifying a few parameters of the pipeline.

```
options(java.parameters = "-Xmx8192m")  # set Java parameter, depend on your data
library(esATAC)

# ATAC-seq data, case&control analysis
case = list(fastqInput1="test1_1.fastq.bz2",fastqInput2="test1_2.fastq.bz2",
            adapter1 = NULL, adapter2 = NULL)
control =list(fastqInput1="test2_1.fastq.bz2",fastqInput2="test2_2.fastq.bz2",
              adapter1 = NULL, adapter2 = NULL)

results <- atacPipe2(case = case,
                     control = control,
                     refdir = "path_to_bowtie2_index",
                     genome = "hg19",
                     tmpdir = "path_to_save_temp_data",
                     threads = 4,
                     chr = c(1:22, "X", "Y"), use.SavedPWM = "JASPAR_verteb_hg19.rds")
```

7 Reference

Buenrostro, J.D., et al. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. Nat. Methods 2013;10(12):1213-1218.

Schubert, M., et al. AdapterRemoval v2: rapid adapter trimming, identification, and read merging. BMC research notes 2016;9:88-88.

Langmead, B., et al. Fast gapped-read alignment with Bowtie 2. Nat. Methods 2012;9(4):357-U354.

Boyle, A.P., et al. F-Seq: a feature density estimator for high-throughput sequence tags. Bioinformatics 2008;24(21):2537-2538.

Koohy, H., et al. A Comparison of Peak Callers Used for DNase-Seq Data. Plos One 2014;9(5).

Yu, G., et al. clusterProfiler: an R Package for Comparing Biological Themes Among Gene Clusters. Omics-a Journal of Integrative Biology 2012;16(5):284-287.

Yu, G., et al. ChIPseeker: an R/Bioconductor package for ChIP peak annotation, comparison and visualization. Bioinformatics 2015;31(14):2382-2383.

Thurman, R.E., et al. The accessible chromatin landscape of the human genome. Nature 2012;489(7414):75-82.

Landrum, M.J., et al. ClinVar: public archive of relationships among sequence variation and human phenotype. Nucleic Acids Res. 2014;42(D1):D980-D985.

Gaidatzis, D., et al. QuasR: quantification and annotation of short reads in R. Bioinformatics 2015;31(7):1130-1132.

Dunham, I., et al. An integrated encyclopedia of DNA elements in the human genome. Nature 2012;489(7414):57-74.

Landt, S.G., et al. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. Genome Res. 2012;22(9):1813-1831.