Package 'esATAC'

November 12, 2026

Type Package

Title An Easy-to-use Systematic pipeline for ATACseq data analysis

Version 1.34.0

Date 2025-07-22

Author Zheng Wei, Wei Zhang

Maintainer Zheng Wei <wzweizheng@qq.com>

Description This package provides a framework and complete preset pipeline for quantification and analysis of ATAC-seq Reads. It covers raw sequencing reads preprocessing (FASTQ files), reads alignment (Rbowtie2), aligned reads file operations (SAM, BAM, and BED files), peak calling (F-seq), genome annotations (Motif, GO, SNP analysis) and quality control report. The package is managed by dataflow graph. It is easy for user to pass variables seamlessly between processes and understand the workflow. Users can process FASTQ files through end-to-end preset pipeline which produces a pretty HTML report for quality control and preliminary statistical results, or customize workflow starting from any intermediate stages with esATAC functions easily and flexibly.

Depends R (>= 4.0.0), Rsamtools, GenomicRanges, ShortRead, pipeFrame

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.11), methods, knitr, Rbowtie2, rtracklayer, ggplot2, Biostrings, ChIPseeker, clusterProfiler, igraph, rJava, magrittr, digest, BSgenome, AnnotationDbi, GenomicAlignments, GenomicFeatures, R.utils, Seqinfo, BiocGenerics, S4Vectors, IRanges, rmarkdown, tools, VennDiagram, grid, JASPAR2018, TFBSTools, grDevices, graphics, stats, utils, parallel, corrplot, BiocManager, motifmatchr

Suggests BSgenome.Hsapiens.UCSC.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene, org.Hs.eg.db, testthat, webshot, prettydoc

LinkingTo Rcpp

SystemRequirements C++11

Collate ATACProc.R BamToBed.R BedToBigWig.R BedUtils.R ConfigVal.R CppInterface.R CutSiteCountR.R CutSitePre.R FRiPQC.R FastQC.R FregLenDistribute.R FindAdapter.R JavaExports.R JavaInterface.R LibComplexQC.R Mapping.R Methods.R PeakCallingFseq.R PeakCallingMACS2.R PeakQC.R RGo.R RMotifScan.R RPeakAnno.R RPeakComp.R RSNPs.R RcppExports.R RemoveAdapter.R Renamer.R Rsortbam.R SamToBam.R SamToBed.R TSSQC.R UnzipAndMerge.R onLoad.R RMotifScanPair.R utilities.R SingleRepReport.R

biocViews ImmunoOncology, Sequencing, DNASeq, QualityControl, Alignment, Preprocessing, Coverage, ATACSeq, DNaseSeq

VignetteBuilder knitr

Archs x64

RoxygenNote 7.1.1

NeedsCompilation yes

URL https://github.com/wzthu/esATAC

BugReports https://github.com/wzthu/esATAC/issues

git_url https://git.bioconductor.org/packages/esATAC

git_branch RELEASE_3_23

git_last_commit 279243b

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

esATAC-package . . . 3
atacPipe2 . . . 6
ATACProc-class . . . 9
atacRepsPipe . . . 9
atacRepsPipe2 . . . 11
BamToBed . . . 15
BedToBigWig . . . 17
BedUtils . . . 19
Bowtie2Mapping . . . 22
CutSiteCountR . . . 24
CutSitePre . . . 27
FastQC . . . 28
FindAdapter . . . 29
FragLenDistr . . . 31
FRiPQC . . . 33
getMotifInfo . . . 34
LibComplexQC . . . 35
PeakCallingFseq . . . 37

PeakCallingMACS2 . . . 39
PeakQC . . . 41
RemoveAdapter . . . 43
Renamer . . . 46
RGo . . . 48
RMotifScan . . . 50
RMotifScanPair . . . 52
RPeakAnno . . . 54
RPeakComp . . . 57
RSNPs . . . 59
Rsortbam . . . 60
SamToBam . . . 61
SamToBed . . . 62
SingleRepReport . . . 65
TSSQC . . . 66
UnzipAndMerge . . . 68

Index **70**

`esATAC-package` *An Easy-to-use Systematic pipeline for ATACseq data analysis*

Description

This package provides a framework and complete preset pipeline for the quantification and analysis of ATAC-seq Reads. It covers raw sequencing reads preprocessing (FASTQ files), reads alignment (Rbowtie2), aligned reads file operation (SAM, BAM, and BED files), peak calling (fseq), genome annotations (Motif, GO, SNP analysis) and quality control report. The package is managed by dataflow graph. It is easy for user to pass variables seamlessly between processes and understand the workflow. Users can process FASTQ files through end-to-end preset pipeline which produces a pretty HTML report for quality control and preliminary statistical results, or customize workflow starting from any intermediate stages with esATAC functions easily and flexibly.

Preset pipeline for single replicate case study is shown below.

For multi-replicates case study, see `atacRepsPipe`.

For single replicate case-control study, see `atacPipe2`.

For multi-replicates case-control study, see `atacRepsPipe2`.

NOTE: Build bowtie index in the function may take some time. If you already have bowtie2 index files or you want to download(`ftp://ftp.ccb.jhu.edu/pub/data/bowtie2_indexes`) instead of building, you can let esATAC skip the steps by renaming them following the format (genome+suffix) and put them in reference installation path (refdir). Example: hg19 bowtie2 index files

- hg19.1.bt2
- hg19.2.bt2
- hg19.3.bt2
- hg19.4.bt2

- hg19.rev.1.bt2
- hg19.rev.2.bt2

For single end reads FASTQ files, The required parameters are fastqInput1 and adapter1. For paired end reads non-interleaved FASTQ files (interleave=FALSE,defualt), The required parameters are fastqInput1 and fastqInput2. Otherwise, parameter fastqInput2 is not required (interleave=TRUE)

The paths of sequencing data replicates can be a `Character` vector. For example:

fastqInput1=c("file_1.rep1.fastq","file_1.rep2.fastq")

fastqInput2=c("file_2.rep1.fastq","file_2.rep2.fastq")

The result will be return by the function. An HTML report file will be created for paired end reads. Intermediate files will be save at tmpdir path (default is ./)

Usage

```
atacPipe(
  genome,
  fastqInput1,
  fastqInput2 = NULL,
  tmpdir = file.path(getwd(), "esATAC-pipeline"),
  refdir = file.path(tmpdir, "refdir"),
  threads = 2,
  adapter1 = NULL,
  adapter2 = NULL,
  interleave = FALSE,
  basicAnalysis = FALSE,
  createReport = TRUE,
  motifs = NULL,
  pipelineName = "pipe",
  chr = c(1:22, "X", "Y"),
  p.cutoff = 1e-06,
  ...
)
```

Arguments

`genome`	`Character` scalar. The genome(like hg19, mm10, etc.) reference data in "refdir" to be used in the pipeline.
`fastqInput1`	`Character` vector. For single-end sequencing, it contains sequence file paths. For paired-end sequencing, it can be file paths with #1 mates paired with file paths in fastqInput2 And it can also be interleaved file paths when argument interleaved=`TRUE`
`fastqInput2`	`Character` vector. It contains file paths with #2 mates paired with file paths in fastqInput1. For single-end sequencing files and interleaved paired-end sequencing files(argument interleaved=`TRUE`), it must be `NULL`.
`tmpdir`	`Character` scalar. The temporary file storage path.
`refdir`	`Character` scalar. The path for reference data being installed to and storage.

threads	Integer scalar. The max threads allowed to be created.
adapter1	Character scalar. It is an adapter sequence for file1. For single end data, it is requied.
adapter2	Character scalar. It is an adapter sequence for file2.
interleave	Logical scalar. Set TRUE when files are interleaved paired-end sequencing data.
basicAnalysis	Logical scalar. If it is TRUE, the pipeline will skip the time consuming steps like GO annoation and motif analysis
createReport	Logical scalar. If the HTML report file will be created.
motifs	eitherPFMatrix, PFMatrixList, PWMatrix, PWMatrixList, default: vertebrates motif from JASPAR.
pipelineName	Character scalar. Temporary file prefix for identifying files when multiple pipeline generating file in the same tempdir.
chr	Which chromatin the program will processing. It must be identical with the filename of cut site information files or subset of . Default:c(1:22, "X", "Y").
p.cutoff	p-value cutoff for returning motifs, default: 1e-6.
...	Additional arguments, currently unused.

Details

See packageDescription('esATAC') for package details.

Value

List scalar. It is a list that save the result of the pipeline. Slot "filelist": the input file paths. Slot "wholesummary": a dataframe that for quality control summary Slot "atacProcs": ATACProc-class objects generated by each process in the pipeline. Slot "filtstat": a dataframe that summary the reads filted in each process.

Author(s)

Zheng Wei and Wei Zhang

See Also

printMap, atacPipe2, atacRenamer, atacRemoveAdapter, atacBowtie2Mapping, atacPeakCalling, atacMotifScan, atacRepsPipe, atacRepsPipe2

Examples

```
## Not run:
## These codes are time consuming so they will not be run and
## checked by bioconductor checker.


# call pipeline
# for a quick example(only CTCF and BATF3 will be processing)
conclusion <-
```

```
  atacPipe(
       # MODIFY: Change these paths to your own case files!
       # e.g. fastqInput1 = "your/own/data/path.fastq"
       fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
       fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz"),
       # MODIFY: Set the genome for your data
       genome = "hg19",
     motifs = getMotifInfo(motif.file = system.file("extdata", "CustomizedMotif.txt", package="esATAC")))

# call pipeline
# for overall example(all vertebrates motif in JASPAR will be processed)
conclusion <-
  atacPipe(
       # MODIFY: Change these paths to your own case files!
       # e.g. fastqInput1 = "your/own/data/path.fastq"
       fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
       fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz"),
       # MODIFY: Set the genome for your data
       genome = "hg19")

## End(Not run)
```

`atacPipe2` *Pipeline for single replicate case-control paired-end sequencing data*

Description

The preset pipeline to process case control study sequencing data. An HTML report file, result files(e.g. BED, BAM files) and conclusion list will generated. See detail for usage.

Usage

```
atacPipe2(
  genome,
  case = list(fastqInput1 = "paths/To/fastq1", fastqInput2 = "paths/To/fastq2",
    adapter1 = NULL, adapter2 = NULL),
 control = list(fastqInput1 = "paths/To/fastq1", fastqInput2 = "paths/To/fastq2",
    adapter1 = NULL, adapter2 = NULL),
  tmpdir = file.path(getwd(), "esATAC-pipeline"),
  refdir = file.path(tmpdir, "refdir"),
  threads = 2,
  interleave = FALSE,
  createReport = TRUE,
  motifs = NULL,
  chr = c(1:22, "X", "Y"),
  p.cutoff = 1e-06,
  ...
)
```

Arguments

genome	Character scalar. The genome(like hg19, mm10, etc.) reference data in "refdir" to be used in the pipeline.
case	List scalar. Input for case sample. fastqInput1, the path(s) of the mate 1 fastq file(s), is required. fastqInput2, the path(s) of the mate 2 fastq file(s), is required, when interleave=FALSE. adapter1 and adapter2 are optional.
control	List scalar. Input for control sample. fastqInput1, the path(s) of the mate 1 fastq file(s), is required. fastqInput2, the path(s) of the mate 2 fastq file(s), is required, when interleave=FALSE. adapter1 and adapter2 are optional.
tmpdir	Character scalar. The temporary file storage path.
refdir	Character scalar. The path for reference data being installed to and storage.
threads	Integer scalar. The max threads allowed to be created.
interleave	Logical scalar. Set TRUE when files are interleaved paired-end sequencing data.
createReport	Logical scalar. If the HTML report file will be created.
motifs	eitherPFMatrix, PFMatrixList, PWMatrix, PWMatrixList, default: vertebrates motif from JASPAR.
chr	Which chromatin the program will processing. It must be identical with the filename of cut site information files or subset of . Default:c(1:22, "X", "Y").
p.cutoff	p-value cutoff for returning motifs, default: 1e-6.
...	Additional arguments, currently unused.

Details

NOTE: Build bowtie index in this function may take some time. If you already have bowtie2 index files or you want to download(ftp://ftp.ccb.jhu.edu/pub/data/bowtie2_indexes) instead of building, you can let esATAC skip the steps by renaming them following the format (genome+suffix) and put them in reference installation path (refdir). Example: hg19 bowtie2 index files

- hg19.1.bt2
- hg19.2.bt2
- hg19.3.bt2
- hg19.4.bt2
- hg19.rev.1.bt2
- hg19.rev.2.bt2

For single end reads FASTQ files, The required parameters are fastqInput1 and adapter1. For paired end reads non-interleaved FASTQ files (interleave=FALSE,defualt), The required parameters are fastqInput1 and fastqInput2. Otherwise, parameter fastqInput2 is not required (interleave=TRUE)

The paths of sequencing data replicates can be a Character vector. For example:

fastqInput1=c("file_1.rep1.fastq","file_1.rep2.fastq")

fastqInput2=c("file_2.rep1.fastq","file_2.rep2.fastq")

The result will be return by the function. An HTML report file will be created for paired end reads. Intermediate files will be save at tmpdir path (default is ./)

Value

List scalar. It is a list that save the result of the pipeline. Slot "wholesummary": a dataframe for quality control summary of case and control data Slot "caselist" and "ctrlist": Each of them is a list that save the result for case or control data. Slots of "caselist" and "ctrllist": Slot "filelist": the input file paths. Slot "wholesummary": a dataframe for quality control summary of case or control data Slot "atacProcs": ATACProc-class objects generated by each process in the pipeline. Slot "filtstat": a dataframe that summary the reads filted in each process.

Author(s)

Zheng Wei and Wei Zhang

See Also

atacPipe

Examples

```
## Not run:
## These codes are time consuming so they will not be run and
## checked by bioconductor checker.


# call pipeline
# for a quick example(only CTCF and BATF3 will be processed)
conclusion <-
   atacPipe2(
       # MODIFY: Change these paths to your own case files!
       # e.g. fastqInput1 = "your/own/data/path.fastq"
     case=list(fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
             fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz")),
       # MODIFY: Change these paths to your own control files!
       # e.g. fastqInput1 = "your/own/data/path.fastq"
     control=list(fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.2.fq.bz2"),
             fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.2.fq.bz2")),
       # MODIFY: Set the genome for your data
       genome = "hg19",
     motifs = getMotifInfo(motif.file = system.file("extdata", "CustomizedMotif.txt", package="esATAC")))

# call pipeline
# for overall example(all vertebrates motif in JASPAR will be processed)
conclusion <-
   atacPipe2(
       # MODIFY: Change these paths to your own case files!
       # e.g. fastqInput1 = "your/own/data/path.fastq"
     case=list(fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
             fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz")),
       # MODIFY: Change these paths to your own control files!
       # e.g. fastqInput1 = "your/own/data/path.fastq"
     control=list(fastqInput1 = system.file(package="esATAC", "extdata", "chr20_1.2.fq.bz2"),
             fastqInput2 = system.file(package="esATAC", "extdata", "chr20_2.2.fq.bz2")),
```

```
        # MODIFY: Set the genome for your data
        genome = "hg19")

## End(Not run)
```

`ATACProc-class` *Base class of this package*

Description

This class is inherit from `Step` in pipeFrame package, no more method is extended or override. Please see `Step` class for detail.

`atacRepsPipe` *Pipeline for multi-replicates case paired-end sequencing data*

Description

The preset pipeline to process multi-replicates case study sequencing data. HTML report files, result files(e.g. BED, BAM files) and conclusion list will generated. See detail for usage.

Usage

```
atacRepsPipe(
  genome,
  fastqInput1,
  fastqInput2 = NULL,
  refdir = NULL,
  tmpdir = NULL,
  threads = 2,
  adapter1 = NULL,
  adapter2 = NULL,
  interleave = FALSE,
  createReport = TRUE,
  motifs = NULL,
  prefix = NULL,
  chr = c(1:22, "X", "Y"),
  p.cutoff = 1e-06,
  ...
)
```

Arguments

genome	Character scalar. The genome(like hg19, mm10, etc.) reference data in "refdir" to be used in the pipeline.
fastqInput1	List scalar. For single-end sequencing, it contains sequence file paths. For paired-end sequencing, it can be file paths with #1 mates paired with file paths in fastqInput2 And it can also be interleaved file paths when argument interleaved=TRUE. Each element in the fastqInput1 List is for a replicate It can be a Character vector of FASTQ files paths to be merged.
fastqInput2	List scalar. It contains file paths with #2 mates paired with file paths in fastqInput1. For single-end sequencing files and interleaved paired-end sequencing files(argument interleaved=TRUE), it must be NULL. Each element in the fastqInput1 List is for a replicate It can be a Character vector of FASTQ files paths to be merged.
refdir	Character scalar. The path for reference data being installed to and storage.
tmpdir	Character scalar. The temporary file storage path.
threads	Integer scalar. The max threads allowed to be created.
adapter1	Character scalar. It is an adapter sequence for file1. For single end data, it is requied.
adapter2	Character scalar. It is an adapter sequence for file2.
interleave	Logical scalar. Set TRUE when files are interleaved paired-end sequencing data.
createReport	Logical scalar. If the HTML report file will be created.
motifs	eitherPFMatrix, PFMatrixList, PWMatrix, PWMatrixList, default: vertebrates motif from JASPAR.
prefix	Character scalar. Temporary file prefix for identifying files when multiple pipeline generating file in the same tempdir.
chr	Which chromatin the program will processing. It must be identical with the filename of cut site information files or subset of . Default:c(1:22, "X", "Y").
p.cutoff	p-value cutoff for returning motifs, default: 1e-6.
...	Additional arguments, currently unused.

Value

List scalar. It is a list that save the result of the pipeline. Slot "filelist": the input file paths. Slot "wholesummary": a dataframe that for quality control summary Slot "atacProcs": ATACProc-class objects generated by each process in the pipeline. Slot "filtstat": a dataframe that summary the reads filted in each process.

Author(s)

Zheng Wei and Wei Zhang

See Also

printMap, atacPipe2, atacRenamer, atacRemoveAdapter, atacBowtie2Mapping, atacPeakCalling, atacMotifScan

Examples

```
## Not run:
## These codes are time consuming so they will not be run and
## checked by bioconductor checker.


# call pipeline
# for a quick example(only CTCF and BATF3 will be processing)
conclusion <-
  atacRepsPipe(
       # MODIFY: Change these paths to your own case files!
       # e.g. fastqInput1 = "your/own/data/path.fastq"
       fastqInput1 = list(system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
                          system.file(package="esATAC", "extdata", "chr20_1.2.fq.bz2")),
       fastqInput2 = list(system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz"),
                          system.file(package="esATAC", "extdata", "chr20_2.2.fq.bz2")),
       # MODIFY: Set the genome for your data
       genome = "hg19",
     motifs = getMotifInfo(motif.file = system.file("extdata", "CustomizedMotif.txt", package="esATAC")))

# call pipeline
# for overall example(all vertebrates motif in JASPAR will be processed)
conclusion <-
  atacRepsPipe(
       # MODIFY: Change these paths to your own case files!
       # e.g. fastqInput1 = "your/own/data/path.fastq"
       fastqInput1 = list(system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
                          system.file(package="esATAC", "extdata", "chr20_1.2.fq.bz2")),
       fastqInput2 = list(system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz"),
                          system.file(package="esATAC", "extdata", "chr20_2.2.fq.bz2")),
       # MODIFY: Set the genome for your data
       genome = "hg19")

## End(Not run)
```

`atacRepsPipe2` *Pipeline for multi-replicates case-control paired-end sequencing data*

Description

The preset pipeline to process multi-replicates case control study sequencing data. HTML report files, result files(e.g. BED, BAM files) and conclusion list will generated. See detail for usage.

Usage

```
atacRepsPipe2(
  genome,
  caseFastqInput1,
  caseFastqInput2,
```

```
  ctrlFastqInput1,
  ctrlFastqInput2,
  caseAdapter1 = NULL,
  caseAdapter2 = NULL,
  ctrlAdapter1 = NULL,
  ctrlAdapter2 = NULL,
  refdir = NULL,
  tmpdir = NULL,
  threads = 2,
  interleave = FALSE,
  createReport = TRUE,
  motifs = NULL,
  chr = c(1:22, "X", "Y"),
  p.cutoff = 1e-06,
  ...
)
```

Arguments

genome	Character scalar. The genome(like hg19, mm10, etc.) reference data in "refdir" to be used in the pipeline.
caseFastqInput1	List scalar. Input for case samples. For single-end sequencing, it contains sequence file paths. For paired-end sequencing, it can be file paths with #1 mates paired with file paths in fastqInput2 And it can also be interleaved file paths when argument interleaved=TRUE. Each element in the caseFastqInput1 List is for a replicate It can be a Character vector of FASTQ files paths to be merged.
caseFastqInput2	List scalar. Input for case samples. It contains file paths with #2 mates paired with file paths in caseFastqInput1 For single-end sequencing files and interleaved paired-end sequencing files(argument interleaved=TRUE), it must be NULL. Each element in the caseFastqInput2 List is for a replicate
ctrlFastqInput1	List scalar. Input for control samples. For single-end sequencing, it contains sequence file paths. For paired-end sequencing, it can be file paths with #1 mates paired with file paths in ctrlFastqInput2 And it can also be interleaved file paths when argument interleaved=TRUE. Each element in the ctrlFastqInput1 List is for a replicate It can be a Character vector of FASTQ files paths to be merged.
ctrlFastqInput2	List scalar. Input for control samples. It contains file paths with #2 mates paired with file paths in fastqInput1. For single-end sequencing files and interleaved paired-end sequencing files(argument interleaved=TRUE), it must be NULL. Each element in the ctrlFastqInput1 List is for a replicate
caseAdapter1	Character scalar. Adapter for caseFastqInput1.
caseAdapter2	Character scalar. Adapter for caseFastqInput2.
ctrlAdapter1	Character scalar. Adapter for ctrlFastqInput1.

ctrlAdapter2	Character scalar. Adapter for ctrlFastqInput2.
refdir	Character scalar. The path for reference data being installed to and storage.
tmpdir	Character scalar. The temporary file storage path.
threads	Integer scalar. The max threads allowed to be created.
interleave	Logical scalar. Set TRUE when files are interleaved paired-end sequencing data.
createReport	Logical scalar. If the HTML report file will be created.
motifs	eitherPFMatrix, PFMatrixList, PWMatrix, PWMatrixList, default: vertebrates motif from JASPAR.
chr	Which chromatin the program will processing. It must be identical with the filename of cut site information files or subset of . Default:c(1:22, "X", "Y").
p.cutoff	p-value cutoff for returning motifs, default: 1e-6.
...	Additional arguments, currently unused.

Details

NOTE: Build bowtie index in this function may take some time. If you already have bowtie2 index files or you want to download(ftp://ftp.ccb.jhu.edu/pub/data/bowtie2_indexes) instead of building, you can let esATAC skip the steps by renaming them following the format (genome+suffix) and put them in reference installation path (refdir). Example: hg19 bowtie2 index files

- hg19.1.bt2
- hg19.2.bt2
- hg19.3.bt2
- hg19.4.bt2
- hg19.rev.1.bt2
- hg19.rev.2.bt2

For single end reads FASTQ files, The required parameters are fastqInput1 and adapter1. For paired end reads non-interleaved FASTQ files (interleave=FALSE,defualt), The required parameters are fastqInput1 and fastqInput2. Otherwise, parameter fastqInput2 is not required (interleave=TRUE)

The paths of sequencing data replicates can be a Character vector. For example:

fastqInput1=c("file_1.rep1.fastq","file_1.rep2.fastq")

fastqInput2=c("file_2.rep1.fastq","file_2.rep2.fastq")

The result will be return by the function. An HTML report file will be created for paired end reads. Intermediate files will be save at tmpdir path (default is ./)

Value

List scalar. It is a list that save the result of the pipeline. Slot "caselist" and "ctrlist": Each of them is a list that save the result for case or control data. Slot "comp_result": compare analysis result for case and control data

Author(s)

Zheng Wei and Wei Zhang

See Also

atacPipe

Examples

```
## Not run:
## These codes are time consuming so they will not be run and
## checked by bioconductor checker.


# call pipeline
# for a quick example(only CTCF will be processed)
conclusion <-
    atacRepsPipe2(
        # MODIFY: Change these paths to your own case files!
        # e.g. fastqInput1 = "your/own/data/path.fastq"
     caseFastqInput1=list(system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
                          system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz")),
     # MODIFY: Change these paths to your own case files!
     # e.g. fastqInput1 = "your/own/data/path.fastq"
     caseFastqInput2=list(system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz"),
                          system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz")),
     # MODIFY: Change these paths to your own control files!
     # e.g. fastqInput1 = "your/own/data/path.fastq"
     ctrlFastqInput1=list(system.file(package="esATAC", "extdata", "chr20_1.2.fq.bz2"),
                          system.file(package="esATAC", "extdata", "chr20_1.2.fq.bz2")),
     # MODIFY: Change these paths to your own control files!
     # e.g. fastqInput1 = "your/own/data/path.fastq"
     ctrlFastqInput2=list(system.file(package="esATAC", "extdata", "chr20_2.2.fq.bz2"),
                          system.file(package="esATAC", "extdata", "chr20_2.2.fq.bz2")),
     # MODIFY: Set the genome for your data
     genome = "hg19",
    motifs = getMotifInfo(motif.file = system.file("extdata", "CustomizedMotif.txt", package="esATAC")))


# call pipeline
# for overall example(all human motif in JASPAR will be processed)
conclusion <-
    atacRepsPipe2(
        # MODIFY: Change these paths to your own case files!
        # e.g. fastqInput1 = "your/own/data/path.fastq"
     caseFastqInput1=list(system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz"),
                          system.file(package="esATAC", "extdata", "chr20_1.1.fq.gz")),
     # MODIFY: Change these paths to your own case files!
     # e.g. fastqInput1 = "your/own/data/path.fastq"
     caseFastqInput2=list(system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz"),
                          system.file(package="esATAC", "extdata", "chr20_2.1.fq.gz")),
     # MODIFY: Change these paths to your own control files!
     # e.g. fastqInput1 = "your/own/data/path.fastq"
     ctrlFastqInput1=list(system.file(package="esATAC", "extdata", "chr20_1.2.fq.bz2"),
                          system.file(package="esATAC", "extdata", "chr20_1.2.fq.bz2")),
     # MODIFY: Change these paths to your own control files!
```

```
    # e.g. fastqInput1 = "your/own/data/path.fastq"
    ctrlFastqInput2=list(system.file(package="esATAC", "extdata", "chr20_2.2.fq.bz2"),
                         system.file(package="esATAC", "extdata", "chr20_2.2.fq.bz2")),
    # MODIFY: Set the genome for your data
    genome = "hg19"
    )

## End(Not run)
```

BamToBed *Convert bam format to bed format.*

Description

This function is used to convert SAM file to BED file and merge interleave paired end reads, shift reads, filter reads according to chromosome, filter reads according to fragment size, sort, remove duplicates reads before generating BED file.

Usage

```
atacBam2Bed(
  atacProc,
  bamInput = NULL,
  bedOutput = NULL,
  reportOutput = NULL,
  bsgenome = NULL,
  mergePairIntoFrag = c("auto", "yes", "no"),
  posOffset = +4,
  negOffset = -5,
  chrFilterList = "chrM|_",
  sortBed = TRUE,
  rmMultiMap = TRUE,
  minFragLen = 0,
  maxFragLen = 2000,
  saveExtLen = FALSE,
  uniqueBed = c("auto", "yes", "no"),
  ...
)

## S4 method for signature 'ATACProc'
atacBam2Bed(
  atacProc,
  bamInput = NULL,
  bedOutput = NULL,
  reportOutput = NULL,
  bsgenome = NULL,
  mergePairIntoFrag = c("auto", "yes", "no"),
  posOffset = +4,
```

```
  negOffset = -5,
  chrFilterList = "chrM|_",
  sortBed = TRUE,
  rmMultiMap = TRUE,
  minFragLen = 0,
  maxFragLen = 2000,
  saveExtLen = FALSE,
  uniqueBed = c("auto", "yes", "no"),
  ...
)

bam2bed(
  bamInput,
  bedOutput = NULL,
  reportOutput = NULL,
  bsgenome = NULL,
  mergePairIntoFrag = c("auto", "yes", "no"),
  posOffset = +4,
  negOffset = -5,
  chrFilterList = "chrM|_",
  sortBed = TRUE,
  rmMultiMap = TRUE,
  minFragLen = 0,
  maxFragLen = 2000,
  saveExtLen = FALSE,
  uniqueBed = c("auto", "yes", "no"),
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacBamSort, atacSam2Bam.
bamInput	Character scalar. Bam file input path.
bedOutput	Character scalar. Bed file output path. If ignored, bed file will be put in the same path as the bam file.
reportOutput	Character scalar. Report file path.
bsgenome	BSgenome object. This object from bioconductor
mergePairIntoFrag	Logical scalar Merge paired end reads.
posOffset	Integer scalar The offset that positive strand reads will shift.
negOffset	Integer scalar The offset that negative strand reads will shift.
chrFilterList	Character vector The chromatin(or regex of chromatin) will be discard
sortBed	Logical scalar Sort bed file in the order of chromatin, start, end
rmMultiMap	Logical scalar. Remove multi-map reads.
minFragLen	Integer scalar The minimum fragment size will be retained.

`maxFragLen`	`Integer` scalar The maximum fragment size will be retained.
`saveExtLen`	`Logical` scaler. Save the fragment that are not in the range of minFragLen and maxFragLen
`uniqueBed`	`Logical` scalar Remove duplicates reads in bed if TRUE. default: FALSE
`...`	Additional arguments, currently unused.

Details

The bam file wiil be automatically obtained from object(`atacProc`) or input by hand. Output can be ignored.

Value

An invisible `ATACProc-class` object scalar for downstream analysis.

Author(s)

Zheng Wei, Wei Zhang

See Also

`atacBamSort` `atacSam2Bam`

Examples

```
library(Rsamtools)
# change dataset !!
# ex1_file <- system.file("extdata", "ex1.bam", package="Rsamtools")
# bam2bed(bamInput = ex1_file)
```

BedToBigWig *generate BigWig file from BED file*

Description

This function is used to generate BigWig file from BED reads file. The BigWig file can be shown reads coverage on genome browser.

Usage

```
atacBedToBigWig(
  atacProc,
  bedInput = NULL,
  bsgenome = NULL,
  bwOutput = NULL,
  toWig = FALSE,
  ...
```

```
)

## S4 method for signature 'ATACProc'
atacBedToBigWig(
  atacProc,
  bedInput = NULL,
  bsgenome = NULL,
  bwOutput = NULL,
  toWig = FALSE,
  ...
)

bedToBigWig(bedInput, bsgenome = NULL, bwOutput = NULL, toWig = FALSE, ...)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacSamToBed, atacBedUtils.
bedInput	Character scalar. Bed file input path.
bsgenome	BSGenome object scalar. BSGenome object for specific species.
bwOutput	Character scalar. BigWig file output path.
toWig	Logical scalar.
...	Additional arguments, currently unused. Save as wig file instead of binary BigWig file

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use bedToBigWig instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacSamToBed samToBed atacBedUtils bedUtils

Examples

```
library(R.utils)
td <- tempdir()
setTmpDir(td)

bedbzfile <- system.file(package="esATAC", "extdata", "chr20.50000.bed.bz2")
bedfile <- file.path(td,"chr20.50000.bed")
## Not run:
bunzip2(bedbzfile,destname=bedfile,overwrite=TRUE,remove=FALSE)

library(BSgenome.Hsapiens.UCSC.hg19)
bedToBigWig(bedfile, BSgenome.Hsapiens.UCSC.hg19)

dir(td)

## End(Not run)
```

`BedUtils` *process bed file with limit memory*

Description

This function is used to merge interleave paired end reads in bed, downsample bed reads, shift bed reads, filter bed reads according to chromosome, filter bed reads according to fragment size, sort bed, remove duplicates reads in bed.

Usage

```
atacBedUtils(
  atacProc,
  bedInput = NULL,
  bedOutput = NULL,
  mergePair = FALSE,
  downSample = NULL,
  posOffset = 0L,
  negOffset = 0L,
  chrFilterList = c("chrM"),
  select = FALSE,
  sortBed = FALSE,
  uniqueBed = FALSE,
  minFragLen = 0,
  maxFragLen = 2e+09,
  newStepType = "BedUtils",
  ...
)

## S4 method for signature 'ATACProc'
atacBedUtils(
```

```
  atacProc,
  bedInput = NULL,
  bedOutput = NULL,
  mergePair = FALSE,
  downSample = NULL,
  posOffset = 0L,
  negOffset = 0L,
  chrFilterList = c("chrM"),
  select = FALSE,
  sortBed = FALSE,
  uniqueBed = FALSE,
  minFragLen = 0,
  maxFragLen = 2e+09,
  newStepType = "BedUtils",
  ...
)

bedUtils(
  bedInput,
  bedOutput = NULL,
  mergePair = FALSE,
  downSample = NULL,
  reportOutput = NULL,
  posOffset = 0L,
  negOffset = 0L,
  chrFilterList = c("chrM"),
  select = FALSE,
  sortBed = FALSE,
  uniqueBed = FALSE,
  minFragLen = 0,
  maxFragLen = 2e+09,
  newStepType = "BedUtils",
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacBam2Bed bam2bed atacSamToBed samToBed
bedInput	Character scalar. Bed file input path.
bedOutput	Character scalar. Bed file output path.
mergePair	Logical scalar Merge paired end interleave reads.
downSample	Integer scalar Down sample reads if the number is less than total number
posOffset	Integer scalar The offset that positive strand reads will shift.
negOffset	Integer scalar The offset that negative strand reads will shift.
chrFilterList	Character vector The chromatin(or regex of chromatin) will be retain/discard if select is TRUE/FALSE

select	Logical scalar The chromatin in chrFilterList will be retain if TRUE. default: FALSE
sortBed	Logical scalar Sort bed file in the order of chromatin, start, end
uniqueBed	Logical scalar Remove duplicates reads in bed if TRUE. default: FALSE
minFragLen	Integer scalar The minimum fragment size will be retained.
maxFragLen	Integer scalar The maximum fragment size will be retained.
newStepType	Character scalar. New step type name for different default parameters.
...	Additional arguments, currently unused.
reportOutput	Character scalar. Report output file path.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use bedUtils instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacBam2Bed bam2bed atacSamToBed samToBed atacFragLenDistr atacExtractCutSite atacPeakCalling atacTSSQC atacBedToBigWig

Examples

```
library(R.utils)
library(magrittr)
td <- tempdir()
setTmpDir(td)

sambzfile <- system.file(package="esATAC", "extdata", "Example.sam.bz2")
samfile <- file.path(td,"Example.sam")
bunzip2(sambzfile,destname=samfile,overwrite=TRUE,remove=FALSE)
atacproc<-samToBed(samInput = samfile) %>%
atacBedUtils(maxFragLen = 100, chrFilterList = NULL)
```

Bowtie2Mapping *Use bowtie2 aligner to map reads to reference genome*

Description

Use bowtie2 aligner to map reads to reference genome

Usage

```
atacBowtie2Mapping(
  atacProc,
  samOutput = NULL,
  reportOutput = NULL,
  bt2Idx = NULL,
  fastqInput1 = NULL,
  fastqInput2 = NULL,
  interleave = FALSE,
  threads = getThreads(),
  paramList = "--no-discordant --no-unal --no-mixed -X 2000",
  ...
)

## S4 method for signature 'ATACProc'
atacBowtie2Mapping(
  atacProc,
  samOutput = NULL,
  reportOutput = NULL,
  bt2Idx = NULL,
  fastqInput1 = NULL,
  fastqInput2 = NULL,
  interleave = FALSE,
  threads = getThreads(),
  paramList = "--no-discordant --no-unal --no-mixed -X 2000",
  ...
)

bowtie2Mapping(
  fastqInput1,
  fastqInput2 = NULL,
  samOutput = NULL,
  reportOutput = NULL,
  bt2Idx = NULL,
  interleave = FALSE,
  threads = getThreads(),
  paramList = "--no-discordant --no-unal --no-mixed -X 2000",
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacRemoveAdapter removeAdapter
samOutput	Character scalar. A path to a SAM file used for the alignment output.
reportOutput	Character scalar. The prefix of report files path.
bt2Idx	Character scalar. bowtie2 index files prefix: 'dir/basename' (minus trailing '.*.bt2' of 'dir/basename.*.bt2').
fastqInput1	Character vector. For single-end sequencing, it contains sequence file paths. For paired-end sequencing, it can be file paths with #1 mates paired with file paths in fastqInput2. And it can also be interleaved file paths when argument interleaved=TRUE
fastqInput2	Character vector. It contains file paths with #2 mates paired with file paths in fastqInput1. For single-end sequencing files and interleaved paired-end sequencing files(argument interleaved=TRUE), it must be NULL.
interleave	Logical. Set TRUE when files are interleaved paired-end sequencing data.
threads	Integer scalar. The threads will be created in this process. default: getThreads()
paramList	Additional arguments to be passed on to the binaries. See below for details.
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use bowtie2Mapping instead. additional parameters to be passed on to bowtie2. You can put all aditional arguments in one Character(e.g. "–threads 8 –no-mixed") with white space splited just like command line, or put them as Character vector (e.g. c("–threads","8","–no-mixed")). Note that some arguments("-x","–interleaved","-U","-1","-2","-S","threads") to the bowtie2 are invalid if they are already handled as explicit function arguments. See the output of bowtie2_usage() for details about available parameters.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacRemoveAdapter removeAdapter bowtie2 bowtie2_build bowtie2_usage atacSam2Bam atacSamToBed atacLibComplexQC

Examples

```
td <- tempdir()
setTmpDir(td)

## Building a bowtie2 index
library("Rbowtie2")
refs <- dir(system.file(package="esATAC", "extdata", "bt2","refs"),
full=TRUE)
bowtie2_build(references=refs, bt2Index=file.path(td, "lambda_virus"),
"--threads 4 --quiet",overwrite=TRUE)
## Alignments
reads_1 <- system.file(package="esATAC", "extdata", "bt2", "reads",
"reads_1.fastq")
reads_2 <- system.file(package="esATAC", "extdata", "bt2", "reads",
"reads_2.fastq")
if(file.exists(file.path(td, "lambda_virus.1.bt2"))){
    (bowtie2Mapping(bt2Idx = file.path(td, "lambda_virus"),
       samOutput = file.path(td, "result.sam"),
       fastqInput1=reads_1,fastqInput2=reads_2,threads=3))
    head(readLines(file.path(td, "result.sam")))
}
```

CutSiteCountR *Count cut site number in given motif region and plot footprint.*

Description

This function is used to count cut site number in given motif regions and plot footprint. Multi-motif is supported. NOTE: The input parameter is a a little bit complex, `atacExtractCutSite` and `atacMotifScan` is recommended to use which makes the entire procedure easier.

Usage

```
atacCutSiteCount(
  atacProcCutSite,
  atacProcMotifScan = NULL,
  csInput = NULL,
  motif_info = NULL,
  chr = c(1:22, "X", "Y"),
  matrixOutput = NULL,
  strandLength = 100,
  FootPrint = TRUE,
  prefix = NULL,
  ...
)

## S4 method for signature 'ATACProc'
atacCutSiteCount(
```

```
  atacProcCutSite,
  atacProcMotifScan = NULL,
  csInput = NULL,
  motif_info = NULL,
  chr = c(1:22, "X", "Y"),
  matrixOutput = NULL,
  strandLength = 100,
  FootPrint = TRUE,
  prefix = NULL,
  ...
)

cutsitecount(
  csInput = NULL,
  motif_info = NULL,
  chr = c(1:22, "X", "Y"),
  matrixOutput = NULL,
  strandLength = 100,
  FootPrint = TRUE,
  prefix = NULL,
  ...
)
```

Arguments

atacProcCutSite	ATACProc-class object scalar. It has to be the return value of upstream process: atacExtractCutSite.
atacProcMotifScan	ATACProc-class object scalar. It has to be the return value of upstream process: atacMotifScan.
csInput	Your cut site information file(from atacExtractCutSite function, separated by chromatin name and all cut site are sorted) path with prefix. e.g. "/your_cut_site_information_path/prefix"
motif_info	A rds file from function atacMotifScan. In the rds file, it saves 3 column information(motif, motif exact position information file path and motif length).
chr	Which chromatin the program will processing. It must be identical with the filename of cut site information files or subset of . Default:c(1:22, "X", "Y").
matrixOutput	The output directory, where to save your cut site count of every motif position. an empty folder would be great. Default:tmpdir/Footprint
strandLength	How many bp(base pair) do you want to count up/downstream of the motif. default:100.
FootPrint	TRUE or FALSE, plot footprint or not.
prefix	prefix for the pdf file.
...	Additional arguments, currently unused.

Details

The parameter is simplified because of too many input file. parameter atacProcCutSite and atacProcMotifScan contains all input information so function atacExtractCutSite and atacMotifScan is recommended to use together. For instance, if you want footprint of 3 TFs (transcription factor) of human in chr1-22, X, Y, then you need 24 chromatin cut site files, 3 motif position files as well as 3 integers of the motif. Function atacExtractCutSite and atacMotifScan will do all this, you just specify which motif you want. Therefore, atacExtractCutSite and atacMotifScan is recommended to use together.

Value

An invisible ATACProc-class object scalar.

Author(s)

Wei Zhang

See Also

atacExtractCutSite atacMotifScan

Examples

```
library(R.utils)
library(BSgenome.Hsapiens.UCSC.hg19)
## processing bed file
fra_path <- system.file("extdata", "chr20.50000.bed.bz2", package="esATAC")
frag <- as.vector(bunzip2(filename = fra_path,
destname = file.path(getwd(), "chr20.50000.bed"),
ext="bz2", FUN=bzfile, overwrite=TRUE, remove = FALSE))
cs.data <- extractcutsite(bedInput = frag, prefix = "ATAC")

## find motif position
p1bz <- system.file("extdata", "Example_peak1.bed.bz2", package="esATAC")
peak1_path <- as.vector(bunzip2(filename = p1bz,
destname = file.path(getwd(), "Example_peak1.bed"),
ext="bz2", FUN = bzfile, overwrite=TRUE, remove = FALSE))
# motif <- readRDS(system.file("extdata", "MotifPFM.rds", package="esATAC"))
# motif.data <- motifscan(peak = peak1_path, genome = BSgenome.Hsapiens.UCSC.hg19, motifs = motif)

## plot footprint
# atacCutSiteCount(atacProcCutSite = cs.data, atacProcMotifScan = motif.data)
```

CutSitePre *Extract ATAC-seq cutting site from bed file.*

Description

Extract cutting site from ATAC-seq fangment bed file (from atacSamToBed).

Usage

```
atacExtractCutSite(
  atacProc,
  bedInput = NULL,
  csOutput.dir = NULL,
  prefix = NULL,
  ...
)

## S4 method for signature 'ATACProc'
atacExtractCutSite(
  atacProc,
  bedInput = NULL,
  csOutput.dir = NULL,
  prefix = NULL,
  ...
)

extractcutsite(bedInput, csOutput.dir = NULL, prefix = NULL, ...)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacSamToBed.
bedInput	Character scalar. Input bed file path, must be merged bed file(a line is a fragment). The input file should be UCSC bed format(0-based).
csOutput.dir	Character scalar. The output path, an empty folder would be great. Default: a folder in the same path as input bed file.
prefix	Character scalar. Output file name prefix, e.g. prefix_chr*.bed, default "Cutsite".
...	Additional arguments, currently unused.

Details

In ATAC-seq data, every line in merged bed file (from atacSamToBed, the first 3 column is chr, start, end) means a DNA fragment, the cutting site is start+1 and end, this function extract and sort this information for the next step (atacCutSiteCount).

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Wei Zhang

See Also

atacCutSiteCount

Examples

```
library(R.utils)
fra_path <- system.file("extdata", "chr20.50000.bed.bz2", package="esATAC")
frag <- as.vector(bunzip2(filename = fra_path,
destname = file.path(getwd(), "chr20.50000.bed"),
ext="bz2", FUN=bzfile, overwrite=TRUE, remove = FALSE))
extractcutsite(bedInput = frag, prefix = "ATAC")
```

FastQC *Quality control for ATAC-seq data.*

Description

Generate quality control plots from fastq of ATAC-seq data.

Usage

```
atacQCReport(atacProc, input_file = NULL, output_file = NULL, ...)

## S4 method for signature 'ATACProc'
atacQCReport(atacProc, input_file = NULL, output_file = NULL, ...)

qcreport(input_file, output_file = NULL, ...)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacUnzipAndMerge, atacRenamer
input_file	Character scalar. Input file path. One or more(vector) fastq file path.
output_file	Character scalar. output file path. Defult:"input_file_QC.pdf" in the same folder as your input file.
...	Additional arguments, currently unused.

Details

Every highthroughput sequencing need quality control analysis, this function provide QC for ATAC-seq, such as GC content.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Wei Zhang

See Also

atacUnzipAndMerge, atacRenamer

Examples

```
library(R.utils)
fra_path <- system.file("extdata", "chr20_1.2.fq.bz2", package="esATAC")
fq1 <- as.vector(bunzip2(filename = fra_path,
destname = file.path(getwd(), "chr20_1.fq"),
ext="bz2", FUN=bzfile, overwrite=TRUE, remove = FALSE))
fra_path <- system.file("extdata", "chr20_2.2.fq.bz2", package="esATAC")
fq2 <- as.vector(bunzip2(filename = fra_path,
destname = file.path(getwd(), "chr20_2.fq"),
ext="bz2", FUN=bzfile, overwrite=TRUE, remove = FALSE))
## Not run:
qcreport(input_file = c(fq1, fq2))

## End(Not run)
```

FindAdapter *Use AdapterRemoval to identify adapters*

Description

Use AdapterRemoval to identify adapters for paired end data

Usage

```
atacFindAdapter(
  atacProc,
  fastqInput1 = NULL,
  fastqInput2 = NULL,
  reportPrefix = NULL,
```

```
  interleave = FALSE,
  findParamList = NULL,
  threads = getThreads(),
  ...
)

## S4 method for signature 'ATACProc'
atacFindAdapter(
  atacProc,
  fastqInput1 = NULL,
  fastqInput2 = NULL,
  reportPrefix = NULL,
  interleave = FALSE,
  findParamList = NULL,
  threads = getThreads(),
  ...
)

findAdapter(
  fastqInput1,
  fastqInput2 = NULL,
  reportPrefix = NULL,
  interleave = FALSE,
  findParamList = NULL,
  threads = getThreads(),
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacRenamer renamer atacUnzipAndMerge unzipAndMerge
fastqInput1	Character vector. For single-end sequencing, it contains sequence file paths. For paired-end sequencing, it can be file paths with #1 mates paired with file paths in fastqInput2 And it can also be interleaved file paths when argument interleaved=TRUE
fastqInput2	Character vector. It contains file paths with #2 mates paired with file paths in fastqInput1 For single-end sequencing files and interleaved paired-end sequencing files(argument interleaved=TRUE), it must be NULL.
reportPrefix	Character. The prefix of report files path. Default: generate from known parameters
interleave	Logical. Set TRUE when files are interleaved paired-end sequencing data.
findParamList	Additional arguments to be passed on to the binaries for identifying adapter. See below for details.
threads	The number of threads used in this step.
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use findAdapter instead. You can put all aditional arguments in one Character(e.g. "–threads 8") with white space splited just like command line, or put them in Character vector(e.g. c("–threads","8")). Note that some arguments("–file1","–file2","–adapter1","–adapter2","–output1","–output2", "–basename","–interleaved","thread") to the findParamList are invalid if they are already handled as explicit function arguments. See the output of adapterremoval_usage() for details about available parameters.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacRenamer renamer atacUnzipAndMerge unzipAndMerge atacBowtie2Mapping

Examples

```
library(magrittr)
td <- tempdir()
setTmpDir(td)

# Identify adapters
prefix<-system.file(package="esATAC", "extdata", "uzmg")
(reads_1 <-file.path(prefix,"m1",dir(file.path(prefix,"m1"))))
(reads_2 <-file.path(prefix,"m2",dir(file.path(prefix,"m2"))))

reads_merged_1 <- file.path(td,"reads1.fastq")
reads_merged_2 <- file.path(td,"reads2.fastq")
atacproc <-
atacUnzipAndMerge(fastqInput1 = reads_1,fastqInput2 = reads_2) %>%
atacRenamer %>% atacFindAdapter

dir(td)
```

FragLenDistr *Quality control for fragment length distribution*

Description

These functions are used to generate fragment distribution plot. The fourier transform of fragment distribution will be calculated. Strength distribution around period at 10.4bp and 180bp will be shown in another two plots.

Usage

```
atacFragLenDistr(atacProc, reportPrefix = NULL, bedInput = NULL, ...)

## S4 method for signature 'ATACProc'
atacFragLenDistr(atacProc, reportPrefix = NULL, bedInput = NULL, ...)

fragLenDistr(bedInput, reportPrefix = NULL, ...)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacSamToBed samToBed atacBedUtils bedUtils
reportPrefix	Character scalar. The prefix of report files path.
bedInput	Character scalar. BED file input path.
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use fragLenDistr instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacSamToBed samToBed atacBedUtils bedUtils

Examples

```
library(R.utils)
td <- tempdir()
setTmpDir(td)

bedbzfile <- system.file(package="esATAC", "extdata", "chr20.50000.bed.bz2")
bedfile <- file.path(td,"chr20.50000.bed")
## Not run:
bunzip2(bedbzfile,destname=bedfile,overwrite=TRUE,remove=FALSE)
fragLenDistr(bedfile)

## End(Not run)
```

```
dir(td)
```

FRiPQC *Quality control for fraction of reads in peaks (FRiP)*

Description

Calculate the fraction of reads falling within peak regions

Usage

```
atacFripQC(
  atacProc,
  atacProcPeak = NULL,
  bsgenome = NULL,
  reportOutput = NULL,
  readsBedInput = NULL,
  peakBedInput = NULL,
  ...
)

## S4 method for signature 'ATACProc'
atacFripQC(
  atacProc,
  atacProcPeak = NULL,
  bsgenome = NULL,
  reportOutput = NULL,
  readsBedInput = NULL,
  peakBedInput = NULL,
  ...
)

fripQC(readsBedInput, peakBedInput, bsgenome = NULL, reportOutput = NULL, ...)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacSamToBed samToBed atacBedUtils bedUtils
atacProcPeak	ATACProc-class object scalar. It has to be the return value of upstream process: atacPeakCalling, peakCalling.
bsgenome	BSGenome object scalar. BSGenome object for specific species.
reportOutput	Character scalar. The report file path
readsBedInput	Character scalar. Reads BED file for peak calling.
peakBedInput	Character scalar. Peaks BED file
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, or you can use fripQC instead.

Value

An invisible fripQC object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacSamToBed atacBedUtils

Examples

```
library(R.utils)
library(BSgenome.Hsapiens.UCSC.hg19)
library(magrittr)
td <- tempdir()
setTmpDir(td)

bedbzfile <- system.file(package="esATAC", "extdata", "chr20.50000.bed.bz2")
bedfile <- file.path(td,"chr20.50000.bed")
bunzip2(bedbzfile,destname=bedfile,overwrite=TRUE,remove=FALSE)

bedUtils(bedInput = bedfile,maxFragLen = 100, chrFilterList = NULL)  %>%
atacPeakCalling %>% atacFripQC(bsgenome=BSgenome.Hsapiens.UCSC.hg19)


dir(td)
```

---

getMotifInfo *Generate PFMatrix or PFMatrixList from file.*

Description

atacMotifScan and atacMotifScanPair accept PFM in a list, this function convert JASPAR PFM file to PFMatrix or PFMatrixList.

Usage

```
getMotifInfo(motif.file = NULL)
```

Arguments

motif.file	Motif PFM file downloaded from JASPAR.

Details

Generate PFMatrix or PFMatrixList.

Value

PFMatrix or PFMatrixList.

Author(s)

Wei Zhang

Examples

```
motif_file <- system.file("extdata", "CustomizedMotif.txt", package="esATAC")
pfm <- getMotifInfo(motif.file = motif_file)
```

LibComplexQC *Quality control for library complexity*

Description

The function calculate the nonredundant fraction of reads (NRF). Its definition is number of distinct uniquely mapping reads (i.e. after removing duplicates) / Total number of reads. The function also Calculate PCR Bottlenecking Coefficient 1 (PBC1) and PCR Bottlenecking Coefficient 2 (PBC2). PBC1=M1/M_DISTINCT and PBC2=M1/M2, where M1: number of genomic locations where exactly one read maps uniquely, M2: number of genomic locations where two reads map uniquely M_DISTINCT: number of distinct genomic locations to which some read maps uniquely.

Usage

```
atacLibComplexQC(
  atacProc,
  reportOutput = NULL,
  samInput = NULL,
  singleEnd = FALSE,
  subsampleSize = Inf,
  ...
)
```

```
## S4 method for signature 'ATACProc'
atacLibComplexQC(
  atacProc,
  reportOutput = NULL,
  samInput = NULL,
  singleEnd = FALSE,
  subsampleSize = Inf,
  ...
)

libComplexQC(
  samInput,
  reportOutput = NULL,
  singleEnd = FALSE,
  subsampleSize = Inf,
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacBowtie2Mapping bowtie2Mapping
reportOutput	Character scalar. The report file path
samInput	Character scalar. The SAM file input path.
singleEnd	Character scalar. Single end data if TRUE. Paired end data if FALSE.
subsampleSize	Integer scalar. Down sample reads if the number is less than total number when subsample is TRUE
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use libComplexQC instead.

Value

An invisible libComplexQC object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacBowtie2Mapping bowtie2Mapping

Examples

```
library(R.utils)
td <- tempdir()
setTmpDir(td)

sambzfile <- system.file(package="esATAC", "extdata", "Example.sam.bz2")
samfile <- file.path(td,"Example.sam")
bunzip2(sambzfile,destname=samfile,overwrite=TRUE,remove=FALSE)
atacproc<-libComplexQC(samInput = samfile)
```

PeakCallingFseq *Use F-seq to call peak*

Description

Use F-seq to call peak

Usage

```
atacPeakCalling(
  atacProc,
  bedInput = NULL,
  background = NULL,
  genomicReadsCount = NULL,
  fragmentSize = 0,
  featureLength = NULL,
  bedOutput = NULL,
  ploidyDir = NULL,
  fileformat = c("bed", "wig", "npf"),
  wiggleTrackStep = NULL,
  threshold = NULL,
  verbose = TRUE,
  wgThresholdSet = NULL,
  ...
)

## S4 method for signature 'ATACProc'
atacPeakCalling(
  atacProc,
  bedInput = NULL,
  background = NULL,
  genomicReadsCount = NULL,
  fragmentSize = 0,
  featureLength = NULL,
  bedOutput = NULL,
  ploidyDir = NULL,
```

```
  fileformat = c("bed", "wig", "npf"),
  wiggleTrackStep = NULL,
  threshold = NULL,
  verbose = TRUE,
  wgThresholdSet = NULL,
  ...
)

peakCalling(
  bedInput,
  background = NULL,
  genomicReadsCount = NULL,
  fragmentSize = 0,
  featureLength = NULL,
  bedOutput = NULL,
  ploidyDir = NULL,
  fileformat = c("bed", "wig", "npf"),
  wiggleTrackStep = NULL,
  threshold = NULL,
  verbose = TRUE,
  wgThresholdSet = NULL,
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacSamToBed, atacBedUtils.
bedInput	Character scalar. BED file input path.
background	Character scalar. background directory default: NULL (none)
genomicReadsCount	Integer scalar. genomic count of sequence reads. default: NULL (calculated)
fragmentSize	Integer scalar. fragment size. set NULL to estimat from data. default:0
featureLength	Character scalar. feature length default: NULL (600)
bedOutput	Character scalar. the output bed file path
ploidyDir	Character scalar. ploidy/input directory. default: NULL
fileformat	Character scalar. File format of result. default: bed
wiggleTrackStep	Integer scalar. wiggle track step default: NULL (1)
threshold	Numeric scalar. threshold (standard deviations) default: NULL (4.0)
verbose	Logical scalar. verbose output if TRUE.
wgThresholdSet	Character scalar. wg threshold set default: NULL (calculated)
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use peakCalling instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacSamToBed samToBed atacBedUtils bedUtils

Examples

```
library(R.utils)
library(magrittr)
td <- tempdir()
setTmpDir(td)

bedbzfile <- system.file(package="esATAC", "extdata", "chr20.50000.bed.bz2")
bedfile <- file.path(td,"chr20.50000.bed")
bunzip2(bedbzfile,destname=bedfile,overwrite=TRUE,remove=FALSE)

bedUtils(bedInput = bedfile,maxFragLen = 100, chrFilterList = NULL) %>%
atacPeakCalling

dir(td)
```

PeakCallingMACS2 *Use MACS2 to call peak*

Description

Use MACS2 installed by to call peak

Usage

```
atacPeakCallingMACS2(
  atacProc,
  bedInput = NULL,
  background = NULL,
  outputPrefix = NULL,
```

```
  genomeSize = NULL,
  pvalueThreshold = 0.01,
  extsize = 150,
  shift = -round(extsize/2),
  ...
)

## S4 method for signature 'ATACProc'
atacPeakCallingMACS2(
  atacProc,
  bedInput = NULL,
  background = NULL,
  outputPrefix = NULL,
  genomeSize = NULL,
  pvalueThreshold = 0.01,
  extsize = 150,
  shift = -round(extsize/2),
  ...
)

peakCallingMACS2(
  bedInput,
  background = NULL,
  outputPrefix = NULL,
  genomeSize = NULL,
  pvalueThreshold = 0.01,
  extsize = 150,
  shift = -round(extsize/2),
  ...
)

testPeakCallingMACS2()
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacSamToBed, atacBedUtils.
bedInput	Character scalar. BED file input path.
background	Character scalar. background directory default: NULL (none)
outputPrefix	Character scalar. the output bed file path
extsize	Logical scalar. verbose output if TRUE.
shift	Character scalar. wg threshold set default: NULL (calculated)
...	Additional arguments, currently unused.
genomeSize	Numeric scalar. the size of genome
pvalueThreshold	Numeric scalar. threshold (standard deviations) default: NULL (4.0)

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use peakCalling instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacSamToBed samToBed atacBedUtils bedUtils

Examples

```
library(R.utils)
library(magrittr)
td <- tempdir()
setTmpDir(td)

bedbzfile <- system.file(package="esATAC", "extdata", "chr20.50000.bed.bz2")
bedfile <- file.path(td,"chr20.50000.bed")
bunzip2(bedbzfile,destname=bedfile,overwrite=TRUE,remove=FALSE)

bedUtils(bedInput = bedfile,maxFragLen = 100, chrFilterList = NULL) %>%
atacPeakCalling

dir(td)
```

PeakQC *Quality control for peak overlap*

Description

These functions are used to calculate the overlap ratio in specific quality control rigion. Blacklist and DHS region are provided. You can also set your own BED file as quality control rigion.

Usage

```
atacPeakQC(
  atacProc,
  bsgenome = NULL,
  reportOutput = NULL,
```

```
  qcbedInput = c("DHS", "blacklist", "path/to/bed"),
  bedInput = NULL,
  newStepType = "PeakQC",
  ...
)

## S4 method for signature 'ATACProc'
atacPeakQC(
  atacProc,
  bsgenome = NULL,
  reportOutput = NULL,
  qcbedInput = c("DHS", "blacklist", "path/to/bed"),
  bedInput = NULL,
  newStepType = "PeakQC",
  ...
)

peakQC(
  bedInput,
  bsgenome = NULL,
  reportOutput = NULL,
  qcbedInput = c("DHS", "blacklist", "path/to/bed"),
  newStepType = "PeakQC",
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacSamToBed, atacBedUtils.
bsgenome	BSGenome object scalar. BSGenome object for specific species.
reportOutput	Character scalar. The report file path.
qcbedInput	Character scalar. It can be "DHS","blacklist" or Other quality control BED file input path.
bedInput	Character scalar. BED file input path for quality control.
newStepType	Character scalar. New step type name for different default parameters.
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use peakQC instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacSamToBed atacBedUtils

Examples

```
library(R.utils)
library(magrittr)
td <- tempdir()
setTmpDir(td)

bedbzfile <- system.file(package="esATAC", "extdata", "chr20.50000.bed.bz2")
bedfile <- file.path(td,"chr20.50000.bed")
bunzip2(bedbzfile,destname=bedfile,overwrite=TRUE,remove=FALSE)
blacklistfile <- system.file(package="esATAC", "extdata", "hg19.blacklist.bed")
library(BSgenome.Hsapiens.UCSC.hg19)
bedUtils(bedInput = bedfile,maxFragLen = 100, chrFilterList = NULL) %>%
atacPeakCalling %>% atacPeakQC(qcbedInput = blacklistfile, bsgenome = BSgenome.Hsapiens.UCSC.hg19)
dir(td)
```

RemoveAdapter *Use AdapterRemoval to remove adapters*

Description

Use AdapterRemoval to remove adapters

Usage

```
atacRemoveAdapter(
  atacProc,
  adapter1 = NULL,
  adapter2 = NULL,
  fastqOutput1 = NULL,
  reportPrefix = NULL,
  fastqOutput2 = NULL,
  fastqInput1 = NULL,
  fastqInput2 = NULL,
  interleave = FALSE,
  threads = getThreads(),
  paramList = NULL,
  findParamList = NULL,
  ...
)
```

```
## S4 method for signature 'ATACProc'
atacRemoveAdapter(
  atacProc,
  adapter1 = NULL,
  adapter2 = NULL,
  fastqOutput1 = NULL,
  reportPrefix = NULL,
  fastqOutput2 = NULL,
  fastqInput1 = NULL,
  fastqInput2 = NULL,
  interleave = FALSE,
  threads = getThreads(),
  paramList = NULL,
  findParamList = NULL,
  ...
)

removeAdapter(
  fastqInput1,
  fastqInput2,
  adapter1,
  adapter2,
  fastqOutput1 = NULL,
  reportPrefix = NULL,
  fastqOutput2 = NULL,
  interleave = FALSE,
  threads = getThreads(),
  paramList = NULL,
  findParamList = NULL,
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacRenamer renamer atacUnzipAndMerge unzipAndMerge
adapter1	Character. It is an adapter sequence for file1. For single end data, it is requied.
adapter2	Character. It is an adapter sequence for file2.
fastqOutput1	Character. The trimmed mate1 reads output file path for fastqInput2. Defualt: basename.pair1.truncated (paired-end), basename.truncated (single-end), or basename.paired.truncated (interleaved)
reportPrefix	Character. The prefix of report files path. Default: generate from known parameters
fastqOutput2	Character. The trimmed mate2 reads output file path for fastqInput2. Default: BASENAME.pair2.truncated (only used in PE mode, but not if –interleaved-output is enabled)

fastqInput1	Character vector. For single-end sequencing, it contains sequence file paths. For paired-end sequencing, it can be file paths with #1 mates paired with file paths in fastqInput2 And it can also be interleaved file paths when argument interleaved=TRUE
fastqInput2	Character vector. It contains file paths with #2 mates paired with file paths in fastqInput1 For single-end sequencing files and interleaved paired-end sequencing files(argument interleaved=TRUE), it must be NULL.
interleave	Logical. Set TRUE when files are interleaved paired-end sequencing data.
threads	Numeric. The max threads allowed to be used by this step. Default: getThreads()
paramList	Additional arguments to be passed on to the binaries for removing adapter. See below for details.
findParamList	Additional arguments to be passed on to the binaries for identifying adapter. See below for details.
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use removeAdapter instead. You can put all aditional arguments in one Character(e.g. "–threads 8") with white space splited just like command line, or put them in Character vector(e.g. c("–threads","8")). Note that some arguments("–file1","–file2","–adapter1","–adapter2","–output1","–output2", "–basename","–interleaved","thread") to the paramList and findParamList are invalid if they are already handled as explicit function arguments. See the output of adapterremoval_usage() for details about available parameters.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacRenamer renamer atacUnzipAndMerge unzipAndMerge atacBowtie2Mapping

Examples

```
library(magrittr)
td <- tempdir()
setTmpDir(td)

# Identify adapters
prefix<-system.file(package="esATAC", "extdata", "uzmg")
```

```
(reads_1 <-file.path(prefix,"m1",dir(file.path(prefix,"m1"))))
(reads_2 <-file.path(prefix,"m2",dir(file.path(prefix,"m2"))))

reads_merged_1 <- file.path(td,"reads1.fastq")
reads_merged_2 <- file.path(td,"reads2.fastq")
atacproc <-
atacUnzipAndMerge(fastqInput1 = reads_1,fastqInput2 = reads_2) %>%
atacRenamer %>% atacFindAdapter %>% atacRemoveAdapter

dir(td)
```

Renamer *Rename reads name in fastq*

Description

Rename reads name in fastq with increasing integer

Usage

```
atacRenamer(
  atacProc,
  fastqOutput1 = NULL,
  fastqOutput2 = NULL,
  fastqInput1 = NULL,
  fastqInput2 = NULL,
  interleave = FALSE,
  threads = getThreads(),
  ...
)

## S4 method for signature 'ATACProc'
atacRenamer(
  atacProc,
  fastqOutput1 = NULL,
  fastqOutput2 = NULL,
  fastqInput1 = NULL,
  fastqInput2 = NULL,
  interleave = FALSE,
  threads = getThreads(),
  ...
)

renamer(
  fastqInput1 = NULL,
  fastqInput2 = NULL,
  fastqOutput1 = NULL,
  fastqOutput2 = NULL,
```

```
  interleave = FALSE,
  threads = getThreads(),
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacUnzipAndMerge unzipAndMerge
fastqOutput1	Character scalar. The output file path of renamed fastqInput1.
fastqOutput2	Character scalar. The output file path of renamed fastqInput2.
fastqInput1	Character scalar. For single-end sequencing, it contains sequence file paths. For paired-end sequencing, it can be file path with #1 mates paired with file path in file2 And it can also be interleaved file paths when argument interleave=TRUE
fastqInput2	Character scalar. It contains file path with #2 mates paired with file paths in fastqInput1 For single-end sequencing files and interleaved paired-end sequencing files(argument interleaved=TRUE), it must be NULL.
interleave	Character scalar. Set TRUE when files are interleaved paired-end sequencing data.
threads	Integer scalar. The threads will be created in this process. default: 1
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use renamer instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacUnzipAndMerge unzipAndMerge atacQCReport atacRemoveAdapter

Examples

```
ignoreCheck() # warnning: run this for fast test only
library(magrittr)
td <- tempdir()
setTmpDir(td)
```

```
# Identify adapters
prefix<-system.file(package="esATAC", "extdata", "uzmg")
(reads_1 <-file.path(prefix,"m1",dir(file.path(prefix,"m1"))))
(reads_2 <-file.path(prefix,"m2",dir(file.path(prefix,"m2"))))

reads_merged_1 <- file.path(td,"reads1.fastq")
reads_merged_2 <- file.path(td,"reads2.fastq")
atacproc <-
atacUnzipAndMerge(fastqInput1 = reads_1,fastqInput2 = reads_2) %>%
atacRenamer

dir(td)
```

RGo *Gene Ontology Analysis*

Description

Ranking functional groups based on a set of genes. For more information, please see enrichGO.

Usage

```
atacGOAnalysis(
  atacProc,
  gene = NULL,
  OrgDb = NULL,
  keytype = "ENTREZID",
  ont = "MF",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  universe = NULL,
  qvalueCutoff = 0.2,
  readable = FALSE,
  pool = FALSE,
  goOutput = NULL,
  ...
)

## S4 method for signature 'ATACProc'
atacGOAnalysis(
  atacProc,
  gene = NULL,
  OrgDb = NULL,
  keytype = "ENTREZID",
  ont = "MF",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
```

```
  universe = NULL,
  qvalueCutoff = 0.2,
  readable = FALSE,
  pool = FALSE,
  goOutput = NULL,
  ...
)

goanalysis(
  gene,
  OrgDb = NULL,
  keytype = "ENTREZID",
  ont = "MF",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  universe = NULL,
  qvalueCutoff = 0.2,
  readable = FALSE,
  pool = FALSE,
  goOutput = NULL,
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacPeakAnno.
gene	A vector of entrez gene id.
OrgDb	Genome wide annotation databese.
keytype	Keytype of input gene.
ont	One of "MF", "BP", and "CC" subontologies. "MF" for molecular function, "BP" for biological process, "CC" for cellular component.
pvalueCutoff	pvalueCutoff.
pAdjustMethod	One of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none".
universe	Background genes.
qvalueCutoff	qvalue cutoff.
readable	whether mapping gene ID to gene Name.
pool	If ont='ALL', whether pool 3 GO sub-ontologies.
goOutput	Character scalar. Output file path. Defult:in the same folder as your input file with the suffix "df".
...	Additional arguments, currently unused.

Details

This function using enrichGO to do GO analysis but fixed some parameters. If atacProc is not NULL, it will read the gene ID from the output of atacPeakAnno.

Value

An invisible ATACProc-class object scalar.

Author(s)

Wei Zhang

References

Guangchuang Yu., Li-Gen Wang, Yanyan Han, Qing-Yu He. clusterProfiler: an R package for comparing biological themes among gene clusters. OMICS: A Journal of Integrative Biology. 2012, 16(5):284-287

See Also

atacPeakAnno enrichGO function enrichGO in package "clusterProfiler"

Examples

```
## Not run:
library(org.Hs.eg.db)
# generate simulated geneID
geneId <- as.character(sample(seq(10000), 100))
goanalysis(gene = geneId, OrgDb = 'org.Hs.eg.db')

## End(Not run)
```

RMotifScan *Search Motif Position in Given Regions*

Description

Search motif position in genome according thr given motif and peak information.

Usage

```
atacMotifScan(
  atacProc,
  peak = NULL,
  genome = NULL,
  motifs = NULL,
  p.cutoff = 1e-06,
  scanO.dir = NULL,
  prefix = NULL,
  ...
)
```

```
## S4 method for signature 'ATACProc'
atacMotifScan(
  atacProc,
  peak = NULL,
  genome = NULL,
  motifs = NULL,
  p.cutoff = 1e-06,
  scanO.dir = NULL,
  prefix = NULL,
  ...
)

motifscan(
  peak = NULL,
  genome = NULL,
  motifs = NULL,
  p.cutoff = 1e-06,
  scanO.dir = NULL,
  prefix = NULL,
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacPeakCalling.
peak	Character scalar. Input region path. UCSC bed file is recommented. Other file should be able to import as GRanges objects through import in package rtracklayer.
genome	BSgenome object, Default: from getRefRc.
motifs	eitherPFMatrix, PFMatrixList, PWMatrix, PWMatrixList.
p.cutoff	p-value cutoff for returning motifs.
scanO.dir	Character scalar. the output file directory. This function will use the name in motifs as the file name to save the motif position information in separate files.
prefix	prefix for Output file.
...	Additional arguments, currently unused.

Details

This function scan motif position in a given genome regions.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Wei Zhang

See Also

atacPeakCalling atacCutSiteCount

Examples

```
## Not run:
library(R.utils)
library(BSgenome.Hsapiens.UCSC.hg19)
peak.path <- system.file("extdata", "Example_peak1.bed.bz2", package="esATAC")
peak.path <- as.vector(bunzip2(filename = peak.path, destname = file.path(getwd(), "Example_peak1.bed"), ext="bz2

motif <- readRDS(system.file("extdata", "MotifPFM.rds", package="esATAC"))

motifscan(peak = peak.path, genome = BSgenome.Hsapiens.UCSC.hg19, motifs = motif)

## End(Not run)
```

RMotifScanPair *Search Motif Position in Given Regions*

Description

Search motif position in genome according thr given motif and peak information.

Usage

```
atacMotifScanPair(
  atacProc,
  peak1 = NULL,
  peak2 = NULL,
  background = NULL,
  genome = NULL,
  motifs = NULL,
  p.cutoff = 1e-04,
  scanO.dir = NULL,
  prefix = NULL,
  ...
)

## S4 method for signature 'ATACProc'
atacMotifScanPair(
  atacProc,
  peak1 = NULL,
  peak2 = NULL,
  background = NULL,
  genome = NULL,
```

```
  motifs = NULL,
  p.cutoff = 1e-04,
  scanO.dir = NULL,
  prefix = NULL,
  ...
)

motifscanpair(
  peak1 = NULL,
  peak2 = NULL,
  background = NULL,
  genome = NULL,
  motifs = NULL,
  p.cutoff = 1e-04,
  scanO.dir = NULL,
  prefix = NULL,
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacPeakComp.
peak1	peak file path.
peak2	peak file path.
background	background peak file path.
genome	BSgenome object, Default: from getRefRc.
motifs	eitherPFMatrix, PFMatrixList, PWMatrix, PWMatrixList.
p.cutoff	p-value cutoff for returning motifs.
scanO.dir	Character scalar. the output file directory. This function will use the name in motifs as the file name to save the motif position information in separate files.
prefix	prefix for Output file. Order: peak1, peak2, backgroud.
...	Additional arguments, currently unused.

Details

This function scan motif position in a given genome regions.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Wei Zhang

See Also

atacPeakComp

Examples

```
## Not run:
library(R.utils)
library(BSgenome.Hsapiens.UCSC.hg19)
p1bz <- system.file("extdata", "Example_peak1.bed.bz2", package="esATAC")
p2bz <- system.file("extdata", "Example_peak2.bed.bz2", package="esATAC")
peak1_path <- as.vector(bunzip2(filename = p1bz,
destname = file.path(getwd(), "Example_peak1.bed"),
ext="bz2", FUN=bzfile, overwrite=TRUE , remove = FALSE))
peak2_path <- as.vector(bunzip2(filename = p2bz,
destname = file.path(getwd(), "Example_peak2.bed"),
ext="bz2", FUN=bzfile, overwrite=TRUE, remove = FALSE))
peakcom.output <- peakcomp(bedInput1 = peak1_path, bedInput2 = peak2_path,
olap.rate = 0.1)

motif <- readRDS(system.file("extdata", "MotifPFM.rds", package="esATAC"))
output <- atacMotifScanPair(atacProc = peakcom.output,
genome = BSgenome.Hsapiens.UCSC.hg19, motifs = motif)

## End(Not run)
```

RPeakAnno *Annotate ATAC-seq Peak*

Description

This function annotates ATAC-seq peak by a given annotation database. For more information, please see annotatePeak.

Usage

```
atacPeakAnno(
  atacProc,
  peakInput = NULL,
  tssRegion = c(-1000, 1000),
  TxDb = NULL,
  level = "transcript",
  genomicAnnotationPriority = c("Promoter", "5UTR", "3UTR", "Exon", "Intron",
    "Downstream", "Intergenic"),
  annoDb = NULL,
  addFlankGeneInfo = FALSE,
  flankDistance = 5000,
  sameStrand = FALSE,
```

```
  ignoreOverlap = FALSE,
  ignoreUpstream = FALSE,
  ignoreDownstream = FALSE,
  overlap = "TSS",
  annoOutput = NULL,
  ...
)

## S4 method for signature 'ATACProc'
atacPeakAnno(
  atacProc,
  peakInput = NULL,
  tssRegion = c(-1000, 1000),
  TxDb = NULL,
  level = "transcript",
  genomicAnnotationPriority = c("Promoter", "5UTR", "3UTR", "Exon", "Intron",
    "Downstream", "Intergenic"),
  annoDb = NULL,
  addFlankGeneInfo = FALSE,
  flankDistance = 5000,
  sameStrand = FALSE,
  ignoreOverlap = FALSE,
  ignoreUpstream = FALSE,
  ignoreDownstream = FALSE,
  overlap = "TSS",
  annoOutput = NULL,
  ...
)

peakanno(
  peakInput,
  tssRegion = c(-1000, 1000),
  TxDb = NULL,
  level = "transcript",
  genomicAnnotationPriority = c("Promoter", "5UTR", "3UTR", "Exon", "Intron",
    "Downstream", "Intergenic"),
  annoDb = NULL,
  addFlankGeneInfo = FALSE,
  flankDistance = 5000,
  sameStrand = FALSE,
  ignoreOverlap = FALSE,
  ignoreUpstream = FALSE,
  ignoreDownstream = FALSE,
  overlap = "TSS",
  annoOutput = NULL,
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacPeakCalling.
peakInput	Character scalar. Input peak file path. UCSC bed file is recommented. Other file should be able to import as GRanges objects through import in package rtracklayer.
tssRegion	Region range of TSS, default:c(-1000, 1000).
TxDb	TxDb object, annotation database.
level	"transcript" or "gene".
genomicAnnotationPriority	genomic annotation priority.
annoDb	Gene annotation database.
addFlankGeneInfo	logical, add flanking gene information from the peaks.
flankDistance	distance of flanking sequence.
sameStrand	logical, whether find nearest/overlap gene in the same strand.
ignoreOverlap	logical, whether ignore overlap of TSS with peak.
ignoreUpstream	logical, if True only annotate gene at the 3' of the peak.
ignoreDownstream	logical, if True only annotate gene at the 5' of the peak.
overlap	one of 'TSS' or 'all', if overlap="all", then gene overlap with peak will be reported as nearest gene, no matter the overlap is at TSS region or not.
annoOutput	Character scalar. the output file path.
...	Additional arguments, currently unused.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Wei Zhang

References

Guangchuang Yu, Li-Gen Wang, Qing-Yu He. ChIPseeker: an R/Bioconductor package for ChIP peak annotation, comparison and visualization. Bioinformatics 2015, 31(14):2382-2383

See Also

atacPeakCalling atacGOAnalysis

Examples

```
library(R.utils)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
p1bz <- system.file("extdata", "Example_peak1.bed.bz2", package="esATAC")
peak1_path <- as.vector(bunzip2(filename = p1bz,
destname = file.path(getwd(), "Example_peak1.bed"),
ext="bz2", FUN=bzfile, overwrite=TRUE, remove = FALSE))
#peakanno(peakInput = peak1_path, TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene,
#annoDb = 'org.Hs.eg.db')
```

RPeakComp *Find the overlap or differential peaks between two samples.*

Description

This function compares two peak file and report overlap or differential peaks according to the parameter "operation".

Usage

```
atacPeakComp(
  atacProcPeak1,
  atacProcPeak2,
  bedInput1 = NULL,
  bedInput2 = NULL,
  bedOutput = NULL,
  olap.rate = 0.2,
  ...
)

## S4 method for signature 'ATACProc'
atacPeakComp(
  atacProcPeak1,
  atacProcPeak2,
  bedInput1 = NULL,
  bedInput2 = NULL,
  bedOutput = NULL,
  olap.rate = 0.2,
  ...
)

peakcomp(
  bedInput1 = NULL,
  bedInput2 = NULL,
  bedOutput = NULL,
```

```
  olap.rate = 0.2,
  ...
)
```

Arguments

atacProcPeak1	ATACProc-class object scalar. It has to be the return value of upstream process: atacPeakCalling.
atacProcPeak2	ATACProc-class object scalar. It has to be the return value of upstream process: atacPeakCalling.
bedInput1	Character scalar. Input peak file path. UCSC bed file is recommented. Other file should be able to import as GRanges objects through import in package rtracklayer.
bedInput2	Character scalar. Input peak file path. UCSC bed file is recommented. Other file should be able to import as GRanges objects through import in package rtracklayer.
bedOutput	The output file path for overlap peaks.
olap.rate	Overlap rate, if the overlap region between 2 peak is more than this rate of the short peak, these two peak are considered to be overlap and will be merged to a bigger peak. Default: 0.2. NOTICE: multi-peak will be merged together!
...	Additional arguments, currently unused.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Wei Zhang

See Also

atacPeakCalling

Examples

```
library(R.utils)
p1bz <- system.file("extdata", "Example_peak1.bed.bz2", package="esATAC")
p2bz <- system.file("extdata", "Example_peak2.bed.bz2", package="esATAC")
## Not run:
peak1_path <- as.vector(bunzip2(filename = p1bz,
destname = file.path(getwd(), "Example_peak1.bed"),
ext="bz2", FUN=bzfile, overwrite=TRUE , remove = FALSE))
peak2_path <- as.vector(bunzip2(filename = p2bz,
destname = file.path(getwd(), "Example_peak2.bed"),
ext="bz2", FUN=bzfile, overwrite=TRUE, remove = FALSE))
output <- peakcomp(bedInput1 = peak1_path, bedInput2 = peak2_path,
olap.rate = 0.1)

## End(Not run)
```

RSNPs *Find whether snps are in the given regions.*

Description

Find snps(user providing) in given regions. This function do not consider strand.

Usage

```
atacSNPAnno(
  atacProc,
  snp.info = NULL,
  region.info = NULL,
  annoOutput = NULL,
  ...
)

## S4 method for signature 'ATACProc'
atacSNPAnno(
  atacProc,
  snp.info = NULL,
  region.info = NULL,
  annoOutput = NULL,
  ...
)

snpanno(snp.info = NULL, region.info = NULL, annoOutput = NULL, ...)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacPeakCalling atacMotifScan. If from atacPeakCalling, the output file would contain the snps in given region. If from atacMotifScan, the output file would contain file path to the output of every motif.
snp.info	Character scalar. Input snp info path. There are two type of input files(you can specify by parameter withend). 1.The first 2 column must be chr, position. e.g. chr13 39776775 rs7993214. Other columns could be other information about snps. 2.The first 3 column must be chr, start, end. e.g. chr13 39776775 39776775 rs7993214. Other columns could be other information about snps. When genome is hg19, using human disease as default.
region.info	Character scalar. Input region info path. The first 3 column must be chr, position, end. The standard BED format is recommended.
annoOutput	Character scalar. Output path.
...	withend Your snp data has only one position column or 2.

Value

An invisible ATACProc-class object scalar.

Author(s)

Wei Zhang

See Also

atacPeakCalling atacMotifScan

Examples

```
library(R.utils)
p1bz <- system.file("extdata", "Example_peak1.bed.bz2", package="esATAC")
peak1_path <- as.vector(bunzip2(filename = p1bz,
destname = file.path(getwd(), "Example_peak1.bed"),
ext="bz2", FUN=bzfile, overwrite=TRUE, remove = FALSE))
snps <- system.file("extdata", "snp_info", package="esATAC")
#snpanno(snp.info = snps, region.info = peak1_path)
```

Rsortbam *Sort bam file and rebuild bai index.*

Description

Sort bamfile and build index.

Usage

```
atacBamSort(atacProc, bamInput = NULL, bamOutput = NULL, ...)

## S4 method for signature 'ATACProc'
atacBamSort(atacProc, bamInput = NULL, bamOutput = NULL, ...)

bamsort(bamInput = NULL, bamOutput = NULL, ...)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacSam2Bam.
bamInput	Character scalar. Input bam file path.
bamOutput	Character scalar. Output bam file path.
...	Additional arguments, currently unused.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Wei Zhang

See Also

atacSam2Bam atacBam2Bed

Examples

```
library(Rsamtools)
ex1_file <- system.file("extdata", "ex1.bam", package="Rsamtools")
bamsort(bamInput = ex1_file)
```

SamToBam *Convert sam format to bam format.*

Description

This function convert a sam file into a bam file.

Usage

```
atacSam2Bam(atacProc, samInput = NULL, bamOutput = NULL, isSort = TRUE, ...)

## S4 method for signature 'ATACProc'
atacSam2Bam(atacProc, samInput = NULL, bamOutput = NULL, isSort = TRUE, ...)

sam2bam(samInput, bamOutput = NULL, isSort = TRUE, ...)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacBowtie2Mapping.
samInput	Character scalar. Sam file input path.
bamOutput	Character scalar. Bam file output path. If ignored, bed file will be put in the same path as the sam file.
isSort	Logical scalar. Sort bam.
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use bamToBed instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Wei Zhang

See Also

atacBowtie2Mapping atacBam2Bed atacBamSort

Examples

```
library(R.utils)
sam_bz <- system.file("extdata", "Example.sam.bz2", package="esATAC")
sam_path <- as.vector(bunzip2(filename = sam_bz,
destname = file.path(getwd(), "Example.sam"),
ext="bz2", FUN=bzfile, remove = FALSE))
sam2bam(samInput = sam_path)
```

SamToBed *Convert SAM file to BED file*

Description

This function is used to convert SAM file to BED file and merge interleave paired end reads, shift reads, filter reads according to chromosome, filter reads according to fragment size, sort, remove duplicates reads before generating BED file.

Usage

```
atacSamToBed(
  atacProc,
  reportOutput = NULL,
  merge = c("auto", "yes", "no"),
  posOffset = +4,
  negOffset = -5,
  chrFilterList = "chrM",
  samInput = NULL,
  bedOutput = NULL,
```

```
  sortBed = TRUE,
  minFragLen = 0,
  maxFragLen = 100,
  saveExtLen = FALSE,
  uniqueBed = TRUE,
  ...
)

## S4 method for signature 'ATACProc'
atacSamToBed(
  atacProc,
  reportOutput = NULL,
  merge = c("auto", "yes", "no"),
  posOffset = +4,
  negOffset = -5,
  chrFilterList = "chrM",
  samInput = NULL,
  bedOutput = NULL,
  sortBed = TRUE,
  minFragLen = 0,
  maxFragLen = 100,
  saveExtLen = FALSE,
  uniqueBed = TRUE,
  ...
)

samToBed(
  samInput,
  reportOutput = NULL,
  merge = c("auto", "yes", "no"),
  posOffset = +4,
  negOffset = -5,
  chrFilterList = "chrM",
  bedOutput = NULL,
  sortBed = TRUE,
  minFragLen = 0,
  maxFragLen = 100,
  saveExtLen = FALSE,
  uniqueBed = TRUE,
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacBowtie2Mapping bowtie2Mapping
reportOutput	Character scalar report file path
merge	Logical scalar Merge paired end reads.

posOffset	Integer scalar The offset that positive strand reads will shift.
negOffset	Integer scalar The offset that negative strand reads will shift.
chrFilterList	Character vector The chromatin(or regex of chromatin) will be discard
samInput	Character scalar. SAM file input path.
bedOutput	Character scalar. Bed file output path.
sortBed	Logical scalar Sort bed file in the order of chromatin, start, end
minFragLen	Integer scalar The minimum fragment size will be retained.
maxFragLen	Integer scalar The maximum fragment size will be retained.
saveExtLen	Logical scaler Save the fragment that are not in the range of minFragLen and maxFragLen
uniqueBed	Logical scalar Remove duplicates reads in bed if TRUE. default: FALSE
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, you can use samToBed instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacBowtie2Mapping bowtie2Mapping atacFragLenDistr atacExtractCutSite atacPeakCalling atacBedUtils atacTSSQC atacBedToBigWig

Examples

```
library(R.utils)
library(magrittr)
td <- tempdir()
setTmpDir(td)

sambzfile <- system.file(package="esATAC", "extdata", "Example.sam.bz2")
samfile <- file.path(td,"Example.sam")
bunzip2(sambzfile,destname=samfile,overwrite=TRUE,remove=FALSE)
samToBed(samInput = samfile)
```

SingleRepReport *Final report for single group of regions*

Description

When user call all steps in the pipeline, the final report can be generated.

Usage

```
atacSingleRepReport(prevStep, htmlOutput = NULL, createHTML = TRUE, ...)

## S4 method for signature 'Step'
atacSingleRepReport(prevStep, htmlOutput = NULL, createHTML = TRUE, ...)
```

Arguments

prevStep	Step-class object scalar. Any steps object in this package is acceptable when the pipeline is ready.
htmlOutput	Character scalar. HTML report file directory Default: NULL ("Report.html")
createHTML	Logical scalar. If create HTML file. Default: TRUE. This parameter needs to be set FALSE when pandoc or other dependence softwares are not available for rmarkdown package.
...	Additional arguments, currently unused.

Details

The report is HTML format. All link in HTML file is the relative directory in report step folder and other step folder If user want to move HTML file and keep all link access available, they should move the whole pipeline folder at the same time.

Value

An invisible ATACProc-class object (Step-class based) scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacPipe

TSSQC *Quality control for transcription start site(TSS) reads enrichment*

Description

These functions are used to generate the reads coverage plot around TSS.

Usage

```
atacTSSQC(
  atacProc,
  txdbKnownGene = NULL,
  bsgenome = NULL,
  reportPrefix = NULL,
  bedInput = NULL,
  fragLenRange = c(0, 2000),
  tssUpdownstream = 1000,
  newStepType = "TSSQC",
  ...
)

## S4 method for signature 'ATACProc'
atacTSSQC(
  atacProc,
  txdbKnownGene = NULL,
  bsgenome = NULL,
  reportPrefix = NULL,
  bedInput = NULL,
  fragLenRange = c(0, 2000),
  tssUpdownstream = 1000,
  newStepType = "TSSQC",
  ...
)

tssQC(
  bedInput,
  txdbKnownGene = NULL,
  bsgenome = NULL,
  reportPrefix = NULL,
  fragLenRange = c(0, 2000),
  tssUpdownstream = 1000,
  newStepType = "TSSQC",
  ...
)
```

Arguments

atacProc	ATACProc-class object scalar. It has to be the return value of upstream process: atacSamToBed, atacBedUtils.
txdbKnownGene	TxDb object scalar. TxDb object for specific species.
bsgenome	BSGenome object scalar. BSGenome object for specific species.
reportPrefix	Character scalar. The prefix of report files path.
bedInput	Character scalar. BED file input path.
fragLenRange	Interger vector of 2 element. The fragment length ranges.
tssUpdownstream	Interger scalar. The upstream and downstrem from TSS locations.
newStepType	Character scalar. New class name
...	Additional arguments, currently unused.

Details

The parameter related to input and output file path will be automatically obtained from ATACProc-class object(atacProc) or generated based on known parameters if their values are default(e.g. NULL). Otherwise, the generated values will be overwrited. If you want to use this function independently, atacProc should be set NULL or you can use tssQC instead.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacSamToBed samToBed atacBedUtils bedUtils

Examples

```
library(R.utils)
td <- tempdir()
setTmpDir(td)

bedbzfile <- system.file(package="esATAC", "extdata", "chr20.50000.bed.bz2")
bedfile <- file.path(td,"chr20.50000.bed")
bunzip2(bedbzfile,destname=bedfile,overwrite=TRUE,remove=FALSE)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(BSgenome.Hsapiens.UCSC.hg19)
tssQC(bedfile,TxDb.Hsapiens.UCSC.hg19.knownGene,BSgenome.Hsapiens.UCSC.hg19,fragLenRange=c(180,247))

dir(td)
```

UnzipAndMerge *Unzip and merge fastq files*

Description

Unzip and merge fastq files that are in format of bzip, gzip or fastq

Usage

```
atacUnzipAndMerge(
  fastqInput1,
  fastqInput2 = NULL,
  fastqOutput1 = NULL,
  fastqOutput2 = NULL,
  interleave = FALSE,
  ...
)

unzipAndMerge(
  fastqInput1,
  fastqInput2 = NULL,
  fastqOutput1 = NULL,
  fastqOutput2 = NULL,
  interleave = FALSE,
  ...
)
```

Arguments

fastqInput1	Character vector. For single-end sequencing, it contains sequence file paths. For paired-end sequencing, it can be file paths with #1 mates paired with file paths in fastqInput2 And it can also be interleaved file paths when argument interleaved=TRUE
fastqInput2	Character vector. It contains file paths with #2 mates paired with file paths in fastqInput1 For single-end sequencing files and interleaved paired-end sequencing files(argument interleaved=TRUE), it must be NULL.
fastqOutput1	Character. The trimmed mate1 reads output file path for fastqInput2.
fastqOutput2	Character. The trimmed mate2 reads output file path for fastqInput2.
interleave	Logical. Set TRUE when files are interleaved paired-end sequencing data.
...	Additional arguments, currently unused.

Value

An invisible ATACProc-class object scalar for downstream analysis.

Author(s)

Zheng Wei

See Also

atacRenamer atacQCReport

Examples

```
ignoreCheck() # warnning: run this for fast test only

td<-tempdir()
setTmpDir(td)

# Identify adapters
prefix<-system.file(package="esATAC", "extdata", "uzmg")
(reads_1 <-file.path(prefix,"m1",dir(file.path(prefix,"m1"))))
(reads_2 <-file.path(prefix,"m2",dir(file.path(prefix,"m2"))))

reads_merged_1 <- file.path(td,"reads_1.fq")
reads_merged_2 <- file.path(td,"reads_2.fq")
atacproc <- atacUnzipAndMerge(fastqInput1 = reads_1,fastqInput2 = reads_2)
dir(td)
```

Index

annotatePeak, *54*
atacBam2Bed, *20, 21*, *61, 62*
atacBam2Bed *(*BamToBed*)*, 15
atacBam2Bed,ATACProc-method *(*BamToBed*)*, 15
atacBamSort, *16, 17*, *62*
atacBamSort *(*Rsortbam*)*, 60
atacBamSort,ATACProc-method *(*Rsortbam*)*, 60
atacBedToBigWig, *21*, *64*
atacBedToBigWig *(*BedToBigWig*)*, 17
atacBedToBigWig,ATACProc-method *(*BedToBigWig*)*, 17
atacBedUtils, *18*, *32–34*, *38–43*, *64*, *67*
atacBedUtils *(*BedUtils*)*, 19
atacBedUtils,ATACProc-method *(*BedUtils*)*, 19
atacBowtie2Mapping, *5*, *10*, *31*, *36*, *45*, *61–64*
atacBowtie2Mapping *(*Bowtie2Mapping*)*, 22
atacBowtie2Mapping,ATACProc-method *(*Bowtie2Mapping*)*, 22
atacCutSiteCount, *27, 28*, *52*
atacCutSiteCount *(*CutSiteCountR*)*, 24
atacCutSiteCount,ATACProc-method *(*CutSiteCountR*)*, 24
atacExtractCutSite, *21*, *25, 26*, *64*
atacExtractCutSite *(*CutSitePre*)*, 27
atacExtractCutSite,ATACProc-method *(*CutSitePre*)*, 27
atacFindAdapter *(*FindAdapter*)*, 29
atacFindAdapter,ATACProc-method *(*FindAdapter*)*, 29
atacFragLenDistr, *21*, *64*
atacFragLenDistr *(*FragLenDistr*)*, 31
atacFragLenDistr,ATACProc-method *(*FragLenDistr*)*, 31
atacFripQC *(*FRiPQC*)*, 33
atacFripQC,ATACProc-method *(*FRiPQC*)*, 33
atacGOAnalysis, *56*
atacGOAnalysis *(*RGo*)*, 48
atacGOAnalysis,ATACProc-method *(*RGo*)*, 48
atacLibComplexQC, *23*
atacLibComplexQC *(*LibComplexQC*)*, 35
atacLibComplexQC,ATACProc-method *(*LibComplexQC*)*, 35
atacMotifScan, *5*, *10*, *25, 26*, *59, 60*
atacMotifScan *(*RMotifScan*)*, 50
atacMotifScan,ATACProc-method *(*RMotifScan*)*, 50
atacMotifScanPair *(*RMotifScanPair*)*, 52
atacMotifScanPair,ATACProc-method *(*RMotifScanPair*)*, 52
atacPeakAnno, *49, 50*
atacPeakAnno *(*RPeakAnno*)*, 54
atacPeakAnno,ATACProc-method *(*RPeakAnno*)*, 54
atacPeakCalling, *5*, *10*, *21*, *33*, *51, 52*, *56*, *58–60*, *64*
atacPeakCalling *(*PeakCallingFseq*)*, 37
atacPeakCalling,ATACProc-method *(*PeakCallingFseq*)*, 37
atacPeakCallingMACS2 *(*PeakCallingMACS2*)*, 39
atacPeakCallingMACS2,ATACProc-method *(*PeakCallingMACS2*)*, 39
atacPeakComp, *53, 54*
atacPeakComp *(*RPeakComp*)*, 57
atacPeakComp,ATACProc-method *(*RPeakComp*)*, 57
atacPeakQC *(*PeakQC*)*, 41
atacPeakQC,ATACProc-method *(*PeakQC*)*, 41
atacPipe, *8*, *14*, *65*
atacPipe *(*esATAC-package*)*, 3
atacPipe2, *3*, *5*, 6, *10*
ATACProc-class, 9
atacQCReport, *47*, *69*
atacQCReport *(*FastQC*)*, 28
atacQCReport,ATACProc-method *(*FastQC*)*,

28
atacRemoveAdapter, *5*, *10*, *23*, *47*
atacRemoveAdapter *(*RemoveAdapter*)*, 43
atacRemoveAdapter,ATACProc-method *(*RemoveAdapter*)*, 43
atacRenamer, *5*, *10*, *28–31*, *44, 45*, *69*
atacRenamer *(*Renamer*)*, 46
atacRenamer,ATACProc-method *(*Renamer*)*, 46
atacRepsPipe, *3*, *5*, 9
atacRepsPipe2, *3*, *5*, 11
atacSam2Bam, *16, 17*, *23*, *60, 61*
atacSam2Bam *(*SamToBam*)*, 61
atacSam2Bam,ATACProc-method *(*SamToBam*)*, 61
atacSamToBed, *18*, *20, 21*, *23*, *27*, *32–34*, *38–43*, *67*
atacSamToBed *(*SamToBed*)*, 62
atacSamToBed,ATACProc-method *(*SamToBed*)*, 62
atacSingleRepReport *(*SingleRepReport*)*, 65
atacSingleRepReport,Step-method *(*SingleRepReport*)*, 65
atacSNPAnno *(*RSNPs*)*, 59
atacSNPAnno,ATACProc-method *(*RSNPs*)*, 59
atacTSSQC, *21*, *64*
atacTSSQC *(*TSSQC*)*, 66
atacTSSQC,ATACProc-method *(*TSSQC*)*, 66
atacUnzipAndMerge, *28–31*, *44, 45*, *47*
atacUnzipAndMerge *(*UnzipAndMerge*)*, 68

bam2bed, *20, 21*
bam2bed *(*BamToBed*)*, 15
bamsort *(*Rsortbam*)*, 60
BamToBed, 15
BedToBigWig, 17
bedToBigWig *(*BedToBigWig*)*, 17
BedUtils, 19
bedUtils, *18*, *32, 33*, *39*, *41*, *67*
bedUtils *(*BedUtils*)*, 19
bowtie2, *23*
bowtie2_build, *23*
bowtie2_usage, *23*
Bowtie2Mapping, 22
bowtie2Mapping, *36*, *63, 64*
bowtie2Mapping *(*Bowtie2Mapping*)*, 22

cutsitecount *(*CutSiteCountR*)*, 24
CutSiteCountR, 24
CutSitePre, 27

enrichGO, *48–50*
esATAC-package, 3
extractcutsite *(*CutSitePre*)*, 27

FastQC, 28
FindAdapter, 29
findAdapter *(*FindAdapter*)*, 29
FragLenDistr, 31
fragLenDistr *(*FragLenDistr*)*, 31
FRiPQC, 33
fripQC, *34*
fripQC *(*FRiPQC*)*, 33

getMotifInfo, 34
getRefRc, *51*, *53*
goanalysis *(*RGo*)*, 48
GRanges, *51*, *56*, *58*

LibComplexQC, 35
libComplexQC, *36*
libComplexQC *(*LibComplexQC*)*, 35

motifscan *(*RMotifScan*)*, 50
motifscanpair *(*RMotifScanPair*)*, 52

peakanno *(*RPeakAnno*)*, 54
peakCalling, *33*
peakCalling *(*PeakCallingFseq*)*, 37
PeakCallingFseq, 37
PeakCallingMACS2, 39
peakCallingMACS2 *(*PeakCallingMACS2*)*, 39
peakcomp *(*RPeakComp*)*, 57
PeakQC, 41
peakQC *(*PeakQC*)*, 41
PFMatrix, *5*, *7*, *10*, *13*, *34, 35*, *51*, *53*
PFMatrixList, *5*, *7*, *10*, *13*, *34, 35*, *51*, *53*
printMap, *5*, *10*
PWMatrix, *5*, *7*, *10*, *13*, *51*, *53*
PWMatrixList, *5*, *7*, *10*, *13*, *51*, *53*

qcreport *(*FastQC*)*, 28

RemoveAdapter, 43
removeAdapter, *23*
removeAdapter *(*RemoveAdapter*)*, 43
Renamer, 46
renamer, *30, 31*, *44, 45*

renamer *(*Renamer*)*, 46
RGo, 48
RMotifScan, 50
RMotifScanPair, 52
RPeakAnno, 54
RPeakComp, 57
RSNPs, 59
Rsortbam, 60

sam2bam *(*SamToBam*)*, 61
SamToBam, 61
SamToBed, 62
samToBed, *18*, *20, 21*, *32, 33*, *39*, *41*, *67*
samToBed *(*SamToBed*)*, 62
SingleRepReport, 65
snpanno *(*RSNPs*)*, 59

testPeakCallingMACS2
*(*PeakCallingMACS2*)*, 39
TSSQC, 66
tssQC *(*TSSQC*)*, 66

UnzipAndMerge, 68
unzipAndMerge, *30, 31*, *44, 45*, *47*
unzipAndMerge *(*UnzipAndMerge*)*, 68