Package 'scruff'

November 13, 2026

Title Single Cell RNA-Seq UMI Filtering Facilitator (scruff)

Version 1.31.0

Date 2024-03-27

Description A pipeline which processes single cell RNA-seq (scRNA-seq) reads from CEL-seq and CEL-seq2 protocols. Demultiplex scRNA-seq FASTQ files, align reads to reference genome using Rsubread, and generate UMI filtered count matrix. Also provide visualizations of read alignments and pre- and post-alignment QC metrics.

Depends R (>= 4.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.3.3

VignetteBuilder knitr

Imports data.table, GenomicAlignments, GenomicFeatures, txdbmaker, GenomicRanges, Rsamtools, ShortRead, parallel, plyr, BiocGenerics, BiocParallel, S4Vectors, AnnotationDbi, Biostrings, methods, ggplot2, ggthemes, scales, GenomeInfoDb, stringdist, ggbio, rtracklayer, SingleCellExperiment, SummarizedExperiment, Rsubread, parallelly, patchwork

Suggests BiocStyle, knitr, rmarkdown, testthat

biocViews Software, Technology, Sequencing, Alignment, RNASeq, SingleCell, WorkflowStep, Preprocessing, QualityControl, Visualization, ImmunoOncology

BugReports https://github.com/campbio/scruff/issues

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

git_branch devel

git_last_commit 3278f3a

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-11-12

Author Zhe Wang [aut, cre],
Junming Hu [aut],
Joshua Campbell [aut]

Maintainer Zhe Wang <zhe@bu.edu>

Contents

alignRsubread 2
bamExample 4
barcodeExample 4
cbtop10000 5
countUMI 5
demultiplex 7
gview 9
qcplots 10
rview 11
sceExample 12
scruff 12
tenxBamqc 16
validCb 17

Index **18**

`alignRsubread` *A wrapper to* `Rsubread` *read alignment function* `align`

Description

This function is **not** available in Windows environment. Align cell specific reads to reference genome and write sequence alignment results to output directory. A wrapper to the `align` function in `Rsubread` package. For details please refer to `Rsubread` manual.

Usage

```
alignRsubread(
  sce,
  index,
  unique = FALSE,
  nBestLocations = 1,
  format = "BAM",
  outDir = "./Alignment",
  cores = max(1, parallelly::availableCores() - 2),
  threads = 1,
  summaryPrefix = "alignment",
  overwrite = FALSE,
  verbose = FALSE,
  logfilePrefix = format(Sys.time(), "%Y%m%d_%H%M%S"),
  ...
)
```

Arguments

`sce`	A `SingleCellExperiment` object of which the `colData` slot contains the **fastq_path** column with paths to input cell-specific FASTQ files.
`index`	Path to the `Rsubread` index of the reference genome. For generation of Rsubread indices, please refer to `buildindex` function in `Rsubread` package.

unique	Argument passed to `align` function in `Rsubread` package. Boolean indicating if only uniquely mapped reads should be reported. A uniquely mapped read has one single mapping location that has less mis-matched bases than any other candidate locations. If set to **FALSE**, multi-mapping reads will be reported in addition to uniquely mapped reads. Number of alignments reported for each multi-mapping read is determined by the nBestLocations parameter. Default is **FALSE**.
nBestLocations	Argument passed to `align` function in `Rsubread` package. Numeric value specifying the maximal number of equally-best mapping locations that will be reported for a multi-mapping read. 1 by default. The allowed value is between 1 to 16 (inclusive). In the mapping output, "NH" tag is used to indicate how many alignments are reported for the read and "HI" tag is used for numbering the alignments reported for the same read. This argument is only applicable when unique option is set to **FALSE**. `Scruff` package does not support counting alignment files with `nBestLocations > 1`.
format	File format of sequence alignment results. **"BAM"** or **"SAM"**. Default is **"BAM"**.
outDir	Output directory for alignment results. Sequence alignment files will be stored in folders in this directory, respectively. **Make sure the folder is empty.** Default is `"./Alignment"`.
cores	Number of cores used for parallelization. Default is `max(1, parallelly::availableCores() - 2)`, i.e. the number of available cores minus 2.
threads	**Do not change**. Number of threads/CPUs used for mapping for each core. Refer to `align` function in `Rsubread` for details. Default is **1**. It should not be changed in most cases.
summaryPrefix	Prefix for alignment summary filename. Default is `"alignment"`.
overwrite	Boolean indicating whether to overwrite the output directory. Default is **FALSE**.
verbose	Boolean indicating whether to print log messages. Useful for debugging. Default to **FALSE**.
logfilePrefix	Prefix for log file. Default is current date and time in the format of `format(Sys.time(), "%Y%m%d_%H%M%S")`.
...	Additional arguments passed to the `align` function in `Rsubread` package.

Value

A **SingleCellExperiment** object containing the alignment summary information in the `colData` slot. The `alignment_path` column of the annotation table contains the paths to output alignment files.

Examples

```
# The SingleCellExperiment object returned by demultiplex function is
# required for running alignRsubread function

## Not run:
data(barcodeExample, package = "scruff")
fastqs <- list.files(system.file("extdata", package = "scruff"),
    pattern = "\\.fastq\\.gz", full.names = TRUE)

de <- demultiplex(
    project = "example",
```

```
    experiment = c("1h1"),
    lane = c("L001"),
    read1Path = c(fastqs[1]),
    read2Path = c(fastqs[2]),
    barcodeExample,
    bcStart = 1,
    bcStop = 8,
    umiStart = 9,
    umiStop = 12,
    keep = 75,
    overwrite = TRUE)

# Alignment
library(Rsubread)
# Create index files for GRCm38_MT.
fasta <- system.file("extdata", "GRCm38_MT.fa", package = "scruff")
# Specify the basename for Rsubread index
indexBase <- "GRCm38_MT"
buildindex(basename = indexBase, reference = fasta, indexSplit = FALSE)

al <- alignRsubread(de, indexBase, overwrite = TRUE)

## End(Not run)
```

`bamExample` *Example GAlignments Object*

Description

An example GAlignments object containing read alignment information for cell "vandenBrink_b1_cell_0095" of example FASTQ files. Used as an example for `rview` function.

Usage

```
bamExample
```

Format

A `GAlignments` object.

`barcodeExample` *A vector of example cell barcodes.*

Description

A vector containing 48 predefined cell barcodes which will be used for demultiplexing the example FASTQ files. Only the cell barcodes from 49 to 96 of the original 96 barcodes are used here to reduce the time to run example codes and compile the vignette.

Usage

```
barcodeExample
```

Format

A vector of cell barcode sequences. Cell barcodes for this study (van den Brink, et al.) are of length 8.

`cbtop10000` *Top 10,000 rows for v1, v2, and v3 cell barcode whitelist files*

Description

The first 10,000 cell barcodes in v1 (737K-april-2014_rc.txt), v2 (737K-august-2016.txt), and v3 (3M-february-2018.txt) cell barcode whitelist files. This object is used for testing the validity of input assay chemistry `validCb` for tenxBamqc function. The cell barcodes for the first 10,000 alignments in the input BAM file will be mapped to each chemistry's whitelist to determine the assay chemistry of the BAM file.

Usage

```
cbtop10000
```

Format

A `data.table` object.

`countUMI` *Count the number of UMIs for each gene and output count matrix*

Description

Count unique *UMI:gene* pairs for single cell RNA-sequencing alignment files. Write resulting count matrix to output directory. Columns are samples (cells) and rows are gene IDs. The input sequence alignment files must be generated using FASTQ files generated by the `demultiplex` function in scruff package. Return a SingleCellExperiment object containing the count matrix, cell and gene annotations, and all QC metrics.

Usage

```
countUMI(
  sce,
  reference,
  umiEdit = 0,
  format = "BAM",
  outDir = "./Count",
  cellPerWell = 1,
  cores = max(1, parallelly::availableCores() - 2),
  outputPrefix = "countUMI",
  verbose = FALSE,
  logfilePrefix = format(Sys.time(), "%Y%m%d_%H%M%S")
)
```

Arguments

sce	A SingleCellExperiment object of which the colData slot contains the **alignment_path** column with paths to input cell-specific sequence alignment files (BAM or SAM format).
reference	Path to the reference GTF file. The TxDb object of the GTF file will be generated and saved in the current working directory with ".sqlite" suffix.
umiEdit	Maximally allowed Hamming distance for UMI correction. For read alignments in each gene, by comparing to a more abundant UMI with more reads, UMIs having fewer reads and with mismatches equal or fewer than umiEdit will be assigned a corrected UMI (the UMI with more reads). Default is 0, meaning no UMI correction is performed. Doing UMI correction will decrease the number of transcripts per gene.
format	Format of input sequence alignment files. **"BAM"** or **"SAM"**. Default is **"BAM"**.
outDir	Output directory for UMI counting results. UMI corrected count matrix will be stored in this directory. Default is "./Count".
cellPerWell	Number of cells per well. Can be an integer (e.g. 1) indicating the number of cells in each well or an vector with length equal to the total number of cells in the input alignment files specifying the number of cells in each file. Default is 1.
cores	Number of cores used for parallelization. Default is max(1, parallelly::availableCores() - 2), i.e. the number of available cores minus 2.
outputPrefix	Prefix for expression table filename. Default is "countUMI".
verbose	Print log messages. Useful for debugging. Default to **FALSE**.
logfilePrefix	Prefix for log file. Default is current date and time in the format of format(Sys.time(), "%Y%m%d_%H%M%S").

Value

A **SingleCellExperiment** object.

Examples

```
## Not run:
data(barcodeExample, package = "scruff")
# The SingleCellExperiment object returned by alignRsubread function and the
# alignment BAM files are required for running countUMI function
# First demultiplex example FASTQ files
fastqs <- list.files(system.file("extdata", package = "scruff"),
    pattern = "\\.fastq\\.gz", full.names = TRUE)

de <- demultiplex(
    project = "example",
    experiment = c("1h1"),
    lane = c("L001"),
    read1Path = c(fastqs[1]),
    read2Path = c(fastqs[2]),
    barcodeExample,
    bcStart = 1,
    bcStop = 8,
    umiStart = 9,
```

```
    umiStop = 12,
    keep = 75,
    overwrite = TRUE)

# Alignment
library(Rsubread)
# Create index files for GRCm38_MT.
fasta <- system.file("extdata", "GRCm38_MT.fa", package = "scruff")
# Specify the basename for Rsubread index
indexBase <- "GRCm38_MT"
buildindex(basename = indexBase, reference = fasta, indexSplit = FALSE)

al <- alignRsubread(de, indexBase, overwrite = TRUE)

# Counting
gtf <- system.file("extdata", "GRCm38_MT.gtf", package = "scruff")
sce = countUMI(al, gtf, cellPerWell=c(rep(1, 46), 0, 0))

## End(Not run)

# or use the built-in SingleCellExperiment object generated using
# example dataset (see ?sceExample)
data(sceExample, package = "scruff")
```

`demultiplex` *Demultiplex cell barcodes and assign cell specific reads*

Description

Demultiplex fastq files and write cell specific reads in compressed fastq format to output directory

Usage

```
demultiplex(
  project = paste0("project_", Sys.Date()),
  experiment,
  lane,
  read1Path,
  read2Path,
  bc,
  bcStart,
  bcStop,
  bcEdit = 0,
  umiStart,
  umiStop,
  keep,
  minQual = 10,
  yieldReads = 1e+06,
  outDir = "./Demultiplex",
  summaryPrefix = "demultiplex",
  overwrite = FALSE,
  cores = max(1, parallelly::availableCores() - 2),
  verbose = FALSE,
```

```
  logfilePrefix = format(Sys.time(), "%Y%m%d_%H%M%S")
)
```

Arguments

project	The project name. Default is paste0("project_", Sys.Date()).
experiment	A character vector of experiment names. Represents the group label for each FASTQ file, e.g. "patient1, patient2, ...". The number of cells in a experiment equals the length of cell barcodes bc. The length of experiment equals the number of FASTQ files to be processed.
lane	A character or character vector of flow cell lane numbers. FASTQ files from lanes having the same experiment will be concatenated. If FASTQ files from multiple lanes are already concatenated, any placeholder would be sufficient, e.g. "L001".
read1Path	A character vector of file paths to the read 1 FASTQ files. These are the read files containing UMI and cell barcode sequences.
read2Path	A character vector of file paths to the read 2 FASTQ files. These read files contain genomic transcript sequences.
bc	A character vector of pre-determined cell barcodes. For example, see ?barcodeExample.
bcStart	Integer or vector of integers containing the cell barcode start positions (inclusive, one-based numbering).
bcStop	Integer or vector of integers containing the cell barcode stop positions (inclusive, one-based numbering).
bcEdit	Maximally allowed Hamming distance for barcode correction. Barcodes with mismatches equal or fewer than this will be assigned a corrected barcode if the inferred barcode matches uniquely in the provided predetermined barcode list. Default is 0, meaning no cell barcode correction is performed.
umiStart	Integer or vector of integers containing the start positions (inclusive, one-based numbering) of UMI sequences.
umiStop	Integer or vector of integers containing the stop positions (inclusive, one-based numbering) of UMI sequences.
keep	Read trimming. Read length or number of nucleotides to keep for read 2 (the read that contains transcript sequence information). Longer reads will be clipped at 3' end. Shorter reads will not be affected.
minQual	Minimally acceptable Phred quality score for barcode and UMI sequences. Phread quality scores are calculated for each nucleotide in the sequence. Sequences with at least one nucleotide with score lower than this will be filtered out. Default is **10**.
yieldReads	The number of reads to yield when drawing successive subsets from a fastq file, providing the number of successive records to be returned on each yield. This parameter is passed to the n argument of the FastqStreamer function in *ShortRead* package. Default is **1e06**.
outDir	Output folder path for demultiplex results. Demultiplexed cell specifc FASTQ files will be stored in folders in this path, respectively. **Make sure the folder is empty.** Default is "./Demultiplex".
summaryPrefix	Prefix for demultiplex summary filename. Default is "demultiplex".
overwrite	Boolean indicating whether to overwrite the output directory. Default is **FALSE**.

cores	Number of cores used for parallelization. Default is `max(1, parallelly::availableCores() - 2)`, i.e. the number of available cores minus 2.
verbose	Poolean indicating whether to print log messages. Useful for debugging. Default to **FALSE**.
logfilePrefix	Prefix for log file. Default is current date and time in the format of `format(Sys.time(), "%Y%m%d_%H%M%S")`.

Value

A SingleCellExperiment object containing the demultiplex summary information in the `colData` slot.

Examples

```
# Demultiplex example FASTQ files
data(barcodeExample, package = "scruff")
fastqs <- list.files(system.file("extdata", package = "scruff"),
    pattern = "\\.fastq\\.gz", full.names = TRUE)

de <- demultiplex(
    project = "example",
    experiment = c("1h1"),
    lane = c("L001"),
    read1Path = c(fastqs[1]),
    read2Path = c(fastqs[2]),
    barcodeExample,
    bcStart = 1,
    bcStop = 8,
    umiStart = 9,
    umiStop = 12,
    keep = 75,
    overwrite = TRUE)
```

---

gview *Visualize gene isoforms*

Description

Visualize reference genome. Rectangles represent exons. Arrow represents orientation of transcripts.

Usage

```
gview(
  gtfFile,
  chr = 1,
  start = 1,
  end = NULL,
  rect_width = 0.3,
  line_width = 0.5,
  arrow_segments = 10,
  arrow_width = 30,
```

```
  arrow_length = 0.08,
  arrow_type = "open",
  text_size = 4
)
```

Arguments

gtfFile	A genome annotation file in GTF format.
chr	Chromosome name. Integer or "X", "Y", "MT".
start	Genomic coordinate of the start position.
end	Genomic coordinate of the end position. If NULL, then the maximum length of the chromosome will be used. Default NULL.
rect_width	Exon widths. Default 0.3.
line_width	Line weight. Default 0.5.
arrow_segments	The number of segments lines be divided to. The greater the number, more arrows there are. Default 10.
arrow_width	The angle of the arrow head in degrees (smaller numbers produce narrower, pointier arrows). Essentially describes the width of the arrow head. Passed to the angle parameter of arrow function. Default 30.
arrow_length	The length of the arrow head. Passed to the length argument of arrow function. Default 0.08.
arrow_type	One of "open" or "closed" indicating whether the arrow head should be a closed triangle. Passed to the type argument of arrow function. Default "open".
text_size	Size of text. Passed to the size argument of the geom_text function. Default 4.

Value

A ggplot object of genomic view

Examples

```
gtf <- system.file("extdata", "GRCm38_MT.gtf", package = "scruff")
g <- gview(gtf, chr = "MT")
g
```

---

qcplots *Visualize data quality*

Description

Visualize data quality from the colData of the SingleCellExperiment object and return a list of figures in arrangelist object.

Usage

```
qcplots(sce)
```

Arguments

sce	An SingleCellExperiment object returned from scruff, countUMI, or tenxBamqc function.

Value

A list of grobs objects ready for plotting

Examples

```
data(sceExample, package = "scruff")
qcplots(sceExample)
```

---

rview *Visualize aligned reads*

Description

Visualize read alignments for UMI tagged single cell RNA-sequencing data. Read names must contain UMI sequences at the end delimited by "**:**". Arrow represents orientation of alignment. Reads are colored by their UMI and sorted by their start positions and UMI.

Usage

```
rview(
  bamGA,
  chr = "1",
  start = 1,
  end = max(BiocGenerics::end(bamGA)),
  legend = FALSE
)
```

Arguments

bamGA	A GenomicAlignment object
chr	Chromosome. Integer or "X", "Y", "MT".
start	Genomic coordinate of the start position.
end	Genomic coordinate of the end position.
legend	Show legend. Default is FALSE.

Value

A ggplot object of aligned reads

Examples

```
data(bamExample, package = "scruff")
g <- rview(bamExample, chr = "MT", legend = TRUE)
g
```

sceExample *Example SingleCellExperiment Object*

Description

An example SingleCellExperiment object containing count matrix, cell and gene annotations, and all QC metrics for mouse mitochonrial genes generated from example FASTQ reads.

Usage

```
sceExample
```

Format

A `SingleCellExperiment` object.

scruff *Run scruff pipeline*

Description

Run the `scruff` pipeline. This function performs all `demultiplex`, `alignRsubread`, and `countUMI` functions. Write demultiplex statistics, alignment statistics, and UMI filtered count matrix in output directories. Return a SingleCellExperiment object containing the count matrix, cell and gene annotations, and all QC metrics.

Usage

```
scruff(
  project = paste0("project_", Sys.Date()),
  experiment,
  lane,
  read1Path,
  read2Path,
  bc,
  index,
  reference,
  bcStart,
  bcStop,
  bcEdit = 0,
  umiStart,
  umiStop,
  umiEdit = 0,
  keep,
  cellPerWell = 1,
  unique = FALSE,
  nBestLocations = 1,
  minQual = 10,
  yieldReads = 1e+06,
```

```
  alignmentFileFormat = "BAM",
  demultiplexOutDir = "./Demultiplex",
  alignmentOutDir = "./Alignment",
  countUmiOutDir = "./Count",
  demultiplexSummaryPrefix = "demultiplex",
  alignmentSummaryPrefix = "alignment",
  countPrefix = "countUMI",
  logfilePrefix = format(Sys.time(), "%Y%m%d_%H%M%S"),
  overwrite = FALSE,
  verbose = FALSE,
  cores = max(1, parallelly::availableCores() - 2),
  threads = 1,
  ...
)
```

Arguments

`project`	The project name. Default is `paste0("project_", Sys.Date())`.
`experiment`	A character vector of experiment names. Represents the group label for each FASTQ file, e.g. "patient1, patient2, ...". The number of cells in a experiment equals the length of cell barcodes `bc`. The length of `experiment` equals the number of FASTQ files to be processed.
`lane`	A character or character vector of flow cell lane numbers. If FASTQ files from multiple lanes are concatenated, any placeholder would be sufficient, e.g. "L001".
`read1Path`	A character vector of file paths to the read1 FASTQ files. These are the read files with UMI and cell barcode information.
`read2Path`	A character vector of file paths to the read2 FASTQ files. These read files contain genomic sequences.
`bc`	A vector of pre-determined cell barcodes. For example, see `?barcodeExample`.
`index`	Path to the `Rsubread` index of the reference genome. For generation of Rsubread indices, please refer to `buildindex` function in `Rsubread` package.
`reference`	Path to the reference GTF file. The TxDb object of the GTF file will be generated and saved in the current working directory with ".sqlite" suffix.
`bcStart`	Integer or vector of integers containing the cell barcode start positions (inclusive, one-based numbering).
`bcStop`	Integer or vector of integers containing the cell barcode stop positions (inclusive, one-based numbering).
`bcEdit`	Maximally allowed Hamming distance for barcode correction. Barcodes with mismatches equal or fewer than this will be assigned a corrected barcode if the inferred barcode matches uniquely in the provided predetermined barcode list. Default is 0, meaning no cell barcode correction is performed.
`umiStart`	Integer or vector of integers containing the start positions (inclusive, one-based numbering) of UMI sequences.
`umiStop`	Integer or vector of integers containing the stop positions (inclusive, one-based numbering) of UMI sequences.
`umiEdit`	Maximally allowed Hamming distance for UMI correction. For read alignments in each gene, by comparing to a more abundant UMI with more reads, UMIs having fewer reads and with mismatches equal or fewer than `umiEdit` will be

assigned a corrected UMI (the UMI with more reads). Default is 0, meaning no UMI correction is performed. Doing UMI correction will decrease the number of transcripts per gene.

`keep` Read trimming. Read length or number of nucleotides to keep for read 2 (the read that contains transcript sequence information). Longer reads will be clipped at 3' end. Shorter reads will not be affected. This number should be determined based on the sequencing kit that was used in library preparation step.

`cellPerWell` Number of cells per well. Can be an integer (e.g. 1) indicating the number of cells in each well or an vector with length equal to the total number of cells in the input alignment files specifying the number of cells in each file. Default is 1.

`unique` Argument passed to `align` function in `Rsubread` package. Boolean indicating if only uniquely mapped reads should be reported. A uniquely mapped read has one single mapping location that has less mis-matched bases than any other candidate locations. If set to **FALSE**, multi-mapping reads will be reported in addition to uniquely mapped reads. Number of alignments reported for each multi-mapping read is determined by the nBestLocations parameter. Default is **FALSE**.

`nBestLocations` Argument passed to `align` function in `Rsubread` package. Numeric value specifying the maximal number of equally-best mapping locations that will be reported for a multi-mapping read. 1 by default. The allowed value is between 1 to 16 (inclusive). In the mapping output, "NH" tag is used to indicate how many alignments are reported for the read and "HI" tag is used for numbering the alignments reported for the same read. This argument is only applicable when unique option is set to **FALSE**.

`minQual` Minimally acceptable Phred quality score for cell barcode and UMI sequences. Phread quality scores are calculated for each nucleotide in these tags. Tags with at least one nucleotide with score lower than this will be filtered out. Default is **10**.

`yieldReads` The number of reads to yield when drawing successive subsets from a fastq file, providing the number of successive records to be returned on each yield. This parameter is passed to the `n` argument of the `FastqStreamer` function in *ShortRead* package. Default is **1e06**.

`alignmentFileFormat`
File format of sequence alignment results. **"BAM"** or **"SAM"**. Default is **"BAM"**.

`demultiplexOutDir`
Output folder path for demultiplex results. Demultiplexed cell specifc FASTQ files will be stored in folders in this path, respectively. **Make sure the folder is empty.** Default is `"./Demultiplex"`.

`alignmentOutDir`
Output directory for alignment results. Sequence alignment maps will be stored in folders in this directory, respectively. **Make sure the folder is empty.** Default is `"./Alignment"`.

`countUmiOutDir` Output directory for UMI counting results. UMI filtered count matrix will be stored in this directory. Default is `"./Count"`.

`demultiplexSummaryPrefix`
Prefix for demultiplex summary filename. Default is `"demultiplex"`.

`alignmentSummaryPrefix`
Prefix for alignment summary filename. Default is `"alignment"`.

countPrefix	Prefix for UMI filtered count matrix filename. Default is "countUMI".
logfilePrefix	Prefix for log file. Default is current date and time in the format of format(Sys.time(), "%Y%m%d_%H%M%S").
overwrite	Boolean indicating whether to overwrite the output directory. Default is **FALSE**.
verbose	Boolean indicating whether to print log messages. Useful for debugging. Default to **FALSE**.
cores	Number of cores to use for parallelization. Default is max(1, parallelly::availableCores() - 2), i.e. the number of available cores minus 2.
threads	**Do not change**. Number of threads/CPUs used for mapping for each core. Refer to align function in Rsubread for details. Default is **1**. It should not be changed in most cases.
...	Additional arguments passed to the align function in Rsubread package.

Value

A SingleCellExperiment object.

Examples

```
## Not run:
# prepare required files

data(barcodeExample, package = "scruff")
fastqs <- list.files(system.file("extdata", package = "scruff"),
    pattern = "\\.fastq\\.gz", full.names = TRUE)
fasta <- system.file("extdata", "GRCm38_MT.fa", package = "scruff")
gtf <- system.file("extdata", "GRCm38_MT.gtf", package = "scruff")

library(Rsubread)
# Specify the basename for Rsubread index
indexBase <- "GRCm38_MT"
# Create index files for GRCm38_MT.
buildindex(basename = indexBase, reference = fasta, indexSplit = FALSE)

# run scruff pipeline
sce <- scruff(project = "example",
    experiment = c("1h1"),
    lane = c("L001"),
    read1Path = c(fastqs[1]),
    read2Path = c(fastqs[2]),
    bc = barcodeExample,
    index = indexBase,
    reference = gtf,
    bcStart = 1,
    bcStop = 8,
    umiStart = 9,
    umiStop = 12,
    keep = 75,
    cellPerWell = c(rep(1, 46), 0, 0),
    overwrite = TRUE,
    verbose = TRUE)

## End(Not run)
```

```
# or use the built-in SingleCellExperiment object generated using
# example dataset (see ?sceExample)
data(sceExample, package = "scruff")
```

`tenxBamqc` *Generate and output 10X read alignment data quality metrics*

Description

Read BAM file generated by Cell Ranger pipeline and output QC metrics including number of aligned reads and reads aligned to an gene.

Usage

```
tenxBamqc(
  bam,
  experiment,
  filter,
  validCb = NA,
  tags = c("NH", "GX", "CB", "MM"),
  yieldSize = 1e+06,
  outDir = "./",
  cores = max(1, parallelly::availableCores() - 2)
)
```

Arguments

`bam`	Paths to input BAM files generated by Cell Ranger pipeline. These files are usually named *"possorted_genome_bam.bam"* in the *"outs"* folder of the top-level project output folders, respectively.
`experiment`	A character vector of experiment names. Represents the group label for each BAM file, e.g. "patient1, patient2, ...". The length of `experiment` equals the number of BAM files to be processed.
`filter`	Paths to the filtered barcode files. Should be in the same length and order of the input BAM files. These files are named *"barcodes.tsv"* located at *outs/filtered_gene_bc_matrices/<ref*
`validCb`	Path to the cell barcode whitelist file. By default uses the file "737K-august-2016.txt" which is compatible with the v2 chemistry protocol. The file can be inspected by calling `data(validCb, package = "scruff")`. If the library is generated using the v1 chemistry protocol, the path to the v1 barcode whitelist file ("737K-april-2014_rc.txt") needs to be provided. For library generated by v3 chemistry protocol, path to "3M-february-2018.txt" is needed.
`tags`	BAM tags used for collecting QC metrics. Contains non-standard tags locally-defined by Cell Ranger pipeline. Should not be changed in most cases.
`yieldSize`	The number of records (alignments) to yield when drawing successive subsets from a BAM file, providing the number of successive records to be returned on each yield. This parameter is passed to the `yieldSize` argument of the `BamFile` function in `Rsamtools` package. Default is **1e06**.
`outDir`	Output directory. The location to write resulting QC table.
`cores`	Number of cores used for parallelization. Default is `max(1, parallelly::availableCores() - 2)`, i.e. the number of available cores minus 2.

Value

A SingleCellExperiment object. The `colData` contains the number of aligned reads (reads_mapped_to_genome) and reads aligned to genes (reads_mapped_to_genes).

Examples

```
# first 5000 records in the bam file downloaded from here:
# http://sra-download.ncbi.nlm.nih.gov/srapub_files/
# SRR5167880_E18_20160930_Neurons_Sample_01.bam
# see details here:
# https://trace.ncbi.nlm.nih.gov/Traces/sra/?study=SRP096558
# and here:
# https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE93421
bamfile10x <- system.file("extdata",
    "SRR5167880_E18_20160930_Neurons_Sample_01_5000.bam",
    package = "scruff")

# library(TENxBrainData)
# library(data.table)
# tenx <- TENxBrainData()
# # get filtered barcodes for sample 01
# filteredBcIndex <- tstrsplit(colData(tenx)[, "Barcode"], "-")[[2]] == 1
# filteredBc <- colData(tenx)[filteredBcIndex, ]["Barcode"]]

filteredBc <- system.file("extdata",
    "SRR5167880_E18_20160930_Neurons_Sample_01_filtered_barcode.tsv",
    package = "scruff")
# QC results are saved to current working directory
sce <- tenxBamqc(bam = bamfile10x,
    experiment = "Neurons_Sample_01",
    filter = filteredBc)
sce
```

`validCb` *Cell barcode whitelist (737K-august-2016.txt)*

Description

A barcode whitelist is the list of all known barcode sequences that have been included in the assay kit and are available during library preparation. There are roughly 737,000 cell barcodes in the whitelist (737K-august-2016.txt) for Cell Ranger's Single Cell 3' and V(D)J applications.

Usage

```
validCb
```

Format

A `data.table` object.

Index

∗ **datasets**
bamExample, 4
barcodeExample, 4
cbtop10000, 5
sceExample, 12
validCb, 17

alignRsubread, 2

bamExample, 4
barcodeExample, 4

cbtop10000, 5
countUMI, 5, *11*

demultiplex, 7

gview, 9

qcplots, 10

rview, 11

sceExample, 12
scruff, *11*, 12
SingleCellExperiment, *9*, *17*

tenxBamqc, *5*, *11*, 16

validCb, 17