Package 'FLAMES'

September 24, 2023

Type Package

Title FLAMES: Full Length Analysis of Mutations and Splicing in long read RNA-seq data

Version 1.6.0

Date 2022-4-21

Description Semi-supervised isoform detection and annotation from both bulk and single-cell long read RNA-seq data. Flames provides automated pipelines for analysing isoforms, as well as intermediate functions for manual execution.

biocViews RNASeq, SingleCell, Transcriptomics, DataImport, DifferentialSplicing, AlternativeSplicing, GeneExpression

License GPL (>= 2)

Encoding UTF-8

Imports basilisk, bambu, Biostrings, BiocGenerics, circlize, ComplexHeatmap, cowplot, dplyr, DropletUtils, GenomicRanges, GenomicFeatures, GenomeInfoDb, ggplot2, ggbio, grid, gridExtra, igraph, jsonlite, magrittr, Matrix, parallel, reticulate, Rsamtools, rtracklayer, RColorBrewer, SingleCellExperiment, SummarizedExperiment, scater, S4Vectors, scuttle, stats, scran, stringr, MultiAssayExperiment, tidyr, utils, withr, zlibbioc,

Suggests BiocStyle, GEOquery, knitr, rmarkdown, markdown, BiocFileCache, R.utils, ShortRead, uwot, testthat (>= 3.0.0)

LinkingTo Rcpp, Rhtslib, zlibbioc

SystemRequirements GNU make, C++11

RoxygenNote 7.2.3

VignetteBuilder knitr

URL https://github.com/OliverVoogd/FLAMES

Config/testthat/edition 3

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

git_branch RELEASE_3_17

git_last_commit 31277b6

git_last_commit_date 2023-04-25

Date/Publication 2023-09-24

Author Luyi Tian [aut],
Oliver Voogd [aut, cre],
Jakob Schuster [aut],
Changqing Wang [aut],
Shian Su [aut],
Matthew Ritchie [ctb]

Maintainer Oliver Voogd <voogd.o@wehi.edu.au>

R topics documented:

`annotation_to_fasta` *GTF/GFF to FASTA conversion*

Description

convert the transcript annotation to transcriptome assembly as FASTA file.

Usage

```
annotation_to_fasta(isoform_annotation, genome_fa, outdir)
```

Arguments

isoform_annotation	Path to the annotation file (GTF/GFF3)
genome_fa	The file path to genome fasta file.
outdir	The path to directory to store the transcriptome as `transcript_assembly.fa`.

Value

Path to the outputted transcriptome assembly

Examples

```
fasta <- annotation_to_fasta(system.file("extdata/rps24.gtf.gz", package = "FLAMES"), system.file("extdata/rps24
cat(readChar(fasta, nchars = 1e3))
```

---

barcode_info_plots *Barcode demultiplexing QC plots*

Description

Plot the barcode demultiplexing statistics

Usage

```
barcode_info_plots(sce)
```

Arguments

sce	The `SingleCellExperiment` object from FLAMES pipeline, or the returned list from `find_barcode`

Value

a list of QC plots for the barcode demultiplexing step (`find_barcode`)

Examples

```
outdir <- tempfile()
dir.create(outdir)
bc_allow <- file.path(outdir, 'bc_allow.tsv')
R.utils::gunzip(filename = system.file('extdata/bc_allow.tsv.gz', package = 'FLAMES'), destname = bc_allow, remov
barcode_info <- find_barcode(
   fastq_dir = system.file('extdata/fastq', package = 'FLAMES'),
   stats_file = file.path(outdir, 'bc_stat'),
   out_fastq = file.path(outdir, 'demultiplexed.fq.gz'),
   ref_csv = bc_allow,
   MAX_DIST = 2,
   UMI_LEN = 10
)
barcode_info_plots(barcode_info)
```

---

bulk_long_pipeline *Pipeline for Bulk Data*

Description

Semi-supervised isofrom detection and annotation for long read data. This variant is meant for bulk samples. Specific parameters relating to analysis can be changed either through function arguments, or through a configuration JSON file.

Usage

```
bulk_long_pipeline(
  annotation,
  fastq,
  outdir,
  genome_fa,
  minimap2_dir = NULL,
  config_file = NULL
)
```

Arguments

annotation	The file path to the annotation file in GFF3 format
fastq	The file path to input fastq file
outdir	The path to directory to store all output files.
genome_fa	The file path to genome fasta file.
minimap2_dir	Path to the directory containing minimap2, if it is not in PATH. Only required if either or both of do_genome_align and do_read_realign are TRUE.
config_file	File path to the JSON configuration file. If specified, config_file overrides all configuration parameters

Details

By default FLAMES use minimap2 for read alignment. After the genome alignment step (`do_genome_align`), FLAMES summarizes the alignment for each read by grouping reads with similar splice junctions to get a raw isoform annotation (`do_isoform_id`). The raw isoform annotation is compared against the reference annotation to correct potential splice site and transcript start/end errors. Transcripts that have similar splice junctions and transcript start/end to the reference transcript are merged with the reference. This process will also collapse isoforms that are likely to be truncated transcripts. If `isoform_id_bambu` is set to `TRUE`, `bambu::bambu` will be used to generate the updated annotations. Next is the read realignment step (`do_read_realign`), where the sequence of each transcript from the update annotation is extracted, and the reads are realigned to this updated `transcript_assembly.fa` by minimap2. The transcripts with only a few full-length aligned reads are discarded. The reads are assigned to transcripts based on both alignment score, fractions of reads aligned and transcript coverage. Reads that cannot be uniquely assigned to transcripts or have low transcript coverage are discarded. The UMI transcript count matrix is generated by collapsing the reads with the same UMI in a similar way to what is done for short-read scRNA-seq data, but allowing for an edit distance of up to 2 by default. Most of the parameters, such as the minimal distance to splice site and minimal percentage of transcript coverage can be modified by the JSON configuration file (`config_file`).

The default parameters can be changed either through the function arguments are through the configuration JSON file `config_file`. the `pipeline_parameters` section specifies which steps are to be executed in the pipeline - by default, all steps are executed. The `isoform_parameters` section affects isoform detection - key parameters include:

- `Min_sup_cnt` which causes transcripts with less reads aligned than it's value to be discarded
- `MAX_TS_DIST` which merges transcripts with the same intron chain and TSS/TES distace less than `MAX_TS_DIST`
- `strand_specific` which specifies if reads are in the same strand as the mRNA (1), or the reverse complemented (-1) or not strand specific (0), which results in strand information being based on reference annotation.

Value

if `do_transcript_quantification` set to true, `bulk_long_pipeline` returns a SummarizedExperiment object, containing a count matrix as an assay, gene annotations under metadata, as well as a list of the other output files generated by the pipeline. The pipeline also outputs a number of output files into the given `outdir` directory. These output files generated by the pipeline are:

- transcript_count.csv.gz - a transcript count matrix (also contained in the SummarizedExperiment)
- isoform_annotated.filtered.gff3 - isoforms in gff3 format (also contained in the SummarizedExperiment)
- transcript_assembly.fa - transcript sequence from the isoforms
- align2genome.bam - sorted BAM file with reads aligned to genome
- realign2transcript.bam - sorted realigned BAM file using the transcript_assembly.fa as reference
- tss_tes.bedgraph - TSS TES enrichment for all reads (for QC)

if `do_transcript_quantification` set to false, nothing will be returned

See Also

sc_long_pipeline() for single cell data, SummarizedExperiment() for how data is outputted

Examples

```
# download the two fastq files, move them to a folder to be merged together
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <-
  "https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data"
# download the required fastq files, and move them to new folder
fastq1 <- bfc[[names(BiocFileCache::bfcadd(bfc, "Fastq1", paste(file_url, "fastq/sample1.fastq.gz", sep = "/")))]
fastq2 <- bfc[[names(BiocFileCache::bfcadd(bfc, "Fastq2", paste(file_url, "fastq/sample2.fastq.gz", sep = "/")))]
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, "annot.gtf", paste(file_url, "SIRV_isoforms_multi-fasta-anno
genome_fa <- bfc[[names(BiocFileCache::bfcadd(bfc, "genome.fa", paste(file_url, "SIRV_isoforms_multi-fasta_17061
fastq_dir <- paste(temp_path, "fastq_dir", sep = "/") # the downloaded fastq files need to be in a directory to be me
dir.create(fastq_dir)
file.copy(c(fastq1, fastq2), fastq_dir)
unlink(c(fastq1, fastq2)) # the original files can be deleted

outdir <- tempfile()
dir.create(outdir)
if (is.character(locate_minimap2_dir())) {
  se <- bulk_long_pipeline(
    annotation = annotation, fastq = fastq_dir, outdir = outdir, genome_fa = genome_fa,
    config_file = system.file("extdata/SIRV_config_default.json", package = "FLAMES")
  )

  se_2 <- create_se_from_dir(outdir = outdir, annotation = annotation)
}
```

`callBasilisk` *Internal utility function for simplifying calls to basiliskRun using a given basilisk environment*

Description

Internal utility function for simplifying calls to basiliskRun using a given basilisk environment

Usage

```
callBasilisk(env_name, FUN, ...)
```

Arguments

`env_name`	the name of the basilisk env (made through BasiliskEnvironment) to execute code within
`FUN`	the function to execute from with the basilisk environment
`...`	extra parameters required by FUN

Value

the result of 'FUN'

`combine_sce` *Combine SCE*

Description

Combine long- and short-read SingleCellExperiment objects

Usage

```
combine_sce(
  short_read_large,
  short_read_small,
  long_read_sce,
  remove_duplicates = FALSE
)
```

Arguments

`short_read_large`	The SCE object, or path to the HDF5 file, or folder containing the matrix file, corresponding to the larger short-read sample
`short_read_small`	The SCE object, or path to the HDF5 file, or folder containing the matrix file, corresponding to the smaller short-read sample
`long_read_sce`	The SCE object of the transcript counts, from the long-read pipelines.
`remove_duplicates`	determines whether cells with duplicated barcodes aer kept in the smaller library (they are always removed from the larger library)

Details

Takes the long-read SCE object from the long-read pipeline and the short-read SCE object, creates a `MultiAssayExperiment` object with the two `SingleCellExperiment` objects. Cells with duplicated barcodes are removed from the larger library.

Value

A `MultiAssayExperiment` object, with 'gene_counts' and 'transcript_counts' experiments.

Examples

```
library(SingleCellExperiment)
a <- SingleCellExperiment(assays = list(counts = matrix(rpois(100, 5), ncol = 10)))
b <- SingleCellExperiment(assays = list(counts = matrix(rpois(100, 5), ncol = 10)))
long_read <- SingleCellExperiment(assays = list(counts = matrix(rpois(100, 5), ncol = 10)))
colData(a)$Barcode <- paste0(1:10, '-1')
colData(b)$Barcode <- paste0(8:17, '-1')
colnames(long_read) <- as.character(2:11)
rownames(a) <- as.character(101:110)
rownames(b) <- as.character(103:112)
rownames(long_read) <- as.character(1001:1010)
combine_sce(short_read_large = a, short_read_small = b, long_read_sce = long_read)
```

---

create_config *Create Configuration File From Arguments*

Description

Create Configuration File From Arguments

Usage

```
create_config(outdir, type = "sc_5end", ...)
```

Arguments

outdir	the destination directory for the configuratio nfile
type	use an example config, available values: • "sc_5end" - config for 5' end ONT reads • "SIRV" - config for the SIRV example reads
...	Configuration parameters. • do_genome_align - Boolean. Specifies whether to run the genome alignment step. TRUE is recommended • do_isoform_id - Boolean. Specifies whether to run the isoform identification step. TRUE is recommended • do_read_realign - Boolean. Specifies whether to run the read realignment step. TRUE is recommended • do_transcript_quanti - Boolean. Specifies whether to run the transcript quantification step. TRUE is recommended • gen_raw_isoform - Boolean. • has_UMI - Boolean. Specifies if the data contains UMI. • max_dist - Maximum distance allowed when merging splicing sites in isoform consensus clustering.

- max_ts_dist - Maximum distance allowed when merging transcript start/end position in isoform consensus clustering.
- max_splice_match_dist - Maximum distance allowed when merging splice site called from the data and the reference annotation.
- min_fl_exon_len - Minimum length for the first exon outside the gene body in reference annotation. This is to correct the alignment artifact
- max_site_per_splice - Maximum transcript start/end site combinations allowed per splice chain
- min_sup_cnt - Minimum number of read support an isoform decrease this number will significantly increase the number of isoform detected.
- min_cnt_pct - Minimum percentage of count for an isoform relative to total count for the same gene.
- min_sup_pct - Minimum percentage of count for an splice chain that support a given transcript start/end site combination.
- strand_specific - 0, 1 or -1. 1 indicates if reads are in the same strand as mRNA, -1 indicates reads are reverse complemented, 0 indicates reads are not strand specific.
- remove_incomp_reads - The strenge of truncated isoform filtering. larger number means more stringent filtering.
- use_junctions - whether to use known splice junctions to help correct the alignment results
- no_flank - Boolean. for synthetic spike-in data. refer to Minimap2 document for detail
- use_annotation - Boolean. whether to use reference to help annotate known isoforms
- min_tr_coverage - Minimum percentage of isoform coverage for a read to be aligned to that isoform
- min_read_coverage - Minimum percentage of read coverage for a read to be uniquely aligned to that isoform

Details

Create a list object containing the arguments supplied in a format usable for the FLAMES pipeline. Also writes the object to a JSON file, which is located with the prefix 'config_' in the supplied `outdir`. Default values from `extdata/config_sclr_nanopore_5end.json` will be used for unprovided parameters.

Value

file path to the config file created

Examples

```
# create the default configuration file
outdir <- tempdir()
config <- create_config(outdir)
```

create_sce_from_dir *Create* SingleCellExperiment *object from* FLAMES *output folder*

Description

Create SingleCellExperiment object from FLAMES output folder

Usage

```
create_sce_from_dir(outdir, annotation)
```

Arguments

outdir	The folder containing FLAMES output files
annotation	(Optional) the annotation file that was used to produce the output files

Value

a list of SingleCellExperiment objects if multiple transcript matrices were found in the output folder, or a SingleCellExperiment object if only one were found

Examples

```
outdir <- tempfile()
dir.create(outdir)
bc_allow <- file.path(outdir, "bc_allow.tsv")
genome_fa <- file.path(outdir, "rps24.fa")
R.utils::gunzip(filename = system.file("extdata/bc_allow.tsv.gz", package = "FLAMES"), destname = bc_allow, remov
R.utils::gunzip(filename = system.file("extdata/rps24.fa.gz", package = "FLAMES"), destname = genome_fa, remove =
annotation <- system.file("extdata/rps24.gtf.gz", package = "FLAMES")

if (is.character(locate_minimap2_dir())) {
    sce <- FLAMES::sc_long_pipeline(
        genome_fa = genome_fa,
        fastq = system.file("extdata/fastq", package = "FLAMES"),
        annotation = annotation,
        outdir = outdir,
        match_barcode = TRUE,
        reference_csv = bc_allow
    )
    sce_2 <- create_sce_from_dir(outdir, annotation)
}
```

create_se_from_dir *Create* SummarizedExperiment *object from* FLAMES *output folder*

Description

Create SummarizedExperiment object from FLAMES output folder

Usage

```
create_se_from_dir(outdir, annotation)
```

Arguments

outdir	The folder containing FLAMES output files
annotation	(Optional) the annotation file that was used to produce the output files

Value

a SummarizedExperiment object

Examples

```
# download the two fastq files, move them to a folder to be merged together
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <-
  "https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data"
# download the required fastq files, and move them to new folder
fastq1 <- bfc[[names(BiocFileCache::bfcadd(bfc, "Fastq1", paste(file_url, "fastq/sample1.fastq.gz", sep = "/")))]
fastq2 <- bfc[[names(BiocFileCache::bfcadd(bfc, "Fastq2", paste(file_url, "fastq/sample2.fastq.gz", sep = "/")))]
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, "annot.gtf", paste(file_url, "SIRV_isoforms_multi-fasta-anno
genome_fa <- bfc[[names(BiocFileCache::bfcadd(bfc, "genome.fa", paste(file_url, "SIRV_isoforms_multi-fasta_17061
fastq_dir <- paste(temp_path, "fastq_dir", sep = "/") # the downloaded fastq files need to be in a directory to be me
dir.create(fastq_dir)
file.copy(c(fastq1, fastq2), fastq_dir)
unlink(c(fastq1, fastq2)) # the original files can be deleted

outdir <- tempfile()
dir.create(outdir)
if (is.character(locate_minimap2_dir())) {
  se <- bulk_long_pipeline(
    annotation = annotation, fastq = fastq_dir, outdir = outdir, genome_fa = genome_fa,
    config_file = system.file("extdata/SIRV_config_default.json", package = "FLAMES")
  )

  se_2 <- create_se_from_dir(outdir = outdir, annotation = annotation)
}
```

demultiplex_sockeye *Demultiplex reads using Sockeye outputs*

Description

Demultiplex reads using the `cell_umi_gene.tsv` file from Sockeye.

Usage

```
demultiplex_sockeye(fastq_dir, sockeye_tsv, out_fq)
```

Arguments

`fastq_dir`	The folder containing FASTQ files from Sockeye's output under `ingest/chunked_fastqs`.
`sockeye_tsv`	The `cell_umi_gene.tsv` file from Sockeye.
`out_fq`	The output FASTQ file.

Value

returns NULL

find_barcode *Match Cell Barcodes*

Description

Match cell barcodes in the given fastq directory with the given barcode allow-list. For each read, the left flanking sequence is located, FLAMES then takes the next 16 characters and match it to barcodes in the allow-list. If there is an unambigious match within the given edit distance (`MAX_DIST`), the barcode and following `UMI_LEN` characters are tirmmed, along with potential polyT tail. The trimmed read is then saved to `out_fastq`, with the identier field formatted as [barcode]_[UMI]#[original read ID].

Usage

```
find_barcode(
  fastq_dir,
  stats_file,
  out_fastq,
  ref_csv,
  MAX_DIST,
  UMI_LEN = 10L,
  left_seq = "CTACACGACGCTCTTCCGATCT",
  min_length = 20L,
  reverse_complement = TRUE,
  fixed_range = FALSE
)
```

Arguments

fastq_dir	directory containing fastq files to match
stats_file	NEEDED
out_fastq	output filename for matched barcodes
ref_csv	NEEDED
MAX_DIST	int; maximum edit distance
UMI_LEN	int; length of UMI sequences
left_seq	String; sequence that appears at the left of the barcode
min_length	int; minimum read length to be filtered after timming barcodes
reverse_complement	boolean; whether to check the reverse complement of the reads
fixed_range	boolean; deprecated, whether to skip finding flanking sequence by infering its position from previous reads. Setting to TRUE may decrease performance and accuracy.

Value

returns a list containing statistics of the reads demultiplexed.

Examples

```
outdir <- tempfile()
dir.create(outdir)
bc_allow <- file.path(outdir, 'bc_allow.tsv')
R.utils::gunzip(filename = system.file('extdata/bc_allow.tsv.gz', package = 'FLAMES'), destname = bc_allow, remov
find_barcode(
   fastq_dir = system.file('extdata/fastq', package = 'FLAMES'),
   stats_file = file.path(outdir, 'bc_stat'),
   out_fastq = file.path(outdir, 'demultiplexed.fq.gz'),
   ref_csv = bc_allow,
   MAX_DIST = 2,
   UMI_LEN = 10
)
```

---

find_isoform *Isoform identification*

Description

Long-read isoform identification with FLAMES or bambu.

Usage

```
find_isoform(annotation, genome_fa, genome_bam, outdir, config)
```

Arguments

annotation	Path to annotation file. If configured to use bambu, the annotation must be provided as GTF file.
genome_fa	The file path to genome fasta file.
genome_bam	File path to BAM alignment file. Multiple files could be provided.
outdir	The path to directory to store all output files.
config	Parsed FLAMES configurations.

Value

The updated annotation and the transcriptome assembly will be saved in the output folder as `isoform_annotated.gff3` (GTF if bambu is selected) and `transcript_assembly.fa` respectively.

Examples

```
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <- "https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data"
fastq1 <- bfc[[names(BiocFileCache::bfcadd(bfc, "Fastq1", paste(file_url, "fastq/sample1.fastq.gz", sep = "/")))]
genome_fa <- bfc[[names(BiocFileCache::bfcadd(bfc, "genome.fa", paste(file_url, "SIRV_isoforms_multi-fasta_17061
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, "annot.gtf", paste(file_url, "SIRV_isoforms_multi-fasta-anno
outdir <- tempfile()
dir.create(outdir)
if (is.character(locate_minimap2_dir())) {
   config <- jsonlite::fromJSON(system.file("extdata/SIRV_config_default.json", package = "FLAMES"))
    minimap2_align(
        config = config,
        fa_file = genome_fa,
        fq_in = fastq1,
        annot = annotation,
        outdir = outdir
    )
## Not run:
    find_isoform(
        annotation = annotation, genome_fa = genome_fa,
        genome_bam = file.path(outdir, "align2genome.bam"),
        outdir = outdir, config = config
    )

## End(Not run)
}
```

---

get_GRangesList *Parse FLAMES' GFF output*

Description

Parse FLAMES' GFF ouputs into a Genomic Ranges List

Usage

```
get_GRangesList(file)
```

Arguments

file	the GFF file to parse

Value

A Genomic Ranges List

Examples

```
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <- "https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data"
fastq1 <- bfc[[names(BiocFileCache::bfcadd(bfc, "Fastq1", paste(file_url, "fastq/sample1.fastq.gz", sep = "/")))]
genome_fa <- bfc[[names(BiocFileCache::bfcadd(bfc, "genome.fa", paste(file_url, "SIRV_isoforms_multi-fasta_17061
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, "annot.gtf", paste(file_url, "SIRV_isoforms_multi-fasta-anno
outdir <- tempfile()
dir.create(outdir)
if (is.character(locate_minimap2_dir())) {
   config <- jsonlite::fromJSON(system.file("extdata/SIRV_config_default.json", package = "FLAMES"))
    minimap2_align(
        config = config,
        fa_file = genome_fa,
        fq_in = fastq1,
        annot = annotation,
        outdir = outdir
    )
    find_isoform(
        annotation = annotation, genome_fa = genome_fa,
        genome_bam = file.path(outdir, "align2genome.bam"),
        outdir = outdir, config = config
    )
    grlist <- get_GRangesList(file = file.path(outdir, "isoform_annotated.gff3"))
}
```

---

locate_minimap2_dir *locate parent folder of minimap2*

Description

locate minimap2, or validate that minimap2 exists under minimap2_dir

Usage

```
locate_minimap2_dir(minimap2_dir = NULL)
```

Arguments

minimap2_dir	(Optional) folder that includes minimap2

Value

Path to the parent folder of minimap2 if available, or FALSE if minimap2 could not be found.

Examples

```
locate_minimap2_dir()
```

---

minimap2_align *Minimap2 Align to Genome*

Description

Uses minimap2 to align sequences agains a reference databse. Uses options '-ax splice -t 12 -k14 –secondary=no fa_file fq_in'

Usage

```
minimap2_align(
  config,
  fa_file,
  fq_in,
  annot,
  outdir,
  minimap2_dir,
  samtools = NULL,
  prefix = NULL,
  threads = NULL
)
```

Arguments

config	Parsed list of FLAMES config file
fa_file	Path to the fasta file used as a reference database for alignment
fq_in	File path to the fastq file used as a query sequence file
annot	Genome annotation file used to create junction bed files
outdir	Output folder
minimap2_dir	Path to the directory containing minimap2
samtools	path to the samtools binary, required for large datasets since Rsamtools does not support CSI indexing
prefix	String, the prefix (e.g. sample name) for the outputted BAM file
threads	Integer, threads for minimap2 to use, see minimap2 documentation for details, FLAMES will try to detect cores if this parameter is not provided.

Value

a data.frame summarising the reads aligned

See Also

[minimap2_realign()]

Examples

```
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <- 'https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data'
fastq1 <- bfc[[names(BiocFileCache::bfcadd(bfc, 'Fastq1', paste(file_url, 'fastq/sample1.fastq.gz', sep = '/')))]
genome_fa <- bfc[[names(BiocFileCache::bfcadd(bfc, 'genome.fa', paste(file_url, 'SIRV_isoforms_multi-fasta_17061
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, 'annot.gtf', paste(file_url, 'SIRV_isoforms_multi-fasta-anno
outdir <- tempfile()
dir.create(outdir)
if (is.character(locate_minimap2_dir())) {
    minimap2_align(
     config = jsonlite::fromJSON(system.file('extdata/SIRV_config_default.json', package = 'FLAMES')),
        fa_file = genome_fa,
        fq_in = fastq1,
        annot = annotation,
        outdir = outdir
    )
}
```

---

minimap2_realign *Minimap2 re-align reads to transcriptome*

Description

Uses minimap2 to re-align reads to transcriptome

Usage

```
minimap2_realign(
  config,
  fq_in,
  outdir,
  minimap2_dir,
  prefix = NULL,
  threads = NULL
)
```

Arguments

config	Parsed list of FLAMES config file
fq_in	File path to the fastq file used as a query sequence file
outdir	Output folder
minimap2_dir	Path to the directory containing minimap2
prefix	String, the prefix (e.g. sample name) for the outputted BAM file
threads	Integer, threads for minimap2 to use, see minimap2 documentation for details, FLAMES will try to detect cores if this parameter is not provided.

Value

a data.frame summarising the reads aligned

See Also

[minimap2_align()]

Examples

```
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <- 'https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data'
fastq1 <- bfc[[names(BiocFileCache::bfcadd(bfc, 'Fastq1', paste(file_url, 'fastq/sample1.fastq.gz', sep = '/')))]
genome_fa <- bfc[[names(BiocFileCache::bfcadd(bfc, 'genome.fa', paste(file_url, 'SIRV_isoforms_multi-fasta_17061
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, 'annot.gtf', paste(file_url, 'SIRV_isoforms_multi-fasta-anno
outdir <- tempfile()
dir.create(outdir)
if (is.character(locate_minimap2_dir())) {
    fasta <- annotation_to_fasta(annotation, genome_fa, outdir)
    minimap2_realign(
     config = jsonlite::fromJSON(system.file('extdata/SIRV_config_default.json', package = 'FLAMES')),
        fq_in = fastq1,
        outdir = outdir
    )
}
```

---

parse_gff_tree *Parse Gff3 file*

Description

Parse a Gff3 file into 3 components: chromasome to gene name, a transcript dictionary, a gene to transcript dictionary and a transcript to exon dictionary. These components are returned in a named list.

Usage

```
parse_gff_tree(gff_file)
```

Arguments

gff_file	the file path to the gff3 file to parse

Value

a named list with the elements "chr_to_gene", "transcript_dict", "gene_to_transcript", "transcript_to_exon", containing the data parsed from the gff3 file.

Examples

```
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <-
    "https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data"
gff <- bfc[[names(BiocFileCache::bfcadd(bfc, "GFF", paste(file_url, "SIRV_isoforms_multi-fasta-annotation_C_1706
## Not run: parsed_gff <- parse_gff_tree(gff)
```

---

quantify *Transcript quantification*

Description

Calculate the transcript count matrix by parsing the re-alignment file.

Usage

```
quantify(annotation, outdir, config, pipeline = "sc_single_sample")
```

Arguments

annotation	The file path to the annotation file in GFF3 format
outdir	The path to directory to store all output files.
config	Parsed FLAMES configurations.
pipeline	The pipeline type as a character string, either `sc_single_sample` (single-cell, single-sample), `bulk` (bulk, single or multi-sample), or `sc_multi_sample` (single-cell, multiple samples)

Value

The count matrix will be saved in the output folder as `transcript_count.csv.gz`.

Examples

```
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <- "https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data"
fastq1 <- bfc[[names(BiocFileCache::bfcadd(bfc, "Fastq1", paste(file_url, "fastq/sample1.fastq.gz", sep = "/")))]
genome_fa <- bfc[[names(BiocFileCache::bfcadd(bfc, "genome.fa", paste(file_url, "SIRV_isoforms_multi-fasta_17061
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, "annot.gtf", paste(file_url, "SIRV_isoforms_multi-fasta-anno
outdir <- tempfile()
dir.create(outdir)
fasta <- annotation_to_fasta(annotation, genome_fa, outdir)
config <- jsonlite::fromJSON(create_config(outdir, bambu_isoform_identification = TRUE, min_tr_coverage = 0.1, mi
file.copy(annotation, file.path(outdir, "isoform_annotated.gtf"))
## Not run:
if (is.character(locate_minimap2_dir())) {
    minimap2_realign(
        config = config, outdir = outdir,
        fq_in = fastq1
    )
    quantify(annotation, outdir, config, pipeline = "bulk")
}

## End(Not run)
```

`sc_annotate_plots` *FLAMES Annotated Plottings*

Description

(deprecated) Plot isoform exons alignments for a given gene, along with UMAP showing expression levels. Superseded by `sc_umap_expression` and `sc_heatmap_expression`.

Usage

```
sc_annotate_plots(
  gene,
  multiAssay,
  cluster_annotation,
  n_isoforms = 4,
  n_pcs = 40,
  return_multiAssay = TRUE,
  heatmap_annotation_colors = "BrBG",
  isoform_legend_width = 7,
  heatmap_color_quantile = 0.95,
  col_low = "#313695",
  col_mid = "#FFFFBF",
  col_high = "#A50026"
)
```

Arguments

gene	The gene symbol of interest.
multiAssay	The MultiAssayExperiment object from combine_sce().
cluster_annotation	Path to the cluster annotation CSV (required for heatmap, if cluster_annotation.csv is not in path and multiAssay$cell_type does not exist)
n_isoforms	The number of expressed isoforms to keep.
n_pcs	The number of principal components to generate.
return_multiAssay	Whether to return the processed MultiAssayExperiment object.
heatmap_annotation_colors	Name of color palette to use for cell group annotation in heatmaps, see RColorBrewer::brewer.pal() available diverging palettes are: BrBG PiYG PRGn PuOr RdBu RdGy RdYlBu RdYlGn Spectral when there are more than 11 groups, this argument will be ignored and random palettes will be generated.
isoform_legend_width	The width of isoform legends in heatmaps, in cm.
heatmap_color_quantile	Float; Expression levels higher than this quantile will all be shown with col_high. Expression levels lower than 1 - heatmap_color_quantile will all be shown with col_low;
col_low	Color for cells with low expression levels in UMAPs.
col_mid	Color for cells with intermediate expression levels in UMAPs.
col_high	Color for cells with high expression levels in UMAPs.

Details

This function takes a combined MultiAssayExperiment object containing 'gene_counts' and 'transcript_counts' experiments to generate a combined UMAP, the expression levels of isoforms (using long read data) are then overlayed on top of the UMAP. SNN inference based on gene counts were performed to impute isoform expression for cells in the larger library. The MultiAssayExperiment object should have a boolean columen named 'Lib_small' in its columen data file to idicate which subsample the cells are in. The MultiAssayExperiment object can be created using the combine_sce function.

Value

a list containing the combined UMAP, the isoform exon alignments and the UMAP with isoform expression levels.

See Also

combine_sce() for combining gene count SingleCellExperiment object with transcript counts.

`sc_DTU_analysis` *FLAMES Differential Transcript Usage Analysis*

Description

Chi-square based differential transcription usage analysis. This variant is meant for single cell data. Takes the `SingleCellExperiment` object from `sc_long_pipeline` as input. Alternatively, the path to the output folder could be provided instead of the SCE object. A cluster annotation file `cluster_annotation.csv` is required, please provide this file under the output folder of `sc_long_pipeline`.

Usage

```
sc_DTU_analysis(sce, path, min_count = 15)
```

Arguments

`sce`	The `SingleCellExperiment` object from `sc_long_pipeline`, an additional `cluster_annotation.csv` file is required under the output folder of the SCE object.
`path`	The path to the output folder of `sc_long_pipeline` the folder needs to contain: • transcript_count.csv.gz - the transcript count matrix • isoform_FSM_annotation.csv - the full splice match annotation file • cluster_annotation.csv - cluster annotation file
`min_count`	The minimum UMI count threshold for filtering isoforms.

Details

This function will search for genes that have at least two isoforms, each with more than `min_count` UMI counts. For each gene, the per cell transcript counts were merged by group to generate pseudo bulk samples. Grouping is specified by the `cluster_annotation.csv` file. The top 2 highly expressed transcripts for each group were selected and a UMI count matrix where the rows are selected transcripts and columns are groups was used as input to a chi-square test of independence (chisq.test). Adjusted P-values were calculated by Benjamini–Hochberg correction.

Value

a `data.frame` containing the following columns:

- gene_name - differentially transcribed genes
- X_value - the X value for the DTU gene
- df - degrees of freedom of the approximate chi-squared distribution of the test statistic
- DTU_tr - the transcript_id with the highest squared residuals
- DTU_group - the cell group with the highest squared residuals
- p_value - the p-value for the test

- adj_p - the adjusted p-value (by Benjamini–Hochberg correction)

The table is sorted by decreasing P-values. It will also be saved as `sc_DTU_analysis.csv` under the output folder.

Examples

```
outdir <- tempfile()
dir.create(outdir)
bc_allow <- file.path(outdir, "bc_allow.tsv")
genome_fa <- file.path(outdir, "rps24.fa")
R.utils::gunzip(filename = system.file("extdata/bc_allow.tsv.gz", package = "FLAMES"), destname = bc_allow, remov
R.utils::gunzip(filename = system.file("extdata/rps24.fa.gz", package = "FLAMES"), destname = genome_fa, remove =

if (is.character(locate_minimap2_dir())) {
  sce <- FLAMES::sc_long_pipeline(
    genome_fa = genome_fa,
    fastq = system.file("extdata/fastq", package = "FLAMES"),
    annotation = system.file("extdata/rps24.gtf.gz", package = "FLAMES"),
    outdir = outdir,
    match_barcode = TRUE,
    reference_csv = bc_allow
  )
 group_anno <- data.frame(barcode_seq = colnames(sce), groups = SingleCellExperiment::counts(sce)["ENSMUST000001
  write.csv(group_anno, file.path(outdir, "cluster_annotation.csv"), row.names = FALSE)
  sc_DTU_analysis(sce, outdir, min_count = 1)
}
```

---

sc_heatmap_expression *FLAMES heetmap plots*

Description

Plot expression heatmap of top n isoforms of a gene

Usage

```
sc_heatmap_expression(
  gene,
  multiAssay,
  impute = FALSE,
  n_isoforms = 4,
  transcript_ids,
  n_pcs = 40,
  isoform_legend_width = 7,
  col_low = "#313695",
  col_mid = "#FFFFBF",
  col_high = "#A50026",
  color_quantile = 0.95
)
```

Arguments

gene	The gene symbol of interest.
multiAssay	The MultiAssayExperiment object from combine_sce().
impute	Whether to impute expression levels for cells without transcript counts
n_isoforms	The number of expressed isoforms to keep.
transcript_ids	specify the transcript ids instead of selecting the top n_isoforms
n_pcs	The number of principal components to generate.
isoform_legend_width	The width of isoform legends in heatmaps, in cm.
col_low	Color for cells with low expression levels in UMAPs.
col_mid	Color for cells with intermediate expression levels in UMAPs.
col_high	Color for cells with high expression levels in UMAPs.
color_quantile	The lower and upper expression quantile to be displayed bewteen col_low and col_high, e.g. with color_quantile = 0.95, cells with expressions higher than 95% of other cells will all be shown in col_high, and cells with expression lower than 95% of other cells will all be shown in col_low.

Details

This function takes the combined MultiAssayExperiment object from combine_sce and plots an expression heatmap with the isoform alignment visualisations.

Value

a ggplot object of the heatmap

Examples

```
source(system.file("examples/scmixology1.R", package = "FLAMES"))
scm_lib80 <- scuttle::addPerCellQC(scm_lib80)
scm_lib20 <- scuttle::addPerCellQC(scm_lib20)
qc_80 <- scuttle::quickPerCellQC(colData(scm_lib80))
qc_20 <- scuttle::quickPerCellQC(colData(scm_lib20))
qc_80$discard[scm_lib80$sum < 10000] <- TRUE
qc_20$discard[scm_lib20$sum < 20000] <- TRUE

combined_sce <- combine_sce(
    short_read_large = scm_lib80[,!qc_80$discard],
    short_read_small = scm_lib20[,!qc_20$discard],
    long_read_sce = scm_lib20_transcripts,
    remove_duplicates = FALSE)

sc_heatmap_expression(gene = "ENSG00000108107", multiAssay = combined_sce)
```

`sc_long_multisample_pipeline` *Pipeline for Multi-sample Single Cell Data*

Description

Semi-supervised isoform detection and annotation for long read data. This variant is for multi-sample single cell data. By default, this pipeline demultiplexes input fastq data (`match_cell_barcode = TRUE`). Specific parameters relating to analysis can be changed either through function arguments, or through a configuration JSON file.

Usage

```
sc_long_multisample_pipeline(
  annotation,
  fastqs,
  outdir,
  genome_fa,
  minimap2_dir = NULL,
  reference_csv,
  match_barcode = TRUE,
  config_file = NULL
)
```

Arguments

`annotation`	The file path to the annotation file in GFF3 format
`fastqs`	Paths to the folder containing fastq files, or vector of paths to each fastq file.
`outdir`	The path to directory to store all output files.
`genome_fa`	The file path to genome fasta file.
`minimap2_dir`	Path to the directory containing minimap2, if it is not in PATH. Only required if either or both of `do_genome_align` and `do_read_realign` are `TRUE`.
`reference_csv`	The file path to the reference csv used for demultiplexing
`match_barcode`	Boolean; specifies if demultiplexing should be performed using `FLAMES::find_barcode`
`config_file`	File path to the JSON configuration file. If specified, `config_file` overrides all configuration parameters

Details

By default FLAMES use minimap2 for read alignment. After the genome alignment step (`do_genome_align`), FLAMES summarizes the alignment for each read in every sample by grouping reads with similar splice junctions to get a raw isoform annotation (`do_isoform_id`). The raw isoform annotation is compared against the reference annotation to correct potential splice site and transcript start/end errors. Transcripts that have similar splice junctions and transcript start/end to the reference transcript are merged with the reference. This process will also collapse isoforms that are likely to be

truncated transcripts. If `isoform_id_bambu` is set to `TRUE`, `bambu::bambu` will be used to generate the updated annotations (Not implemented for multi-sample yet). Next is the read realignment step (`do_read_realign`), where the sequence of each transcript from the update annotation is extracted, and the reads are realigned to this updated `transcript_assembly.fa` by minimap2. The transcripts with only a few full-length aligned reads are discarded (Not implemented for multi-sample yet). The reads are assigned to transcripts based on both alignment score, fractions of reads aligned and transcript coverage. Reads that cannot be uniquely assigned to transcripts or have low transcript coverage are discarded. The UMI transcript count matrix is generated by collapsing the reads with the same UMI in a similar way to what is done for short-read scRNA-seq data, but allowing for an edit distance of up to 2 by default. Most of the parameters, such as the minimal distance to splice site and minimal percentage of transcript coverage can be modified by the JSON configuration file (`config_file`).

The default parameters can be changed either through the function arguments are through the configuration JSON file `config_file`. the `pipeline_parameters` section specifies which steps are to be executed in the pipeline - by default, all steps are executed. The `isoform_parameters` section affects isoform detection - key parameters include:

- `Min_sup_cnt` which causes transcripts with less reads aligned than it's value to be discarded
- `MAX_TS_DIST` which merges transcripts with the same intron chain and TSS/TES distace less than `MAX_TS_DIST`
- `strand_specific` which specifies if reads are in the same strand as the mRNA (1), or the reverse complemented (-1) or not strand specific (0), which results in strand information being based on reference annotation.

Value

a list of `SingleCellExperiment` objects if "do_transcript_quantification" set to true. Otherwise nothing will be returned.

See Also

`bulk_long_pipeline()` for bulk long data, `SingleCellExperiment()` for how data is outputted

Examples

```
reads <- ShortRead::readFastq(system.file("extdata/fastq/musc_rps24.fastq.gz", package = "FLAMES"))
outdir <- tempfile()
dir.create(outdir)
dir.create(file.path(outdir, "fastq"))
bc_allow <- file.path(outdir, "bc_allow.tsv")
genome_fa <- file.path(outdir, "rps24.fa")
R.utils::gunzip(filename = system.file("extdata/bc_allow.tsv.gz", package = "FLAMES"), destname = bc_allow, remov
R.utils::gunzip(filename = system.file("extdata/rps24.fa.gz", package = "FLAMES"), destname = genome_fa, remove =
ShortRead::writeFastq(sample(reads, size = 500, replace = TRUE), file.path(outdir, "fastq/sample1.fq.gz"), mode =
ShortRead::writeFastq(sample(reads, size = 500, replace = TRUE), file.path(outdir, "fastq/sample2.fq.gz"), mode =
ShortRead::writeFastq(sample(reads, size = 500, replace = TRUE), file.path(outdir, "fastq/sample3.fq.gz"), mode =

if (is.character(locate_minimap2_dir())) {
    sce_list <- FLAMES::sc_long_multisample_pipeline(
        annotation = system.file("extdata/rps24.gtf.gz", package = "FLAMES"),
```

```
        fastqs = file.path(outdir, "fastq", list.files(file.path(outdir, "fastq"))),
        outdir = outdir,
        genome_fa = genome_fa,
        reference_csv = rep(bc_allow, 3)
    )
}
```

`sc_long_pipeline` *Pipeline for Single Cell Data*

Description

Semi-supervised isoform detection and annotation for long read data. This variant is for single cell data. By default, this pipeline demultiplexes input fastq data (`match_cell_barcode = TRUE`). Specific parameters relating to analysis can be changed either through function arguments, or through a configuration JSON file.

Usage

```
sc_long_pipeline(
  annotation,
  fastq,
  genome_bam = NULL,
  outdir,
  genome_fa,
  minimap2_dir = NULL,
  reference_csv,
  match_barcode,
  config_file = NULL
)
```

Arguments

`annotation`	The file path to the annotation file in GFF3 format
`fastq`	The file path to input fastq file
`genome_bam`	Optional file path to a bam file to use instead of fastq file (skips initial alignment step)
`outdir`	The path to directory to store all output files.
`genome_fa`	The file path to genome fasta file.
`minimap2_dir`	Path to the directory containing minimap2, if it is not in PATH. Only required if either or both of `do_genome_align` and `do_read_realign` are `TRUE`.
`reference_csv`	The file path to the reference csv used for demultiplexing
`match_barcode`	Boolean; specifies if demultiplexing should be performed using `FLAMES::find_barcode`
`config_file`	File path to the JSON configuration file. If specified, `config_file` overrides all configuration parameters

Details

By default FLAMES use minimap2 for read alignment. After the genome alignment step (`do_genome_align`), FLAMES summarizes the alignment for each read by grouping reads with similar splice junctions to get a raw isoform annotation (`do_isoform_id`). The raw isoform annotation is compared against the reference annotation to correct potential splice site and transcript start/end errors. Transcripts that have similar splice junctions and transcript start/end to the reference transcript are merged with the reference. This process will also collapse isoforms that are likely to be truncated transcripts. If `isoform_id_bambu` is set to `TRUE`, `bambu::bambu` will be used to generate the updated annotations. Next is the read realignment step (`do_read_realign`), where the sequence of each transcript from the update annotation is extracted, and the reads are realigned to this updated `transcript_assembly.fa` by minimap2. The transcripts with only a few full-length aligned reads are discarded. The reads are assigned to transcripts based on both alignment score, fractions of reads aligned and transcript coverage. Reads that cannot be uniquely assigned to transcripts or have low transcript coverage are discarded. The UMI transcript count matrix is generated by collapsing the reads with the same UMI in a similar way to what is done for short-read scRNA-seq data, but allowing for an edit distance of up to 2 by default. Most of the parameters, such as the minimal distance to splice site and minimal percentage of transcript coverage can be modified by the JSON configuration file (`config_file`).

The default parameters can be changed either through the function arguments are through the configuration JSON file `config_file`. the `pipeline_parameters` section specifies which steps are to be executed in the pipeline - by default, all steps are executed. The `isoform_parameters` section affects isoform detection - key parameters include:

- `Min_sup_cnt` which causes transcripts with less reads aligned than it's value to be discarded
- `MAX_TS_DIST` which merges transcripts with the same intron chain and TSS/TES distace less than `MAX_TS_DIST`
- `strand_specific` which specifies if reads are in the same strand as the mRNA (1), or the reverse complemented (-1) or not strand specific (0), which results in strand information being based on reference annotation.

Value

if `do_transcript_quantification` set to true, `sc_long_pipeline` returns a `SingleCellExperiment` object, containing a count matrix as an assay, gene annotations under metadata, as well as a list of the other output files generated by the pipeline. The pipeline also outputs a number of output files into the given `outdir` directory. These output files generated by the pipeline are:

- transcript_count.csv.gz - a transcript count matrix (also contained in the SingleCellExperiment)
- isoform_annotated.filtered.gff3 - isoforms in gff3 format (also contained in the SingleCellExperiment)
- transcript_assembly.fa - transcript sequence from the isoforms
- align2genome.bam - sorted BAM file with reads aligned to genome
- realign2transcript.bam - sorted realigned BAM file using the transcript_assembly.fa as reference
- tss_tes.bedgraph - TSS TES enrichment for all reads (for QC)

if `do_transcript_quantification` set to false, nothing will be returned

See Also

bulk_long_pipeline() for bulk long data, SingleCellExperiment() for how data is outputted

Examples

```
outdir <- tempfile()
dir.create(outdir)
bc_allow <- file.path(outdir, "bc_allow.tsv")
genome_fa <- file.path(outdir, "rps24.fa")
R.utils::gunzip(filename = system.file("extdata/bc_allow.tsv.gz", package = "FLAMES"), destname = bc_allow, remov
R.utils::gunzip(filename = system.file("extdata/rps24.fa.gz", package = "FLAMES"), destname = genome_fa, remove =

if (is.character(locate_minimap2_dir())) {
    sce <- FLAMES::sc_long_pipeline(
        genome_fa = genome_fa,
        fastq = system.file("extdata/fastq", package = "FLAMES"),
        annotation = system.file("extdata/rps24.gtf.gz", package = "FLAMES"),
        outdir = outdir,
        match_barcode = TRUE,
        reference_csv = bc_allow
    )
}
```

---

sc_mutations *FLAMES variant calling*

Description

Candidate SNVs identified with filtering by coverage threshold, and allele frequency range.

Usage

```
sc_mutations(
  sce,
  barcode_tsv,
  bam_short,
  out_dir,
  genome_fa,
  annot,
  known_positions = NULL,
  min_cov = 100,
  report_pct = c(0.1, 0.9)
)
```

Arguments

sce	The SingleCellExperiment object from sc_long_pipeline
barcode_tsv	TSV file for cell barcodes

bam_short (Optional) short read alignment BAM file. If provided, it is used to filter the variations. Variations in long-read data with enough short read coverage but no alternative allele will not be reported.

out_dir (Optional) Output folder of sc_long_pipeline. Output files from this function will also be saved here. Use this parameter if you do not have the SingleCellExperiment object.

genome_fa (Optional) Reference genome FASTA file. Use this parameter is if you do not wish sc_mutation to use the reference genome FASTA file from the sce's metadata.

annot (Optional) The file path to gene annotation file in gff3 format. If provided as FALSE then the isoform_annotated.gff3 from sc_longread_pipeline will be used, if not provided then the path in the SingleCellExperiment object will be used.

known_positions (Optional) A list of known positions, with by chromosome name followed by the position, e.g. ('chr1', 123, 'chr1', 124, 'chrX', 567). These locations will not be filtered and its allele frequencies will be reported.

min_cov The coverage threshold for filtering candidate SNVs. Positions with reads less then this number will not be considered.

report_pct The allele frequency range for filtering candidate SNVs. Positions with less or higher allele frequency will not be reported. The default is 0.10-0.90

Details

Takes the SingleCellExperiment object from sc_long_pipeline and the cell barcodes as barcode. Alternatively, input can also be provided as out_dir, genome_fa, annot, barcode.

Value

a data.frame containing the following columns:

- chr - the chromosome where the mutation is located
- position
- REF - the reference allele
- ALT - the alternative allele
- REF_frequency - reference allele frequency
- REF_frequency_in_short_reads - reference allele frequency in short reads (-1 when short reads not provided)
- hypergeom_test_p_value
- sequence_entropy
- INDEL_frequency
- adj_p - the adjusted p-value (by Benjamini–Hochberg correction)

The table is sorted by decreasing adjusted P value.

files saved to out_dir/mutation:

- ref_cnt.csv.gz
- alt_cnt.csv.gz
- allele_stat.csv.gz
- freq_summary.csv

Examples

```
outdir <- tempfile()
dir.create(outdir)
bc_allow <- file.path(outdir, "bc_allow.tsv")
genome_fa <- file.path(outdir, "rps24.fa")
R.utils::gunzip(filename = system.file("extdata/bc_allow.tsv.gz", package = "FLAMES"), destname = bc_allow, remov
R.utils::gunzip(filename = system.file("extdata/rps24.fa.gz", package = "FLAMES"), destname = genome_fa, remove =

## Not run:
if (is.character(locate_minimap2_dir())) {
    sce <- FLAMES::sc_long_pipeline(
        genome_fa = genome_fa,
        fastq = system.file("extdata/fastq", package = "FLAMES"),
        annotation = system.file("extdata/rps24.gtf.gz", package = "FLAMES"),
        outdir = outdir,
        match_barcode = TRUE,
        reference_csv = bc_allow
    )
   sc_mutations(sce, barcode_tsv = file.path(outdir, "bc_allow.tsv"), min_cov = 2, report_pct = c(0, 1))
}

## End(Not run)
```

---

sc_umap_expression *FLAMES UMAP plots*

Description

Plot expression UMAPs of top n isoforms of a gene

Usage

```
sc_umap_expression(
  gene,
  multiAssay,
  impute = FALSE,
  grided = TRUE,
  n_isoforms = 4,
  transcript_ids,
  n_pcs = 40,
  col_low = "#313695",
  col_mid = "#FFFFBF",
  col_high = "#A50026"
)
```

Arguments

gene	The gene symbol of interest.
multiAssay	The MultiAssayExperiment object from combine_sce().
impute	Whether to impute expression levels for cells without transcript counts
grided	Wheter to produce multiple UMAP plots, with each showing expression level for an isoform, to allow plotting more than 2 isoforms.
n_isoforms	The number of expressed isoforms to keep. n_isoforms > 2 requires girded = TRUE
transcript_ids	specify the transcript ids instead of selecting the top n_isoforms
n_pcs	The number of principal components to generate.
col_low	Color for cells with low expression levels in UMAPs.
col_mid	Color for cells with intermediate expression levels in UMAPs.
col_high	Color for cells with high expression levels in UMAPs.

Details

This function takes the combined MultiAssayExperiment object from cexample("MultiAssayExperiment")ombine_sce and plots UMAPs for each isoform of gene, where cells are colored by expression levels. When grided = TRUE, the UMAPs are combined into a grid, along with the isoforms' visualization along genomic coordinates. Produces a single UMAP with isoform expressions colored by col_low and col_high when grided = FALSE.

Value

a ggplot object of the UMAP(s)

Examples

```
source(system.file("examples/scmixology1.R", package = "FLAMES"))
scm_lib80 <- scuttle::addPerCellQC(scm_lib80)
scm_lib20 <- scuttle::addPerCellQC(scm_lib20)
qc_80 <- scuttle::quickPerCellQC(colData(scm_lib80))
qc_20 <- scuttle::quickPerCellQC(colData(scm_lib20))
qc_80$discard[scm_lib80$sum < 10000] <- TRUE
qc_20$discard[scm_lib20$sum < 20000] <- TRUE

combined_sce <- combine_sce(
    short_read_large = scm_lib80[,!qc_80$discard],
    short_read_small = scm_lib20[,!qc_20$discard],
    long_read_sce = scm_lib20_transcripts,
    remove_duplicates = FALSE)

sc_umap_expression(gene = "ENSG00000108107", multiAssay = combined_sce)
```

Index