Package 'FLAMES'

April 2, 2025

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

Version 2.1.8

Date 2023-03-27

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, LongRead

BugReports https://github.com/mritchielab/FLAMES/issues

License GPL (>= 3)

Encoding UTF-8

Imports basilisk, bambu, BiocParallel, Biostrings, BiocGenerics, circlize, ComplexHeatmap, cowplot, dplyr, DelayedArray, DropletUtils, GenomicRanges, GenomicFeatures, txdbmaker, GenomicAlignments, GenomeInfoDb, ggplot2, ggbio, grid, gridExtra, igraph, jsonlite, magrittr, magick, Matrix, MatrixGenerics, readr, reticulate, Rsamtools, rtracklayer, RColorBrewer, SingleCellExperiment, SummarizedExperiment, SpatialExperiment, scater, scatterpie, S4Vectors, scuttle, stats, scran, stringr, tidyr, utils, withr, future, methods, tibble, tidyselect, IRanges

Suggests BiocStyle, GEOquery, knitr, rmarkdown, BiocFileCache, R.utils, ShortRead, uwot, testthat (>= 3.0.0), xml2

LinkingTo Rcpp, Rhtslib, testthat

SystemRequirements GNU make, C++17, samtools (>= 1.19), minimap2 (>= 2.17)

RoxygenNote 7.3.2

VignetteBuilder knitr

URL https://mritchielab.github.io/FLAMES

Config/testthat/edition 3

Depends R (>= 4.1.0)

LazyData true

LazyLoad yes

StagedInstall no

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

git_branch devel

git_last_commit 67e4189

git_last_commit_date 2025-03-31

Repository Bioconductor 3.21

Date/Publication 2025-04-02

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

Maintainer Changqing Wang <wang.ch@wehi.edu.au>

Contents

`addRowRanges` *Add rowRanges by rownames to* `SummarizedExperiment` *object Assumes rownames are transcript_ids Assumes transcript_id is present in the annotation file*

Description

Add rowRanges by rownames to `SummarizedExperiment` object Assumes rownames are transcript_ids Assumes transcript_id is present in the annotation file

Usage

```
addRowRanges(sce, annotation, outdir)
```

Value

a `SummarizedExperiment` object with rowRanges added

add_gene_counts *Add gene counts to a* SingleCellExperiment *object*

Description

Add gene counts to a SingleCellExperiment object as an altExps slot named gene.

Usage

```
add_gene_counts(sce, gene_count_file)
```

Arguments

sce	A SingleCellExperiment object.
gene_count_file	The file path to the gene count file. If missing, the function will try to find the gene count file in the output directory.

Examples

```
# Set up a mock SingleCellExperiment object
sce <- SingleCellExperiment::SingleCellExperiment(
  assays = list(counts = matrix(0, nrow = 10, ncol = 10))
)
colnames(sce) <- paste0('cell', 1:10)
# Set up a mock gene count file
gene_count_file <- tempfile()
gene_mtx <- matrix(1:10, nrow = 2, ncol = 5)
colnames(gene_mtx) <- paste0('cell', 1:5)
rownames(gene_mtx) <- c("gene1", "gene2")
write.csv(gene_mtx, gene_count_file)
# Add gene counts to the SingleCellExperiment object
sce <- add_gene_counts(sce, gene_count_file)
# verify the gene counts are added
SingleCellExperiment::altExps(sce)$gene
```

annotation_to_fasta *GTF/GFF to FASTA conversion*

Description

convert the transcript annotation to transcriptome assembly as FASTA file. The genome annotation is first imported as TxDb object and then used to extract transcript sequence from the genome assembly.

Usage

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

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`.
extract_fn	(optional) Function to extract `GRangesList` from the genome TxDb object. E.g. `function(txdb){GenomicFeatures::cdsBy(txdb, by="tx", use.names=TRUE)}`

Value

Path to the outputted transcriptome assembly

Examples

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

---

blaze *BLAZE Assign reads to cell barcodes.*

Description

Uses BLAZE to generate barcode list and assign reads to cell barcodes.

Usage

```
blaze(expect_cells, fq_in, ...)
```

Arguments

expect_cells	Integer, expected number of cells. Note: this could be just a rough estimate. E.g., the targeted number of cells.
fq_in	File path to the fastq file used as a query sequence file
...	Additional BLAZE configuration parameters. E.g., setting ‘’output-prefix’=’some_prefix’‘ is equivalent to specifying ‘–output-prefix some_prefix‘ in BLAZE; Similarly, ‘overwrite=TRUE‘ is equivalent to switch on the ‘–overwrite‘ option. Note that the specified parameters will override the parameters specified in the configuration file. All available options can be found at https://github.com/shimlab/BLAZE.

Value

A `data.frame` summarising the reads aligned. Other outputs are written to disk. The details of the output files can be found at https://github.com/shimlab/BLAZE.

Examples

```
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
fastq1_url <- 'https://raw.githubusercontent.com/shimlab/BLAZE/main/test/data/FAR20033_pass_51e510db_100.fastq'
fastq1 <- bfc[[names(BiocFileCache::bfcadd(bfc, 'Fastq1', fastq1_url))]]
outdir <- tempfile()
dir.create(outdir)
## Not run:
blaze(expect_cells=10, fastq1, overwrite=TRUE)

## End(Not run)
```

---

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 = NULL,
  k8 = 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`	Path to minimap2, if it is not in PATH. Only required if either or both of `do_genome_align` and `do_read_realign` are `TRUE`.

k8	Path to the k8 Javascript shell binary. Only required if do_genome_align is 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)
se <- bulk_long_pipeline(
  annotation = annotation, fastq = fastq_dir, outdir = outdir, genome_fa = genome_fa,
  config_file = create_config(outdir, type = "sc_3end", threads = 1, no_flank = TRUE)
)
```

`combine_sce` *Combine SCE*

Description

Combine FLT-seq SingleCellExperiment objects

Usage

```
combine_sce(sce_with_lr, sce_without_lr)
```

Arguments

`sce_with_lr`	A `SingleCellExperiment` object with both long and short reads. The long-read transcript counts should be stored in the 'transcript' altExp slot.
`sce_without_lr`	A `SingleCellExperiment` object with only short reads.

Details

For protcols like FLT-seq that generate two libraries, one with both short and long reads, and one with only short reads, this function combines the two libraries into a single `SingleCellExperiment` object. For the library with both long and short reads, the long-read transcript counts should be stored in the 'transcript' altExp slot of the `SingleCellExperiment` object. This function will combine the short-read gene counts of both libraries, and for the transcripts counts, it will leave `NA` values for the cells from the short-read only library. The `sc_impute_transcript` function can then be used to impute the `NA` values.

Value

A `SingleCellExperiment` object with combined gene counts and a "transcript" altExp slot.

Examples

```
with_lr <- SingleCellExperiment::SingleCellExperiment(assays = list(counts = matrix(rpois(100, 5), ncol = 10)))
without_lr <- SingleCellExperiment::SingleCellExperiment(assays = list(counts = matrix(rpois(200, 5), ncol = 20)))
long_read <- SingleCellExperiment::SingleCellExperiment(assays = list(counts = matrix(rpois(50, 5), ncol = 10)))
SingleCellExperiment::altExp(with_lr, "transcript") <- long_read
SummarizedExperiment::colData(with_lr)$Barcode <- paste0(1:10, "-1")
SummarizedExperiment::colData(without_lr)$Barcode <- paste0(8:27, "-1")
rownames(with_lr) <- as.character(101:110)
rownames(without_lr) <- as.character(103:112)
rownames(long_read) <- as.character(1001:1005)
combined_sce <- FLAMES::combine_sce(sce_with_lr = with_lr, sce_without_lr = without_lr)
combined_sce
```

---

convolution_filter *Convolution filter for smoothing transcript coverages*

Description

Filter out transcripts with sharp drops / rises in coverage, to be used in `filter_coverage` to remove transcripts with potential misalignments / internal priming etc. Filtering is done by convolving the coverage with a kernal of 1s and -1s (e.g. `c(1, 1, -1, -1)`, where the width of the 1s and -1s are determined by the `width` parameter), and check if the maximum absolute value of the convolution is below a threshold. If the convolution is below the threshold, `TRUE` is returned, otherwise `FALSE`.

Usage

```
convolution_filter(x, threshold = 0.15, width = 2, trim = 0.05)
```

Arguments

`x`	numeric vector of coverage values
`threshold`	numeric, the threshold for the maximum absolute value of the convolution

`width`	numeric, the width of the 1s and -1s in the kernal. E.g. `width = 2` will result in a kernal of `c(1, 1, -1, -1)`
`trim`	numeric, the proportion of the coverage values to ignore at both ends before convolution.

Value

logical, `TRUE` if the transcript passes the filter, `FALSE` otherwise

Examples

```
# A >30% drop in coverage will fail the filter with threshold = 0.3
convolution_filter(c(1, 1, 1, 0.69, 0.69, 0.69), threshold = 0.3)
convolution_filter(c(1, 1, 1, 0.71, 0.7, 0.7), threshold = 0.3)
```

`create_config` *Create Configuration File From Arguments*

Description

Create Configuration File From Arguments

Usage

```
create_config(outdir, type = "sc_3end", ...)
```

Arguments

`outdir`	the destination directory for the configuratio nfile
`type`	use an example config, available values: **"sc_3end"** - config for 10x 3' end ONT reads **"SIRV"** - config for the SIRV example reads
`...`	Configuration parameters. **seed** - Integer. Seed for minimap2. **threads** - Number of threads to use. **do_barcode_demultiplex** - Boolean. Specifies whether to run the barcode demultiplexing step. **do_genome_alignment** - Boolean. Specifies whether to run the genome alignment step. `TRUE` is recommended **do_gene_quantification** - Boolean. Specifies whether to run gene quantification using the genome alignment results. `TRUE` is recommended **do_isoform_identification** - Boolean. Specifies whether to run the isoform identification step. `TRUE` is recommended **bambu_isoform_identification** - Boolean. Whether to use Bambu for isoform identification.

multithread_isoform_identification - Boolean. Whether to use FLAMES' new multithreaded Cpp implementation for isoform identification.

do_read_realignment - Boolean. Specifies whether to run the read realignment step. `TRUE` is recommended

do_transcript_quantification - Boolean. Specifies whether to run the transcript quantification step. `TRUE` is recommended

barcode_parameters - List. Parameters for barcode demultiplexing passed to `find_barcode` (except `fastq`, `barcodes_file`, `stats_out`, `reads_out`) and `threads`, which are set by the pipeline, see `?find_barcode` for more details.

generate_raw_isoform - Boolean. Whether to generate all isoforms for debugging purpose.

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_3end.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, quantification = "FLAMES")
```

Arguments

`outdir`	The folder containing `FLAMES` output files
`annotation`	the annotation file that was used to produce the output files
`quantification`	(Optional) the quantification method used to generate the output files (either "FLAMES" or "Oarfish".). If not specified, the function will attempt to determine the quantification method.

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, remove = FALSE
)
R.utils::gunzip(
  filename = system.file("extdata", "rps24.fa.gz", package = "FLAMES"),
  destname = genome_fa, remove = FALSE
)
annotation <- system.file("extdata", "rps24.gtf.gz", package = "FLAMES")

sce <- FLAMES::sc_long_pipeline(
  genome_fa = genome_fa,
  fastq = system.file("extdata", "fastq", "musc_rps24.fastq.gz", package = "FLAMES"),
  annotation = annotation,
  outdir = outdir,
  barcodes_file = bc_allow,
  config_file = create_config(outdir, oarfish_quantification = FALSE)
)
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)
se <- bulk_long_pipeline(
  annotation = annotation, fastq = fastq_dir, outdir = outdir, genome_fa = genome_fa,
  config_file = create_config(outdir, type = "sc_3end", threads = 1, no_flank = TRUE)
)
```

---

create_spe *Create a SpatialExperiment object*

Description

This function creates a SpatialExperiment object from a SingleCellExperiment object and a spatial barcode file.

Usage

```
create_spe(
  sce,
  spatial_barcode_file,
  mannual_align_json,
  image,
  tissue_positions_file
)
```

Arguments

sce	The SingleCellExperiment object obtained from running the sc_long_pipeline function.
spatial_barcode_file	The path to the spatial barcode file, e.g. "spaceranger-2.1.1/lib/python/cellranger/barcodes/vis

mannual_align_json
: The path to the mannual alignment json file.

image
: 'DataFrame' containing the image data. See ?SpatialExperiment::readImgData and ?SpatialExperiment::SpatialExperiment.

tissue_positions_file
: The path to Visium positions file, e.g. "spaceranger-2.1.1/lib/python/cellranger/barcodes/visi

Value

A SpatialExperiment object.

---

cutadapt *cutadapt wrapper*

Description

trim TSO adaptor with cutadapt

Usage

```
cutadapt(args)
```

Arguments

args	arguments to be passed to cutadapt

Value

Exit code of cutadapt

Examples

```
cutadapt("-h")
```

---

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

---

fake_stranded_gff *Fake stranded GFF file*

Description

Check if all the transcript in the annotation is stranded. If not, convert to '+'.

Usage

```
fake_stranded_gff(gff_file)
```

Value

Path to the temporary file with unstranded transcripts converted to '+'.

---

filter_annotation *filter annotation for plotting coverages*

Description

Removes isoform annotations that could produce ambigious reads, such as isoforms that only differ by the 5' / 3' end. This could be useful for plotting average coverage plots.

Usage

```
filter_annotation(annotation, keep = "tss_differ")
```

Arguments

annotation	path to the GTF annotation file, or the parsed GenomicRanges object.
keep	string, one of 'tss_differ' (only keep isoforms that all differ by the transcription start site position), 'tes_differ' (only keep those that differ by the transcription end site position), 'both' (only keep those that differ by both the start and end site), or 'single_transcripts' (only keep genes that contains a sinlge transcript).

Value

GenomicRanges of the filtered isoforms

Examples

```
filtered_annotation <- filter_annotation(
  system.file("extdata", "rps24.gtf.gz", package = 'FLAMES'), keep = 'tes_differ')
filtered_annotation
```

`filter_coverage` *Filter transcript coverage*

Description

Filter the transcript coverage by applying a filter function to the coverage values.

Usage

```
filter_coverage(x, filter_fn = convolution_filter)
```

Arguments

`x`	The tibble returned by `get_coverage`, or a BAM file path, or a GAlignments object.
`filter_fn`	The filter function to apply to the coverage values. The function should take a numeric vector of coverage values and return a logical value (TRUE if the transcript passes the filter, FALSE otherwise). The default filter function is `convolution_filter`, which filters out transcripts with sharp drops / rises in coverage.

Value

a tibble of the transcript information and coverages, with transcipts that pass the filter

Examples

```
# Create a BAM file with minimap2_realign
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)
minimap2_realign(
```

```
  config = jsonlite::fromJSON(
    system.file("extdata", "config_sclr_nanopore_3end.json", package = "FLAMES")),
  fq_in = fastq1,
  outdir = outdir
)
x <- get_coverage(file.path(outdir, 'realign2transcript.bam'))
nrow(x)
filter_coverage(x) |>
  nrow()
```

`find_barcode` *Match Cell Barcodes*

Description

demultiplex reads with flexiplex

Usage

```
find_barcode(
  fastq,
  barcodes_file,
  max_bc_editdistance = 2,
  max_flank_editdistance = 8,
  reads_out,
  stats_out,
  threads = 1,
 pattern = c(primer = "CTACACGACGCTCTTCCGATCT", BC = paste0(rep("N", 16), collapse =
  ""), UMI = paste0(rep("N", 12), collapse = ""), polyT = paste0(rep("T", 9), collapse
    = "")),
  TSO_seq = "",
  TSO_prime = 3,
  strand = "+",
  cutadapt_minimum_length = 1,
  full_length_only = FALSE
)
```

Arguments

`fastq`	character vector of paths to FASTQ files or folders, if named, the names will be used as sample names, otherwise the file names will be used
`barcodes_file`	path to file containing barcode allow-list, with one barcode in each line
`max_bc_editdistance`	max edit distances for the barcode sequence
`max_flank_editdistance`	max edit distances for the flanking sequences (primer and polyT)

reads_out — path to output FASTQ file; if multiple samples are processed, the sample name will be appended to this argument, e.g. provide path/out.fq for single sample, and path/prefix for multiple samples.

stats_out — path of output stats file; similar to reads_out, e.g. provide path/stats.tsv for single sample, and path/prefix for multiple samples.

threads — number of threads to be used

pattern — named character vector defining the barcode pattern

TSO_seq — TSO sequence to be trimmed

TSO_prime — either 3 (when TSO_seq is on 3' the end) or 5 (on 5' end)

strand — strand of the barcode pattern, either '+' or '-' (read will be reverse complemented after barcode matching if '-')

cutadapt_minimum_length — minimum read length after TSO trimming (cutadapt's –minimum-length)

full_length_only — boolean, when TSO sequence is provided, whether reads without TSO are to be discarded

Details

This function demultiplexes reads by searching for flanking sequences (adaptors) around the barcode sequence, and then matching against allowed barcodes. For single sample, either provide a single FASTQ file or a folder containing FASTQ files. For multiple samples, provide a vector of paths (either to FASTQ files or folders containing FASTQ files). Gzipped file input are supported but the output will be uncompressed.

Value

a list containing: reads_tb (tibble of read demultiplexed information) and input, output, read1_with_adapter from cutadapt report (if TSO trimming is performed)

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, remove = FALSE
)
# single sample
find_barcode(
  fastq = system.file("extdata", "fastq", "musc_rps24.fastq.gz", package = "FLAMES"),
  stats_out = file.path(outdir, "bc_stat"),
  reads_out = file.path(outdir, "demultiplexed.fq"),
  barcodes_file = bc_allow,
  TSO_seq = "AAGCAGTGGTATCAACGCAGAGTACATGGG", TSO_prime = 5,
  strand = '-', cutadapt_minimum_length = 10, full_length_only = TRUE
)
```

```
# multi-sample
fastq_dir <- tempfile()
dir.create(fastq_dir)
file.copy(system.file("extdata", "fastq", "musc_rps24.fastq.gz", package = "FLAMES"),
  file.path(fastq_dir, "musc_rps24.fastq.gz"))
sampled_lines <- readLines(file.path(fastq_dir, "musc_rps24.fastq.gz"), n = 400)
writeLines(sampled_lines, file.path(fastq_dir, "copy.fastq"))
result <- find_barcode(
  # you can mix folders and files. each path will be considered as a sample
 fastq = c(fastq_dir, system.file("extdata", "fastq", "musc_rps24.fastq.gz", package = "FLAMES")),
  stats_out = file.path(outdir, "bc_stat"),
  reads_out = file.path(outdir, "demultiplexed.fq"),
  barcodes_file = bc_allow, TSO_seq = "CCCATGTACTCTGCGTTGATACCACTGCTT"
)
```

`find_bin` *Find path to a binary Wrapper for Sys.which to find path to a binary*

Description

This function is a wrapper for `base::Sys.which` to find the path to a command. It also searches within the `FLAMES` basilisk conda environment. This function also replaces "" with `NA` in the output of `base::Sys.which` to make it easier to check if the binary is found.

Usage

```
find_bin(command)
```

Arguments

`command`	character, the command to search for

Value

character, the path to the command or `NA`

Examples

```
find_bin("minimap2")
```

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)
config <- jsonlite::fromJSON(
  system.file("extdata", "config_sclr_nanopore_3end.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
)
```

`find_variants` *bulk variant identification*

Description

Treat each bam file as a bulk sample and identify variants against the reference

Usage

```
find_variants(
  bam_path,
  reference,
  annotation,
  min_nucleotide_depth = 100,
  homopolymer_window = 3,
  annotated_region_only = FALSE,
  names_from = "gene_name",
  threads = 1
)
```

Arguments

`bam_path`	character(1) or character(n): path to the bam file(s) aligned to the reference genome (NOT the transcriptome!).
`reference`	DNAStringSet: the reference genome
`annotation`	GRanges: the annotation of the reference genome. You can load a GTF/GFF annotation file with `anno <- rtracklayer::import(file)`.
`min_nucleotide_depth`	integer(1): minimum read depth for a position to be considered a variant.
`homopolymer_window`	integer(1): the window size to calculate the homopolymer percentage. The homopolymer percentage is calculated as the percentage of the most frequent nucleotide in a window of `-homopolymer_window` to `homopolymer_window` nucleotides around the variant position, excluding the variant position itself. Calculation of the homopolymer percentage is skipped when `homopolymer_window = 0`. This is useful for filtering out Nanopore sequencing errors in homopolymer regions.
`annotated_region_only`	logical(1): whether to only consider variants outside annotated regions. If `TRUE`, only variants outside annotated regions will be returned. If `FALSE`, all variants will be returned, which could take significantly longer time.
`names_from`	character(1): the column name in the metadata column of the annotation (`mcols(annotation)[, names_from]`) to use for the `region` column in the output.

threads integer(1): number of threads to use. Threading is done over each annotated region and (if annotated_region_only = FALSE) unannotated gaps for each bam file.

Details

Each bam file is treated as a bulk sample to perform pileup and identify variants. You can run sc_mutations with the variants identified with this function to get single-cell allele counts. Note that reference genome FASTA files may have the chromosome names field as '>chr1 1' instead of '>chr1'. You may need to remove the trailing number to match the chromosome names in the bam file, for example with names(ref) <- sapply(names(ref), function(x) strsplit(x, " ")[[1]][1]).

Value

A tibble with columns: seqnames, pos, nucleotide, count, sum, freq, ref, region, homopolymer_pct, bam_path The homopolymer percentage is calculated as the percentage of the most frequent nucleotide in a window of homopolymer_window nucleotides around the variant position, excluding the variant position itself.

Examples

```
outdir <- tempfile()
dir.create(outdir)
genome_fa <- system.file("extdata", "rps24.fa.gz", package = "FLAMES")
minimap2_align( # align to genome
  config = jsonlite::fromJSON(
    system.file("extdata", "config_sclr_nanopore_3end.json", package = "FLAMES")),
  fa_file = genome_fa,
  fq_in = system.file("extdata", "fastq", "demultiplexed.fq.gz", package = "FLAMES"),
  annot = system.file("extdata", "rps24.gtf.gz", package = "FLAMES"),
  outdir = outdir
)
variants <- find_variants(
  bam_path = file.path(outdir, "align2genome.bam"),
  reference = genome_fa,
  annotation = system.file("extdata", "rps24.gtf.gz", package = "FLAMES"),
  min_nucleotide_depth = 4
)
head(variants)
```

FLAMES *FLAMES: full-length analysis of mutations and splicing*

Description

FLAMES: full-length analysis of mutations and splicing

`flexiplex` *Rcpp port of flexiplex*

Description

demultiplex reads with flexiplex, for detailed description, see documentation for the original flexiplex: https://davidsongroup.github.io/flexiplex

Usage

```
flexiplex(
  reads_in,
  barcodes_file,
  bc_as_readid,
  max_bc_editdistance,
  max_flank_editdistance,
  pattern,
  reads_out,
  stats_out,
  bc_out,
  reverseCompliment,
  n_threads
)
```

Arguments

`reads_in`	Input FASTQ or FASTA file
`barcodes_file`	barcode allow-list file
`bc_as_readid`	bool, whether to add the demultiplexed barcode to the read ID field
`max_bc_editdistance`	max edit distance for barcode '
`max_flank_editdistance`	max edit distance for the flanking sequences '
`pattern`	StringVector defining the barcode structure, see [find_barcode]
`reads_out`	output file for demultiplexed reads
`stats_out`	output file for demultiplexed stats
`bc_out`	WIP
`reverseCompliment`	bool, whether to reverse complement the reads after demultiplexing
`n_threads`	number of threads to be used during demultiplexing

Value

integer return value. 0 represents normal return.

get_coverage *Get read coverages from BAM file*

Description

Get the read coverages for each transcript in the BAM file (or a GAlignments object). The read coverages are sampled at 100 positions along the transcript, and the coverage is scaled by dividing the coverage at each position by the total read counts for the transcript. If a BAM file is provided, alignment with MAPQ < 5, secondary alignments and supplementary alignments are filtered out. A `GAlignments` object can also be provided in case alternative filtering is desired.

Usage

```
get_coverage(bam, min_counts = 10, remove_UTR = FALSE, annotation)
```

Arguments

`bam`	path to the BAM file, or a parsed GAlignments object
`min_counts`	numeric, the minimum number of alignments required for a transcript to be included
`remove_UTR`	logical, if `TRUE`, remove the UTRs from the coverage
`annotation`	(Required if `remove_UTR = TRUE`) path to the GTF annotation file

Value

a tibble of the transcript information and coverages, with the following columns:

- `transcript`: the transcript name / ID
- `read_counts`: the total number of aligments for the transcript
- `coverage_1-100`: the coverage at each of the 100 positions along the transcript
- `tr_length`: the length of the transcript

Examples

```
# Create a BAM file with minimap2_realign
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)
minimap2_realign(
  config = jsonlite::fromJSON(
    system.file("extdata", "config_sclr_nanopore_3end.json", package = "FLAMES")),
```

```
  fq_in = fastq1,
  outdir = outdir
)
x <- get_coverage(file.path(outdir, 'realign2transcript.bam'))
head(x)
```

`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)
config <- jsonlite::fromJSON(
  system.file("extdata", "config_sclr_nanopore_3end.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"))
```

`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 = NA,
  k8 = NA,
  samtools = NA,
  prefix = NULL,
  threads = 1
)
```

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`	Path to minimap2 binary
`k8`	Path to the k8 Javascript shell binary
`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)
minimap2_align(
  config = jsonlite::fromJSON(
    system.file("extdata", "config_sclr_nanopore_3end.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,
  samtools = NULL,
  prefix = NULL,
  minimap2_args,
  sort_by,
  threads = 1
)
```

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`	Path to minimap2 binary

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
minimap2_args	vector of command line arguments to pass to minimap2
sort_by	String, If provided, sort the BAM file by this tag instead of by position.
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

```
outdir <- tempfile()
dir.create(outdir)
annotation <- system.file('extdata', 'rps24.gtf.gz', package = 'FLAMES')
genome_fa <- system.file('extdata', 'rps24.fa.gz', package = 'FLAMES')
fasta <- annotation_to_fasta(annotation, genome_fa, outdir)
fastq <- system.file('extdata', 'fastq', 'demultiplexed.fq.gz', package = 'FLAMES')
minimap2_realign(
  config = jsonlite::fromJSON(
    system.file("extdata", "config_sclr_nanopore_3end.json", package = 'FLAMES')
  ),
  fq_in = fastq,
  outdir = outdir
)
```

---

mutation_positions *Calculate mutation positions within the gene body*

Description

Given a set of mutations and gene annotation, calculate the position of each mutation within the gene body they are in.

Usage

```
mutation_positions(
  mutations,
  annotation,
  type = "relative",
  bin = FALSE,
```

```
  by = c(region = "gene_name"),
  threads = 1
)
```

Arguments

mutations	either the tibble output from find_variants. It must have columns seqnames, pos, and a third column for specifying the gene id or gene name. The mutation must be within the gene region.
annotation	Either path to the annotation file (GTF/GFF) or a GRanges object of the gene annotation.
type	character(1): the type of position to calculate. Can be one of "TSS" (distance from the transcription start site), "TES" (distance from the transcription end site), or "relative" (relative position within the gene body).
bin	logical(1): whether to bin the relative positions into 100 bins. Only applicable when type = "relative".
by	character(1): the column name in the annotation to match with the gene annotation. E.g. c("region" = "gene_name") to match the 'region' column in the mutations with the 'gene_name' column in the annotation.
threads	integer(1): number of threads to use.

Value

A numeric vector of positions of each mutation within the gene body. When type = "relative", the positions are normalized to the gene length, ranging from 0 (start of the gene) to 1 (end of the gene). When type = "TSS" / type = "TES", the distances from the transcription start / end site. If bin = TRUE, and type = "relative", the relative positions are binned into 100 bins along the gene body, and the output is a matrix with the number of mutations in each bin, the rows are named by the by column (e.g. gene name).

Examples

```
variants <- data.frame(
  seqnames = rep("chr14", 8),
  pos = c(1084, 1085, 1217, 1384, 2724, 2789, 5083, 5147),
  region = rep("Rps24", 8)
)
positions <-
 mutation_positions(
   mutations = variants,
   annotation = system.file("extdata", "rps24.gtf.gz", package = "FLAMES")
 )
```

mutation_positions_single *mutation positions within the gene body*

Description

Given a set of mutations and a gene annotation, calculate the position of each mutation within the gene body. The gene annotation must have the following types: "gene" and "exon". The gene annotation must be for one gene only. The mutations must be within the gene region. The function will merge overlapping exons and calculate the position of each mutation within the gene body, excluding intronic regions.

Usage

```
mutation_positions_single(mutations, annotation_grange, type, verbose = TRUE)
```

Arguments

mutations	either the tibble output from find_variants or a GRanges object. Make sure to filter it for only the gene of interest.
annotation_grange	GRanges: the gene annotation. Must have the following types: "gene" and "exon".
type	character(1): the type of position to calculate. Can be one of "TSS" (distance from the transcription start site), "TES" (distance from the transcription end site), or "relative" (relative position within the gene body).
verbose	logical(1): whether to print messages.

Value

A numeric vector of positions of each mutation within the gene body. When type = "relative", the positions are normalized to the gene length, ranging from 0 (start of the gene) to 1 (end of the gene). When type = "TSS" / type = "TES", the distances from the transcription start / end site.

plot_coverage *plot read coverages*

Description

Plot the average read coverages for each length bin or a perticular isoform

Usage

```
plot_coverage(
  x,
  quantiles = c(0, 0.2375, 0.475, 0.7125, 0.95, 1),
  length_bins = c(0, 1, 2, 5, 10, Inf),
  weight_fn = weight_transcripts,
  filter_fn,
  detailed = FALSE
)
```

Arguments

x	path to the BAM file (aligning reads to the transcriptome), or the (GenomicAlignments::readGAlignments) parsed GAlignments object, or the tibble returned by get_coverage, or the filtered tibble returned by filter_coverage.
quantiles	numeric vector to specify the quantiles to bin the transcripts lengths by if length_bins is missing. The length bins will be determined such that the read counts are distributed acording to the quantiles.
length_bins	numeric vector to specify the sizes to bin the transcripts by
weight_fn	function to calculate the weights for the transcripts. The function should take a numeric vector of read counts and return a numeric vector of weights. The default function is weight_transcripts, you can change its default parameters by passing an anonymous function like function(x) weight_transcripts(x, type = 'equal').
filter_fn	Optional filter function to filter the transcripts before plotting. See the filter_fn parameter in filter_coverage for more details. Providing a filter fucntion here is the same as providing it in filter_coverage and then passing the result to this function.
detailed	logical, if TRUE, also plot the top 10 transcripts with the highest read counts for each length bin.

Value

a ggplot2 object of the coverage plot(s)

Examples

```
# Create a BAM file with minimap2_realign
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)
minimap2_realign(
```

```
  config = jsonlite::fromJSON(
    system.file("extdata", "config_sclr_nanopore_3end.json", package = "FLAMES")),
  fq_in = fastq1,
  outdir = outdir
)
# Plot the coverages directly from the BAM file
plot_coverage(file.path(outdir, 'realign2transcript.bam'))

# Get the coverage information first
coverage <- get_coverage(file.path(outdir, 'realign2transcript.bam')) |>
  dplyr::filter(read_counts > 2) |> # Filter out transcripts with read counts < 3
 filter_coverage(filter_fn = convolution_filter) # Filter out transcripts with sharp drops / rises
# Plot the filtered coverages
plot_coverage(coverage, detailed = TRUE)
# filtering function can also be passed directly to plot_coverage
plot_coverage(file.path(outdir, 'realign2transcript.bam'), filter_fn = convolution_filter)
```

`plot_demultiplex` *Plot Cell Barcode demultiplex statistics*

Description

produce a barplot of cell barcode demultiplex statistics

Usage

```
plot_demultiplex(find_barcode_result)
```

Arguments

`find_barcode_result`	output from `find_barcode`

Value

a list of ggplot objects:

- reads_count_plot: stacked barplot of: demultiplexed reads
- knee_plot: knee plot of UMI counts before TSO trimming
- flank_editdistance_plot: flanking sequence (adaptor) edit-distance plot
- barcode_editdistance_plot: barcode edit-distance plot
- cutadapt_plot: if TSO trimming is performed, number of reads kept by cutadapt

Examples

```
outdir <- tempfile()
dir.create(outdir)
fastq_dir <- tempfile()
dir.create(fastq_dir)
file.copy(system.file("extdata", "fastq", "musc_rps24.fastq.gz", package = "FLAMES"),
  file.path(fastq_dir, "musc_rps24.fastq.gz"))
sampled_lines <- readLines(file.path(fastq_dir, "musc_rps24.fastq.gz"), n = 400)
writeLines(sampled_lines, file.path(fastq_dir, "copy.fastq"))
bc_allow <- file.path(outdir, "bc_allow.tsv")
R.utils::gunzip(
  filename = system.file("extdata", "bc_allow.tsv.gz", package = "FLAMES"),
  destname = bc_allow, remove = FALSE
)
find_barcode(
  fastq = fastq_dir,
  stats_out = file.path(outdir, "bc_stat"),
  reads_out = file.path(outdir, "demultiplexed.fq"),
  barcodes_file = bc_allow, TSO_seq = "CCCATGTACTCTGCGTTGATACCACTGCTT"
) |>
  plot_demultiplex()
```

---

plot_isoforms *Plot isoforms*

Description

Plot isoforms, either from a gene or a list of transcript ids.

Usage

```
plot_isoforms(
  sce,
  gene_id,
  transcript_ids,
  n = 4,
  format = "plot_grid",
  colors
)
```

Arguments

sce	The SingleCellExperiment object containing transcript counts, rowRanges and rowData with gene_id and transcript_id columns.
gene_id	The gene symbol of interest, ignored if transcript_ids is provided.
transcript_ids	The transcript ids to plot.
n	The number of top isoforms to plot from the gene. Ignored if transcript_ids is provided.

format	The format of the output, either "plot_grid" or "list".
colors	A character vector of colors to use for the isoforms. If not provided, gray will be used. for all isoforms.

Details

This function takes a SingleCellExperiment object and plots the top isoforms of a gene, or a list of specified transcript ids. Either as a list of plots or together in a grid. This function wraps the ggbio::geom_alignment function to plot the isoforms, and orders the isoforms by expression levels (when specifying a gene) or by the order of the transcript_ids.

Value

When format = "list", a list of ggplot objects is returned. Otherwise, a grid of the plots is returned.

Examples

```
plot_isoforms(scmixology_lib10_transcripts, gene_id = "ENSG00000108107")
```

---

plot_isoform_heatmap *FLAMES heetmap plots*

Description

Plot expression heatmap of top n isoforms of a gene

Usage

```
plot_isoform_heatmap(
  sce,
  gene_id,
  transcript_ids,
  n = 4,
  isoform_legend_width = 7,
  col_low = "#313695",
  col_mid = "#FFFFBF",
  col_high = "#A50026",
  color_quantile = 1,
  cluster_palette,
  ...
)
```

Arguments

sce	The SingleCellExperiment object containing transcript counts, rowRanges and rowData with gene_id and transcript_id columns.
gene_id	The gene symbol of interest, ignored if transcript_ids is provided.
transcript_ids	The transcript ids to plot.
n	The number of top isoforms to plot from the gene. Ignored if transcript_ids is provided.
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.
cluster_palette	Optional, named vector of colors for the cluster annotations.
...	Additional arguments to pass to Heatmap.

Details

Takes SingleCellExperiment object and plots an expression heatmap with the isoform visualizations along genomic coordinates.

Value

a ComplexHeatmap

Examples

```
scmixology_lib10_transcripts |>
  scuttle::logNormCounts() |>
  plot_isoform_heatmap(gene = "ENSG00000108107")
```

---

plot_isoform_reduced_dim

FLAMES isoform reduced dimensions plots

Description

Plot expression of top n isoforms of a gene in reduced dimensions

Usage

```
plot_isoform_reduced_dim(
  sce,
  gene_id,
  transcript_ids,
  n = 4,
  reduced_dim_name = "UMAP",
  use_gene_dimred = FALSE,
  expr_func = function(x) {
     SingleCellExperiment::logcounts(x)
 },
  col_low = "#313695",
  col_mid = "#FFFFBF",
  col_high = "#A50026",
  color_quantile = 1,
  format = "plot_grid",
  ...
)
```

Arguments

sce	The SingleCellExperiment object containing transcript counts, rowRanges and rowData with gene_id and transcript_id columns.
gene_id	The gene symbol of interest, ignored if transcript_ids is provided.
transcript_ids	The transcript ids to plot.
n	The number of top isoforms to plot from the gene. Ignored if transcript_ids is provided.
reduced_dim_name	The name of the reduced dimension to use for plotting cells.
use_gene_dimred	Whether to use gene-level reduced dimensions for plotting. Set to TRUE if the SingleCellExperiment has gene counts in main assay and transcript counts in altExp.
expr_func	The function to extract expression values from the SingleCellExperiment object. Default is logcounts. Alternatively, counts can be used for raw counts.
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.
format	The format of the output, either "plot_grid" or "list".
...	Additional arguments to pass to plot_grid.

Details

Takes `SingleCellExperiment` object and plots an expression on reduced dimensions with the isoform visualizations along genomic coordinates.

Value

a `ggplot` object of the UMAP(s)

Examples

```
scmixology_lib10 <-
  scmixology_lib10[, colSums(SingleCellExperiment::counts(scmixology_lib10)) > 0]
sce_lr <- scmixology_lib10[, colnames(scmixology_lib10) %in% colnames(scmixology_lib10_transcripts)]
SingleCellExperiment::altExp(sce_lr, "transcript") <-
  scmixology_lib10_transcripts[, colnames(sce_lr)]
combined_sce <- combine_sce(sce_lr, scmixology_lib90)
combined_sce <- combined_sce |>
  scuttle::logNormCounts() |>
  scater::runPCA() |>
  scater::runUMAP()
combined_imputed_sce <- sc_impute_transcript(combined_sce)
plot_isoform_reduced_dim(combined_sce, 'ENSG00000108107')
plot_isoform_reduced_dim(combined_imputed_sce, 'ENSG00000108107')
```

---

plot_spatial_feature *Plot feature on spatial image*

Description

This function plots a spatial point plot for given feature

Usage

```
plot_spatial_feature(
  spe,
  feature,
  opacity = 50,
  grayscale = TRUE,
  size = 1,
  assay_type = "counts",
  color = "red",
  ...
)
```

Arguments

spe	The SpatialExperiment object.
feature	The feature to plot. Could be either a feature name or index present in the assay or a numeric vector of length nrow(spe).
opacity	The opacity of the background tissue image.
grayscale	Whether to convert the background image to grayscale.
size	The size of the points.
assay_type	The assay that contains the given features. E.g. 'counts', 'logcounts'.
color	The maximum color for the feature. Minimum color is transparent.
...	Additional arguments to pass to geom_point.

Value

A ggplot object.

---

plot_spatial_isoform *Plot spatial pie chart of isoforms*

Description

This function plots a spatial pie chart for given features.

Usage

```
plot_spatial_isoform(spe, isoforms, assay_type = "counts", color_palette, ...)
```

Arguments

spe	The SpatialExperiment object.
isoforms	The isoforms to plot.
assay_type	The assay that contains the given features. E.g. 'counts', 'logcounts'.
color_palette	Named vector of colors for each isoform.
...	Additional arguments to pass to plot_spatial_pie, including opacity, grayscale, pie_scale.

Value

A ggplot object.

plot_spatial_pie *Plot spatial pie chart*

Description

This function plots a spatial pie chart for given features.

Usage

```
plot_spatial_pie(
  spe,
  features,
  assay_type = "counts",
  color_palette,
  opacity = 50,
  grayscale = TRUE,
  pie_scale = 0.8
)
```

Arguments

spe	The SpatialExperiment object.
features	The features to plot.
assay_type	The assay that contains the given features.
color_palette	Named vector of colors for each feature.
opacity	The opacity of the background tissue image.
grayscale	Whether to convert the background image to grayscale.
pie_scale	The size of the pie charts.

Value

A ggplot object.

quantify_gene *Gene quantification*

Description

Calculate the per gene UMI count matrix by parsing the genome alignment file.

Usage

```
quantify_gene(
  annotation,
  outdir,
  infq,
  n_process,
  pipeline = "sc_single_sample",
  samples = NULL,
  random_seed = 2024
)
```

Arguments

annotation	The file path to the annotation file in GFF3 format
outdir	The path to directory to store all output files.
infq	The input FASTQ file.
n_process	The number of processes to use for parallelization.
pipeline	The pipeline type as a character string, either sc_single_sample (single-cell, single-sample),
samples	A vector of sample names, default to the file names of input fastq files, or folder names if fastqs is a vector of folders. bulk (bulk, single or multi-sample), or sc_multi_sample (single-cell, multiple samples)
random_seed	The random seed for reproducibility.

Details

After the genome alignment step (do_genome_align), the alignment file will be parsed to generate the per gene UMI count matrix. For each gene in the annotation file, the number of reads overlapping with the gene's genomic coordinates will be assigned to that gene. If a read overlaps multiple genes, it will be assigned to the gene with the highest number of overlapping nucleotides. If exon coordinates are included in the provided annotation, the decision will first consider the number of nucleotides aligned to the exons of each gene. In cases of a tie, the overlap with introns will be used as a tiebreaker. If there is still a tie after considering both exons and introns, a random gene will be selected from the tied candidates.

After the read-to-gene assignment, the per gene UMI count matrix will be generated. Specifically, for each gene, the reads with similar mapping coordinates of transcript termination sites (TTS, i.e. the end of the the read with a polyT or polyA) will be grouped together. UMIs of reads in the same group will be collapsed to generate the UMI counts for each gene.

Finally, a new fastq file with deduplicated reads by keeping the longest read in each UMI.

Value

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

quantify_transcript *Transcript quantification*

Description

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

Usage

```
quantify_transcript(
  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),
...	Supply sample names as character vector (e.g. samples = c("name1", "name2", ...)) for muti-sample or bulk pipeline. bulk (bulk, single or multi-sample), or sc_multi_sample (single-cell, multiple samples)

Value

A SingleCellExperiment object for single-cell pipeline, a list of SingleCellExperiment objects for multi-sample pipeline, or a SummarizedExperiment object for bulk pipeline.

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 (!any(is.na(find_bin(c("minimap2", "k8"))))) {
  minimap2_realign(
    config = config, outdir = outdir,
    fq_in = fastq1
  )
  quantify_transcript_flames(annotation, outdir, config, pipeline = "bulk")
}

## End(Not run)
```

`quantify_transcript_flames` *FLAMES Transcript quantification*

Description

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

Usage

```
quantify_transcript_flames(
  annotation,
  outdir,
  config,
  pipeline = "sc_single_sample",
  samples
)
```

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),
`samples`	A vector of sample names, required for `sc_multi_sample` pipeline. `bulk` (bulk, single or multi-sample), or `sc_multi_sample` (single-cell, multiple samples)

Value

A `SingleCellExperiment` object for single-cell pipeline, a list of `SingleCellExperiment` objects for multi-sample pipeline, or a `SummarizedExperiment` object for bulk pipeline.

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 (!any(is.na(find_bin(c("minimap2", "k8"))))) {
  minimap2_realign(
    config = config, outdir = outdir,
    fq_in = fastq1
  )
  quantify_transcript_flames(annotation, outdir, config, pipeline = "bulk")
}

## End(Not run)
```

---

scmixology_lib10 *scMixology short-read gene counts - sample 2*

Description

Short-read gene counts from long and short-read single cell RNA-seq profiling of human lung adenocarcinoma cell lines using 10X version 2 chemstry. See Tian, L. et al. Comprehensive characterization of single-cell full-length isoforms in human and mouse with long-read sequencing. Genome Biology 22, 310 (2021).

Usage

```
scmixology_lib10
```

Format

'scmixology_lib10' A `SingleCellExperiment` with 7,240 rows and 60 columns:

Source

<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE154869>

scmixology_lib10_transcripts *scMixology long-read transcript counts - sample 2*

Description

long-read transcript counts from long and short-read single cell RNA-seq profiling of human lung adenocarcinoma cell lines using 10X version 2 chemstry. See Tian, L. et al. Comprehensive characterization of single-cell full-length isoforms in human and mouse with long-read sequencing. Genome Biology 22, 310 (2021).

Usage

```
scmixology_lib10_transcripts
```

Format

'scmixology_lib10_transcripts' A `SingleCellExperiment` with 7,240 rows and 60 columns:

Source

<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE154869>

scmixology_lib90 *scMixology short-read gene counts - sample 1*

Description

Short-read single cell RNA-seq profiling of human lung adenocarcinoma cell lines using 10X version 2 chemstry. Single cells from five human lung adenocarcinoma cell lines (H2228, H1975, A549, H838 and HCC827) were mixed in equal proportions and processed using the Chromium 10X platform, then sequenced using Illumina HiSeq 2500. See Tian L, Dong X, Freytag S, Lê Cao KA et al. Benchmarking single cell RNA-sequencing analysis pipelines using mixture control experiments. Nat Methods 2019 Jun;16(6):479-487. PMID: 31133762

Usage

```
scmixology_lib90
```

Format

'scmixology_lib90' A `SingleCellExperiment`

Source

<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE126906>

`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.

Usage

```
sc_DTU_analysis(sce, min_count = 15, threads = 1)
```

Arguments

`sce`	The `SingleCellExperiment` object from `sc_long_pipeline`,
`min_count`	The minimum UMI count threshold for filtering isoforms.
`threads`	Number of threads to use for parallel processing.

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 `colLabels` of the SCE object. 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_id - 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.

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, remove = FALSE
)
R.utils::gunzip(
  filename = system.file("extdata", "rps24.fa.gz", package = "FLAMES"),
  destname = genome_fa, remove = FALSE
)

sce <- FLAMES::sc_long_pipeline(
  genome_fa = genome_fa,
  fastq = system.file("extdata", "fastq", "musc_rps24.fastq.gz", package = "FLAMES"),
  annotation = system.file("extdata", "rps24.gtf.gz", package = "FLAMES"),
  outdir = outdir,
  barcodes_file = bc_allow,
  config_file = create_config(outdir)
)
group_anno <- data.frame(barcode_seq = colnames(sce), groups = SingleCellExperiment::counts(sce)["ENSMUST0000016
SingleCellExperiment::colLabels(sce) <- group_anno$groups
sc_DTU_analysis(sce, min_count = 1)
```

---

sc_impute_transcript *Impute missing transcript counts*

Description

Impute missing transcript counts using a shared nearest neighbor graph

Usage

```
sc_impute_transcript(combined_sce, dimred = "PCA", ...)
```

Arguments

combined_sce	A SingleCellExperiment object with gene counts and a "transcript" altExp slot.
dimred	The name of the reduced dimension to use for building the shared nearest neighbor graph.
...	Additional arguments to pass to scran::buildSNNGraph. E.g. k = 30.

Details

For cells with NA values in the "transcript" altExp slot, this function imputes the missing values from cells with non-missing values. A shared nearest neighbor graph is built using reduced dimensions from the SingleCellExperiment object, and the imputation is done where the imputed value for a cell is the weighted sum of the transcript counts of its neighbors. Imputed values are stored in the "logcounts" assay of the "transcript" altExp slot. The "counts" assay is used to obtain logcounts but left unchanged.

Value

A SingleCellExperiment object with imputed logcounts assay in the "transcript" altExp slot.

Examples

```
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(counts = matrix(rpois(50, 5), ncol = 10)))
long_read <- SingleCellExperiment::SingleCellExperiment(assays = list(counts = matrix(rpois(40, 5), ncol = 10)))
SingleCellExperiment::altExp(sce, "transcript") <- long_read
SingleCellExperiment::counts(SingleCellExperiment::altExp(sce))[,1:2] <- NA
SingleCellExperiment::counts(SingleCellExperiment::altExp(sce))
imputed_sce <- sc_impute_transcript(sce, k = 4)
SingleCellExperiment::logcounts(SingleCellExperiment::altExp(imputed_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 = NULL,
  k8 = NULL,
  barcodes_file = NULL,
  expect_cell_numbers = NULL,
  config_file = NULL
)
```

Arguments

annotation	The file path to the annotation file in GFF3 format
fastqs	The input fastq files for multiple samples. Should be a named vector of file paths (eithr to FASTQ files or directories containing FASTQ files). The names of the vector will be used as the sample names.
outdir	The path to directory to store all output files.
genome_fa	The file path to genome fasta file.
minimap2	Path to minimap2, if it is not in PATH. Only required if either or both of do_genome_align and do_read_realign are TRUE.
k8	Path to the k8 Javascript shell binary. Only required if do_genome_align is TRUE.
barcodes_file	The file path to the reference csv used for demultiplexing in flexiplex. If not specified, the demultiplexing will be performed using BLAZE. Default is NULL.
expect_cell_numbers	A vector of roughly expected numbers of cells in each sample E.g., the targeted number of cells. Required if using BLAZE for demultiplexing, specifically, when the do_barcode_demultiplex are TRUE in the the JSON configuration file and barcodes_file is not specified. Default is NULL.
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

If "do_transcript_quantification" set to true, a list with two elements:

metadata A list of metadata from the pipeline run.

sces A list of SingleCellExperiment objects, one for each sample.

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, remove = FALSE
)
R.utils::gunzip(
  filename = system.file("extdata", "rps24.fa.gz", package = "FLAMES"),
  destname = genome_fa, remove = FALSE
)
ShortRead::writeFastq(reads[1:100],
  file.path(outdir, "fastq/sample1.fq.gz"), mode = "w", full = FALSE)
reads <- reads[-(1:100)]
ShortRead::writeFastq(reads[1:100],
  file.path(outdir, "fastq/sample2.fq.gz"), mode = "w", full = FALSE)
reads <- reads[-(1:100)]
ShortRead::writeFastq(reads,
  file.path(outdir, "fastq/sample3.fq.gz"), mode = "w", full = FALSE)

sce_list <- FLAMES::sc_long_multisample_pipeline(
  annotation = system.file("extdata", "rps24.gtf.gz", package = "FLAMES"),
  fastqs = c("sampleA" = file.path(outdir, "fastq"),
    "sample1" = file.path(outdir, "fastq", "sample1.fq.gz"),
    "sample2" = file.path(outdir, "fastq", "sample2.fq.gz"),
    "sample3" = file.path(outdir, "fastq", "sample3.fq.gz")),
```

```
  outdir = outdir,
  genome_fa = genome_fa,
  barcodes_file = rep(bc_allow, 4)
)
```

`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 = NULL,
  k8 = NULL,
  barcodes_file = NULL,
  expect_cell_number = NULL,
  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`	Path to minimap2, if it is not in PATH. Only required if either or both of `do_genome_align` and `do_read_realign` are `TRUE`.
`k8`	Path to the k8 Javascript shell binary. Only required if `do_genome_align` is `TRUE`.
`barcodes_file`	The file path to the reference csv used for demultiplexing in flexiplex. If not specified, the demultiplexing will be performed using BLAZE. Default is `NULL`.

`expect_cell_number`
: Expected number of cells for identifying the barcode list in BLAZE. This could be just a rough estimate. E.g., the targeted number of cells. Required if the `do_barcode_demultiplex` are `TRUE` in the the JSON configuration file and `barcodes_file` is not specified. Default is `NULL`.

`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, remove = FALSE
)
R.utils::gunzip(
  filename = system.file("extdata", "rps24.fa.gz", package = "FLAMES"),
  destname = genome_fa, remove = FALSE
)
if (!any(is.na(find_bin(c("minimap2", "k8"))))) {
  sce <- FLAMES::sc_long_pipeline(
    genome_fa = genome_fa,
    fastq = system.file("extdata", "fastq", "musc_rps24.fastq.gz", package = "FLAMES"),
    annotation = system.file("extdata", "rps24.gtf.gz", package = "FLAMES"),
    outdir = outdir,
    barcodes_file = bc_allow
  )
}
```

---

sc_mutations *Variant count for single-cell data*

Description

Count the number of reads supporting each variants at the given positions for each cell.

Usage

```
sc_mutations(bam_path, seqnames, positions, indel = FALSE, threads = 1)
```

Arguments

bam_path	character(1) or character(n): path to the bam file(s) aligned to the reference genome (NOT the transcriptome! Unless the postions are also from the transcriptome).
seqnames	character(n): chromosome names of the postions to count alleles.
positions	integer(n): positions, 1-based, same length as seqnames. The positions to count alleles.
indel	logical(1): whether to count indels (TRUE) or SNPs (FALSE).
threads	integer(1): number of threads to use. Maximum number of threads is the number of bam files * number of positions.

Value

A tibble with columns: allele, barcode, allele_count, cell_total_reads, pct, pos, seqname.

Examples

```
outdir <- tempfile()
dir.create(outdir)
genome_fa <- file.path(outdir, "rps24.fa")
R.utils::gunzip(
  filename = system.file("extdata", "rps24.fa.gz", package = "FLAMES"),
  destname = genome_fa, remove = FALSE
)
minimap2_align( # align to genome
  config = jsonlite::fromJSON(
    system.file("extdata", "config_sclr_nanopore_3end.json", package = "FLAMES")
  ),
  fa_file = genome_fa,
  fq_in = system.file("extdata", "fastq", "demultiplexed.fq.gz", package = "FLAMES"),
  annot = system.file("extdata", "rps24.gtf.gz", package = "FLAMES"),
  outdir = outdir
)
snps_tb <- sc_mutations(
  bam_path = file.path(outdir, "align2genome.bam"),
  seqnames = c("chr14", "chr14"),
  positions = c(1260, 2714), # positions of interest
  indel = FALSE
)
head(snps_tb)
snps_tb |>
  dplyr::filter(pos == 1260) |>
  dplyr::group_by(allele) |>
  dplyr::summarise(count = sum(allele_count)) # should be identical to samtools pileup
```

`weight_transcripts` *Weight transcripts by read counts*

Description

Given a vector of read counts, return a vector of weights. The weights could be either the read counts themselves (`type = 'counts'`), a binary vector of 0s and 1s where 1s are assigned to transcripts with read counts above a threshold (`type = 'equal'`, `min_counts = 1000`), or a sigmoid function of the read counts (`type = 'sigmoid'`). The sigmoid function is defined as `1 / (1 + exp(-steepness/inflection * (x - inflection)))`.

Usage

```
weight_transcripts(
  counts,
  type = "sigmoid",
  min_counts = 1000,
  inflection_idx = 10,
  inflection_max = 1000,
  steepness = 5
)
```

Arguments

`counts`	numeric vector of read counts
`type`	string, one of 'counts', 'sigmoid', or 'equal'
`min_counts`	numeric, the threshold for the 'equal' type
`inflection_idx`	numeric, the index of the read counts to determine the inflection point for the sigmoid function. The default is 10, i.e. the 10th highest read count will be the inflection point.
`inflection_max`	numeric, the maximum value for the inflection point. If the inflection point according to the inflection_idx is higher than this value, the inflection point will be set to this value instead.
`steepness`	numeric, the steepness of the sigmoid function

Value

numeric vector of weights

Examples

```
weight_transcripts(1:2000)
par(mfrow = c(2, 2))
plot(
  1:2000, weight_transcripts(1:2000, type = 'sigmoid'),
  type = 'l', xlab = 'Read counts', ylab = 'Sigmoid weight'
```

```
)
plot(
  1:2000, weight_transcripts(1:2000, type = 'counts'),
  type = 'l', xlab = 'Read counts', ylab = 'Weight by counts'
)
plot(
  1:2000, weight_transcripts(1:2000, type = 'equal'),
  type = 'l', xlab = 'Read counts', ylab = 'Equal weights'
)
```

Index