Package 'FLAMES'

September 19, 2024

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

Version 1.11.1

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, Biostrings, BiocGenerics, circlize, ComplexHeatmap, cowplot, dplyr, DropletUtils, GenomicRanges, GenomicFeatures, txdbmaker, GenomicAlignments, 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, future, methods, tibble, tidyselect, IRanges

Suggests BiocStyle, GEOquery, knitr, rmarkdown, markdown, 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://github.com/OliverVoogd/FLAMES

Config/testthat/edition 3

Depends R (>= 4.1.0)

LazyData true

StagedInstall no

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

git_branch devel

git_last_commit 2884c96

git_last_commit_date 2024-08-20

Repository Bioconductor 3.20

Date/Publication 2024-09-18

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

annotation_to_fasta . . . 3
blaze . . . 3
bulk_long_pipeline . . . 4
combine_sce . . . 6
create_config . . . 8
create_sce_from_dir . . . 9
create_se_from_dir . . . 10
cutadapt . . . 11
demultiplex_sockeye . . . 12
filter_annotation . . . 12
find_barcode . . . 13
find_isoform . . . 14
find_variants . . . 15
FLAMES . . . 17
flexiplex . . . 17
get_GRangesList . . . 18
minimap2_align . . . 19
minimap2_realign . . . 20
parse_gff_tree . . . 21
plot_coverage . . . 22
plot_demultiplex . . . 23
quantify_gene . . . 24
quantify_transcript . . . 25
scmixology_lib10 . . . 26
scmixology_lib10_transcripts . . . 26
scmixology_lib90 . . . 27
sc_DTU_analysis . . . 27
sc_heatmap_expression . . . 29
sc_long_multisample_pipeline . . . 30
sc_long_pipeline . . . 32
sc_mutations . . . 34

sc_reduce_dims 36
sc_umap_expression 37
sys_which 38

Index **39**

`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/rp
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.fa
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-
genome_fa <- bfc[[names(BiocFileCache::bfcadd(bfc, "genome.fa", paste(file_url, "SIRV_isoforms_multi-fasta_1
fastq_dir <- paste(temp_path, "fastq_dir", sep = "/") # the downloaded fastq files need to be in a directory to b
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 (!any(is.na(sys_which(c("minimap2", "k8"))))) {
  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)
}
```

`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_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)
```

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

if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
    sce <- FLAMES::sc_long_pipeline(
        genome_fa = genome_fa,
        fastq = system.file("extdata/fastq", package = "FLAMES"),
        annotation = annotation,
        outdir = outdir,
        barcodes_file = 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-
genome_fa <- bfc[[names(BiocFileCache::bfcadd(bfc, "genome.fa", paste(file_url, "SIRV_isoforms_multi-fasta_1
fastq_dir <- paste(temp_path, "fastq_dir", sep = "/") # the downloaded fastq files need to be in a directory to b
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 (!any(is.na(sys_which(c("minimap2", "k8"))))) {
  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)
}
```

`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

```
## Not run:
 cutadapt("-h")

## End(Not run)
```

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

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
filtered_annotation
```

`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,
  full_length_only = FALSE
)
```

Arguments

`fastq`	input FASTQ file path
`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 of output FASTQ file
`stats_out`	path of output stats file
`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)
`full_length_only`	boolean, when TSO sequence is provided, whether reads without TSO are to be discarded

Value

invisible()

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, re
find_barcode(
  fastq = system.file("extdata/fastq", package = "FLAMES"),
  stats_out = file.path(outdir, "bc_stat"),
  reads_out = file.path(outdir, "demultiplexed.fq.gz"),
  barcodes_file = bc_allow
)
```

`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_1
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, "annot.gtf", paste(file_url, "SIRV_isoforms_multi-fasta-
outdir <- tempfile()
dir.create(outdir)
if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
   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)
}
```

`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 <- file.path(outdir, "rps24.fa")
R.utils::gunzip(filename = system.file("extdata/rps24.fa.gz", package = "FLAMES"), destname = genome_fa, remov
download.file("https://raw.githubusercontent.com/mritchielab/FLAMES/devel/tests/testthat/demultiplexed.fq",
  destfile = file.path(outdir, "demultipelxed.fq")
) # can't be bothered to run demultiplexing again
if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
  minimap2_align( # align to genome
   config = jsonlite::fromJSON(system.file("extdata/SIRV_config_default.json", package = "FLAMES")),
    fa_file = genome_fa,
    fq_in = file.path(outdir, "demultipelxed.fq"),
    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 = GenomicRanges::GRanges("chr14", IRanges::IRanges(1, 1)),
    min_nucleotide_depth = 10
  )
  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,
  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
n_threads	number of threads to be used during demultiplexing

Value

integer return value. 0 represents normal return.

`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_1
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, "annot.gtf", paste(file_url, "SIRV_isoforms_multi-fasta-
outdir <- tempfile()
dir.create(outdir)
if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
   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"))
}
```

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_1
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, 'annot.gtf', paste(file_url, 'SIRV_isoforms_multi-fasta-
outdir <- tempfile()
dir.create(outdir)
if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
    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,
  samtools = NULL,
  prefix = NULL,
  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
`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_1
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, 'annot.gtf', paste(file_url, 'SIRV_isoforms_multi-fasta-
outdir <- tempfile()
dir.create(outdir)
if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
    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_
## Not run: parsed_gff <- parse_gff_tree(gff)
```

`plot_coverage` *plot read coverages*

Description

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

Usage

```
plot_coverage(
  bam,
  isoform = NULL,
  length_bins = c(0, 1, 2, 5, 10, Inf),
  weight_fn = "read_counts"
)
```

Arguments

`bam`	path to the BAM file (aligning reads to the transcriptome), or the (GenomicAlignments::readGAlignments) parsed GAlignments object
`isoform`	string vector, provide isoform names to plot the coverage for the corresponding isoforms, or provide NULL to plot average coverages for each length bin
`length_bins`	numeric vector to specify the sizes to bin the isoforms by
`weight_fn`	"read_counts" or "sigmoid", determins how the transcripts should be weighted within length bins.

Value

a ggplot2 object of the coverage plot(s)

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_1
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, 'annot.gtf', paste(file_url, 'SIRV_isoforms_multi-fasta-
outdir <- tempfile()
dir.create(outdir)
if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
    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
    )
  plot_coverage(bam = file.path(outdir, 'realign2transcript.bam'))
}
```

---

plot_demultiplex *Plot Cell Barcode demultiplex statistics*

Description

produce a barplot of cell barcode demultiplex statistics

Usage

```
plot_demultiplex(outdir, stats_file)
```

Arguments

outdir	folder containing the matched_barcode_stat file, or matched_barcode_stat.SAMPLE files. Ignored if stats_file is provided.
stats_file	matched_barcode_stat file(s) from which the statistics to be plotted.

Value

a ggplot object of the barcode plot

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, re
find_barcode(
  fastq = system.file("extdata/fastq", package = "FLAMES"),
  stats_out = file.path(outdir, "bc_stat"),
  reads_out = file.path(outdir, "demultiplexed.fq.gz"),
  barcodes_file = bc_allow
)
plot_demultiplex(stats_file = file.path(outdir, "bc_stat"))
```

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
)
```

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)

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 whose mapped ranges overlap with the gene's genome coordinates will be assigned to the gene. For reads can be assigned to multiple gene, the read will be assigned to the gene with the highest number of overlapping nucleotides. If the read can be assigned to multiple genes with the same number of overlapping nucleotides, the read will be not be assigned.

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), 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_1
annotation <- bfc[[names(BiocFileCache::bfcadd(bfc, "annot.gtf", paste(file_url, "SIRV_isoforms_multi-fasta-
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
file.copy(annotation, file.path(outdir, "isoform_annotated.gtf"))
## Not run:
if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
    minimap2_realign(
        config = config, outdir = outdir,
        fq_in = fastq1
    )
    quantify_transcript(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. 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, 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.
`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_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. 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, re
R.utils::gunzip(filename = system.file("extdata/rps24.fa.gz", package = "FLAMES"), destname = genome_fa, remov

if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
  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,
    barcodes_file = bc_allow
  )
 group_anno <- data.frame(barcode_seq = colnames(sce), groups = SingleCellExperiment::counts(sce)["ENSMUST00
  write.csv(group_anno, file.path(outdir, "cluster_annotation.csv"), row.names = FALSE)
  sc_DTU_analysis(sce, 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

```
combined_sce <- combine_sce(
    short_read_large = scmixology_lib90,
    short_read_small = scmixology_lib10,
    long_read_sce = scmixology_lib10_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,
  sample_names = NULL,
  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. It can be provided in different way: 1) a single path to the folder containing fastq files, each fastq file will be treated as a sample; or 2) a vector of paths to each fastq file, each fastq file will be treated as a sample; or 3) a vector of paths to folders containing fastq files, each folder will be treated as a sample.
`outdir`	The path to directory to store all output files.
`genome_fa`	The file path to genome fasta file.
`sample_names`	A vector of sample names, Default to the file names of input fastq files, or folder names if `fastqs` is a vector of folders.
`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

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, re
R.utils::gunzip(filename = system.file("extdata/rps24.fa.gz", package = "FLAMES"), destname = genome_fa, remov
ShortRead::writeFastq(sample(reads, size = 500, replace = TRUE), file.path(outdir, "fastq/sample1.fq.gz"), moc
ShortRead::writeFastq(sample(reads, size = 500, replace = TRUE), file.path(outdir, "fastq/sample2.fq.gz"), moc
ShortRead::writeFastq(sample(reads, size = 500, replace = TRUE), file.path(outdir, "fastq/sample3.fq.gz"), moc

if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
    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,
        barcodes_file = 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 = 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, re
R.utils::gunzip(filename = system.file("extdata/rps24.fa.gz", package = "FLAMES"), destname = genome_fa, remov

if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
    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,
        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,
  barcodes,
  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).
barcodes	character(n) when bam_path is a single file, or list of character(n) when bam_path is a list of files paths. The cell barcodes to count alleles for. Only reads with these barcodes will be counted.
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, remov
download.file("https://raw.githubusercontent.com/mritchielab/FLAMES/devel/tests/testthat/demultiplexed.fq",
  destfile = file.path(outdir, "demultipelxed.fq")
) # can't be bothered to run demultiplexing again
if (!any(is.na(sys_which(c("minimap2", "k8"))))) {
  minimap2_align( # align to genome
   config = jsonlite::fromJSON(system.file("extdata/SIRV_config_default.json", package = "FLAMES")),
    fa_file = genome_fa,
    fq_in = file.path(outdir, "demultipelxed.fq"),
    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,
   barcodes = read.delim(system.file("extdata/bc_allow.tsv.gz", package = "FLAMES"), header = FALSE)$V1
  )
```

```
  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
}
```

`sc_reduce_dims` *runPCA and runUMAP wrapper*

Description

runPCA and runUMAP wrapper for combined SCE object from `combine_sce`

Usage

```
sc_reduce_dims(multiAssay, n_pcs = 40, n_hvgs = 2000)
```

Arguments

`multiAssay`	The `MultiAssayExperiment` object from `combine_sce()`.
`n_pcs`	The number of principal components to generate.
`n_hvgs`	The number of variable genes to use for running PCA.

Details

This function takes the combined `MultiAssayExperiment` object from `combine_sce` and run `scater::runUMAP` and / or `scran::fixedPCA`

Value

the `MultiAssayExperiment` with reduced dimensions

Examples

```
combined_sce <- combine_sce(
    short_read_large = scmixology_lib90,
    short_read_small = scmixology_lib10,
    long_read_sce = scmixology_lib10_transcripts,
    remove_duplicates = FALSE)
sc_reduce_dims(multiAssay = combined_sce)
```

`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_s` 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

```
combined_sce <- combine_sce(
    short_read_large = scmixology_lib90,
    short_read_small = scmixology_lib10,
    long_read_sce = scmixology_lib10_transcripts,
    remove_duplicates = FALSE)

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

`sys_which` *Sys.which wrapper Wrapper for Sys.which that replaces "" with NA*

Description

The `base::Sys.which` function returns "" if the command is not found on some systems and `NA` on others. This wrapper replaces "" with `NA`

Usage

```
sys_which(command)
```

Arguments

`command`	character, the command to search for

Value

character, the path to the command or `NA`

Examples

```
sys_which("minimap2")
```

Index

∗ **datasets**
    scmixology_lib10, 26
    scmixology_lib10_transcripts, 26
    scmixology_lib90, 27

annotation_to_fasta, 3

blaze, 3
bulk_long_pipeline, 4
bulk_long_pipeline(), *31*, *34*

combine_sce, 6
create_config, 8
create_sce_from_dir, 9
create_se_from_dir, 10
cutadapt, 11

demultiplex_sockeye, 12

filter_annotation, 12
find_barcode, 13
find_isoform, 14
find_variants, 15
FLAMES, 17
flexiplex, 17

get_GRangesList, 18

minimap2_align, 19
minimap2_realign, 20

parse_gff_tree, 21
plot_coverage, 22
plot_demultiplex, 23

quantify_gene, 24
quantify_transcript, 25

sc_DTU_analysis, 27
sc_heatmap_expression, 29
sc_long_multisample_pipeline, 30
sc_long_pipeline, 32
sc_long_pipeline(), *6*
sc_mutations, 34
sc_reduce_dims, 36
sc_umap_expression, 37
scmixology_lib10, 26
scmixology_lib10_transcripts, 26
scmixology_lib90, 27
SingleCellExperiment(), *31*, *34*
SummarizedExperiment(), *6*
sys_which, 38