Package ‘FLAMES’

April 12, 2022

Type Package

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

Version 1.0.2

Date 2020-12-28

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

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

License GPL (>= 2)

Encoding UTF-8

Imports basilisk, reticulate, SingleCellExperiment, SummarizedExperiment, Rsamtools, utils, zlibbioc, scater, dplyr, tidyr, magrittr, S4Vectors, scuttle, stats

Suggests BiocStyle, knitr, rmarkdown, markdown, BiocFileCache

LinkingTo Rcpp, Rhtslib, zlibbioc

SystemRequirements GNU make, C++11

RoxygenNote 7.1.1

VignetteBuilder knitr

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

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

git_branch RELEASE_3_14

git_last_commit e96c59c

git_last_commit_date 2021-11-03

Date/Publication 2022-04-12

Author Tian Luyi [aut],
Voogd Oliver [aut, cre],
Schuster Jakob [aut],

Wang Changqing [aut],
Su Shian [aut],
Ritchie Matthew [ctb]

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

R topics documented:

`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(
  annot,
  fastq,
  in_bam = NULL,
  outdir,
  genome_fa,
  minimap2_dir = "",
  downsample_ratio = 1,
  config_file = NULL,
  do_genome_align = TRUE,
  do_isoform_id = TRUE,
  do_read_realign = TRUE,
  do_transcript_quanti = TRUE,
  gen_raw_isoform = TRUE,
  has_UMI = FALSE,
  MAX_DIST = 10,
  MAX_TS_DIST = 100,
  MAX_SPLICE_MATCH_DIST = 10,
  min_fl_exon_len = 40,
  Max_site_per_splice = 3,
  Min_sup_cnt = 10,
  Min_cnt_pct = 0.01,
  Min_sup_pct = 0.2,
  strand_specific = 1,
  remove_incomp_reads = 5,
  use_junctions = TRUE,
  no_flank = TRUE,
  use_annotation = TRUE,
  min_tr_coverage = 0.75,
  min_read_coverage = 0.75
)
```

Arguments

`annot`	The file path to gene annotations file in gff3 format
`fastq`	the path to the directory containing the fastq input files to merge into one, `merged.fastq.gz`. If `merged.fastq.gz` already exists, the fastq files are not merged and the existing merged file is used.
`in_bam`	optional BAM file path which replaces fastq directory argument. This skips the genome alignment and realignment steps
`outdir`	The path to directory to store all output files.
`genome_fa`	The file path to genome fasta file.
`minimap2_dir`	Path to the directory containing minimap2, if it is not in PATH. Only required if either or both of `do_genome_align` and `do_read_realign` are `TRUE`.
`downsample_ratio`	Integer; downsampling ratio if performing downsampling analysis.

`config_file`	File path to the JSON configuration file. If specified, `config_file` overrides all configuration parameters
`do_genome_align`	Boolean; specifies whether to run the genome alignment step. `TRUE` is recommended
`do_isoform_id`	Boolean; specifies whether to run the isoform identification step. `TRUE` is recommended
`do_read_realign`	Boolean; specifies whether to run the read realignment step. `TRUE` is recommended
`do_transcript_quanti`	Boolean; specifies whether to run the transcript quantification step. `TRUE` is recommended
`gen_raw_isoform`	Boolean; specifies whether a gff3 should be generated containing the raw isoform information in the isoform identification step
`has_UMI`	Boolean; specifies if the data contains UMI.
`MAX_DIST`	Real; maximum distance allowed when merging splicing sites in isoform consensus clustering.
`MAX_TS_DIST`	Real; maximum distance allowed when merging transcript start/end position in isoform consensus clustering.
`MAX_SPLICE_MATCH_DIST`	Real; maximum distance allowed when merging splice site called from the data and the reference annotation.
`min_fl_exon_len`	Real; minimum length for the first exon outside the gene body in reference annotation. This is to correct the alignment artifact
`Max_site_per_splice`	Real; maximum transcript start/end site combinations allowed per splice chain
`Min_sup_cnt`	Real; minimum number of read support an isoform. Decreasing this number will significantly increase the number of isoform detected.
`Min_cnt_pct`	Real; minimum percentage of count for an isoform relative to total count for the same gene.
`Min_sup_pct`	Real; minimum percentage of count for an splice chain that support a given transcript start/end site combination.
`strand_specific`	1, -1 or 0. 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`	Real; determines the strength of truncated isoform filtering. Larger number means more stringent filtering.
`use_junctions`	Boolean; determiens whether to use known splice junctions to help correct the alignment results
`no_flank`	Boolean; passed to minimap2 for synthetic spike-in data. Refer to Minimap2 document for more details

`use_annotation`	Boolean; specifies whether to use reference to help annotate known isoforms
`min_tr_coverage`	Real; minimum percentage of isoform coverage for a read to be aligned to that isoform
`min_read_coverage`	Real; minimum percentage of read coverage for a read to be uniquely aligned to that isoform

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. Next is the read realignment step (`do_read_realign`), where the sequence of each polished transcript is extracted and used as the updated reference. The reads are realigned to this reference 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

`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)

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="/")))]]
fastq_dir <- paste(temp_path, "fastq_dir", sep="/") # the downloaded fastq files need to be in a directory to be mer
dir.create(fastq_dir)
file.copy(c(fastq1, fastq2), fastq_dir)
unlink(c(fastq1, fastq2)) # the original files can be deleted

## Not run:
# run the FLAMES bulk pipeline, using the downloaded files
se <- bulk_long_pipeline(annot=system.file("extdata/SIRV_anno.gtf", package="FLAMES"),
                   fastq=fastq_dir,
             outdir=tempdir(), genome_fa=system.file("extdata/SIRV_genomefa.fasta", package="FLAMES"),
             config_file=system.file("extdata/SIRV_config_default.json", package="FLAMES"))

## End(Not run)
# OR
# run the FLAMES single cell pipeline
#sce <- sc_long_pipeline(annot, fastq, NULL, outdir, genome_fa, match_barcode=FALSE, config+file=config)
```

---

bulk_windows_pipeline_setup
FLAMES Windows Bulk Pipeline

Description

An implementation of the FLAMES pipeline designed to run on Windows, or any OS without access to minimap2, for read realignment. This pipeline requires external read alignment, in betwen pipeline calls.

Usage

```
bulk_windows_pipeline_setup(
  annot,
  fastq,
  in_bam = NULL,
  outdir,
  genome_fa,
  downsample_ratio = 1,
  config_file
)
```

Arguments

`annot`	gene annotations file in gff3 format
`fastq`	file path to input fastq file
`in_bam`	optional bam file to use instead of fastq files (skips read alignment step)
`outdir`	directory to store all output files.
`genome_fa`	genome fasta file.
`downsample_ratio`	downsampling ratio if performing downsampling analysis.
`config_file`	JSON configuration file. If specified, `config_file` overrides all configuration parameters

Details

This function, `bulk_windows_pipeline_setup` is the first step in the 3 step Windows FLAMES bulk pipeline, and should be run first, read alignment undertaken, then `windows_pipline_isoforms` should be run, read realignment performed, and finally `windows_pipeline_quantification` should be run. For each function, besides `bulk_windows_pipeline_setup`, a list `pipeline_variables` is returned, which contains the information required to continue the pipeline. This list should be passed into each function, and updated with the returned list. In the case of `bulk_windows_pipeline_setup`, `pipeline_variables` is the list returned. See the vignette 'Vignette for FLAMES bulk on Windows' for more details.

Value

a list `pipeline_variables` with the required variables for execution of later Windows pipeline steps. File paths required to perform minimap2 alignment are given in pipeline_variables$return_files. This list should be given as input for `windows_pipeline_isoforms` after minimap2 alignment has taken place; `windows_pipeline_isoforms` is the continuation of this pipeline.

Examples

```
## example windows pipeline for BULK data. See Vignette for single cell data.

# 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="/")))]]
fastq_dir <- paste(temp_path, "fastq_dir", sep="/") # the downloaded fastq files need to be in a directory to be mer
dir.create(fastq_dir)
file.copy(c(fastq1, fastq2), fastq_dir)
unlink(c(fastq1, fastq2)) # the original files can be deleted

# run the FLAMES bulk pipeline setup
#pipeline_variables <- bulk_windows_pipeline_setup(annot=system.file("extdata/SIRV_anno.gtf", package="FLAMES")
#                   fastq=fastq_dir,
#            outdir=tempdir(), genome_fa=system.file("extdata/SIRV_genomefa.fasta", package="FLAMES"),
#            config_file=system.file("extdata/SIRV_config_default.json", package="FLAMES"))
# read alignment is handled externally (below downloads aligned bam for example)
# genome_bam <- paste0(temp_path, "/align2genome.bam")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Genome BAM", paste(file_url, "align2genome.bam", sep="/")))
#
# genome_index <- paste0(temp_path, "/align2genome.bam.bai")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Genome BAM Index", paste(file_url, "align2genome.bam.bai", s
# pipeline_variables$genome_bam = genome_bam
#
# # run the FLAMES bulk pipeline find isoforms step
# pipeline_variables <- windows_pipeline_isoforms(pipeline_variables)
#
# # read realignment is handled externally
# realign_bam <- paste0(temp_path, "/realign2genome.bam")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Realign BAM", paste(file_url, "realign2transcript.bam", sep
#
# realign_index <- paste0(temp_path, "/realign2genome.bam.bai")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Realign BAM Index", paste(file_url, "realign2transcript.bam
# pipeline_variables$realign_bam <- realign_bam
#
# # finally, quantification, which returns a Summarized Experiment object
# se <- windows_pipeline_quantification(pipeline_variables)
```

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

Description

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

Usage

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

Arguments

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

Value

the result of ‘FUN‘

`create_config` *Create Configuration File From Arguments*

Description

Create Configuration File From Arguments

Usage

```
create_config(
  outdir,
  do_genome_align,
  do_isoform_id,
  do_read_realign,
  do_transcript_quanti,
  gen_raw_isoform,
  has_UMI,
  MAX_DIST,
  MAX_TS_DIST,
  MAX_SPLICE_MATCH_DIST,
  min_fl_exon_len,
  Max_site_per_splice,
  Min_sup_cnt,
  Min_cnt_pct,
  Min_sup_pct,
  strand_specific,
  remove_incomp_reads,
  use_junctions,
  no_flank,
  use_annotation,
  min_tr_coverage,
  min_read_coverage
)
```

Arguments

`outdir`	the destination directory for the configuratio nfile
`do_genome_align`	Boolean. Specifies whether to run the genome alignment step. `TRUE` is recommended
`do_isoform_id`	Boolean. Specifies whether to run the isoform identification step. `TRUE` is recommended
`do_read_realign`	Boolean. Specifies whether to run the read realignment step. `TRUE` is recommended
`do_transcript_quanti`	Boolean. Specifies whether to run the transcript quantification step. `TRUE` is recommended
`gen_raw_isoform`	Boolean.
`has_UMI`	Boolean. Specifies if the data contains UMI.
`MAX_DIST`	Maximum distance allowed when merging splicing sites in isoform consensus clustering.
`MAX_TS_DIST`	Maximum distance allowed when merging transcript start/end position in isoform consensus clustering.
`MAX_SPLICE_MATCH_DIST`	Maximum distance allowed when merging splice site called from the data and the reference annotation.
`min_fl_exon_len`	Minimum length for the first exon outside the gene body in reference annotation. This is to correct the alignment artifact
`Max_site_per_splice`	Maximum transcript start/end site combinations allowed per splice chain
`Min_sup_cnt`	Minimum number of read support an isoform decrease this number will significantly increase the number of isoform detected.
`Min_cnt_pct`	Minimum percentage of count for an isoform relative to total count for the same gene.
`Min_sup_pct`	Minimum percentage of count for an splice chain that support a given transcript start/end site combination.
`strand_specific`	1, -1 or 0. 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`.

Value

file path to the config file created

Examples

```
# create the default configuartion file
output = tempfile()
## Not run:
config <- create_config(
    tempfile(), TRUE, TRUE,
    TRUE, TRUE,
    TRUE, FALSE,
    10, 100, 10,
    40, 3, 10,
    0.01, 0.2, 1, 5,
    TRUE, TRUE,
    TRUE, 0.75, 0.75
)

## End(Not run)
```

`generate_umap` *Generate UMAP plot*

Description

Generates the matrix of UMAP visualisation coordinates from a given SummarizedExperiment. The modified SummarizedExperiment object is returned, with UMAP coordinates stored in the ‘reducedDims()‘ slot.

Usage

```
generate_umap(summarizedExperiment, counts = "counts")
```

Arguments

`summarizedExperiment`	the SingleCellExperiment of SummarizedExperiment object containing the counts to plot
`counts`	character string indicating the name of the assay to generate UMAP visualtization from. The assay should be accessable by ‘assay(summarizedExperiemnt, counts)‘.

Value

Modified SummarizedExperiment (or SingleCellExperiment) containing the UMAP visualisation coordinates in the ‘reducedDims()‘ UMAP slot.

See Also

[sc_long_pipeline()]

[bulk_long_pipeline()]

`generic_long_pipeline` *Generic FLAMES pipeline*

Description

Generic implementation of the flames pipeline. Used for both bulk reads and single cell reads.

Usage

```
generic_long_pipeline(
  annot,
  fastq,
  in_bam,
  outdir,
  genome_fa,
  minimap2_dir,
  downsample_ratio,
  config_file,
  do_genome_align,
  do_isoform_id = TRUE,
  do_read_realign,
  do_transcript_quanti,
  gen_raw_isoform,
  has_UMI,
  MAX_DIST,
  MAX_TS_DIST,
  MAX_SPLICE_MATCH_DIST,
  min_fl_exon_len,
  Max_site_per_splice,
  Min_sup_cnt,
  Min_cnt_pct,
  Min_sup_pct,
  strand_specific,
  remove_incomp_reads,
  use_junctions,
  no_flank,
  use_annotation,
  min_tr_coverage,
  min_read_coverage
)
```

Arguments

`annot`	The file path to gene annotations file in gff3 format
`fastq`	The file path to input fastq file
`in_bam`	optional BAM file which replaces fastq directory argument. This skips the genome alignment and realignment steps
`outdir`	The path to directory to store all output files.
`genome_fa`	The file path to genome fasta file.
`minimap2_dir`	Path to the directory containing minimap2, if it is not in PATH. Only required if either or both of `do_genome_align` and `do_read_realign` are `TRUE`.
`downsample_ratio`	Integer; downsampling ratio if performing downsampling analysis.
`config_file`	File path to the JSON configuration file. If specified, `config_file` overrides all configuration parameters
`do_genome_align`	Boolean; specifies whether to run the genome alignment step. `TRUE` is recommended
`do_isoform_id`	Boolean; specifies whether to run the isoform identification step. `TRUE` is recommended
`do_read_realign`	Boolean; specifies whether to run the read realignment step. `TRUE` is recommended
`do_transcript_quanti`	Boolean; specifies whether to run the transcript quantification step. `TRUE` is recommended
`gen_raw_isoform`	Boolean; specifies whether a gff3 should be generated containing the raw isoform information in the isoform identification step
`has_UMI`	Boolean; specifies if the data contains UMI.
`MAX_DIST`	Real; maximum distance allowed when merging splicing sites in isoform consensus clustering.
`MAX_TS_DIST`	Real; maximum distance allowed when merging transcript start/end position in isoform consensus clustering.

`MAX_SPLICE_MATCH_DIST`	Real; maximum distance allowed when merging splice site called from the data and the reference annotation.
`min_fl_exon_len`	Real; minimum length for the first exon outside the gene body in reference annotation. This is to correct the alignment artifact
`Max_site_per_splice`	Real; maximum transcript start/end site combinations allowed per splice chain
`Min_sup_cnt`	Real; minimum number of read support an isoform. Decreasing this number will significantly increase the number of isoform detected.
`Min_cnt_pct`	Real; minimum percentage of count for an isoform relative to total count for the same gene.
`Min_sup_pct`	Real; minimum percentage of count for an splice chain that support a given transcript start/end site combination.
`strand_specific`	1, -1 or 0. 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`	Real; determines the strength of truncated isoform filtering. Larger number means more stringent filtering.
`use_junctions`	Boolean; determiens whether to use known splice junctions to help correct the alignment results
`no_flank`	Boolean; passed to minimap2 for synthetic spike-in data. Refer to Minimap2 document for more details
`use_annotation`	Boolean; specifies whether to use reference to help annotate known isoforms
`min_tr_coverage`	Real; minimum percentage of isoform coverage for a read to be aligned to that isoform
`min_read_coverage`	Real; minimum percentage of read coverage for a read to be uniquely aligned to that isoform

Value

This generic function returns a named list containing the output file names of the provided output files in the given ‘outdir‘ directory. These files are loaded into R in either a SummarizedExperiment or SingleCellExperiment object by the callers to this function, ‘sc_long_pipeline()‘ and ‘bulk_long_pipeline()‘ respectively.

`get_default_config_file` *Default Configuration File*

Description

file path to the default FLAMES configuration file

Usage

```
get_default_config_file()
```

Value

file path to the FLAMES default configuration file

Examples

```
config <- get_default_config_file()
```

`gff3_to_bed12` *GFF3 to BED12*

Description

Converts a gff3 file to a bed12

Usage

```
gff3_to_bed12(gff3_file, bed12_file)
```

Arguments

`gff3_file`	The file path to the GFF3 file to convert
`bed12_file`	The file path of the bed12 output file.

Value

file path to the created bed12_file.

Examples

```
annot <- system.file("extdata/SIRV_anno.gtf", package = "FLAMES")
out_bed <- tempfile(fileext = ".bed12")
## Not run:
gff3_to_bed12(annot, out_bed)

## End(Not run)
```

`match_cell_barcode` *Match Cell Barcodes*

Description

Match cell barcodes in the given fastq directory with the reference csv, `ref_csv`. Matches are returned in the output file `out_fastq`

Usage

```
match_cell_barcode(
  fastq_dir,
  stats_file,
  out_fastq,
  ref_csv,
  MAX_DIST,
  UMI_LEN = 10L
)
```

Arguments

`fastq_dir`	directory containing fastq files to match
`stats_file`	NEEDED
`out_fastq`	output filename for matched barcodes
`ref_csv`	NEEDED
`MAX_DIST`	int; maximum edit distance
`UMI_LEN`	int; length of UMI sequences

Value

returns NULL

`merge_bulk_fastq` *Merge FASTQ*

Description

Merges all fastq files in the given folder into a single file.

Usage

```
merge_bulk_fastq(fastq_dir, out_fastq)
```

Arguments

`fastq_dir`	Path to the folder containing fastq files to merge
`out_fastq`	file path to the fastq file which will be created to store all fastq entries. Overwrites existing files

Value

file path to the created merged fastq file ‘out_fastq‘

Examples

```
# download the fastq files to merge
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 = "/")))]
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

# merge the fastq files
out_fastq <- paste0(temp_path, "/outfastq.fastq.gz")
## Not run:
merge_bulk_fastq(fastq_dir, out_fastq)

## End(Not run)
```

`merge_bulk_fastq_cpp` *Merge Bulk Fastq Files*

Description

Merge all fastq files into a single fastq This function also inserts the original fastq file name into the header line of each read.

Usage

```
merge_bulk_fastq_cpp(fastq_files, out_fastq)
```

Arguments

`fastq_files`	a string vector of the fastq file paths to merge
`out_fastq`	the fastq file path of the output file

Value

returns NULL

`merge_bulk_fastq_python` *Merge FASTQ using python. Depreciated*

Description

Merges all fastq files in the given folder into a single file.

Usage

```
merge_bulk_fastq_python(fastq_dir, out_fastq)
```

Arguments

`fastq_dir`	Path to the folder containing fastq files to merge
`out_fastq`	file path to the fastq file which will be created to store all fastq entries. Overwrites existing files

Value

file path to the created merged fastq file ‘out_fastq‘

Examples

```
# download the fastq files to merge
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 = "/")))]
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

# merge the fastq files
out_fastq <- paste0(temp_path, "/outfastq.fastq.gz")
## Not run:
merge_bulk_fastq(fastq_dir, out_fastq)

## End(Not run)
```

`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(
  minimap2_prog_path = NULL,
  fa_file,
  fq_in,
  sam_out,
  no_flank = FALSE,
  bed12_junc = NULL
)
```

Arguments

`minimap2_prog_path`	Absolute path to the directory containing minimap2
`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
`sam_out`	File path to the output SAM file
`no_flank`	Boolean; used if studying SIRV, to tell minimap2 to ignore additional bases
`bed12_junc`	file path to the gene annotations in BED12 format. If specified, minmap2 prefers splicing in annotations.

Value

file path to the given output BAM file, `bam_out`

`minimap2_check_callable` *Check if minimap2 is available*

Description

Checks if minimap2 is available from given directory or in path. Uses python's subprocess module to check if the help page is accessable.

Usage

```
minimap2_check_callable(mm2_prog_path)
```

Arguments

`mm2_prog_path`	the path to the directory containing minimap2

Value

TRUE if minimap2 is available, FALSE otherwise

`parse_gff_tree` *Parse Gff3 file*

Description

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

Usage

```
parse_gff_tree(gff_file)
```

Arguments

`gff_file`	the file path to the gff3 file to parse

Value

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

Examples

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

## End(Not run)
```

`parse_json_config` *Parse Json Configuration File*

Description

Convert a json configuration file into a named R list, grouped into sub lists according to their usage in the Flames pipeline.

Usage

```
parse_json_config(json_file)
```

Arguments

`json_file`	the file path to the JSON file to convert into an R list. This can be the default FLAMES configuration file found using `get_default_config_file()`

Value

A named R list of the parameters in `json_file`. Subsections are: `pipeline_parameters`, `global_parameters`, `isoform_parameters`, `alignment_parameters`, `realign_parameters` and `transcript_counting`.

Examples

```
config <- get_default_config_file()
## Not run:
parse_json_config(config)

## End(Not run)
```

`print_config` *Print Configuration File*

Description

Print Configuration File

Usage

```
print_config(config)
```

Arguments

`config`	List; the configuration list to print.

Details

Print the configuration file, represented as a named list used for the Flames pipeline.

Value

return NULL

---

samtools_as_bam *Samtools as BAM*

Description

Produces a compressed binary BAM file from a text based SAM file, using Rsamtools.

Usage

```
samtools_as_bam(sam_in, bam_out)
```

Arguments

`sam_in`	file path to the input SAM file
`bam_out`	file path to the output BAM file

Value

file path to the output BAM file.

Examples

```
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <-
    "https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data"
tmp_sam <- paste0(temp_path, "/tmp_sam.sam")
file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Temp SAM", paste(file_url, "tmp_sam.sam", sep = "/")))]], tmp_

samtools_as_bam(tmp_sam, tempfile(fileext = "bam"))
```

`samtools_sort_index` *Samtools Sort and Index*

Description

Sort and index the given BAM file, using Rsamtools.

Usage

```
samtools_sort_index(bam_in, bam_out)
```

Arguments

`bam_in`	the file path to the BAM file to sort and index
`bam_out`	path to the output indexed BAM file.

Value

file path to the created BAM

Examples

```
temp_path <- tempfile()
bfc <- BiocFileCache::BiocFileCache(temp_path, ask = FALSE)
file_url <-
    "https://raw.githubusercontent.com/OliverVoogd/FLAMESData/master/data"
genome_bam <- paste0(temp_path, "/align2genome.bam")
file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Genome BAM", paste(file_url, "align2genome.bam", sep = "/")))]

samtools_sort_index(genome_bam, tempfile(fileext = ".bam"))
```

`sc_DTU_analysis` *FLAMES Differential Transcript Usage Analysis*

Description

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

Usage

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

Arguments

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

Details

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

Value

a `data.frame` containing the following columns:

- gene_name - differentially transcribed genes
- X_value - the X value for the DTU gene
- df - degrees of freedom of the approximate chi-squared distribution of the test statistic
- DTU_tr - the transcript_id with the highest squared residuals
- DTU_group - the cell group with the highest squared residuals
- p_value - the p-value for the test
- adj_p - the adjusted p-value (by Benjamini–Hochberg correction)

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

`sc_long_pipeline` *Pipeline for Single Cell Data*

Description

Semi-supervised isoform detection and annotation for long read data. This varient 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(
  annot,
  fastq,
  in_bam = NULL,
  outdir,
  genome_fa,
  minimap2_dir = "",
  downsample_ratio = 1,
  reference_csv,
  match_barcode = TRUE,
  config_file = NULL,
  do_genome_align = TRUE,
  do_isoform_id = TRUE,
  do_read_realign = TRUE,
  do_transcript_quanti = TRUE,
  gen_raw_isoform = TRUE,
  has_UMI = FALSE,
  UMI_LEN = 10,
  MAX_DIST = 10,
  MAX_TS_DIST = 100,
  MAX_SPLICE_MATCH_DIST = 10,
  min_fl_exon_len = 40,
  Max_site_per_splice = 3,
  Min_sup_cnt = 10,
  Min_cnt_pct = 0.01,
  Min_sup_pct = 0.2,
  strand_specific = 1,
  remove_incomp_reads = 5,
  use_junctions = TRUE,
  no_flank = TRUE,
  use_annotation = TRUE,
  min_tr_coverage = 0.75,
  min_read_coverage = 0.75
)
```

Arguments

`annot`	The file path to gene annotations file in gff3 format
`fastq`	The file path to input fastq file
`in_bam`	Optional file path to a bam file to use instead of fastq file (skips initial alignment step)
`outdir`	The path to directory to store all output files.
`genome_fa`	The file path to genome fasta file.
`minimap2_dir`	Path to the directory containing minimap2, if it is not in PATH. Only required if either or both of `do_genome_align` and `do_read_realign` are `TRUE`.

`downsample_ratio`	Integer; downsampling ratio if performing downsampling analysis.
`reference_csv`	The file path to the reference csv used for demultiplexing
`match_barcode`	Boolean; specifies if demultiplexing should be performed using `FLAMES::match_cell_barcode`
`config_file`	File path to the JSON configuration file. If specified, `config_file` overrides all configuration parameters
`do_genome_align`	Boolean; specifies whether to run the genome alignment step. `TRUE` is recommended
`do_isoform_id`	Boolean; specifies whether to run the isoform identification step. `TRUE` is recommended
`do_read_realign`	Boolean; specifies whether to run the read realignment step. `TRUE` is recommended
`do_transcript_quanti`	Boolean; specifies whether to run the transcript quantification step. `TRUE` is recommended
`gen_raw_isoform`	Boolean; specifies whether a gff3 should be generated containing the raw isoform information in the isoform identification step
`has_UMI`	Boolean; specifies if the data contains UMI.
`UMI_LEN`	Integer; the length of UMI sequence in bases
`MAX_DIST`	Real; maximum distance allowed when merging splicing sites in isoform consensus clustering.
`MAX_TS_DIST`	Real; maximum distance allowed when merging transcript start/end position in isoform consensus clustering.
`MAX_SPLICE_MATCH_DIST`	Real; maximum distance allowed when merging splice site called from the data and the reference annotation.
`min_fl_exon_len`	Real; minimum length for the first exon outside the gene body in reference annotation. This is to correct the alignment artifact
`Max_site_per_splice`	Real; maximum transcript start/end site combinations allowed per splice chain
`Min_sup_cnt`	Real; minimum number of read support an isoform. Decreasing this number will significantly increase the number of isoform detected.
`Min_cnt_pct`	Real; minimum percentage of count for an isoform relative to total count for the same gene.
`Min_sup_pct`	Real; minimum percentage of count for an splice chain that support a given transcript start/end site combination.
`strand_specific`	1, -1 or 0. 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`	Real; determines the strength of truncated isoform filtering. Larger number means more stringent filtering.
`use_junctions`	Boolean; determiens whether to use known splice junctions to help correct the alignment results
`no_flank`	Boolean; passed to minimap2 for synthetic spike-in data. Refer to Minimap2 document for more details
`use_annotation`	Boolean; specifies whether to use reference to help annotate known isoforms
`min_tr_coverage`	Real; minimum percentage of isoform coverage for a read to be aligned to that isoform
`min_read_coverage`	Real; minimum percentage of read coverage for a read to be uniquely aligned to that isoform

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. Next is the read realignment step (`do_read_realign`), where the sequence of each polished transcript is extracted and used as the updated reference. The reads are realigned to this reference 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

`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)

See Also

`bulk_long_pipeline()` for bulk long data, `SingleCellExperiment()` 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="/")))]]
fastq_dir <- paste(temp_path, "fastq_dir", sep="/") # the downloaded fastq files need to be in a directory to be mer
dir.create(fastq_dir)
file.copy(c(fastq1, fastq2), fastq_dir)
unlink(c(fastq1, fastq2)) # the original files can be deleted

## Not run:
# run the FLAMES bulk pipeline, using the downloaded files
se <- bulk_long_pipeline(annot=system.file("extdata/SIRV_anno.gtf", package="FLAMES"),
                   fastq=fastq_dir,
             outdir=tempdir(), genome_fa=system.file("extdata/SIRV_genomefa.fasta", package="FLAMES"),
             config_file=system.file("extdata/SIRV_config_default.json", package="FLAMES"))

## End(Not run)
# OR
# run the FLAMES single cell pipeline
#sce <- sc_long_pipeline(annot, fastq, NULL, outdir, genome_fa, match_barcode=FALSE, config+file=config)
```

sc_mutations *FLAMES variant calling*

Description

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

Usage

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

Arguments

`sce`	The `SingleCellExperiment` object from `sc_long_pipeline`
`barcode_tsv`	TSV file for cell barcodes
`bam_short`	(Optional) short read alignment BAM file. If provided, it is used to filter the variations. Variations in long-read data with enough short read coverage but no alternative allele will not be reported.
`out_dir`	(Optional) Output folder of sc_long_pipeline. Output files from this function will also be saved here. Use this parameter if you do not have the `SingleCellExperiment` object.
`genome_fa`	(Optional) Reference genome FASTA file. Use this parameter is if you do not wish `sc_mutation` to use the reference genome FASTA file from the `sce`'s metadata.
`annot`	(Optional) The file path to gene annotation file in gff3 format. If provided as `FALSE` then the `isoform_annotated.gff3` from `sc_longread_pipeline` will be used, if not provided then the path in the `SingleCellExperiment` object will be used.
`known_positions`	(Optional) A list of known positions, with by chromosome name followed by the position, e.g. ('chr1', 123, 'chr1', 124, 'chrX', 567). These locations will not be filtered and its allele frequencies will be reported.
`min_cov`	The coverage threshold for filtering candidate SNVs. Positions with reads less then this number will not be considered.
`report_pct`	The allele frequency range for filtering candidate SNVs. Positions with less or higher allele frequency will not be reported. The default is 0.10-0.90

Details

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

Value

a `data.frame` containing the following columns:

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

The table is sorted by decreasing adjusted P value.

files saved to out_dir/mutation:

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

`sc_windows_pipeline_setup` *Windows Single Cell FLAMES Pipeline*

Description

An implementation of the FLAMES pipeline designed to run on Windows, or any OS without access to minimap2, for read realignment. This pipeline requires external read alignment, in betwen pipeline calls.

Usage

```
sc_windows_pipeline_setup(
  annot,
  fastq,
  in_bam = NULL,
  outdir,
  genome_fa,
  downsample_ratio = 1,
  config_file,
  match_barcode = TRUE,
  reference_csv = NULL,
  MAX_DIST = 0,
  UMI_LEN = 0
)
```

Arguments

`annot`	gene annotations file in gff3 format
`fastq`	file path to input fastq file
`in_bam`	optional bam file to replace fastq input files
`outdir`	directory to store all output files.
`genome_fa`	genome fasta file.
`downsample_ratio`	downsampling ratio if performing downsampling analysis.
`config_file`	JSON configuration file. If specified, `config_file` overrides all configuration parameters
`match_barcode`	boolean, specify if barcode matching should take place before the pipeline begins
`reference_csv`	reference csv for barcode matching
`MAX_DIST`	max dist
`UMI_LEN`	length of the UMI to find

Details

This function, `sc_windows_pipeline_setup` is the first step in the 3 step Windows FLAMES single cell pipeline, and should be run first, read alignment undertaken, then `windows_pipline_isoforms` should be run, read realignment performed, and finally `windows_pipeline_quantification` should be run. For each function, besides `sc_windows_pipeline_setup`, a list `pipeline_variables` is returned, which contains the information required to continue the pipeline. This list should be passed into each function, and updated with the returned list. In the case of `sc_windows_pipeline_setup`, `pipeline_variables` is the list returned. See the vignette 'Vignette for FLAMES bulk on Windows' for more details.

Value

a list `pipeline_variables` with the required variables for execution of later Windows pipeline steps. File paths required to perform minimap2 alignment are given in pipeline_variables$return_files. This list should be given as input for `windows_pipeline_isoforms` after minimap2 alignment has taken place; `windows_pipeline_isoforms` is the continuation of this pipeline.

Examples

```
## example windows pipeline for BULK data. See Vignette for single cell data.

# 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="/")))]]
fastq_dir <- paste(temp_path, "fastq_dir", sep="/") # the downloaded fastq files need to be in a directory to be mer
dir.create(fastq_dir)
file.copy(c(fastq1, fastq2), fastq_dir)
unlink(c(fastq1, fastq2)) # the original files can be deleted

# run the FLAMES bulk pipeline setup
#pipeline_variables <- bulk_windows_pipeline_setup(annot=system.file("extdata/SIRV_anno.gtf", package="FLAMES")
#                   fastq=fastq_dir,
#             outdir=tempdir(), genome_fa=system.file("extdata/SIRV_genomefa.fasta", package="FLAMES"),
#             config_file=system.file("extdata/SIRV_config_default.json", package="FLAMES"))
# read alignment is handled externally (below downloads aligned bam for example)
# genome_bam <- paste0(temp_path, "/align2genome.bam")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Genome BAM", paste(file_url, "align2genome.bam", sep="/")))
#
# genome_index <- paste0(temp_path, "/align2genome.bam.bai")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Genome BAM Index", paste(file_url, "align2genome.bam.bai", s
# pipeline_variables$genome_bam = genome_bam
#
# # run the FLAMES bulk pipeline find isoforms step
# pipeline_variables <- windows_pipeline_isoforms(pipeline_variables)
#
# # read realignment is handled externally
# realign_bam <- paste0(temp_path, "/realign2genome.bam")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Realign BAM", paste(file_url, "realign2transcript.bam", sep
#
# realign_index <- paste0(temp_path, "/realign2genome.bam.bai")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Realign BAM Index", paste(file_url, "realign2transcript.bam
# pipeline_variables$realign_bam <- realign_bam
#
# # finally, quantification, which returns a Summarized Experiment object
# se <- windows_pipeline_quantification(pipeline_variables)
```

`windows_pipeline_isoforms`
Windows Pipeline - Find Isoforms

Description

This is the second step in the 3 step Windows FLAMES pipeline. Following this step, read realignment should be undertaken, using the file paths given in the return pipeline_variables$return_files. After this has been completed, the final pipeline step, `windows_pipeline_quantification` should be run, giving the returned list from this function as input.

Usage

```
windows_pipeline_isoforms(pipeline_variables)
```

Arguments

`pipeline_variables`	the list returned from `windows_pipeline_isoforms`.

Value

the updated `pipeline_variables` list, with information required for the final pipeline step.

Examples

```
## example windows pipeline for BULK data. See Vignette for single cell data.

# 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="/")))]]
fastq_dir <- paste(temp_path, "fastq_dir", sep="/") # the downloaded fastq files need to be in a directory to be mer
dir.create(fastq_dir)
file.copy(c(fastq1, fastq2), fastq_dir)
unlink(c(fastq1, fastq2)) # the original files can be deleted

# run the FLAMES bulk pipeline setup
#pipeline_variables <- bulk_windows_pipeline_setup(annot=system.file("extdata/SIRV_anno.gtf", package="FLAMES")
#                   fastq=fastq_dir,
#             outdir=tempdir(), genome_fa=system.file("extdata/SIRV_genomefa.fasta", package="FLAMES"),
#             config_file=system.file("extdata/SIRV_config_default.json", package="FLAMES"))
# read alignment is handled externally (below downloads aligned bam for example)
# genome_bam <- paste0(temp_path, "/align2genome.bam")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Genome BAM", paste(file_url, "align2genome.bam", sep="/")))
#
# genome_index <- paste0(temp_path, "/align2genome.bam.bai")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Genome BAM Index", paste(file_url, "align2genome.bam.bai", s
# pipeline_variables$genome_bam = genome_bam
#
# # run the FLAMES bulk pipeline find isoforms step
# pipeline_variables <- windows_pipeline_isoforms(pipeline_variables)
#
# # read realignment is handled externally
# realign_bam <- paste0(temp_path, "/realign2genome.bam")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Realign BAM", paste(file_url, "realign2transcript.bam", sep
#
# realign_index <- paste0(temp_path, "/realign2genome.bam.bai")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Realign BAM Index", paste(file_url, "realign2transcript.bam
# pipeline_variables$realign_bam <- realign_bam
#
# # finally, quantification, which returns a Summarized Experiment object
# se <- windows_pipeline_quantification(pipeline_variables)
```

windows_pipeline_quantification

Windows Pipeline - Quantification

Description

This is the final step in the 3 step Windows FLAMES pipeline. This should be run after read realignment is performed, following `windows_pipeline_isoforms`.

Usage

```
windows_pipeline_quantification(pipeline_vars)
```

Arguments

`pipeline_vars`	the list returned from `windows_pipeline_isoforms`, containing the information required to perform the final step, quantification.

Value

`windows_pipeline_quantification` returns a SummarizedExperiment object, or a SingleCellExperiment in the case of this function being used for the FLAMES single cell pipeline, 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)

Examples

```
## example windows pipeline for BULK data. See Vignette for single cell data.

# 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="/")))]]
fastq_dir <- paste(temp_path, "fastq_dir", sep="/") # the downloaded fastq files need to be in a directory to be mer
dir.create(fastq_dir)
file.copy(c(fastq1, fastq2), fastq_dir)
unlink(c(fastq1, fastq2)) # the original files can be deleted

# run the FLAMES bulk pipeline setup
#pipeline_variables <- bulk_windows_pipeline_setup(annot=system.file("extdata/SIRV_anno.gtf", package="FLAMES")
#                   fastq=fastq_dir,
#             outdir=tempdir(), genome_fa=system.file("extdata/SIRV_genomefa.fasta", package="FLAMES"),
#             config_file=system.file("extdata/SIRV_config_default.json", package="FLAMES"))
# read alignment is handled externally (below downloads aligned bam for example)
# genome_bam <- paste0(temp_path, "/align2genome.bam")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Genome BAM", paste(file_url, "align2genome.bam", sep="/")))
#
# genome_index <- paste0(temp_path, "/align2genome.bam.bai")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Genome BAM Index", paste(file_url, "align2genome.bam.bai", s
# pipeline_variables$genome_bam = genome_bam
#
# # run the FLAMES bulk pipeline find isoforms step
# pipeline_variables <- windows_pipeline_isoforms(pipeline_variables)
#
# # read realignment is handled externally
# realign_bam <- paste0(temp_path, "/realign2genome.bam")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Realign BAM", paste(file_url, "realign2transcript.bam", sep
#
# realign_index <- paste0(temp_path, "/realign2genome.bam.bai")
# file.rename(bfc[[names(BiocFileCache::bfcadd(bfc, "Realign BAM Index", paste(file_url, "realign2transcript.bam
# pipeline_variables$realign_bam <- realign_bam
#
# # finally, quantification, which returns a Summarized Experiment object
# se <- windows_pipeline_quantification(pipeline_variables)
```

`write_config` *Write Configuration Dictionary to File*

Description

Write Configuration Dictionary to File

Usage

```
write_config(config, config_file)
```

Arguments

`config`	List; the configuration list to write to file.
`config_file`	the file to output `config` to. Should be .json extension

Details

Write the configuration file, represented as a named list used for the Flames pipeline.

Value

returns NULL

Index