Package 'scifer'

January 24, 2026

Type Package

Title Scifer: Single-Cell Immunoglobulin Filtering of Sanger Sequences

Version 1.13.1

URL https://github.com/rodrigarc/scifer

BugReports https://github.com/rodrigarc/scifer/issues

Description Have you ever index sorted cells in a 96 or 384-well plate and then sequenced using Sanger sequencing? If so, you probably had some struggles to either check the electropherogram of each cell sequenced manually, or when you tried to identify which cell was sorted where after sequencing the plate. Scifer was developed to solve this issue by performing basic quality control of Sanger sequences and merging flow cytometry data from probed single-cell sorted B cells with sequencing data. scifer can export summary tables, 'fasta' files, electropherograms for visual inspection, and generate reports.

License MIT + file LICENSE

Encoding UTF-8

biocViews Preprocessing, QualityControl, SangerSeq, Sequencing, Software, FlowCytometry, SingleCell

Imports dplyr, rmarkdown, data.table, Biostrings, stats, plyr, knitr, ggplot2, gridExtra, DECIPHER, stringr, sangerseqR, kableExtra, tibble, scales, rlang, flowCore, methods, basilisk, basilisk.utils, reticulate, here, pwalign, utils

RoxygenNote 7.3.2

VignetteBuilder knitr

Suggests BiocBaseUtils, fs, BiocStyle, testthat (>= 3.0.0)

Enhances parallel

Config/testthat/edition 3

StagedInstall no

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

git_branch devel

git_last_commit 110a701

git_last_commit_date 2025-12-02

Repository Bioconductor 3.23

Date/Publication 2026-01-23

Author Rodrigo Arcoverde Cerveira [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-1145-2534>),
Marcel Martin [ctb],
Matthew James Hinchcliff [ctb],
Sebastian Ols [aut, dtc] (ORCID:
<https://orcid.org/0000-0001-9784-7176>),
Karin Loré [dtc, ths, fnd] (ORCID:
<https://orcid.org/0000-0001-7679-9494>)

Maintainer Rodrigo Arcoverde Cerveira <rodrigo.arcoverdi@gmail.com>

Contents

`df_to_fasta` *Fasta file creation from dataframe columns and/or vectors.*

Description

Creates a fasta file from vectors of names and sequences.

Usage

```
df_to_fasta(
  sequence_name,
  sequence_strings,
  file_name = "sequences.fasta",
  output_dir = NULL,
  save_fasta = TRUE
)
```

Arguments

sequence_name	Vector containing the names for each sequence, usually a column from a data.frame. eg. df$sequence_name
sequence_strings	Vector containing the DNA or RNA or AA sequences, usually a column from a data.frame. eg. df$sequences
file_name	Output file name to be saved as a fasta file
output_dir	Output directory for the fasta file. Default is the working directory
save_fasta	Logical argument, TRUE or FALSE, to indicate if fasta files should be saved. Default is TRUE.

Value

Saves a fasta file in the desired location, and also returns the stringset as BStringSet if saved as an object.

Examples

```
## Example with vectors, default for save_fasta ir TRUE
df_to_fasta(
    sequence_name = c("myseq1", "myseq2"),
    sequence_strings = c("GATCGAT", "ATCGTAG"),
    file_name = "my_sequences.fasta",
    output_dir = "",
    save_fasta = FALSE
)
```

---

fcs_plot *Plot flow data from index sorted cells*

Description

Plot a traditional flow density plot with the sorted cells and the selected thresholds for the two probes used in ‘fcs_processing()‘.

Usage

```
fcs_plot(processed_fcs_list = NULL)
```

Arguments

processed_fcs_list	List generated using ‘fcs_processing()‘ containing two data.frames

Value

Returns a ggplot object with a traditional flow density plot with the sorted cells and the selected thresholds for the two probes used in fcs_processing().

Examples

```
index_sort_data <- fcs_processing(
    folder_path = system.file("/extdata/fcs_index_sorting",
        package = "scifer"
    ),
    compensation = TRUE, plate_wells = 96,
    probe1 = "Pre.F", probe2 = "Post.F",
    posvalue_probe1 = 600, posvalue_probe2 = 400
)

fcs_plot_obj <- fcs_plot(index_sort_data)
```

`fcs_processing` *Extract index sorting from flow cytometry data*

Description

Extracts the Mean Fluoresnce Intensity (MFI) values from the flow cytometry index files (.fcs) and assign specificity to each single-cell sorted well according to the fluorescence intensity of the probes.

Usage

```
fcs_processing(
  folder_path = "test/test_dataset/fcs_files/",
  compensation = TRUE,
  plate_wells = 96,
  probe1 = "Pre.F",
  probe2 = "Post.F",
  posvalue_probe1 = 600,
  posvalue_probe2 = 400
)
```

Arguments

`folder_path`	Folder containing all the flow data index filex (.fcs). Files should be named with their sample/plate ID. eg. "E11_01.fcs"
`compensation`	Logical argument, TRUE or FALSE, to indicate if the index files were compensated or not. If TRUE, it will apply its compensation prior assigning specificity
`plate_wells`	Type of plate used for single-cell sorting. eg. "96" or "384"
`probe1`	Name of the first channel used for the probe or the custom name assigned to the channel in the index file. eg. "FSC.A", "FSC.H", "SSC.A","DsRed.A", "PE.Cy5_5.A", "PE.Cy7.A","BV650.A", "BV711.A","Alexa.Fluor.700.A" "APC.Cy7.A","PerCP.Cy5.5.A
`probe2`	Name of the second channel used for the probe or the custom name assigned to the channel in the index file. eg. "FSC.A", "FSC.H", "SSC.A","DsRed.A", "PE.Cy5_5.A", "PE.Cy7.A","BV650.A", "BV711.A","Alexa.Fluor.700.A" "APC.Cy7.A","PerCP.Cy5.5.A

posvalue_probe1
: Threshold used for fluorescence intensities to be considered as positive for the first probe

posvalue_probe2
: Threshold used for fluorescence intensities to be considered as positive for the second probe

Value

If saved as an object, it returns a table containing all the processed flow cytometry index files, with their fluorescence intensities for each channel and well position.

Examples

```
index_sort_data <- fcs_processing(
    folder_path = system.file("/extdata/fcs_index_sorting",
        package = "scifer"
    ),
    compensation = TRUE, plate_wells = 96,
    probe1 = "Pre.F", probe2 = "Post.F",
    posvalue_probe1 = 600, posvalue_probe2 = 400
)
```

---

igblast *Run IgBLAST python wrapper*

Description

A wrapper funtion to run the IgBLAST python script to annotate VDJ sequences. It is python based and relies on conda environments that are built when the funtion is called.

Usage

```
igblast(database, fasta, threads = 1)
```

Arguments

database	Vector containing the database for VDJ sequences
fasta	Vector containing the sequences, usually a column from a data.frame. eg. df$sequences
threads	Variable containing the number of cores when computing in parallel, default threads = 1

Value

Creates a data frame with the IgBLAST annotation where each row is the queried sequence with columns containing the IgBLAST results

Examples

```
## Example with test sequences
## Not run:
igblast(
    database = system.file("/extdata/test_fasta/KIMDB_rm", package = "scifer"),
    fasta = system.file("/extdata/test_fasta/test_igblast.txt", package = "scifer"),
    threads = 1
)

## End(Not run)
```

---

quality_report *Generate general and individualized reports*

Description

This function uses the other functions already described to create a HTML report based on sequencing quality. Besides the HTML reports, it also creates fasta files with all the sequences and individualized sequences, in addition to a csv file with the quality scores and sequences considered as good quality.

Usage

```
quality_report(
  folder_sequences = "path/to/sanger_sequences",
  outputfile = "QC_report.html",
  output_dir = "test/",
  processors = NULL,
  folder_path_fcs = NULL,
  plot_chromatogram = FALSE,
  raw_length = 343,
  trim_start = 65,
  trim_finish = 400,
  trimmed_mean_quality = 30,
  compensation = TRUE,
  plate_wells = "96",
  probe1 = "Pre.F",
  probe2 = "Post.F",
  posvalue_probe1 = 600,
  posvalue_probe2 = 400,
  cdr3_start = 100,
  cdr3_end = 150
)
```

Arguments

`folder_sequences`
: Full file directory for searching all ab1 files in a recursive search method. It includes all files in subfolders

`outputfile`
: Output file name for the report generation

`output_dir`
: Output directory for all the different output files that are generated during the report

`processors`
: Number of processors to use, you can set to NULL to detect automatically all available processors

`folder_path_fcs`
: Full file directory for searching all flow cytometry index files, files with .fcs extensions, in a recursive search method

`plot_chromatogram`
: Logical argument, TRUE or FALSE, to indicate if chromatograms should be plotted or not. Default is FALSE

`raw_length`
: Minimum sequence length for filtering. Default is 343 for B cell receptors

`trim_start`
: Starting position where the sequence should start to have a good base call accuracy. Default is 65 for B cell receptors

`trim_finish`
: Last position where the sequence should have a good base call accuracy. Default is 400 for B cell receptors

`trimmed_mean_quality`
: Minimum Phred quality score expected for an average sequence. Default is 30, which means average of 99.9% base call accuracy

`compensation`
: Logical argument, TRUE or FALSE, to indicate if the index files were compensated or not. If TRUE, it will apply its compensation prior assigning specificities

`plate_wells`
: Type of plate used for single-cell sorting. eg. "96" or "384"

`probe1`
: Name of the first channel used for the probe or the custom name assigned to the channel in the index file. eg. "FSC.A", "FSC.H", "SSC.A","DsRed.A", "PE.Cy5_5.A", "PE.Cy7.A","BV650.A", "BV711.A","Alexa.Fluor.700.A" "APC.Cy7.A","PerCP.Cy5.5.A

`probe2`
: Name of the second channel used for the probe or the custom name assigned to the channel in the index file. eg. "FSC.A", "FSC.H", "SSC.A","DsRed.A", "PE.Cy5_5.A", "PE.Cy7.A","BV650.A", "BV711.A","Alexa.Fluor.700.A" "APC.Cy7.A","PerCP.Cy5.5.A

`posvalue_probe1`
: Threshold used for fluorescence intensities to be considered as positive for the first probe

`posvalue_probe2`
: Threshold used for fluorescence intensities to be considered as positive for the second probe

`cdr3_start`
: Expected CDR3 starting position, that depends on your primer set. Default is position 100

`cdr3_end`
: Expected CDR3 end position, that depends on your primer set. Default is position 150

Value

Saves HTML reports, fasta files, csv files

Examples

```
quality_report(
    folder_sequences = system.file("extdata/sorted_sangerseq",
        package = "scifer"),
    outputfile = "QC-report.html",
     # output to a temporary directory
    output_dir = tempdir(),
    folder_path_fcs = system.file("/extdata/fcs_index_sorting",
        package = "scifer"),
    processors = 1, compensation = TRUE, plate_wells = "96",
    probe1 = "Pre.F", probe2 = "Post.F",
    posvalue_probe1 = 600, posvalue_probe2 = 400,
    cdr3_start = 100,
    cdr3_end = 150
)
```

`scifer` *Scifer: Single-Cell Immunoglobulin Filtering of Sanger Sequences*

Description

Integrating index single-cell sorted files with Sanger sequencing per plates, combining single-cell sorted data (FACS) and specificity with Sanger sequencing information.

Author(s)

Rodrigo Arcoverde Cerveira <rodrigo.arcoverdi@gmail.com>

See Also

Useful links:

- https://github.com/rodrigarc/scifer
- Report bugs at https://github.com/rodrigarc/scifer/issues

secondary_peaks *Check for secondary peaks in a sangerseq object*

Description

This function finds and reports secondary peaks in a sangerseq object. It returns a table of secondary peaks, and optionally saves an annotated chromatogram and a csv file of the peak locations.

Usage

```
secondary_peaks(
  s,
  ratio = 0.33,
  output.folder = NA,
  file.prefix = "seq",
  processors = NULL
)
```

Arguments

s	a sangerseq s4 object from the sangerseqR package
ratio	Ratio of the height of a secondary peak to a primary peak. Secondary peaks higher than this ratio are annotated. Those below the ratio are not.
output.folder	If output.folder is NA (the default) no files are written. If a valid folder is provided, two files are written to that folder: a .csv file of the secondary peaks (see description below) and a .pdf file of the chromatogram.
file.prefix	If output.folder is specified, this is the prefix which will be appended to the .csv and the .pdf file. The default is "seq".
processors	Number of processors to use, or NULL (the default) for all available processors

Value

A list with two elements:

1. secondary.peaks: a data frame with one row per secondary peak above the ratio, and three columns: "position" is the position of the secondary peak relative to the primary sequence; "primary.basecall" is the primary base call; "secondary.basecall" is the secondary basecall.

2. read: the input sangerseq s4 object after having the makeBaseCalls() function from sangerseqR applied to it. This re-calls the primary and secondary bases in the sequence, and resets a lot of the internal data.

Examples

```
## Read abif using sangerseqR package
s4_sangerseq <- sangerseqR::readsangerseq(
    system.file("/extdata/sorted_sangerseq/E18_C1/A1_3_IgG_Inner.ab1",
        package = "scifer"
    )
)

## Summarise using summarise_abi_file()
processed_seq <- scifer:::secondary_peaks(s4_sangerseq)
```

`summarise_abi_file` *Create a summary of a single ABI sequencing file*

Description

Takes a single ABI sequencing file and returns a summary of the file. The summary includes basic quality control metric of the sequence.

Usage

```
summarise_abi_file(
  seq.abif,
  trim.cutoff = 1e-04,
  secondary.peak.ratio = 0.33,
  output.folder = NA,
  prefix = "seq",
  processors = NULL
)
```

Arguments

`seq.abif`	an abif.seq s4 object from the sangerseqR package
`trim.cutoff`	the cutoff at which you consider a base to be bad. This works on a logarithmic scale, such that if you want to consider a score of 10 as bad, you set cutoff to 0.1; for 20 set it at 0.01; for 30 set it at 0.001; for 40 set it at 0.0001; and so on. Contiguous runs of bases below this quality will be removed from the start and end of the sequence. Default is 0.0001.
`secondary.peak.ratio`	the ratio of the height of a secondary peak to a primary peak. Secondary peaks higher than this ratio are annotated. Those below the ratio are not.
`output.folder`	If output.folder is NA (the default) no files are written. If a valid folder is provided, two files are written to that folder: a .csv file of the secondary peaks (see description below) and a .pdf file of the chromatogram.
`prefix`	If output.folder is specified, this is the prefix which will be appended to the .csv and the .pdf file. The default is "seq".
`processors`	Number of processors to use, or NULL (the default) for all available processors

Value

A numeric vector including:

1. raw.length: the length of the untrimmed sequence, note that this is the sequence after conversion to a sangerseq object, and then the recalling the bases with MakeBaseCalls from the sangerseqR package

2. trimmed.length: the length of the trimmed sequence, after trimming using trim.mott from this package and the parameter supplied to this function

3. trim.start: the start position of the good sequence, see trim.mott for more details

4. trim.finish: the finish position of the good sequence, see trim.mott for more details

5. raw.secondary.peaks: the number of secondary peaks in the raw sequence, called with the secondary.peaks function from this package and the parameters supplied to this function

6. trimmed.secondary.peaks: the number of secondary peaks in the trimmed sequence, called with the secondary.peaks function from this package and the parameters supplied to this function

7. raw.mean.quality: the mean quality score of the raw sequence

8. trimmed.mean.quality: the mean quality score of the trimmed sequence

9. raw.min.quality: the minimum quality score of the raw sequence

10. trimmed.min.quality: the minimum quality score of the trimmed sequence

Examples

```
## Read abif using sangerseqR package
abi_seq <- sangerseqR::read.abif(
    system.file("/extdata/sorted_sangerseq/E18_C1/A1_3_IgG_Inner.ab1",
        package = "scifer"
    )
)

## Summarise using summarise_abi_file()
summarise_abi_file(abi_seq)
```

summarise_quality *Summary table of quality measurements from Sanger sequencing*

Description

Generate a summary table containing quality measurements from Sanger sequencing '.abi' files. This function will read all the '.abi' files in a folder, and generate a summary table containing basic quality metrics.

Usage

```
summarise_quality(
  folder_sequences = "input_folder",
  trim.cutoff = 0.01,
  secondary.peak.ratio = 0.33,
  processors = NULL
)
```

Arguments

folder_sequences	Folder containing all the sanger sequencing abi/ab1 files on subfolders. Each subfolder should have have a identifiable name, matching name with fcs data. eg. "E18_01", "E23_06". The first characters of the ab1 file name should be the well location. eg. "A1-sequence1.ab1", "F8_sequence-igg.ab1"
trim.cutoff	Cutoff at which you consider a base to be bad. This works on a logarithmic scale, such that if you want to consider a score of 10 as bad, you set cutoff to 0.1; for 20 set it at 0.01; for 30 set it at 0.001; for 40 set it at 0.0001; and so on. Contiguous runs of bases below this quality will be removed from the start and end of the sequence. Given the high quality reads expected of most modern ABI sequencers, the defualt is 0.0001.
secondary.peak.ratio	Ratio of the height of a secondary peak to a primary peak. Secondary peaks higher than this ratio are annotated, while those below the ratio are not.
processors	Number of processors to use, or NULL (the default) for all available processors

Value

List containing two items: * summaries: contains all the summary results from the processed abi files, * quality_scores: contains all the Phred quality score for each position.

Examples

```
sf <- summarise_quality(
    folder_sequences = system.file("extdata/sorted_sangerseq",
        package = "scifer"
    ),
```

```
        secondary.peak.ratio = 0.33,
        trim.cutoff = 0.01,
        processor = 1
    )
```

Index