Package 'screenCounter'

November 12, 2026

Version 1.12.0

Date 2024-12-11

Title Counting Reads in High-Throughput Sequencing Screens

Description Provides functions for counting reads from high-throughput sequencing screen data (e.g., CRISPR, shRNA) to quantify barcode abundance. Currently supports single barcodes in single- or paired-end data, and combinatorial barcodes in paired-end data.

Depends S4Vectors, SummarizedExperiment

Imports Rcpp, BiocParallel

Suggests BiocGenerics, Biostrings, BiocStyle, knitr, rmarkdown, testthat

LinkingTo Rcpp

License MIT + file LICENSE

VignetteBuilder knitr

SystemRequirements C++17, GNU make

BugReports https://github.com/crisprVerse/screenCounter/issues

URL https://github.com/crisprVerse/screenCounter

RoxygenNote 7.3.2

Encoding UTF-8

biocViews CRISPR, Alignment, FunctionalGenomics, FunctionalPrediction

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

git_branch RELEASE_3_23

git_last_commit e81795c

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Aaron Lun [aut, cre] (ORCID: <https://orcid.org/0000-0002-3564-4813>)

Maintainer Aaron Lun <infinite.monkeys.with.keyboards@gmail.com>

Contents

screenCounter-package . . . 2
combineComboCounts . . . 2
countComboBarcodes . . . 3
countDualBarcodes . . . 6
countDualBarcodesSingleEnd . . . 9
countPairedComboBarcodes . . . 12
countRandomBarcodes . . . 15
countSingleBarcodes . . . 17
matchBarcodes . . . 19
parseBarcodeTemplate . . . 20

Index **22**

`screenCounter-package` *The screenCounter package*

Description

This package provides methods to counting barcodes from read sequences in high-throughput sequencing screen data sets. It does so by loading sequences from FASTQ files and then matching the barcode template to each sequence using a rolling hash (implemented in C++, inspired by Colin Watanabe's code). This process is performed across several files using a range of parallelization schemes available in **BiocParallel**. We return the resulting count matrix and any feature annotations in a SummarizedExperiment object. Currently, single barcodes (in single- or paired-end data) and combinatorial barcodes are supported.

Author(s)

Aaron Lun

`combineComboCounts` *Combine combinatorial barcode counts*

Description

Combine counts for combinatorial barcodes from multiple files into a single count matrix.

Usage

```
combineComboCounts(...)
```

Arguments

`...`	Any number of DataFrames produced by `countComboBarcodes`.

Value

A DataFrame containing:

- `combinations`, a DataFrame containing all unique combinatorial barcodes observed in any ... Each row corresponds to a barcode and each column contains an identifier (either integer or character) for the sequence in the variable region.
- `counts`, a matrix with number of columns equal to number of objects in .... Each row corresponds to a unique combinatorial barcode in `keys` and each column represents the count of that barcode in each entry if .... Column names are set to the names of ..., if supplied.

Author(s)

Aaron Lun

Examples

```
df1 <- DataFrame(combinations=I(DataFrame(X=1:4, Y=1:4)),
   counts=sample(10, 4))

df2 <- DataFrame(combinations=I(DataFrame(X=1:4, Y=4:1)),
   counts=sample(10, 4))

df3 <- DataFrame(combinations=I(DataFrame(X=1, Y=1)),
   counts=sample(10, 1))

combineComboCounts(df1, df2, df3)
```

---

countComboBarcodes *Count combinatorial barcodes*

Description

Count combinatorial barcodes for single-end screen sequencing experiments where entities are distinguished based on random combinations of a small pool of known sequences within a single template.

Usage

```
countComboBarcodes(
  fastq,
  template,
  choices,
  substitutions = 0,
  find.best = FALSE,
  strand = c("both", "original", "reverse"),
  num.threads = 1,
  indices = FALSE
```

```
)

matrixOfComboBarcodes(files, ..., withDimnames = TRUE, BPPARAM = SerialParam())
```

Arguments

fastq	String containing the path to a FASTQ file containing single-end data, or a connection object to such a file.
template	A template for the barcode structure, see ?parseBarcodeTemplate for details.
choices	A List of character vectors, one per variable region in template. The first vector should contain the potential sequences for the first variable region, the second vector for the second variable region and so on.
substitutions	Integer scalar specifying the maximum number of substitutions when considering a match.
find.best	Logical scalar indicating whether to search each read for the best match. Defaults to stopping at the first match.
strand	String specifying which strand of the read to search.
num.threads	Integer scalar specifying the number of threads to use to process a single file.
indices	Logical scalar indicating whether integer indices should be used to define each combinational barcode.
files	A character vector of paths to FASTQ files.
...	Further arguments to pass to countComboBarcodes.
withDimnames	A logical scalar indicating whether the rows and columns should be named.
BPPARAM	A BiocParallelParam object specifying how parallelization is to be performed across files.

Details

Certain screen sequencing experiments take advantage of combinatorial complexity to generate a very large pool of unique barcode sequences. Only a subset of all possible combinatorial barcodes will be used in any given experiment. This function only counts the combinations that are actually observed, improving efficiency over a more conventional approach (i.e., to generate all possible combinations and use countSingleBarcodes to count their frequency).

If strand="both", the original read sequence will be searched first. If no match is found, the sequence is reverse-complemented and searched again. Other settings of strand will only search one or the other sequence. The most appropriate choice depends on both the sequencing protocol and the design (i.e., position and length) of the barcode.

We can handle sequencing errors by setting substitutions to a value greater than zero. This will consider substitutions in both the variable region as well as the constant flanking regions.

By default, the function will stop at the first match that satisfies the requirements above. If find.best=TRUE, we will instead try to find the best match with the fewest mismatches. If there are multiple matches with the same number of mismatches, the read is discarded to avoid problems with ambiguity.

Value

countComboBarcodes returns a DataFrame where each row corresponds to a combinatorial barcode. It contains combinations, a nested DataFrame that contains the sequences that define each combinatorial barcode; and counts, an integer vector containing the frequency of each barcode. The medata contains nreads, an integer scalar of the total number of reads in fastq.

Each column of combinations corresponds to a single variable region in template and one vector in choices. By default, the sequences are reported directly as character vectors. If indices=FALSE, each column contains the indices of the sequences in the corresponding entry of choices.

matrixOfComboBarcodes returns a SummarizedExperiment containing:

- An integer matrix named "counts", containing counts for each combinatorial barcode in each files.
- One or more vectors in the rowData that define each combinatorial barcode, equivalent to combinations.
- Column metadata containing a character vector files, the path to each file; an integer vector nreads, containing the total number of reads in each file; and nmapped, containing the number of reads assigned to a barcode in the output count matrix.

If withDimnames=TRUE, row names are set to "BARCODE_[ROW]" and column names are set to basename(files).

Author(s)

Aaron Lun

Examples

```
# Creating an example dual barcode sequencing experiment.
known.pool <- c("AGAGAGAGA", "CTCTCTCTC",
    "GTGTGTGTG", "CACACACAC")

N <- 1000
barcodes <- sprintf("ACGT%sACGT%sACGT",
   sample(known.pool, N, replace=TRUE),
   sample(known.pool, N, replace=TRUE))
names(barcodes) <- seq_len(N)

library(Biostrings)
tmp <- tempfile(fileext=".fastq")
writeXStringSet(DNAStringSet(barcodes), filepath=tmp, format="fastq")

# Counting the combinations.
output <- countComboBarcodes(tmp,
    template="ACGTNNNNNNNNNACGTNNNNNNNNNACGT",
    choices=list(first=known.pool, second=known.pool))
output$combinations
head(output$counts)

matrixOfComboBarcodes(c(tmp, tmp),
    template="ACGTNNNNNNNNNACGTNNNNNNNNNACGT",
```

```
    choices=list(first=known.pool, second=known.pool))
```

`countDualBarcodes` *Count dual barcodes*

Description

Count the frequency of dual barcodes in a dataset for a paired-end sequencing screen.

Usage

```
countDualBarcodes(
  fastq,
  choices,
  flank5,
  flank3,
  template = NULL,
  substitutions = 0,
  find.best = FALSE,
  strand = "original",
  randomized = FALSE,
  include.invalid = FALSE,
  num.threads = 1
)

matrixOfDualBarcodes(
  files,
  choices,
  ...,
  withDimnames = TRUE,
  include.invalid = FALSE,
  BPPARAM = SerialParam()
)
```

Arguments

`fastq`	Character vector of length 2, containing paths to two FASTQ files with paired-end data.
`choices`	A DataFrame with two character columns specifying valid combinations of variable regions. The first column contains sequences for barcode 1 while the second column contains sequences for barcode 2.
`flank5`	Character vector of length 2 containing the constant sequence on the 5' flank of the variable region for barcodes 1 and 2, respectively. Alternatively, a string can be supplied if the constant sequence is the same for each barcode.
`flank3`	Character vector of length 2 containing the constant sequence on the 3' flank of the variable region for barcodes 1 and 2, respectively. Alternatively, a string can be supplied if the constant sequence is the same for each barcode.

`template`	Character vector of length 2 containing the template for the structure of barcodes 1 and 2, respectively. Alternatively, a string can be supplied if the template is the same for each barcode.
`substitutions`	Integer vector of length 2 specifying how many substitutions should be allowed for barcodes 1 and 2, respectively. Alternatively, an integer scalar can be supplied if this is the same for each barcode.
`find.best`	Logical scalar indicating whether to search each read for the best match. Defaults to stopping at the first match.
`strand`	Character vector of length 2 specifying which strand of the read to search (`"original"`, `"reverse"`) for each barcode. Alternatively, a string can be supplied if this is the same for each barcode.
`randomized`	Logical scalar indicating whether the first FASTQ file always contains the first barcode in `choices`. If not, the opposite orientation is also searched.
`include.invalid`	Logical scalar indicating whether counts for invalid barcode combinations should also be returned.
`num.threads`	Integer scalar specifying the number of threads to use to process a single file.
`files`	A list of character vectors of length 2 containing paths to paired FASTQ files.
`...`	Further arguments to pass to `countDualBarcodes`.
`withDimnames`	A logical scalar indicating whether the rows and columns should be named.
`BPPARAM`	A BiocParallelParam object specifying how parallelization is to be performed across files.

Details

In a dual barcode experiment, each read of a paired-end sequencing experiment contains one barcode. The goal is to count the frequency of each combination of barcodes across the read pairs. This differs from `countComboBarcodes` in that (i) only a subset of combinations are valid and (ii) the two barcodes occur on different reads.

The interpretation of the arguments for matching each barcode to reads is similar to that of `countSingleBarcodes`. Each barcode in the combination can be associated with different search parameters; for example, the search for the "first" barcode in `choices[,1]` will be performed with `flank5[1]`, `flank3[1]`, `substitutions[1]`, `strand[1]`, etc.

By default, the first FASTQ file is assumed to contain the first barcode (i.e., `choices[,1]`) while the second file is assumed to contain the second barcode (`choices[,2]`). However, if `randomized=TRUE`, the orientation is assumed to be random such that the first FASTQ file may contain the second barcode and so on. In such cases, both orientations will be searched to identify a valid combination.

We can handle sequencing errors by setting `substitutions` to a value greater than zero. This will consider substitutions in both the variable region as well as the constant flanking regions.

By default, the function will stop at the first match that satisfies the requirements above. If `find.best=TRUE`, we will instead try to find the best match with the fewest mismatches. If there are multiple matches with the same number of mismatches, the read is discarded to avoid problems with ambiguity.

Value

By default, `countDualBarcodes` will return `choices` with an additional `counts` column. This is an integer vector of length equal to `nrow(choices)` containing the frequency of each barcode combination. The metadata contains `npairs`, the total number of read pairs processed by the function.

`matrixOfDualBarcodes` will return a SummarizedExperiment object containing:

- An integer matrix named `"counts"`, where each column is the output of `countDualBarcodes` for each file in `files`.
- Row metadata containing a character vector `choices`, the sequences of the variable region of the two barcodes for each row.
- Column metadata containing the character vectors `paths1` and `paths2`, storign the path to each pair of FASTQ files; integer vectors corresponding to the metadata described above for `countDualBarcodes`; and `nmapped`, containing the number of read pairs assigned to a barcode combination in the output count matrix.

If `withDimnames=TRUE`, row names are set to `choices` while column names are `basename(files)`.

If `include.invalid=TRUE`, each row contains all observed combinations in addition to those in `choices`. The DataFrame (or `rowData` of the SummarizedExperiment) gains a `valid` field specifying if a combination is valid, i.e., present in `choices`, The metadata also gains the following fields:

- `invalid.pair`, the number of read pairs with matches for each barcode but do not form a valid combination.
- `barcode1.only`, the number of read pairs that only match to barcode 1.
- `barcode2.only`, the number of read pairs that only match to barcode 2.

Author(s)

Aaron Lun

Examples

```
# Creating an example dual barcode sequencing experiment.
known.pool1 <- c("AGAGAGAGA", "CTCTCTCTC",
    "GTGTGTGTG", "CACACACAC")
known.pool2 <- c("ATATATATA", "CGCGCGCGC",
    "GAGAGAGAG", "CTCTCTCTC")
choices <- expand.grid(known.pool1, known.pool2)
choices <- DataFrame(barcode1=choices[,1], barcode2=choices[,2])

N <- 1000
read1 <- sprintf("CAGCTACGTACG%sCCAGCTCGATCG",
   sample(known.pool1, N, replace=TRUE))
names(read1) <- seq_len(N)

read2 <- sprintf("TGGGCAGCGACA%sACACGAGGGTAT",
   sample(known.pool2, N, replace=TRUE))
names(read2) <- seq_len(N)
```

```
library(Biostrings)
tmp <- tempfile()
tmp1 <- paste0(tmp, "_1.fastq")
writeXStringSet(DNAStringSet(read1), filepath=tmp1, format="fastq")
tmp2 <- paste0(tmp, "_2.fastq")
writeXStringSet(DNAStringSet(read2), filepath=tmp2, format="fastq")

# Counting the combinations.
countDualBarcodes(c(tmp1, tmp2), choices=choices,
    template=c("CAGCTACGTACGNNNNNNNNNCCAGCTCGATCG",
               "TGGGCAGCGACANNNNNNNNNACACGAGGGTAT"))

countDualBarcodes(c(tmp1, tmp2), choices=choices,
    flank5=c("CAGCTACGTACG", "TGGGCAGCGACA"),
    flank3=c("CCAGCTCGATCG", "ACACGAGGGTAT"))

matrixOfDualBarcodes(list(c(tmp1, tmp2), c(tmp1, tmp2)),
    choices=choices,
    flank5=c("CAGCTACGTACG", "TGGGCAGCGACA"),
    flank3=c("CCAGCTCGATCG", "ACACGAGGGTAT"))
```

---

countDualBarcodesSingleEnd *Count dual barcodes in single-end data*

Description

Count the frequency of dual barcodes in a single-end sequencing screen.

Usage

```
countDualBarcodesSingleEnd(
  fastq,
  choices,
  template,
  substitutions = 0,
  find.best = FALSE,
  strand = c("both", "original", "reverse"),
  include.invalid = FALSE,
  num.threads = 1
)

matrixOfDualBarcodesSingleEnd(
  files,
  choices,
  ...,
  withDimnames = TRUE,
  include.invalid = FALSE,
  BPPARAM = SerialParam()
)
```

Arguments

fastq	Character vector containing a path to a FASTQ file.
choices	A DataFrame with one or more character columns specifying valid combinations of variable regions. Each column contains sequences for successive barcodes in template.
template	String containing the template for the barcode structure. The number of variable regions should be equal to the number of columns of choices.
substitutions	Integer specifying how many substitutions should be allowed.
find.best	Logical scalar indicating whether to search each read for the best match. Defaults to stopping at the first match.
strand	String specifying the strand of the read to search ("original", "reverse").
include.invalid	Logical scalar indicating whether counts for invalid barcode combinations should also be returned. This is currently only enabled for template with 2 variable regions.
num.threads	Integer scalar specifying the number of threads to use to process a single file.
files	Character vectors containing paths to FASTQ files.
...	Further arguments to pass to countDualBarcodesSingleEnd.
withDimnames	A logical scalar indicating whether the rows and columns should be named.
BPPARAM	A BiocParallelParam object specifying how parallelization is to be performed across files.

Details

In a dual barcode experiment, each read of a single-end sequencing experiment contains a barcode element with multiple variable regions. The goal is to count the frequency of each combination of barcodes. However, unlike countComboBarcodes, only a subset of combinations are valid here as defined in choices.

The interpretation of the arguments for matching each barcode to reads is similar to that of countSingleBarcodes. The strand of the read to search is defined with strand, defaulting to searching both strands. We can handle sequencing errors by setting substitutions to a value greater than zero. This will consider substitutions in both the variable region as well as the constant flanking regions.

By default, the function will stop at the first match that satisfies the requirements above. If find.best=TRUE, we will instead try to find the best match with the fewest mismatches. If there are multiple matches with the same number of mismatches, the read is discarded to avoid problems with ambiguity.

Value

By default, countDualBarcodesSingleEnd will return choices with an additional counts column. This is an integer vector of length equal to nrow(choices) containing the frequency of each barcode combination. The metadata contains nreads, the total number of reads processed by the function.

matrixOfDualBarcodesSingleEnd will return a SummarizedExperiment object containing:

- An integer matrix named "counts", where each column is the output of countDualBarcodes for each file in files.

- Row metadata containing a character vector `choices`, the sequences of the variable region of the two barcodes for each row.
- Column metadata containing a character vector `paths`, the path to each FASTQ file; and integer vectors corresponding to the metadata described above for `countDualBarcodesSingleEnd`.

If `withDimnames=TRUE`, row names are set to `choices` while column names are `basename(files)`.

If `include.invalid=TRUE`, each row contains all observed combinations in addition to those in `choices`. The DataFrame (or `rowData` of the SummarizedExperiment) gains a `valid` field specifying if a combination is valid, i.e., present in `choices`. The metadata also gains the `invalid.reads` field, containing the number of reads with matches for each barcode but do not form a valid combination.

Author(s)

Aaron Lun

Examples

```
# Creating an example dual barcode sequencing experiment.
known.pool1 <- c("AGAGAGAGA", "CTCTCTCTC",
    "GTGTGTGTG", "CACACACAC")
known.pool2 <- c("ATATATATA", "CGCGCGCGC",
    "GAGAGAGAG", "CTCTCTCTC")
choices <- expand.grid(known.pool1, known.pool2)
choices <- DataFrame(barcode1=choices[,1], barcode2=choices[,2])

N <- 1000
read <- sprintf(
   "CAGCTACGTACG%sCCAGCTCGATCG%sACACGAGGGTAT",
   sample(known.pool1, N, replace=TRUE),
   sample(known.pool2, N, replace=TRUE)
)
names(read) <- seq_len(N)

library(Biostrings)
tmp <- tempfile(fileext=".fastq")
writeXStringSet(DNAStringSet(read), filepath=tmp, format="fastq")

# Counting the combinations.
countDualBarcodesSingleEnd(tmp, choices=choices,
    template="CAGCTACGTACGNNNNNNNNNCCAGCTCGATCGNNNNNNNNNACACGAGGGTAT")

matrixOfDualBarcodesSingleEnd(c(tmp, tmp),
    choices=choices,
    template="CAGCTACGTACGNNNNNNNNNCCAGCTCGATCGNNNNNNNNNACACGAGGGTAT")
```

countPairedComboBarcodes *Count paired-end combinatorial barcodes*

Description

Count combinatorial barcodes for paired-end screen sequencing experiments where entities are distinguished based on random combinations of a small pool of known sequences across two templates.

Usage

```
countPairedComboBarcodes(
  fastq,
  choices,
  flank5,
  flank3,
  template = NULL,
  substitutions = 0,
  find.best = FALSE,
  strand = "original",
  num.threads = 1,
  randomized = FALSE,
  indices = FALSE
)

matrixOfPairedComboBarcodes(
  files,
  ...,
  withDimnames = TRUE,
  BPPARAM = SerialParam()
)
```

Arguments

fastq	Character vector of length 2, containing paths to two FASTQ files with paired-end data.
choices	A List of two character vectors. The first vector should contain the potential sequences for the variable region in the first template, the second vector for the variable region in the second template.
flank5	Character vector of length 2 containing the constant sequence on the 5' flank of the variable region for barcodes 1 and 2, respectively. Alternatively, a string can be supplied if the constant sequence is the same for each barcode.
flank3	Character vector of length 2 containing the constant sequence on the 3' flank of the variable region for barcodes 1 and 2, respectively. Alternatively, a string can be supplied if the constant sequence is the same for each barcode.

template	Character vector of length 2 containing the template for the structure of barcodes 1 and 2, respectively. Alternatively, a string can be supplied if the template is the same for each barcode.
substitutions	Integer vector of length 2 specifying how many substitutions should be allowed for barcodes 1 and 2, respectively. Alternatively, an integer scalar can be supplied if this is the same for each barcode.
find.best	Logical scalar indicating whether to search each read for the best match. Defaults to stopping at the first match.
strand	Character vector of length 2 specifying which strand of the read to search ("original", "reverse") for each barcode. Alternatively, a string can be supplied if this is the same for each barcode.
num.threads	Integer scalar specifying the number of threads to use to process a single file.
randomized	Logical scalar indicating whether the first FASTQ file always contains the first barcode in choices. If not, the opposite orientation is also searched.
indices	Logical scalar indicating whether integer indices should be used to define each combinational barcode.
files	A list of character vectors of length 2 containing paths to paired FASTQ files.
...	Further arguments to pass to countDualBarcodes.
withDimnames	A logical scalar indicating whether the rows and columns should be named.
BPPARAM	A BiocParallelParam object specifying how parallelization is to be performed across files.

Details

Here, we consider a barcode design very similar to that of countComboBarcodes, except that the two variable regions are present on different reads rather than occurring within a single template on the same read. This function counts the frequency of these barcode combinations across the two reads.

The interpretation of the arguments for matching each barcode to reads is similar to that of countSingleBarcodes. Each barcode in the combination can be associated with different search parameters; for example, the search for the "first" barcode in choices[[1]] will be performed with flank5[1], flank3[1], substitutions[1], strand[1], etc.

By default, the first FASTQ file is assumed to contain the first barcode (i.e., choices[[1]]) while the second file is assumed to contain the second barcode (choices[[2]]). However, if randomized=TRUE, the orientation is assumed to be random such that the first FASTQ file may contain the second barcode and so on. In such cases, both orientations will be searched to identify the combination. This is most relevant when the constant regions are different between the two reads, otherwise either orientation could be valid.

We can handle sequencing errors by setting substitutions to a value greater than zero. This will consider substitutions in both the variable region as well as the constant flanking regions for each read.

By default, the function will stop at the first match that satisfies the requirements above. If find.best=TRUE, we will instead try to find the best match with the fewest mismatches. If there are multiple matches with the same number of mismatches, the read is discarded to avoid problems with ambiguity.

Value

countPairedComboBarcodes returns a DataFrame where each row corresponds to a combinatorial barcode. It contains combinations, a nested DataFrame that contains the sequences that define each combinatorial barcode; and counts, an integer vector containing the frequency of each barcode. The medata contains:

- npairs, the total number of read pairs processed by the function.
- barcode1.only, the number of read pairs that only match to barcode 1.
- barcode2.only, the number of read pairs that only match to barcode 2.

Each column of combinations corresponds to a single variable region in template and one vector in choices. By default, the sequences are reported directly as character vectors. If indices=FALSE, each column contains the indices of the sequences in the corresponding entry of choices.

matrixOfPairedComboBarcodes returns a SummarizedExperiment containing:

- An integer matrix named "counts", containing counts for each combinatorial barcode in each files.
- One or more vectors in the rowData that define each combinatorial barcode, equivalent to combinations.
- Column metadata containing a character vector files, the path to each file; an integer vector nreads, containing the total number of reads in each file; and nmapped, containing the number of reads assigned to a barcode in the output count matrix.

Author(s)

Aaron Lun

Examples

```
# Creating an example dual barcode sequencing experiment.
known.pool1 <- c("AGAGAGAGA", "CTCTCTCTC", "GTGTGTGTG", "CACACACAC")
known.pool2 <- c("ATATATATA", "CGCGCGCGC", "GAGAGAGAG", "CTCTCTCTC")
choices <- list(barcode1=known.pool1, barcode2=known.pool2)

N <- 1000
read1 <- sprintf("CAGCTACGTACG%sCCAGCTCGATCG", sample(known.pool1, N, replace=TRUE))
names(read1) <- seq_len(N)

read2 <- sprintf("TGGGCAGCGACA%sACACGAGGGTAT", sample(known.pool2, N, replace=TRUE))
names(read2) <- seq_len(N)

library(Biostrings)
tmp <- tempfile()
tmp1 <- paste0(tmp, "_1.fastq")
writeXStringSet(DNAStringSet(read1), filepath=tmp1, format="fastq")
tmp2 <- paste0(tmp, "_2.fastq")
writeXStringSet(DNAStringSet(read2), filepath=tmp2, format="fastq")

# Counting the combinations.
countPairedComboBarcodes(c(tmp1, tmp2), choices=choices,
```

```
    template=c("CAGCTACGTACGNNNNNNNNNCCAGCTCGATCG",
               "TGGGCAGCGACANNNNNNNNNNACACGAGGGTAT"))

matrixOfPairedComboBarcodes(list(c(tmp1, tmp2)),
    template=c("CAGCTACGTACGNNNNNNNNNCCAGCTCGATCG",
               "TGGGCAGCGACANNNNNNNNNNACACGAGGGTAT"),
    choices=list(first=known.pool1, second=known.pool2))
```

---

countRandomBarcodes *Count random barcodes*

Description

Count the frequency of random barcodes in a FASTQ file containing data for a single-end sequencing screen. This differs from countSingleBarcodes in that the barcode is completely random rather than being drawn from a known pool of sequences.

Usage

```
countRandomBarcodes(
  fastq,
  template,
  substitutions = 0,
  find.best = FALSE,
  strand = c("both", "original", "reverse"),
  num.threads = 1
)

matrixOfRandomBarcodes(
  files,
  ...,
  withDimnames = TRUE,
  BPPARAM = SerialParam()
)
```

Arguments

fastq	String containing the path to a FASTQ file containing single-end data.
template	String containing the template for the barcode structure. See parseBarcodeTemplate for more details.
substitutions	Integer scalar specifying the maximum number of substitutions when considering a match.
find.best	Logical scalar indicating whether to search each read for the best match. Defaults to stopping at the first match.
strand	String specifying which strand of the read to search.

num.threads	Integer scalar specifying the number of threads to use to process a single file.
files	A character vector of paths to FASTQ files.
...	Further arguments to pass to countSingleBarcodes.
withDimnames	A logical scalar indicating whether the rows and columns should be named.
BPPARAM	A BiocParallelParam object specifying how parallelization is to be performed across files.

Details

If strand="both", the original read sequence will be searched first. If no match is found, the sequence is reverse-complemented and searched again. Other settings of strand will only search one or the other sequence. The most appropriate choice depends on both the sequencing protocol and the design (i.e., position and length) of the barcode.

We can handle sequencing errors by setting substitutions to a value greater than zero. This will consider substitutions in both the variable region as well as the constant flanking regions.

By default, the function will stop at the first match that satisfies the requirements above. If find.best=TRUE, we will instead try to find the best match with the fewest mismatches. If there are multiple matches with the same number of mismatches, the read is discarded to avoid problems with ambiguity.

Value

countRandomBarcodes will return a DataFrame containing:

- sequences, a character vector containing the sequences of the random barcodes in the variable region.
- counts, an integer vector containing the frequency of each barcode.

The metadata contains nreads, an integer scalar containing the total number of reads in fastq.

matrixOfRandomBarcodes will return a SummarizedExperiment object containing:

- An integer matrix named "counts", where each column is the output of countRandomBarcodes for each file in files.
- Row metadata containing a character vector sequences, the sequence of the variable region of each barcode for each row.
- Column metadata containing a character vector files, the path to each file; an integer vector nreads, containing the total number of reads in each file; and nmapped, containing the number of reads assigned to a barcode in the output count matrix.

If withDimnames=TRUE, row names are set to sequences while column names are basename(files).

Author(s)

Aaron Lun

Examples

```
# Creating an example dataset.
N <- 1000
randomized <- lapply(1:N, function(i) {
    paste(sample(c("A", "C", "G", "T"), 8, replace=TRUE), collapse="")
})
barcodes <- sprintf("CAGCTACGTACG%sCCAGCTCGATCG", randomized)
names(barcodes) <- seq_len(N)

library(Biostrings)
tmp <- tempfile(fileext=".fastq")
writeXStringSet(DNAStringSet(barcodes), filepath=tmp, format="fastq")

# Counting the sequences:
countRandomBarcodes(tmp, template="CAGCTACGTACGNNNNNNNNCCAGCTCGATCG")

matrixOfRandomBarcodes(c(tmp, tmp), template="CAGCTACGTACGNNNNNNNNCCAGCTCGATCG")
```

---

countSingleBarcodes *Count single barcodes*

Description

Count the frequency of barcodes in a FASTQ file containing data for a single-end sequencing screen.

Usage

```
countSingleBarcodes(
  fastq,
  choices,
  flank5,
  flank3,
  template = NULL,
  substitutions = 0,
  find.best = FALSE,
  strand = c("both", "original", "reverse"),
  num.threads = 1
)

matrixOfSingleBarcodes(
  files,
  choices,
  ...,
  withDimnames = TRUE,
  BPPARAM = SerialParam()
)
```

Arguments

fastq	String containing the path to a FASTQ file containing single-end data.
choices	A character vector of sequences for the variable regions, one per barcode.
flank5	String containing the constant sequence on the 5' flank of the variable region.
flank3	String containing the constant sequence on the 3' flank of the variable region.
template	String containing the template for the barcode structure.
substitutions	Integer scalar specifying the maximum number of substitutions when considering a match.
find.best	Logical scalar indicating whether to search each read for the best match. Defaults to stopping at the first match.
strand	String specifying which strand of the read to search.
num.threads	Integer scalar specifying the number of threads to use to process a single file.
files	A character vector of paths to FASTQ files.
...	Further arguments to pass to countSingleBarcodes.
withDimnames	A logical scalar indicating whether the rows and columns should be named.
BPPARAM	A BiocParallelParam object specifying how parallelization is to be performed across files.

Details

If template is specified, it will be used to define the flanking regions. Any user-supplied values of flank5 and flank3 will be ignored. Note that, for this function, the template should only contain a single variable region. See parseBarcodeTemplate for more details.

If strand="both", the original read sequence will be searched first. If no match is found, the sequence is reverse-complemented and searched again. Other settings of strand will only search one or the other sequence. The most appropriate choice depends on both the sequencing protocol and the design (i.e., position and length) of the barcode.

We can handle sequencing errors by setting substitutions to a value greater than zero. This will consider substitutions in both the variable region as well as the constant flanking regions.

By default, the function will stop at the first match that satisfies the requirements above. If find.best=TRUE, we will instead try to find the best match with the fewest mismatches. If there are multiple matches with the same number of mismatches, the read is discarded to avoid problems with ambiguity.

Value

countSingleBarcodes will return a DataFrame containing:

- choices, a character vector equal to the input choices.
- counts, an integer vector of length equal to nrow(choices) containing the frequency of each barcode.

The metadata contains nreads, an integer scalar containing the total number of reads in fastq.

matrixOfSingleBarcodes will return a SummarizedExperiment object containing:

- An integer matrix named "counts", where each column is the output of countSingleBarcodes for each file in files.
- Row metadata containing a character vector choices, the sequence of the variable region of each barcode for each row.
- Column metadata containing a character vector files, the path to each file; an integer vector nreads, containing the total number of reads in each file; and nmapped, containing the number of reads assigned to a barcode in the output count matrix.

If withDimnames=TRUE, row names are set to choices while column names are basename(files).

Author(s)

Aaron Lun

Examples

```
# Creating an example dual barcode sequencing experiment.
known.pool <- c("AGAGAGAGA", "CTCTCTCTC",
    "GTGTGTGTG", "CACACACAC")

N <- 1000
barcodes <- sprintf("CAGCTACGTACG%sCCAGCTCGATCG",
   sample(known.pool, N, replace=TRUE))
names(barcodes) <- seq_len(N)

library(Biostrings)
tmp <- tempfile(fileext=".fastq")
writeXStringSet(DNAStringSet(barcodes), filepath=tmp, format="fastq")

# Counting the combinations.
countSingleBarcodes(tmp, choices=known.pool,
    template="CAGCTACGTACGNNNNNNNNNCCAGCTCGATCG")

countSingleBarcodes(tmp, choices=known.pool,
    flank5="CAGCTACGTACG", flank3="CCAGCTCGATCG")

matrixOfSingleBarcodes(c(tmp, tmp), choices=known.pool,
    flank5="CAGCTACGTACG", flank3="CCAGCTCGATCG")
```

---

matchBarcodes *Match sequences to a pool of barcodes*

Description

Pretty much what it says on the tin. Useful for matching observed sequences (e.g., from countRandomBarcodes) to a pool of known barcode sequences, accounting for substitutions and ambiguous IUPAC codes.

Usage

```
matchBarcodes(sequences, choices, substitutions = 0, reverse = FALSE)
```

Arguments

sequences	Character vector of observed sequences.
choices	Character vector of barcode sequences.
substitutions	Integer scalar specifying the maximum number of substitutions when considering a match.
reverse	Whether to match sequences to the reverse complement of choices.

Value

DataFrame with one row per entry of sequences, containing the following fields:

- index, the index of the matching barcode in choices. This is set to NA if no unambiguous match is found.
- mismatches, the number of mismatching bases with the assigned barcode. This is set to NA if index is NA.

Author(s)

Aaron Lun

Examples

```
choices <- c("AAAAAA", "CCCCCC", "GGGGGG", "TTTTTT")
matchBarcodes(c("AAAAAA", "AAATAA"), choices)
matchBarcodes(c("AAAAAA", "AAATAA"), choices, substitutions=1)
matchBarcodes(c("AAAAAA", "AAATAA"), choices, reverse=TRUE)

# Works with IUPAC codes in the barcodes:
choices <- c("AAARAA", "CCCYCC", "GGGMGG", "TTTSTT")
matchBarcodes(c("AAAAAA", "AAAGAA"), choices)
```

---

parseBarcodeTemplate *Parse barcode template*

Description

Parse a barcode template to identify variable regions based on the run of N's.

Usage

```
parseBarcodeTemplate(template)
```

Arguments

template	String containing template sequence of a barcode. Variable regions should be marked with N's.

Details

The barcode template should contain runs of N's to mark the variable regions. The first run of N's is the first variable region, the second run of N's is the second variable region, and so on. The template is "realized" into a barcode when the N's are replaced with actual DNA sequence. The use of a template provides a convenient format to express the general structure of the barcode while avoiding confusion about barcode-specific variable regions.

Value

A list containing:

- `variable`, a DataFrame containing the position and length of each run of N's.
- `constant`, a character vector of constant regions flanking and separating the variable regions.

Author(s)

Aaron Lun

Examples

```
# Single spacer:
parseBarcodeTemplate("AAAANNNNNNNGGGG")

# Double spacer:
parseBarcodeTemplate("AAAANNNNCCCCNNGGGG")
```

Index

BiocParallelParam, *4*, *7*, *10*, *13*, *16*, *18*

combineComboCounts, 2
countComboBarcodes, *2*, 3, *7*, *10*, *13*
countDualBarcodes, 6
countDualBarcodesSingleEnd, 9
countPairedComboBarcodes, 12
countRandomBarcodes, 15, *19*
countSingleBarcodes, *4*, *7*, *10*, *13*, *15*, 17

DataFrame, *2, 3*, *5, 6*, *10*, *14*, *16*, *18*, *20, 21*

List, *4*, *12*

matchBarcodes, 19
matrixOfComboBarcodes
        *(*countComboBarcodes*)*, 3
matrixOfDualBarcodes
        *(*countDualBarcodes*)*, 6
matrixOfDualBarcodesSingleEnd
        *(*countDualBarcodesSingleEnd*)*, 9
matrixOfPairedComboBarcodes
        *(*countPairedComboBarcodes*)*, 12
matrixOfRandomBarcodes
        *(*countRandomBarcodes*)*, 15
matrixOfSingleBarcodes
        *(*countSingleBarcodes*)*, 17

parseBarcodeTemplate, *4*, *15*, *18*, 20

rowData, *8*, *11*

screenCounter-package, 2
SummarizedExperiment, *2*, *5*, *8*, *10*, *14*, *16*, *18*