Package 'CellBarcode'

October 11, 2022

Type Package

Title Cellular DNA Barcode Analysis toolkit

Version 1.2.0

Description This package performs Cellular DNA Barcode (genetic lineage tracing) analysis. The package can handle all kinds of DNA barcodes, as long as the barcode within a single sequencing read and has a pattern which can be matched by a regular expression. This package can handle barcode with flexible length, with or without UMI (unique molecular identifier). This tool also can be used for pre-processing of some amplicon data such as CRISPR gRNA screening, immune repertoire sequencing and meta genome data.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports methods, stats, Rcpp (>= 1.0.5), data.table (>= 1.12.6), plyr, ggplot2, stringr, magrittr, ShortRead (>= 1.48.0), Biostrings (>= 2.58.0), egg, Ckmeans.1d.dp, utils, S4Vectors

LinkingTo Rcpp

RoxygenNote 7.1.1

Suggests BiocStyle, testthat (>= 3.0.0), knitr, rmarkdown

biocViews Preprocessing, QualityControl, Sequencing, CRISPR

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

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

git_branch RELEASE_3_15

git_last_commit bdde81f

git_last_commit_date 2022-04-26

Date/Publication 2022-10-11

Author Wenjie Sun [cre],
Anne-Marie Lyne [aut],
Leila Perie [aut]

Maintainer Wenjie Sun <sunwjie@gmail.com>

R topics documented:

`BarcodeObj-class` *BarcodeObj object*

Description

A S4 object holds the barcode data and samples' metadata. A set of operations can be applied to the BarcodeObj object for quality control and selecting barcodes/samples subset.

Details

The BarcodeObj object is a S4 object, it has three slots, which can be access by "@" operator, they are `messyBc`, `cleanBc` and `metadata`. A `BarcodeObj` object can be generated by `bc_extract` function. The `bc_extract` function can use various data types as input, such as data.frame, fastq files or ShortReadQ.

Slot messyBc is a list holds the raw barcodes sequence before filtering, where each element is a data.table corresponding to the successive samples. Each table has 5 columns: 1. reads_seq: full read sequence before parsing. 2. match_seq: the sequence matched by pattern given to bc_extract. 3. umi_seq (optional): UMI sequence. 4. barcode_seq: barcode sequence. 5. count: how many reads a full sequence has. In this table, barcode_seq value can be duplicated, as two different full read sequences can contain the same barcode sequence, due to the diversity of the UMI or mutations in the constant region.

Slot cleanBc is a list holds the barcodes sequence after applying filtering, where each element is a data.table corresponding to the successive samples. The "cleanBc" slot contains 2 columns 1. barcode_seq: barcode sequence 2. counts: reads count, or UMI count if the cleanBc was created by bc_cure_umi.

Examples

```
#######
# Create BarcodeObj with fastq file
fq_file <- system.file("extdata", "simple.fq", package="CellBarcode")
library(ShortRead)
bc_extract(fq_file, pattern = "AAAAA(.*)CCCCC")

#######
# data manipulation on BarcodeObj object
data(bc_obj)

bc_obj

# Select barcodes
bc_subset(bc_obj, barcode = c("AACCTT", "AACCTT"))
bc_obj[c("AGAG", "AAAG"), ]

# Select samples by meta data
bc_meta(bc_obj)$phenotype <- c("l", "b")
bc_meta(bc_obj)
bc_subset(bc_obj, sample = phenotype == "l")

# Select samples by sample name
bc_obj[, "test1"]
bc_obj[, c("test1", "test2")]
bc_subset(bc_obj, sample = "test1", barcode = c("AACCTT", "AACCTT"))

# Apply barcodes black list
bc_subset(
bc_obj,
    sample = c("test1", "test2"),
    barcode = c("AACCTT"))

# Join two samples with no barcodes overlap
bc_obj["AGAG", "test1"] + bc_obj["AAAG", "test2"]

# Join two samples with barcodes overlap
bc_obj_join <- bc_obj["AGAG", "test1"] + bc_obj["AGAG", "test2"]
bc_obj_join
```

```
# The same barcode will be merged after applying bc_cure_depth()
bc_cure_depth(bc_obj_join)

# Remove barcodes
bc_obj
bc_obj - "AAAG"

# Select barcodes in a white list
bc_obj
bc_obj * "AAAG"
###
```

---

bc_2df *Transforms BarcodeObj object into other data type*

Description

Transforms BarcodeObj object into `data.frame`, `data.table` or `matrix`.

Usage

```
bc_2df(barcodeObj)

bc_2dt(barcodeObj)

bc_2matrix(barcodeObj)

## S4 method for signature 'BarcodeObj'
bc_2df(barcodeObj)

## S4 method for signature 'BarcodeObj'
bc_2dt(barcodeObj)

## S4 method for signature 'BarcodeObj'
bc_2matrix(barcodeObj)
```

Arguments

`barcodeObj`	A `BarcodeObj` object.

Value

A `data.frame`, with two columns: `barcode_seq` and `count`.

Examples

```
data(bc_obj)

bc_obj <- bc_cure_depth(bc_obj)

# BarcodeObj to data.frame
bc_2df(bc_obj)

# BarcodeObj to data.table
bc_2dt(bc_obj)

# BarcodeObj to matrix
bc_2matrix(bc_obj)

###
```

`bc_auto_cutoff` *Finds barcode count cutoff point*

Description

Finds the cutoff point for the barcode count filtering based on the barcode count distribution.

Usage

```
bc_auto_cutoff(barcodeObj, useCleanBc = TRUE)

## S4 method for signature 'BarcodeObj'
bc_auto_cutoff(barcodeObj, useCleanBc = TRUE)
```

Arguments

`barcodeObj`	A `BarcodeObj` object.
`useCleanBc`	A logical value, if `TRUE`, the `cleanBc` slot in the `BarcodeObj` object will be used, otherwise the `messyBc` slot will be used.

Details

The one dimension kmeans clustering is applied for identify the "true barcode" based on read count. The the algorithm detail is: 1. Remove the barcodes with count below the median of counts. 2. Transform the count by log2(x+1). 3. Apply the 1 dimension clustering to the logarized count, with the cluster number of 2 and weights of the logarized count. 4. Choose the minimum count value in the cluster with higher count as cutoff point.

For more info about 1 dimension kmeans used here please refer to `Ckmeans.1d.dp`, which has been used here.

Value

a numeric vector of the cutoff point.

Examples

```
data(bc_obj)

bc_auto_cutoff(bc_obj)
```

---

bc_barcodes *Gets barcode sequences*

Description

bc_barcodes used to get the barcode sequences in BarcodeObj object. The input BarcodesObj object should be pre-processed by bc_cure_* functions, such as bc_cure_depth, bc_cure_umi.

Usage

```
bc_barcodes(barcodeObj, unlist = TRUE)

## S4 method for signature 'BarcodeObj'
bc_barcodes(barcodeObj, unlist = TRUE)
```

Arguments

barcodeObj	A BarcodeObj object.
unlist	A logical value. If TRUE, the function returns a vector of unique barcode list from all samples; otherwise a list will be returned. In the later case, each element of the list contains the barcodes of a sample.

Value

A character vector or a list.

Examples

```
data(bc_obj)

# get unique barcode vector of all samples
bc_barcodes(bc_obj)

# get a list with each element containing barcodes from one sample
bc_barcodes(bc_obj, unlist = FALSE)

###
```

bc_cleanBc *Accesses cleanBc slot in the BarcodeObj object*

Description

cleanBc slot of BarcodeObj object contains the processed barcode reads frequency data. For more detail about the cleanBc slot, see BarcodeObj. bc_cleanBc is used to access the 'cleanBc' slot in the BarcodeObj.

Usage

```
bc_cleanBc(barcodeObj, isList = TRUE)

## S4 method for signature 'BarcodeObj'
bc_cleanBc(barcodeObj, isList = TRUE)
```

Arguments

barcodeObj	A BarcodeObj objects.
isList	A logical value, if TRUE (default), the return is a list with each sample as an element. Otherwise, the function will return a data.frame contains the data from all the samples with a column named sample_name to keep the sample information.

Value

If a list is requested, each list element a codedata.frame for each sample. In a codedata.frame, there are 2 columns 1. barcode_seq: barcode sequence 2. counts: reads count, or UMI count if the cleanBc was created by bc_cure_umi.

If a data.frame is requested, the data.frame in the list described above are combined into one data.frame by row, with an extra column named sample_name for identifying sample.

Examples

```
 data(bc_obj)
# get the data in cleanBc slot
# default the return value is a list
bc_cleanBc(bc_obj)

# the return value can be a data.frame
bc_cleanBc(bc_obj, isList=FALSE)
###
```

bc_cure_cluster *Merges barcodes by editing distance*

Description

bc_cure_cluster performs clustering of barcodes by editing distance, and merging the barcodes with similar sequence. This function is only applicable for the BarcodeObj object with a cleanBc slot

Usage

```
bc_cure_cluster(
  barcodeObj,
  dist_thresh = 1,
  dist_method = "hamm",
  merge_method = "greedy",
  count_threshold = 1000,
  dist_costs = list(replace = 1, insert = 1, delete = 1)
)

## S4 method for signature 'BarcodeObj'
bc_cure_cluster(
  barcodeObj,
  dist_thresh = 1,
  dist_method = "hamm",
  merge_method = "greedy",
  count_threshold = 1000,
  dist_costs = list(replace = 1, insert = 1, delete = 1)
)
```

Arguments

barcodeObj	A BarcodeObj object.
dist_thresh	A single integer or vector of integers with the length of sample count, specifying the editing distance threshold of merging two similar barcode sequences. If the input is a vector, each value in the vector relates to one sample according to the sample order in BarcodeObj object.
dist_method	A character string, specifying the distance algorithm used for evaluating barcodes similarity. It can be "hamm" for Hamming distance or "leven" for Levenshtein distance.
merge_method	A character string specifying the algorithm used to perform the clustering merging of barcodes. Currently only "greedy" is available, in this case, the least abundant barcode is preferentially merged to the most abundant ones.
count_threshold	An integer, read depth threshold to consider a barcode as a true barcode, when when a barcode with count higher than this threshold it will not be merged into more abundant barcode.

dist_costs A list, the cost of the events of distance algorithm, applicable when Levenshtein distance is applied. The names of vector have to be `insert`, `delete` and `replace`, specifying the weight of insertion, deletion, replacement events respectively. The default cost for each event is 1.

Value

A BarcodeObj object with cleanBc slot updated.

Examples

```
data(bc_obj)

d1 <- data.frame(
    seq = c(
        "ACTTCGATCGATCGAAAAGATCGATCGATC",
        "AATTCGATCGATCGAAGAGATCGATCGATC",
        "CCTTCGATCGATCGAAGAAGATCGATCGATC",
        "TTTTCGATCGATCGAAAAGATCGATCGATC",
        "AAATCGATCGATCGAAGAGATCGATCGATC",
        "CCCTCGATCGATCGAAGAAGATCGATCGATC",
        "GGGTCGATCGATCGAAAAGATCGATCGATC",
        "GGATCGATCGATCGAAGAGATCGATCGATC",
        "ACTTCGATCGATCGAACAAGATCGATCGATC",
        "GGTTCGATCGATCGACGAGATCGATCGATC",
        "GCGTCCATCGATCGAAGAAGATCGATCGATC"
        ),
    freq = c(
        30, 60, 9, 10, 14, 5, 10, 30, 6, 4 , 6
        )
    )

pattern <- "([ACTG]{3})TCGATCGATCGA([ACTG]+)ATCGATCGATC"
bc_obj <- bc_extract(list(test = d1), pattern, sample_name=c("test"),
    pattern_type=c(UMI=1, barcode=2))

# Remove barcodes with depth < 5
(bc_cured <- bc_cure_depth(bc_obj, depth=5))

# Do the clustering, merge the less abundent barcodes to the more abundent
# one by hamming distance <= 1
bc_cure_cluster(bc_cured, dist_thresh = 1)

# Levenshtein distance <= 1
bc_cure_cluster(bc_cured, dist_thresh = 2, dist_method = "leven",
    dist_costs = list("insert" = 2, "replace" = 1, "delete" = 2))

###
```

bc_cure_depth *Filters barcodes by counts*

Description

bc_cure_depth filters barcodes by the read counts or the UMI counts.

Usage

```
bc_cure_depth(barcodeObj, depth = 0, isUpdate = TRUE)

## S4 method for signature 'BarcodeObj'
bc_cure_depth(barcodeObj, depth = 0, isUpdate = TRUE)
```

Arguments

barcodeObj	A BarcodeObj object.
depth	A numeric or a vector of numeric, specifying the threshold of minimum count for a barcode to kept. If the input is a vector, if the vector length is not the same to the sample number the element will be repeatedly used. And when the depth argument is a number with negative value, automatic cutoff point will be chosen by bc_auto_cutoff function for each samples. See bc_auto_cutoff for details.
isUpdate	A logical value. If TRUE, the cleanBc slot in BarcodeObj will be used preferentially, otherwise the messyBc slot will be used. If no cleanBc is available, messyBc will be used instead.

Value

A BarcodeObj object with cleanBc slot updated or created.

Examples

```
data(bc_obj)

d1 <- data.frame(
    seq = c(
        "ACTTCGATCGATCGAAAAGATCGATCGATC",
        "AATTCGATCGATCGAAGAGATCGATCGATC",
        "CCTTCGATCGATCGAAGAAGATCGATCGATC",
        "TTTTCGATCGATCGAAAAGATCGATCGATC",
        "AAATCGATCGATCGAAGAGATCGATCGATC",
        "CCCTCGATCGATCGAAGAAGATCGATCGATC",
        "GGGTCGATCGATCGAAAAGATCGATCGATC",
        "GGATCGATCGATCGAAGAGATCGATCGATC",
        "ACTTCGATCGATCGAACAAGATCGATCGATC",
        "GGTTCGATCGATCGACGAGATCGATCGATC",
        "GCGTCCATCGATCGAAGAAGATCGATCGATC"
```

```
        ),
    freq = c(
        30, 60, 9, 10, 14, 5, 10, 30, 6, 4 , 6
        )
    )

pattern <- "([ACTG]{3})TCGATCGATCGA([ACTG]+)ATCGATCGATC"
bc_obj <- bc_extract(list(test = d1), pattern, sample_name=c("test"),
    pattern_type=c(UMI=1, barcode=2))

# Remove barcodes with depth < 5
(bc_cured <- bc_cure_depth(bc_obj, depth=5))
bc_2matrix(bc_cured)

# Use UMI information, filter the barcode < 5 UMI
bc_umi_cured <- bc_cure_umi(bc_obj, depth =0, doFish=TRUE, isUniqueUMI=TRUE)
bc_cure_depth(bc_umi_cured, depth = 5)

###
```

---

bc_cure_umi *Filters UMI-barcode tag by counts*

Description

When the UMI is applied, bc_cure_umi can filter the UMI-barcode tags by counts.

Usage

```
bc_cure_umi(barcodeObj, depth = 2, doFish = FALSE, isUniqueUMI = FALSE)

## S4 method for signature 'BarcodeObj'
bc_cure_umi(barcodeObj, depth = 2, doFish = FALSE, isUniqueUMI = FALSE)
```

Arguments

barcodeObj	A BarcodeObj object.
depth	A numeric or a vector of numeric, specifying the UMI-barcode tag count threshold. Only the barcodes with UMI-barcode tag count larger than the threshold are kept.
doFish	A logical value, if true, for barcodes with UMI read depth above the threshold, "fish" for identical barcodes with UMI read depth below the threshold. The consequence of doFish will not increase the number of identified barcodes, but the UMI counts will increase due to including the low depth UMI barcodes.
isUniqueUMI	A logical value, In the case that a UMI relates to several barcodes, if you believe that the UMI is absolute unique, then only the UMI-barcodes tags with highest count are chosen for each UMI.

Details

When invoke this function, it processes the data with following steps:

1. (if isUniqueUMI is TRUE) Find dominant sequence in each UMI.
2. UMI-barcode depth filtering.
3. (if doFish is TRUE) Fishing the UMI with low UMI-barcode depth.

Value

A `BarcodeObj` object with `cleanBc` slot updated (or created).

Examples

```
data(bc_obj)

d1 <- data.frame(
   seq = c(
       "ACTTCGATCGATCGAAAAGATCGATCGATC",
       "AATTCGATCGATCGAAGAGATCGATCGATC",
       "CCTTCGATCGATCGAAGAAGATCGATCGATC",
       "TTTTCGATCGATCGAAAAGATCGATCGATC",
       "AAATCGATCGATCGAAGAGATCGATCGATC",
       "CCCTCGATCGATCGAAGAAGATCGATCGATC",
       "GGGTCGATCGATCGAAAAGATCGATCGATC",
       "GGATCGATCGATCGAAGAGATCGATCGATC",
       "ACTTCGATCGATCGAACAAGATCGATCGATC",
       "GGTTCGATCGATCGACGAGATCGATCGATC",
       "GCGTCCATCGATCGAAGAAGATCGATCGATC"
       ),
   freq = c(
       30, 60, 9, 10, 14, 5, 10, 30, 6, 4 , 6
       )
   )

pattern <- "([ACTG]{3})TCGATCGATCGA([ACTG]+)ATCGATCGATC"
bc_obj <- bc_extract(list(test = d1), pattern, sample_name=c("test"),
    pattern_type=c(UMI=1, barcode=2))

# Use UMI information to remove the barcode < 5 UMI-barcode tags
bc_umi_cured <- bc_cure_umi(bc_obj, depth =0, doFish=TRUE, isUniqueUMI=TRUE)
bc_cure_depth(bc_umi_cured, depth = 5)
```

`bc_extract` *Extract barcode from sequences*

Description

bc_extract identifies the barcodes (and UMI) from the sequences using regular expressions. pattern and pattern_type arguments are necessary, which provide the barcode (and UMI) pattern and their location within the sequences.

Usage

```
bc_extract(
  x,
  pattern = "",
  sample_name = NULL,
  metadata = NULL,
  maxLDist = 0,
  pattern_type = c(barcode = 1),
  costs = list(sub = 1, ins = 99, del = 99),
  ordered = TRUE
)

## S4 method for signature 'data.frame'
bc_extract(
  x,
  pattern = "",
  sample_name = NULL,
  maxLDist = 0,
  pattern_type = c(barcode = 1),
  costs = list(sub = 1, ins = 99, del = 99),
  ordered = TRUE
)

## S4 method for signature 'ShortReadQ'
bc_extract(
  x,
  pattern = "",
  sample_name = NULL,
  maxLDist = 0,
  pattern_type = c(barcode = 1),
  costs = list(sub = 1, ins = 99, del = 99),
  ordered = TRUE
)

## S4 method for signature 'DNAStringSet'
bc_extract(
  x,
  pattern = "",
  sample_name = NULL,
  maxLDist = 0,
  pattern_type = c(barcode = 1),
  costs = list(sub = 1, ins = 99, del = 99),
```

```
  ordered = TRUE
)

## S4 method for signature 'integer'
bc_extract(
  x,
  pattern = "",
  sample_name = NULL,
  maxLDist = 0,
  pattern_type = c(barcode = 1),
  costs = list(sub = 1, ins = 99, del = 99),
  ordered = TRUE
)

## S4 method for signature 'character'
bc_extract(
  x,
  pattern = "",
  sample_name = NULL,
  metadata = NULL,
  maxLDist = 0,
  pattern_type = c(barcode = 1),
  costs = list(sub = 1, ins = 99, del = 99),
  ordered = TRUE
)

## S4 method for signature 'list'
bc_extract(
  x,
  pattern = "",
  sample_name = NULL,
  metadata = NULL,
  maxLDist = 0,
  pattern_type = c(barcode = 1),
  costs = list(sub = 1, ins = 99, del = 99),
  ordered = TRUE
)
```

Arguments

x	A single or a list of fastq file, ShortReadQ, DNAStringSet, data.frame, or named integer.
pattern	A string, specifying the regular expression with capture. It matchs the barcode (and UMI) with capture pattern.
sample_name	A string vector, applicable when x is a list or fastq file vector. This argument specifies the sample names. If not provided, the function will look for sample name in the rownames of metadata, the fastqfile name or the list names.

metadata	A data.frame with sample names as the row names, and each metadata record by column, specifying the sample characteristics.
maxLDist	A integer. The mismatch threshold for barcode matching, when maxLDist is 0, the str_match is invoked for barcode matching which is faster, otherwise aregexec is invoked and the costs parameters can be used to specifying the weight of the distance calculation.
pattern_type	A vector. It defines the barcode (and UMI) capture group. See Details.
costs	A named list, applicable when maxLDist > 0, specifying the weight of each mismatch events while extracting the barcodes. The list element name have to be sub (substitution), ins (insertion) and del (deletion). The default value is list(sub = 1, ins = 99, del = 99). See aregexec for more detail information.
ordered	A logical value. If the value is true, the return barcodes (UMI-barcode tags) are sorted by the reads counts.

Details

The pattern argument is a regular expression, the capture operation () identifying the barcode or UMI. pattern_type argument annotates capture, denoting the UMI or the barcode captured pattern. In the example:

```
([ACTG]{3})TCGATCGATCGA([ACTG]+)ATCGATCGATC
|---------| starts with 3 base pairs UMI.
           |----------| constant sequence in the backbone.
                      |-------| flexible barcode sequences.
                              |---------| 3' constant sequence.
```

In UMI part [ACGT]{3}, [ACGT] means it can be one of the "A", "C", "G" and "T", and {3} means it repeats 3 times. In the barcode pattern [ACGT]+, the + denotes that there is at least one of the A or C or G or T.

Value

This function returns a BarcodeObj object if the input is a list or a vector of Fastq files, otherwise it returns a data.frame. In the later case the data.frame has 5 columns:

1. reads_seq: full sequence.
2. match_seq: part of the full sequence matched by pattern.
3. umi_seq (optional): UMI sequence, applicable when there is UMI in ‘pattern‘ and ‘pattern_type‘ argument.
4. barcode_seq: barcode sequence.
5. count: reads number.

The match_seq is part of reads_seq; The umi_seq and barcode_seq are part of match_seq. The reads_seq is the full sequence, and is unique id for each record (row), On the contrast, match_seq, umi_seq or barcode_seq may duplicated between rows.

Examples

```
fq_file <- system.file("extdata", "simple.fq", package="CellBarcode")

library(ShortRead)

# barcode from fastq file
bc_extract(fq_file, pattern = "AAAAA(.*)CCCCC")

# barcode from ShortReadQ object
sr <- readFastq(fq_file)  # ShortReadQ
bc_extract(sr, pattern = "AAAAA(.*)CCCCC")

# barcode from DNAStringSet object
ds <- sread(sr)  # DNAStringSet
bc_extract(ds, pattern = "AAAAA(.*)CCCCC")

# barcode from integer vector
iv <- tables(ds, n = Inf)$top # integer vector
bc_extract(iv, pattern = "AAAAA(.*)CCCCC")

# barcode from data.frame
df <- data.frame(seq = names(iv), freq = as.integer(iv)) # data.frame
bc_extract(df, pattern = "AAAAA(.*)CCCCC")

# barcode from list of DNAStringSet
l <- list(sample1 = ds, sample2 = ds) # list
bc_extract(l, pattern = "AAAAA(.*)CCCCC")

# Extract UMI and barcode
d1 <- data.frame(
    seq = c(
        "ACTTCGATCGATCGAAAAGATCGATCGATC",
        "AATTCGATCGATCGAAGAGATCGATCGATC",
        "CCTTCGATCGATCGAAGAAGATCGATCGATC",
        "TTTTCGATCGATCGAAAAGATCGATCGATC",
        "AAATCGATCGATCGAAGAGATCGATCGATC",
        "CCCTCGATCGATCGAAGAAGATCGATCGATC",
        "GGGTCGATCGATCGAAAAGATCGATCGATC",
        "GGATCGATCGATCGAAGAGATCGATCGATC",
        "ACTTCGATCGATCGAACAAGATCGATCGATC",
        "GGTTCGATCGATCGACGAGATCGATCGATC",
        "GCGTCCATCGATCGAAGAAGATCGATCGATC"
        ),
    freq = c(
        30, 60, 9, 10, 14, 5, 10, 30, 6, 4 , 6
    )
  )
# barcode backbone with UMI and barcode
pattern <- "([ACTG]{3})TCGATCGATCGA([ACTG]+)ATCGATCGATC"
bc_extract(
    list(test = d1),
    pattern,
```

```
    sample_name=c("test"),
    pattern_type=c(UMI=1, barcode=2))

###
```

---

bc_messyBc *Accesses messyBc slot in the BarcodeObj object*

Description

messyBc slot of BarcodeObj object contains the raw barcode reads frequency data. For more detail about the messyBc slot, see BarcodeObj. bc_messyBc is used to access the 'messyBc' slot in the BarcodeObj.

Usage

```
bc_messyBc(barcodeObj, isList = TRUE)

## S4 method for signature 'BarcodeObj'
bc_messyBc(barcodeObj, isList = TRUE)
```

Arguments

barcodeObj	A BarcodeObj objects.
isList	A logical value, if TRUE (default), the return is a list with each sample as an element. Otherwise, the function will return a data.frame contains the data from all the samples with a column named sample_name to keep the sample information.

Value

If a list is requested, in the list each element is a data.frame corresponding to the successive samples. Each data.frame has 5 columns: 1. reads_seq: full read sequence before parsing. 2. match_seq: the sequence matched by pattern given to bc_extract. 3. umi_seq (optional): UMI sequence. 4. barcode_seq: barcode sequence. 5. count: how many reads a full sequence has. In this table, barcode_seq value can be duplicated, as two different full read sequences can contain the same barcode sequence, due to the diversity of the UMI or mutations in the constant region.

#' If a data.frame is requested, the data.frame in the list described above are combined into one data.frame by row, with an extra column named sample_name for identifying sample.

Examples

```
 data(bc_obj)
# get the data in messyBc slot
# default the return value is a list
bc_messyBc(bc_obj)

# the return value can be a data.frame
```

```
bc_messyBc(bc_obj, isList=FALSE)
###
```

---

bc_meta *Accesses and sets metadata in BarcodeObj object*

Description

Sample information is kept in metadata. bc_meta is for accessing and updating metadata in BarcodeObj object

Usage

```
bc_meta(barcodeObj)

bc_meta(barcodeObj, key = NULL) <- value

## S4 method for signature 'BarcodeObj'
bc_meta(barcodeObj)

## S4 replacement method for signature 'BarcodeObj'
bc_meta(barcodeObj, key = NULL) <- value
```

Arguments

barcodeObj	A BarcodeObj object.
key	A string, identifying the metadata record name to be modified.
value	A string vector or a data.frame. If the value is a vector, it should have the same length of sample number in the BarcodeObj object. Otherwise, if the value is data.frame, the row name of the data.frame should be the sample name, and each column as a metadata variable.

Value

A data.frame

Examples

```
data(bc_obj)

# get the metadata data.frame
bc_meta(bc_obj)

# assign value to a metadata by $ operation
bc_meta(bc_obj)$phenotype <- c("l", "b")

# assign value to a metasta by "key" argument
bc_meta(bc_obj, key = "sample_type") <- c("l", "b")
```

```
# show the updated metadata
bc_meta(bc_obj)

# assign a new data.frame to metadata
metadata <- data.frame(
    sample_name <- c("test1", "test2"),
    phenotype <- c("1", "b")
    )
rownames(metadata) = bc_names(bc_obj)
bc_meta(bc_obj) <- metadata
###
```

---

bc_names *Access & update sample names in BarcodeObj & and BarcodeQcSet*

Description

Get or update sample names in BarcodeObj object and BarcodeQcSet.

Usage

```
bc_names(x)

bc_names(x) <- value

## S4 method for signature 'BarcodeObj'
bc_names(x)

## S4 replacement method for signature 'BarcodeObj,character'
bc_names(x) <- value

## S4 method for signature 'BarcodeQcSet'
bc_names(x)

## S4 replacement method for signature 'BarcodeQcSet,ANY'
bc_names(x) <- value
```

Arguments

x	A BarcodeObj object or a BarcodeQcSet object.
value	A character vector setting the new sample names, with the length of the samples number in BarcodeObj or BarcodeQcSet object.

Value

A character vector

Examples

```
data(bc_obj)

bc_names(bc_obj)
bc_names(bc_obj) <- c("new1", "new2")
```

---

bc_obj *A dummy BarcodeObj object*

Description

Dataset contains a BarcodeObj with makeup barcode data.

Usage

```
data(bc_obj)
```

Format

This is a BarcodeObj object

Source

This is a BarcodeObj object derived from makeup data by:

```
d1 = data.frame(
    seq = c(
       "ACTTCGATCGATCGAAAAGATCGATCGATC",
       "AATTCGATCGATCGAAGAGATCGATCGATC",
       "CCTTCGATCGATCGAAGAAGATCGATCGATC",
       "TTTTCGATCGATCGAAAAGATCGATCGATC",
       "AAATCGATCGATCGAAGAGATCGATCGATC",
       "CCCTCGATCGATCGAAGAAGATCGATCGATC",
       "GGGTCGATCGATCGAAAAGATCGATCGATC",
       "GGATCGATCGATCGAAGAGATCGATCGATC",
       "ACTTCGATCGATCGAACAAGATCGATCGATC",
       "GGTTCGATCGATCGACGAGATCGATCGATC",
       "GCGTCCATCGATCGAAGAAGATCGATCGATC"
       ),
    freq = c(
        30, 60, 9, 10, 14, 5, 10, 30, 6, 4 , 6
        )
    )

d2 = data.frame(
    seq = c(
       "ACTTCGATCGATCGAAACGATCGATCGATC",
```

```
        "AATTCGATCGATCGAAGAGATCGATCGATC",
        "TTTTCGATCGATCGAAAAGATCGATCGATC",
        "AAATCGATCGATCGAAGAGATCGATCGATC",
        "CCCTCGATCGATCGAAGAAGATCGATCGATC",
        "GGGTCGATCGATCGAAAAGATCGATCGATC",
        "GGATCGATCGATCGAAGAGATCGATCGATC",
        "ACTTCGATCGATCGAACAAGATCGATCGATC",
        "GGTTCGATCGATCGACGAGATCGATCGATC",
        "GCGTCCATCGATCGAAGAAGATCGATCGATC"
        ),
    freq = c(
        30, 9, 10, 14, 5, 10, 30, 6, 4 , 6
        )
    )

pattern = "TCGATCGATCGA([ACTG]+)ATCGATCGATC"
bc_obj = bc_extract(
    list(test1 = d1, test2 = d2),
    pattern, sample_name=c("test1", "test2"))

bc_obj = bc_cure_depth(bc_obj, depth=5)

# save the dummy data
# save(bc_obj, file = "./data/bc_obj.RData")
###
```

---

bc_plot_mutual *Barcode read count 2D scatter plot of sample combination*

Description

Draw barcode count scatter plot for all pairwise combination of samples within a BarcodeObj object. It uses cleanBc slot in the BarcodeObj object is used to draw the figure. If the BarcodeObj object does not have a cleanBc slot, you have to run the bc_cure* functions in ahead, such as bc_cure_depth, bc_cure_umi.

Usage

```
bc_plot_mutual(
  barcodeObj,
  count_marks = NULL,
  highlight = NULL,
  log_coord = TRUE,
  alpha = 0.7
)

## S4 method for signature 'BarcodeObj'
```

```
bc_plot_mutual(
  barcodeObj,
  count_marks = NULL,
  highlight = NULL,
  log_coord = TRUE,
  alpha = 0.7
)
```

Arguments

barcodeObj	A BarcodeObj object, which has a cleanBc slot
count_marks	A numeric or numeric vector, specifying the read count cutoff in the scatter plot for each sample.
highlight	A character vector, specifying the barcodes to be highlighted.
log_coord	A logical value, if TRUE (default), the x and y coordinates of the scatter plot will be logarized by log10.
alpha	A numeric between 0 and 1, specifies the transparency of the dots in the scatter plot.

Value

A scatter plot matrix.

Examples

```
data(bc_obj)

bc_plot_mutual(barcodeObj=bc_obj, count_marks=c(30, 20))
###
```

---

bc_plot_pair *Barcode read count 2D scatter plot for given pairs*

Description

Draws scatter plot for barcode read count between given pairs of samples with a BarcodeObj object. This function will return scatter plot matrix contains the scatter plots for all given sample pairs.

Usage

```
bc_plot_pair(
  barcodeObj,
  sample_x,
  sample_y,
  count_marks_x = NULL,
  count_marks_y = NULL,
  highlight = NULL,
```

```
  log_coord = TRUE,
  alpha = 0.7
)

## S4 method for signature 'BarcodeObj'
bc_plot_pair(
  barcodeObj,
  sample_x,
  sample_y,
  count_marks_x = NULL,
  count_marks_y = count_marks_x,
  highlight = NULL,
  log_coord = TRUE,
  alpha = 0.7
)
```

Arguments

barcodeObj	A BarcodeObj object.
sample_x	A character vector or a integer vector, specifying the sample in x axis of each scatter plot. It can be the sample names in BarcodeObj or the sample index value.
sample_y	A character vector or a integer vector, similar to sample_x, specifying the samples used for y axis. It can be the sample names or the sample index value.
count_marks_x	A numeric vector used to mark the cutoff point for samples in x axis
count_marks_y	A number vector used to mark the cutoff point for samples in y axis.
highlight	A character vector, specifying the barcodes need to be highlighted.
log_coord	A logical value, if TRUE (default), the x and y coordinates of the scatter will be logarized by log10.
alpha	A numeric between 0 and 1, specifies the transparency of the dots in the scatter plot.

Value

Scatter plot matrix.

Examples

```
data(bc_obj)

bc_names(bc_obj)

bc_plot_pair(barcodeObj=bc_obj, sample_x="test1", sample_y="test2",
    count_marks_x=30, count_marks_y=20)
###
```

bc_plot_single *Scatter plot of barcode count distribution per sample*

Description

Draws barcode count distribution for each sample in a BarcodeObj object.

Usage

```
bc_plot_single(
  barcodeObj,
  sample_names = NULL,
  count_marks = NULL,
  highlight = NULL,
  log_coord = TRUE,
  alpha = 0.7
)

## S4 method for signature 'BarcodeObj'
bc_plot_single(
  barcodeObj,
  sample_names = bc_names(barcodeObj),
  count_marks = NULL,
  highlight = NULL,
  log_coord = TRUE,
  alpha = 0.7
)
```

Arguments

barcodeObj	A BarcodeObj object has a cleanBc slot
sample_names	A character vector or integer vector, specifying the samples used for plot.
count_marks	A numeric or numeric vector, specifying the read count cutoff in the scatter plot for each sample.
highlight	A character vector, specifying the barcodes need to be highlighted.
log_coord	A logical value, if TRUE (default), the x and y coordinates of the scatter plot will be logarized by log10.
alpha	A numeric between 0 and 1, specifies the transparency of the dots in the scatter plot.

Value

1D distribution graph matrix.

Examples

```
data(bc_obj)

bc_plot_single(bc_obj, count_marks=c(10, 11))
###
```

`bc_seq_filter` *Remove low quality sequence*

Description

Remove low quality sequences by base-pair quality, sequence length or unknown base "N".

Usage

```
bc_seq_filter(
  x,
  min_average_quality = 30,
  min_read_length = 0,
  N_threshold = 0,
  sample_name = ""
)

## S4 method for signature 'ShortReadQ'
bc_seq_filter(
  x,
  min_average_quality = 30,
  min_read_length = 0,
  N_threshold = 0
)

## S4 method for signature 'DNAStringSet'
bc_seq_filter(x, min_read_length = 0, N_threshold = 0)

## S4 method for signature 'data.frame'
bc_seq_filter(x, min_read_length = 0, N_threshold = 0)

## S4 method for signature 'character'
bc_seq_filter(
  x,
  min_average_quality = 30,
  min_read_length = 0,
  N_threshold = 0,
  sample_name = basename(x)
)

## S4 method for signature 'integer'
```

```
bc_seq_filter(x, min_read_length = 0, N_threshold = 0)

## S4 method for signature 'list'
bc_seq_filter(
  x,
  min_average_quality = 30,
  min_read_length = 0,
  N_threshold = 0,
  sample_name = names(x)
)
```

Arguments

x	A single or a list of Fastq file, ShortReadQ, DNAStringSet, data.frame, integer vector.
min_average_quality	A numeric or a vector of numeric, specifying the threshold of the minimum average base quality of a sequence to be kept.
min_read_length	A single or a vector of integer, specifying the sequence length threshold.
N_threshold	A integer or a vector of integer, specifying the maximum N can be in a sequence.
sample_name	A string vector, specifying the sample name in the output.

Value

A ShortReadQ or DNAStringSet object with sequences passed the filters.

Examples

```
library(ShortRead)

fq_file <- system.file("extdata", "simple.fq", package="CellBarcode")

# apply filter to fastq files
bc_seq_filter(fq_file)

# read in fastq files to get ShortReadQ object
sr <- readFastq(fq_file[1])
# apply sequencing quality filter to ShortReadQ
bc_seq_filter(sr)

# get DNAStringSet object
ds <- sread(sr)
# apply sequencing quality filter to DNAStringSet
bc_seq_filter(ds)

###
```

bc_seq_qc *Evaluates sequences quality*

Description

bc_seq_qc evaluates sequences quality. See the return value for detail.

Usage

```
bc_seq_qc(x, sample_name = NULL)

bc_plot_seqQc(x)

## S4 method for signature 'ShortReadQ'
bc_seq_qc(x)

## S4 method for signature 'DNAStringSet'
bc_seq_qc(x)

## S4 method for signature 'data.frame'
bc_seq_qc(x)

## S4 method for signature 'integer'
bc_seq_qc(x)

## S4 method for signature 'character'
bc_seq_qc(x, sample_name = basename(x))

## S4 method for signature 'list'
bc_seq_qc(x, sample_name = names(x))

## S4 method for signature 'BarcodeQc'
bc_plot_seqQc(x)

## S4 method for signature 'BarcodeQcSet'
bc_plot_seqQc(x)
```

Arguments

x	A single or list of Fastq file, ShortReadQ object, DNAStringSet object, data.frame or named integer vector.
sample_name	A character vector with the length of sample number, used to set the sample name.

Value

A barcodeQc or a barcodeQcSet class. The barcodeQc is a list with four slots,

- top: a data.frame with top 50 most frequency sequence,
- distribution: a data.frame with the distribution of read depth. It contains nOccurrences (depth), and nReads (unique sequence) columns.
- base_quality_per_cycle: data.frame with base-pair location (NGS sequencing cycle) by row, and the base-pair quality summary by column, including Mean, P5 (5 P75 (75
- base_freq_per_cycle: data.frame with three columns: 1. Cycle, the sequence base-pair location (NGS sequencing cycle); 2. Base, DNA base; Count: reads count.
- summary: a numeric vector with following elements: total_read, median_read_length, p5_read_length, p95_read_length.

The barcodeQcSet is a list of barcodeQc.

Examples

```
library(ShortRead)
# fastq file
fq_file <- system.file("extdata", "simple.fq", package="CellBarcode")
bc_seq_qc(fq_file)

# ShortReadQ
sr <- readFastq(fq_file[1])
bc_seq_qc(sr)

# DNAStringSet
ds <- sread(sr)
bc_seq_qc(ds)

# List of DNAStringSet
l <- list(sample1 = ds, sample2 = ds)
bc_plot_seqQc(bc_seq_qc(l))

# List of ShortRead
l_sr <- list(sample1 = sr, sample2 = sr)
bc_plot_seqQc(bc_seq_qc(l_sr))

###
```

---

bc_subset *Manages barcodes and samples in a BarcodeObj object*

Description

A set of functions and operators for subset or join of BarcodeObj object(s). The bc_subset, * and - are used to select barcodes or samples in a BarcodeObj object. Two BarcodeObj objects can be joined by +.

Usage

```
bc_subset(
  barcodeObj,
  sample = NULL,
  barcode = NULL,
  black_list = NULL,
  is_sample_quoted_exp = FALSE
)

bc_merge(barcodeObj_x, barcodeObj_y)

## S4 method for signature 'BarcodeObj'
bc_subset(
  barcodeObj,
  sample = NULL,
  barcode = NULL,
  black_list = NULL,
  is_sample_quoted_exp = FALSE
)

## S4 method for signature 'BarcodeObj,BarcodeObj'
bc_merge(barcodeObj_x, barcodeObj_y)

## S3 method for class 'BarcodeObj'
barcodeObj_x + barcodeObj_y

## S3 method for class 'BarcodeObj'
barcodeObj - black_list

## S3 method for class 'BarcodeObj'
barcodeObj * white_list
```

Arguments

barcodeObj	A BarcodeObj object.
sample	A character vector or integer vector or an expression (expressio not applicable for [] operator), specifying the samples in the subsets. When the value is an expression, the columns in the metadata can be used as variable.
barcode	A vector of integer or string, indicating the selected barcode.
black_list	A character vector, specifying the black list with excluded barcodes.
is_sample_quoted_exp	A logical value. If TRUE, the expression in sample argument will not be evaluated before executing the function.
barcodeObj_x	A BarcodeObj object.
barcodeObj_y	A BarcodeObj object.
white_list	A character vector, giving the barcode white list.

Details

bc_subset and []: Gets samples and barcodes subset from a BarcodeObj object.

+: Combines two BarcodeObj objects. The metadata, cleanBc and messyBc slot in the BarcodeObj objects will be joined. For the metadata slot, the sample_name column, and the *Full outer join* (the record in either BarcodeObj object) will be performed with rownames as the key. The messyBc and cleanBc from two objects are combined by rows for the same sample from two BarcodeObj objects.

-: removes barcodes in the black_list.

*: selects barcodes in the white_list.

Value

A BarcodeObj object.

Examples

```
data(bc_obj)

bc_obj

# Select barcodes
bc_subset(bc_obj, barcode = c("AACCTT", "AACCTT"))
bc_obj[c("AGAG", "AAAG"), ]

# Select samples by meta data
bc_meta(bc_obj)$phenotype <- c("l", "b")
bc_meta(bc_obj)
bc_subset(bc_obj, phenotype == "l")

# Select samples by sample name
bc_obj[, "test1"]
bc_obj[, c("test1", "test2")]
bc_subset(bc_obj, sample = "test1", barcode = c("AACCTT", "AACCTT"))

# Apply barcodes black list
bc_subset(
bc_obj,
    sample = c("test1", "test2"),
    barcode = c("AACCTT"))

# Join two samples with different barcode sets
bc_obj["AGAG", "test1"] + bc_obj["AAAG", "test2"]

# Join two samples with overlap barcodes
bc_obj_join <- bc_obj["AGAG", "test1"] + bc_obj["AGAG", "test2"]
bc_obj_join
# The same barcode will merged after applying bc_cure_depth()
bc_cure_depth(bc_obj_join)

# Remove barcodes
```

```
bc_obj
bc_obj - "AAAG"

# Select barcodes in white list
bc_obj
bc_obj * "AAAG"
###
```

---

bc_summary_barcode *Summary and evaluate barcode diversity*

Description

bc_summary_barcode evaluates sequence diversity metrics using the barcodes data in the cleanBc slot of BarcodeObj object. It also generates Lorenz curve and barcode frequency distribution graphs.

Usage

```
bc_summary_barcode(barcodeObj, plot = TRUE, log_x = TRUE)

## S4 method for signature 'BarcodeObj'
bc_summary_barcode(barcodeObj, plot = TRUE, log_x = TRUE)
```

Arguments

barcodeObj	A BarcodeObj object.
plot	A logical value, if TRUE, draw the Lorenz curve and barcode distribution graphs.
log_x	A logical value, if TRUE, the x axis is logarized.

Details

Followings are the metrics used for evaluating the barcode diversity:

Richness: The unique barcodes number R, it evaluates the richness of the barcodes.

Shannon index: Shannon diversity index is weighted geometric average of the proportion p of barcodes.

$$H' = -\sum_{i=1}^{R} p_i ln p_i$$

Equitability index: Shannon equitability E_H characterize the evenness of the barcodes, it is a value between 0 and 1, with 1 being complete evenness.

$$E_H = H'/H'_{max} = H/ln(R)$$

Bit: Shannon entropy H, with a units of bit,

$$H = -\sum_{i=1}^{R} p_i log_2 p_i$$

Value

A data.frame with following columns:

- `total_reads`: total read number.
- `uniq_barcode`: how many barcodes in the dataset.
- `shannon_index`: Shannon's diversity index or Shannon–Wiener index.
- `equitability_index`: Shannon's equitability.
- `bit_index`: Shannon bit information.

Examples

```
data(bc_obj)

# filter barcode by depth
bc_obj <- bc_cure_depth(bc_obj)

# Output the summary of the barcodes
bc_summary_barcode(bc_obj)
```

`bc_summary_seqQc` *Summary barcodeQcSet*

Description

Summary the "total read count" and "read length" of each samples within a `BarcodeQcSet` object, and output a `data.frame` with sample by row and different metrics by column.

Usage

```
bc_summary_seqQc(x)

## S4 method for signature 'BarcodeQcSet'
bc_summary_seqQc(x)
```

Arguments

`x`	a barcodeQcSet object.

Value

A `data.frame` with 5 columns: `sample_name`, `total_read`, `median_read_length`, `p5_read_length` and `p95_read_length`.

Examples

```
fq_file <- dir(
    system.file("extdata", "mef_test_data", package = "CellBarcode"),
    full=TRUE)

bc_summary_seqQc(bc_seq_qc(fq_file))
###
```

`CellBarcode` *DNA Barcode Analysis toolkit*

Description

This package performs DNA Barcode (genetic lineage tracing) analysis. The package can handle all kinds of DNA barcodes, as long as the barcode within a single sequencing read and has a pattern which can be matched by a regular expression. `CellBarcode` can handle barcode with flexible length, with or without UMI (unique molecular identifier). This tool also can be used for pre-processing of some amplicon data such as CRISPR gRNA screening, immune repertoire sequencing and meta genome data.

`format,BarcodeObj-method`
Formats BarcodeObj object

Description

Format the summary of BarcodeObj object for pretty print.

Usage

```
## S4 method for signature 'BarcodeObj'
format(x)
```

Arguments

`x`	A BarcodeObj object

Value

Formated summary text.

Examples

```
data(bc_obj)

# format BarcodeObj for pretty print
format(bc_obj)

###
```

show,BarcodeObj-method *Show BarcodeObj object*

Description

Show the summary of BarcodeObj object for pretty print.

Show the summary of BarcodeQc object for pretty print.

Show the summary of BarcodeQcSet object for pretty print.

Usage

```
## S4 method for signature 'BarcodeObj'
show(object)

## S4 method for signature 'BarcodeQc'
show(object)

## S4 method for signature 'BarcodeQcSet'
show(object)
```

Arguments

object	A BarcodeQcSet object

Value

Formated summary text.

Formated summary text.

Formated summary text.

Examples

```
data(bc_obj)

# show BarcodeObj for pretty print
bc_obj

###
```

`[,BarcodeQcSet,ANY,ANY,ANY-method` *Subset the BarcodeQcSet*

Description

Subset the BarcodeQcSet

Usage

```
## S4 method for signature 'BarcodeQcSet,ANY,ANY,ANY'
x[i, drop = TRUE]
```

Arguments

x	A BarcodeQcSet object
i	A integer vector or a character vector, specifying the selected samples.
drop	a logical value, if TRUE, when only one sample is selected, the output will be a BarcodeQc object.

Value

A BarcodeQcSet or BarcodeQc

Examples

```
example_data <- system.file("extdata", "mef_test_data", package = "CellBarcode")
fq_files <- dir(example_data, "fastq.gz", full=TRUE)
qc_noFilter <- bc_seq_qc(fq_files)
qc_noFilter[1:3]
```

Index