Package 'CellBarcode'

October 15, 2026

Type Package

Title Cellular DNA Barcode Analysis toolkit

Version 1.18.0

Description The package CellBarcode performs Cellular DNA Barcode analysis. It can handle all kinds of DNA barcodes, as long as the barcode is within a single sequencing read and has a pattern that can be matched by a regular expression. \code{CellBarcode} can handle barcodes with flexible lengths, with or without UMI (unique molecular identifier). This tool also can be used for pre-processing some amplicon data such as CRISPR gRNA screening, immune repertoire sequencing, and metagenome data.

License Artistic-2.0

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, seqinr, Rsamtools

LinkingTo Rcpp, BH

RoxygenNote 7.3.2

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

biocViews Preprocessing, QualityControl, Sequencing, CRISPR

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

URL https://wenjie1991.github.io/CellBarcode/

BugReports https://github.com/wenjie1991/CellBarcode/issues

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

git_branch RELEASE_3_23

git_last_commit 18ef734

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-10-15

Author Wenjie Sun [cre, aut] (ORCID: <https://orcid.org/0000-0002-3100-2346>),
Anne-Marie Lyne [aut],
Leila Perie [aut]

Maintainer Wenjie Sun <sunwjie@gmail.com>

Contents

BarcodeObj-class . . . 3
bc_2df . . . 4
bc_auto_cutoff . . . 5
bc_barcodes . . . 6
bc_cleanBc . . . 7
bc_create_BarcodeObj . . . 8
bc_cure_cluster . . . 9
bc_cure_depth . . . 11
bc_cure_umi . . . 12
bc_extract . . . 14
bc_extract_sc_fastq . . . 18
bc_extract_sc_sam . . . 19
bc_messyBc . . . 21
bc_meta . . . 22
bc_names . . . 23
bc_obj . . . 24
bc_plot_count . . . 25
bc_plot_mutual . . . 26
bc_plot_pair . . . 27
bc_plot_single . . . 29
bc_seq_filter . . . 30
bc_seq_qc . . . 32
bc_splitVDJ . . . 34
bc_subset . . . 35
bc_summary_barcode . . . 37
bc_summary_seqQc . . . 39
CellBarcode . . . 39
format,BarcodeObj-method . . . 40
parse_10x_sam . . . 41
seq_correct . . . 41
show,BarcodeObj-method . . . 43
subset,BarcodeQcSet-method . . . 44

Index **45**

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 that holds the raw barcodes sequence without filtering, where each element is a `data.table` corresponding to the successive samples. Each table has 3 columns: 1. `umi_seq` (optional): UMI sequence. 2. `barcode_seq`: barcode sequence. 3. `count`: how many reads a full sequence has. In this table, `barcode_seq` value can be duplicated, as two different full read sequences can have 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 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`.

Value

A `BarcodeObj` object.

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 metadata
bc_meta(bc_obj)$phenotype <- c("l", "b")
```

```
bc_meta(bc_obj)
bc_subset(bc_obj, sample = phenotype == "1")

# 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 blacklist
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 overlap barcodes
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 another 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 to identify the "true barcode" based on the read count. The algorithm detail is: 1. Remove the barcodes with counts below the median of counts. 2. Transform the count by log2(x+1). 3. Apply the 1-dimension clustering to the log count, with the cluster number of 2 and weights of the log count. 4. Choose the minimum count value in the cluster with more counts as cutoff point.

For more info about 1 dimension kmeans used here please refer to Ckmeans.1d.dp.

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 latter 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 barcodes 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 is a `data.frame` for each sample. In a `data.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_create_BarcodeObj` *Create a BarcodeObj object from extracted barcodes data*

Description

Create a BarcodeObj object from extracted barcodes data

Usage

```
bc_create_BarcodeObj(x, sample_name = NULL, metadata = NULL, ordered = TRUE)

## S4 method for signature 'matrix'
bc_create_BarcodeObj(x, sample_name = NULL, metadata = NULL)

## S4 method for signature 'data.frame'
bc_create_BarcodeObj(x, sample_name = NULL, metadata = NULL)
```

Arguments

`x`	The barcodes data, it can be matrix, data.frame with each row as a barcode each column as a sample. The row names should be given as the barcode sequences, and the column names can be given as the sample names.
`sample_name`	A character vector, optional, specifying the sample name.
`metadata`	A data.frame, optional, specifying the metadata of each sample. The row names of the metadata should be the same as the sample names.
`ordered`	A logical value. If the value is true, the return barcodes (UMI-barcode tags) are sorted by the read counts.

Value

A BarcodeObj object.

Examples

```
data(bc_obj)
m = bc_2matrix(bc_obj)
bc_create_BarcodeObj(m)
```

bc_cure_cluster *Clean barcodes by editing distance*

Description

bc_cure_cluster performs clustering of barcodes by editing distance, and remove the minority barcodes with a similar sequence. This function is only applicable for the BarcodeObj object with a cleanBc slot. The barcodes with a smaller reads count will be removed.

Usage

```
bc_cure_cluster(
  barcodeObj,
  dist_threshold = 1,
  depth_fold_threshold = 1,
  dist_method = "hamm",
  cluster_method = "greedy",
  count_threshold = 1e+09,
  dist_costs = list(replace = 1, insert = 1, delete = 1)
)

## S4 method for signature 'BarcodeObj'
bc_cure_cluster(
  barcodeObj,
  dist_threshold = 1,
  depth_fold_threshold = 1,
  dist_method = "hamm",
  cluster_method = "greedy",
  count_threshold = 1e+07,
  dist_costs = list(replace = 1, insert = 1, delete = 1)
)
```

Arguments

barcodeObj	A BarcodeObj object.
dist_threshold	A single integer, or vector of integers with the length of sample number, specifying the editing distance threshold for defining two similar barcode sequences. If the input is a vector, each value in the vector relates to one sample according to its order in BarcodeObj object. The sequences with editing distance equal to or less than the threshold will be considered similar barcodes.
depth_fold_threshold	A single numeric or vector of numeric with the length of sample number, specifying the depth fold change threshold of removing the similar minority barcode. The majority of barcodes should have at least depth_fold_threshold times of reads of the similar minotiry one, to remove the minority similar barcode. (TODO: more precise description)

`dist_method` A character string, specifying the editing distance used for evaluating barcode similarity. It can be "hamm" for Hamming distance or "leven" for Levenshtein distance.

`cluster_method` A character string specifying the algorithm used to perform the clustering of barcodes. Currently only "greedy" is available, in this case, The most and the least abundant barcode will be used for comparing, the least abundant barcode is preferentially removed.

`count_threshold`
An integer, read depth threshold to consider a barcode as a true barcode. If a barcode with a count higher than this threshold it will not be removed, even if the barcode is similar to a more abundant one. Default is 1e9.

`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, and 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, remove the less abundant barcodes
```

```
# one by hamming distance <= 1
bc_cure_cluster(bc_cured, dist_threshold = 1)

# Levenshtein distance <= 1
bc_cure_cluster(bc_cured, dist_threshold = 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 keep. If the input is a vector and the vector length is not the same as the sample number, the element will be repeatedly used. And when the depth argument is a number with a 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.

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 = 1, 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 equal to or 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 absolutely unique, then only the UMI-barcodes tags with the highest count are kept for each UMI.

Details

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

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

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 provides 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 files, ShortReadQ, DNAStringSet, data.frame, or named integer.

pattern	A string or a string vector with the same number of files, specifying the regular expression with capture. It matches 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 names in the rownames of metadata, the fastqfile name or the list names.
metadata	A data.frame with sample names as the row names, with each metadata record by column, specifying the sample characteristics.
maxLDist	An integer. The minimum 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 specify 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 event 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 detailed information.
ordered	A logical value. If the value is true, the return barcodes (UMI-barcode tags) are sorted by the read 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 columns:

1. umi_seq (optional): UMI sequence, applicable when there is UMI in ‘pattern‘ and ‘pattern_type‘ argument.
2. barcode_seq: barcode sequence.
3. count: reads number.

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_extract_sc_fastq` *Extract barcode from single-cell sequencing fastq file*

Description

`bc_extract_10X_fastq` can extract cellular barcode, UMI, and lineage barcode sequences from 10X Genomics scRNASeq fastq file. This function can process the barcodes in the scRNASeq fastq file or target amplified fastq files directly.

Usage

```
bc_extract_sc_fastq(
  fq1,
  fq2 = NULL,
  patternCellBarcode = NULL,
  patternUMI = NULL,
  patternBarcode = NULL
)
```

Arguments

`fq1`	A string, the fastq file contains the cellular barcode and lineage barcode
`fq2`	A string, it is optional, it provides the second fastq file contains the cellular barcode and lineage barcode. Two fastq files will be concatenated for the barcode extraction
`patternCellBarcode`	A string, defines the regular expression to match the single cell cellular barcode sequence. The expected sequence should be in the first catch. Please see the documents of `bc_extract` and example for more information.
`patternUMI`	A string, defines the regular expression to match the UMI sequence. The expected sequence should be in the first catch. Please see the documents of `bc_extract` and example for more information.
`patternBarcode`	the regular expression to match the lineage barcode. The expected sequence should be in the first catch. Please see the documents of `bc_extract` and example for more information.

Details

It should take some effort to define the regular expression to match the barcode sequence. Here I also provide the example to extract the barcode from 10X Genomics scRNASeq results. It also can be used to extract the barcode from other system.

The function can process the barcodes in the scRNASeq fastq file or target amplified fastq files. For the 10X scRNASeq fastq file, the cellular barcode is in the first 16bp of the read1, the UMI is in the next 12bp, and the lineage barcode is in the read2.

The usage of the function will be like this:

```
bc_extract_sc_fastq(
   fq1 = "read1.fastq.gz",
   fq2 = "read2.fastq.gz",
   patternCellBarcode = "(.{16})",
   patternUMI = ".{16}(.{12})",
   patternBarcode = "CGAAGTATCAAG(.+)CCGTAGCAAG"
)
```

Value

A BarcodeObj object with each cell as a sample.

See Also

bc_extract, bc_extract_sc_sam,

`bc_extract_sc_sam` *Extract barcode from single-cell sequencing sam file*

Description

`bc_extract_sc_sam` can extract cellular barcode, UMI, and lineage barcode sequences from 10X Genomics scRNASeq sam file (or bam file have similar data structure). This function can not process bam file directly, users need to uncompress the bam file to get a sam file to run this function See example.

Usage

```
bc_extract_sc_sam(sam, pattern, cell_barcode_tag = "CR", umi_tag = "UR")

bc_extract_sc_bam(bam, pattern, cell_barcode_tag = "CR", umi_tag = "UR")
```

Arguments

`sam`	A string, define the un-mapped sequences
`pattern`	A string, define the regular expression to match the barcode sequence. The barcode sequence should be in the first catch. Please see the documents of `bc_extract` and example for more information.
`cell_barcode_tag`	A string, define the tag of cellular barcode field in sam file. The default is "CR".
`umi_tag`	A string, define the tag of a UMI field in the sam file.
`bam`	A string, define the bam file, it will be converted to sam file

Details

Although the function ‘bc_extract_sc_bam‘ can process bam file directly, some optimization is still working on, it will be much more efficient to use ‘samtools‘ to get the sam file.

What’s more, if the barcode sequence does not map to the reference genome. The user should use the samtools to get the un-mapped reads and save it as sam format for using as the input. It can save a lot of time. The way to get the un-mapped reads:

```
samtools view -f 4 input.bam > output.sam
```

Value

A BarcodeObj object with each cell as a sample.

See Also

bc_extract, bc_extract_sc_fastq

Examples

```
## NOT run
# In the case that when the barcode sequence is not mapped to
# reference genome, it will be much more efficient to get
# the un-mapped sequences as the input.

## Get un-mapped reads
# samtools view -f 4 input.bam > scRNASeq_10X.sam

sam_file <- system.file("extdata", "scRNASeq_10X.sam", package = "CellBarcode")

bc_extract_sc_sam(
  sam = sam_file,
  pattern = "AGATCAG(.*)TGTGGTA",
  cell_barcode_tag = "CR",
  umi_tag = "UR"
)

## Read bam file directly
bam_file <- system.file("extdata", "scRNASeq_10X.bam", package = "CellBarcode")
bc_extract_sc_bam(
   bam = bam_file,
   pattern = "AGATCAG(.*)TGTGGTA",
   cell_barcode_tag = "CR",
   umi_tag = "UR"
)
```

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 at most 3 columns: 1. umi_seq (optional): UMI sequence. 2. barcode_seq: barcode sequence. 3. count: how many reads a full sequence has.

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 metadata by $ operation
bc_meta(bc_obj)$phenotype <- c("l", "b")

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

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

```
# assign 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

Value

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_count *Plot for counts distribution*

Description

This function is used to summarize the counts of each barcode.

Usage

```
bc_plot_count(barcodeObj, bins = 20, useCleaned = TRUE)

## S4 method for signature 'BarcodeObj'
bc_plot_count(barcodeObj, bins = 20, useCleaned = TRUE)
```

Arguments

barcodeObj	A BarcodeObj object
bins	The number of bins for the histogram
useCleaned	Whether to use the cleaned barcode data

Details

When useCleaned is TRUE, the cleaned barcode data will be used. Otherwise, the messy barcode data will be used. The output will be different when useCleaned is TRUE or FALSE. It also depends on whether the UMI is available. The counts include:

1. reads count (with barcode) versus the total reads
2. reads count per UMI
3. UMI count per barcode
4. barcode count per sample
5. reads or UMI count (dominant barcode) versus total count per sample
6. reads or UMI count (dominant barcode) distribution

Value

A egg::ggarrange object

Examples

```
data(bc_obj)
bc_plot_count(barcodeObj=bc_obj)
```

---

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

Description

Draw barcode count scatter plot for all pairwise combinations 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 a 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 the y-axis.
highlight	A character vector, specifying the barcodes that need to be highlighted.
log_coord	A logical value, if TRUE (default), the x and y coordinates of the scatter will be transformed 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 the 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 that need to be highlighted.
`log_coord`	A logical value, if TRUE (default), the `x` and `y` coordinates of the scatter plot are transformed 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 a 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, reads_sample_size = 1e+05)

bc_plot_seqQc(x)

## S4 method for signature 'ShortReadQ'
bc_seq_qc(x, reads_sample_size = 1e+05)

## S4 method for signature 'DNAStringSet'
bc_seq_qc(x, reads_sample_size = 1e+05)

## S4 method for signature 'data.frame'
bc_seq_qc(x, reads_sample_size = 1e+05)

## S4 method for signature 'integer'
bc_seq_qc(x, reads_sample_size = 1e+05)

## S4 method for signature 'character'
bc_seq_qc(x, sample_name = basename(x), reads_sample_size = 1e+05)

## 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 files, 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.

reads_sample_size
: A integer value defines the sample size of the sequences for quality control analysis. If there are fewer sequences comparing to this value, all the sequences will be used. The default is 1e5.

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_splitVDJ *Parse VDJ recombination (experimental)*

Description

Script to split barcodes from the genetic 'barcode mouse' construct as generated in the lab of Ton Schumacher (NKI, NL) in its remaining constant V, D and J elements and the modified elements (additions/deletions) in between those constant parts.

Usage

```
bc_splitVDJ(
  seqs,
  v_part = "TCCAGTAG",
  d_fwd = "TCTACTATCGTTACGAC",
  d_inv = "GTCGTAACGATAGTAGA",
  j_part = "GTAGCTACTACCG"
)
```

Arguments

seqs	a character vector contains the barcode sequences.
v_part	a string given the V part sequence.
d_fwd	a string given the D region forwrad sequence.
d_inv	a string given the D region inverted sequence.
j_part	a string given the J region sequence.

Value

A list contains two data.frame named `add.del.ok` and `add.del.err`, which contain columns with the remaining constant parts and inserted/deleted parts

Examples

```
## prepare input sequence
seq_v <- c(
    "TCCAGTAGCTACTATCGTTACGAGTAGCTACTACCG",
    "TCCAGTAGCTACTATCGTTACGACGTAGCTACTACCG",
    "TCCATACTATCGTTACGACGTAGCTACTACG",
    "TCCAGTAGTCGTAACGATAGTAGAGTAGCTACTACCG"
    )

## split the sequences
bc_splitVDJ(seq_v)
```

bc_subset *Manages barcodes and samples in a BarcodeObj object*

Description

A set of functions and operators for subsetting or joining 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 a 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 row names 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 metadata
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 barcode blacklist
```

```
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 be removed after applying bc_cure_depth()
bc_cure_depth(bc_obj_join)

# Remove barcodes
bc_obj
bc_obj - "AAAG"

# Select barcodes in a whitelist
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 the 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 the 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

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

Author(s)

Maintainer: Wenjie Sun <sunwjie@gmail.com> (ORCID)

Authors:

- Anne-Marie Lyne <Anne-Marie.Lyne@curie.fr>
- Leila Perie <leila.perie@curie.fr>

See Also

Useful links:

- https://wenjie1991.github.io/CellBarcode/
- Report bugs at https://github.com/wenjie1991/CellBarcode/issues

---

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)

###
```

`parse_10x_sam` *Parse 10X bam file*

Description

Parse 10X bam file

Usage

```
parse_10x_sam(in_file_path, regex_str, cell_barcode_tag = "CR", umi_tag = "UR")
```

Arguments

`in_file_path`	A string, define the un-mapped sequences
`regex_str`	A string, define the regular expression to match the barcode sequence. The barcode sequence should be in the first catch. Please see the `bc_extract` for detail.
`cell_barcode_tag`	A string, define the tag of 10X cell barcode field in sam file. The default is "CR".
`umi_tag`	A string, define the tag of UMI field in the sam file.

Value

A data.frame with 4 columns:

1. `cell_barcode`: 10X cellular barcode.
2. `umi`: UMI sequence.
3. `barcode_seq`: lineage barcode.
4. `count`: reads count.

`seq_correct` *Sequence clustering*

Description

This function will merge the UMIs by using the hamming distance. If two UMIs have hamming distance no more than 1, only the UMI with more reads will be kept.

Usage

```
seq_correct(
  seq,
  count,
  count_threshold,
  dist_threshold,
  depth_fold_threshold = 1,
  dist_method = 1L,
  insert_cost = 1L,
  delete_cost = 1L,
  replace_cost = 1L
)
```

Arguments

seq	A string vector.
count	An integer vector with the same order and length of UMI
count_threshold	An integer, barcode count threshold to consider a barcode as a true barcode, when when a barcode with count higher than this threshold it will not be removed.
dist_threshold	A integer, distance threshold to consider two barcodes are related.
depth_fold_threshold	An numeric, control the fold cange threshold between the ' major barcodes and the potential contamination that need to be removed.
dist_method	A integer, if 2 the levenshtein distance will be used, otherwise the hamming distance will be applied.
insert_cost	A integer, the insert cost when levenshtein distance is applied.
delete_cost	A integer, the delete cost when levenshtein distance is applied.
replace_cost	A integer, the replace cost when levenshtein distance is applied.

Details

This function will return the corrected UMI list.

Value

a list with two data.frame. seq_freq_tab: table with barcode and corrected ' sequence reads; link_tab: data table record for the clustering process with ' first column of barcode be removed and second column of the majority barcode barcode.

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

###
```

`subset,BarcodeQcSet-method` *Subset the BarcodeQcSet*

Description

Subset the BarcodeQcSet

Usage

```
## S4 method for signature 'BarcodeQcSet'
subset(x, i, drop = TRUE)

## 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

∗ **dataset**
    bc_obj, 24
*.BarcodeObj *(bc_subset)*, 35
+.BarcodeObj *(bc_subset)*, 35
-.BarcodeObj *(bc_subset)*, 35
[,BarcodeQcSet,ANY,ANY,ANY-method *(subset,BarcodeQcSet-method)*, 44

aregexec, *16*

BarcodeObj, *7*, *21*
BarcodeObj *(BarcodeObj-class)*, 3
BarcodeObj-class, 3
BarcodeQc *(bc_seq_qc)*, 32
BarcodeQc-class *(bc_seq_qc)*, 32
BarcodeQcSet *(bc_seq_qc)*, 32
BarcodeQcSet-class *(bc_seq_qc)*, 32
bc_2df, 4
bc_2df,BarcodeObj-method *(bc_2df)*, 4
bc_2dt *(bc_2df)*, 4
bc_2dt,BarcodeObj-method *(bc_2df)*, 4
bc_2matrix *(bc_2df)*, 4
bc_2matrix,BarcodeObj-method *(bc_2df)*, 4
bc_auto_cutoff, 5, *11*
bc_auto_cutoff,BarcodeObj-method *(bc_auto_cutoff)*, 5
bc_barcodes, 6
bc_barcodes,BarcodeObj-method *(bc_barcodes)*, 6
bc_cleanBc, 7
bc_cleanBc,BarcodeObj-method *(bc_cleanBc)*, 7
bc_create_BarcodeObj, 8
bc_create_BarcodeObj,data.frame-method *(bc_create_BarcodeObj)*, 8
bc_create_BarcodeObj,matrix-method *(bc_create_BarcodeObj)*, 8
bc_cure_cluster, 9
bc_cure_cluster,BarcodeObj-method *(bc_cure_cluster)*, 9
bc_cure_depth, 11, *26*
bc_cure_depth,BarcodeObj-method *(bc_cure_depth)*, 11
bc_cure_umi, 12, *26*
bc_cure_umi,BarcodeObj-method *(bc_cure_umi)*, 12
bc_extract, 14, *18–20*, *41*
bc_extract,character-method *(bc_extract)*, 14
bc_extract,data.frame-method *(bc_extract)*, 14
bc_extract,DNAStringSet-method *(bc_extract)*, 14
bc_extract,integer-method *(bc_extract)*, 14
bc_extract,list-method *(bc_extract)*, 14
bc_extract,ShortReadQ-method *(bc_extract)*, 14
bc_extract_sc_bam *(bc_extract_sc_sam)*, 19
bc_extract_sc_fastq, 18, *20*
bc_extract_sc_sam, *19*, 19
bc_merge *(bc_subset)*, 35
bc_merge,BarcodeObj,BarcodeObj-method *(bc_subset)*, 35
bc_messyBc, 21
bc_messyBc,BarcodeObj-method *(bc_messyBc)*, 21
bc_meta, 22
bc_meta,BarcodeObj-method *(bc_meta)*, 22
bc_meta<- *(bc_meta)*, 22
bc_meta<-,BarcodeObj-method *(bc_meta)*, 22
bc_names, 23
bc_names,BarcodeObj-method *(bc_names)*, 23
bc_names,BarcodeQcSet-method

(bc_names), 23
bc_names<- (bc_names), 23
bc_names<-,BarcodeObj,character-method
(bc_names), 23
bc_names<-,BarcodeQcSet,ANY-method
(bc_names), 23
bc_obj, 24
bc_plot_count, 25
bc_plot_count,BarcodeObj-method
(bc_plot_count), 25
bc_plot_mutual, 26
bc_plot_mutual,BarcodeObj-method
(bc_plot_mutual), 26
bc_plot_pair, 27
bc_plot_pair,BarcodeObj-method
(bc_plot_pair), 27
bc_plot_seqQc (bc_seq_qc), 32
bc_plot_seqQc,BarcodeQc-method
(bc_seq_qc), 32
bc_plot_seqQc,BarcodeQcSet-method
(bc_seq_qc), 32
bc_plot_single, 29
bc_plot_single,BarcodeObj-method
(bc_plot_single), 29
bc_seq_filter, 30
bc_seq_filter,character-method
(bc_seq_filter), 30
bc_seq_filter,data.frame-method
(bc_seq_filter), 30
bc_seq_filter,DNAStringSet-method
(bc_seq_filter), 30
bc_seq_filter,integer-method
(bc_seq_filter), 30
bc_seq_filter,list-method
(bc_seq_filter), 30
bc_seq_filter,ShortReadQ-method
(bc_seq_filter), 30
bc_seq_qc, 32
bc_seq_qc,character-method (bc_seq_qc),
32
bc_seq_qc,data.frame-method
(bc_seq_qc), 32
bc_seq_qc,DNAStringSet-method
(bc_seq_qc), 32
bc_seq_qc,integer-method (bc_seq_qc), 32
bc_seq_qc,list-method (bc_seq_qc), 32
bc_seq_qc,ShortReadQ-method
(bc_seq_qc), 32
bc_splitVDJ, 34
bc_subset, 35
bc_subset,BarcodeObj-method
(bc_subset), 35
bc_summary_barcode, 37
bc_summary_barcode,BarcodeObj-method
(bc_summary_barcode), 37
bc_summary_seqQc, 39
bc_summary_seqQc,BarcodeQcSet-method
(bc_summary_seqQc), 39

CellBarcode, 39
CellBarcode-package (CellBarcode), 39
Ckmeans.1d.dp, *6*

format,BarcodeObj-method, 40

parse_10x_sam, 41

seq_correct, 41
show,BarcodeObj-method, 43
show,BarcodeQc-method
(show,BarcodeObj-method), 43
show,BarcodeQcSet-method
(show,BarcodeObj-method), 43
str_match, *16*
subset,BarcodeQcSet-method, 44