Package 'genBaRcode'

December 11, 2023

Title Analysis and Visualization Tools for Genetic Barcode Data

Version 1.2.7

Author Lars Thielecke <lars.thielecke@tu-dresden.de>

Maintainer Lars Thielecke <lars.thielecke@tu-dresden.de>

Description Provides the necessary functions to identify and extract a selection of already available barcode constructs (Cornils, K. et al. (2014) <doi:10.1093/nar/gku081>) and freely choosable barcode designs from next generation sequence (NGS) data. Furthermore, it offers the possibility to account for sequence errors, the calculation of barcode similarities and provides a variety of visualisation tools (Thielecke, L. et al. (2017) <doi:10.1038/srep43249>).

Depends R (>= 3.4)

License LGPL

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Suggests BiocManager, testthat, knitr, rmarkdown

Imports methods, RColorBrewer, ape, ggnetwork, ggplot2, ggraph, igraph, network, phangorn, stringdist, visNetwork, reshape2, S4Vectors, shiny, ggtree, dplyr, VennDiagram, futile.logger, future, future.apply, tools, ShortRead, Biostrings

RoxygenNote 7.2.3

Collate 'BCdata-class.R' 'BCdata-class-methods.R' 'analysis_functions.R' 'data.R' 'error_correction_function.R' 'helper_functions.R' 'plot_functions.R' 'raw_data_processing.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2023-12-11 13:10:05 UTC

R topics documented:

`.createPatternFile` *Internal function*

Description

Creates a search file for a command line grep search.

Usage

```
.createPatternFile(bc_backbone, patterns_file)
```

Arguments

`bc_backbone`	a character string (barcode pattern).
`patterns_file`	a character string (file name)

`.generateColors` *Color list generation*

Description

Generates a collection of colors for a list of barcodes based on their identified minimum hamming distances.

Usage

```
.generateColors(minHD, type = "rainbow", alpha = 1)
```

Arguments

`minHD`	a numeric vector of all the minimum hamming distances.
`type`	a character string. Possible Values are "rainbow", "heat.colors", "topo.colors", "greens", "wild".
`alpha`	a numeric value between 0 and 1, modifies colour transparency.

.getBarcodeFilter *Internal function*

Description

Identifies the barcode positions within the barcode backbone and generates a awk command.

Usage

```
.getBarcodeFilter(wobble_pos)
```

Arguments

wobble_pos	a character string.

.getDiagonalIndex *Index Generation*

Description

Generates a matrix index to create a square triangular matrix.

Usage

```
.getDiagonalIndex(n)
```

Arguments

n	an integer indicating the size of the resulting index matrix.

Value

a locigal matrix of size n x n

.getMinDist *Distance calculation*

Description

Calculates the minimum distance to a set of predefined barcodes for a given list of barcode.

Usage

```
.getMinDist(BC_dat, ori_BCs, m = "hamming")
```

Arguments

BC_dat	a BCdat object
ori_BCs	a character vector containing barcodes to which the minimal hamming distance will be calculated.
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information).

.getWobblePos *getWobblePos*

Description

Extracts barcode positions.

Usage

```
.getWobblePos(bc_backbone = "")
```

Arguments

bc_backbone	a character vector.

.hex2rgbColor *Converts hex colors into gephi usable rgb colors*

Description

Converts hex colors into gephi usable rgb colors

Usage

```
.hex2rgbColor(colrs)
```

Arguments

colrs	a character vector containing a list of hex colors

Value

a color vector.

.revComp *DNA string manipulation*

Description

Converts a vector of character strings (DNA sequences) into its reverse complement.

Usage

```
.revComp(seq_dat)
```

Arguments

seq_dat	a character vector containing DNA sequences

.revComp_EqLength *DNA string manipulation for equal string sizes*

Description

Converts a vector of equally long character strings into its reverse complement.

Usage

```
.revComp_EqLength(seq_dat, word_length)
```

Arguments

seq_dat	a character vector.
word_length	an integer giving the word length.

.revComp_UneqLength *DNA string manipulation for unequal string sizes*

Description

Converts a vector of unequally long character strings into the reverse complement.

Usage

```
.revComp_UneqLength(seq_dat)
```

Arguments

seq_dat	A character vector.

.testDirIdentifier *Internal function*

Description

Checks directory paths for correctness and if nessesary corrects them.

Usage

```
.testDirIdentifier(s)
```

Arguments

s	a character string.

asBCdat *Data Type Conversion*

Description

Converts a data.frame into a BCdat object.

Usage

```
asBCdat(dat, label = "empty", BC_backbone = "none", resDir = getwd())
```

Arguments

dat	a data.frame object with two columns containing read counts and barcode sequences.
label	a optional character string used as label.
BC_backbone	a optional character string, describing the barcode backbone structure.
resDir	a optional character string, identifying the path to the results directory, default is current working directory.

Value

a BCdat object.

BCdat-class *BCdat class.*

Description

BCdat class.

Slots

`reads` data.frame containing barcode sequences and their corresponding read counts.

`results_dir` character string of the working directory path.

`label` character string identifying the particular experiment (will be part of the names of any file created).

`BC_backbone` character string of the used barcode design (also called barcode backbone).

BC_dat — *Barcode distribution of an example experiment.*

Description

A dataset containing an example BCdat object which consists of 98 barcode sequences and with no error correction yet.

Usage

```
BC_dat
```

Format

A S4 data object with the following slots:

class sequence overview

barcode read counts a data frame consisting of read counts and barcode sequences

results dir path to a directory for any kind of results

barcode backbone a string clarifying the barcode backbone structure

label character string, used as label for file names etc.

Details

BC_dat:

BC_dat_EC — *Barcode distribution of an example experiment.*

Description

A dataset containing an example BCdat object after error-correction which consists of 10 barcode sequences.

Usage

```
BC_dat_EC
```

Format

A S4 data object with the following slots:

class sequence overview

barcode read counts a data frame consisting of read counts and barcode sequences

results dir path to a directory for any kind of results

barcode backbone a string clarifying the barcode backbone structure

label character string, used as label for file names etc.

Details

BC_dat_EC:

`com_pair` *Compairing two BCdat Objects*

Description

Compairing two BCdat Objects

Usage

```
com_pair(BC_dat1 = NULL, BC_dat2 = NULL)
```

Arguments

`BC_dat1`	the first BCdat object.
`BC_dat2`	the second BCdat object.

Value

a list containing the shared and the unqiue barcodes.

`createGDF` *Creating a gdf File*

Description

createGDF creates a data file usable with the free graph visualisation tool gephi. The nodes represent barcodes and its respective size reflects the corresponding read counts. Edges between nodes indicate a distance between two barcodes of maximal `minD`. If `ori_BCs` is provided the node color refelects the distance of a particular barcode to one of the provided barcode sequences.

Usage

```
createGDF(
  BC_dat,
  minDist = 1,
  loga = TRUE,
  ori_BCs = NULL,
  col_type = "rainbow",
  m = "hamming"
)
```

Arguments

BC_dat	a BCdat object.
minDist	an integer value representing the maximal distance value for which the graph will contain edges.
loga	a logical value indicating the use or non-use of logarithmic read count values.
ori_BCs	a vector of character strings containing the barcode sequences (without the fixed positions of the barcode construct).
col_type	character sting, choosing one of the available color palettes.
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information).

Examples

```
## Not run:

data(BC_dat)
createGDFFile(BC_dat, minDist = 1, loga = TRUE, ori_BCs = NULL, col_type = "rainbow")


## End(Not run)
```

---

errorCorrection *Error Correction*

Description

Corrects a list of equally long (barcode) sequences. Based on calculated hamming distances as a measure of similarity, highly similar sequences are clustered together and the cluster label will be the respective sequence with the highest read count.

Usage

```
errorCorrection(
  BC_dat,
  maxDist,
  save_it = FALSE,
  cpus = 1,
  strategy = "sequential",
  m = "hamming",
  type = "standard",
  only_EC_BCs = TRUE,
  EC_analysis = FALSE,
  start_small = TRUE
)
```

Arguments

BC_dat	one or a list of BCdat objects, containing the necessary sequences.
maxDist	an integer value representing the maximal hamming distance for which it is allowed to cluster two sequences together.
save_it	a logical value. If TRUE the data will be saved as csv-file.
cpus	an integer value, in case multiple BCdat objects are provided a CPU number greater than one would allow for a parallelized calculation (one CPU per BCdat object).
strategy	since the future package is used for parallelisation a strategy has to be stated, the default is "sequential" (cpus = 1) and "multiprocess" (cpus > 1). It is not necessary to chose a certain strategy, since it will be adjusted accordingly to the number of cpus which were choosen. For further information please read future::plan() R-Documentation.
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information)
type	there are different error correction strategies avalable ("standard", "connectivity based", "graph based", "clustering").
only_EC_BCs	a logical value. If TRUE only informations about barcodes which are still present after error correction will be saved. Only meaningful if EC_analysis is set to TRUE.
EC_analysis	a logical value. If TRUE additional error correction details will be returned, which can also be visualised with the respective "error correction" plots.
start_small	a logical value. If TRUE, the error correcton type "standard" will cluster always the smallest highly similar BC with the BC of interest. IF FALSE, the error correcton type "standard" will adapt its cluster strategy and cluster always BC of interest with the most frequent highly similar BC.

Examples

```
data(BC_dat)
BC_dat_EC <- errorCorrection(BC_dat, maxDist = 8, save_it = FALSE, m = "hamming")
```

---

error_correction_circlePlot

Circle Plot

Description

creates a circle plot based on the additional data gathered by the error_correction function (EC_analysis needs to be set to TRUE). This function is intended to visualize the error correction procedure.

Usage

```
error_correction_circlePlot(edges, vertices)
```

Arguments

edges	a data frame containing edge definitions by two columns calles "from" and "to". Such data frame will be returned by the error_correction function with the EC_analysis parameter set to TRUE.
vertices	a data frame with at least one column containing a list of nodes (also returned by the error_correction function with the EC_analysis parameter set to TRUE)

Value

a ggplot2 object.

---

error_correction_clustered_HDs

Clustered HD Plot

Description

This function will create a jitter plot displaying the maximal distances within each of the barcode sequence clusters.

Usage

```
error_correction_clustered_HDs(datEC, size = 0.75)
```

Arguments

datEC	a BC_dat object, returned by the error_correction function with the EC_analysis parameter set to TRUE.
size	a numeric value, specifying the dot size.

Value

a ggplot2 object.

error_correction_treePlot *Tree Plot*

Description

creates a Tree Plot visualising of the barcode clustering as part of the error correction process.

Usage

```
error_correction_treePlot(edges, vertices)
```

Arguments

edges	a data frame containing edge definitions by two columns calles "from" and "to". Such data frame will be returned by the error_correction function with the EC_analysis parameter set to TRUE.
vertices	a data frame with at least one column containing a list of nodes (also returned by the error_correction function with the EC_analysis parameter set to TRUE)

Value

a ggplot2 object.

extractBarcodes *Barcode extraction*

Description

Extracts barcodes according to the given barcode design from a fastq file.

Usage

```
extractBarcodes(
  dat,
  label,
  results_dir = "./",
  mismatch = 0,
  indels = FALSE,
  bc_backbone,
  full_output = FALSE,
  cpus = 1,
  strategy = "sequential",
  wobble_extraction = TRUE,
  dist_measure = "hamming"
)
```

Arguments

dat	a ShortReadQ object.
label	a character string.
results_dir	a character string which contains the path to the results directory.
mismatch	an positive integer value, default is 0, if greater values are provided they indicate the number of allowed mismatches when identifing the barcode constructe.
indels	under construction.
bc_backbone	a character string or character vector describing the barcode design, variable positions have to be marked with the letter 'N'.
full_output	a logical value. If TRUE additional output files will be generated in order to identify errors.
cpus	an integer value, indicating the number of available cpus.
strategy	since the future package is used for parallelisation a strategy has to be stated, the default is "sequential" (cpus = 1) and "multiprocess" (cpus > 1). For further information please read future::plan() R-Documentation.
wobble_extraction	a logical value. If TRUE, single reads will be stripped of the backbone and only the "wobble" positions will be left.
dist_measure	a character value. If "bc_backbone = 'none'", single reads will be clustered based on a distance measure. Available distance methods are Optimal string aligment ("osa"), Levenshtein ("lv"), Damerau-Levenshtein ("dl"), Hamming ("hamming"), Longest common substring ("lcs"), q-gram ("qgram"), cosine ("cosine"), Jaccard ("jaccard"), Jaro-Winkler ("jw"), distance based on soundex encoding ("soundex"). For more detailed information see stringdist function of the stringdist-package for more information)

Value

one or a list of frequency table(s) of barcode sequences.

Examples

```
## Not run:

bc_backbone <- "ACTNNCGANNCTTNNCGANNCTTNNGGANNCTANNACTNNCGANNCTTNNCGANNCTTNNGGANNCTANNACTNNCGANN"
source_dir <- system.file("extdata", package = "genBaRcode")
dat <- ShortRead::readFastq(dirPath = source_dir, pattern = "test_data.fastq.gz")

extractBarcodes(dat, label = "test", results_dir = getwd(), mismatch = 0,
indels = FALSE, bc_backbone)


## End(Not run)
```

genBaRcode_app *Shiny App*

Description

Launches the corresponding shiny app.

Usage

```
genBaRcode_app(dat_dir = system.file("extdata", package = "genBaRcode"))
```

Arguments

dat_dir	a character string, identifying the path to one or more fast(q) files which shall be analysed, default is the path to the package inherent example fastq file

generateKirchenplot *Plotting a Kirchenplot*

Description

Generates a barplot based on read counts. If `ori_BCs` is provided the bar color reflects the distance between a particular barcode to one of the provided barcode sequences.

Usage

```
generateKirchenplot(
  BC_dat,
  ori_BCs = NULL,
  ori_BCs2 = NULL,
  loga = TRUE,
  col_type = NULL,
  m = "hamming",
  setLabels = c("BC-Set 1", "Rest", "BC-Set 2")
)
```

Arguments

BC_dat	a BCdat object.
ori_BCs	a vector of character strings containing known barcode sequences (without the fixed positions of the barcode construct).
ori_BCs2	a vector of character strings containing a 2nd set of known barcode sequences (also without the fixed positions).
loga	a logical value, indicating the use or non-use of logarithmic read count values.

col_type	character string, choosing one of the availabe color palettes ("rainbow", "heat.colors", "topo.colors", "greens", "wild" - see package "grDevices")
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information). If neither 'ori_BCs' nor 'ori_BCs2' are provided with input the choice of 'm' does not matter.
setLabels	a character vector, containing three strings serving as plot labels.

Value

a ggplot2 object

`generateTimeSeriesData` *Generating Time Series Data Object*

Description

Generates a matrix containing barcodes sequences as rows and consecutive measurements at columns. It serves as the necessary data object for the plotting function 'plotTimeSeries'.

Usage

```
generateTimeSeriesData(BC_dat_list)
```

Arguments

BC_dat_list	a list of BCdat objects.

Value

a data.frame containing every identified barcode and its read count per time point/measurement.

`getBackbone` *Accessing the Barcode Backbone slot of a BCdat objects.*

Description

Accessing the Barcode Backbone slot of a BCdat objects.

Usage

```
getBackbone(object)
```

Arguments

object	a BCdat object.

Value

A character string.

Examples

```
data(BC_dat)
getBackbone(BC_dat)
```

---

getBackboneSelection *Predefined Barcode Backbone Sequences*

Description

allows the user to choose between predefined backbone sequences. Excecution of the function without any parameter value will display all available backbone sequences. The id parameter will accept the name of the backbone or the rownumber of the shown selection.

Usage

```
getBackboneSelection(id = NULL)
```

Arguments

id	an integer or character value in order to choose a specific backbone.

Value

a character string.

Examples

```
getBackboneSelection()
getBackboneSelection(2)
getBackboneSelection("BC32-Venus")
```

getLabel *Accessing the Label slot of a BCdat objects.*

Description

Accessing the Label slot of a BCdat objects.

Usage

```
getLabel(object)
```

Arguments

object	a BCdat object.

Value

A character string.

Examples

```
data(BC_dat)
getLabel(BC_dat)
```

getReads *Accessing the Read-Count slot of a BCdat objects.*

Description

Accessing the Read-Count slot of a BCdat objects.

Usage

```
getReads(object)
```

Arguments

object	a BCdat object.

Value

A data.frame containing the read count table of the object paramter.

Examples

```
data(BC_dat)
getReads(BC_dat)
```

getResultsDir *Accessing the Results Directory slot of a BCdat objects.*

Description

Accessing the Results Directory slot of a BCdat objects.

Usage

```
getResultsDir(object)
```

Arguments

object	a BCdat object.

Value

A character string.

Examples

```
data(BC_dat)
getResultsDir(BC_dat)
```

ggplotDistanceGraph *Plotting a Distance Network*

Description

ggplotDistanceGraph will create a graph-like visualisation (ripple plot) of the corresponding barcode sequences and their similarity based on the ggplot2 and the ggnetwork packages. The nodes represent the barcode sequences and their respective size reflects the corresponding read counts. Edges between nodes indicate a distance between two barcodes of maximal `minDist`. If `ori_BCs` is provided the node color also refelects the distance of a particular barcode to one of the initial barcodes.

Usage

```
ggplotDistanceGraph(
  BC_dat,
  minDist = 1,
  loga = TRUE,
  ori_BCs = NULL,
  lay = "fruchtermanreingold",
  complete = FALSE,
```

```
  col_type = "rainbow",
  m = "hamming",
  scale_nodes = 1,
  scale_edges = 1,
  legend_size = 4
)
```

Arguments

BC_dat	a BCdat object.
minDist	an integer value representing the maximal distance for which the graph will contain edges.
loga	a logical value, indicating the use or non-use of logarithmic read count values.
ori_BCs	a vector of character strings containing the barcode sequences (without the fixed positions of the barcode construct).
lay	a character string, identifying the prefered layout algorithm (see ggnetwork layout option, "?gplot.layout"). Default value is "fruchtermanreingold", but possible are also "circle", "eigen", "kamadakawai", "spring" and many more. Or the user provides a two-column matrix with as many rows as there are nodes in the network, in which case the matrix is used as nodes coordinates.
complete	a logical value. If TRUE, every node will have at least one edge.
col_type	a character sting, choosing one of the available color palettes ("rainbow", "heat.colors", "topo.colors", "greens", "wild" - see package "grDevices").
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information).
scale_nodes	a numeric value, scaling the node size.
scale_edges	a numeric value, scaling the edge size.
legend_size	a numeric value, scaling the legend symbol size, if legend_size equals 0, the legend will be dismissed.

Value

a ggplot2 object

Examples

```
## Not run:

data(BC_dat)
ggplotDistanceGraph(BC_dat, minDist = 1, loga = TRUE, ori_BCs = NULL, lay = "fruchtermanreingold",
complete = FALSE, col_type = "rainbow")


## End(Not run)
```

ggplotDistanceGraph_EC *Plotting a Distance Network (error correction)*

Description

ggplotDistanceGraph will create a graph-like visualisation (ripple plot) of the corresponding barcode sequences and their similarity based on the ggplot2 and the ggnetwork packages. The nodes represent the barcode sequences and their respective size reflects the corresponding read counts. Edges between nodes indicate a distance between two barcodes of maximal `minDist`. If `ori_BCs` is provided the node color also refelects the distance of a particular barcode to one of the initial barcodes.

Usage

```
ggplotDistanceGraph_EC(
  BC_dat,
  BC_dat_EC,
  minDist = 1,
  loga = TRUE,
  equal_node_sizes = TRUE,
  BC_threshold = NULL,
  ori_BCs = NULL,
  lay = "fruchtermanreingold",
  complete = FALSE,
  col_type = "rainbow",
  m = "hamming",
  scale_nodes = 1,
  scale_edges = 1
)
```

Arguments

`BC_dat`	a BCdat object.
`BC_dat_EC`	the error corrected BCdat object (the EC_analysis parameter needs to be set to TRUE).
`minDist`	an integer value representing the maximal distance for which the graph will contain edges.
`loga`	a logical value, indicating the use or non-use of logarithmic read count values.
`equal_node_sizes`	a logical value. If TRUE, every node will have the same size.
`BC_threshold`	a nnumeric value, limiting the number of barcodes for which their error correction "history" will be colored (if BC_threshold = 5 then the five biggest barcodes will be evaluated)

ori_BCs	a vector of character strings containing barcode sequences (without the fixed positions of the barcode construct). Similar to BC_threshold but allowing for barcode identification via sequence.
lay	a character string, identifying the prefered layout algorithm (see ggnetwork layout option).
complete	a logical value. If TRUE, every node will have at least one edge.
col_type	a character sting, choosing one of the available color palettes.
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information).
scale_nodes	a numeric value, scaling the node size.
scale_edges	a numeric value, scaling the edge size.

Value

a ggplot2 object

---

hybridsIdentification *Identifies hybrid barcodes*

Description

Experimental function to identify hybrid barcodes which can occure due to unfinished synthesis of a template in-between PCR cycles.

Usage

```
hybridsIdentification(dat, min_seq_length = 10)
```

Arguments

dat	a character vector containing barcode sequences or a BCdat object.
min_seq_length	a positive integer value indicating the minimal length of the two barcodes which give rise to a hybrid barcode.

Value

a hybrid-free frequency table of barcode sequences

plotClusterGgTree *Plotting a Cluster ggTree*

Description

Generates a tree plot based on a herachical clustering of the complete distance matrix.

Usage

```
plotClusterGgTree(BC_dat, tree_est = "NJ", type = "rectangular", m = "hamming")
```

Arguments

BC_dat	a BCdat object.
tree_est	a character string, indicating the particular cluster algorithm, possible algorithms are "Neighbor-Joining" ("NJ") and "Unweighted Pair Group Method" ("UPGMA").
type	a character string, the graph layout style ('rectangular', 'slanted', 'fan', 'circular', 'radial', 'equal_angle' or 'daylight').
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information).

Value

a ggtree object.

Examples

```
## Not run:
data(BC_dat)
plotClusterGgTree(BC_dat, tree_est = "UPGMA", type = "circular")

## End(Not run)
```

plotClusterTree *Plotting a Cluster Tree*

Description

Generates a tree plot based on a herachical clustering of the complete distance matrix.

Usage

```
plotClusterTree(
  BC_dat,
  tree_est = "NJ",
  type = "unrooted",
  tipLabel = FALSE,
  m = "hamming"
)
```

Arguments

BC_dat	a BCdat object.
tree_est	a character string, indicating the particular cluster algorithm, possible algorithms are "Neighbor-Joining" ("NJ") and "Unweighted Pair Group Method" ("UPGMA").
type	a character string, the graph layout style ("unrooted", "phylogram", "cladogram", "fan", "radial").
tipLabel	a logical value, indicating the use of labeled tree leaves.
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information).

---

plotDistanceIgraph *Plotting a Distance Network*

Description

plotDistanceIgraph will create a graph-like visualisation (ripple plot) of the corresponding barcode sequences and their similarity based on the igraph package. The nodes represent the barcode sequences and their respective size reflects the corresponding read counts. Edges between nodes indicate a distance between two barcodes of maximal `minD`. If `ori_BCs` is provided the node color also refelects the distance of a particular barcode to one of the initial barcodes.

Usage

```
plotDistanceIgraph(
  BC_dat,
  minDist = 1,
  loga = TRUE,
  ori_BCs = NULL,
  threeD = FALSE,
  complete = FALSE,
  col_type = "rainbow",
  leg_pos = "left",
```

```
  inset = -0.125,
  title = "Distance",
  m = "hamming"
)
```

Arguments

BC_dat	a BCdat object.
minDist	an integer value representing the maximal distance value for which the graph will contain edges.
loga	a logical value, indicating the use or non-use of logarithmic read count values.
ori_BCs	a vector of character strings containing the barcode sequences (without the fixed positions of the barcode construct).
threeD	a logical value to chose between 2D and 3D visualisation.
complete	a logical value. If TRUE, every node will have at least one edge.
col_type	a character sting, choosing one of the available color palettes.
leg_pos	a character string, containing the position of the legend (e.g. topleft), if NULL no legend will be plotted
inset	a numeric value, specifying the distance from the margins as a fraction of the plot region
title	a character string, containing the legend title
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information).

Value

an igraph object.

---

plotDistanceVisNetwork

Plotting a Distance Network

Description

plotDistanceVisNetwork will create a graph-like visualisation (ripple plot) of the corresponding barcode sequences and their similarity based on the ggplot2 and the ggnetwork packages. The nodes represent the barcode sequences and their respective size reflects the corresponding read counts. Edges between nodes indicate a distance between two barcodes of maximal `minDist`. If `ori_BCs` is provided the node color also refelects the distance of a particular barcode to one of the given barcodes.

Usage

```
plotDistanceVisNetwork(
  BC_dat,
  minDist = 1,
  loga = TRUE,
  ori_BCs = NULL,
  complete = FALSE,
  col_type = "rainbow",
  m = "hamming"
)
```

Arguments

BC_dat	a BCdat object.
minDist	an integer value representing the maximal distance value for which the graph will contain edges.
loga	a logical value indicating the use or non-use of logarithmic read count values.
ori_BCs	a vector of character strings containing the barcode sequences (without the fixed positions of the barcode construct).
complete	a logical value. If TRUE, every node will have at least one edge.
col_type	a character sting, choosing one of the available color palettes.
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information).

Value

a visNetwork object.

---

plotDistanceVisNetwork_EC

Plotting a Distance Network (error correction)

Description

plotDistanceVisNetwork will create a graph-like visualisation (ripple plot) of the corresponding barcode sequences and their similarity based on the ggplot2 and the ggnetwork packages. The nodes represent the barcode sequences and their respective size reflects the corresponding read counts. Edges between nodes indicate a distance between two barcodes of maximal `minDist`. If `ori_BCs` is provided the effects of the error correction function will be color-coded only for those sequences.

Usage

```
plotDistanceVisNetwork_EC(
  BC_dat,
  BC_dat_EC,
  minDist = 1,
  loga = TRUE,
  equal_node_sizes = TRUE,
  BC_threshold = NULL,
  ori_BCs = NULL,
  complete = FALSE,
  col_type = "rainbow",
  m = "hamming"
)
```

Arguments

BC_dat	a BCdat object.
BC_dat_EC	the corresponding error corrected BCdat object (EC_analysis has to be TRUE)
minDist	an integer value representing the maximal distance value for which the graph will contain edges.
loga	a logical value indicating the use or non-use of logarithmic read count values.
equal_node_sizes	a logical value. If TRUE, every node will have the sames size.
BC_threshold	an integer value representing the number of barcodes for which the color-coding should be applied (starting with the barcodes with the most read counts).
ori_BCs	a vector of character strings containing the barcode sequences (without the fixed positions of the barcode construct).
complete	a logical value. If TRUE, every node will have at least one edge.
col_type	a character sting, choosing one of the available color palettes.
m	a character string, Method for distance calculation, default value is Hamming distance. Possible values are "osa", "lv", "dl", "hamming", "lcs", "qgram", "cosine", "jaccard", "jw", "soundex" (see stringdist function of the stringdist-package for more information).

Value

a visNetwork object.

plotNucFrequency *Plotting Nucleotide Frequency*

Description

Creates a plot visualising the nucleotide frequency within the entire fastq file.

Usage

```
plotNucFrequency(source_dir, file_name)
```

Arguments

source_dir	a character string containing the path to the sequencing file.
file_name	a character string containng the name of the sequencing file.

Value

a ggplot2 object.

plotQualityScoreDis *Plotting Quality Score Distribution*

Description

Creates a plot of the quality values accommodated by the fastq file.

Usage

```
plotQualityScoreDis(source_dir, file_name, type = "median", rel = FALSE)
```

Arguments

source_dir	a character string of the path to the source directory.
file_name	a character string of the file name.
type	a character string, possible values are "mean" and "median".
rel	a logical value. If TRUE the y-axis will show relative frequency instead of the absolut counts.

Value

a ggplot2 object.

Examples

```
## Not run:

source_dir <- system.file("extdata", package = "genBaRcode")

plotQualityScoreDis(source_dir, file_name = "test_data.fastq", type = "mean")


## End(Not run)
```

`plotQualityScorePerCycle` *Plotting Quality Score per Cycle*

Description

Visualises the mean, median, 25

Usage

```
plotQualityScorePerCycle(source_dir, file_name)
```

Arguments

`source_dir`	a character string containing the path to the sequencing file.
`file_name`	a character string containng the name of the sequencing file.

Value

a ggplot2 object.

`plotReadFrequencies` *Plotting a Barplot*

Description

Generates a barplot visualising the abundances of unique read count frequencies.

Usage

```
plotReadFrequencies(
  BC_dat,
  b = 30,
  bw = NULL,
  show_it = FALSE,
  log = FALSE,
  dens = FALSE
)
```

Arguments

BC_dat	a BCdat object.
b	an integer value, defining the number of bins. Overridden by bw. Defaults to 30. (see '?ggplot2::geom_histogram')
bw	an integer value, defining the width of the bins.
show_it	a logical vaue. If TRUE, the respective values are printed on the console?
log	a logical vaue. If TRUE, the y-axis will be on a log scale.
dens	a logical vaue. If TRUE, the density of the read frequencies will be plotted.

Value

ggplot2 object

Examples

```
data(BC_dat)
plotReadFrequencies <- function(BC_dat, b = 10, show_it = TRUE)
```

---

plotSeqLogo *Plots a sequence logo*

Description

Plots a sequence logo

Usage

```
plotSeqLogo(BC_dat, colrs = NULL)
```

Arguments

BC_dat	a chatacter vector or BCdat object containing the respective sequences
colrs	a character vector containing the desired colors for the nucleotides A, T, C, G and N (in that order)

Value

a ggplot2 object

plotTimeSeries *Plotting Time Series Data*

Description

Uses the result of the generateTimeSeriesData function as inout and generates a visualisation of the clonal contributions over a number of given time points (similar to a stacked barplot).

Usage

```
plotTimeSeries(
  ov_dat,
  colr = NULL,
  tp = NULL,
  x_label = "time",
  y_label = "contribution"
)
```

Arguments

ov_dat	a numeric matrix consisting of all time points as columns and all barcode sequences as rows and the corresponding read counts as numerical values (see function generateTimeSeriesData()).
colr	a vector of character strings identifying a certain color palette.
tp	a numeric vector containing the time points of measurement (in case of unequally distributed time points).
x_label	a character string providing the x-axis label.
y_label	a character string providing the y-axis label.

Value

a ggplot2 object.

Examples

```
ov_dat <- matrix(round(runif(1:100, min = 0, max = 1000)), ncol = 5)
rownames(ov_dat) <- paste("barcode", 1:20)
plotTimeSeries(ov_dat)
```

plotVennDiagram *Plotting a VennDiagram*

Description

plotVennDiagramm will create a Venn Diagram ans is based on the VennDiagram package. It accepts a list of BCdat objects and will return a ggplot2 output object.

Usage

```
plotVennDiagram(
  BC_dat,
  alpha_value = 0.4,
  colrs = NA,
  border_color = NA,
  plot_title = "",
  legend_sort = NULL,
  annotationSize = 5
)
```

Arguments

BC_dat	a list of BCdat objects.
alpha_value	color transparency value [0-1].
colrs	a character vector containing the desired colors, if NA the colors will be chosen automatically.
border_color	a character value specifying the desired border color, if NA no border will be drawn.
plot_title	a character value.
legend_sort	a character or factor vector in case the order of legend items needs to be changed.
annotationSize	an integer value specifying the venn diagramm internal text size.

Value

ggplot2 object.

prepareDatObject *Data Object Preparation*

Description

generates BCdat object after barcode backbone identification.

Usage

```
prepareDatObject(dat, results_dir, label, bc_backbone, min_reads, save_it)
```

Arguments

dat	a tbl_df object (e.g. created by dplyr::count)
results_dir	a character string which contains the path to the results directory.
label	a character string which serves as a label for every kind of created output file.
bc_backbone	a character string describing the barcode design, variable positions have to be marked with the letter 'N'.
min_reads	positive integer value, all extracted barcode sequences with a read count smaller than min_reads will be excluded from the results
save_it	a logical value. If TRUE, the raw data will be saved as a csv-file.

Value

a BCdat object.

processingRawData *Data processing*

Description

Reads the corresponding fast(a/q) file(s), extracts the defined barcode constructs and counts them. Optionally, a Phred-Score based quality filtering will be conducted and the results will be saved within a csv file.

Usage

```
processingRawData(
  file_name,
  source_dir,
  results_dir = NULL,
  mismatch = 0,
  indels = FALSE,
  label = "",
```

```
  bc_backbone,
  bc_backbone_label = NULL,
  min_score = 30,
  min_reads = 2,
  save_it = TRUE,
  seqLogo = FALSE,
  cpus = 1,
  strategy = "sequential",
  full_output = FALSE,
  wobble_extraction = TRUE,
  dist_measure = "hamming"
)
```

Arguments

file_name	a character string or a character vector, containing the file name(s).
source_dir	a character string which contains the path to the source files.
results_dir	a character string which contains the path to the results directory. If no value is assigned the source_dir will automatically also become the results_dir.
mismatch	an positive integer value, default is 0, if greater values are provided they indicate the number of allowed mismtaches when identifying the barcode constructes.
indels	a logical value. If TRUE the chosen number of mismatches will be interpreted as edit distance and allow for insertions and deletions as well (currently under construction).
label	a character string which serves as a label for every kind of created output file.
bc_backbone	a character string describing the barcode design, variable positions have to be marked with the letter 'N'. If only a clustering of the sequenced reads should be applied bc_backbone is expecting the string "none" and the mismatch parameter will then be interpreted as maximum dissimilarity for which two reads will be clustered together.
bc_backbone_label	a character vector, an optional list of barcode backbone names serving as additional identifier within file names and BCdat labels. If not provided ordinary numbers will serve as alternative.
min_score	a positive integer value, all fastq sequence with an average score smaller then min_score will be excluded, if min_score = 0 there will be no quality score filtering
min_reads	positive integer value, all extracted barcode sequences with a read count smaller than min_reads will be excluded from the results
save_it	a logical value. If TRUE, the raw data will be saved as a csv-file.
seqLogo	a logical value. If TRUE, the sequence logo of the entire NGS file will be generated and saved.
cpus	an integer value, indicating the number of available cpus.
strategy	since the future package is used for parallelisation a strategy has to be stated, the default is "sequential" (cpus = 1) and "multisession" (cpus > 1). For further information please read future::plan() R-Documentation.

full_output a logical value. If TRUE, additional output files will be generated.

wobble_extraction
a logical value. If TRUE, single reads will be stripped of the backbone and only the "wobble" positions will be left.

dist_measure a character value. If "bc_backbone = 'none'", single reads will be clustered based on a distance measure. Available distance methods are Optimal string aligment ("osa"), Levenshtein ("lv"), Damerau-Levenshtein ("dl"), Hamming ("hamming"), Longest common substring ("lcs"), q-gram ("qgram"), cosine ("cosine"), Jaccard ("jaccard"), Jaro-Winkler ("jw"), distance based on soundex encoding ("soundex"). For more detailed information see stringdist function of the stringdist-package for more information)

Value

a BCdat object which will include read counts, barcode sequences, the results directory and the search barcode backbone.

Examples

```
## Not run:
bc_backbone <- "ACTNNCGANNCTTNNCGANNCTTNNGGANNCTANNACTNNCGANNCTTNNCGANNCTTNNGGANNCTANNACTNNCGANN"

source_dir <- system.file("extdata", package = "genBaRcode")

BC_dat <- processingRawData(file_name = "test_data.fastq.gz", source_dir,
          results_dir = "/my/test/directory/", mismatch = 2, label = "test", bc_backbone,
          min_score = 30, indels = FALSE, min_reads = 2, save_it = FALSE, seqLogo = FALSE)

## End(Not run)
```

---

qualityFiltering *Quality Filtering*

Description

Excludes all sequences of a given fastq file below a certain quality value.

Usage

```
qualityFiltering(file_name, source_dir, min_score = 30)
```

Arguments

file_name a character string containing the name of the source file.

source_dir a character string containing the path to the source directory.

min_score an integer value representing the minimal average phred score a read has to achieve in order to be accepted.

Value

a ShortRead object.

Examples

```
## Not run:
source_dir <- system.file("extdata", package = "genBaRcode")
qualityFiltering(file_name = "test_data.fastq.gz", source_dir,
results_dir = getwd(), min_score = 30)

## End(Not run)
```

`readBCdat` *Data Input*

Description

Reads a data table (csv-file) and returns a BCdat objects.

Usage

```
readBCdat(path, label = "", BC_backbone = "", file_name, s = ";")
```

Arguments

`path`	a character string containing the path to a saved read count table (two columns containing read counts and barcode sequences).
`label`	a character string containing a label of the data set.
`BC_backbone`	a character string containing the barcode structure information.
`file_name`	a character string containing the name of the file to read in.
`s`	a character value, identifying the column separating char.

Value

a BCdat object.

setBackbone *Replacing the Barcode Backbone slot of a BCdat objects.*

Description

Replacing the Barcode Backbone slot of a BCdat objects.

Usage

```
setBackbone(object, value)
```

Arguments

object	a BCdat object.
value	a character string consisting of exclusively IUPAC-nucleotide-code conform letters.

Value

a BCdat object.

Examples

```
data(BC_dat)
new_backbone <- getBackboneSelection("BC32-T-Sapphire")
BC_dat_new <- setBackbone(BC_dat, new_backbone)
```

setLabel *Replacing the Label slot of a BCdat objects.*

Description

Replacing the Label slot of a BCdat objects.

Usage

```
setLabel(object, value)
```

Arguments

object	a BCdat object.
value	a character string.

Value

a BCdat object.

Examples

```
data(BC_dat)
new_label <- "foo-bar"
BC_dat_new <- setLabel(BC_dat, new_label)
```

---

setReads *Replacing the Read-Count slot of a BCdat objects.*

Description

Replacing the Read-Count slot of a BCdat objects.

Usage

```
setReads(object, value)
```

Arguments

object	a BCdat object.
value	a data.frame caontaining two columns called "read_count" and "barcode".

Value

a BCdat object.

Examples

```
data(BC_dat)
require("dplyr")

bcs <- unlist(lapply(1:20, function(x) {
       c("A", "C", "T", "G") %>% sample(replace = TRUE, size = 32) %>% paste0(collapse = "")
       }))
new_read_count_table <- data.frame(read_count = sample(1:1000, size = 20), barcode = bcs)
BC_dat_new <- setReads(BC_dat, new_read_count_table)
```

`setResultsDir` *Replacing the Results Directory slot of a BCdat objects.*

Description

Replacing the Results Directory slot of a BCdat objects.

Usage

```
setResultsDir(object, value)
```

Arguments

`object`	a BCdat object.
`value`	a character string of an existing path.

Value

a BCdat object.

Examples

```
data(BC_dat)
new_path <- getwd()
BC_dat_new <- setResultsDir(BC_dat, new_path)
```

Index