Package 'genBaRcode'

August 10, 2018

Title Analysis and Visualization Tools for Genetic Barcode Data

Version 1.1.0

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 (>= 3.4.0)

License LGPL

Encoding UTF-8

LazyData true

Suggests testthat

Imports methods, Biostrings, RColorBrewer, ShortRead, ape, ggnetwork, ggplot2, igraph, network, phangorn, stringdist, visNetwork, reshape2, S4Vectors, shiny, ggseqlogo, ggtree, foreach, doParallel, dplyr, VennDiagram, futile.logger

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-08-10 15:20:07 UTC

R topics documented:

`asBCdat` *Data Type Conversion*

Description

Converts a data.frame into a BCdat object.

Usage

```
asBCdat(dat, label = "without_label", mask = "", 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.
`mask`	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

S4 data class containing every relevant information.

Value

a BCdat object.

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).

`mask` character string of the used barcode design.

`BC_dat` *Barcode distribution of an example experiment.*

Description

A dataset containing an example BCdat object which consists of 301 barcode sequences.

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 layout a string clarifying the barcode backbone structure

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

Details

BC_dat:

checkBarcodeData *checkBarcodeData*

Description

Checks data slots of BCdat object for correctness.

Usage

```
checkBarcodeData(object)
```

Arguments

object	a BCdat object.

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, m = "hamming")
```

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).
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

```
errorCorrection(BC_dat, maxDist = 8, save_it = FALSE, m = "hamming")
```

errorCorrection_multiple *Error Correction for multiple objects*

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_multiple(BC_dat, maxDist, save_it = FALSE, cpus = 1,
  m = "hamming")
```

Arguments

BC_dat	a BCdat object, 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).
m	a character string. Method for distance calculation, default 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 list of BCdat objects.

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)
```

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.

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")

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


## End(Not run)
```

---

extractBarcodes_multiple

Barcode extraction for multiple backbones

Description

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

Usage

```
extractBarcodes_multiple(dat, label, results_dir, mismatch = 0,
  indels = FALSE, bc_backbone, full_output)
```

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.

Value

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

---

extractBarcodes_single

Barcode extraction for a single backbone

Description

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

Usage

```
extractBarcodes_single(dat, label, results_dir, mismatch = 0,
  indels = FALSE, bc_backbone, full_output = FALSE, cpus = 1)
```

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.

Value

one frequency table of barcode sequences.

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 sting, choosing one of the availabe color palettes, e.g. rainbow.
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).
setLabels	a character vector, containing three strings serving as plot labels.

Value

a ggplot2 object

Examples

```
data(BC_dat)
generateKirchenplot(BC_dat, ori_BCs, loga = TRUE, col_type = NULL)
```

`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` *Predefined Barcode Backbone Sequences*

Description

allows one 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

```
getBackbone(id = NULL)
```

Arguments

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

Value

a character string.

`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",
  outline = 0.1, 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).
`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.
`outline`	a nnumeric value which adjusts the thickness of the black outline of each node.
`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)
```

---

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 = 2)
```

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

Examples

```
data(BC_dat)
hybridsIdentification(BC_dat, min_seq_length = 2)
```

ori_BCs *List of Barcodes.*

Description

An object called ori_BCs containing specific predefined barcode sequences.

Usage

```
ori_BCs
```

Format

a character vector:

Details

ori_BCs:

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

```
data(BC_dat)
plotClusterGgTree(BC_dat, tree_est = "UPGMA", type = "circular")
```

`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).

Examples

```
data(BC_dat)
plotClusterTree(BC_dat, tree_est = "UPGMA", type = "unrooted", tipLabel = FALSE)
```

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.

Examples

```
data(BC_dat)
plotDistanceIgraph(BC_dat, minDist = 1, loga = TRUE, ori_BCs, threeD = FALSE,
complete = FALSE, col_type = "rainbow")
```

---

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.

Examples

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

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)
```

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".

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, show_it = FALSE, log = FALSE)
```

Arguments

BC_dat	a BCdat object.
b	an integer value, defining the number of 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.

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

Examples

```
data(BC_dat)
plotSeqLogo(BC_dat)
```

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)
```

plotVennDiagramm *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

```
plotVennDiagramm(BC_dat, alpha_value = 0.4, colrs = NA, border_color = NA,
  plot_title = "", legend_sort = NA, 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(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, mismatch = 0,
  indels = FALSE, label = "", bc_backbone, bc_backbone_label = "",
  min_score = 30, min_reads = 2, save_it = TRUE, seqLogo = FALSE,
  cpus = 1, full_output = FALSE)
```

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

Value

a BCdat object which includes reads, seqs, directories, masks.

Examples

```
## Not run:

bc_backbone <- "ACTNNCGANNCTTNNCGANNCTTNNGGANNCTANNACTNNCGANNCTTNNCGANNCTTNNGGANNCTANNACTNNCGANN"

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

processingRawData(file_name = "test_data.fastq", source_dir, results_dir = getwd(), mismatch = 0,
label = "test", bc_backbone, min_score = 30, indels = FALSE,
min_reads = 2, save_it = TRUE, 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, results_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.
results_dir	a character string containing the path to the directory of the results.
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", source_dir, results_dir = getwd(), min_score = 30)

## End(Not run)
```

`readBCdat` *Data Input*

Description

Reads in a data table and returns a BCdat objects.

Usage

```
readBCdat(path = "./", label = "", mask = "", 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.
`mask`	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.

Index