Package 'BadRegionFinder'

November 11, 2026

Type Package

Title BadRegionFinder: an R/Bioconductor package for identifying regions with bad coverage

Version 1.40.0

Date 2016-03-07

Author Sarah Sandmann

Maintainer Sarah Sandmann <sarah.sandmann@uni-muenster.de>

Description BadRegionFinder is a package for identifying regions with a bad, acceptable and good coverage in sequence alignment data available as bam files. The whole genome may be considered as well as a set of target regions. Various visual and textual types of output are available.

License LGPL-3

Imports VariantAnnotation, Rsamtools, biomaRt, GenomicRanges, S4Vectors, utils, stats, grDevices, graphics

Suggests BSgenome.Hsapiens.UCSC.hg19

biocViews Coverage, Sequencing, Alignment, WholeGenome, Classification

NeedsCompilation no

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

git_branch RELEASE_3_23

git_last_commit 9c97404

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

BadRegionFinder-package 2
determineCoverage 4
determineCoverageQuality 6

determineQuantiles 8
determineRegionsOfInterest 9
plotDetailed 11
plotSummary 13
plotSummaryGenes 14
reportBadRegionsDetailed 16
reportBadRegionsGenes 19
reportBadRegionsSummary 21

Index **24**

`BadRegionFinder-package`
BadRegionFinder: an R/Bioconductor package for identifying regions with bad coverage

---

Description

BadRegionFinder is a package for identifying regions with a bad, acceptable and good coverage in sequence alignment data available as bam files. The whole genome may be considered as well as a set of target regions. Various visual and textual types of output are available.

Details

This package was not yet installed at build time.
In the use case of targeted sequencing it is most important to design the set of used primers in a way that the targeted regions are sequenced with a sufficient coverage. Yet, due to e.g. high GC-content the aimed at coverage may not always be obtained. Thus, a tool performing a detailed coverage analysis comparing many samples at a time – and not considering all available samples individually – appears to be most useful. Furthermore, with regards to reads mapping off target, it seems helpful to have a tool for investigating those regions, which show a relatively high coverage, but which were not originally targeted.

BadRegionFinder is a package for classifying a selection of regions or the whole genome into the user-definable categories of bad, acceptable and good coverage in any sequence alignment data available as bam files. Various visual and textual types of output are available including detailed output files considering every base that is or should be covered and an overview file considering the coverage of the different genes that were targeted.

Index: This package was not yet installed at build time.
The package contains a function performing the coverage determination - `determineCoverage` (switch for whole-genome- and target region analyses). The actual classification of the coverage is performed by the function `determineCoverageQuality`. If any subsets of regions are of interest, these may be selected by the function `determineRegionsOfInterest`.

There are three different forms of textual reports available: a summary variant (`reportBadRegionsSummary`), a detailed variant (`reportBadRegionsDetailed`) and a summary variant focussing on the coverage of the genes (`reportBadRegionsGenes`).

Furthermore, there exist three different forms of visual reports: a summary variant (`plotSummary`), a detailed variant (`plotDetailed`) and a summary variant visualizing the coverage of the genes as a barplot (`plotSummaryGenes`).

Additionally, BadRegionFinder may be used to determine user-definable, basewise quantiles over all samples at any position (`determineQuantiles`).

Author(s)

Sarah Sandmann

Maintainer: Sarah Sandmann <sarah.sandmann@uni-muenster.de>

References

More information on the bam format can be found at: http://samtools.github.io/hts-specs/SAMv1.pdf

See Also

`determineCoverage`, `determineCoverageQuality`, `determineRegionsOfInterest`, `reportBadRegionsSummary`, `reportBadRegionsDetailed`, `reportBadRegionsGenes`, `plotSummary`, `plotDetailed`, `plotSummaryGenes`, `determineQuantiles`

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")

threshold1 <- 20
threshold2 <- 100
percentage1 <- 0.80
percentage2 <- 0.90
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)

coverage_summary <- determineCoverage(samples, bam_input, targetRegions,
                                      output, TRonly = FALSE)
coverage_indicators <- determineCoverageQuality(threshold1, threshold2,
                                                percentage1, percentage2,
                                                coverage_summary)
badCoverageSummary <- reportBadRegionsSummary(threshold1, threshold2,
                                              percentage1, percentage2,
                                              coverage_indicators, "", output)
coverage_indicators_temp <- reportBadRegionsDetailed(threshold1, threshold2,
                                                     percentage1, percentage2,
                                                     coverage_indicators, "",
```

```
                                                        samples, output)
badCoverageOverview <- reportBadRegionsGenes(threshold1, threshold2, percentage1,
                                             percentage2, badCoverageSummary,
                                             output)

plotSummary(threshold1, threshold2, percentage1, percentage2,
            badCoverageSummary, output)
plotDetailed(threshold1, threshold2, percentage1, percentage2,
             coverage_indicators_temp, output)
plotSummaryGenes(threshold1, threshold2, percentage1, percentage2,
                 badCoverageOverview, output)

quantiles <- c(0.5)
coverage_summary2 <- determineQuantiles(coverage_summary, quantiles, output)
```

`determineCoverage` *Determines the coverage (recommended for whole-genome analyses)*

Description

BadRegionFinder performs a coverage analysis of various samples at a time. The first, essential step of the analysis pipeline – the coverage determination – is performed by the function `determineCoverage`. Thereby, the whole genome is scanned and wherever a covered base is registered or an originally targeted base is detected, detailed information concerning this position is written out.

Usage

```
determineCoverage(samples, bam_input, targetRegions, output, TRonly)
```

Arguments

`samples`	Data frame object containing the names of the samples to be analyzed (in one column).
`bam_input`	Folder containing the alignment data in bam- and bai format or BamFileList.
`targetRegions`	Data frame- or GRanges object containing the target regions to be analyzed (chromosome: first column, start position: second column and end position: third column).
`output`	The folder to write the output files into. If an empty string is provided, no files are written out.
`TRonly`	Boolean, indicating whether the coverage of the whole genome should be analyzed and reported (`FALSE`) or the coverage of the target regions only (`TRUE`).

Details

The coverage which is determined by the function `determineCoverage` contains different steps:

For every sample that is defined in `samples`, the coverage is determined using the function `coverage` ("Determine Coverage"). To combine information on the coverage with information on whether a set of bases were originally targeted by some sequencing experiment, the `targetRegions` get processed ("Determine target bases"). Finally, the information gets combined ("Combine information"): Those positions where no sample shows any coverage and no target base is registered, are summed up. All other positions are reported basewise.

Files get written out in the form: "Summary_chr<chromosomename>.txt".

As sequencing does often not mean whole-genome- or whole-exome sequencing, but targeted sequencing, the function `determienCoverage` contains a switch: `TRonly`. In case misaligned reads in a targeted sequencing experiment shall be analyzed, it is advisable to set `TRonly` to `FALSE`. Yet, if only the coverage of the targeted regions are of interest, it is advisable to set `TRonly` to `TRUE`.

Value

A GRangesList is returned. Every GRanges object contains the coverage information of one chromosome. The metadata columns contain information on the concrete coverage of each sample at a specific position. Furthermore, the column 'TargetBases' contains information on whether the considered region or position contains target bases (value 1) or not (value 0). A region cannot contain both as two regions would be defined in this case.

If a chromosome is not covered and was not targeted as well, the GRanges object solely contains a single line, considering a whole chromosome if `TRonly=FALSE`. If `TRonly=TRUE` the starting and end position of the corresponding chromosome is set to zero.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

`BadRegionFinder`, `determineCoverageQuality`, `determineRegionsOfInterest`, `reportBadRegionsSummary`, `reportBadRegionsDetailed`, `reportBadRegionsGenes`, `plotSummary`, `plotDetailed`, `plotSummaryGenes`, `determineQuantiles`

Examples

```
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)
```

```
coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                    TRonly = FALSE)
```

`determineCoverageQuality` *Classifies the determined coverage*

Description

The previously determined coverage (using `determineCoverage` with `TRonly = TRUE` or `TRonly = FALSE`) for all samples gets combined to be classified into six categories: bad coverage off target, bad coverage on target, acceptable coverage off target, acceptable coverage on target, good coverage off target, good coverage on target. These categories are user-defined.

Usage

```
determineCoverageQuality(threshold1, threshold2, percentage1, percentage2,
                         coverage_summary)
```

Arguments

`threshold1`	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as acceptable.
`threshold2`	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as good. To obtain useful results, `threshold2` has to be greater than `threshold1`.
`percentage1`	Float, defining the percentage of samples that have to feature a coverage of at least `threshold1` so that the position is classified as acceptably covered.
`percentage2`	Float, defining the percentage of samples that have to feature a coverage of at least `threshold2` so that the position is classified as well covered. To obtain useful results, `percentage2` should be greater than zero.
`coverage_summary`	GRangesList object, return value of function `determineCoverage`.

Details

Every chromosome is analyzed individually. First, the coverage of each sample is categorized according to `threshold1` and `threshold2` into three different categories:

bad coverage: less than `threshold1` reads

acceptable coverage: at least `threshold1`, but less than `threshold2` reads

good coverage: at least `threshold2` reads

Subsequently this information gets combined with the defined precentages to obtain a numerically coded quality value for each region saved in the previously created list object `coverage_summary`:

0: off target; not even `percentage1` percent of all samples have a good or acceptable coverage (bad region)

1: on target; not even `percentage1` percent of all samples have a good or acceptable coverage (bad region)

2: off target; at least `percentage1` percent of all samples have a good or acceptable coverage, but less than `percentage2` percent of all samples have a good coverage (acceptable region)

3: on target; at least `percentage1` percent of all samples have a good or acceptable coverage, but less than `percentage2` percent of all samples have a good coverage (acceptable region)

4: off target; at least `percentage2` percent of all samples have a good coverage (good region)

5: on target; at least `percentage2` percent of all samples have a good coverage (good region)

Value

A list is returned. Every component contains the coverage information of one chromosome as a GRanges object. The metadata columns contain information on the concrete coverage of each sample at a specific position. Furthermore, the column 'TargetBases' contains information on whether the considered region or position contains target bases (value 1) or not (value 0). The column 'indicator' contains information on the coverage quality of the corresponding region/position.

If a chromosome is not covered and was not targeted as well, the GRanges object solely contains a single line, considering a whole chromosome if `TRonly=FALSE`. If `TRonly=TRUE` the corresponding component is "NA".

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

`BadRegionFinder`, `determineCoverage`, `determineRegionsOfInterest`, `reportBadRegionsSummary`, `reportBadRegionsDetailed`, `reportBadRegionsGenes`, `plotSummary`, `plotDetailed`, `plotSummaryGenes`, `determineQuantiles`

Examples

```
threshold1 <- 20
threshold2 <- 100
percentage1 <- 0.80
percentage2 <- 0.90
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)
```

```
coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                                      TRonly = FALSE)
coverage_indicators <- determineCoverageQuality(threshold1, threshold2,
                                                percentage1, percentage2,
                                                coverage_summary)
```

`determineQuantiles` *Determines basewise user-defined quantiles*

Description

The function `determineQuantiles` provides a possibility to determine user-definable quantiles for every base previously analyzed. Thereby, the quantiles are determined over all samples.

Usage

```
determineQuantiles(coverage_summary, quantiles, output)
```

Arguments

`coverage_summary`	List object, return value of function `determineCoverage`.
`quantiles`	Vector determining the quantiles to be calculated.
`output`	The folder to write the output files into.

Details

The function `determineQuantiles` serves to determine a set of user-defined quantiles at each position over all samples. Every single base is analyzed, except for the case when the bases were not originally targeted and if no coverage is detected by any of the samples. In this case the corresponding region is summed up.

Files get written out in the form: "Quantiles_chr<chromosomename>.txt".

Value

A list is returned. Every component contains the coverage information of one chromosome as a GRanges object. The metadata columns contain information on the coverage according to the previously defined quantiles. Furthermore, the column 'TargetBases' contains information on whether the considered region or position contains target bases (value 1) or not (value 0).

If a chromosome has not been targeted and/or not covered by any sample, but defined in `regionsOfInterest`, the component is "NA".

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

BadRegionFinder, determineCoverage, determineCoverageQuality, determineRegionsOfInterest, reportBadRegionsSummary, reportBadRegionsDetailed, reportBadRegionsGenes, plotSummary, plotDetailed, plotSummaryGenes

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")

sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)

coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                                      TRonly = TRUE)
quantiles <- c(0.25, 0.5, 0.75)
coverage_summary2 <- determineQuantiles(coverage_summary, quantiles, output)
```

---

determineRegionsOfInterest

Determines the regions of interest

Description

The function determineRegionsOfInterest serves to select the coverage information (including the coverage of all samples and - depending on the input object - their assigned quality value) of one or more subsets of regions.

Usage

```
determineRegionsOfInterest(regionsOfInterest, coverage_indicators)
```

Arguments

regionsOfInterest
: Data frame- or GRanges object containing the regions of interest (if data frame: chromosome: first column, start position: second column and end position: third column).

coverage_indicators
: List object, return value of function determineCoverageQuality.

Details

Every chromosome is analyzed individually. For every base defined in `regionsOfInterest` the previously determined coverage information is written out. The function thereby serves to select special subsets of regions, e.g. targeted and untargeted regions when using `determineCoverage` with `TRonly=FALSE` in the first place or particular targeted regions when using `TRonly=TRUE`.

It is not recommended to use `TRonly=TRUE`, but to select regions off target using `determineRegionsOfInterest`. In this case, no coverage will be registered for all bases off target, as this information was not saved during the step of `determineCoverage`.

If `regionsOfInterest` is identical compared to `targetRegions` and the function `determineCoverage` with `TRonly=TRUE` has been used in the first place, the selection of regions that is returned is not changed at all.

Value

A list is returned. Every component contains the coverage information of one chromosomeas as a GRanges object. The metadata columns contain information on the concrete coverage of each sample at a specific position. Furthermore, the column 'TargetBases' contains information on whether the considered region or position contains target bases (value 1) or not (value 0). The column 'indicator' contains information on the coverage quality of the corresponding region/position (0: bad region off target; 1: bad region on target; 2: acceptable region off target; 3: acceptable region on target; 4: good region off target; 5: good region on target).

If a chromosome has not been targeted and/or not covered by any sample, but defined in `regionsOfInterest`, the component is "NA".

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

`BadRegionFinder`, `determineCoverage`, `determineCoverageQuality`, `reportBadRegionsSummary`, `reportBadRegionsDetailed`, `reportBadRegionsGenes`, `plotSummary`, `plotDetailed`, `plotSummaryGenes`, `determineQuantiles`

Examples

```
threshold1 <- 20
threshold2 <- 100
percentage1 <- 0.80
percentage2 <- 0.90
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)
```

```
coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                                      TRonly = FALSE)
coverage_indicators <- determineCoverageQuality(threshold1, threshold2,
                                                percentage1, percentage2,
                                                coverage_summary)
coverage_indicators_2 <- determineRegionsOfInterest(targetRegions,
                                                    coverage_indicators)
```

`plotDetailed` *Plots a more detailed overview of the coverage quality*

Description

The function `plotDetailed` provides a possibility to visualize the output of `reportBadRegionsDetailed`. A line graph is returned, visualizing the median coverage at each base that was chosen to be in the region of interest. For each base, the category of coverage quality is color coded. Furthermore, information on the genes that are located at the positions analyzed is included.

Usage

```
plotDetailed(threshold1, threshold2, percentage1, percentage2,
             coverage_indicators_temp, output)
```

Arguments

`threshold1`	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as acceptable.
`threshold2`	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as good.
`percentage1`	Float, defining the percentage of samples that have to feature a coverage of at least `threshold1` so that the position is classified as acceptably covered.
`percentage2`	Float, defining the percentage of samples that have to feature a coverage of at least `threshold2` so that the position is classified as well covered.
`coverage_indicators_temp`	List object, return value of function `reportBadRegionsDetailed`.
`output`	The folder to write the output file into. If this argument is an empty string, the plot is printed on the screen.

Details

The function `plotDetailed` serves to summarize the previously determined coverage quality in a visual way, including additionally information on the mean coverage over all samples at every position.

On the y axis the median coverage over all samples is coded. Every position is considered individually.

On the x axis the detected genes are printed. Wherever a new region covering a new gene is registered, a dashed line is drawn.

Yet, additionally to the mere median coverage, the corresponding coverage quality at each position is also included in the plot. The different categories of coverage quality are color coded in the following way: red - bad region on target; yellow - acceptable region on target; green - good region on target; black - bad region off target; dark gray - acceptable region off target; light gray - good region off target.

Value

No value is returned.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

BadRegionFinder, determineCoverage, determineCoverageQuality, determineRegionsOfInterest, reportBadRegionsSummary, reportBadRegionsDetailed, reportBadRegionsGenes, plotSummary, plotSummaryGenes, determineQuantiles

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")
threshold1 <- 20
threshold2 <- 100
percentage1 <- 0.80
percentage2 <- 0.90
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)

coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                                      TRonly = FALSE)
coverage_indicators <- determineCoverageQuality(threshold1, threshold2,
                                                percentage1, percentage2,
                                                coverage_summary)
coverage_indicators_temp <- reportBadRegionsDetailed(threshold1, threshold2,
                                                     percentage1, percentage2,
                                                     coverage_indicators, "",
                                                     samples, output)
plotDetailed(threshold1, threshold2, percentage1, percentage2,
```

```
coverage_indicators_temp, output)
```

---

plotSummary *Plots a summary of the coverage quality*

Description

The function plotSummary provides a possibility to visualize the output of reportBadRegionsSummary. A line graph is returned, visualizing the number of bases that fall into each category of coverage quality. Furthermore, information on the genes located in these regions is included.

Usage

```
plotSummary(threshold1, threshold2, percentage1, percentage2,
            badCoverageSummary, output)
```

Arguments

threshold1	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as acceptable.
threshold2	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as good.
percentage1	Float, defining the percentage of samples that have to feature a coverage of at least threshold1 so that the position is classified as acceptably covered.
percentage2	Float, defining the percentage of samples that have to feature a coverage of at least threshold2 so that the position is classified as well covered.
badCoverageSummary	GRanges object, return value of function reportBadRegionsSummary.
output	The folder to write the output file into. If this argument is an empty string, the plot is printed on the screen.

Details

The function plotSummary serves to summarize the previously determined coverage quality in a visual way.

On the y axis the coverage quality is coded. The different categories are color coded as well as height coded. As numbers from 0 to 5 were previously assigned to the different categories, thick lines are now drawn at the height of the category. Furthermore, the categories are color coded in the following way: red - bad region on target; yellow - acceptable region on target; green - good region on target; black - bad region off target; dark gray - acceptable region off target; light gray - good region off target.

On the x axis the detected genes are printed. Wherever a new region covering a new gene is registered, a dashed line is drawn.

Value

No value is returned.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

BadRegionFinder, determineCoverage, determineCoverageQuality, determineRegionsOfInterest, reportBadRegionsSummary, reportBadRegionsDetailed, reportBadRegionsGenes, plotDetailed, plotSummaryGenes, determineQuantiles

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")

threshold1 <- 20
threshold2 <- 100
percentage1 <- 0.80
percentage2 <- 0.90
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)

coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                                      TRonly = TRUE)
coverage_indicators <- determineCoverageQuality(threshold1, threshold2,
                                                percentage1, percentage2,
                                                coverage_summary)
badCoverageSummary <- reportBadRegionsSummary(threshold1, threshold2,
                                              percentage1, percentage2,
                                              coverage_indicators, "", output)
plotSummary(threshold1, threshold2, percentage1, percentage2,
            badCoverageSummary, output)
```

---

plotSummaryGenes *Plots a summary of the coverage quality concerning the genes only*

Description

The function `plotSummaryGenes` provides a possibility to visualize the output of `reportBadRegionsGenes`. A barplot is returned, visualizing the percent of each gene that falls into each category of coverage quality. The plot thereby serves to quickly distinguish well from bad covered genes.

Usage

```
plotSummaryGenes(threshold1, threshold2, percentage1, percentage2,
                 badCoverageGenes, output)
```

Arguments

`threshold1`	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as acceptable.
`threshold2`	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as good.
`percentage1`	Float, defining the percentage of samples that have to feature a coverage of at least `threshold1` so that the position is classified as acceptably covered.
`percentage2`	Float, defining the percentage of samples that have to feature a coverage of at least `threshold2` so that the position is classified as well covered.
`badCoverageGenes`	Data frame object, return value of function `reportBadRegionsGenes`.
`output`	The folder to write the output file into. If this argument is an empty string, the plot is printed on the screen.

Details

The function `plotSummaryGenes` serves to summarize the previously determined coverage quality in a visual way concerning the genes only.

For every gene either one or two stacked bars are plotted. If a gene is covered, but it was not originally targeted, a bar is plotted containing the following color code: black - bad region off target; dark gray - acceptable region off target; light gray - good region off target. If a gene was originally targeted, a bar is plotted containing the following color code: red - bad region on target; yellow - acceptable region on target; green - good region on target.

Value

No value is returned.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

`BadRegionFinder`, `determineCoverage`, `determineCoverageQuality`, `determineRegionsOfInterest`, `reportBadRegionsSummary`, `reportBadRegionsDetailed`, `reportBadRegionsGenes`, `plotSummary`, `plotDetailed`, `determineQuantiles`

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")
threshold1 <- 20
threshold2 <- 100
percentage1 <- 0.80
percentage2 <- 0.90
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)

coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                                      TRonly = TRUE)
coverage_indicators <- determineCoverageQuality(threshold1, threshold2,
                                                percentage1, percentage2,
                                                coverage_summary)
badCoverageSummary <- reportBadRegionsSummary(threshold1, threshold2,
                                              percentage1, percentage2,
                                              coverage_indicators, "", output)
badCoverageGenes <- reportBadRegionsGenes(threshold1, threshold2, percentage1,
                                          percentage2, badCoverageSummary,
                                          output)
plotSummaryGenes(threshold1, threshold2, percentage1, percentage2,
                 badCoverageGenes, output)
```

---

reportBadRegionsDetailed

Gives a detailed report on the coverage quality

Description

The function `reportBadRegionsDetailed` creates a detailed report containing all regions of interest (basewise), the coverage of each sample at the corresponding positions, the indicator whether the bases were originally targeted, their coverage quality and the corresponding gene (name and geneID).

Usage

```
reportBadRegionsDetailed(threshold1, threshold2, percentage1, percentage2,
                         coverage_indicators, mart, samples, output)
```

Arguments

threshold1	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as acceptable.
threshold2	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as good.
percentage1	Float, defining the percentage of samples that have to feature a coverage of at least threshold1 so that the position is classified as acceptably covered.
percentage2	Float, defining the percentage of samples that have to feature a coverage of at least threshold2 so that the position is classified as well covered.
coverage_indicators	List object, return value of function determineCoverageQuality or determineRegionsOfInterest.
mart	mart as defined in the manual for package 'biomaRt'. If the human genome (hg19) shall be used, an empty string may be provided and the mart is automatically generated.
samples	Data frame object containing the names of the samples to be analyzed (in one column).
output	The folder to write the output files into. If output is just an empty string, no output file is written out.

Details

To gain more detailed information of the coverage quality, a file for every chromosome to be analyzed may be created by the function reportBadRegionsDetailed. The function may either take information on the whole genome (output from determineCoverage with TRonly=FALSE, processed using determineCoverageQuality) as an input, or information on the target regions (output from determineCoverage with TRonly=TRUE, processed using determineCoverageQuality), or information on a selection of regions of interest (output from determineRegionsOfInterest).

Different from the summed-up variant reportBadRegionsSummary, information on every single base of interest gets reported (except for completely uncovered and untargeted regions, which are summed up). For every base its position, the coverage of each sample, information on whether this base was originally targeted (value 1) or not (value 0), the coverage quality and the most likely gene (name and geneID) that was targeted by the original experiment get reported. Information on the gene names and the geneIDs results from biomaRt. If no gene can be found at a position, "NA" is reported for the corresponding base.

The output files are saved as: "BadCoverageChromosome<chromosomename>;threshold1;percentage1;threshold2;perc
The output file may be visualized using plotDetailed.

Value

A list is returned. Every component contains the coverage information of one chromosome as a GRanges object. The metadata columns contain information on the concrete coverage of each sample at a specific position. Furthermore, the column 'TargetBases' contains information on whether the considered region or position contains target bases (value 1) or not (value 0). The column 'indicator' contains information on the coverage quality of the corresponding region/position (0: bad region off target; 1: bad region on target; 2: acceptable region off target; 3: acceptable region on

target; 4: good region off target; 5: good region on target). Furthermore, the name and the geneID of the gene that is located at the corresponding position is saved.

If a chromosome is not covered and was not targeted as well, the component is "NA".

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

References

More information on the R/Bioconductor package 'biomaRt' may be found at:

http://www.bioconductor.org/packages/release/bioc/html/biomaRt.html

See Also

BadRegionFinder, determineCoverage, determineCoverageQuality, determineRegionsOfInterest, reportBadRegionsSummary, reportBadRegionsGenes, plotSummary, plotDetailed, plotSummaryGenes, determineQuantiles

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")

threshold1 <- 20
threshold2 <- 100
percentage1 <- 0.80
percentage2 <- 0.90
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)

coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                                      TRonly = TRUE)
coverage_indicators <- determineCoverageQuality(threshold1, threshold2,
                                                percentage1, percentage2,
                                                coverage_summary)
coverage_indicators_temp <- reportBadRegionsDetailed(threshold1, threshold2,
                                                     percentage1, percentage2,
                                                     coverage_indicators, "",
                                                     samples, output)
```

reportBadRegionsGenes *Sums up the coverage quality on a gene basis*

Description

The function reportBadRegionsGenes creates a summary report considering the coverage quality on a genewise level. Taking the output of reportBadRegionsSummary as an input, the coverage quality for every previously identified gene is reported.

Usage

```
reportBadRegionsGenes(threshold1, threshold2, percentage1, percentage2,
                      badCoverageSummary, output)
```

Arguments

threshold1	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as acceptable.
threshold2	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as good.
percentage1	Float, defining the percentage of samples that have to feature a coverage of at least threshold1 so that the position is classified as acceptably covered.
percentage2	Float, defining the percentage of samples that have to feature a coverage of at least threshold2 so that the position is classified as well covered.
badCoverageSummary	Data frame object, return value of function reportBadRegionsSummary.
output	The folder to write the output file into. If output is just an empty string, no output file is written out.

Details

To gain an overview of the coverage quality of each targeted/covered gene, a summary file may be created by the function reportBadRegionsGenes. The function takes the output of reportBadRegionsSummary as an input.

All regions covering the same gene are summed up in the following way:

The number of bases falling into each quality category is summed up. Thereby, regions which were orignially targeted may easily be separated from those which were not, as targeted regions always feature an uneven number characterizing their coverage quality. If a region is broader than the detected gene, but the quality category is the same for the whole region, the whole region is assigned to the gene.

If no gene is reported in the input file, the coverage quality is summed up for a gene named "NA".

The output file is saved as: "BadCoverageGenesthreshold1;percentage1;threshold2;percentage2.txt". The output file may be visualized using plotSummaryGenes.

Value

A data frame object is returned. The first column contains the name and the geneID of the gene. The following columns contain the percentage of bases falling into the following categories: bad region off traget, bad region on target, acceptable region off target, acceptable region on target, good region off target, good region on target.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

See Also

BadRegionFinder, determineCoverage, determineCoverageQuality, determineRegionsOfInterest, reportBadRegionsSummary, reportBadRegionsDetailed, plotSummary, plotDetailed, plotSummaryGenes, determineQuantiles

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")

threshold1 <- 20
threshold2 <- 100
percentage1 <- 0.80
percentage2 <- 0.90
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)

coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                                      TRonly = TRUE)
coverage_indicators <- determineCoverageQuality(threshold1, threshold2,
                                                percentage1, percentage2,
                                                coverage_summary)
badCoverageSummary <- reportBadRegionsSummary(threshold1, threshold2,
                                              percentage1, percentage2,
                                              coverage_indicators, "",output)
badCoverageGenes <- reportBadRegionsGenes(threshold1, threshold2, percentage1,
                                          percentage2, badCoverageSummary, output)
```

reportBadRegionsSummary
Sums up the coverage quality

Description

The function reportBadRegionsSummary creates a summary report containing all regions of interest, their coverage quality and the corresponding gene (name and geneID).

Usage

```
reportBadRegionsSummary(threshold1, threshold2, percentage1, percentage2,
                        coverage_indicators, mart, output)
```

Arguments

threshold1	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as acceptable.
threshold2	Integer, threshold defining the number of reads that have to be registered for a sample that its coverage is classified as good.
percentage1	Float, defining the percentage of samples that have to feature a coverage of at least threshold1 so that the position is classified as acceptably covered.
percentage2	Float, defining the percentage of samples that have to feature a coverage of at least threshold2 so that the position is classified as well covered.
coverage_indicators	List object, return value of function determineCoverageQuality or determineRegionsOfInterest.
mart	mart as defined in the manual for package 'biomaRt'. If the human genome (hg19) shall be used, an empty string may be provided and the mart is automatically generated.
output	The folder to write the output file into. If output is just an empty string, no output file is written out.

Details

To gain an overview of the coverage quality, a summary file may be created by the function reportBadRegionsSummary. The function may either take information on the whole genome (output from determineCoverage with TRonly=FALSE, processed using determineCoverageQuality) as an input, or information on the target regions (output from determineCoverage with TRonly=TRUE, processed using determineCoverageQuality), or information on a selection of regions of interest (output from determineRegionsOfInterest).

Wherever subsequent bases feature the same coverage quality, the region gets summed up. Although it is not directly reported whether a region contains on or off target bases, this information can be gained from the coverage quality: all bases off target feature an even number characterizing the coverage quality; all bases on target feature an uneven number characterizing the coverage quality.

For each summed up region the gene that is most likely to be targeted by the original experiment gets reported using biomaRt. If no gene can be found, "NA" is saved for the corresponding region. If not all bases in the summed up region cover a gene, the gene gets reported for the whole region nonetheless.

The output file is saved as: "BadCoverageSummary`threshold1`;`percentage1`;`threshold2`;`percentage2`.txt". The output file may be visualized using `plotSummary`.

Value

A GRanges object is returned. It represents a summary of the those adjacent regions that feature the same base quality. In the metadata columns the coverage quality of the region, the name and the geneID of the gene that is located in the corresponding region is saved.

Author(s)

Sarah Sandmann <sarah.sandmann@uni-muenster.de>

References

More information on the R/Bioconductor package 'biomaRt' may be found at:

http://www.bioconductor.org/packages/release/bioc/html/biomaRt.html

See Also

`BadRegionFinder`, `determineCoverage`, `determineCoverageQuality`, `determineRegionsOfInterest`, `reportBadRegionsDetailed`, `reportBadRegionsGenes`, `plotSummary`, `plotDetailed`, `plotSummaryGenes`, `determineQuantiles`

Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")

threshold1 <- 20
threshold2 <- 100
percentage1 <- 0.80
percentage2 <- 0.90
sample_file <- system.file("extdata", "SampleNames.txt",
                           package = "BadRegionFinder")
samples <- read.table(sample_file)
bam_input <- system.file("extdata", package = "BadRegionFinder")
output <- system.file("extdata", package = "BadRegionFinder")
target_regions <- system.file("extdata", "targetRegions.bed",
                              package = "BadRegionFinder")
targetRegions <- read.table(target_regions, header = FALSE,
                            stringsAsFactors = FALSE)

coverage_summary <- determineCoverage(samples, bam_input, targetRegions, output,
                                      TRonly = TRUE)
coverage_indicators <- determineCoverageQuality(threshold1, threshold2,
                                                percentage1, percentage2,
                                                coverage_summary)
```

```
badCoverageSummary <- reportBadRegionsSummary(threshold1, threshold2, percentage1,
                                              percentage2, coverage_indicators,
                                              "", output)
```

Index

∗ **package**
BadRegionFinder-package, 2

Bad Coverage *(*BadRegionFinder-package*)*, 2
BadRegionFinder, *5*, *7*, *9, 10*, *12*, *14, 15*, *18*, *20*, *22*
BadRegionFinder *(*BadRegionFinder-package*)*, 2
BadRegionFinder-package, 2

Coverage Classifier *(*BadRegionFinder-package*)*, 2

determine Coverage Quality *(*determineCoverageQuality*)*, 6
determine Quantiles *(*determineQuantiles*)*, 8
determine Regions Of Interest *(*determineRegionsOfInterest*)*, 9
determineCoverage, *3*, 4, *7*, *9, 10*, *12*, *14, 15*, *18*, *20*, *22*
determineCoverageQuality, *3*, *5*, 6, *9, 10*, *12*, *14, 15*, *18*, *20*, *22*
determineQuantiles, *3*, *5*, *7*, 8, *10*, *12*, *14, 15*, *18*, *20*, *22*
determineRegionsOfInterest, *3*, *5*, *7*, *9*, 9, *12*, *14, 15*, *18*, *20*, *22*

plot Detailed *(*plotDetailed*)*, 11
plot Summary *(*plotSummary*)*, 13
plot Summary Genes *(*plotSummaryGenes*)*, 14
plotDetailed, *3*, *5*, *7*, *9, 10*, 11, *14, 15*, *18*, *20*, *22*
plotSummary, *3*, *5*, *7*, *9, 10*, *12*, 13, *15*, *18*, *20*, *22*
plotSummaryGenes, *3*, *5*, *7*, *9, 10*, *12*, *14*, 14, *18*, *20*, *22*

report Bad Regions Detailed *(*reportBadRegionsDetailed*)*, 16
report Bad Regions Genes *(*reportBadRegionsGenes*)*, 19
report Bad Regions Summary *(*reportBadRegionsSummary*)*, 21
reportBadRegionsDetailed, *3*, *5*, *7*, *9, 10*, *12*, *14, 15*, 16, *20*, *22*
reportBadRegionsGenes, *3*, *5*, *7*, *9, 10*, *12*, *14, 15*, *18*, 19, *22*
reportBadRegionsSummary, *3*, *5*, *7*, *9, 10*, *12*, *14, 15*, *18*, *20*, 21