Package 'CNVMetrics'

November 11, 2026

Type Package

Version 1.16.0

Date 2021-11-23

Title Copy Number Variant Metrics

Description The CNVMetrics package calculates similarity metrics to facilitate copy number variant comparison among samples and/or methods. Similarity metrics can be employed to compare CNV profiles of genetically unrelated samples as well as those with a common genetic background. Some metrics are based on the shared amplified/deleted regions while other metrics rely on the level of amplification/deletion.
The data type used as input is a plain text file containing the genomic position of the copy number variations, as well as the status and/or the log2 ratio values.
Finally, a visualization tool is provided to explore resulting metrics.

Encoding UTF-8

License Artistic-2.0

Depends R (>= 4.0)

Imports GenomicRanges, IRanges, S4Vectors, BiocParallel, methods, magrittr, stats, pheatmap, gridExtra, grDevices, rBeta2009

Suggests BiocStyle, knitr, rmarkdown, testthat, XVector

biocViews BiologicalQuestion, Software, CopyNumberVariation

VignetteBuilder knitr

URL https://github.com/krasnitzlab/CNVMetrics, https://krasnitzlab.github.io/CNVMetrics/

BugReports https://github.com/krasnitzlab/CNVMetrics/issues

RoxygenNote 7.3.3

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

git_branch RELEASE_3_23

git_last_commit 2dfb3a7

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Astrid Deschênes [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7846-6749>),
Pascal Belleau [aut] (ORCID: <https://orcid.org/0000-0002-0802-1071>),
David A. Tuveson [aut] (ORCID: <https://orcid.org/0000-0002-8017-2712>),
Alexander Krasnitz [aut]

Maintainer Astrid Deschênes <adeschen@hotmail.com>

Contents

CNVMetrics-package 2
calculateJaccard 3
calculateLog2ratioMetric 4
calculateOneLog2valueMetricT 6
calculateOneOverlapMetricT 8
calculateOverlapMetric 9
calculateSorensen 12
calculateSzymkiewicz 13
calculateWeightedEuclideanDistanceFor2Samples 14
createDisjoinSegmentsForTwoSamples 16
is.CNVMetric 17
plotMetric 17
plotOneMetric 19
print.CNVMetric 20
processChr 21
processSim 22
simChr 24
validatecalculateLog2ratioMetricParameters 25
validateCalculateOverlapMetricParameters 26

Index **28**

`CNVMetrics-package` *CNVMetrics: Copy number variant metrics*

Description

The CNVMetrics package calculates similarity metrics to facilitate copy number variant comparison among samples and/or methods. Similarity metrics can be employed to compare CNV profiles of genetically unrelated samples as well as those with a common genetic background. Some metrics are based on the shared amplified/deleted regions while other metrics rely on the level of amplification/deletion. The data type used as input is a plain text file containing the genomic position of the copy number variations, as well as the status and/or the log2 ratio values. Finally, a visualization tool is provided to explore resulting metrics.

Author(s)

Astrid Deschênes, Pascal Belleau, David A. Tuveson and Alexander Krasnitz

Maintainer: Astrid Deschênes <adeschen@hotmail.com>

See Also

- `calculateOverlapMetric` for calculating metric using overlapping amplified/deleted regions
- `calculateLog2ratioMetric` for calculating metric using log2ratio values
- `processSim` for generating simulations
- `plotMetric` for plotting metrics

`calculateJaccard` *Calculate Jaccard metric*

Description

Calculate Jaccard metric using overlapping regions between two samples.

Usage

```
calculateJaccard(sample01, sample02)
```

Arguments

`sample01`	a `GRanges` which contains a collection of genomic ranges representing copy number events for the first sample.
`sample02`	a `GRanges` which contains a collection of genomic ranges representing copy number events for the second sample.

Details

The method calculates the Jaccard metric using overlapping regions between the samples. All regions present in both samples are used for the calculation of the metric.

The Jaccard metric is calculated by dividing the size of the intersection by the size of the union of the two sets. If the the size of the union of the two sets is zero; the value `NA` is returned instead. The strand of the regions is not taken into account while calculating the intersection.

Value

a `numeric`, the value of the Jaccard metric. If the metric cannot be calculated, `NA` is returned.

Author(s)

Astrid Deschênes

References

Jaccard, P. (1912), The Distribution of the Flora in the Alpine Zone. New Phytologist, 11: 37-50. DOI: https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

Examples

```
## Load required package to generate the two samples
require(GenomicRanges)

## Generate two samples with identical sequence levels
sample01 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1905048, 4554832, 31686841),
    end=c(2004603, 4577608, 31695808)), strand="*")
sample02 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1995066, 31611222),
    end=c(2204505, 31689898)), strand="*")

## Calculate Sorensen metric
CNVMetrics:::calculateJaccard(sample01=sample01, sample02=sample02)
```

---

calculateLog2ratioMetric

Calculate metric using overlapping amplified/deleted regions

Description

This function calculates a specific metric, as specified by the user, using overlapping amplified/deleted regions between to samples. The metric is calculated for the amplified and deleted regions separately. When more than 2 samples are present, the metric is calculated for each sample pair.

Usage

```
calculateLog2ratioMetric(
  segmentData,
  method = c("weightedEuclideanDistance"),
  minThreshold = 0.2,
  excludedRegions = NULL,
  nJobs = 1
)
```

Arguments

segmentData	a GRangesList that contains a collection of genomic ranges representing copy number events, including amplified/deleted status, from at least 2 samples. All samples must have a metadata column called 'log2ratio' with the log2ratio values.

`method` a `character` string representing the metric to be used. This should be (an unambiguous abbreviation of) one of "weightedEuclideanDistance". Default: "weightedEuclideanDistance".

`minThreshold` a single positive `numeric` setting the minimum value to consider two segments as different during the metric calculation. If the absolute difference is below or equal to threshold, the difference will be replaced by zero. Default: 0.2.

`excludedRegions` an optional `GRanges` containing the regions that have to be excluded for the metric calculation. Default: `NULL`.

`nJobs` a single positive `integer` specifying the number of worker jobs to create in case of distributed computation. Default: `1` and always `1` for Windows.

Details

The weighted euclidean distance is $(\sum((x_i - y_i)^2 * log(nbrBases_i))^0.5$ where `x` and `y` are the values of 2 samples for a specific segment `i` and `nbrBases` the number of bases of the segment `i`.

Value

an object of class "`CNVMetric`" which contains the calculated metric. This object is a list with the following components:

- `LOG2RATIO` a lower-triangular `matrix` with the results of the selected metric on the log2ratio values for each paired samples. The value `NA` is present when the metric cannot be calculated. The value `NA` is also present in the top-triangular section, as well as the diagonal, of the matrix.

The object has the following attributes (besides "class" equal to "CNVMetric"):

- `metric` the metric used for the calculation.
- `names` the names of the two matrix containing the metrics for the amplified and deleted regions.

Author(s)

Astrid Deschênes, Pascal Belleau

Examples

```
## Load required package to generate the samples
require(GenomicRanges)

## Create a GRangesList object with 3 samples
## The stand of the regions doesn't affect the calculation of the metric
demo <- GRangesList()
demo[["sample01"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1905048, 4554832, 31686841),
    end=c(2004603, 4577608, 31695808)), strand="*",
    log2ratio=c(2.5555, 1.9932, -0.9999))

demo[["sample02"]] <- GRanges(seqnames="chr1",
```

```
    ranges=IRanges(start=c(1995066, 31611222, 31690000),
    end=c(2204505, 31689898, 31895666)), strand=c("-", "+", "+"),
    log2ratio=c(0.3422, 0.5454, -1.4444))

## The amplified region in sample03 is a subset of the amplified regions
## in sample01
demo[["sample03"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1906069, 4558838),
    end=c(1909505, 4570601)), strand="*",
    log2ratio=c(3.2222, -1.3232))

## Calculating Sorensen metric
calculateLog2ratioMetric(demo, method="weightedEuclideanDistance", nJobs=1)
```

`calculateOneLog2valueMetricT` *Calculate metric using the log2ratio values between two samples.*

Description

Calculate a specific metric using the level of amplification/deletion, in log2 ratio, between two samples.

Usage

```
calculateOneLog2valueMetricT(
  entry,
  segmentData,
  method,
  minThreshold,
  bedExclusion
)
```

Arguments

`entry`	a `list` which contains the row and column indexes (always in this order) of the metric in the final matrix. Those values correspond to the positions of the two samples used to calculate the metric in the `GRangesList` (`segmentData`).
`segmentData`	a `GRangesList` that contains a collection of genomic ranges representing copy number events, including amplified/deleted status, from at least 2 samples. All samples must have a metadata column called '`log2ratio`' with the log2ratio values.
`method`	a `character` string representing the metric to be used ('`weightedEuclideanDistance`').
`minThreshold`	a single `numeric` setting the minimum value to consider two segments as different during the metric calculation. If the absolute difference is below or equal to threshold, the difference will be replaced by zero.

bedExclusion an optional GRanges containing the regions that have to be excluded for the metric calculation or NULL.

Details

The method calculates a specified metric using overlapping regions between the samples. Only regions corresponding to the type specified by user are used in the calculation of the metric. The strand of the regions is not taken into account while calculating the metric.

The Sorensen metric is calculated by dividing twice the size of the intersection by the sum of the size of the two sets. If the sum of the size of the two sets is zero; the value NA is returned instead.

Value

a list containing 1 entry:

- metric a data.frame, which contains 3 columns. The 2 first columns, called row and column correspond to the indexes of the metric in the final matrix. Those 2 first columns match to the entry parameter. The third column, called metric, contains the values of the specified metric for each combination. If the metric cannot be calculated, NA is present.

Author(s)

Astrid Deschênes

Examples

```
## Load required package to generate the two samples
require(GenomicRanges)

## Create a GRangesList object with 3 samples
## The stand of the regions doesn't affect the calculation of the metric
demo <- GRangesList()

## Generate two samples with log2value information as a metadata column
demo[["sample01"]]  <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(100, 201, 400),
    end=c(200, 350, 500)), strand="*",
    log2ratio=c(1.1111, 2.2222, -0.9999))
demo[["sample02"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(150, 200, 450),
    end=c(250, 350, 500)), strand="*",
    log2ratio=c(2.2121, 1.1212, -1.3939))

## The 2 samples used to calculate the metric
entries <- data.frame(row=c(2), col=c(1))

## Calculate weighted Euclidean distance
CNVMetrics:::calculateOneLog2valueMetricT(entry=entries,
    segmentData=demo, method="weightedEuclideanDistance",
    minThreshold=0.2, bedExclusion=NULL)
```

`calculateOneOverlapMetricT` *Calculate metric using overlapping amplified/deleted regions between two samples.*

Description

Calculate a specific metric using overlapping amplified/deleted regions between two samples.

Usage

```
calculateOneOverlapMetricT(entry, segmentData, method, type)
```

Arguments

`entry`	a `list` which contains the row and column indexes (always in this order) of the metric in the final matrix. Those values correspond to the positions of the two samples used to calculate the metric in the `GRangesList` (`segmentData`).
`segmentData`	a `GRangesList` that contains a collection of genomic ranges representing copy number events, including amplified/deleted status, from at least 2 samples. All samples must have a metadata column called '`state`' with a state, in an character string format, specified for each region (ex: DELETION, LOH, AMPLIFICATION, NEUTRAL, etc.).
`method`	a `character` string representing the metric to be used ('`sorensen`' or '`szymkiewicz`'.
`type`	a `character` string representing the type of copy number events to be used ('`AMPLIFICATION`' or '`DELETION`').

Value

a `list` containing 1 entry:

- `metric` a `data.frame`, which contains 3 columns. The 2 first columns, called `row` and `column` correspond to the indexes of the metric in the final matrix. Those 2 first columns match to the `entry` parameter. The third column, called `metric`, contains the values of the specified metric for each combination. If the metric cannot be calculated, `NA` is present.

Author(s)

Astrid Deschênes

Examples

```
## Load required package to generate the samples
require(GenomicRanges)

## Create a GRangesList object with 3 samples
## The stand of the regions doesn't affect the calculation of the metric
```

```
demo <- GRangesList()
demo[["sample01"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1905048, 4554832, 31686841, 32686222),
    end=c(2004603, 4577608, 31695808, 32689222)), strand="*",
    state=c("AMPLIFICATION", "AMPLIFICATION", "DELETION", "LOH"))

demo[["sample02"]] <- GRanges(seqnames="chr1",
    ranges= IRanges(start=c(1995066, 31611222, 31690000, 32006222),
    end=c(2204505, 31689898, 31895666, 32789233)),
    strand=c("-", "+", "+", "+"),
    state=c("AMPLIFICATION", "AMPLIFICATION", "DELETION", "LOH"))

## The amplified region in sample03 is a subset of the amplified regions
## in sample01
demo[["sample03"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1906069, 4558838),
    end=c(1909505, 4570601)), strand="*",
    state=c("AMPLIFICATION", "DELETION"))

## The 2 samples used to calculate the metric
entries <- data.frame(row=c(2, 3), col=c(1, 1))

## Calculate Sorensen metric for the amplified regions on samples 2 and 3
CNVMetrics:::calculateOneOverlapMetricT(entry=entries, segmentData=demo,
    method="sorensen", type="AMPLIFICATION")

## Calculate Szymkiewicz-Simpson metric for the amplified regions
## in samples 1 and 2
## Amplified regions of sample02 are a subset of the amplified
## regions in sample01
CNVMetrics:::calculateOneOverlapMetricT(entry=entries, segmentData=demo,
    method="szymkiewicz", type="AMPLIFICATION")

## Calculate Sorensen metric for the deleted regions in samples 1 and 2
CNVMetrics:::calculateOneOverlapMetricT(entry=entries, segmentData=demo,
    method="sorensen", type="DELETION")
```

`calculateOverlapMetric` *Calculate metric using overlapping amplified/deleted regions*

Description

This function calculates a specific metric, as specified by the user, using overlapping regions of specific state between to samples. The metric is calculated for each state separately. When more than 2 samples are present, the metric is calculated for each sample pair. By default, the function is calculating metrics for the AMPLIFICATION and DELETION states. However, the user can specify the list of states to be analyzed.

Usage

```
calculateOverlapMetric(
  segmentData,
  states = c("AMPLIFICATION", "DELETION"),
  method = c("sorensen", "szymkiewicz", "jaccard"),
  nJobs = 1
)
```

Arguments

segmentData	a GRangesList that contains a collection of genomic ranges representing copy number events, including amplified/deleted status, from at least 2 samples. All samples must have a metadata column called 'state' with a state, in an character string format, specified for each region (ex: DELETION, LOH, AMPLIFICATION, NEUTRAL, etc.).
states	a vector of character string with at least one entry. The strings are representing the states that will be analyzed. Default: c('AMPLIFICATION', 'DELETION').
method	a character string representing the metric to be used. This should be (an unambiguous abbreviation of) one of "sorensen", "szymkiewicz" or "jaccard". Default: "sorensen".
nJobs	a single positive integer specifying the number of worker jobs to create in case of distributed computation. Default: 1 and always 1 for Windows.

Details

The two methods each estimate the overlap between paired samples. They use different metrics, all in the range [0, 1] with 0 indicating no overlap. The NA is used when the metric cannot be calculated.

The available metrics are (written for two GRanges):

sorensen:

This metric is calculated by dividing twice the size of the intersection by the sum of the size of the two sets. With this metric, an overlap metric value of 1 is only obtained when the two samples are identical.

szymkiewicz:

This metric is calculated by dividing the size of the intersection by the size of the smallest set. With this metric, if one set is a subset of the other set, the overlap metric value is 1.

jaccard:

This metric is calculated by dividing the size of the intersection by the size of the union of the two sets. With this metric, an overlap metric value of 1 is only obtained when the two samples are identical.

Value

an object of class "CNVMetric" which contains the calculated metric. This object is a list where each entry corresponds to one state specified in the 'states' parameter. Each entry is a matrix:

- `state` a lower-triangular `matrix` with the results of the selected metric on the amplified regions for each paired samples. The value `NA` is present when the metric cannot be calculated. The value `NA` is also present in the top-triangular section, as well as the diagonal, of the matrix.

The object has the following attributes (besides "class" equal to "CNVMetric"):

- `metric` the metric used for the calculation.
- `names` the names of the two matrix containing the metrics for the amplified and deleted regions.

Author(s)

Astrid Deschênes, Pascal Belleau

References

Sørensen, Thorvald. n.d. "A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species and Its Application to Analyses of the Vegetation on Danish Commons." Biologiske Skrifter, no. 5: 1–34.

Vijaymeena, M. K, and Kavitha K. 2016. "A Survey on Similarity Measures in Text Mining." Machine Learning and Applications: An International Journal 3 (1): 19–28. doi: https://doi.org/10.5121/mlaij.2016.3103

Jaccard, P. (1912), The Distribution of the Flora in the Alpine Zone. New Phytologist, 11: 37-50. doi: https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

Examples

```
## Load required package to generate the samples
require(GenomicRanges)

## Create a GRangesList object with 3 samples
## The stand of the regions doesn't affect the calculation of the metric
demo <- GRangesList()
demo[["sample01"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1905048, 4554832, 31686841, 32686222),
    end=c(2004603, 4577608, 31695808, 32689222)), strand="*",
    state=c("AMPLIFICATION", "AMPLIFICATION", "DELETION", "LOH"))

demo[["sample02"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1995066, 31611222, 31690000, 32006222),
    end=c(2204505, 31689898, 31895666, 32789233)),
    strand=c("-", "+", "+", "+"),
    state=c("AMPLIFICATION", "AMPLIFICATION", "DELETION", "LOH"))

## The amplified region in sample03 is a subset of the amplified regions
## in sample01
demo[["sample03"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1906069, 4558838),
    end=c(1909505, 4570601)), strand="*",
    state=c("AMPLIFICATION", "DELETION"))
```

```
## Calculating Sorensen metric for both AMPLIFICATION and DELETION
calculateOverlapMetric(demo, method="sorensen", nJobs=1)

## Calculating Szymkiewicz-Simpson metric on AMPLIFICATION only
calculateOverlapMetric(demo, states="AMPLIFICATION", method="szymkiewicz",
    nJobs=1)

## Calculating Jaccard metric on LOH only
calculateOverlapMetric(demo, states="LOH", method="jaccard", nJobs=1)
```

`calculateSorensen` *Calculate Sorensen metric*

Description

Calculate Sorensen metric using overlapping regions between two samples.

Usage

```
calculateSorensen(sample01, sample02)
```

Arguments

`sample01`	a `GRanges` which contains a collection of genomic ranges representing copy number events for the first sample.
`sample02`	a `GRanges` which contains a collection of genomic ranges representing copy number events for the second sample.

Details

The method calculates the Sorensen metric using overlapping regions between the samples. All regions present in both samples are used for the calculation of the metric.

The Sorensen metric is calculated by dividing twice the size of the intersection by the sum of the size of the two sets. If the sum of the size of the two sets is zero; the value `NA` is returned instead. The strand of the regions is not taken into account while calculating the intersection.

Value

a `numeric`, the value of the Sorensen metric. If the metric cannot be calculated, `NA` is returned.

Author(s)

Astrid Deschênes

References

Sørensen, Thorvald. n.d. "A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species and Its Application to Analyses of the Vegetation on Danish Commons." Biologiske Skrifter, no. 5: 1–34.

Examples

```
## Load required package to generate the two samples
require(GenomicRanges)

## Generate two samples with identical sequence levels
sample01 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1905048, 4554832, 31686841),
    end=c(2004603, 4577608, 31695808)), strand="*")
sample02 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1995066, 31611222),
    end=c(2204505, 31689898)), strand="*")

## Calculate Sorensen metric
CNVMetrics:::calculateSorensen(sample01=sample01, sample02=sample02)
```

`calculateSzymkiewicz` *Calculate Szymkiewicz-Simpson metric*

Description

Calculate Szymkiewicz-Simpson metric using overlapping regions between two samples.

Usage

```
calculateSzymkiewicz(sample01, sample02)
```

Arguments

`sample01`	a `GRanges` which contains a collection of genomic ranges representing copy number events for the first sample.
`sample02`	a `GRanges` which contains a collection of genomic ranges representing copy number events for the second sample.

Details

The method calculates the Szymkiewicz-Simpson metric using overlapping regions between the samples. All regions present in both samples all used for the calculation of the metric.

The Szymkiewicz-Simpson metric is calculated by dividing the size of the intersection by the smaller of the size of the two sets. If one sample has a size of zero, the metric is not calculated; the value `NA` is returned instead. The strand of the regions is not taken into account while calculating the intersection.

Value

a `numeric`, the value of the Szymkiewicz-Simpson metric. If the metric cannot be calculated, `NA` is returned.

Author(s)

Astrid Deschênes

References

Vijaymeena, M. K, and Kavitha K. 2016. "A Survey on Similarity Measures in Text Mining." Machine Learning and Applications: An International Journal 3 (1): 19–28. doi: https://doi.org/10.5121/mlaij.2016.3103

Examples

```
## Load required package to generate the two samples
require(GenomicRanges)

## Generate two samples with identical sequence levels
sample01 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1905048, 4554832, 31686841),
    end=c(2004603, 4577608, 31695808)), strand="*")
sample02 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1995066, 31611222),
    end=c(2204505, 31689898)), strand=c("+", "-"))

## Calculate Szymkiewicz-Simpson metric
CNVMetrics:::calculateSzymkiewicz(sample01=sample01, sample02=sample02)
```

---

calculateWeightedEuclideanDistanceFor2Samples

Calculate Weighted Euclidean distance-based metric between samples.

Description

The weighted Euclidean distance-based metric corresponds to the euclidean distance between 2 samples multiplied by the natural logarithm of the number of bases of the analyzed segment. The final metric is 1 over 1 added to the squared sum of the values obtained for all segments that are not excluded of the analysis.

Usage

```
calculateWeightedEuclideanDistanceFor2Samples(segmentData, minThreshold)
```

Arguments

`segmentData`	a `list` marked as a `preMetricSegments` `class` that contains the disjoint segment information from 2 samples and the log2ratio values of the samples in the metadata columns.

`minThreshold` a single `numeric` setting the minimum value to consider two segments as different for the metric calculation. If the absolute difference is below or equal to threshold, the value will be replaced by zero.

Details

The weighted euclidean distance is $1/(1 + (\sum((x_i - y_i)^2 * log2(nbrBases_i))^0.5)$ where `x` and `y` are the values of 2 samples for a specific segment `i` and `nbrBases` the number of bases of the segment `i`.

Value

a `numeric` representing the weighted euclidean distance between the two samples. If the distance cannot be calculated as the two samples don't share any segments with log2ratio value, the value NA is assigned.

Author(s)

Astrid Deschênes

Examples

```
## Load required package to generate the two samples
require(GenomicRanges)

# Create first Granges representing first sample
sample01 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(100, 201, 400), end=c(200, 350, 500)),
    strand="*", log2ratio=c(0.3091175, 0.4582058, -0.3798390))

# Create second Granges representing second sample
sample02 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(150, 200, 450), end=c(250, 350, 500)),
    strand="*", log2ratio=c(0.222174, 0.3282156, -0.2728292))

# Create disjoint segment using the 2 samples and without any region
# excluded from the analysis (parameter bedExclusion set to null)
disjoinGRange <- CNVMetrics:::createDisjoinSegmentsForTwoSamples(
    segmentDataSample1=sample01, segmentDataSample2=sample02,
    bedExclusion=NULL)

## Calculate the weighted ecucidean distance between the two samples
CNVMetrics:::calculateWeightedEuclideanDistanceFor2Samples(
    segmentData=disjoinGRange, minThreshold=0.2)
```

`createDisjoinSegmentsForTwoSamples` *Generate common segments to enable calculation of metrics on two segmented samples.*

Description

The two segments are gathered together, including excluded regions when specified, and a disjoint operation is done to create a collection of non-overlapping ranges. The ranges overlapping the excluded regions are marked as so to be removed from future analysis. The log2value of each samples are assigned to the new disjointed segments for each sample in the metadata columns.

Usage

```
createDisjoinSegmentsForTwoSamples(
  segmentDataSample1,
  segmentDataSample2,
  bedExclusion = NULL
)
```

Arguments

`segmentDataSample1`	a `GRanges`, the segments from the first sample.
`segmentDataSample2`	a `GRanges`, the segments from the second sample.
`bedExclusion`	a `GRanges`, the regions that must be excluded from the analysis. Default: `NULL`.

Value

a `GRanges` containing the common segment information for the two samples. The log2ration value are present, for the two samples, in the metadata columns. When there is not log2ratio value for one sample, NA is the assigned value. A metadata column also specifies if the segments should be included in the analysis.

Author(s)

Astrid Deschênes

Examples

```
## Load required package to generate the two samples
require(GenomicRanges)

# Create first Granges representing first sample
sample01 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(100, 201, 400), end=c(200, 350, 500)),
    strand="*", log2ratio=c(0.3091175, 0.4582058, -0.3798390))
```

```
# Create second Granges representing second sample
sample02 <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(150, 200, 450), end=c(250, 350, 500)),
    strand="*", log2ratio=c(0.222174, 0.3282156, -0.2728292))

# Create disjoint segment using the 2 samples and without any region
# excluded from the analysis (parameter bedExclusion set to null)
CNVMetrics:::createDisjoinSegmentsForTwoSamples(segmentDataSample1=sample01,
    segmentDataSample2=sample02, bedExclusion=NULL)
```

---

is.CNVMetric *Is an object of class* CNVMetric

Description

Functions to test inheritance relationships between an object and class CNVMetric.

Usage

```
## S3 method for class 'CNVMetric'
is(object, class2)
```

Arguments

object	an object.
class2	a character string giving the name of the class.

Value

a logical.

---

plotMetric *Plot metrics present in a* CNVMetric *object*

Description

This function plots one heatmap (or two heatmaps) of the metrics present in a CNVMetric object. For the overlapping metrics, the user can select to print the heatmap related to amplified or deleted regions or both. The NA values present in the metric matrix are transformed into zero for the creation of the heatmap.

Usage

```
plotMetric(
  metric,
  type = "ALL",
  colorRange = c(c("white", "darkblue")),
  show_colnames = FALSE,
  silent = TRUE,
  ...
)
```

Arguments

metric	a CNVMetric object containing the metrics calculated by calculateOverlapMetric or by calculateLog2ratioMetric.
type	a single character string indicating which graph to generate. This should be a type present in the CNVMetric object or "ALL". This is useful for the overlapping metrics that have multiple types specified by the user. Default: "ALL".
colorRange	a vector of 2 character string representing the 2 colors that will be assigned to the lowest (0) and highest value (1) in the heatmap. Default: c("white", "darkblue").
show_colnames	a boolean specifying if column names are be shown. Default: FALSE.
silent	a boolean specifying if the plot should not be drawn. Default: TRUE.
...	further arguments passed to pheatmap::pheatmap() method. Beware that the filename argument cannot be used when type is "ALL".

Value

a gtable object containing the heatmap(s) of the specified metric(s).

Author(s)

Astrid Deschênes

See Also

The default method pheatmap::pheatmap().

Examples

```
## Load required package to generate the samples
require(GenomicRanges)

## Create a GRangesList object with 3 samples
## The stand of the regions doesn't affect the calculation of the metric
demo <- GRangesList()
demo[["sample01"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1905048, 4554832, 31686841),
    end=c(2004603, 4577608, 31695808)), strand="*",
```

```
    state=c("AMPLIFICATION", "AMPLIFICATION", "DELETION"))

demo[["sample02"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1995066, 31611222, 31690000),
    end=c(2204505, 31689898, 31895666)), strand=c("-", "+", "+"),
    state=c("AMPLIFICATION", "AMPLIFICATION", "DELETION"))

## The amplified region in sample03 is a subset of the amplified regions
## in sample01
demo[["sample03"]] <- GRanges(seqnames="chr1",
    ranges=IRanges(start=c(1906069, 4558838),
    end=c(1909505, 4570601)), strand="*",
    state=c("AMPLIFICATION", "DELETION"))

## Calculating Sorensen metric
metric <- calculateOverlapMetric(demo, method="sorensen")

## Plot both amplification and deletion metrics
plotMetric(metric, type="ALL")

## Extra parameters, used by pheatmap(), can also be passed to the function
## Here, we have the metric values print to the cell while the
## row names and column names are removed
plotMetric(metric, type="DELETION", show_rownames=FALSE,
    show_colnames=FALSE, main="deletion", display_numbers=TRUE,
    number_format="%.2f")
```

`plotOneMetric` *Plot one graph related to one set of metrics.*

Description

Plot one heatmap of one set of metrics present in a a `CNVMetric` object.

Usage

```
plotOneMetric(metric, type, colorRange, show_colnames, silent, ...)
```

Arguments

`metric`	a `CNVMetric` object containing the metrics calculated by `calculateOverlapMetric`.
`type`	a `character` string indicating which graph to generate. This should be (an unambiguous abbreviation of) one of "`AMPLIFICATION`" or "`DELETION`" or "`LOG2RATIO`".
`show_colnames`	a `boolean` specifying if column names are be shown.
`silent`	a `boolean` specifying if the plot should not be drawn.
`...`	further arguments passed to `pheatmap::pheatmap()` method.

Value

a gtable object containing the heatmap for the specified metric.

Author(s)

Astrid Deschênes

See Also

The default method pheatmap::pheatmap().

Examples

```
## Load required package to generate the samples
require(GenomicRanges)

## Create a GRangesList object with 3 samples
## The stand of the regions doesn't affect the calculation of the metric
demo <- GRangesList()
demo[["sample01"]] <- GRanges(seqnames = "chr1",
    ranges =  IRanges(start = c(1905048, 4554832, 31686841),
    end = c(2004603, 4577608, 31695808)), strand =  "*",
    state = c("AMPLIFICATION", "AMPLIFICATION", "DELETION"))

demo[["sample02"]] <- GRanges(seqnames = "chr1",
    ranges =  IRanges(start = c(1995066, 31611222, 31690000),
    end = c(2204505, 31689898, 31895666)), strand =  c("-", "+", "+"),
    state = c("AMPLIFICATION", "AMPLIFICATION", "DELETION"))

## The amplified region in sample03 is a subset of the amplified regions
## in sample01
demo[["sample03"]] <- GRanges(seqnames = "chr1",
    ranges =  IRanges(start = c(1906069, 4558838),
    end = c(1909505, 4570601)), strand =  "*",
    state = c("AMPLIFICATION", "DELETION"))

## Calculating Sorensen metric
metric <- calculateOverlapMetric(demo, method="sorensen")

## Plot amplification metrics using darkorange color
CNVMetrics:::plotOneMetric(metric, type="AMPLIFICATION",
    colorRange=c("white", "darkorange"), show_colnames=FALSE, silent=TRUE)
```

---

print.CNVMetric *Print CNVMetric object*

Description

Print a CNVMetric object and returns it invisibly.

Usage

```
## S3 method for class 'CNVMetric'
print(x, ...)
```

Arguments

x	the output object from calculateOverlapRegionsMetric function to be printed.
...	further arguments passed to or from other methods.

Value

the argument x.

See Also

The default method print.default.

---

processChr *Generates a simulation for one chromosome*

Description

TODO

Usage

```
processChr(curSample, simChr, chrCur)
```

Arguments

curSample	a GRanges that contains a collection of genomic ranges representing copy number events, including amplified/deleted status, from one sample. The sample must have a metadata column called 'state' with a state, in an character string format, specified for each region (ex: DELETION, LOH, AMPLIFICATION, NEUTRAL, etc.) and a metadata column called 'CN' that contains the log2 copy number ratios.
simChr	a data.frame containing the information from one simulated chromosome (shuffled segments). The starting position and the ending position of the segments should be between zero and one. The segment width is representing the proportional size of the segment relative to the global segment size for the chromosome. The data.frame columns names should be: 'ID', 'chr', 'start', 'end', 'log2ratio', 'state'.
chrCur	a character string representing the name of the chromosome.

Value

a data.frame containing the simulation for the specified chromosome.

Author(s)

Astrid Deschênes, Pascal Belleau

Examples

```
## Load required package to generate the samples
require(GenomicRanges)

## Create one 'demo' genome with 2 chromosomes and few segments
## The stand of the regions doesn't affect the calculation of the metric
sample01 <- GRanges(seqnames=c(rep("chr1", 4), rep("chr2", 3)),
    ranges=IRanges(start=c(1905048, 4554832, 31686841, 32686222,
        1, 120331, 725531),
    end=c(2004603, 4577608, 31695808, 32689222, 117121,
        325555, 1225582)),
    strand="*",
    state=c("AMPLIFICATION", "NEUTRAL", "DELETION", "LOH",
        "DELETION", "NEUTRAL", "NEUTRAL"),
    log2ratio=c(0.5849625, 0, -1, -1, -0.87777, 0, 0))

## The simulated chromosome with shuffled segment
## The simulated chromosome can be a different chromosome
simulatedChr <- data.frame(ID=rep("S4", 4),
    chr=rep("chr2", 4), start=c(0, 0.02515227, 0.09360992, 0.25903561),
    end=c(0.02515227, 0.09360992, 0.25903561, 1),
    log2ratio=c(-1.0, -0.9999, 0.0, 0.5843),
    state=c("LOH", "DELETION", "NEUTRAL", "AMPLIFICATION"))

## Generates a simulation for chromosome 1 using a simulated chromosome
## The segments from the simulated chromosome will be positioned on the
## chromosome 1 after resizing for the size of chromosome 1.
## The spaces between the segments in chromosome 1 will be
## preserved.
CNVMetrics:::processChr(curSample=sample01, simChr=simulatedChr,
    chrCur="chr1")
```

`processSim` *Generate simulated samples with copy number profiles derived from a specific sample*

Description

The function uses the input sample to simulate new samples. The simulated samples will possess similar sizes of events, proportional to the original chromosome. To generate realistic simulations, the specified sample must contain segments covering the majority of the genome. Most importantly, the NEUTRAL segments should be present.

Usage

```
processSim(curSample, nbSim)
```

Arguments

curSample	a GRanges that contains a collection of genomic ranges representing copy number events, including amplified/deleted status, from one sample. The sample must have a metadata column called 'state' with a state, in an character string format, specified for each region (ex: DELETION, LOH, AMPLIFICATION, NEUTRAL, etc.) and a metadata column called 'CN' that contains the log2 copy number ratios.
nbSim	a single positive integer which is corresponding to the number of simulations that will be generated.

Value

a data.frame containing the segments for each simulated sample. The data.frame has 6 columns:

- ID a character string, the name of the simulated sample
- chr a character string, the name fo the chromosome
- start a integer, the starting position of the segment
- end a integer, the ending position of the segment
- log2ratio a numerical, the log2 copy number ratio assigned to the segment
- state a character string, the state of the segment (ex: DELETION, AMPLIFICATION, NEUTRAL, etc.)

Author(s)

Astrid Deschênes, Pascal Belleau

Examples

```
## Load required package to generate the sample
require(GenomicRanges)

## Create one 'demo' genome with 2 chromosomes and few segments
## The stand of the regions doesn't affect the calculation of the metric
sample01 <- GRanges(seqnames=c(rep("chr1", 4), rep("chr2", 3)),
    ranges=IRanges(start=c(1905048, 4554832, 31686841, 32686222,
        1, 120331, 725531),
    end=c(2004603, 4577608, 31695808, 32689222, 117121,
        325555, 1225582)),
    strand="*",
    state=c("AMPLIFICATION", "NEUTRAL", "DELETION", "LOH",
        "DELETION", "NEUTRAL", "NEUTRAL"),
    log2ratio=c(0.5849625, 0, -1, -1, -0.87777, 0, 0))

## Generates 10 simulated genomes based on the 'demo' genome
simRes <- processSim(curSample=sample01, nbSim=10)
```

simChr *Generate a simulated chromosome based on a reference sample*

Description

The function generates a list of simulated segments that represent a simulated chromosome based on a reference sample specified by the user. The function only accounts for the positions where a segment is assigned. In addition, the total number of segments is preserved. A Dirichlet distribution is used to assigned new sizes to the segments with respect to the relative initial size of the segment. Then, those new segments are shuffled without replacement. The positions are replaced by values between zero and one that represent the relative position in a chromosome where positions without segment have been removed. To ensure valuable results, the reference sample should have segments covering a good proportion of the chromosome; those should include NEUTRAL segments.

Usage

```
simChr(curSample, chrCur, nbSim)
```

Arguments

curSample	a GRanges that contains a collection of genomic ranges representing copy number events, including amplified/deleted status, from exactly one sample. The sample must have a metadata column called 'state' with a state, in an character string format, specified for each region (ex: DELETION, LOH, AMPLIFICATION, NEUTRAL, etc.) and a metadata column called 'CN' that contains the log2 copy number ratios.
chrCur	a character string representing the name of the chromosome that is used as reference for the simulation.
nbSim	a single positive integer which is corresponding to the number of simulations that will be generated.

Value

a list containing one entry per simulation. Each entry is a data.frame containing shuffled segments with 6 columns:

- ID The name of the simulation.
- chr The name fo the chromosome.
- start The starting position of the segment; the positions are between zero and one. The segment width is representing the proportional size of the segment relative to the global segment size.
- end The ending position of the segment; the positions are between zero and one. The segment width is representing the proportional size of the segment relative to the global segment size.
- log2ratio The log2 copy number ratio assigned to the segment.
- state The state of the region (ex: DELETION, LOH, AMPLIFICATION, NEUTRAL, etc.).

Author(s)

Astrid Deschênes, Pascal Belleau

Examples

```
## Load required package to generate the samples
require(GenomicRanges)

## Create one 'demo' genome with 2 chromosomes
## in a GRanges object
## The stand of the regions doesn't affect the calculation of the metric
sample01 <- GRanges(seqnames=c(rep("chr1", 4), rep("chr2", 3)),
    ranges=IRanges(start=c(1905048, 4554832, 31686841, 32686222,
        1, 120331, 725531),
    end=c(2004603, 4577608, 31695808, 32689222, 117121,
        325555, 1225582)),
    strand="*",
    state=c("AMPLIFICATION", "NEUTRAL", "DELETION", "LOH",
        "DELETION", "NEUTRAL", "NEUTRAL"),
    log2ratio=(c(0.5849625, 0, -1, -1, -0.87777, 0, 0)))

## Generates 10 simulated chromosomes (one chromosome per simulated sample)
## based on chromosome 2 from the input sample.
## The shuffled chromosomes have a start and an end between 0 an 1
CNVMetrics:::simChr(curSample=sample01, chrCur="chr2", nbSim=10)

## Generates 4 simulated chromosomes (one chromosome per simulated sample)
## based on chromosome 1 from the input sample.
## The shuffled chromosomes have a start and an end between 0 an 1
CNVMetrics:::simChr(curSample=sample01, chrCur="chr1", nbSim=4)
```

---

validatecalculateLog2ratioMetricParameters

Parameters validation for the `calculateLog2ratioMetric` *function*

Description

Validation of all parameters needed by the public `calculateLog2ratioMetric` function.

Usage

```
validatecalculateLog2ratioMetricParameters(
  minThreshold,
  excludedRegions,
  nJobs
)
```

Arguments

minThreshold	a single positive numeric setting the minimum value to consider two segments as different during the metric calculation. If the absolute difference is below or equal to threshold, the difference will be replaced by zero.
excludedRegions	an optional GRanges containing the regions that have to be excluded for the metric calculation or NULL.
nJobs	a single positive integer specifying the number of worker jobs to create in case of distributed computation.

Value

0.

Author(s)

Astrid Deschênes

Examples

```
## Return zero as all parameters are valid
CNVMetrics:::validatecalculateLog2ratioMetricParameters(
    minThreshold=0.9, excludedRegions=NULL, nJobs=1)
```

---

validateCalculateOverlapMetricParameters

Parameters validation for the calculateOverlapMetric *function*

Description

Validation of all parameters needed by the public calculateOverlapMetric function.

Usage

```
validateCalculateOverlapMetricParameters(states, nJobs)
```

Arguments

states	a vector of character string with at least one entry. The strings are representing the states that will be analyzed.
nJobs	a single positive integer specifying the number of worker jobs to create in case of distributed computation.

Value

0.

Author(s)

Astrid Deschênes

Examples

```
## Return zero as all parameters are valid
CNVMetrics:::validateCalculateOverlapMetricParameters(
    states="GAIN", nJobs=1)
```

Index

∗ **internal**
  calculateJaccard, 3
  calculateOneLog2valueMetricT, 6
  calculateOneOverlapMetricT, 8
  calculateSorensen, 12
  calculateSzymkiewicz, 13
  calculateWeightedEuclideanDistanceFor2Samples, 14
  createDisjoinSegmentsForTwoSamples, 16
  plotOneMetric, 19
  processChr, 21
  simChr, 24
  validatecalculateLog2ratioMetricParameters, 25
  validateCalculateOverlapMetricParameters, 26
∗ **package**
  CNVMetrics-package, 2

calculateJaccard, 3
calculateLog2ratioMetric, *3*, 4, *25*
calculateOneLog2valueMetricT, 6
calculateOneOverlapMetricT, 8
calculateOverlapMetric, *3*, 9, *26*
calculateSorensen, 12
calculateSzymkiewicz, 13
calculateWeightedEuclideanDistanceFor2Samples, 14
CNVMetrics *(*CNVMetrics-package*)*, 2
CNVMetrics-package, 2
createDisjoinSegmentsForTwoSamples, 16

is.CNVMetric, 17

pheatmap::pheatmap(), *18–20*
plotMetric, *3*, 17
plotOneMetric, 19
print.CNVMetric, 20
print.default, *21*
processChr, 21
processSim, *3*, 22

simChr, 24

validatecalculateLog2ratioMetricParameters, 25
validateCalculateOverlapMetricParameters, 26