Package 'svpluscnv'

March 2, 2020

Title svpluscnv: analysis and visualization of complex structural variation data

Version 0.99.1

Author person(``Gonzalo", ``Lopez",
role = c(``aut", ``cre"),
email = ``gonzolgarcia@gmail.com",
comment = c(ORCID = ``0000-0002-5092-1284"))
person(``Laura", ``Egolf",
role = c(``aut"),
email = ``laura.e.egolf@gmail.com",
comment = c(ORCID = ``0000-0002-7103-4801"))
person(``Federico", ``Giorgi",
role = c(``ctb"),
email = ``federico.giorgi@gmail.com",
comment = c(ORCID = ``0000-0002-7325-9908"))

Maintainer Gonzao Lopez <gonzolgarcia@gmail.com>

Description svpluscnv R package is a ``swiss army knife"" for the integration and interpretation of orthogonal datasets including copy number variant (CNV) segmentation profiles and sequencing-based structural variant calls (SVC). The package implements analysis and visualization tools to evaluate chromosomal instability and ploidy, identify genes harboring recurrent SVs and systematically characterize hot-spot genomic locations harboring complex rearrangements such as chromothripsis and chromoplexia.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

biocViews StructuralVariation, VariantAnnotation

Depends R (>= 2.10)

Imports IRanges, GenomicRanges, tidyr, data.table, circlize, D3GB, shape, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg19.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene, methods, stats, graphics, utils, grDevices, taRifx, S4Vectors, AnnotationDbi,GenomicAlignments,GenomicFeatures,scales

Suggests BiocStyle, knitr, rmarkdown

Collate validate.input.data.r internal_functions.r break.annot.r breakpoint.density.r shattered.regions.r chr.arm.cnv.r

segment.means.r circular.plot.r cnv.freq.plot.r
clean.cnv.artifact.r freq.p.test.r gene.cnv.r gene.track.view.r
get.genesgr.r hot.spot.samples.R pct.genome.changed.r
shattered.map.plot.r shattered.regions.cnv.r sv.model.view.r
svpluscnv.data.r

VignetteBuilder knitr

git_url https://github.com/ccbiolab/svpluscnv

NeedsCompilation no

R topics documented:

`amp.del` *Amplifications and deletions*

Description

Retrieve amplification and deletion events from a 'genecnv.obj' generated by 'gene.cnv' function

Usage

```
amp.del(genecnv.obj, logr.cut = 2)
```

Arguments

`genecnv.obj`	(genecnv) an instance of the class 'genecnv' containing gene level copy number info
`logr.cut`	(numeric) the log-ratio cutoff above which genes are considered amplified (e.g 2 = 8 copies for amplification and 0.5 copies for deep deletions, in diploid regions)

Value

(list) A list of lists including amplified.list, amplified.rank, deepdel.list and deepdel.rank

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)

genecnv.obj <- gene.cnv(cnv)

geneampdel <- amp.del(genecnv.obj, logr.cut = 2)
lapply(geneampdel,head)
```

ave.segmean *Average sample CNV*

Description

Obtain the weighted average segment mean log2 ratios from each sample within a CNV segmentaton data.frame

Usage

```
ave.segmean(cnv)
```

Arguments

cnv	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv

Value

(numeric) a vector containing the weighted average logR from segmented data

Examples

```
## validate input CNV data.frames
cnv <- validate.cnv(segdat_lung_ccle)

ave_seg_mean <- ave.segmean(cnv)
head(ave_seg_mean)
```

break.annot-class *break.annot class*

Description

Class instance to store breakpoint annotations in association with genomic features (e.g. gene loci)

Arguments

input	(data.frame): the breakpoint info containing data.frame, this will be occupied by the CNV segmentation data in the case of cnv.break.annot or SV for sv.break.annot. Unique random string rownames are added to the provided data.frame.
genesgr	(GRanges): a GRanges object with genomic features (e.g. genes) to which breakpoints are mapped
disruptSamples	(list): a list which names correspond to genomic features and values correspond to sample ids harboring breakpoints overlapping with said features
disruptBreaks	(list): a list which names correspond to genomic features and values correspond to the ids of breakpount mapped onto them. Break ids are linked to the 'input' data.frame rownames

`upstreamSamples`
: (list): a list which names correspond to genomic features and values correspond to sample ids harboring breakpoints overlapping with upstream region of said features

`upstreamBreaks`
: (list): a list which names correspond to genomic features and values correspond to the ids of breakpount mapped onto upstream regions Break ids are linked to the 'input' data.frame rownames

`dnstreamSamples`
: (list): a list which names correspond to genomic features and values correspond to sample ids harboring breakpoints overlapping with downstream region of said features

`dnstreamBreaks`
: (list): a list which names correspond to genomic features and values correspond to the ids of breakpount mapped onto downstream regions Break ids are linked to the "input' brk object

`param`
: (list): a list of parametres provided for the annotation function

Value

an instance of the class 'break.annot' containing breakpoint mapping onto genes

---

`break.density` *Breakpoint density map*

Description

Generating a genomic map based on a defined bin size and sliding window and counts the number of breakpoints mapped onto each bin. This function is used internally by svpluscnv::shattered.regions and svpluscnv::shattered.regions.cnv

Usage

```
break.density(
  brk,
  chr.lim = NULL,
  genome.v = "hg19",
  window.size = 10,
  slide.size = 2,
  verbose = TRUE
)
```

Arguments

`brk`
: (breaks) An instance of the class 'breaks' obtained from CNV segmentation data (svpluscnv::cnv.breaks) or Structural Variant calls (svpluscnv::svc.breaks).

`chr.lim`
: (data.frame) 3 column table (chrom, begin, end) indicating the chromosome most distal coordinates with coverage. Also returned by the function svpluscnv::chromosome.limit.co

`genome.v`
: (hg19 or hg38) reference genome version to draw chromosome limits and centromeres

window.size	(numeric) size in megabases of the genmome bin onto which breakpoints will be mapped
slide.size	(numeric) size in megabases of the sliding genomic window; if slide.size < window.size the genomic bins will overlap
verbose	(logical) whether to return internal messages

Value

a matrix of samples (rows) and genomic bins (cols) qith the number of breakpoints mapped in heach cell

Examples

```
# initialize CNV data
cnv <- validate.cnv(segdat_lung_ccle)

# obtain CNV breakpoints
brk <- cnv.breaks(cnv)

break.density(brk)
```

---

breaks-class *Data class breaks*

Description

Class to store breakpoint annotations in association with genomic features (e.g. gene loci)

Arguments

breaks	(data.table): the breakpoint info containing data.table, this will be occupied by the CNV segmentation data in the case of cnv.break.annot or SV for sv.break.annot. Unique random string rownames are added to the returned breaks data.frame.
burden	(numeric): a vector containing the total number of breakpoints in each sample
param	(list): a list of parametres provided

Value

an instance of the class 'breaks' containing breakpoint and breakpoint burden information

chr.arm.cnv *Chromosome arm mean CNV*

Description

Obtains a matrix with the weighted average CN per chromosome arm

Usage

```
chr.arm.cnv(cnv, genome.v = "hg19", verbose = FALSE)
```

Arguments

cnv	(S4) an object of class svcnvio containing data type 'cnv' validated by validate.cnv
genome.v	(character) (hg19 or hg38) reference genome version to draw chromosome limits and centromeres
verbose	(logical) whether to return internal messages

Value

a matrix of chromosome arms (rows) versus samples (cols) with average segment logRs per cell

Examples

```
# initialize CNV data
cnv <- validate.cnv(segdat_lung_ccle)

arm_mat <- chr.arm.cnv(cnv, genome.v="hg19")
dim(arm_mat)
```

chr.sort *Chromosome ordering*

Description

A function to order a list of chromosomes

Usage

```
chr.sort(chrlist)
```

Arguments

chrlist	(character): a vector containing chromosome names (chr1, chr2...chrX,chrY)

Value

a character vector of sorted chromosomes

Examples

```
chrlist <- paste("chr",c("X","Y",sample(1:22)),sep="")
chr_sorted <- chr.sort(chrlist)
```

`chromo.regs-class` *Data class chromo.regs*

Description

Class to store shattered regions and information produced by shattered.regions and shattered.regions.cnv functions

Arguments

`regions.summary`	(list): a list of data.frames sumarizing the information of shattered regions found in each sample
`high.density.regions`	(matrix): a numeric matrix representing high breakpoint density genomic bins in each sample (values 1 = high density break; 0 = normal)
`high.density.regions.hc`	(matrix): a numeric matrix representing high breakpoint density genomic bins in each sample (values 1 = high density break; 0 = normal). Only those bins that overlap with high confidence regions defined in regions.summary are set to = 1
`cnv.brk.dens`	(matrix): a numeric matrix representing the number of CNV segmentation breakpoints found in at genomic bins in each sample
`svc.brk.dens`	(matrix): a numeric matrix representing the number of SV breakpoints found at genomic bins in each sample
`cnv.brk.common.dens`	(matrix): a numeric matrix representing the number of CNV breakpoints colocalizing SV breakpoints found at genomic bins in each sample
`svc.brk.common.dens`	(matrix): a numeric matrix representing the number of SV breakpoints colocalizing CNV breakpoints found at genomic bins in each sample
`cnvbrk`	(S4): on object generated by cnv.breaks function
`svcbrk`	(S4): on object generated by svc.breaks function
`common.brk`	(list): on object generated by match.breaks function
`cnv`	(S4) an object of class svcnvio containing data type 'cnv' validated by validate.cnv
`svc`	(S4) an object of class svcnvio containing data type 'svc' validated by validate.svc
`param`	(list): list of configuration parameters provided or set as default

Value

an instance of the class 'chromo.regs' containing breakpoint mapping onto genes

chromosome.limit.coords *Chromosome limit map*

Description

Obtain chromosome start and end positions based on mapped regions from CNV segmentation data

Usage

```
chromosome.limit.coords(cnv)
```

Arguments

cnv	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv

Value

data.table indicating start and end mapped positions of each chromosome

Examples

```
## validate input data.frame
cnv <- validate.cnv(segdat_lung_ccle)

chr.lim <- chromosome.limit.coords(cnv)
```

circ.chromo.plot *Circular visualization of shattered regions*

Description

Produces a circos plot combining CNV and SVC date sooming into the chromosomes harboring shattered regions

Usage

```
circ.chromo.plot(
  chromo.regs.obj,
  sample.id,
  genome.v = "hg19",
  lrr.pct = 0.2,
  lrr.max = 4,
  chrlist = NULL,
  ...
)
```

Arguments

chromo.regs.obj	(chromo.regs) An object of class chromo.regs
sample.id	(character) the id of a sample to be plotted within
genome.v	(character) (hg19 or h38) reference genome version to draw chromosome limits and centromeres
lrr.pct	(numeric) copy number change between 2 consecutive segments: i.e (default) cutoff = 0.2 represents 20 percent fold change
lrr.max	(numeric) CNV plot limit
chrlist	(character) vector containing chromosomes to plot; by default only chromosomes with shattered regions are ploted
...	Additional graphical parameters

Value

circos plot into open device

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)
svc <- validate.svc(svdat_lung_ccle)

## obtain shattered regions
shatt.regions <- shattered.regions(cnv,svc)

# select a random sample from the
id <- "SCLC21H_LUNG"

circ.chromo.plot(shatt.regions, sample.id = id)
```

---

circ.wg.plot *Circular visualization CNV and SVC*

Description

Produces a circos plot combining CNV and SVC of the whole genome

Usage

```
circ.wg.plot(
  cnv,
  svc,
  sample.id = NULL,
  genome.v = "hg19",
  lrr.pct = 0.2,
  lrr.max = 4,
  chrlist = NULL
)
```

Arguments

cnv	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv
svc	(S4) an object of class svcnvio containing data type 'svc' initialized by validate.svc
sample.id	(character) the id of the sample to be plotted
genome.v	(character) (hg19 or h38) reference genome version to draw chromosome limits and centromeres
lrr.pct	(numeric) copy number change between 2 consecutive segments: i.e (default) cutoff = 0.2 represents a fold change of 0.8 or 1.2
lrr.max	(numeric) maximum CNV to be plotted
chrlist	(character) vector containing chromosomes to plot; by default all chromosomes plotted

Value

circos plot into open device

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)
svc <- validate.svc(svdat_lung_ccle)

## select a random sample id
id <- "A549_LUNG"

circ.wg.plot(cnv, svc, sample.id=id)
```

---

clean.cnv.artifact *CNV artifact detection and filtering*

Description

Detects identical or near-identical CNV segments across multiple samples susceptible of representing common variants or technical artifacts. Then those segments CNV log-ratio is replaced by the flanking segments average

Usage

```
clean.cnv.artifact(
  cnv,
  n.reps = 4,
  cnv.size = 2e+06,
  pc.overlap = 0.99,
  fill.gaps = TRUE,
  minsize = 5000,
  verbose = TRUE
)
```

Arguments

cnv	(S4) an object of class svcnvio containing data type 'cnv' validated by validate.cnv
n.reps	(numeric) number of samples with identical segment to consider artifact
cnv.size	(numeric) only smaller segments will be modified in the cnv data.frame
pc.overlap	(numeric) minimun percentage overlap for a pair of segments to be consider identical
fill.gaps	(logical) whether to fill gaps from the segmentaed file after filtering artifacts
minsize	(numeric) the minimum gap size required to fill the gap. Only used if 'fill.gaps=TRUE'
verbose	(logical) whether to print internal messages

Value

a data.frame containing CNV data

Examples

```
## validate input data.frame
cnv <- validate.cnv(segdat_lung_ccle)

cnvcl <- clean.cnv.artifact(cnv)
cnvcl
```

---

cnv.break.annot *Identification of recurrently altered genes using CNV data Identify recurrently altered genes by CNV. The function will identify overlaps between genomic features (e.g. genes) and CNV breakpoints. As opposed to 'gene.cnv' function that returns the overal CNV of each gene, this function allows identifying sub-genic events and may help detecting other rearrangements.*

Description

Identification of recurrently altered genes using CNV data Identify recurrently altered genes by CNV. The function will identify overlaps between genomic features (e.g. genes) and CNV breakpoints. As opposed to 'gene.cnv' function that returns the overal CNV of each gene, this function allows identifying sub-genic events and may help detecting other rearrangements.

Usage

```
cnv.break.annot(
  cnv,
  fc.pct = 0.2,
  genome.v = "hg19",
  genesgr = NULL,
  upstr = 150000,
  dnstr = 150000,
  break.width = 10000,
```

```
  min.cnv.size = NULL,
  min.num.probes = NULL,
  low.cov = NULL,
  clean.brk = NULL,
  verbose = TRUE
)
```

Arguments

cnv	(S4) an object of class svcnvio containing data type 'cnv' validated by validate.cnv
fc.pct	(numeric) copy number change between 2 consecutive segments: i.e (default) cutoff = 0.2 represents a fold change of 0.8 or 1.2.
genome.v	(character): either 'hg19' or 'hg38' accepted; reference genome version to retrieve gene annotations including genomic coordinates and strand
genesgr	(S4) a GenomicRanges object containing gene annotations (if not NULL overides genome.v). It is crutial that the genome version 'genesgr' and the input 'sv' are the same. The GRanges object must contain 'strand' and a metadata field 'gene_id' with unique values. Seqnames are expected in the format (chr1, chr2, ...).
upstr	(numeric) size in base pairs to define gene upstream region onto which breakpoint overlaps will be identified. The strand value, start and stop positions defined in genesgr will be used to create a GRanges object of upstream regions.
dnstr	(numeric) size in base pairs to define gene downstream region onto which breakpoint overlaps will be identified. The strand value, start and stop positions defined in genesgr will be used to create a GRanges object of downstream regions.
break.width	(numeric) maximum breakpoint size to be considered
min.cnv.size	(numeric) The minimun segment size (in base pairs) to include in the analysis
min.num.probes	(numeric) The minimun number of probes per segment to include in the analysis
low.cov	(data.frame) a data.frame (chr, start, end) indicating low coverage regions to exclude from the analysis
clean.brk	(numeric) Identical segments removal when present in above a given number. Identical CNV segments across multiple samples may represent artifact of common germline variants, this is particularly relevant when the segmentation data was generated with a non-paired reference. For paired datasets (e.g. tumor vs. normal) better leave as NULL.
verbose	(logical) whether to return internal messages

Value

an instance of the class 'break.annot' containing breakpoint mapping onto genes

Examples

```
# Initialize CNV data
cnv <- validate.cnv(segdat_lung_ccle)

cnv.break.annot(cnv)
```

`cnv.breaks` *Identify CNV breakpoints*

Description

Identify CNV breakpoints filtered by the change in copy number log-ratio between contiguous segments

Usage

```
cnv.breaks(
  cnv,
  fc.pct = 0.2,
  break.width = 10000,
  min.cnv.size = NULL,
  min.num.probes = NULL,
  chrlist = NULL,
  low.cov = NULL,
  clean.brk = NULL,
  verbose = TRUE
)
```

Arguments

`cnv`	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv
`fc.pct`	(numeric) copy number change between 2 consecutive segments: i.e (default) cutoff = 0.2 represents a fold change of 0.8 or 1.2
`break.width`	(numeric) the maximum distance between a segment end and the subsequent segment start positions beyond which breakpoints are discarded
`min.cnv.size`	(numeric) The minimun segment size (in base pairs) to include in the analysis
`min.num.probes`	(numeric) The minimun number of probes per segment to include in the analysis
`chrlist`	(character) list of chromosomes to include chr1, chr2, etc...
`low.cov`	(data.frame) a data.frame (chr, start, end) indicating low coverage regions to exclude from the analysis
`clean.brk`	(numeric) identical breakpoints across multiple samples tend to be artifacts; remove breaks > N
`verbose`	(logical) whether to return

Value

an instance of the class 'breaks' containing breakpoint and breakpoint burden information

Examples

```
# initialized CNV data
cnv <- validate.cnv(segdat_lung_ccle)

cnv.breaks(cnv)
```

`cnv.freq` *CNV frequency map*

Description

Creates a map of CNVs using genome binning and plots CNV frequency across the genome. This function optionally returns text, graphical or both outputs.

Usage

```
cnv.freq(
  cnv,
  fc.pct = 0.2,
  genome.v = "hg19",
  ploidy = FALSE,
  g.bin = 1,
  sampleids = NULL,
  cex.axis = 1,
  cex.lab = 1,
  label.line = -1.2,
  plot = TRUE,
  summary = TRUE,
  verbose = TRUE
)
```

Arguments

`cnv`	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv
`fc.pct`	(numeric) percentage CNV gain/loss for a segment to be considered changed (i.e. 0.2 = 20 percent change 0.8 < segmean && segmean > 1.2)
`genome.v`	(character) (hg19 or h38) reference genome version to draw chromosome limits and centromeres
`ploidy`	(logical) whether to apply ploidy correction; the function med.segmean will be used to obtain each sample's ploidy logR then this value substracted to each sample's logR values
`g.bin`	(numeric) size in megabases of the genmome bin to compute break density
`sampleids`	(character) vector containing list of samples to include in plot. if set to NULL, all samples in the input will be used
`cex.axis, cex.lab, label.line`	(numeric) plot parameters

plot	(logical) whether to produce a graphical output
summary	(logical) whether to return an object with a the summary
verbose	(logical) whether to return internal messages

Value

an instance of the class 'cnvfreq' and optionally a plot into open device

Examples

```
## validate input data.frame
cnv <- validate.cnv(nbl_segdat)

cnv.freq(cnv, genome.v = "hg19")
```

---

cnvfreq-class *Data class cnvfreq*

Description

Class to store breakpoint annotations in association with genomic features (e.g. gene loci)

Arguments

freqsum	(data.frame): the frequency of gains and losses in each defined genomic bin
bin.mat	(numeric): a matrix of genomic bins versus samples
param	(list): a list of parametres provided

Value

an instance of the class 'cnvfreq'

---

cnv_blacklist_regions *Low coverage regions*

Description

Low coverage regions

Usage

```
cnv_blacklist_regions
```

Format

An object of class `data.frame` with 60 rows and 3 columns.

createRandomString *Unique random string generator*

Description

Generates n unique random character strings of a given length. Note that the length must be big enought in order to avoid offsetting the number n of strings requested

Usage

```
createRandomString(n = 1, strlen = 10)
```

Arguments

n	the number of unique random strings to return
strlen	random string length

Value

a vector of unique random character strings

Examples

```
# To ensure reproducibility make sure to set the seed
set.seed(123456789)

createRandomString(1, 10)
```

d3gb.chr.lim *Chromosome start and end*

Description

Obtains a chromosome start and end positions from a reference genome version

Usage

```
d3gb.chr.lim(genome.v)
```

Arguments

genome.v	(character) reference genome version to retrieve gene annotations (hg19 or GRCh37 and hg38 or GRCh38)

Value

(data.table) a table containing start and end positions for each chromosome

Examples

```
d3gb.chr.lim(genome.v="hg19")
```

`dngr` *Generate GRanges of downstream regions*

Description

Generate GRanges of downstream regions

Usage

```
dngr(ggr, dnstr = 50000)
```

Arguments

`ggr`	(S4) a GenomicRanges object containing gene annotations. It is crutial that the genome version 'genesgr' and the input 'sv' are the same. The GRanges object must contain 'strand' and a metadata field 'gene_id' with unique values. Seqnames are expected in the format (chr1, chr2, ...).
`dnstr`	(numeric) size in base pairs to define gene downstream region onto which breakpoint overlaps will be identified. The strand value, start and stop positions defined in genesgr will be used to create a GRanges object of downstream regions.

Value

(S4) aa GRanges object of downstream regions

`freq.p.test` *Frequency hot spot detection Obtains significance cutoff for the frequency of binary events encoded in a matrix such as that generated by shattered.regions and shattered.regions.cnv algorithms*

Description

Frequency hot spot detection

Obtains significance cutoff for the frequency of binary events encoded in a matrix such as that generated by shattered.regions and shattered.regions.cnv algorithms

Usage

```
freq.p.test(
  mat,
  method = "fdr",
  p.cut = 0.05,
  iter = 100,
  zerofreq = TRUE,
  plot = TRUE,
  verbose = FALSE
)
```

Arguments

mat	(numeric matrix) a binary matrix where columns will be tested for their sum value compared to a permutated matrix
method	(character) the method to pass to p.adjust function
p.cut	(numeric) the cutoff for multiple hypothesis corrected p.value
iter	(numeric) Number of iterations to produce null distribution (note that null size will be iter*ncol(mat))
zerofreq	(logical) whether to remove bins with observed frequency = 0; It is recommended to set to TRUE when the bins span genomic regions of low coverage
plot	(logical) whether to generate a histogram comparing observed and null frequency distributions
verbose	(logical) whether to return messages

Value

an instance of the class 'freq.cut'

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)

## obtain a matrix of genomic bins vs samples indicating high density of breaks
shatt.regions <- shattered.regions.cnv(cnv)
mat <- shatt.regions@high.density.regions.hc

freq.p.test(mat)
```

---

freq.threshold *Return frequency threshold from null.freq object*

Description

Return frequency threshold from null.freq object

Usage

```
freq.threshold(object)

## S4 method for signature 'null.freq'
freq.threshold(object)
```

Arguments

object	(null.freq) An object of class null.freq

Value

an instance of the class 'chromo.regs' containing breakpoint mapping onto genes

`gene.cnv` *Gene-level CNV*

Description

Obtains a gene-level copy number matrix from a segmentation profile.

Usage

```
gene.cnv(
  cnv,
  genome.v = "hg19",
  genesgr = NULL,
  chrlist = NULL,
  fill.gaps = FALSE,
  verbose = TRUE
)
```

Arguments

`cnv`	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv
`genome.v`	(hg19 or hg38) reference genome version to draw chromosome limits and centromeres
`genesgr`	(S4) a GenomicRanges object containing gene annotations (if not NULL overides genome.v). It must containg 'strand' and a metadata field 'gene_id' with unique values. Seqnames are expected in the format (chr1, chr2, ...)
`chrlist`	(character) list of chromosomes to include chr1, chr2, etc...
`fill.gaps`	(logical) whether to fill the gaps in the segmentation file using gap neighbour segmean average as log ratio
`verbose`	(logical)

Value

an instance of the class 'genecnv' containing gene level copy number info

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)

gene.cnv(cnv)
```

gene.symbol.info *Return coordinates of an specified gene*

Description

Return coordinates of an specified gene

Usage

```
gene.symbol.info(object, symbol)

## S4 method for signature 'refSeqDat'
gene.symbol.info(object, symbol)
```

Arguments

object	(refSeqDat) An object of class refSeqDat containing gene transcript mapping. svpluscnv includes two selfloaded objects: refseq_hg19 & refseq_hg38
symbol	(character) a valid HGNC gene symbol included in the refseq object

gene.track.view *Gene track visualization*

Description

Creates a track visualization of a genomic region defined by gene boundaries or custom provided

Usage

```
gene.track.view(
  chrom = NULL,
  start = NULL,
  stop = NULL,
  symbol = NULL,
  upstr = NULL,
  dnstr = NULL,
  genome.v = "hg19",
  cex.text = 0.6,
  addtext = TRUE,
  plot = TRUE,
  summary = TRUE,
  ...
)
```

Arguments

chrom	(character) Chromosome (e.g. chr9)
start	(numeric) Genomic coordinate from specified chromosome to start plotting
stop	(numeric) Genomic coordinate from specified chromosome to stop plotting
symbol	(character) Gene acceoted hgnc symbol to retrieve coordinates and area plotting ()
upstr	(numeric) Distance upstream specified gene to extend the area plotted
dnstr	(numeric) Distance downstream specified gene to extend the area plotted
genome.v	(character) Reference genome version to draw chromosome limits and centromeres (hg19 or hg38)
cex.text	(numeric) The magnification to be used for transcript RefSeq text added
addtext	(logic) Whether to include transcript RefSeq ids in the plot
plot	(logic) Whether to generate plot in open device
summary	(logic) Whether to produce a data.table output with transcript information
...	Additional graphical parameters

Value

A data.frame with gene isoform annotations and/or plot into open device

Examples

```
# obtain the coordinates of a desired genomic regionbased on a known gene locus
refSeqGene <- gene.symbol.info(refseq_hg19,"PTPRD")
chrom <- refSeqGene$chrom
start <- refSeqGene$start - 150000;
stop <- refSeqGene$stop + 50000;

gene.track.view(symbol="PTPRD", genome.v="hg19")
```

---

genecnv-class *Data class cnvmat*

Description

Class to store breakpoint annotations

Arguments

cnvmat	(data.frame): matrix containing average CNV per gene (rows) for each sample (columns)
genesgr	(S4): a GenomicRanges object with genomic feature annotations such as gene coordinates
cnv	(S4) an object of class svcnvio containing data type 'cnv' validated by validate.cnv
param	(list):

Value

an instance of the class 'genecnv' containing gene level copy number info

---

`get.genesgr` *Genes GRanges*

Description

Retrieves a GRanges object containinng gene annotations for an specified genome version

Usage

```
get.genesgr(genome.v = "hg19", chrlist = NULL)
```

Arguments

`genome.v`	(hg19 or GRCh37 and hg38 or GRCh38) reference genome version to retrieve gene annotations
`chrlist`	(character)

Value

a GRanges class object from the specified human genome version

Examples

```
get.genesgr(genome.v = "hg19",chrlist=NULL)
```

`hbd.mat` *Return the binary matrix containing high confidence high-breakpoint-densityregion definitions*

Description

Return the binary matrix containing high confidence high-breakpoint-densityregion definitions

Usage

```
hbd.mat(object, conf = "hc")

## S4 method for signature 'chromo.regs'
hbd.mat(object, conf = "hc")
```

Arguments

`object`	(chromo.regs) An object of class chromo.regs
`conf`	(character) Either "hc" for high confidence HBD or else include all

Value

an instance of the class 'chromo.regs' containing breakpoint mapping onto genes

---

`hot.spot.samples` *Hot-spot sample retrieval*

Description

Collects sample ids with shattered regions detected at hot-spots based on certain p-value cutoff

Usage

```
hot.spot.samples(chromo.regs.obj, freq.cut)
```

Arguments

`chromo.regs.obj`	(chromo.regs) An object of class chromo.regs
`freq.cut`	(numeric) the hot spot threshold above which peaks are defined for sample ID retrieval

Value

a list comprising two lists: peakRegions, peakRegionsSamples

Examples

```
# validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)
svc <- validate.svc(svdat_lung_ccle)

chromo.regs.obj <- shattered.regions(cnv,svc)
mat<-hbd.mat(chromo.regs.obj)

pcut.obj <- freq.p.test(mat,plot=FALSE)
pcut <- freq.threshold(pcut.obj)

res <- hot.spot.samples(chromo.regs.obj,pcut)
```

IQM *Inter-quantile mean*

Description

Obtains interquantile mean for a defined 'x' vector and both lower and upper quantiles

Usage

```
IQM(x, lowQ = 0.1, upQ = 0.9)
```

Arguments

x	numeric vector to compute interquantile average
lowQ	lower quantile
upQ	upper quantile

Value

(numeric) the IQM value

Examples

```
x <- rnorm(100)
IQM(x)
```

IQSD *Inter-quantile standard deviation*

Description

Obtains inter quantile standard deviation for a defined 'x' vector and both lower and upper quantiles

Usage

```
IQSD(x, lowQ = 0.1, upQ = 0.9)
```

Arguments

x	numeric vector to compute interquantile standard deviation
lowQ	lower quantile
upQ	upper quantile

Value

(numeric) the IQSD value

Examples

```
x <- rnorm(100)
IQSD(x)
```

`map2color` *Color map from numeric vector*

Description

Produces a vector of colors based on a given palette. The colors are defined by the inpuit vector

Usage

```
map2color(x, pal = NULL, limits = NULL)
```

Arguments

`x`	numeric vector
`pal`	color palette
`limits`	numeric limit fr color mapping

Value

a color vector graded according to x

Examples

```
x <- rnorm(100)
x_color <- map2color(x)
head(x_color)
```

`match.breaks` *Breakpoint matching*

Description

Match common breakpoints from two different datasets or data types based on their co-localization in the genome.

Usage

```
match.breaks(brk1, brk2, maxgap = 1e+05, verbose = FALSE, plot = TRUE)
```

Arguments

brk1	(S4) an object of class breaks as returned by ‘svc.breaks‘ and ‘cnv.breaks‘
brk2	(S4) an object of class breaks as returned by ‘svc.breaks‘ and ‘cnv.breaks‘ to compare against brk1
maxgap	(numeric) distance (base pairs) limit for nreakpoints to be consider colocalized
verbose	(logical) whether to return internal messages
plot	(logical) whether to plot into open device

Value

an object containing co-localizing breakpoints from two input ’breaks’

Examples

```
# initialize CNV and SVC data
cnv <- validate.cnv(segdat_lung_ccle)
svc <- validate.svc(svdat_lung_ccle)

## Obtain breakpoints from CNV and SVC
brk1 <- cnv.breaks(cnv)
brk2 <- svc.breaks(svc)

common.brk <- match.breaks(brk1, brk2)
```

---

med.segmean *Median sample CNV*

Description

Obtain the median weighted segment mean from a segmentaton file; The weighted median refers to the logR that occupies a center of all segments ordered by their log ratio

Usage

```
med.segmean(cnv)
```

Arguments

cnv	(S4) an object of class svcnvio containing data type ’cnv’ initialized by validate.cnv

Value

(numeric) a vector containing the median logR value of a segmented data.frame

Examples

```
## validate input CNV data.frames
cnv <- validate.cnv(segdat_lung_ccle)

med_seg_mean <- med.segmean(cnv)
head(med_seg_mean)
```

---

merge2lists *Merge two lists*

Description

Merge of 2 lists into one that contains unique or intersect vectors for each list entry with shared names

Usage

```
merge2lists(x, y, fun = "unique")
```

Arguments

x	(list): input list 1
y	(list): input list 2
fun	(character): Either 'unique' or 'intersect' are accepted

Value

(list) merged list from x and y

Examples

```
x <- sapply(letters[1:10], function(i) sample(1:10)[1:sample(2:10)[1]], simplify=FALSE )
y <- sapply(letters[5:15], function(i) sample(1:10)[1:sample(2:10)[1]], simplify=FALSE )
merge2lists(x,y)
```

---

nbl_segdat *TARGET Neuroblastoma CNV*

Description

TARGET CNV segmentation: https://target-data.nci.nih.gov/

Usage

```
nbl_segdat
```

Format

An object of class `data.frame` with 17680 rows and 6 columns.

nbl_svdat *TARGET Neuroblastoma SVC*

Description

TARGET CGI structural variants: https://target-data.nci.nih.gov/

Usage

```
nbl_svdat
```

Format

An object of class `data.frame` with 7366 rows and 8 columns.

null.freq-class *Data class null.freq*

Description

Class to store observed and null distr. as well as ampirical corrected p-values associated with observed values

Arguments

`freq.cut`	(numeric): the value from observed distribution that satisfies certain p-value cutoff
`pvalues`	(numeric): a vector containing the total number of breakpoints in each sample
`observed`	(numeric): vector of observed distribution
`null`	(numeric): vector of null distribution
`param`	(list): a list of parametres provided

Value

an instance of the class 'freq.cut'

pct.genome.changed *Percent genome change calculation*

Description

Calculates the percentage of genome changed using CNV segmentation profiles. Genome change is defined based on the fold change CNV log-ratio between a sampele and a reference.

Usage

```
pct.genome.changed(cnv, fc.pct = 0.2, discard.sex = TRUE)
```

Arguments

cnv	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv
fc.pct	(numeric) percentage CNV gain/loss for a segment to be considered changed (e.g. 0.2 = 20 percent change 0.8 < segmean && segmean > 1.2)
discard.sex	(logical) whether sex chromosomes should be included

Value

(numeric) vector containing percent genome changed values (0-1)

See Also

Additional data format information in the man pages of validate.cnv

Examples

```
## validate input CNV data.frames
cnv <- validate.cnv(segdat_lung_ccle)

pct_changed <- pct.genome.changed(cnv)
head(pct_changed)
```

refSeqDat-class *Data class refSeqDat*

Description

Class to store refseq data from UCSC containing exon level info for known transcripts

Arguments

data	(data.table): transcript information
exonStarts	(list): every transcript exonic end position
genome.v	(character): the genome version encoding transcript data

Value

an instance of the class 'refSeqDat' containing transcript exonic coordinates

---

refseq_hg19 *Reference transcript and exon annotations for hg19*

Description

refSeq annotations for hg19 version from UCSC (http://genome.ucsc.edu/cgi-bin/hgTables)

Usage

```
refseq_hg19
```

Format

An object of class `refSeqDat` of length 1.

---

refseq_hg38 *Reference transcript and exon annotations for hg38*

Description

refSeq annotations for hg38 version from UCSC (http://genome.ucsc.edu/cgi-bin/hgTables)

Usage

```
refseq_hg38
```

Format

An object of class `refSeqDat` of length 1.

---

segdat_lung_ccle *Lung CCLE CNV data*

Description

CCLE CNV segmentation data from LUNG tissue cell lines (DepMap): https://depmap.org/portal/download/

Usage

```
segdat_lung_ccle
```

Format

An object of class `data.frame` with 162800 rows and 6 columns.

segment.gap *CNV segmentation gap filling*

Description

Fills the gaps in a segmentation data.frame. Chromosome limits are defined for the complete segmentation dataset then segments fill the missing terminal regions. The CN log-ratio of the added segments is set to the average of the closest neighbours in each sample.

Usage

```
segment.gap(cnv, minsize = 5000, chrlist = NULL, verbose = FALSE)
```

Arguments

cnv	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv
minsize	(numeric) the minimum gap size required to fill the gap
chrlist	(character) list of chromosomes to include chr1, chr2, etc...
verbose	(logical) whether to return internal messages

Value

a data.frame containing CNV data

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)

cnv2 <- segment.gap(cnv)
cnv2
```

shattered.eval *Evaluate true catastrophic events Evaluate shattered regions based on interleaved breaks and breakpoint dispersion parameters in order to identify true catastrophic chromosomal alterations*

Description

Evaluate true catastrophic events Evaluate shattered regions based on interleaved breaks and breakpoint dispersion parameters in order to identify true catastrophic chromosomal alterations

Usage

```
shattered.eval(
  chromo.regs.obj,
  interleaved.cut = 0.5,
  dist.iqm.cut = 1e+05,
  verbose = TRUE
)
```

Arguments

chromo.regs.obj	(chromo.regs) An object of class chromo.regs
interleaved.cut	(numeric) the percentage of non interleaved structural variant calls
dist.iqm.cut	(numeric) interquantile average of the distance between breakpoints within a shattered region
verbose	(logical)

Value

an instance of the class 'chromo.regs' containing breakpoint mapping onto genes

---

shattered.map.plot *Shattered regions genomic map*

Description

Plots a genome wide map of shattered region frequencies

Usage

```
shattered.map.plot(
  chromo.regs.obj,
  conf = "hc",
  genome.v = "hg19",
  freq.cut = NULL,
  add.legend = "top"
)
```

Arguments

chromo.regs.obj	(chromo.regs) An object of class chromo.regs
conf	(character) either 'hc' for high confidence objects or else all included
genome.v	(character) reference genome version to draw chromosome limits and centromeres either hg19 or hg38 accepted
freq.cut	the value to draw an horizontal line; use 'freq.p.test' to obtain a threshold for statisticaly significant hot spots
add.legend	the position of the legend in the plot; if null, no legend will be draw

Value

a plot into open device

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)
svc <- validate.svc(svdat_lung_ccle)

## obtain shattered regions
chromo.regs.obj <- shattered.regions(cnv,svc)

shattered.map.plot(chromo.regs.obj)
```

`shattered.regions` *Shattered region detection*

Description

Caller for the identification of shattered genomic regions based on CNV and SVC data

Usage

```
shattered.regions(
  cnv,
  svc,
  fc.pct = 0.2,
  min.cnv.size = 0,
  min.num.probes = 0,
  low.cov = NULL,
  clean.brk = NULL,
  window.size = 10,
  slide.size = 2,
  num.cnv.breaks = 6,
  num.cnv.sd = 5,
  num.svc.breaks = 6,
  num.svc.sd = 5,
  num.common.breaks = 3,
  num.common.sd = 3,
  maxgap = 10000,
  chrlist = NULL,
  interleaved.cut = 0.5,
  dist.iqm.cut = 1e+05,
  verbose = TRUE
)
```

Arguments

`cnv`	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv
`svc`	(S4) an object of class svcnvio containing data type 'svc' initialized by validate.svc

`fc.pct` (numeric) inherited from cnv.breaks(); copy number change between 2 consecutive segments: i.e (default) cutoff = 0.2 represents a fold change of 0.8 or 1.2

`min.cnv.size` (numeric) inherited from cnv.breaks(); The minimun segment size (in base pairs) to include in the analysis

`min.num.probes` (numeric) inherited from cnv.breaks(); The minimun number of probes per segment to include in the analysis

`low.cov` (data.frame) inherited from cnv.breaks(), svc.breaks() and match.breaks; a data.frame (chr, start, end) indicating low coverage regions to exclude from the analysis

`clean.brk` (numeric) inherited from cnv.breaks(); n cutoff for redundant breakpoints to filter out; if NULL, no filter will be applied

`window.size` (numeric) size in megabases of the genmome bin to compute break density

`slide.size` (numeric) size in megabases of the sliding genmome window

`num.cnv.breaks` (numeric) number of segmentation breakpoints per segments to be considered high-density break

`num.cnv.sd` (numeric) number of standard deviations above the sample average for num.cnv.breaks

`num.svc.breaks` (numeric) number of svc breakpoints per segments to be considered high-density break

`num.svc.sd` (numeric) number of standard deviations above the sample average for num.svc.breaks

`num.common.breaks` (numeric) number of common SV and segmentation breakpoints per segments to be considered high-density break

`num.common.sd` (numeric) number of standard deviations above the sample average for num.common.breaks

`maxgap` (numeric) inherited from match.breaks(); sets the maximum gap between co-localizing orthogonal breakpoints

`chrlist` (character) vector containing chromosomes to include in the analysis; if NULL all chromosomes available in the input will be included

`interleaved.cut` (numeric) 0-1 value indicating percentage of interleaved (non-contiguous) SV breakpoint pairs

`dist.iqm.cut` (numeric) interquantile average of the distance between breakpoints within a shattered region

`verbose` (logical)

Value

an instance of the class 'chromo.regs' containing breakpoint mapping onto genes

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)
svc <- validate.svc(svdat_lung_ccle)

shattered.regions(cnv,svc)
```

`shattered.regions.cnv`

CNV-only based shattered region detection

Description

Caller for the identification of shattered genomic regions based on CNV breakpoint densities

Usage

```
shattered.regions.cnv(
  cnv,
  fc.pct = 0.2,
  min.cnv.size = 0,
  min.num.probes = 0,
  low.cov = NULL,
  clean.brk = NULL,
  window.size = 10,
  slide.size = 2,
  num.breaks = 10,
  num.sd = 5,
  dist.iqm.cut = 1e+05,
  verbose = TRUE
)
```

Arguments

`cnv`	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv
`fc.pct`	(numeric) copy number change between 2 consecutive segments: i.e (default) cutoff = 0.2 represents 20 percent fold change
`min.cnv.size`	(numeric) The minimun segment size (in base pairs) to include in the analysis
`min.num.probes`	(numeric) The minimun number of probes per segment to include in the analysis
`low.cov`	(data.frame) a data.frame (chr, start, end) indicating low coverage regions to exclude from the analysis
`clean.brk`	(numeric) inherited from cnv.breaks(); n cutoff for redundant breakpoints to filter out; if NULL, no filter will be applied
`window.size`	(numeric) size in megabases of the genmome bin to compute break density
`slide.size`	(numeric) size in megabases of the sliding genmome window
`num.breaks`	(numeric) size in megabases of the genmome bin to compute break density
`num.sd`	(numeric) size in megabases of the sliding genmome window
`dist.iqm.cut`	(numeric) interquantile average of the distance between breakpoints within a shattered region
`verbose`	(logical)

Value

an instance of the class 'chromo.regs' containing breakpoint mapping onto genes

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)

shattered.regions.cnv(cnv)
```

`sv.model.view` *SV integrated visualization*

Description

Integrated visualization of SVC and CNV data for defined genomic locations. CNV and SVC data is overlayed into a sample-based track visualization map.

Usage

```
sv.model.view(
  cnv,
  svc,
  chrom,
  start,
  stop,
  sampleids = NULL,
  cnvlim = c(-2, 2),
  addlegend = "both",
  cex.legend = 1,
  interval = NULL,
  addtext = NULL,
  cex.text = 0.8,
  plot = TRUE,
  summary = TRUE,
  ...
)
```

Arguments

`cnv`	(S4) an object of class svcnvio containing data type 'cnv' initialized by validate.cnv
`svc`	(S4) an object of class svcnvio containing data type 'svc' initialized by validate.svc
`chrom`	(character) chromosome (e.g chr9)
`start`	(numeric) genomic coordinate from specified chromosome to start plotting
`stop`	(numeric) genomic coordinate from specified chromosome to stop plotting
`sampleids`	(character) a vector containing a list of sample ids represented in svc and/or cnv objects to be plotted
`cnvlim`	(numeric) limits for color coding of background CNV log-ratios. Use to modify the CNV color contrast at different levels.

addlegend	(character) One of 'sv' (show SV type legend), 'cnv' (show CNV background color legend) or 'both'.
cex.legend	(numeric) The cex values for each legend
interval	(numeric) The axis interval in base pairs
addtext	(character) a vector indicating what SV types should include text labels indicating brakpoint partners genomic locations. The added labels are point breakpoint locations outside the plot area. (e.g. c("TRA","INV"))
cex.text	(numeric) The magnification to be used for SV text info added
plot	(logic) whether to produce a graphical output
summary	(logic) whether the function shoud return CNV segment 'segbrk' and SV 'svbrk' breakpoints tabular output
...	additional plot parameters from graphics plot function

Value

a data.frame with CNV and SVN breakpoint annotations and/or plot into open device

Examples

```
## validate input data.frames
cnv <- validate.cnv(segdat_lung_ccle)
svc <- validate.svc(svdat_lung_ccle)

# obtain the coordinates of a desired genomic regionbased on a known gene locus
refSeqGene <- gene.symbol.info(refseq_hg19,"PTPRD")
start <- refSeqGene$start - 150000;
stop <- refSeqGene$stop+ 50000;
chrom <- refSeqGene$chrom

sv.model.view(cnv, svc, chrom, start, stop)
```

---

svc.break.annot *Identification of recurrently altered genes using SVC data*

Description

Identify recurrently altered genes by strutural variants. The function will identify overlaps between genomic features (e.g. genes) and SVs breakpoints.

Usage

```
svc.break.annot(
  svc,
  genome.v = "hg19",
  genesgr = NULL,
  upstr = 50000,
  dnstr = 50000,
  svc.seg.size = 2e+05,
  verbose = TRUE
)
```

Arguments

svc	(S4) an object of class svcnvio containing data type 'svc' validated by validate.svc
genome.v	(character): either 'hg19' or 'hg38' accepted; reference genome version to retrieve gene annotations including genomic coordinates and strand
genesgr	(S4) a GenomicRanges object containing gene annotations (if not NULL overides genome.v). It is crutial that the genome version 'genesgr' and the input 'sv' are the same. The GRanges object must contain 'strand' and a metadata field 'gene_id' with unique values. Seqnames are expected in the format (chr1, chr2, ...).
upstr	(numeric) size in base pairs to define gene upstream region onto which breakpoint overlaps will be identified. The strand value, start and stop positions defined in genesgr will be used to create a GRanges object of upstream regions.
dnstr	(numeric) size in base pairs to define gene downstream region onto which breakpoint overlaps will be identified. The strand value, start and stop positions defined in genesgr will be used to create a GRanges object of downstream regions.
svc.seg.size	(numeric) base pairs for maximum allowed segmental variants (DEL, DUP, INV or INS) size. Larger segmental SVs are treated as translocations and only the breakpoint position will be overlapped with genomic features.
verbose	(logical) whether to return internal messages

Value

an instance of the class 'break.annot' containing breakpoint mapping onto genes

Examples

```
# Initialize SVC data
svc <- validate.svc(svdat_lung_ccle)

svc.break.annot(svc, genome.v="hg19")
```

---

svc.breaks *Identify SVC breakpoints*

Description

Transform structural varian (SVC) data.frame into a 'breaks' object

Usage

```
svc.breaks(svc, low.cov = NULL)
```

Arguments

svc	(S4) an object of class svcnvio containing data type 'svc' initialized by validate.svc
low.cov	(data.table) a data.table (chrom, start, end) indicating low coverage regions to exclude from the analysis

Value

an instance of the class 'breaks' containing breakpoint and breakpoint burden information

Examples

```
## Obtain breakpoints from SV calls data
svc <- validate.svc(svdat_lung_ccle)

svc.breaks(svc)
```

`svcnvio-class` *Data class svcnvio*

Description

Class to store CNV segmentation data

Arguments

`data`	(data.table): cnv or svc data.table to be validated by 'validate.cnv' or 'validate.svc' respectivelly
`type`	(character): the data type "cnv" or "svc" defined by "validate.cnv" or "validate.svc" respectivelly

Value

an instance of the class 'svcnvio' containing SV data derived from CNV or SVC data types; A unique id (uid) column is also added

See Also

Additional data format information in the man pages of validate.cnv and validate.svc

---

`svdat_lung_ccle` *Lung CCLE SVC data*

Description

CCLE translocation data from LUNG tissue cell lines (DepMap): https://depmap.org/portal/download/

Usage

```
svdat_lung_ccle
```

Format

An object of class `data.frame` with 23040 rows and 8 columns.

upgr *Generate GRanges of upstream regions*

Description

Generate GRanges of upstream regions

Usage

```
upgr(ggr, upstr = 50000)
```

Arguments

ggr	(S4) a GenomicRanges object containing gene annotations. It is crutial that the genome version 'genesgr' and the input 'sv' are the same. The GRanges object must contain 'strand' and a metadata field 'gene_id' with unique values. Seqnames are expected in the format (chr1, chr2, ...).
upstr	(numeric) size in base pairs to define gene upstream region onto which breakpoint overlaps will be identified. The strand value, start and stop positions defined in genesgr will be used to create a GRanges object of upstream regions.

Value

(S4) aa GRanges object of upstream regions

validate.cnv *Initialization of CNV data*

Description

This function validates and reformats the CNV segmentation data type containing copy number log-ratios. It is used internaly by 'svpluscnv' functions that require this type of data.

Usage

```
validate.cnv(cnv.df)
```

Arguments

cnv.df	(data.frame) segmentation data with at least 6 columns: sample, chromosome, start, end, probes, segment_mean

Value

an instance of the class 'svcnvio' containing segmentation data derived from CNV data type; A unique id (uid) column is also added

Examples

```
validate.cnv(segdat_lung_ccle)
```

validate.svc *Initialization of SVC data*

Description

This function validates and reformats the SV (structural variant) calls input. It is used internaly by 'svpluscnv' functions that require this type of data. A few formatting rules are enforced: 1) The input must obtain 8 columns in the following order(sample ID, chromosome of origin, strand of origin, position of origin,, chromosome of destination, strand of destination, position of destination, SV class) 2) SV classes accepted: DEL(deletion), DUP(duplication), INS(insertion), TRA(translocation), INV(inversion) and BND(break end) 3) Any variant in which chromosome of origin and destination differ are encoded as TRA (translocation) 4) pos1 < pos2 is enforced for all variants in which chromosome of origin and destination are the same 5) The class BND can be used to operate with complex events as long as both break ends are the same chromosome

Usage

```
validate.svc(sv.df)
```

Arguments

sv.df	(data.frame) structural variant table including the following fields: sample, chrom1, pos1, strand1, chrom2, pos2, strand2, svclass

Value

an instance of the class 'svcnvio' containing SV data derived from SVC data type; A unique id (uid) column is also added

Examples

```
validate.svc(svdat_lung_ccle)
```

Index