Package 'CNVScope'

January 11, 2021

Type Package

Title A Versatile Toolkit for Copy Number Variation Relationship Data Analysis and Visualization

Version 3.5.6

Date 2021-01-08

Author James Dalgeish, Yonghong Wang, Jack Zhu, Paul Meltzer

Maintainer James Dalgleish <james.dalgleish@nih.gov>

BugReports https://github.com/jamesdalg/CNVScope/issues/

Depends R (>= 3.5.0),ggplot2

Imports tidyr,reshape2,magrittr, jointseg,shiny,RCurl,foreach, GenomicInteractions,Matrix,OpenImageR,biomaRt,matrixStats, plyr,data.table,dplyr,numbers,rtracklayer, doParallel,stringr

Suggests knitr, remotes,pwr,ComplexHeatmap,rmarkdown, HiCseg,igraph,visNetwork,circlize,plotly, InteractionSet,GenomicRanges,GenomicFeatures,IRanges,rslurm, shinythemes,shinycssloaders,DT,logging,heatmaply, S4Vectors,BiocManager,shinyjs,htmltools,htmlwidgets, GenomeInfoDb,BSgenome.Hsapiens.UCSC.hg19,tibble,smoothie

VignetteBuilder knitr

URL https://github.com/jamesdalg/CNVScope/

biocViews

Description

Provides the ability to create interaction maps, discover CNV map domains (edges), gene annotate interactions, and create interactive visualizations of these CNV interaction maps.

License BSD_3_clause + file LICENSE

RoxygenNote 7.1.1

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-10 23:20:07 UTC

R topics documented:

`averageMatrixEdges` *Average edges of a matrix to facilitate downsampling.*

Description

Averages the columns and rows of a matrix by a certain amount.

Usage

```
averageMatrixEdges(unchangedmatrix, nedges = 1, dimension = c("row", "column"))
```

Arguments

unchangedmatrix	A matrix to have edges averaged with genomic coordinates in the form chr1_50_100 set as the column and row names.
nedges	The number of edges to be averaged
dimension	Selectively averages edges in one dimension. Performs symmetric edge averaging by default.

Value

averaged_matrix A matrix with edges averaged, which may be more amenable to downsampling

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
dim(nbl_result_matrix_sign_small)
nbl_result_matrix_sign_small_avg<-averageMatrixEdges(nbl_result_matrix_sign_small,
nedges=1,dimension="row")
dim(nbl_result_matrix_sign_small_avg)
nbl_result_matrix_sign_small_avg<-averageMatrixEdges(nbl_result_matrix_sign_small,
nedges=1,dimension="column")
dim(nbl_result_matrix_sign_small_avg)
```

---

calcCNVKernelProbDist *Calculate the probability distribution of CNV concordance events with a fast kernel*

Description

This function produces several matrices, including a Z-score matrix from a matrix of the same size and a percentile matrix of these Z-scores

Arguments

submatrix	A matrix of CNV data in an intrachromosomal region (e.g. chr1 vs chr1 or chr5 vs chr5)
win	a window size for the matrix that calculates the windowed average using the kernel function
debug	extra output for debugging.
parallel	use parallelization using mcmapply and doParallel?
mcmcores	The number of cores used for parallelization.

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
mat_prob_dist<-calcCNVKernelProbDist(nbl_result_matrix_sign_small,parallel=FALSE)
mat_prob_dist
```

`calcVecLMs` *Create a linear regression matrix.*

Description

Creates a matrix of linear regression p-values, log transformed from every combination of columns in the parent matrix.

Usage

```
calcVecLMs(
  bin_data,
  use_slurm = F,
  job_finished = F,
  slurmjob = NULL,
  n_nodes = NULL,
  cpus_on_each_node = 2,
  memory_per_node = "2g",
  walltime = "4:00:00"
)
```

Arguments

`bin_data`	The parent matrix, with columns to have linear regression performed on them.
`use_slurm`	Paralleize over a number of slurm HPC jobs? If false, the program will simply run locally.
`job_finished`	Are all the slurm jobs finished and the results need retrieving?
`slurmjob`	the slurm job object produced by rslurm::slurm_apply(), after running the function initially.
`n_nodes`	the number of nodes used in your slurm job.
`cpus_on_each_node`	The number of cpus used on each node
`memory_per_node`	the amount of ram per node (e.g. "32g" or "2g")
`walltime`	Time for job to be completed for SLURM scheduler in hh:mm:ss format. Defaults to 4h.

Value

The output matrix, or if using slurm, the slurm job object (which should be saved as an rds file and reloaded when creating the output matrix).

Examples

```
#small example
#bin_data<-matrix(runif(5*5),ncol=5)
foreach::registerDoSEQ()
#full_matrix<-suppressWarnings(calcVecLMs(bin_data))
#Please note that lm() will make a warning when there are two vectors that are too close
#numerically (this will always happen along the diagonal).
#This is normal behavior and is controlled & accounted for using this function as well as
#the postProcessLinRegMatrix function (which converts the infinite values to a maximum).
```

`CNVScopeserver` *Server component of the CNVScope plotly shiny application.*

Description

Server function of the CNVScope shiny application. run with runCNVScopeShiny

Arguments

`session`	The shiny session object for the application.
`input`	shiny server input
`output`	shiny server output
`debug`	enable debugging mode

Value

None

Examples

```
## Not run:
runCNVScopeShiny()

## End(Not run)
```

createChromosomalMatrixSet *Create chromosomal interaction matrices for CNVScope shiny application.*

Description

Takes a linear regression matrix and sets infinites to a finite value, and changes the sign to match the sign of the correlation for each value.

Usage

```
createChromosomalMatrixSet(
  whole_genome_mat,
  output_dir = NULL,
  prefix = "nbl_"
)
```

Arguments

whole_genome_mat	The matrix containing all of the data, from which the individual matrices will be split.
output_dir	the folder where the matrices in RData format, will be written.
prefix	filename prefix for individual matrices. Default: "nbl_"

Value

The list of files already written to disk, with full filenames and paths.

Examples

```
#examples for this function would be too large to
#include and should be run on an HPC machine node.
#illustration of this process is shown clearly in
#the vignette and can be done if a user properly
#follows the instructions.
# The function is intended to be run on a whole interactome matrix (chr1-X).
```

`downsample_genomic_matrix` *Rescale positive and negative data, preserving sign information.*

Description

Downsamples a matrix by a specified factor.

Arguments

`whole_matrix`	A matrix to be downsampled, on a single chromosome
`downsamplefactor`	A factor by which to reduce the matrix. Must be something that both the row and columns can be divisible by.
`singlechromosome`	Single chromosome mode; Multi-chromosome not yet implemented (leave T)

Value

whole_matrix_dsamp A downsampled matrix.

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
downsample_genomic_matrix(whole_matrix=nbl_result_matrix_sign_small,
downsamplefactor=5,singlechromosome=TRUE)
```

`extractNegLogPval` *Find the negative log p-value of a pair of vectors.*

Description

Finds the negative log p-value of a matrix, if it exists. Checks first to see if there is a p-value to return.

Usage

```
extractNegLogPval(x, y, repval = 300, lowrepval = 0, signed = F)
```

Arguments

x	a vector that is regressed in the fashion y~x.
y	a vector that is regressed in the fashion y~x.
repval	the replacement value if the regression cannot be performed, default 300 (the vectors are identical if this is used).
lowrepval	The low replacement value in the case that a regression p-value is undefined.
signed	change the sign of the negative log p-value based on the sign of beta? e.g. if the line has a negative slope, so will the returned value. If there is a positive slope, there will be a positive negative log p-value. if this option is disabled, then no sign changes will happen based on the sign of the slope.

Value

The negative log p-value or replacement value.

Examples

```
#small example
xval<-c(1,1,1,1,1)
yval<-c(1,2,3,4,5)
a<-c(3,4,5,6,7)
extractNegLogPval(x=xval,y=yval) #no possible p-value if one vector is constant.
#Some edge cases this may not be correct (if the data lies near a constant),
# but the indiviual sample data should reveal true trends.
suppressWarnings(cor(xval,yval)) #you can't get a correlation value either.
cor(a,a) #gives correlation of 1.
extractNegLogPval(a,a)
#gives replacement value.
suppressWarnings(extractNegLogPval(x=a,y=yval))
#gives 107.3909 and warns about a nearly perfect fit.
```

---

formSampleMatrixFromRawGDCData

Form sample matrix from GDC copy number data files.

Description

Reads a GDC segmetnation files, adds sample information, and forms a data matrix of samples and bins of a specified size.

Arguments

tcga_files	GDC files to be read
format	file format, TCGA or TARGET.
binsize	the binsize, in base pairs (default 1Mb or 1e6). This value provides a good balance of resolution and speed with memory sensitive applications.

freadskip	the number of lines to skip in the GDC files, typically 14 (the first 13 lines are metadata and the first is a blank line in NBL data). Adjust as needed.
debug	debug mode enable (allows specific breakpoints to be checked).
chromosomes	A vector of chromosomes to be used. Defaults to chr1-chrX, but others can be added e.g. chrY or chrM for Y chromosome or mitochondrial DNA. Format expected is a character vector, e.g. c("chr1", "chr2", "chr3").
sample_pat	Pattern used to extract sample name from filename. Use "" to use the filename.
sample_col	The name of the sample column (for custom format input).
chrlabel	The name of the chromosome column (for custom format input).
startlabel	The name of the start column (for custom format input).
endlabel	The name of the end column (for custom format input).

Value

A dataframe containing the aggregated copy number values, based on the parameters provided.

Examples

```
#Pipeline examples would be too large to include in package checks.
#please see browseVignettes("CNVScope") for a demonstration.
```

---

freadGDCfile *Read GDC segmentation datafile for low-pass sequencing data.*

Description

Reads a GDC segmetnation file and extract the segmetnation data.

Usage

```
freadGDCfile(
  file,
  fread_skip = NULL,
  format = "TARGET",
  CN_colname = "log2",
  sample_pattern = "[^_]+",
  sample_colname = NULL
)
```

Arguments

file	GDC file to be read
fread_skip	The number of metadata lines to be skipped(typically 14)
format	The format of the files (TCGA,TARGET, or custom).
CN_colname	The name of the column containing the copy number values.
sample_pattern	Regex pattern to obtain the sample ID from the filename.
sample_colname	Alternatively, a column can be specified with the sample ID on each line.

Value

input_tsv_with_sample_info A data frame containing the sample information extracted from the filename, including sample name & comparison type.

References

https://docs.gdc.cancer.gov/Encyclopedia/pages/TCGA_Barcode/

Examples

```
freadGDCfile(file =
system.file("extdata","somaticCnvSegmentsDiploidBeta_TARGET-30-PANRVJ_NormalVsPrimary.tsv",
package = "CNVScope"))
```

---

getAnnotationMatrix *Get the genes in the genomic ranges indicated by the row and column labels.*

Description

Gets the genes in the ranges within each cell of the matrix.

Usage

```
getAnnotationMatrix(
  genomic_matrix,
  prot_only = T,
  sequential = F,
  flip_row_col = F
)
```

Arguments

genomic_matrix	A matrix with row and column names of the format chr1_100_200 (chr,start,end)
prot_only	Inlcude only the protein coding genes from ensembl?
sequential	Turn off parallelism with doParallel?
flip_row_col	Give column genes along the rows and row genes down columns?

Value

concatenated_gene_matrix A matrix with row and column genes

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
load(system.file("extdata","ensembl_gene_tx_table_prot.rda",package = "CNVScope"))
load(system.file("extdata","grch37.rda",package = "CNVScope"))
getAnnotationMatrix(genomic_matrix=nbl_result_matrix_sign_small[1:5,1:5],sequential=TRUE,
prot_only=TRUE)
```

`getAsymmetricBlockIndices` *Get Block Indices from an asymmetric (or symmetric) matrix.*

Description

This function segments a matrix, including asymmetric matrices using multiple imputation (MI) techniques and a segmentation algorithm to generate breakpoints for column and row.

Usage

```
getAsymmetricBlockIndices(
  genomicmatrix = NULL,
  algorithm = "HiCseg",
  nb_change_max = 100,
  distrib = "G",
  model = "D",
  MI_strategy = "average",
  transpose = T
)
```

Arguments

`genomicmatrix`	the large, whole matrix from which blocks are taken
`algorithm`	Algorithm to be used: HiCseg or jointSeg.
`nb_change_max`	the maximal number of changepoints, passed to HiCseg (if this algorithm is used). Note: HiCseg doesn't actually obey this limit. Rather, use it as a parameter to increase/decrease segmentation extent.
`distrib`	Passed to Hicseg_linkC_R, from their documentation: Distribution of the data: "B" is for Negative Binomial distribution, "P" is for the Poisson distribution and "G" is for the Gaussian distribution."
`model`	Passed on to HiCseg_linkC_R: "Type of model: "D" for block-diagonal and "Dplus" for the extended block-diagonal model."

MI_strategy	strategy to make the matrix temporarily symmetric. "average" adds a number of values equal to the average of the matrix, while copy copies part of the matrix to the shorter side, making a square matrix.
transpose	transpose the matrix and output the breakpoints? Some segmentation algorithms (e.g. HiCseg) produces different results when used against the transposed version of the matrix, as it expects symmetry. This allows the output of additional breakpoints Users can choose to take intersect() or union() on the results to get conserved changepoints or additional changepoints, depending on need.

Value

An output list of the following:

breakpoints_col A vector of breakpoints for the columns.

breakpoints_row A vector of breakpoints for the rows.

breakpoints_col A vector of breakpoints for columns on the transposed genomic matrix.

breakpoints_row A vector of breakpoints for the rows on the transposed genomic matrix.

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
submatrix_tiny<-nbl_result_matrix_sign_small
tiny_test<-getAsymmetricBlockIndices(submatrix_tiny,nb_change_max=10,algorithm="jointSeg")
## Not run:
submatrix_wide<-submatrix_tiny[1:5,]
submatrix_narrow<-submatrix_tiny[,1:5]
wide_test<-getAsymmetricBlockIndices(submatrix_wide,distrib = "G",model = "Dplus",
 nb_change_max = 1e4)
 #the below work, but the time to run all of these would be greater than 10 seconds..
random_wide<-matrix(runif(n = 400*200),ncol=400,nrow=200)
random_narrow<-matrix(runif(n = 400*200),ncol=200,nrow=400)
random_wide_test_avg<-getAsymmetricBlockIndices(random_wide,
 distrib = "G",model = "Dplus",nb_change_max = 1e4)
random_narrow_test_avg<-getAsymmetricBlockIndices(random_narrow,
 distrib = "G",model = "Dplus",nb_change_max = 1e4)
random_wide_test_copy<-getAsymmetricBlockIndices(random_wide,
 distrib = "G",model = "Dplus",nb_change_max = 1e4,MI_strategy = "copy")
random_narrow_test_copy<-getAsymmetricBlockIndices(random_narrow,
 distrib = "G",model = "Dplus",nb_change_max = 1e4,MI_strategy = "copy")
genomicmatrix=random_narrow
nb_change_max=100
model = "D"
distrib = "G"
MI_strategy="copy"
#question-- does it pick different breakpoints if transposed first?
#Answer: yes, at least in Dplus model.
rm(genomicmatrix)
rm(model)
rm(distrib)
rm(MI_strategy)
```

```
random_wide_test_copy<-getAsymmetricBlockIndices(genomicmatrix = random_wide,
                                                 distrib = "G",
                                  model = "Dplus",nb_change_max = 1e2,MI_strategy = "copy")
random_narrow_test_copy<-getAsymmetricBlockIndices(random_narrow,distrib = "G",
                                                   model = "Dplus",
                                                nb_change_max = 1e2,MI_strategy = "copy")
random_wide_test_copy_t<-getAsymmetricBlockIndices(genomicmatrix = t(random_wide),
                                                   distrib = "G",model = "Dplus",
                                                nb_change_max = 1e2,MI_strategy = "copy")
random_narrow_test_copy_t<-getAsymmetricBlockIndices(genomicmatrix = t(random_narrow),
                                                     distrib = "G",model = "Dplus",
                                                nb_change_max = 1e2,MI_strategy = "copy")
length(intersect(random_wide_test_copy$breakpoints_col,
random_wide_test_copy_t$breakpoints_row))/length(unique(c(random_wide_test_copy$breakpoints_col,
random_wide_test_copy_t$breakpoints_row)))
random_wide_test_copy_with_transpose<-getAsymmetricBlockIndices(genomicmatrix = random_wide,
 distrib = "G",model = "Dplus",nb_change_max = 1e2,MI_strategy = "copy",transpose = T)
random_narrow_test_copy_with_transpose<-getAsymmetricBlockIndices(genomicmatrix = random_narrow,
 distrib = "G",model = "Dplus",nb_change_max = 1e2,MI_strategy = "copy",transpose = T)
random_narrow_test_copy_with_transpose<-getAsymmetricBlockIndices(genomicmatrix = random_narrow,
 distrib = "G",model = "Dplus",nb_change_max = 1e2,MI_strategy = "copy",transpose = T)
conserved_breakpoints_col<-intersect(random_narrow_test_copy_with_transpose$breakpoints_col,
 random_narrow_test_copy_with_transpose$t_breakpoints_row)
conserved_breakpoints_row<-intersect(random_narrow_test_copy_with_transpose$breakpoints_row,
 random_narrow_test_copy_with_transpose$t_breakpoints_col)
random_wide_test_copy_with_transpose<-getAsymmetricBlockIndices(genomicmatrix = random_wide,
 distrib = "G",model = "Dplus",nb_change_max = 1e2,MI_strategy = "copy",transpose = T)
conserved_breakpoints_col<-intersect(random_wide_test_copy_with_transpose$breakpoints_col,
 random_wide_test_copy_with_transpose$t_breakpoints_row)
conserved_breakpoints_row<-intersect(random_wide_test_copy_with_transpose$breakpoints_row,
 random_wide_test_copy_with_transpose$t_breakpoints_col)

## End(Not run)
```

`getBlockAverageMatrixFromBreakpoints`
Calculate block averages and areas in a matrix given breakpoints.

Description

This function produces several matrix outputs of averages and areas of matrix blocks, given a pair of vectors for breakpoints.

Arguments

`whole_matrix`	the large, whole matrix from which blocks are taken
`breakpoints_col`	An integer list of column breakpoints, including 1 and the number of columns in the whole matrix.

breakpoints_row
: An integer list of row breakpoints, including 1 and the number of rows in the whole matrix.

outputs
: A list of the following possible outputs (default all): "blockaverages_reformatted_by_index","blockaverag or "blockaverages_matrix_label_area"

Value

An output list of the following:

blockaverages_reformatted_by_index a matrix of the block averages and areas, in long format, with indexes used to generate the averages.

blockaverages_reformatted_by_label a matrix of the block averages and areas, in long format, with labels of the indexes used to generate the averages.

blockaverages_matrix_idx_area a matrix of the block areas, with indexes based on the original row/col index used to generate the data.

blockaverages_matrix_idx_avg a matrix of the block averages, with indexes based on the original row/col index used to generate the data.

blockaverages_matrix_label_area a matrix of the block areas, with indexes based on the original row/col label used to generate the data.

blockaverages_matrix_label_avg a matrix of the block averages, with indexes based on the original row/col label used to generate the data.

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
set.seed(303)
mat<-matrix(data=runif(n = 25),nrow=5,ncol=5,dimnames = list(c("chr1_0_5000",
"chr1_5000_10000","chr1_10000_15000","chr1_15000_20000","chr1_20000_25000"),
c("chr1_0_5000","chr1_5000_10000","chr1_10000_15000","chr1_15000_20000","chr1_20000_25000")))
breakpoints_col<-c(1,2,4,5)
breakpoints_row<-c(1,2,4,5)
foreach::registerDoSEQ()
getBlockAverageMatrixFromBreakpoints(whole_matrix=mat,breakpoints_col=breakpoints_col,
breakpoints_row=breakpoints_row)
## Not run:  #extra examples
mat<-matrix(data=round(runif(min = 0,max=100,n = 25)),nrow=5,ncol=5,
dimnames = list(c("chr1_0_5000","chr1_5000_10000","chr1_10000_15000","chr1_15000_20000",
"chr1_20000_25000"),c("chr2_0_50000","chr2_50000_100000",
"chr2_100000_150000","chr2_150000_200000","chr2_200000_250000")))
breakpoints_col<-c(1,2,4,5)
breakpoints_row<-c(1,2,4,5)
avg_results<-getBlockAverageMatrixFromBreakpoints(whole_matrix=mat,
breakpoints_col=breakpoints_col,breakpoints_row=breakpoints_row)
avg_results$blockaverages_reformatted_by_label
avg_results$blockaverages_reformatted_by_index
whole_matrix=mat
mat<-matrix(data=round(runif(min = 0,max=100,n = 25)),nrow=5,ncol=5,
dimnames = list(c("chr1_0_5000","chr1_5000_10000","chr1_10000_15000",
"chr1_15000_20000","chr1_20000_25000"),c("chr2_0_50000",
```

```
"chr2_50000_100000","chr2_100000_150000",
"chr2_150000_200000","chr2_200000_250000")))
breakpoints_col<-c(1,2,4,5)
breakpoints_row<-c(1,2,4,5)
avg_results<-getBlockAverageMatrixFromBreakpoints(whole_matrix=mat,
breakpoints_col=breakpoints_col,breakpoints_row=breakpoints_row)
avg_results$blockaverages_reformatted_by_label
avg_results$blockaverages_reformatted_by_index
whole_matrix=mat
submatrix<-nbl_result_matrix_sign_small
breakpoints_row_jointseg<-jointseg::jointSeg(submatrix,K=5)$bestBkp
breakpoints_col_jointseg<-jointseg::jointSeg(t(submatrix),K=5)$bestBkp
submatrix_avg_results<-getBlockAverageMatrixFromBreakpoints(whole_matrix=submatrix,
breakpoints_col=breakpoints_col_jointseg,breakpoints_row=breakpoints_row_jointseg)

## End(Not run)
```

`getGlobalRescalingStats` *Calculate several base statistics for color rescaling.*

Description

calculates several statistics from a large matrix that can then be applied to smaller submatrices without needing to load the entire matrix into memmory

Usage

```
getGlobalRescalingStats(whole_matrix, saveToDisk = F, output_fn = NULL)
```

Arguments

`whole_matrix`	the whole matrix to get stats for.
`saveToDisk`	Save the statistics to disk as an RDS file in the local directory?
`output_fn`	the name of the output file.

Value

A list of the output statistics, including: the global min, max, length, sigma (matrix variance), pos_sigma (variance of the positive values), neg_sigma(variance of the negative values), global mean (global_mu), est_max_cap (global_mu+global_sigma_pos*2), as well as the number of rows and columns of the matrix.

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
getGlobalRescalingStats(nbl_result_matrix_sign_small)
```

getInterchromosomalInteractivePlot

Create an HTML widget for use in shiny or webshot for a given pair of chromosomes.

Description

This function requires a matrix with genomic coordinates in the row and column names, and produces a heatmap with a tooltip

Arguments

whole_matrix	the large, whole genomic matrix from which the submatrix is taken (rows)
chrom1	The first chromsome used for the map (columns).
chrom2	The second chromsome used for a map axis.

Value

An HTML widget.

Examples

```
## Not run:
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
getInterchromosomalInteractivePlot(whole_matrix=nbl_result_matrix_sign_small,chrom1=1,
chrom2=1)

## End(Not run)
```

GRanges_to_underscored_pos

Convert GRanges object to underscord positions.

Description

This function converts row or column names (or any character vector of the format) into a GenomicRanges object.

Usage

```
GRanges_to_underscored_pos(input_gr, minusOneToEnd = T)
```

Arguments

input_gr	A GenomicRanges object
minusOneToEnd	Minus one position to end of each Genomic Range?

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
col_gr<-underscored_pos_to_GRanges(colnames(nbl_result_matrix_sign_small))
GRanges_to_underscored_pos(col_gr)
```

---

importBreakpointBed *Import a breakpoint BED file.*

Description

Imports a BED file with breakpoints or other interactions, in a dual position format.

Arguments

breakpoint_fn	the filename of the breakpoint bed file

Value

a Genomic Interactions Object

Examples

```
importBreakpointBed(breakpoint_fn = system.file("extdata",
"sample_breakpoints.bed",package = "CNVScope"))
closeAllConnections()
```

---

mathead *Gets a small piece of a matrix (top left corner) for viewing, rather than pulling the first n rows.*

Description

Gives a small square of a matrix to get an idea of content rather than grabbing the entire row. When this row is thousands of numbers long, this can be a problem.

Usage

```
mathead(mat, n = 6L)
```

Arguments

mat	A matrix.
n	The length and width of the piece to view.

Value

averaged_matrix a small matrix of size n.

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
mathead(nbl_result_matrix_sign_small)
```

`nbl_result_matrix_sign_small`

Neuroblastoma sample CNV relationship matrix

Description

The first 25 Mb of chromosome 1, neuroblastoma copy number signed relation matrix.

Format

A matrix with 25 rows and 25 variables

Source

https://gdc.cancer.gov/

`postProcessLinRegMatrix`

Postprocess linear regression matrix.

Description

Takes a linear regression matrix and sets infinites to a finite value, and changes the sign to match the sign of the correlation for each value.

Usage

```
postProcessLinRegMatrix(
  input_matrix,
  LM_mat,
  cor_type = "pearson",
  inf_replacement_val = 300
)
```

Arguments

input_matrix	The input matrix, which consists of bins and samples (no LM or correlation has been done on the segmentation values)
LM_mat	The linear regression matrix, with rows and columns consisting of bins and the values being the negative log p-value between them.
cor_type	The correlation type ("pearson" (linear), "spearman" (rank), "kendall"(also rank-based)). Rank correlations capture nonlinear relationships as well as linear. Passed to stats::cor's method parameter.
inf_replacement_val	the value for which infinites are replaced, by default 300.

Value

The output matrix, or if using slurm, the slurm job object (which should be saved as an rds file and reloaded when creating the output matrix).

Examples

```
inputmat<-matrix(runif(15),nrow=3)
colnames(inputmat)<-c("chr2_1_1000","chr2_1001_2000","chr2_2001_3000","chr2_3001_4000",
"chr2_4001_5000")
rownames(inputmat)<-c("PAFPJK","PAKKAT","PUFFUM")
outputmat<-matrix(runif(15),nrow=3)
outputmat<-cor(inputmat)*matrix(runif(25,-30,500),nrow=5)
diag(outputmat)<-Inf
postProcessLinRegMatrix(input_matrix=t(inputmat),LM_mat=outputmat,cor_type="pearson",
inf_replacement_val=300)
```

---

rebinGenomicInteractions

Assign GenomicInteractions to a predefined series of bins for row and column, corresponding to a genomic matrix.

Description

This function allows the user to assign a set of genomicinteractions to a pre-existing matrix with known dimensions and column/row names. It finds the row/column index of each point and produces a merged dataframe with the original annotation columns that correspond to each bin in the matrix, with appropriate labels & indexes.

Arguments

gint	A GenomicInteractions object needing to be binned.
whole_genome_matrix	A matrix with underscored positions for column and rownames e.g. chr1_1_5000,chr1_5001_10000. If this is provided, it will override rown/column names and GRanges objects.

rownames_gr	A Genomic Ranges object created from the whole genome matrix row names in chr_start_end format, e.g. chr1_1_5000. No effect if whole_genome_mattrix is specified.
colnames_gr	A Genomic Ranges object created from the whole genome matrix column names in chr_start_end format. No effect if whole_genome_mattrix is specified.
rownames_mat	The row names of the whole_genome_matrix in chr_start_end format.
colnames_mat	The column names of the whole_genome_matrix in chr_start_end format.
method	Method to rebin with– can use overlap and nearest methods.Default: nearest.

Examples

```
foreach::registerDoSEQ()
gint_small_chr1<-importBreakpointBed(breakpoint_fn = system.file("extdata",
"sample_breakpoints_chr1.bed",package = "CNVScope"))
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
rebinGenomicInteractions(gint=gint_small_chr1,whole_genome_matrix=NULL,
rownames_gr=underscored_pos_to_GRanges(rownames(nbl_result_matrix_sign_small)),
colnames_gr=underscored_pos_to_GRanges(colnames(nbl_result_matrix_sign_small)),
rownames_mat = rownames(nbl_result_matrix_sign_small),
colnames_mat = colnames(nbl_result_matrix_sign_small),
method="nearest")
```

---

runCNVScopeLocal *Runs the CNVScope plotly shiny application.*

Description

Runs the interactive suite of tools locally.

Usage

```
runCNVScopeLocal()
```

Value

none. Runs the application if the correct files are present.

Examples

```
## Not run:
CNVScope::runCNVScopeLocal()

## End(Not run)
```

runCNVScopeShiny *Runs the CNVScope plotly shiny application.*

Description

Runs the interactive suite of tools locally or on a server if called in a script file (e.g. App.R). Data sources are required. For a simple installation, please use the runCNVScopeLocal function.

Usage

```
runCNVScopeShiny(
  baseurl = NULL,
  basefn = NULL,
  osteofn = NULL,
  debug = F,
  useCNVScopePublicData = F
)
```

Arguments

baseurl	the url of the source files for the application (e.g. the contents of plotly_dashboard_ext). This will be pulled from remotely.
basefn	the linux file path of the same source files.
osteofn	the linux file path of the OS files.
debug	Enable debugging output.
useCNVScopePublicData	Use files from the CNVScopePublicData package.

Value

none. Runs the application if the correct files are present.

Examples

```
#see runCNVScopeLocal(useCNVScopePublicData=T).
## Not run:
runCNVScopeShiny(useCNVScopePublicData=T)

## End(Not run)
```

signedRescale *Rescale positive and negative data, preserving sign information.*

Description

Performs a signed rescale on the data, shrinking the negative and positive ranges into the [0,1] space, such that negative is always less than 0.5 and positive is always greater.

Usage

```
signedRescale(
  matrix,
  global_max = NULL,
  global_min = NULL,
  global_sigma = NULL,
  global_mu = NULL,
  max_cap = NULL,
  method = "minmax",
  tan_transform = F,
  global_sigma_pos = NULL,
  global_sigma_neg = NULL,
  asymptotic_max = T
)
```

Arguments

matrix	A matrix to be transformed
global_max	the global maximum (used if scaling using statistics from a large matrix upon a submatrix).
global_min	the global minimum
global_sigma	the global signma
global_mu	the global mu
max_cap	the maximum saturation– decreases the ceiling considered for the scaling function. Useful to see greater differences if an image is too white, increase it if there is too much color to tell apart domains.
method	method to perform the rescaling. Options are "minmax" (default), "tan" for tangent, and "sd" for standard devation
tan_transform	apply a tangent transformation?
global_sigma_pos	The positive global sigma. See getGlobalRescalingStats.
global_sigma_neg	The negative global sigma. See getGlobalRescalingStats.
asymptotic_max	make the maximum value in the matrix not 1, but rather something slightly below.

Value

transformedmatrix A transformed matrix.

Examples

```
mat<-matrix(c(5,10,15,20,0,40,-45,300,-50),byrow=TRUE,nrow=3)
rescaled_mat<-signedRescale(mat)
mat
rescaled_mat<-signedRescale(abs(mat))
```

---

underscored_pos_to_GRanges

Convert coordinates in underscored format to a GRanges object.

Description

This function creates a new GRanges object from a character vector of coordinates in the form "chr1_0_5000" and creates a GRanges object from them.

Usage

```
underscored_pos_to_GRanges(
  underscored_positions = NULL,
  extended_data = NULL,
  zeroToOneBasedStart = T,
  zeroToOneBasedEnd = F
)
```

Arguments

underscored_positions	A vector of positions of the form c("chr1_0_5000","chr1_7500_10000","chr1_10000_15000")
extended_data	Optional metadata columns. These columns cannot be named "start", "end", "width", or "element". Passed to GRanges object as ...
zeroToOneBasedStart	Converts a set of underscored positions that begin with zero to GRanges where the lowest positional value on a chromosome is 1. Essentially adds 1 to start
zeroToOneBasedEnd	Adds 1 to the end of the underscored positions

Value

A GRanges object

Examples

```
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
underscored_pos_to_GRanges(colnames(nbl_result_matrix_sign_small))
```

`writeAsymmetricMeltedChromosomalMatrixToDisk` *Write a matrix, with genes, of a submatrix of a whole genome interaction matrix to disk.*

Description

Writes an RData file with a ggplot2 object within.

Usage

```
writeAsymmetricMeltedChromosomalMatrixToDisk(
  whole_genome_matrix,
  chrom1,
  chrom2,
  extra_data_matrix = NULL,
  transpose = F,
  sequential = T,
  debug = T,
  desired_range_start = 50,
  desired_range_end = 300,
  saveToDisk = T,
  max_cap = NULL,
  rescale = T
)
```

Arguments

`whole_genome_matrix`	A matrix to have edges averaged with genomic coordinates in the form chr1_50_100 set as the column and row names.
`chrom1`	first chromosome of the two which will subset the matrix. (this is done in row-column fasion).
`chrom2`	second chromosome of the two which will subset the matrix. (this is done in row-column fasion).
`extra_data_matrix`	A matrix with additional variables about each point, one position per row with as many variables as remaining columns.
`transpose`	transpose the matrix?
`sequential`	disable parallelization with registerDoSEQ()?
`debug`	extra output
`desired_range_start`	start of range for width and height of matrix for downsampling
`desired_range_end`	end of range for width and height of matrix for downsampling

saveToDisk	saves the matrix to disk
max_cap	maximum saturation cap, passed to signedRescale
rescale	perform signedRescale() on matrix?

Value

ggplotmatrix a matrix with values sufficient to create a ggplot2 heatmap with geom_tile() or with ggiraph's geom_tile_interactive()

Examples

```
load(system.file("extdata","grch37.rda",package = "CNVScope"))
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
load(system.file("extdata","ensembl_gene_tx_table_prot.rda",package = "CNVScope"))
writeAsymmetricMeltedChromosomalMatrixToDisk(whole_genome_matrix =
nbl_result_matrix_sign_small,
chrom1 = 1,chrom2 = 1,desired_range_start = 25, desired_range_end = 25)
file.remove("chr1_chr1_melted.RData")
```

---

writeMeltedChromosomalMatrixToDisk

Write a matrix, with genes, of a submatrix of a whole genome interaction matrix to disk.

Description

Writes an RData file with a ggplot2 object within the current directory.

Usage

```
writeMeltedChromosomalMatrixToDisk(
  whole_genome_matrix,
  chrom1,
  chrom2,
  filename,
  extra_data_matrix = NULL,
  transpose = F,
  sequential = T,
  debug = T,
  desired_range_start = 50,
  desired_range_end = 300
)
```

Arguments

whole_genome_matrix	A matrix to have edges averaged with genomic coordinates in the form chr1_50_100 set as the column and row names.
chrom1	first chromosome of the two which will subset the matrix. (this is done in row-column fasion).
chrom2	second chromosome of the two which will subset the matrix. (this is done in row-column fasion).
filename	the filename to be written
extra_data_matrix	A matrix with additional variables about each point, one position per row with as many variables as remaining columns.
transpose	transpose the matrix?
sequential	Disable paralleization with doParallel? registerDoSEQ() is used for this.
debug	verbose output for debugging
desired_range_start	the downsampled matrix must be of this size (rows & cols) at minimum
desired_range_end	the downsampled matrix must be of this size (rows & cols) at maximum

Value

ggplotmatrix a matrix with values sufficient to create a ggplot2 heatmap with geom_tile() or with ggiraph's geom_tile_interactive()

Examples

```
load(system.file("extdata","grch37.rda",package = "CNVScope"))
load(system.file("extdata","nbl_result_matrix_sign_small.rda",package = "CNVScope"))
load(system.file("extdata","ensembl_gene_tx_table_prot.rda",package = "CNVScope"))
writeMeltedChromosomalMatrixToDisk(whole_genome_matrix = nbl_result_matrix_sign_small,
chrom1 = 1,chrom2 = 1,desired_range_start = 25, desired_range_end = 25)
file.remove("chr1_chr1_melted.RData")
```

Index