Package 'CopulaHiC'

January 13, 2019

Type Package

Title Normalization avoiding modelling and detection of differential interactions in Hi-C data via copula models.

Version 0.1.0

Author Rafal Zaborowski

Maintainer Rafal Zaborowski <r.zaborowski@mimuw.edu.pl>

Description Normalization, detection of differential interactions, visualization and manipulation routines for Hi-C data.

License MIT

Encoding UTF-8

LazyData true

Imports magrittr,
reticulate,
Matrix,
fields,
fitdistrplus,
VineCopula,
parallel,
igraph,
raster,
latex2exp,
intervals

Suggests ggplot2,
reshape2,
gridExtra,
knitr,
rmarkdown

RoxygenNote 6.1.1

VignetteBuilder knitr

R topics documented:

`balance` *Inserts 0 columns and rows after last row/column to symmetrize matrix.*

Description

Inserts 0 columns and rows after last row/column to symmetrize matrix.

Usage

```
balance(mtx, N = NULL)
```

Arguments

`mtx`	matrix in dense format to be symmetrized
`N`	positive integer; additional argument for symmetrizing matrix to desired N x N dimension; N need not be larger than `ncol(mtx)` or `nrow(mtx)` in which case submatrix `mtx[1:N,1:N]` will be extraced

Value

N by N matrix which is either submatrix of `mtx` or `mtx` extended with 0's row and/or columns

Examples

```
mtx1 <- matrix(1:24, ncol = 4)
mtx2 <- matrix(1:24, nrow = 4)
print(mtx1)
print(mtx2)
balance(mtx1)
balance(mtx2)
balance(mtx1, N = 8)
balance(mtx1, N = 3)
```

`best_fit_bilinear` *Fits bilinear model to set of x,y points.*

Description

Fits bilinear model to set of x,y points.

Usage

```
best_fit_bilinear(x.vec, y.vec, truncate.left = 0, truncate.right = 0)
```

Arguments

`x.vec`	numeric vector of x coordinates
`y.vec`	numeric vector of y coordinates, must be the same length as x
`truncate.left`	positive integer - number of points to exclude from left hand side
`truncate.right`	positive integer - number of points to exclude from right hand side

Value

list with two componenets: numeric 2 by 2 matrix of coefficients, where row indicate model (left or right) and columns are intercept and slope; numeric vector intersection.x with: x coordinate of first point in left model closest to y (from right hand side), x coordinate of intersection point between left and right models and x coordinate of first point in right model closest to y (from left hand side); these points may be used as cutoff

See Also

The code was taken from https://stackoverflow.com/questions/15874214/piecewise-function-fitting-with-nl (with minor modifications).

Examples

```
# fix parameters of left linear model
a.left <- 0
b.left <- 0.8
# fix parameters of right linear model
a.right <- 15
b.right <- 0.1
# make models
x.left <- 1:20
y.left <- a.left + b.left * x.left
x.right <- 25:45
y.right <- a.right + b.right * x.right
# add some noise
y.left <- y.left + rnorm(length(y.left))
y.right <- y.right + rnorm(length(y.right))
# get y vector
```

```
x <- c(x.left, x.right)
y <- c(y.left, y.right)
# find best fit bilinear model
bf.model <- best_fit_bilinear(x, y)
print(bf.model[["coefficients"]])
print(bf.model[["intersection.x"]])
# plot results: points
plot(x, y, cex = 0.1)
# plot left model
abline(a = a.left, b = b.left, col = "blue")
# plot right model
abline(a = a.right, b = b.right, col = "green")
# plot left model fit
abline(a = bf.model[["coefficients"]]["left","intercept"], b = bf.model[["coefficients"]]["left","slope"], col =
# plot left model fit
abline(a = bf.model[["coefficients"]]["right","intercept"], b = bf.model[["coefficients"]]["right","slope"], col
```

`bootstrap_interactions` *Hi-C interactions bootstrapping.*

Description

Randomly samples interactions from data frame with atomic interactions into `length(ratio)` data frames with interactions in such a way that i-th dataframe have `ratio[i]` fraction of interactions of initial atomic interactions data frame.

Usage

```
bootstrap_interactions(interactions, ratio = c(0.5, 0.5))
```

Arguments

`interactions`	data frame containing row with i and j coordinate for every single interaction
`ratio`	numeric vector indicating on how many atomic interaction sets should initial atomic interaction set be divided; each entry of ratio vector contains fraction of interaction to be put in corresponding atomic interactions set; ratio vector must sum to 1 and all its entries must be larger than one

Value

data frame representing sparse Hi-C maps (with i, j, val columns) belonging to corresponding bootstrapped interactions subset - ratio.number column indicates index of fraction from ratio vector

Examples

```
# create data frame with artificial interactions, where val
# indicates total number of interactions between bins i and j
sparse.mtx <- data.frame(i = c(1,3,5,7,8,9,11), j = c(7,2,1,1,3,10,9), val = c(10,8,3,1,1,2,20))
atomic.interactions <- sparse2interactions(sparse.mtx)
print(head(atomic.interactions))
b1 <- bootstrap_interactions(atomic.interactions)
print(b1)
ratios.desired <- c(0.4,0.3,0.2,0.1)
b2 <- bootstrap_interactions(atomic.interactions, ratio = ratios.desired)
print(b2)
sum.interactions <- nrow(atomic.interactions)
ratios.sampled <- sapply(split(b2, b2$ratio.number), function(x){ sum(x$val) / sum.interactions })
print(ratios.desired)
print(ratios.sampled)
```

`bootstrap_sparse` *Bootstraps interactions from contact map given in sparse format.*

Description

For details of bootstrapping procedure see `bootstrap_interactions`.

Usage

```
bootstrap_sparse(sparse.mtx, ratio = c(0.5, 0.5))
```

Arguments

`sparse.mtx`	data.frame Hi-C contact map in sparse format with mandatory columns i, j, val
`ratio`	numeric vector indicating on how many atomic interaction sets should initial atomic interaction set be divided; each entry of ratio vector contains fraction of interaction to be put in corresponding atomic interactions set; ratio vector must sum to 1 and all its entries must be larger than one

Value

list with data frames (Hi-C maps in sparse format) containing sampled interactions (according to specified ratio vector)

See Also

`bootstrap_interactions` for details of Hi-C interactions bootstrapping procedure

Examples

```
sparse.mtx <- data.frame(i = c(1,3,5,7,8,9,11), j = c(7,2,1,1,3,10,9), val = c(10,8,3,1,1,2,20))
bootstrapped <- bootstrap_sparse(sparse.mtx)
print(bootstrapped)
```

`compartment_ranges` *Calculates ranges of each consecutive compartment along given pc vector.*

Description

Entries with the same sign (i.e. positive or negative) comprise the same compartment. Positives are assigned to A compartment and negatives to B compartment.

Usage

```
compartment_ranges(pc)
```

Arguments

pc	numeric, compartment vector (eigenvector)

Value

data.frame where each row corrspond to interval of consecutive same sign values of eigenvector; columns are start, end and compartment

Examples

```
# make artificial eigenvector
ev <- c(-0.3,-0.5,0.2,0.3,0.4,0.4,-0.5,0.2,0.1,0.3,-0.9,-0.7)
compartment_ranges(ev)
```

`copula_pvals` *Compute depletion or enrichment p-value given copula model.*

Description

Calculates p-value of enrichment or depletion of given point/s with u,v coordinates given copula F(U,V):

- depletion probability is defined as: P(U < u, V > v) = F(U < u, V < 1) - F(U < u, V < v), so its upper left rectangle of copula distribution
- enrichment probability is defined as: P(U > u, V < v) = F(U < 1, V < v) - F(U < u, V < v), so its lower right rectangle of copula distribution

NOTES:

- copula is symmetric with respect to U = V
- enrichment is relative to condition u = F(X < x), in other words enriched means chromatin is enriched in interactions in conditon X comparing with Y
- P(U,V) distribution can be plotted as U horizontal and increasing from left (start at 0) to right (ends at 1) and V vertical and increasing from bottom (start at 0) to top (ends at 1)
- both uv and model is for single (and the same) diagonal
- uv must all be points for either top left (depletion) or bottom right (enrichment) part of distribution

For calculation of copula cdf `VineCopula::::BiCopCDF` is used.

Usage

```
copula_pvals(uv, copula.model, copula.tail = "upper.left")
```

Arguments

`uv`	matrix of dimension n x 2 with U,V r.v. ~ Uniform(0,1), see `VineCopula::pobs` for generation of U,V from X,Y; here U,V matrix must be such that either U >= V (lower.right corner) or U <= V (upper.left corner)
`copula.model`	object of class `VineCopula::BiCop`
`copula.tail`	character indicating tail (corner) of copula to calculate cdf, either "upper.left" or "lower.right"

Value

numeric vector of p-value/s of length equal to `nrow(uv)`

Examples

```
library("copula")
library("MASS")
# make bivariate standard normal copula of highly correlated variables (0.8)
cop <- BiCop(1, 0.8)
# convert to package copula class --> for illustration purposes
cop.copula.object <- copulaFromFamilyIndex(cop$family, cop$par)
# illustrate copula
persp(cop.copula.object, dCopula) # 3D
```

```
contourplot2(cop.copula.object, dCopula, col.regions = terrain.colors) # 2D heatmap
# simulate sample of size 10000 and draw copula 2d density plot (heatmap)
sample.cop <- data.frame(rCopula(10000, cop.copula.object))
plot_copula_density(sample.cop) # 2D heatmap with ggplot
# now simulate sample from bivariate standard normal with much lower correlation than that of copula
sigma <- matrix(c(1, 0.4, 0.4, 1), nrow = 2)
xy <- mvrnorm(500, mu = c(0, 0), Sigma = sigma, empirical = T)
# convert X,Y to U,V ~ Uniform(0,1) using copula::pobs
uv <- pobs(xy)
# keep only upper V >= U (\code{copula_pvals} calculates p-values separately for V >= U and U >= V)
uv <- uv[uv[,2] >= uv[,1],]
# illustrate observations on top of copula density
uv.df <- data.frame(uv)
colnames(uv.df) <- c("U","V")
plot_copula_density(sample.cop) + geom_point(aes(x = U, y = V), data = uv.df, size = 0.5)
# calculate p-values of observations given the copula model (use only )
uv.df$pval <- copula_pvals(uv, cop)
# convert to pvalues to negative log10(pval)
uv.df$neg.log.pval <- -log10(uv.df$pval)
# illustrate observations significance on top of copula model
plot_copula_density(sample.cop) + geom_point(aes(x = U, y = V, color = neg.log.pval), data = uv.df, size = 0.3) +
```

`decay_correlation.HiCcomparator`

Calculates correlations between diagonals.

Description

Computes correlations (Pearson, Spearman, Kendall) and significances of corresponding diagonals between 2 Hi-C maps of HiCcomparator object.

Usage

```
## S3 method for class 'HiCcomparator'
decay_correlation(hic.comparator)
```

Arguments

`hic.comparator`	object of type HiCcomparator

Value

dataframe with following columns: diagonal, pcc, pearson.pval, rho, spearman.pval, tau, kendall.pval, name which can be used to conveniently visualise dependancy between 2 Hi-C maps being compared (see examples)

See Also

`HiCcomparator` on how to construct HiCcomparator object

Examples

```
first create HiCcomparator object - see ?HiCcomparator for examples
library("ggplot2")
library("reshape2")
decay.cors <- decay_correlation(hic.comparator)
# wide to long
decay.cors.long <- reshape2::melt(decay.cors[c("name","diagonal","pcc","rho","tau")], id.vars = c("name","diago
# remove 0 diagonal (as it is non informative anyways) and illustrate results
ggplot(decay.cors.long[decay.cors.long$diagonal != 0,],
  aes(x = diagonal, y = coefficient, color = correlation)) +
 geom_point(size = 0.3) +
 facet_wrap(~ name, ncol = 1, scales = "free") +
 theme(legend.position = "bottom")
```

---

dense2sparse *Converts matrix given in dense format to sparse format data frame.*

Description

This function only keeps non-zero cells. In case given dense matix is symmetric dense2sparse will return upper triangular part of the matrix (i.e. where rows <= columns)

Usage

```
dense2sparse(mtx, add.diagonal = TRUE, name = NULL)
```

Arguments

mtx	matrix in dense format
add.diagonal	logical, if true an additional column indicating diagonal of each cell will be appended to resulting data frame
name	character, additional argument, if specified column with name will be appended to resulting data frame

Value

data.frame with columns c("i","j","val") and optionally c("diagonal","name") columns; every row of resulting dataframe corresponds to cell in given dense matrix with i-th row, j-th column and value val

Examples

```
dense2sparse(matrix(1:24, ncol = 3))
dense2sparse(matrix(1:24, ncol = 3), name = "some.matrix")
dense2sparse(matrix(1:24, ncol = 3), add.diagonal = FALSE)
# symmetric matrix
mtx.sym <- matrix(1:25, ncol = 5)
```

```
mtx.sym <- mtx.sym + t(mtx.sym)
dense2sparse(matrix(mtx.sym))
```

`differential_interactions.HiCcopula`
Finds significantly interacting rectangle-like regions.

Description

This function works in 3 steps:

- first it calculates differential map,
- then it takes negative log10 p-value vector of cells, sorts it, divides on negative and positive parts and lastly fits bilinear model to each of them
- finally it retains only those cells that are to the left (right) of intersection point of bilinear model in positive (negative) p-values vector and searches for connected components in both of them separately

Fitting bilinear model is performed using `best_fit_bilinear` function, while for connected components search `raster` package is used. After detection of significantly interacting regions (connected components) one may further filter list to only retain those with number of non zero cells (n.cells column in interacting.regions data frame) larger than some threshold. There are 3 possible ways of selecting significant interactions (cells):

- bilinear model is used to determine significance threshold and then this threshold is compared with `pval` parameter - if threshold is less significant than `pval` then threshold is substituted with `pval` - this is the default behaviour,
- only `pval` is used as a significance threshold, i.e. hard thresholding,
- only bilinear model is used to determine significance threshold (unrecomended, as it may yield non significant interactions).

When using option 1 and 3 its recommended to plot the fit (enabled by default). An indication of properly determined significance threshold would be when red vertical line (the significance threshold) is located to the right side of grey vertical line.

Usage

```
## S3 method for class 'HiCcopula'
differential_interactions(hic.copula,
  plot.models = TRUE, pval = 0.05, sig.thr.selection = c(1, 2, 3)[1],
  which.significance = c("qval", "pval")[1], cc.direction = c(4, 8)[1])
```

Arguments

hic.copula	object of class HiCcopula
plot.models	logical if true then plot bilinear model fit for every matrix in hic.copula object; it will plot models for depleted and enriched models separately; if you want to save this results to file open device before calling this function (see for instance pdf) and close device after function call (see dev.off)
pval	numeric, p-value cutoff to qualify interaction as significant
sig.thr.selection	numeric, if 3 then only use bilinear model fit to establish p-value cutoff for significant interactions, if 2 then select significant interactions using only pval parameter, if 1 (default) use bilinear model, but if p-value threshold is larger than pval, use pval instead
which.significance	character either "qval" or "pval" indicating, which of the 2 shold be used as a measure of interaction significance
cc.direction	specifies criterium for two cells to be considered as neighbors during connected components search, for details see directions parameter of raster::clump function

Value

list with number of entries equal to hic.copula$names; each entry is a list with 2 elements: interacting.regions - data frame containing rows with rectangle like regions of significant interactions with coordinates n.cells (number of non zero cells inside rectangle), start.x, end.x, start.y, end.y, effect; connected.components list with cells comprising given connected component; connected components list is named list where each entry name is unique id, which can be mapped to row in interacting.regions (its row names)

See Also

best_fit_bilinear for fitting bilinear model, raster::raster and raster::clump for connected components search

Examples

```
# first create HiCcopula object - see ?HiCcopula for examples
di <- differential_interactions(hic.copula)
di18 <- di[["18"]]
# if you want to plot results create pvalue map (see hicdiff function) and from that dense map for some chromosome
plot_diff_map(dense)
# plot regions having at least 10 significant cells in connected component
plot_regions(di18[di18$n.cells >= 10,2:6], pal.colors = c("blue","red"))
```

dominating_signal.HiCcomparator *Calculates coverage or decay of Hi-C maps.*

Description

Computes coverages or decays of every Hi-C maps in both data sets of given HiCcomparator object. Coverage is defined as sum of contacts on given bin. Decay is sum or mean of contacts for every diagonal.

Usage

```
## S3 method for class 'HiCcomparator'
dominating_signal(hic.comparator,
  which.signal = c("coverage", "decay")[1])
```

Arguments

hic.comparator	object of type HiCcomparator

Value

dataframe with following columns: (i, sum.contacts, mean.contacts, sd.contacts, name, dataset), which can be used to conveniently visualise coverages or decays (see examples)

See Also

HiCcomparator on how to construct HiCcomparator object

Examples

```
# first create HiCcomparator object - see ?HiCcomparator for examples
coverage <- dominating_signal(hic.comparator)
# visualise results
library("ggplot2")
ggplot(coverage, aes(x = i, y = sum.contacts, color = dataset)) +
geom_point(size = 0.5) +
 geom_smooth(alpha = 0.5) +
 facet_wrap(~ name, ncol = 1, scales = "free") +
 theme(legend.position = "bottom")
# get decay
decay <- dominating_signal(hic.comparator, which.signal = "decay")
ggplot(decay[decay$diagonal != 0,], aes(x = diagonal, y = mean.contacts, color = dataset)) +
geom_point(size = 0.5) +
 scale_x_log10() +
 scale_y_log10() +
 facet_wrap(~ name, ncol = 1, scales = "free") +
 theme(legend.position = "bottom")
```

do_pca *PCA analysis of Hi-C contact maps.*

Description

Performs PCA on Hi-C contact map as described in Liebermann-Aiden et al 2009. More specifically it runs following routines on dense matrix:

- removes unmappable regions (all zeros rows and columns)
- divides each diagonal of every cell by its corresponding mean of cells on diagonal
- converts matrix from 2. into PCC matrix
- performs PCA on such matrix
- fills in unmappable regions into PCA object vector/matrix components

Usage

```
do_pca(dense.mtx, ...)
```

Arguments

dense.mtx	numeric matrix - Hi-C contact map
...	optional arguments passed to prcomp

Value

PCA object returned by prcomp function.

See Also

prcomp for how is PCA performed, Lieberman-Aiden E. et al., 2009 "Comprehensive mapping of long-range interactions reveals folding principles of the human genome." for compartment detection in Hi-C contact maps.

Examples

```
# load Hi-C contact maps from npz file
# get sample npz file name
mtx.fname <- system.file("extdata", "MSC-HindIII-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
mtx.sparse.list <- read_npz(mtx.fname, sparse.format = TRUE)
# get matrix for selected chromosome
mtx <- mtx.sparse.list[["18"]]
# do PCA
pca <- do_pca(mtx)
print(pca)
# it is also possible to visualize results
pairs(pca)
```

`HiCcomparator` *An S3 class to represent object for Hi-C maps comparisons.*

Description

HiC comparator object stores Hi-C contact maps from 2 experiments and (optionally) TADs and allows for convenient access to contact matrices, A/B compartments or TADs. HiCcomparator is constructed from npz files containing Hi-C maps in python dict with numpy matrices. Additionally TAD set may be given to HiCcomparator (as list of data frames, where data frames names match those of Hi-C matrices names). One can also choose to determine TADs based on given Hi-C contact maps - only first, only second or determine both and take intersecting intervals between them.

Usage

```
HiCcomparator(path1, path2, tads = NULL, mtx.names = "all",
  which.tads = 4, do.pca = FALSE)
```

Arguments

`path1`	character - path to npz file containing first set of Hi-C maps
`path2`	character - path to npz file containing second set of Hi-C maps
`tads`	list (optional), set of TADs as named list of data frames, each with at least start, end columns
`mtx.names`	character vector with subset of Hi-C maps names to be selected for analysis, by default all matrices are used
`which.tads`	numeric indicating what to do if no TADs are specified: 1 - determine TADs from first set of Hi-C maps, 2 - determine TADs from second set of Hi-C maps, 3 - determine from both sets and then take their intersection, 4 - do not determine TADs
`do.pca`	logical whether to perform PCA for given maps and determine A/B compartments

Value

S3 object of class HiCcomparator

See Also

`read_npz` for reading npz files, `do_pca` on how A/B compartments are determined, `map2tads` how TADs are determined

Examples

```
# get path of first sample maps
mtx1.fname <- system.file("extdata", "IMR90-MboI-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
# get path of second sample maps
mtx2.fname <- system.file("extdata", "MSC-HindIII-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
# get sample TADs
tads <- CopulaHiC::sample_tads[c("IMR90-MboI-1_40kb-raw", "MSC-HindIII-1_40kb-raw")]
# construct HiCcomparator object for chromosomes 18 and 19
hic.comparator <- HiCcomparator(mtx1.fname, mtx2.fname, tads, mtx.names = c("18","19"))
# plot A/B compartments for first and second map in chromosome 19
plot_pc_vector(hic.comparator$pc1.maps1[["19"]]) # first map
plot_pc_vector(hic.comparator$pc1.maps2[["19"]]) # second map
```

HiCcopula *An S3 object to represent Hi-C copula model.*

Description

Models diagonal-wise dependencies between Hi-C data sets with copulas. Model is constructed as follows:

- merge maps1 with maps2
- for each diagonal in diagonals
  - take all points from this diagonal, such that they are non zero in map1 (X) and non zero in map2 (Y)
  - model X with gamma distribution
  - model Y with gamma distribution
  - transform X and Y to U and V ~ Uniform(0,1) - see `VineCopula::pobs`
  - model F(U,V) with copula (see `VineCopula::BiCopSelect`)

Before fitting the model it's recommended to first inspect correlations between analyzed Hi-C maps before fixing this variable. As the ratio of noise / signal in Hi-C data increases rapidly with decay it's unadvised to use all diagonals for modelling. The number of diagonals to be used will depend on chromosome length, resolution and data quality. As a rule of thumb the number of diagonals should not exceed 0.2 times length of chromosome.

Usage

```
HiCcopula(hic.comparator, diagonals = 0.12, include.zero.cells = FALSE,
  n.cores = 1)
```

Arguments

hic.comparator	object of type HiCcomparator
diagonals	fraction or numeric vector or character "all" which diagonals to use to fit models, by default fraction of chromsome length is used to indicate number of diagonals.
include.zero.cells	logical, whether to include cells when merging maps (see merge.HiCcomparator)
n.cores	numeric number of cores to distribute model computations

Value

S3 object of class HiCcopula

See Also

HiCcomparator on how to construct HiCcomparator object, fitdistrplus::fitdist on distribution fitting, VineCopula::pobs on pseudo observations generation and VineCopula::BiCopSelect on finding optimal fit bivariate copula

Examples

```
# first create HiCcomparator object - see ?HiCcomparator for examples
# construct model
hic.copula <- HiCcopula(hic.comparator)
# load below packages for visualisation
library("fitdistrplus")
library("VineCopula")
# illustrate gamma fit of X and Y for chromosome 18, diagonal 5
plot(copula$model[["18"]][["5"]]$marginal.x)
plot(copula$model[["18"]][["5"]]$marginal.y)
# illustrate copula fit of U and V for chromosome 18, diagonal 5
plot(copula$model[["18"]][["5"]]$bf.copula)
```

---

hicdiff.HiCcopula *Computes differential (p-value) map.*

Description

Given HiCcopula object calculates depletion/enrichment p-values of cell along diagonals w.r.t. background copula models (separate for every diagonal, see HiCcopula for details).

Usage

```
## S3 method for class 'HiCcopula'
hicdiff(hic.copula, marginal.distr = c("fit",
  "obs")[1])
```

Arguments

hic.copula	object of class HiCcopula
marginal.distr	character wither fit or obs; if fit then fitted gamma distribution for X and Y is used to convert them to U and V respectively

Value

list of data frames corresponding to Hi-C contact maps; each data frame contain columns: i, j, name, val.x, val.y, u, v, effect, p.value, p.value.corrected

See Also

HiCcopula on how to construct HiCcopula, maps_difference_diagonal and copula_pvals on how p-values are calculated

Examples

```
# first create HiCcopula object - see ?HiCcopula for examples
# calculate p-values and select chromosome of interest (18 in this example)
md <- hicdiff(hic.copula)[["18"]]
# convert corrected p-values to -log10(pvals)
md$neg.log.cor.pvals <- md$p.value.corrected
# make neg.log.cor.pvals negative for depleted cells
md[md$effect == "depletion", "neg.log.cor.pvals"] <- -md[md$effect == "depletion", "neg.log.cor.pvals"]
# convert sparse matrix to dense matrix
dense <- sparse2dense(md[c("i","j","neg.log.cor.pvals")], N = hic.copula$maps.dims[["18"]][1,1])
# plot results
plot_diff_map(dense)
```

---

hicdiff2mtx *Converts significance maps to dense matrices.*

Description

Given list of significance maps in sparse format produced by hicdiff function converts them to dense matrix format.

Usage

```
hicdiff2mtx(hicdiff.list, maps.dims, val.column = c("p.value.corrected",
  "p.value")[1], neg.log = TRUE, mark.depleted = TRUE)
```

Arguments

hicdiff.list	list of data frames, output from hicdiff function
maps.dims	list of data frames, usually an attribute of HiCcopula object that was used to produce hicdiff.list
val.column	character indicating which column of sparse significance map to use as cell value for dense matrix (one of p.value or p.value.corrected)
neg.log	logical wheter to apply -log10(.) to val.column
mark.depleted	logical whether to mark depleted cells as negative

Value

list with dense matrices

See Also

hicdiff for generation of hicdiff.list

Examples

```
# first create HiCcopula object - see ?HiCcopula for examples
# then produce hicdiff.list
hicdiff.list <- hicdiff(hic.copula)
dense.hicdiff.list <- hicdiff2mtx(hicdiff.list)
# elements of dense.list can be visualized
plot_diff_map(dense.hicdiff.list[["18"]], breaks = 100)
```

---

image_plot_na *Wrapper for fields::image.plot function.*

Description

Allows to mark -Inf, Inf and NA values in heatmaps with different colors. It will automatically detect such values and assign them color.

Usage

```
image_plot_na(z, breaks, col, na.color = "black",
  neg.inf.color = "gold", pos.inf.color = "darkgreen",
  colorbar = TRUE, ...)
```

Arguments

z	numeric matrix to be plotted; may contain -Inf, Inf and NA values
breaks	numeric vector of breaks for colorscale
col	character vector of hex color strings; usually generated from some color pallette; it's length must be equal to `length(breaks) - 1`
na.color	color for NA values
neg.inf.color	color for -Inf values
pos.inf.color	color for Inf values
...	additional arguments passed to `fields::image.plot`

See Also

`fields::image.plot` for function which finally handles heatmap plotting

Examples

```
# matrix of data
mtx <- toeplitz(c(5:1))
# make lower triangle part negative
mtx[lower.tri(mtx)] <- -mtx[lower.tri(mtx)]
# make some cells -Inf
mtx[matrix(c(2,1,2,2,3,2), ncol = 2, byrow = TRUE)] <- -Inf
# make some cells Inf
mtx[matrix(c(3,5,4,5), ncol = 2, byrow = TRUE)] <- Inf
# make some cells NA
mtx[matrix(c(1,3,2,3,5,3), ncol = 2, byrow = TRUE)] <- NA
print(mtx)
# prepare breaks --> symmetric, from -5 to 5, spaced by 2, with 0 in the middle
# one can also introduce here log, sqrt or other scales by applying proper transformations
breaks <- sort(c(0,seq(-5,5,2)))
print(breaks)
# prepare symmetric color pallette indicating negtive values with blue, middle with white and positive with red
colors.pal = c("blue","white","red")
pal = colorRampPalette(colors.pal)
colors <- pal(length(breaks) - 1L)
# finally plot matrix
image_plot_na(mtx, breaks, colors)
```

IMR90-MboI-1_40kb-raw_maps

Npz file with sample Hi-C contact maps dataset.

Description

Npz file containing Hi-C contact maps of human IMR90 in 40kb resolution. The data comes from Rao et al., 2014 study and was processed by Imakaev et al., 2012 pipeline without iterative correction step (so it is raw data). It contains contact maps for chromosomes 17, 18, 19.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63525

See Also

https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.savez_compressed.html for npz format description, Rao et al., 2014 "A three-dimensional map of the human genome at kilobase resolution reveals prinicples of chromatin looping" for study where this dataset comes from, Imakaev et al., 2012 "Iterative correction of Hi-C data reveals hallmarks of chromosome organization." for Iterative Correction of Hi-C contact maps and https://mirnylab.bitbucket.io/hiclib/index.html for its python implementation

Examples

```
# get file name
mtx.fname <- system.file("extdata", "IMR90-MboI-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
```

---

interactions2tads *Maps interactions to TADs.*

Description

Maps cells of a contact map given in sparse format to TADs. User must provide interactions data frame and TADs for single and the same chromosome, otherwise the function will throw an error or behaviour will undefined.

Usage

```
interactions2tads(mtx.sparse, tads, cols = c("val"))
```

Arguments

mtx.sparse	data.frame with Hi-C contact map in sparse format; must have i and j columns, i.e. cell coordinates
tads	data frame containg TADs, for single chromosome; it is assumed that first TAD on every chromosome start at 0 and end of TAD equals start of next TAD (in case if there is no break between consecutive TADs); also user is responsible for converting TAD boundary coordinates to bins
cols	character vector with columns from mtx.sparse or tads data.frames to be included in resulting data frame; by default only val column is included

Value

data.frame with following columns: i, j, val, tad.id, start, end, name (and additional if cols specified), where each row corresponds to cell in Hi-C contact map with i, j coordinates and cells are assigned to TADs (cells which do not belong to any TAD are discarded)

Examples

```
# create artificial interactions set
sparse.mtx <- data.frame(i = c(1,3,4,4,8,9,11), j = c(7,2,4,5,3,10,9), val = c(10,8,3,1,1,2,20), compartment = c(
# create artificial TAD set in 20000 bp resolution
resolution <- 20000
tads <- data.frame(start = c(0,2,10) * resolution, end = c(2,8,13) * resolution, name = as.character(c(1,1,1)))
print(tads)
# convert basepairs to bins
tads$start <- tads$start / resolution
tads$end <- tads$end / resolution
print(tads)
# map interactions to TADs
interactions2tads(sparse.mtx, tads)
# map interactions to TADs and keep also compartment columns
interactions2tads(sparse.mtx, tads, cols = c("val","compartment"))
```

`intersect_tads` *Finds intersection between 2 sets of TADs.*

Description

Intersection is defined for the same chromosomes, for example for 2 TAD sets from chromosome 18 if there is TAD in set1 with coordinates 100, 120 (start, end respectively) and in set2 there is a TAD with coordinates 110, 140 then their intersection is interval with coordinates: 110, 120.

Usage

```
intersect_tads(tads1, tads2)
```

Arguments

`tads1`	data frame of TAD set 1 with columns: start, end
`tads2`	data frame of TAD set 2 with columns: start, end

Value

data frame with start, end columns containing TAD intersection intervals

See Also

`intervals::Intervals`, `intervals::interval_intersection` for functions used to find intersection between 2 sets of intervals

Examples

```
# simple artificial data set of TADs
tads1 <- data.frame(start = c(1,10,30), end = c(5,15,35))
tads2 <- data.frame(start = c(3,14,28), end = c(12,18,40))
tads.intersection <- intersect_tads(tads1, tads2)
print(tads.intersection)
```

`left.max` *Find first local maximum.*

Description

Seeks first local maximum starting from the right hand side of given vector and moving towards left hand side.

Usage

```
left.max(v)
```

Arguments

v	numeric vector

Value

numeric value of first local maximum in v starting from the end of v and going left; NA if v is nonincresing starting at last element of v and going left (i.e. v is either constant or there is minmum first)

See Also

`CopulaHiC::local.min`, `CopulaHiC::local.max` for finding local minma and maxima in vector v

Examples

```
# maximum in 7 (index 2) starting from 1 (index 8) and moving with decreasing indices
v1 <- c(5,6,7,6,4,3,2,1)
# no maximum or minimum
v2 <- c(2,2,2,2,2,2)
# minimum at 2 (index 3) and no maximum starting at 6 (index 7)
v3 <- c(4,3,2,3,4,5,6)
# starting from rightmost element (3 with index 14) and going left, minimum first (in -3, index 9) then maximum (i
v4 <- c(3,4,5,6,4,3,2,-1,-3,-2,0,1,2,3)
print(left.max(v1))
print(left.max(v2))
print(left.max(v3))
print(left.max(v4))
```

local.max *Finds local maximas indices.*

Description

Seeks for local maximas in vector v using simple second order difference.

Usage

```
local.max(v)
```

Arguments

v	numeric vector

Value

numeric vector with indices of local maxima elements of vector v (i.e. where elements of second order difference equals -2)

Examples

```
v <- c(1,2,3,4,5,6,5,3,2,-1,-4,-3,-1,2,4,10,12,11,5,3)
idx <- local.max(v)
print(idx)
print(v[idx])
```

local.min *Finds local minimas indices.*

Description

Seeks for local minimas in vector v using simple second order difference.

Usage

```
local.min(v)
```

Arguments

v	numeric vector

Value

numeric vector with indices of local minima elements of vector v (i.e. where elements of second order difference equals 2)

Examples

```
v <- c(1,2,3,4,5,6,5,3,2,-1,-4,-3,-1,2,4,10,12,11,5,3)
idx <- local.min(v)
print(idx)
print(v[idx])
```

`map2tads` *Determines TAD boundaries using Insulation Score.*

Description

For details on Insulation Score approach of TAD boundaries detection see Crane et al. 2015 "Condensin-driven remodelling of X chromosome topology during dosage compensation" methods section, paragraph TAD calling (insulation square analysis).

Usage

```
map2tads(dense.mtx, resolution = 40000, window.bp = 500 * 1000,
  delta.bp = 100 * 1000, without_unmappable = TRUE)
```

Arguments

`dense.mtx`	numeric matrix in dense format representing Hi-C contact map
`resolution`	numeric resolution of Hi-C contact map in base pairs
`window.bp`	numeric size of sliding window in base pairs
`delta.bp`	numeic size of delta window in base pairs

Value

data frame with TAD positions containing start, end columns

Examples

```
# get Hi-C contact map
sparse.mtx <- CopulaHiC::sample_hic_maps[["MSC-HindIII-1_40kb-raw"]][["18"]]
dense.mtx <- sparse2dense(sparse.mtx[c("i","j","val")], N = 1952)
# get tads
tads <- map2tads(dense.mtx)
# plot results
plot_contact_map(dense.mtx)
plot_regions(tads)
```

`maps_difference_diagonal` *Computes p-values given copula model.*

Description

Given copula model for diagonal of Hi-C contact map it calculates deviation from this model (p-values) in both directions (i.e. depletion and enrichment). It divides U,V matrix on 3 groups: U > V (enrichment), U == V (no effect), U < V (depletion) and calls `copula_pvals` for each group. Afterwards it merge results.

Usage

```
maps_difference_diagonal(uv, copula.model, mht.correction = TRUE)
```

Arguments

`uv`	matrix of dimension n x 2 with U,V r.v. ~ Uniform(0,1), see `VineCopula::pobs` for generation of U,V from X,Y
`copula.model`	object of class `VineCopula::BiCop`
`mht.correction`	logical whether to apply Benjamini Hochberg correction for multiple hypothesis testing since we are testing a number of hypothesis (diagonal cells) against null model

Value

data frame with columns: u, v, effect, p.value, p.value.corrected

See Also

`copula_pvals` for p-value calculation

Examples

```
# see copula_pvals function
```

`mean_expected` *Constructs Toeplitz matrix with means on diagonals (calculated as arithmetic mean of diagonal).*

Description

Constructs Toeplitz matrix with means on diagonals (calculated as arithmetic mean of diagonal).

Usage

```
mean_expected(dense.mtx)
```

Arguments

dense.mtx	matrix in dense format

Value

numerical dense matrix where all entries on i-th diagonal contain mean of i-th diagonal

See Also

toeplitz for more information about toeplitz matrices

Examples

```
mtx1 <- toeplitz(c(1,2,3,4))
mean_expected(mtx1)
mtx2 <- matrix(1:16, ncol = 4)
mean_expected(mtx2)
```

---

merge.HiCcomparator *Finds regions, which interacts in both experiments (maps).*

Description

Merges Hi-C contact maps data frames 1 and 2 of HiCcomparator object by i, j, diagonal, name columns.

Usage

```
## S3 method for class 'HiCcomparator'
merge(x, include.zero.cells = FALSE)
```

Arguments

x	HiCcomparator object
include.zero.cells	logical, whether to include cells, which have non zero number of contacts in one map, but not the other, not recommended

Value

list of data frames with merged contact maps data in sparse format

See Also

HiCcomparator on how to construct HiCcomparator object

Examples

```
# first create HiCcomparator object - see ?HiCcomparator for examples
merged <- merge(hic.comparator)
```

MSC-HindIII-1_40kb-raw_maps

Npz file with sample Hi-C contact maps dataset.

Description

Npz file containing Hi-C contact maps of human MSC in 40kb resolution. The data comes from Dixon et al., 2015 study and was processed by Imakaev et al., 2012 pipeline without iterative correction step (so it is raw data). It contains contact maps for chromosomes 17, 18, 19.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE52457

See Also

https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.savez_compressed.html for npz format description, Dixon et al., 2015 "Chromatin architecture reorganization during stem cell differentiation" for study where this dataset comes from, Imakaev et al., 2012 "Iterative correction of Hi-C data reveals hallmarks of chromosome organization." for Iterative Correction of Hi-C contact maps and https://mirnylab.bitbucket.io/hiclib/index.html for its python implementation

Examples

```
# get file name
mtx.fname <- system.file("extdata", "MSC-HindIII-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
```

MSC-HindIII-1_40kb-raw_tads

Csv file with sample TADs dataset.

Description

Contains TAD boundaries of MSC-HindIII-1_40kb-raw_maps determined using Insulation Score (Crane et al. 2015 "Condensin-driven remodelling of X chromosome topology during dosage compensation") with parameters of window size 1Mbp and delta window size 200 Kbp.

Examples

```
# get file name
tads.fname <- system.file("extdata", "MSC-HindIII-1_40kb-raw.tadIS", package = "CopulaHiC", mustWork = TRUE)
```

pc2mtx

Matches compartment with entries of contact map given in sparse format.

Description

Compartment vector indices correspond to bins along chromosome.

Usage

```
pc2mtx(pc, sparse.mtx)
```

Arguments

pc	numeric, compartment vector or principal component (eigenvector)
sparse.mtx	data.frame, matrix in sparse format containg manadatory fields i and j

Value

data.frame - contact map in sparse format with additional columns for every row (dense matrix cell): compartment.i, compartment.j, compartment

Examples

```
# make artificial eigenvector --> its length is 12 so it represents artificial chromosome with 12 bins
pc <- c(-0.3,-0.5,0.2,0.3,0.4,0.4,-0.5,0.2,0.1,0.3,-0.9,-0.7)
# make artificial sparse contact map
sparse.mtx <- data.frame(i = c(1,3,5,7,8,9,11), j = c(7,2,1,1,3,10,9), val = c(10,8,3,1,1,2,20))
print(sparse.mtx)
pc2mtx(pc, sparse.mtx)
```

`plot_contact_map` *Plots simple contact map or fold-change contact map with log2 scale.*

Description

Plots simple contact map or fold-change contact map with log2 scale.

Usage

```
plot_contact_map(contact.map, fc = FALSE, breaks = 100,
  zeros.na = TRUE, colors.pal = c("white", "red"), ...)
```

Arguments

`contact.map`	numeric matrix (with non-negative entries) to be plotted
`fc`	logical whether given matrix is fold change
`breaks`	numeric number of breaks on color scale
`colors.pal`	colors for pallette
`title`	character plot title

See Also

`CopulaHiC::iamge_plot_na` for function handling heatmap plotting

Examples

```
# get sample contact map (MSC replicate 1) for chromosome 18
mtx1.sparse <- CopulaHiC::sample_hic_maps[["MSC-HindIII-1_40kb-raw"]][["18"]]
# convert to dense
mtx1 <- sparse2dense(mtx1.sparse[c("i","j","val")], N = 1952)
plot_contact_map(mtx1)
# now make fold-change matrix
# get another sample contact map (MSC replicate 2) for chromosome 18
mtx2.sparse <- CopulaHiC::sample_hic_maps[["MSC-HindIII-2_40kb-raw"]][["18"]]
merged <- base::merge(mtx1.sparse, mtx2.sparse, by = c("i", "j"))
merged$fc <- merged$val.x / merged$val.y
fc.mtx <- sparse2dense(merged[c("i","j","fc")], N = 1952)
plot_contact_map(fc.mtx, fc = TRUE, colors.pal = c("blue","white","red"))
```

plot_copula_density *Plots bivariate 2D density plot (heatmap).*

Description

Plots bivariate 2D density plot (heatmap).

Usage

```
plot_copula_density(data, palette = "RdBu", direction = -1)
```

Arguments

data	data frame with columns U,V ~ Uniform(0,1)
palette	character, Rcolor palette, by default RdBu
direction	numeric 1 or -1 (revert color palette)

Value

ggplot object

See Also

CopulaHiC::copula_pvals for nice example of this function usage

Examples

```
# see ?CopulaHiC::copula_pvals
```

plot_diff_map *Plots differential map.*

Description

Draws differential map (i.e. one with positive and negative entries).

Usage

```
plot_diff_map(mtx.dense, zeros.na = TRUE, breaks = 10,
  colors.pal = c("blue", "white", "red"), color.range = NULL,
  sqrt.transform = FALSE, na.color = "black", neg.inf.color = "gold",
  pos.inf.color = "darkgreen", ...)
```

Arguments

mtx.dense	numeric matrix with positive and negative entries; if matrix does not contain any negative values use plot_contact_map function
zeros.na	logical if TRUE convert zero cells to NA
breaks	numeric number of breaks on color scale
colors.pal	colors for pallette
color.range	numeric vector of length 2 or NULL; if specified gives minimum and maximum values for color scale; this is manual adjustment of scale
sqrt.transform	logical if TRUE apply sqrt transformation: sqrt(pos) to positive elements of matrix and -sqrt(abs(neg)) to negative elements of matrix

See Also

CopulaHiC::iamge_plot_na for function handling heatmap plotting

Examples

```
# get sample contact map (MSC replicate 1) for chromosome 18
mtx1.sparse <- CopulaHiC::sample_hic_maps[["MSC-HindIII-1_40kb-raw"]][["18"]]
# get another sample contact map (MSC replicate 2) for chromosome 18
mtx2.sparse <- CopulaHiC::sample_hic_maps[["MSC-HindIII-2_40kb-raw"]][["18"]]
# make differential map
merged <- base::merge(mtx1.sparse, mtx2.sparse, by = c("i", "j"))
merged$difference <- merged$val.x - merged$val.y
dense <- sparse2dense(merged[c("i","j","difference")], N = 1952)
# plot
plot_diff_map(dense)
# plot with sqrt transformation of data
plot_diff_map(dense, sqrt.transform = TRUE)
```

---

plot_pc_vector *Plots A/B compartment vector.*

Description

Plots A/B compartment vector.

Usage

```
plot_pc_vector(pc, colors = c("blue", "red"), ...)
```

Arguments

pc	numeric vector

See Also

do_pca on A/B compartments and how to determine them, HiCcomparator object on real data example of compartments and plotting them (examples section)

Examples

```
# for real data example check ?HiCcomparator
# below is with simulated data
v <- sin(seq(1, 10, length.out = 100))
plot_pc_vector(v)
```

`plot_regions` *Plots regions (TADs or Long Range interactions) on contact map.*

Description

Plots regions (TADs or Long Range interactions) on contact map.

Usage

```
plot_regions(regions, pal.colors = NULL)
```

Arguments

`regions`	data frame or matrix with 2 (+1) or 4 (+1) columns; 2 columns format is for tads - first column is start bin, second column is end bin; 4 columns format is for LR interactions (rectangle like regions) - columns 1,2 are start and end bin of first interacting region, columns 3,4 are start and end bin of second interacting region; Last column (optional) is category column - like effect column with depletion, no.change, enrichment values to color TADs or LR interactions accordingly
`pal.colors`	character vector of colors if additional category column is specified - how to color categories

See Also

plot_contact_map, plot_diff_map for plotting contact maps or differential_interactions for determining and plotting p-value map with differential interactions

Examples

```
# plot contact map or differential map - see ?plot_diff_map
plot_diff_map(dense, sqrt.transform = TRUE)
# then get tads and plot them
tads <- CopulaHiC::sample_tads[["MSC-HindIII-1_40kb-raw"]]
tads18 <- tads[tads$name == "18",]
plot_regions(tads18[c("start","end")])
# for plotting differential interactions see ?differential_interactions
```

plot_with_inset *Plots contact map or diff map with inset.*

Description

Additionally it can plot regions on both orignal image and inset.

Usage

```
plot_with_inset(args.map, xlim, ylim, which.map = c("contact.map",
  "diff.map")[1], args.regions = NULL)
```

Arguments

args.map	named list of arguments for map plotting function for type of args see plot_contact_map and plot_diff_map
xlim	numeric 2-element vector o x limits
ylim	numeric 2-element vector o y limits
which.map	character string indicating if contact map or diff map should be plotted
args.regions	named list of arguments passed to plot_regions function, if NULL then don not plot regions

See Also

plot_contact_map, plot_diff_map for plotting contact maps and difference maps and plot_regions for plotting regions and its arguments

Examples

```
# get Hi-C map file
mtx.fname <- system.file("extdata", "MSC-HindIII-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
# read it and take chromosome 18
m.sparse18 <- read_npz(mtx.fname, mtx.names = c("18"))[["18"]]
dense <- sparse2dense(m.sparse18[c("i","j","val")], N = 1952)
plot_with_inset(list(dense), c(500,800), c(500,800))
# get TADs
tads <- map2tads(dense)
# plot with TADs
plot_with_inset(list(dense), c(500,800), c(500,800), args.regions = list(tads))
```

read_dense *Reads given npz file and returns list with dense matrices.*

Description

This function requires python and numpy.

Usage

```
read_dense(path, mtx.names = "all")
```

Arguments

path	character, string specifying path to npz file
mtx.names	character vector specyfing subset of matrices names to read from npz dict like file; by default all matrices are loaded

See Also

https://www.python.org/ for python, https://www.numpy.org/ for numpy

Examples

```
# get sample npz file name
mtx.fname <- system.file("extdata", "MSC-HindIII-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
mtx.dense.list <- read_dense(mtx.fname) # reads all chromosomes
# limiting number of chromosomes
mtx.dense.sublist <- read_dense(mtx.fname, mtx.names = c("18","19")) # only read chromosome 18 and 19
```

read_npz *Reads npz file with matrices and converts them sparse matrices.*

Description

Reads npz file with matrices and converts them sparse matrices.

Usage

```
read_npz(path, mtx.names = "all", sparse.format = TRUE)
```

Arguments

path	character, string specifying path to npz file
mtx.names	character vector specyfing subset of matrices names to read from npz dict like file; by default all matrices are loaded
sparse.format	logical if FALSE then this function is equivalent to read_dense function

Value

list with matrices in sparse format

See Also

dense2sparse for conversion of dense matrix to sparse format, read_dense on reading npz files

Examples

```
# get sample npz file name
mtx.fname <- system.file("extdata", "MSC-HindIII-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
mtx.sparse.list <- read_npz(mtx.fname) # reads all chromosomes
mtx.sparse.sublist <- read_npz(mtx.fname, mtx.names = c("18","19")) # only read chromosome 18 and 19
```

---

read_size *Reads dimension of every matrix in npz file.*

Description

Reads dimension of every matrix in npz file.

Usage

```
read_size(path, mtx.names = "all")
```

Arguments

path	character, string specifying path to npz file
mtx.names	character vector specyfing subset of matrices names to read from npz dict like file; by default all matrices are loaded

Value

matrix where row names are names of matrices from npz file and columns are rows and cols; each cell of the matrix contains number of rows and columns respectively that contact map with given name consits of

Examples

```
# get sample npz file name
mtx.fname <- system.file("extdata", "MSC-HindIII-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
sizes <- read_size(mtx.fname)
print(sizes)
```

remove_unmappable *Removes unmappable regions (all zeros columns and rows) from dense matrix.*

Description

Removes unmappable regions (all zeros columns and rows) from dense matrix.

Usage

```
remove_unmappable(dense.mtx)
```

Arguments

dense.mtx	numeric matrix (contact map) in dense format

Value

list containing 3 elements: indices of removed rows, indices of removed columns and matrix without unmappable regions

Examples

```
# construct matrix
mtx <- matrix(1:32, ncol = 4)
mtx[c(5,7,8),] <- 0
mtx[,c(2,4)] <- 0
l <- remove_unmappable(mtx)
print(l[["indices.rows"]])
print(l[["indices.cols"]])
print(l[["matrix"]])
```

restore_unmappable_mtx *Restores unmappable regions (all zeros columns and rows) in dense matrix.*

Description

Restores unmappable regions (all zeros columns and rows) in dense matrix.

Usage

```
restore_unmappable_mtx(dense.mtx.mappable, idx.rows, idx.cols = NULL,
  empty.elem = NA)
```

Arguments

dense.mtx.mappable	numeric matrix
idx.rows	integer vector, row indices of 0's elements in initial matrix (before rows removal) - the one which is to be restored
idx.cols	integer vector, column indices of 0's elements in initial matrix (before columns removal) - the one which is to be restored, by default it is equal to idx.rows
empty.elem	numeric or NA how to fill missing (restored) cells

Value

numeric matrix in dense format

Examples

```
# construct matrix with 0's row 5,7,8 and 0's columns 2,4
m <- matrix(1:32, ncol = 4); m[c(5,7,8),] <- 0; m[,c(2,4)] <- 0
l <- remove_unmappable(mtx)
restore_unmappable_mtx(l[["matrix"]], l[["indices.rows"]], idx.cols = l[["indices.cols"]])
restore_unmappable_mtx(l[["matrix"]], l[["indices.rows"]], idx.cols = l[["indices.cols"]], empty.elem = 0)
```

---

restore_unmappable_vec

Restores deleted regions (cells) in vector, for example unmappable regions in pc vector.

Description

Restores deleted regions (cells) in vector, for example unmappable regions in pc vector.

Usage

```
restore_unmappable_vec(vec, idx, empty.elem = NA)
```

Arguments

vec	numeric vector
idx	indices of elements in initial vector (before cells removal) - the one which is to be restored
empty.elem	numeric or NA how to fill missing (restored) cells

Examples

```
# create vector with zeros
v <- c(1,2,3,0,0,0,2,2,0,9,8,0)
# get indices of 0 elements
idx <- which(v == 0)
v.without <- v[-idx]
restore_unmappable_vec(v.without, idx)
restore_unmappable_vec(v.without, idx, empty.elem = 0)
```

---

right.min *Find first local minimum.*

Description

Seeks first local minimum starting from the left hand side of given vector and moving towards right hand side.

Usage

```
right.min(v)
```

Arguments

v	numeric vector

Value

numeric value of first local minimum in v starting from the begining of v and going right; NA if v is nondecreasing starting at first element of v and going right (i.e. v is either constant or there is maximum first)

See Also

CopulaHiC::local.min, CopulaHiC::local.max for finding local minma and maxima in vector v

Examples

```
# maximum first (in 7, index 3)
v1 <- c(5,6,7,6,4,3,2,1)
# no maximum or minimum
v2 <- c(2,2,2,2,2,2)
# minimum at 2 (index 3) and no maximum starting at 6 (index 7)
v3 <- c(4,3,2,3,4,5,6)
print(left.max(v1))
print(left.max(v2))
print(left.max(v3))
```

sample_hic_maps *Sample Hi-C contact maps dataset.*

Description

The list contains 4 entries - Hi-C datasets: IMR90-MboI-1_40kb-raw, IMR90-MboI-2_40kb-raw, MSC-HindIII-1_40kb-raw, MSC-HindIII-2_40kb-raw in 40kb resolution. This data was taken from 2 studies: Rao et al. 2014 "A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping." (IMR90 datasetss) and Dixon 2015 et al. "Chromatin architecture reorganization during stem cell differentiation" (MSC datasets). Data was processed using ICE pipeline https://mirnylab.bitbucket.io/hiclib/index.html without iterative correction. Each entry in list is a list with Hi-C maps in sparse format (data frames). The data is truncated - it only contains chromosomes 18 and 19.

Usage

```
data(sample_hic_maps)
```

Format

list of length 4 with Hi-C contact maps of 2 cell lines, each in 2 replicates. Each entry of list is a 2 element list with data frames. Every data frame is Hi-C contact map of single chromosome in sparse format:

i row (y) coordinate

j column (x) coordinate

val number of contact between i-th and j-th region

diagonal distance between i and j, i.e. abs(i-j) and diagonal of Hi-C contact map which this cell belongs

name Hi-C contact map name, chromosome in this case

sample_tads *Sample TAD dataset.*

Description

The list contains a set of TADs determined on sampleHiCmaps dataset using Insulation Score with window size 1Mbp and delta window size 200Kbp

Usage

```
data(sample_tads)
```

Format

list of length 4 with TADs of 2 cell lines, each in 2 replicates. Each entry in list is a data frame with TADs of chromsome 18 and 19:

start start bin of a TAD (first TAD starts at 0)

end end bin of a TAD (for consecutive TADs it is equal to next TAD start bin)

name name of Hi-C contact map on which this TADs were determined - chromosome in this case

---

save_npz *Saves list with dense matrices to npz compressed file.*

Description

Saves list with dense matrices to npz compressed file.

Usage

```
save_npz(mtx.list, path)
```

Arguments

`mtx.list`	list containg dense matrices, list should have names
`path`	character string specifying path together with filename to save matrices

Value

None

See Also

https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.savez_compressed.html for python numpy method used to save matrices in npz compressed file

Examples

```
# get sample npz file name
mtx.fname <- system.file("extdata", "MSC-HindIII-1_40kb-raw.npz", package = "CopulaHiC", mustWork = TRUE)
mtx.dense.sublist <- read_dense(mtx.fname, mtx.names = c("18","19"))
print(names(mtx.dense.sublist))
print(typeof(mtx.dense.sublist))
print(typeof(mtx.dense.sublist[["18"]]))
print(dim(mtx.dense.sublist[["18"]]))
# save
save_npz(mtx.dense.sublist, "example.npz")
```

`sparse2dense` *Converts sparse matrix to dense matrix.*

Description

This function handles can distinguish between symmetric and non symmetric matrices when given sparse matrix can be interpreted as square.

Usage

```
sparse2dense(sparse.as.df, N = NULL, balancing = TRUE,
  missing.cells.na = FALSE)
```

Arguments

`sparse.as.df`	data.frame with sparse matrix containing 3 columns: i, j, val
`N`	positive integer, optional desired dimension of dense matrix enforced with `balancing` function
`balancing`	logical, if perform balancing; see `balancing` description for information on function behaviour when N is NULL
`missing.cells.na`	logical, if TRUE then substitute cells with i,j coordinates missing in given sparse matrix with NA, otherwise substitute with 0

See Also

`balancing` for balancing

Examples

```
# produce dense matrix
mtx.dense <- matrix(1:24, ncol = 3)
# add some zeros
mtx.dense <- rbind(matrix.dense, c(0,100,0))
# construct sparse matrix
mtx.sparse <- dense2sparse(mtx.dense, add.diagonal = FALSE)
# get back dense matrix
sparse2dense(mtx.sparse, balancing = FALSE)
sparse2dense(mtx.sparse, balancing = FALSE, missing.cells.na = TRUE)
# symmetric matrices
mtx.sym <- matrix(1:16, ncol = 4)
mtx.sym <- mtx.sym + t(mtx.sym)
sparse2dense(dense2sparse(mtx.sym, add.diagonal = FALSE))
# square non symmetric matrices
mtx.sq <- matrix(1:16, ncol = 4)
mtx.sq[c(2,3,4,7,12)] <- 0
sparse2dense(mtx.sparse)
sparse2dense(mtx.sparse, missing.cells.na = TRUE)
```

sparse2interactions *Convert sparse contact map to atomic interactions.*

Description

Converts contact map in sparse data frame format into data frame with interaction, where one row is single interaction. This is usefull for bootstraping Hi-C interactions.

Usage

```
sparse2interactions(sparse.mtx)
```

Arguments

sparse.mtx	data.frame Hi-C contact map in sparse format with mandatory columns i, j, val

Value

data.frame with 2 columns, which are i-th (x) and j-th (y) coordinates of every interaction.

See Also

dense2sparse

Examples

```
# create data frame with artificial interactions, where val
# indicates total number of interactions between bins i and j
sparse.mtx <- data.frame(i = c(1,3,5,7,8,9,11), j = c(7,2,1,1,3,10,9), val = c(10,8,3,1,1,2,20))
print(sparse.mtx)
sparse2interactions(sparse.mtx)
```

superdiag *Efficiently slices k-th diagonal from matrix A.*

Description

Efficiently slices k-th diagonal from matrix A.

Usage

```
superdiag(A, k, return.idx = FALSE)
```

Arguments

A	matrix
k	integer diagonal to be sliced, 1 is main diagonal, positive k will yield diagonals from lower triangular matrix while negative k will yield diagonals from upper triangular
return.idx	logical whether to return indices of this diagonal

Value

numeric, vector containing entries on k-th diagonal of matrix A

Examples

```
mtx <- matrix(1:25, ncol = 5)
superdiag(mtx, k = 1)
superdiag(mtx, k = 2)
superdiag(mtx, k = -2)
```

Index