Package ‘shaman’

October 18, 2026

Type Package

Title Sampling Hi-C Contact Matrices for Aparametric Normalization

Version 2.1.1

Description Generate expected Hi-C (genome-wide chromosome conformation capture) data maintaining marginal coverage, and the decay in contact probability with increasing genomic distance. Expected Hi-C data is used for normalization and graphical presentation of statistical significance of observed Hi-C contact enrichments. The method is described in Mendelson Cohen et al. (2017) <doi:10.1101/187203>.

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

LinkingTo Rcpp

SystemRequirements C++17

Suggests knitr, testthat, rmarkdown, Gviz, foreach

VignetteBuilder rmarkdown, knitr

Depends R (>= 4.0.0)

Imports misha (>= 3.5.6), Rcpp (>= 0.12.11), RANN, data.table, ggplot2 (>= 3.4.0), reshape2, grid, plyr, parallel

URL https://tanaylab.github.io/shaman/, https://github.com/tanaylab/shaman

OS_type unix

BugReports https://github.com/tanaylab/shaman/issues

NeedsCompilation yes

Author Amos Tanay [aut],
Netta Mendelson Cohen [aut],
Aviezer Lifshitz [aut, cre],
Weizmann Institute of Science [cph]

Maintainer Aviezer Lifshitz <aviezer.lifshitz@weizmann.ac.il>

Repository CRAN

Date/Publication 2026-10-18 11:00:02 UTC

Contents

ctcf_forward . . . 2
ctcf_reverse . . . 3
get_param . . . 3
get_param_list . . . 4
init_params . . . 4
shaman_generate_feature_grid . . . 5
shaman_get_test_track_db . . . 6
shaman_gplot_map . . . 7
shaman_gplot_map_score . . . 8
shaman_kk_norm . . . 9
shaman_plot_feature_grid . . . 10
shaman_plot_map_score_with_annotations . . . 11
shaman_plot_tracks_and_annotations . . . 13
shaman_score_hic_mat . . . 15
shaman_score_hic_mat_for_track . . . 16
shaman_score_hic_points . . . 18
shaman_score_hic_track . . . 20
shaman_score_pal . . . 22
shaman_shuffle_and_score_hic_mat . . . 22
shaman_shuffle_hic_mat_for_track . . . 24
shaman_shuffle_hic_track . . . 26

Index **29**

`ctcf_forward` *Human CTCF motifs found on forward strand on hg19 genome*

Description

A dataset containing the chromosomal coordinates of all CTCF forward strand motifs including a chip-seq peak and not overlapping with a TSS and a reverse strand CTCF motif.

Usage

```
ctcf_forward
```

Format

An object of class `data.frame` with 8791 rows and 3 columns.

ctcf_reverse — *Human CTCF motifs found on reverse strand on hg19 genome*

Description

A dataset containing the chromosomal coordinates of all CTCF reverse strand motifs including a chip-seq peak and not overlapping with a TSS and a forward strand CTCF motif.

Usage

```
ctcf_reverse
```

Format

An object of class `data.frame` with 8842 rows and 3 columns.

get_param — *Get params from saved var*

Description

Get params from saved var

Usage

```
get_param(nm, params)
```

Arguments

`nm`	Parameter name.
`params`	Named list of parameters.

Value

The parameter, or NA if it is missing.

get_param_list *Get params from saved var*

Description

Get params from saved var

Usage

```
get_param_list(nm, params)
```

Arguments

nm	Parameter name.
params	Named list of parameters.

Value

The comma-separated parameter split into a list, or NA if it is missing.

init_params *Read params from files*

Description

Read params from files

Usage

```
init_params(fn, prev_params = NULL)
```

Arguments

fn	Parameter file, one `name=value` per line.
prev_params	Previously read parameters; names found in both are an error.

Value

A named list of the parameters.

shaman_generate_feature_grid

Calculate spatial enrichment of contacts between two features

Description

shaman_generate_feature_grid

Usage

```
shaman_generate_feature_grid(
  feature1,
  feature2,
  obs_track_nm,
  exp_track_nm = paste0(obs_track_nm, "_shuffle"),
  score_track_nm = paste0(obs_track_nm, "_score"),
  score_filter = 30,
  range = 25000,
  resolution = 500,
  min_dist = 200000,
  max_dist = 1000000
)
```

Arguments

feature1	1d intervals representing feature 1. Strand, if provided will be ignored.
feature2	1d intervals representing feature 2. Strand, if provided will be ignored.
obs_track_nm	Name of observed 2D genomic track for the hic data.
exp_track_nm	Name of expected 2D genomic track.
score_track_nm	Name of 2D score track. If track doesn't exist, no filter on grid will be applied.
score_filter	Minimum value of score within the expanded window surrounding the grid point for the window to be considered in the statistics.
range	Window size around the grid point.
resolution	Size of bins (in base pairs).
min_dist	Minimum distance between features on the grid.
max_dist	Maximum distance between features on the grid.

Details

Build a grid of all combinations of intervals from feature 1 and feature 2 that fall within a band defined by min_dist and max_dist. For each point on the grid, look at the surrounding window, defined by range parameter. Discard all windows that do not contain a point with a score (defined in score_track_nm) above the score_filter parameter. This allows for focusing on potentially enriched pairs. Dissect the window into small bins, size in base pairs defined by the resolution parameter,

and count the number of observed contacts, and the number of expected contacts in each bin. All windows are then summed together, generating a single matrix of observed and expected contacts, which is returned by function. Note that grid contains only points in which feature 1 position is smaller than feature 2 position.

Value

list containing obs - observed matrix, sum of the number of observed contacts in each bin in the window, exp - expected matrix, sum of the number of expected contacts in each bin in the window, grid_size - number of windows included in statistics

Examples

```
# Set misha db to test
library(misha)
gsetroot(shaman_get_test_track_db())
# the test db has chr2 only, with contacts in chr2:176.5e06-177e06
grid <- shaman_generate_feature_grid(
   ctcf_forward[ctcf_forward$chrom == "chr2", ], ctcf_reverse[ctcf_reverse$chrom == "chr2", ],
    "hic_obs",
    exp_track_nm = "hic_exp", min_dist = 1e05, max_dist = 5e05
)
plot <- shaman_plot_feature_grid(list(grid), 25000, 500, 1000)
```

---

shaman_get_test_track_db

Returns the example misha database

Description

shaman_get_test_track_db Returns the path of an example misha database with Hi-C contacts from the ELA K562 dataset.

Usage

```
shaman_get_test_track_db(full = FALSE)
```

Arguments

full	Whether to return the full example database (downloaded on first use) instead of the small one.

Details

By default this is a small database that is built in the session's temporary directory on first use; later calls in the session reuse it. It has one chromosome (hg19 chr2) and the observed (hic_obs), expected (hic_exp) and score (hic_score) tracks of chr2:176.5e06-177e06, around the hoxd locus (46,735 observed contacts, each stored in both orientations).

With `full = TRUE` it returns the full example database of earlier shaman versions: 4.6 million contacts from the ELA K562 dataset covering the hoxd locus (chr2:175e06-178e06) and convergent CTCF regions, with scores for chr2:175e06-178e06. It is downloaded (about 100MB, from the lab's public S3 bucket) and extracted into the user cache directory (`tools::R_user_dir("shaman", "cache")`, about 480MB) on first use; later calls reuse it. Processing the complete matrix from this study requires downloading the full contact list and regenerating the reshuffled matrix.

Value

The path of the database.

Examples

```
library(misha)
gsetroot(shaman_get_test_track_db())
gtrack.ls()
```

`shaman_gplot_map` *Plot an observed or expected hic map*

Description

`shaman_gplot_map`

Usage

```
shaman_gplot_map(
  points,
  interval_range = NA,
  rotate = TRUE,
  point_size = 0.1,
  add_axis = TRUE
)
```

Arguments

`points`	A dataframe containing the points (start1, start2).
`interval_range`	Data frame with the start and end of the range to plot. If NA, the range of start1 in points.
`rotate`	Binary flag, indicating if the plot should be rotated by 45 degrees.
`point_size`	Cex size of the points in the plot.
`add_axis`	Binary flag, indicating if axis should be added to plot.

Details

Plots hic contact matrix.

Value

A ggplot object with the map.

Examples

```
# Set misha db to test
library(misha)
gsetroot(shaman_get_test_track_db())
points <- gextract("hic_obs", gintervals.2d(2, 176.5e06, 177e06, 2, 176.5e06, 177e06))
shaman_gplot_map(points)
```

---

shaman_gplot_map_score

Plot a normalized hic map

Description

shaman_gplot_map_score

Usage

```
shaman_gplot_map_score(
  points_score,
  interval_range = NA,
  rotate = TRUE,
  point_size = 0.1,
  add_axis = TRUE
)
```

Arguments

points_score	A dataframe containing the points (start1, start2) and their normalized score.
interval_range	Data frame with the start and end of the range to plot. If NA, the range of start1 in points_score.
rotate	Binary flag, indicating if the plot should be rotated by 45 degrees.
point_size	Cex size of the points in the plot.
add_axis	Binary flag, indicating if axis should be added to plot.

Details

Plots observed hic contact matrix color-coded by normalized scores. Data can be either extracted directly from score track or computed via the functions: shaman_score_hic_mat(), shaman_shuffle_and_score_hic_mat()

Value

A ggplot object with the map.

Examples

```
# Set misha db to test
library(misha)
gsetroot(shaman_get_test_track_db())
points <- gextract("hic_score", gintervals.2d(2, 176.5e06, 177e06, 2, 176.5e06, 177e06),
    colnames = "score"
)
shaman_gplot_map_score(points)
```

---

shaman_kk_norm *Compute a score matrix for observed data based on the expected for a given set of points*

Description

shaman_kk_norm

Usage

```
shaman_kk_norm(obs, exp, points, k = 100, k_exp = 100)
```

Arguments

obs	Dataframe containing the observed points
exp	Dataframe containing the expected (shuffled) points.
points	A score will be computed for each of the points.
k	The number of neighbor distances used for the score on observed data. For higher resolution maps, increase k. For lower resolution maps, decrease k.
k_exp	The number of neighbor distances used for the score on expected data. Should reflect the ratio between the total number of observed and expected over the entire chromosome.

Details

This function receives observed and expected data and computes the score on a given set of points. The score for a point is the KS D-statistic of the distances to its k nearest neighbors in the observed data compared to the expected data.

Value

NULL if insufficient observed data, otherwise returns a list containing 3 elements: 1) points - start1, start2 and score for all observed points. 2) obs - the observed points. 3) exp - the expected points.

Examples

```
# Set misha db to test
library(misha)
gsetroot(shaman_get_test_track_db())
focus <- gintervals.2d(2, 176.6e06, 176.9e06, 2, 176.6e06, 176.9e06)
regional <- gintervals.2d(2, 176.5e06, 177e06, 2, 176.5e06, 177e06)
points <- gextract("hic_obs", focus, band = c(-5e05, -1024))
obs <- gextract("hic_obs", regional, band = c(-5e05, -1024))
exp <- gextract("hic_exp", regional, band = c(-5e05, -1024))
mat_score <- shaman_kk_norm(obs, exp, points, k = 100, k_exp = 200)
shaman_gplot_map_score(mat_score$points)
```

---

shaman_plot_feature_grid

Plot spatial enrichment of contacts between two features

Description

shaman_plot_feature_grid

Usage

```
shaman_plot_feature_grid(
  grids,
  range,
  grid_resolution,
  plot_resolution,
  type = "enrichment",
  fig_fn = "",
  fig_width = 400,
  fig_height = 400,
  pal = colorRampPalette(c("blue", "#87FFFF", "white", "#FF413D", "black")),
  zlim = c(-1, 1)
)
```

Arguments

grids	List of grids generated for the same two feature sets from replicate observed / expected hic tracks.
range	Window size around the grid point that was used to generate the grid.
grid_resolution	Size of bins that was used to generate the grid.
plot_resolution	Size of bins to display.
type	The type of data to plot, can be "observed" for log10(obs/sum(obs)), "expected" for log10(exp/sum(exp)) or "enrichment" for log2(obs/sum(obs) / exp/sum(exp))

fig_fn	Name of png file to output to. Empty string will cause the figure to be plotted to the current device.
fig_width	Width in pixels of output png figure.
fig_height	Height in pixels of output png figure.
pal	Color palette to use for image
zlim	The minimum and maximum values for which colors should be plotted. Suggested zlim values by type: enrichment: (-1,1), obs,exp: (-4.5, -3)

Details

After contact feature grids are generated via shaman_generate_feature_grid, use this function to visualize them.

Value

A list with the observed (obs) and expected (exp) matrices summed over the grids, at plot_resolution.

Examples

```
# Set misha db to test
library(misha)
gsetroot(shaman_get_test_track_db())
# the test db has chr2 only, with contacts in chr2:176.5e06-177e06
grid <- shaman_generate_feature_grid(
   ctcf_forward[ctcf_forward$chrom == "chr2", ], ctcf_reverse[ctcf_reverse$chrom == "chr2", ],
     "hic_obs",
   exp_track_nm = "hic_exp", score_track_nm = "hic_score", min_dist = 1e05, max_dist = 5e05
)
shaman_plot_feature_grid(list(grid), 25000, 500, 500)
shaman_plot_feature_grid(list(grid), 25000, 500, 1000)
shaman_plot_feature_grid(list(grid), 25000, 500, 2000)
```

---

shaman_plot_map_score_with_annotations

Plot a normalized hic map with annotations

Description

shaman_plot_map_score_with_annotations

Usage

```
shaman_plot_map_score_with_annotations(
  genome,
  points_score,
  interval_range,
  point_size = 0.1,
```

```
  misha_tracks = list(),
  mt_colors = getOption("shaman.track_colors"),
  mt_ylims = NULL,
  annotations = list(),
  a_colors = getOption("shaman.annotation_colors"),
  add_genes = TRUE,
  add_ideogram = TRUE,
  add_axis = TRUE,
  gene_stacking = "squish",
  gene_size = 0.7,
  track_size = 0.8,
  annotation_size = 0.7,
  fig_fn = "",
  fig_width = 900,
  fig_height = 5/6 * 900
)
```

Arguments

genome	Name of reference genome (e.g. "hg19", "mm10")
points_score	A dataframe containing the points (start1, start2) and their normalized score.
interval_range	1D interval (chrom, start, end) specifying the region to plot
point_size	Cex size of the points in the plot.
misha_tracks	List of 1D track expressions (virtual tracks also supported) which can be extracted from.
mt_colors	Array of colors, one for each misha_track, which will be used to plot each 1d track.
mt_ylims	Y-axis limits for each misha track. If not provided, generated automatically based on the data to be displayed in the region.
annotations	List of gintervals highlighting annotated regions.
a_colors	Array of colors, one for each annotation set.
add_genes	Boolean flag indicating whether to show gene annotations.
add_ideogram	Boolean flag indicating whether to add a chromosome ideogram.
add_axis	Boolean flag indicating whether to show the chromosomal axis.
gene_stacking	Describes the viewing option of the genes track. Can be either "squish", or "dense".
gene_size	Size of gene annotation view.
track_size	Size of track view.
annotation_size	Size of annotation view.
fig_fn	Name of png file to output to. Empty string will cause the figure to be plotted to the current device.
fig_width	Width in pixels of output png figure.
fig_height	Height in pixels of output png figure.

Details

Plots observed hic contact matrix color-coded by normalized scores, aligned with linear data such as gene annotations, rna, chromatin, transcription factor binding, etc.).

Value

No return value, called for the plot. Needs the Bioconductor package Gviz.

Examples

```
library(misha)
gsetroot(shaman_get_test_track_db())
points <- gextract("hic_score", gintervals.2d(2, 176.5e06, 177e06, 2, 176.5e06, 177e06),
    colnames = "score"
)

# loading Gviz takes a few seconds
if (requireNamespace("Gviz", quietly = TRUE)) {
   shaman_plot_map_score_with_annotations("hg19", points, gintervals(2, 176.5e06, 177e06),
        add_genes = FALSE, add_ideogram = FALSE
    )
}

# gene annotations and the ideogram are downloaded from UCSC
if (interactive() && requireNamespace("Gviz", quietly = TRUE)) {
   shaman_plot_map_score_with_annotations("hg19", points, gintervals(2, 176.5e06, 177e06))
}
```

---

shaman_plot_tracks_and_annotations

Plot misha 1D tracks and annotations

Description

shaman_plot_tracks_and_annotations

Usage

```
shaman_plot_tracks_and_annotations(
  genome,
  interval_range,
  misha_tracks = list(),
  mt_colors = getOption("shaman.track_colors"),
  mt_ylims = NULL,
  annotations = list(),
  a_colors = getOption("shaman.annotation_colors"),
  add_genes = TRUE,
  add_ideogram = TRUE,
```

```
  add_axis = TRUE,
  gene_stacking = "squish",
  gene_size = 0.7,
  track_size = 0.8,
  annotation_size = 0.7
)
```

Arguments

genome	Name of reference genome (e.g. "hg19", "mm10")
interval_range	1D interval (chrom, start, end) specifying the region to plot
misha_tracks	List of 1D track expressions (virtual tracks also supported) which can be extracted from.
mt_colors	Array of colors, one for each misha_track, which will be used to plot each 1d track.
mt_ylims	Y-axis limits for each misha track. If not provided, generated automatically based on the data to be displayed in the region.
annotations	List of gintervals highlighting annotated regions.
a_colors	Array of colors, one for each annotation set.
add_genes	Boolean flag indicating whether to show gene annotations.
add_ideogram	Boolean flag indicating whether to add a chromosome ideogram.
add_axis	Boolean flag indicating whether to show the chromosomal axis.
gene_stacking	Describes the viewing option of the genes track. Can be either "squish", or "dense"
gene_size	Size of gene annotation view
track_size	Size of track view
annotation_size	Size of annotation view

Details

Plots 1D tracks in UCSC format. Along with gene annotations and chromosomal axis, this function plots one dimensional misha tracks (e.g. rna-seq, chip-seq data) and marks annotated intervals in the plotting region.

Value

Called for the plot. Returns what `Gviz::plotTracks()` returns, invisibly. Needs the Bioconductor package Gviz.

Examples

```
library(misha)
gsetroot(shaman_get_test_track_db())
# the test db has chr2 only
ctcf <- list(
```

```
    ctcf_forward[ctcf_forward$chrom == "chr2", ],
    ctcf_reverse[ctcf_reverse$chrom == "chr2", ]
)

# loading Gviz takes a few seconds
if (requireNamespace("Gviz", quietly = TRUE)) {
    shaman_plot_tracks_and_annotations("hg19", gintervals(2, 176.5e06, 177e06),
        annotations = ctcf, add_genes = FALSE, add_ideogram = FALSE
    )
}

# gene annotations and the ideogram are downloaded from UCSC
if (interactive() && requireNamespace("Gviz", quietly = TRUE)) {
   shaman_plot_tracks_and_annotations("hg19", gintervals(2, 176.5e06, 177e06), annotations = ctcf)
}
```

`shaman_score_hic_mat` *Generate a score matrix for observed data based on the expected for a given 2D focus interval*

Description

`shaman_score_hic_mat`

Usage

```
shaman_score_hic_mat(
  obs_track_nms,
  exp_track_nms,
  focus_interval,
  regional_interval,
  points_track_nms = obs_track_nms,
  min_dist = 1024,
  k = 100,
  k_exp = 2 * k
)
```

Arguments

`obs_track_nms`	Names of observed 2D genomic tracks for the hic data. Pooling of multiple observed tracks is supported.
`exp_track_nms`	Names of expected (shuffled) 2D genomic tracks. Pooling of multiple expected tracks is supported.
`focus_interval`	2D interval on which to compute the scores.
`regional_interval`	An expansion of the focus interval, including points outside the focus matrix for accurate computing of the score. Note that for each observed point, its k-nearest neighbors must be included in the expanded matrix.

points_track_nms
: Names of 2D genomic tracks that contain points on which to compute normalized score. Pooling points from multiple tracks is supported.

min_dist
: The minimum distance between points.

k
: The number of neighbor distances used for the score. For higher resolution maps, increase k. For lower resolution maps, decrease k.

k_exp
: The number of neighbor distances used for the score on the expected tracks. Note, that when comparing expected generated by shuffling the observed, k_exp should be 2*k as the number of contacts in the expected track will always be twice the observed. However, if comparing between two datasets that are independent, k_exp should be set to NA and the will be determined by the ratio between the total number of contacts in this region.

Details

This function extracts observed data and expected data in an expanded matrix and computes the score for each observed point. The score for a point is the KS D-statistic of the distances to its k nearest neighbors in the observed data compared to the expected data.

Value

NULL if insufficient observed data, otherwise a list with one element: points - the points in the focus interval (chrom1, start1, end1, chrom2, start2, end2) and their score.

Examples

```
# Set misha db to test
library(misha)
gsetroot(shaman_get_test_track_db())
mat_score <- shaman_score_hic_mat(
    obs_track_nms = "hic_obs", exp_track_nms = "hic_exp",
    focus_interval = gintervals.2d(2, 176.7e06, 176.8e06, 2, 176.7e06, 176.8e06),
    regional_interval = gintervals.2d(2, 176.5e06, 177e06, 2, 176.5e06, 177e06)
)
shaman_gplot_map_score(mat_score$points)
```

---

shaman_score_hic_mat_for_track

Generate a score matrix for observed data based on the expected

---

Description

shaman_score_hic_mat_for_track

Usage

```
shaman_score_hic_mat_for_track(
  track_db,
  work_dir,
  obs_track_nms,
  exp_track_nms,
  points_track_nms,
  chrom,
  start1,
  end1,
  start2,
  end2,
  expand = 2000000,
  k = 100,
  min_dist = 1024
)
```

Arguments

`track_db`	Directory of the misha database.
`work_dir`	Centralized directory to store temporary files.
`obs_track_nms`	Names of observed 2D genomic tracks for the hic data. Pooling of multiple observed tracks is supported.
`exp_track_nms`	Names of expected (shuffled) 2D genomic tracks. Pooling of multiple expected tracks is supported.
`points_track_nms`	Names of 2D genomic tracks that contain points on which to compute normalized score. Pooling points from multiple tracks is supported.
`chrom`	The chromosome of the matrix.
`start1`	The start coordinate of the first dimension.
`end1`	The end coordinate of the first dimension.
`start2`	The start coordinate of the second dimension.
`end2`	The end coordinate of the second dimension.
`expand`	Size of expansion, points to include outside the matrix for accurate computing of the score. Note that for each observed point, its k-nearest neighbors must be included in the expanded matrix.
`k`	The number of neighbor distances used for the score. For higher resolution maps, increase k. For lower resolution maps, decrease k.
`min_dist`	The minimum distance between points.

Details

This function extracts observed data and expected data in an expanded matrix and computes the score for each observed point. The score for a point is the KS D-statistic of the distances to its k nearest neighbors in the observed data compared to the expected data.

chrom, start1, end1, start2 and end2 can be vectors, to score several matrices in one call. Matrices with the same chrom, start1 and end1 then share one extraction of each track over the union of their expanded intervals, instead of one extraction per matrix (each extraction of a 2D track reads the whole chromosome pair). The output files are the same as scoring the matrices one by one.

`options(shaman.score.threads = N)` computes the kNN distances and scores on N threads (default 1).

Value

0, 1 or -1 per matrix.

Examples

```
# The example below runs on the test misha db provided with shaman.
# Called by shaman_score_hic_track() for each matrix of its grid; here it scores the
# observed contacts of chr2:176.7e06-176.8e06.
library(misha)
track_db <- shaman_get_test_track_db()
gsetroot(track_db)
work_dir <- tempfile("shaman_score_")
dir.create(work_dir)
shaman_score_hic_mat_for_track(track_db, work_dir,
    obs_track_nms = "hic_obs", exp_track_nms = "hic_exp", points_track_nms = "hic_obs",
    chrom = "chr2", start1 = 176.7e06, end1 = 176.8e06, start2 = 176.7e06, end2 = 176.8e06
)
# the score of each observed contact (start1 <= start2) in the matrix
scores <- read.delim(list.files(work_dir, full.names = TRUE))
head(scores)
unlink(work_dir, recursive = TRUE)
```

---

shaman_score_hic_points

Generate a score matrix for observed data based on the expected for a given set of points

---

Description

`shaman_score_hic_points`

Usage

```
shaman_score_hic_points(
  obs_track_nms,
  exp_track_nms,
  points,
  regional_interval,
  min_dist = 1024,
  k = 100,
  k_exp = 2 * k
)
```

Arguments

- `obs_track_nms` Names of observed 2D genomic tracks for the hic data. Pooling of multiple observed tracks is supported.
- `exp_track_nms` Names of expected (shuffled) 2D genomic tracks. Pooling of multiple expected tracks is supported.
- `points` A score will be computed for each of the points.
- `regional_interval` An expansion of the focus interval, including points outside the focus matrix for accurate computing of the score. Note that for each observed point, its k-nearest neighbors must be included in the expanded matrix.
- `min_dist` The minimum distance between points.
- `k` The number of neighbor distances used for the score. For higher resolution maps, increase k. For lower resolution maps, decrease k.
- `k_exp` The number of neighbor distances used for the score on the expected tracks (see `shaman_score_hic_mat`).

Details

This function extracts observed data and expected data in an expanded matrix and computes the score for each observed point. The score for a point is the KS D-statistic of the distances to its k nearest neighbors in the observed data compared to the expected data.

Value

NULL if insufficient observed data, otherwise a list with one element: points - the given points with their score (column score).

Examples

```
# Set misha db to test
library(misha)
gsetroot(shaman_get_test_track_db())
focus <- gintervals.2d(2, 176.7e06, 176.8e06, 2, 176.7e06, 176.8e06)
points <- gextract("hic_obs", focus, band = c(-5e05, -1024))
mat_score <- shaman_score_hic_points(
    obs_track_nms = "hic_obs", exp_track_nms = "hic_exp",
   points = points, regional_interval = gintervals.2d(2, 176.5e06, 177e06, 2, 176.5e06, 177e06)
)
shaman_gplot_map_score(mat_score$points)
```

shaman_score_hic_track

Generate a score hic track based on observed and expected (shuffled) hic data

Description

shaman_score_hic_track

Usage

```
shaman_score_hic_track(
  track_db,
  work_dir,
  score_track_nm,
  obs_track_nms,
  exp_track_nms = paste0(obs_track_nms, "_shuffle"),
  points_track_nms = obs_track_nms,
  near_cis = 5000000,
  expand = 2000000,
  k = 100,
  max_jobs = 100
)
```

Arguments

track_db	Directory of the misha database.
work_dir	Centralized directory to store temporary files.
score_track_nm	Score track that will be created.
obs_track_nms	Names of observed 2D genomic tracks for the hic data. Pooling of multiple observed tracks is supported.
exp_track_nms	Names of expected (shuffled) 2D genomic tracks. Pooling of multiple expected tracks is supported.
points_track_nms	Names of 2D genomic tracks that contain points on which to compute normalized score. Pooling points from multiple tracks is supported.
near_cis	Size of matrix in grid.
expand	Size of expansion, points to include outside the matrix for accurate computing of the score. Note that for each observed point, its k-nearest neighbors must be included in the expanded matrix.
k	The number of neighbor distances used for the score. For higher resolution maps, increase k. For lower resolution maps, decrease k.
max_jobs	Maximal number of qsub jobs.

Details

This function generates a 2D score track based on observed and expected hic data. The score is computed by generating a grid of small matrices spanning all chromosomes and computing the score of each matrix independently. The model for computing the score relies on the KS D statistic computed for each observed point, over the distances of the k-nearest neighbors in the observed compared to the expected. High scores represent contact enrichment while low scores depict insulation. Note that this function requires either sge (qsub) or multicore to compute in a timely manner. Parameters can be set via shaman.sge_support or shaman.mc_support in shaman.conf file. Sun Grid Engine mode, `options(shaman.sge_support = 1)`, is preferred; work_dir must then be accessible by all jobs. Multi-core mode is `options(shaman.mc_support = 1)`. `options(shaman.score.threads = N)` computes the kNN distances and scores of each matrix on N threads (default 1); in multi-core mode each of the max_jobs processes uses N threads, and in SGE mode each job does, so shaman.sge_flags should ask for N slots. Each extraction of a 2D track can keep about the whole chromosome's track file in memory; misha's `options(gtrack.num.chunks = 1000)` bounds that (2-3x lower peak memory on dense matrices in our runs) without changing the scores.

Each step creates temporary files of the matrix scores which are then joined to a track. Temporary files are deleted upon track creation. Matrices that still have no score after 3 rounds of jobs stop the run with an error that names them. The finished score files stay in work_dir, so a rerun with the same work_dir computes only the missing ones.

Value

No return value, called for side effects: creates the 2D score track score_track_nm in track_db.

Examples

```
# The example below runs on the test misha db provided with shaman.
# Note that this is a toy db sampled from K562 ela data -
# scoring based on the observed and expected tracks will not produce the score track,
# as most of the genome is missing.
library(misha)
track_db <- shaman_get_test_track_db()
gsetroot(track_db)
old_opts <- options(shaman.mc_support = 1) # configuring multi-core mode
if (gtrack.exists("hic_score_new")) {
    gtrack.rm("hic_score_new", force = TRUE)
    gdb.reload()
}
ret <- shaman_score_hic_track(track_db,
    # work_dir can be tempdir() only in multi-core mode.
    # For sge mode, work_dir must be accessible by all jobs.
    work_dir = tempdir(),
    score_track_nm = "hic_score_new",
    obs_track_nms = "hic_obs",
    exp_track_nms = "hic_exp",
   near_cis = 1e09, # this test db contains very little data, can increase the size of each job
    k = 20, # default is set to 100
    max_jobs = 2
) # increase number of jobs for optimal runtime when running in sge mode
gdb.reload()
```

```
gtrack.ls("hic_score_new") # new score track that was created
options(old_opts)
```

---

shaman_score_pal *Returns the color palette used to display normalized scores*

Description

shaman_score_pal

Usage

```
shaman_score_pal()
```

Value

A vector of 201 colors, for scores -100 to 100.

Examples

```
pal <- shaman_score_pal()
# the colors of the scores -100, 0, 50 and 100
pal[c(-100, 0, 50, 100) + 101]
```

---

shaman_shuffle_and_score_hic_mat *Inline function for generating an expected matrix and computing the score for a given interval*

Description

shaman_shuffle_and_score_hic_mat

Usage

```
shaman_shuffle_and_score_hic_mat(
  obs_track_nms,
  interval,
  work_dir,
  expand = 1000000,
  min_dist = 1024,
  k = 100,
  dist_resolution = NA,
  decay_smooth = NA,
  hic_mcmc_max_resolution = 400,
  shuffle = 80,
```

```
  grid_small = 500000,
  grid_high = 1000000,
  grid_increase = 500000,
  grid_step_iter = 40,
  seed = NULL
)
```

Arguments

obs_track_nms	Name of observed 2D genomic tracks for the hic data.
interval	2D interval on which to compute the scores.
work_dir	Centralized directory to store temporary files.
expand	Size of expansion, points to include outside the matrix for accurate computing of the score. Note that for each observed point, its k-nearest neighbors must be included in the expanded matrix.
min_dist	The minimum distance between points.
k	The number of neighbor distances used for the score. For higher resolution maps, increase k. For lower resolution maps, decrease k.
dist_resolution	Number of bins in each log2 distance unit. If NA, value is determined based on observed data (recommended).
decay_smooth	Number of bins to use for smoothing the MCMC target function: the decay curve. If NA, value is determined based on observed data (recommended).
hic_mcmc_max_resolution	Maximum number of bins for each log2 unit.
shuffle	Number of shuffling rounds for each observed point.
grid_small	Initial size of maximum distance between contact pairs considered for switching
grid_high	Final size of maximum distance between contact pairs considered for switching
grid_increase	Grid increase size
grid_step_iter	Number of iterations in each grid size
seed	Seed for the shuffler: NULL (default) seeds it from the current time (in seconds), or a whole number between 0 and 2^31 - 1. The shuffler prints the seed it uses.

Details

This function generates an expected 2D hic matrix based on observed hic data, and computes its score.

Value

NULL if insufficient observed data, otherwise returns a list containing: 1) points - start1, start2 and score for all observed points. 2) obs - the observed points. 3) exp - the expected points. 4) exp_fn - the name of the expected (shuffled) data file 5) seed - the seed the shuffler used

Examples

```
# Set misha db to test
library(misha)
gsetroot(shaman_get_test_track_db())
mat_score <- shaman_shuffle_and_score_hic_mat(
    obs_track_nms = "hic_obs",
    interval = gintervals.2d(2, 176.6e06, 176.9e06, 2, 176.6e06, 176.9e06),
    expand = 1e05,
    work_dir = tempdir(),
    shuffle = 2, # default is set to 80
    grid_step_iter = 1, # default is set to 40
    seed = 1
)
shaman_gplot_map_score(mat_score$points)
```

---

shaman_shuffle_hic_mat_for_track

Generate an expected matrix from observed data as a process for generating an expected track

---

Description

shaman_shuffle_hic_mat_for_track

Usage

```
shaman_shuffle_hic_mat_for_track(
  track_db,
  track,
  work_dir,
  chrom,
  start1,
  end1,
  start2,
  end2,
  min_dist = 1024,
  max_dist = max(gintervals.all()$end),
  dist_resolution = NA,
  decay_smooth = NA,
  proposal_iterations = 10000000,
  shuffle = 80,
  hic_mcmc_max_resolution = 400,
  raw_ext = "raw",
  shuffled_ext = "shuffled",
  grid_small = 500000,
  grid_high = 1000000,
  grid_increase = 500000,
```

```
  grid_step_iter = 40,
  sort_uniq = FALSE,
  seed = NULL
)
```

Arguments

track_db	Directory of the misha database.
track	Name of observed 2D genomic track for the hic data.
work_dir	Centralized directory to store temporary files.
chrom	The chromosome of the matrix.
start1	The start coordinate of the first dimension.
end1	The end coordinate of the first dimension.
start2	The start coordinate of the second dimension.
end2	The end coordinate of the second dimension.
min_dist	The minimum distance between contact end points.
max_dist	The maximum distance between contact end points.
dist_resolution	Number of bins in each log2 distance unit. If NA, value is determined based on observed data (recommended).
decay_smooth	Number of bins to use for smoothing the MCMC target function: the decay curve. If NA, value is determined based on observed data (recommended).
proposal_iterations	Number of MCMC sampling iterations between proposal corrections.
shuffle	Number of shuffling rounds for each observed point.
hic_mcmc_max_resolution	Maximum number of bins for each log2 unit
raw_ext	File extension of the observed data.
shuffled_ext	File extension of the shuffled data.
grid_small	Initial size of maximum distance between contact pairs considered for switching
grid_high	Final size of maximum distance between contact pairs considered for switching
grid_increase	Grid increase size
grid_step_iter	Number of iterations in each grid size
sort_uniq	Binary flag, indicating whether the shuffled matrix file should be sorted and contacts combined. This is required prior to importing the track to misha, and should be applied to full chromosomes only.
seed	Seed for the shuffler: NULL (default) seeds it from the current time (in seconds), or a whole number between 0 and 2^31 - 1. The shuffler prints the seed it uses.

Details

This function generates an expected 2D hic matrix from observed hic data. Should not be called externally, but rather as jobs when creating a 2D shuffled track. The observed data is a combination of observed contacts in scope plus already shuffled near-cis contacts (stored in work_dir) which we sample from to maintain the decay probability curve.

Each step creates temporary files of the shuffled matrices which are then joined to a track. Temporary files are deleted upon track creation.

Value

The seed the shuffler used, or NA if the matrix was not shuffled here (no or too few contacts, or the shuffled file already existed).

Examples

```
# The example below runs on the test misha db provided with shaman.
# Called by shaman_shuffle_hic_track() for each chromosome; here it shuffles the contacts
# of chr2:176.5e06-177e06.
library(misha)
track_db <- shaman_get_test_track_db()
work_dir <- tempfile("shaman_shuffle_")
dir.create(work_dir)
seed <- shaman_shuffle_hic_mat_for_track(track_db, "hic_obs", work_dir,
    chrom = "chr2", start1 = 176.5e06, end1 = 177e06, start2 = 176.5e06, end2 = 177e06,
    shuffle = 1, # default is set to 80
    grid_step_iter = 1, # default is set to 40
    seed = 1, # the same seed gives the same shuffle
    sort_uniq = TRUE
)
seed
# the shuffled contacts (in both orientations) and their counts
shuffled <- read.delim(list.files(work_dir, pattern = "uniq$", full.names = TRUE))
head(shuffled)
unlink(work_dir, recursive = TRUE)
```

---

shaman_shuffle_hic_track

Generate an expected hic track based on observed hic data

Description

shaman_shuffle_hic_track

Usage

```
shaman_shuffle_hic_track(
  track_db,
  obs_track_nm,
```

```
  work_dir,
  exp_track_nm = paste0(obs_track_nm, "_shuffle"),
  max_jobs = 25,
  shuffle = 80,
  grid_small = 500000,
  grid_high = 1000000,
  grid_step_iter = 40,
  dist_resolution = NA,
  smooth = NA,
  seed = NULL
)
```

Arguments

`track_db`	Directory of the misha database.
`obs_track_nm`	Name of observed 2D genomic track for the hic data.
`work_dir`	Centralized directory to store temporary files.
`exp_track_nm`	Name of expected 2D genomic track.
`max_jobs`	Maximal number of qsub or local jobs - for optimal performance provide the number of chromosomes.
`shuffle`	Average number of shuffling transitions for each observed point in the chromosomal contact matrix.
`grid_small`	Initial size of maximum distance between contact pairs considered for switching
`grid_high`	Final size of maximum distance between contact pairs considered for switching
`grid_step_iter`	Number of iterations in each grid size
`dist_resolution`	Number of bins in each log2 distance unit. If NA, value is determined based on observed data (recommended).
`smooth`	Number of bins to use for smoothing the MCMC target function: the decay curve. If NA, value is determined based on observed data (recommended).
`seed`	Seed for the shuffler: NULL (default) seeds each chromosome from the time (in seconds) its shuffle starts. With a whole number, chromosome i of `gintervals.all()` is shuffled with seed + i - 1 (modulo 2^31), so the same seed gives the same track. The seed each chromosome was shuffled with is stored in the track attribute `seed` (NA for chromosomes this call did not shuffle). A chromosome's seed depends on its position in `gintervals.all()`, so the same seed reproduces a track only in a database with the same chromosomes; the `seed` attribute reproduces each chromosome.

Details

This function generates an expected 2D hic track based on observed hic data. Each chromosome is shuffled separately, to generate an expected shuffled contact matrix. Note that this function requires sge (qsub) or multicore to be enabled. Parameter can be set via shaman.sge_support or shaman.mc_support in shaman.conf file. Sun Grid Engine mode, `options(shaman.sge_support =`

1), is preferred; work_dir must then be accessible by all jobs. Multi-core mode is `options(shaman.mc_support = 1)`.

Each step creates temporary files of the shuffled matrices which are then joined to a track. Temporary files are deleted upon track creation. If the shuffle of a chromosome fails (an error, or its process or job is killed), no track is imported and work_dir keeps the shuffled chromosomes, so a rerun with the same work_dir shuffles only the others.

Value

No return value, called for side effects: creates the 2D track exp_track_nm in track_db, with the seeds in its attribute `seed`.

Examples

```
# The example below runs on the test misha db provided with shaman.
# Note that this is a toy db sampled from K562 ela data - shuffling the observed track
# will not produce the expected track.
library(misha)
track_db <- shaman_get_test_track_db()
gsetroot(track_db)
old_opts <- options(shaman.mc_support = 1) # configuring multi-core mode
if (gtrack.exists("hic_obs_shuffle")) {
    gtrack.rm("hic_obs_shuffle", force = TRUE)
    gdb.reload()
}
ret <- shaman_shuffle_hic_track(track_db,
    obs_track_nm = "hic_obs",
    # work_dir can be tempdir() only in multi-core mode.
    # For sge mode, work_dir must be accessible by all jobs.
    work_dir = tempdir(),
    shuffle = 1, # default is set to 80
    grid_step_iter = 1, # default is set to 40
    max_jobs = 2, # optimally set to number of chromosomes
    seed = 1 # the same seed gives the same track
)
gdb.reload()
gtrack.ls("hic_obs_shuffle") # new shuffled track that was created
gtrack.attr.get("hic_obs_shuffle", "seed") # the seed of each chromosome
options(old_opts)
```

Index

∗ **datasets**
    ctcf_forward, 2
    ctcf_reverse, 3

ctcf_forward, 2
ctcf_reverse, 3

get_param, 3
get_param_list, 4

init_params, 4

shaman_generate_feature_grid, 5
shaman_get_test_track_db, 6
shaman_gplot_map, 7
shaman_gplot_map_score, 8
shaman_kk_norm, 9
shaman_plot_feature_grid, 10
shaman_plot_map_score_with_annotations, 11
shaman_plot_tracks_and_annotations, 13
shaman_score_hic_mat, 15
shaman_score_hic_mat_for_track, 16
shaman_score_hic_points, 18
shaman_score_hic_track, 20
shaman_score_pal, 22
shaman_shuffle_and_score_hic_mat, 22
shaman_shuffle_hic_mat_for_track, 24
shaman_shuffle_hic_track, 26