Package ‘scHOTTER’

September 2, 2025

Type Package

Title scHOTTER: Parametric testing for spatially heterogeneous co-expression in single cell data

Version 0.0.0.9000

Author Liam Skender, Shila Ghazanfar

Maintainer Liam Skender <liamskender@protonmail.com>

Description scHOTTER is a package that implements kernel-based local correlation analysis for gene pairs and an analytic null for the sample variance of Fisher z correlations across kernels that comprises a parametric alternative to Ghazanfar et al.'s single-cell higher-order testing (scHOT) methodology (Ghazanfar et al., 2020). This pipeline offers improvements in computational efficiency and statistical power, supports both 2D spatial and 1D trajectory (1D) data, and provides a high-level pipeline with access to all intermediate results.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

Config/testthat/edition 3

Imports Matrix,
methods,
RANN

VignetteBuilder knitr

Contents

`approximate_between_coefficient_correlations_effective`

Approximate correlation between kernel-wise Fisher coefficients

---

Description

Computes an overlap-based correlation between kernels from a point×kernel weight matrix W. First, columns are normalized to $A = WD^{-1}$, where $D = \mathrm{diag}(\sum_\ell w_{k\ell})$. Then $\Omega = A^\top A$ measures weighted overlap, and we form the cosine-similarity-style correlation

$$C = D_\Omega^{-1/2}\,\Omega\,D_\Omega^{-1/2},$$

where $D_\Omega = \mathrm{diag}(\Omega)$.

Usage

```
approximate_between_coefficient_correlations_effective(weight_matrix)
```

Arguments

`weight_matrix`	Point $\times$ kernel weight matrix (preferably `Matrix::dgCMatrix`).

Details

Optionally, if effective sample sizes `neff` are provided, we apply the Fisher-z attenuation factor $s_k = \sqrt{(n_{\mathrm{eff},k} - 3)/(n_{\mathrm{eff},k} - 1)}$ (for $n_{\mathrm{eff},k} > 3$) by

$$C \leftarrow S\,C\,S, \quad S = \mathrm{diag}(s_1, \ldots, s_K).$$

The diagonal is set to 1 for kernels with nonzero overlap norm and `NA` for empty/degenerate kernels.

Value

A symmetric numeric matrix $K \times K$ with row/column names equal to kernel names. Diagonal entries are 1 when the kernel has positive norm (nonempty) and `NA` otherwise.

Examples

```
set.seed(1)
coords  <- cbind(runif(50), runif(50))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
W      <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "gaussian")
C1       <- approximate_between_coefficient_correlations_effective(W)
neff     <- get_effective_sample_sizes(W)
C2       <- approximate_between_coefficient_correlations_effective(W)
range(C1, na.rm = TRUE)
```

---

approximate_expectation_effective

Expectation of the sample variance of Fisher z's under dependence

---

Description

Computes $\mathbb{E}[S^2]$ for the unbiased sample variance $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Z_i - \bar{\boldsymbol{Z}})^2$ when the kernel-wise Fisher z vector $\boldsymbol{Z}$ has covariance matrix $\boldsymbol{\Sigma}$. The formula used is

$$\mathbb{E}[S^2] = \frac{1}{n-1}\left(\mathrm{tr}(\boldsymbol{H}\boldsymbol{\Sigma})\right), \quad \boldsymbol{H} = \boldsymbol{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$$

where $\boldsymbol{H}$ is the centreing matrix. For independent components with common variance σ^2, this reduces to $\mathbb{E}[S^2] = \sigma^2$.

Usage

```
approximate_expectation_effective(sigma_matrix)
```

Arguments

`sigma_matrix`	A symmetric $n \times n$ covariance matrix for the kernel-wise Fisher z coefficients (e.g., from `get_sigma_matrix()`). May be a base matrix or a **Matrix** object. Rows/columns containing `NA`s are dropped prior to computation.

Value

A single numeric: $\mathbb{E}[S^2]$. Returns `NA_real_` if, after dropping `NA` rows/cols, fewer than 2 kernels remain.

Examples

```
# Sanity check: independent equal-variance case
n <- 8; v <- 0.2
Sigma <- diag(rep(v, n))
approximate_expectation_effective(Sigma)  # = v
```

`approximate_variance_effective` *Variance of the sample variance of Fisher z's under dependence*

Description

Computes $\mathrm{Var}(S^2)$ for the unbiased sample variance $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Z_i - \bar{\boldsymbol{Z}})^2$ when the kernel-wise Fisher z vector Z has covariance matrix $\boldsymbol{\Sigma}$. The formula used is

$$\mathrm{Var}(S^2) = \frac{2}{(n-1)^2}\left(\mathrm{tr}(\boldsymbol{\Sigma}^2) - \frac{2}{n}\,\mathbf{1}^\top\boldsymbol{\Sigma}^2\mathbf{1} + \frac{(\mathbf{1}^\top\boldsymbol{\Sigma}\mathbf{1})^2}{n^2}\right),$$

which reduces to $2\sigma^4/(n-1)$ for independent components with common variance σ^2.

Usage

```
approximate_variance_effective(sigma_matrix)
```

Arguments

`sigma_matrix`	A symmetric $n \times n$ covariance matrix for the kernel-wise Fisher z coefficients (e.g., from `get_sigma_matrix()`). May be a base matrix or a **Matrix** object. Rows/columns containing `NA`s are dropped prior to computation.

Value

A single numeric: $\mathrm{Var}(S^2)$. Returns `NA_real_` if, after dropping `NA` rows/cols, fewer than 2 kernels remain.

Examples

```
# Sanity check: independent equal-variance case
set.seed(1)
n <- 10
v <- 0.2
Sigma <- diag(rep(v, n))
approximate_variance_effective(Sigma)  # ~ 2 * v^2 / (n-1)
```

`compute_p_values` *Upper-tail p-values for standardized test statistics*

Description

Computes one-sided p-values $P(Z > z)$ assuming the input test statistics are standard normal under the null. This is appropriate when larger values of the statistic (e.g., a standardized sample variance) indicate stronger evidence against the null.

Usage

```
compute_p_values(test_stats)
```

Arguments

`test_stats`	Named numeric vector of Z-scores (e.g., from `compute_test_statistics_standardised_effecti`

Value

A named numeric vector of upper-tail p-values in $[0, 1]$. Non-finite inputs (NA/Inf) return `NA_real_`.

Examples

```
s2  <- c(a_b = 3, a_c = 5, b_c = 4)
mu  <- 4
var <- 1
z   <- compute_test_statistics_standardised_effective(var, mu, s2)
p   <- compute_p_values(z)
p
```

`compute_test_statistics_standardised_effective`
Standardize sample-variance test statistics into Z-scores

Description

Converts the raw sample variance of local Fisher z correlations (one value per gene pair) into standardized Z-scores using the null mean and variance derived from the kernel dependence structure. The resulting standardised statistic is distributed approximately according to the standard Normal distribution.

Usage

```
compute_test_statistics_standardised_effective(
  variances,
  expectations,
  vars_of_local_corrs
)
```

Arguments

`variances`	Numeric scalar or vector: $\mathrm{Var}(S^2)$ under the null. Typically the scalar output of `approximate_variance_effective(Sigma)`. Can also be a named vector matching `vars_of_local_corrs`.
`expectations`	Numeric scalar or vector: $\mathbb{E}[S^2]$ under the null. Typically the scalar output of `approximate_expectation_effective(Sigma)`. Can also be a named vector matching `vars_of_local_corrs`.
`vars_of_local_corrs`	Named numeric vector: observed sample variances (one per gene pair), e.g. from `compute_var_of_local_corrs()`.

Value

A named numeric vector of Z-scores aligned with `names(vars_of_local_corrs)`. Entries with non-finite or non-positive variances are returned as `NA_real_`.

Examples

```
# Simple algebraic example
s2  <- c(a_b = 3, a_c = 5, b_c = 4)
mu  <- 4      # E[S^2]
var <- 1.0    # Var(S^2)
compute_test_statistics_standardised_effective(var, mu, s2)
```

---

compute_var_of_local_corrs
Sample variance of kernel-wise Fisher z per gene pair

Description

Given a matrix of local Fisher z correlations (rows = kernels, columns = gene pairs like "GENE1_GENE2"), compute the unbiased sample variance S^2 down each column, ignoring NAs. Columns with fewer than two finite values return NA_real_.

Usage

```
compute_var_of_local_corrs(local_corr_matrix)
```

Arguments

local_corr_matrix	Numeric matrix with rows = kernels and columns = gene-pair labels (e.g., from get_local_correlation_matrix() with fisher_transform = TRUE).

Value

A named numeric vector of length ncol(local_corr_matrix) containing the sample variance for each gene pair.

Examples

```
set.seed(1)
expr <- cbind(a = rnorm(60), b = rnorm(60), c = rnorm(60))
coords <- cbind(runif(60), runif(60))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
W <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "gaussian")
Z <- get_local_correlation_matrix(expr, W, fisher_transform = TRUE)
s2 <- compute_var_of_local_corrs(Z)
head(s2)
```

determine_overlaps — *Determine overlapping kernels (share $\geq$ 1 point)*

Description

From a binary membership matrix M (points $\times$ kernels), compute which kernels overlap, where overlap means at least one shared point. Internally uses $A = M^\top M$; $A_{ij} > 0$ indicates that kernels i and j share at least one point. This is an intermediate step required to determine which kernels are "light" and shouldn't be considered in downstream analysis.

Usage

```
determine_overlaps(membership_matrix)
```

Arguments

`membership_matrix`	A 0/1 matrix (preferably `Matrix::dgCMatrix`) with rows = points and cols = kernels. You can obtain this via `generate_membership_matrix`.

Value

A named list of length k (number of kernels). Element `[[i]]` is a character vector of kernel names that overlap with kernel `i` **including itself**.

Examples

```
# Minimal controlled example
M <- Matrix::sparseMatrix(i = c(1,2,2,3), j = c(1,1,2,2), x = 1,
                          dims = c(3,2), dimnames = list(NULL, c("k_1","k_2")))
determine_overlaps(M)

# Typical workflow
set.seed(1)
coords  <- cbind(runif(40), runif(40))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
W       <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "block")
Memb    <- generate_membership_matrix(W)
ovl     <- determine_overlaps(Memb)
str(ovl[1:2])
```

find_light_kernels — *Identify "light" kernels with too few unique points*

Description

Flags kernels that don't have enough assigned points relative to how many other kernels they overlap with. The criterion used is `num_points < num_overlaps`, where `num_points` is the column sum of the (0/1) membership matrix and `num_overlaps` is the length of the overlap set for that kernel (typically including itself).

Usage

```
find_light_kernels(overlapping_list, membership_matrix)
```

Arguments

`overlapping_list`	A named list where each element `overlapping_list[[k]]` is a character vector of kernel names that overlap kernel `k`. See `determine_overlaps`.
`membership_matrix`	A 0/1 matrix (preferably `Matrix::dgCMatrix`) of point–kernel memberships with rows = points, cols = kernels.

Details

This function assumes the overlap list includes each kernel itself (as produced by `determine_overlaps()`). If you supply an overlap list that excludes self, the threshold will be one smaller; adjust upstream if needed.

Value

A character vector of kernel names deemed "light".

Examples

```
set.seed(1)
coords  <- cbind(runif(40), runif(40))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
W       <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "block")
Memb    <- generate_membership_matrix(W)
ovl     <- determine_overlaps(Memb)
find_light_kernels(ovl, Memb)
```

---

generate_kernel_centres_by_density

Generate approximate kernel centres on a grid to hit a target span

---

Description

Places kernel centres on a regular grid over the 2D coordinate bounds so that, on average, each kernel would contain roughly `span * n_points` data points (under a uniform-density approximation). This is useful for building spatial kernels whose size scales with the observed point density.

Usage

```
generate_kernel_centres_by_density(coords, span = NULL)
```

Arguments

`coords`	A numeric matrix or data frame with two columns giving x,y coordinates of points (rows are points). Column order is `c(x, y)`.
`span`	Optional numeric in $(0, 1]$. Interpreted as the target proportion of data points per kernel. If `NULL`, a heuristic default $13/n_{\text{points}}$ is used (capped at `0.5`); a message is emitted.

Details

Let n be the number of points and A the area of the bounding box of `coords`. We approximate point density by n/A and choose a square kernel with side length $\sqrt{(\text{target\_n})/(n/A)}$ where $\text{target\_n} = \lceil \text{span} \times n \rceil$. Grid spacing equals this side length, producing $n_x \times n_y$ centres that tile the bounding box.

Value

A numeric matrix with two columns `c("x","y")` giving kernel centre coordinates.

Examples

```
set.seed(1)
coords <- cbind(runif(100, 0, 10), runif(100, 0, 5))
centres <- generate_kernel_centres_by_density(coords, span = 0.1)
dim(centres)       # number of centres
head(centres)
```

`generate_kernel_centres_by_density_1d`
Generate kernel centres for 1D (trajectory) coordinates

Description

For data where exactly one coordinate dimension varies across points (i.e. an effectively 1D trajectory embedded in $\mathbb{R}^p$), place kernel centres along the varying axis so that, on average, each kernel collects about `span * n_points` points (under a uniform-density approximation). Non-varying dimensions are fixed at their column medians.

Usage

```
generate_kernel_centres_by_density_1d(coords, span = NULL)
```

Arguments

`coords`	Numeric matrix or data frame of coordinates with `n` rows (points) and `p` columns (axes). Exactly one column must have non-zero range; other columns may be constant (or nearly so). The nonzero coordinate may correspond to a pseudo-timepoint or similar.
`span`	Optional numeric in $(0, 1]$. Target proportion of points per kernel. If `NULL`, a heuristic default $13/n$ is used (capped at `0.5`); a message is emitted.

Details

Let L be the range length of the varying axis and n the number of points. We approximate the 1D density by n/L, target the number of points per kernel as $\lceil \text{span} \cdot n \rceil$, set the kernel length to $\text{target\_n}/(n/L)$, and place $\lceil L/\text{kernel\_len} \rceil$ centres evenly over the axis.

Value

A numeric matrix with the same columns as coords. The varying axis is filled with an evenly spaced grid over its range; constant axes are filled with their (NA-robust) medians. Row count equals the number of kernel centres.

Examples

```
set.seed(1)
t  <- sort(runif(100, 0, 10))   # pseudotime
y0 <- rep(0, 100)               # constant second dim
centres <- generate_kernel_centres_by_density_1d(cbind(t, y0), span = 0.2)
head(centres)
```

---

generate_membership_matrix
Convert a weight matrix to a 0/1 membership matrix

Description

Turns any nonzero weight into 1 and zeros into 0, preserving the sparse structure and dimension names. This is useful for assigning points to kernels based on whether their weight is positive (regardless of magnitude). This is the intermediate step required to generate the list of overlapping kernels.

Usage

```
generate_membership_matrix(weight_matrix)
```

Arguments

weight_matrix	A numeric matrix or a Matrix::dgCMatrix of weights with rows as points and columns as kernels (e.g., from generate_weight_matrix_euclidean).

Value

A Matrix::dgCMatrix with the same dimensions and dimnames as weight_matrix, containing only 0/1 entries.

Examples

```
set.seed(1)
coords  <- cbind(runif(30), runif(30))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
W      <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "gaussian")
Mb      <- generate_membership_matrix(W)
Matrix::nnzero(Mb)      # number of assigned point-kernel memberships
unique(Mb@x)            # should be c(1) (no stored zeros)
```

generate_weight_matrix_euclidean

Build a point-by-kernel weight matrix (Euclidean neighborhoods)

Description

Computes a sparse $n_{\text{points}} \times n_{\text{kernels}}$ weight matrix `W` using Euclidean distance from each kernel centre to each point. Two schemes are supported:

- **block**: weights are 1 for neighbors within the search radius, 0 otherwise.
- **gaussian**: weights are $\exp\{-d^2/(2\sigma^2)\}$, with $\sigma = \text{gap}/2$, where gap is the minimum grid spacing, and d is the distance from the kernel centre to the point of interest.

Usage

```
generate_weight_matrix_euclidean(
  coords,
  grid_centres,
  span = NULL,
  type = c("block", "gaussian")
)
```

Arguments

`coords`	Numeric matrix or data frame with two columns `c(x, y)` giving point coordinates (rows are points).
`grid_centres`	Numeric matrix or data frame with two columns `c(x, y)` giving kernel-centre coordinates (e.g., from `generate_kernel_centres_by_density`).
`span`	Optional numeric in $(0, 1]$. Target proportion of points to associate with each kernel (used to choose `k = ceiling(span * n_points)` neighbors). If `NULL`, defaults to $13/n_{\text{points}}$ (capped at 0.5).
`type`	Character; either `"block"` or `"gaussian"`.

Details

Grid spacing (gap). The search radius is set to the smallest spacing between unique `x`- or `y`-coordinates of `grid_centres`. If only a single centre exists on an axis, we fall back to a coarse spacing based on the data range.

Neighbor search. We query up to $k = \lceil \text{span}\, n \rceil$ nearest neighbors per kernel centre using **RANN**. If your **RANN** version does not support radius-restricted search, a standard k-NN query still works; the `"block"` scheme then effectively selects the returned neighbors.

Value

A `Matrix::dgCMatrix` sparse matrix `W` with rownames taken from `coords` (if present) and column names `"k_1", ..., "k_m"` for m kernels.

See Also

`generate_kernel_centres_by_density`

Examples

```
set.seed(1)
coords <- cbind(runif(100, 0, 10), runif(100, 0, 5))
centres <- generate_kernel_centres_by_density(coords, span = 0.1)
Wb <- generate_weight_matrix_euclidean(coords, centres, span = 0.1, type = "block")
Wg <- generate_weight_matrix_euclidean(coords, centres, span = 0.1, type = "gaussian")
dim(Wb); Matrix::nnzero(Wb)
```

`generate_weight_matrix_euclidean_1d` *Build a point-by-kernel weight matrix for 1D (trajectory) data*

Description

Computes a sparse $n_{\text{points}} \times n_{\text{kernels}}$ weight matrix using Euclidean distance along the single varying coordinate (pseudotime). Two schemes are supported:

- **block**: weights are 1 for neighbors within the search radius, 0 otherwise.
- **gaussian**: weights are $\exp\{-d^2/(2\sigma^2)\}$, with $\sigma = \text{gap}/2$, where gap is the kernel spacing (or a fallback radius computed from density) and d is the distance from the kernel centre to the point of interest.

Usage

```
generate_weight_matrix_euclidean_1d(
  coords,
  grid_centres,
  span = NULL,
  type = c("block", "gaussian")
)
```

Arguments

`coords`	Numeric matrix/data frame of coordinates. Exactly one column must vary (effective 1D). Other columns may be constant and are ignored for distances.
`grid_centres`	Numeric matrix/data frame of kernel-centre coordinates (e.g., from `generate_kernel_centres_by_`
`span`	Optional numeric in $(0, 1]$. Target proportion of points to associate with each kernel (`k = ceiling(span * n_points)`). If `NULL`, defaults to $13/n_{\text{points}}$ (capped at 0.5).
`type`	Character; either `"block"` or `"gaussian"`.

Value

A `Matrix::dgCMatrix` with rows = points and columns = kernels (`"k_1", ..., "k_m"`).

See Also

`generate_kernel_centres_by_density_1d`, `generate_weight_matrix_euclidean`

Examples

```
set.seed(1)
t  <- sort(runif(80, 0, 10))
y0 <- rep(0, 80)
coords <- cbind(t = t, y = y0)
centres <- generate_kernel_centres_by_density_1d(coords, span = 0.2)
Wb <- generate_weight_matrix_euclidean_1d(coords, centres, span = 0.2, type = "block")
Wg <- generate_weight_matrix_euclidean_1d(coords, centres, span = 0.2, type = "gaussian")
dim(Wb); Matrix::nnzero(Wb)
```

`get_effective_sample_sizes`

Effective sample size per kernel (Kish, 1965)

Description

Computes Kish's effective sample size for each kernel/column of a weight matrix:

$$n_{\text{eff},k} = \frac{(\sum_\ell w_{k\ell})^2}{\sum_\ell w_{k\ell}^2}.$$

Usage

```
get_effective_sample_sizes(weight_mat_trimmed, tol = 1e-12)
```

Arguments

`weight_mat_trimmed`	A point×kernel weight matrix (preferably `Matrix::dgCMatrix`); can be the output of `trim_weight_matrix`.
`tol`	Numeric tolerance. Columns with $\sum w^2 \leq$ `tol` are treated as empty and return `0`.

Details

For block weights (0/1), $n_{\text{eff},k}$ equals the number of positively weighted points in kernel k. For general nonnegative weights, $n_{\text{eff},k} \leq \sum_\ell 1\{w_{k\ell} > 0\}$.

Value

A named numeric vector of length k (kernels), giving n_{eff} for each kernel (0 for empty kernels).

References

Kish, L. (1965). *Survey Sampling*. Wiley.

Examples

```
set.seed(1)
coords  <- cbind(runif(40), runif(40))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
Wg <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "gaussian")
neff <- get_effective_sample_sizes(Wg)
head(neff)
```

---

get_local_correlation_matrix

Local (kernel-wise) Fisher correlations for all gene pairs

Description

For every unordered gene pair in `expr`, compute a vector of weighted Pearson correlations $\hat{r}$—one per kernel—using `weight_matrix`. By default the correlations are Fisher-transformed ($z = \operatorname{atanh}(\hat{r})$). Columns are labeled as `"GENE1_GENE2"`.

Usage

```
get_local_correlation_matrix(
  expr,
  weight_matrix,
  fisher_transform = TRUE,
  drop_zero_weight = TRUE
)
```

Arguments

`expr`	Numeric matrix (rows = points/cells, cols = genes).
`weight_matrix`	Point×kernel weight matrix (preferably `Matrix::dgCMatrix`) with `nrow(weight_matrix) == nrow(expr)`.
`fisher_transform`	Logical; if `TRUE` (default), return Fisher-z values; otherwise return raw correlations.
`drop_zero_weight`	Logical; if `TRUE` (default), restrict each kernel's calculation to rows with strictly positive weight for that kernel.

Value

A numeric matrix with `nrow = ncol(weight_matrix)` (kernels) and `ncol = choose(p, 2)` (gene pairs, where `p = ncol(expr)`). Row names are kernel names; column names are `"GENE1_GENE2"`. Entries are Fisher-z if `fisher_transform=TRUE`, else raw $\hat{r}$.

See Also

`get_local_correlation_vector_pair`

Examples

```
set.seed(1)
expr <- cbind(g1 = rnorm(60), g2 = rnorm(60), g3 = rnorm(60))
coords <- cbind(runif(60), runif(60))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
W <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "gaussian")
Zmat <- get_local_correlation_matrix(expr, W)
dim(Zmat)   # kernels x choose(3,2) = kernels x 3
```

`get_local_correlation_vector_pair`
Local (kernel-wise) weighted correlation for a gene pair

Description

For a gene pair encoded as `"GENE1_GENE2"`, compute a vector of weighted Pearson correlations $\hat{r}$—one per kernel—from a point×kernel weight matrix. By default, correlations are Fisher-transformed ($z = \operatorname{atanh}(\hat{r})$).

Usage

```
get_local_correlation_vector_pair(
  expr,
  weight_matrix,
  gene_pair,
  fisher_transform = TRUE,
  drop_zero_weight = TRUE
)
```

Arguments

`expr`	Numeric matrix (rows = points/cells, cols = genes) with column names containing the requested genes.
`weight_matrix`	A point×kernel weight matrix (preferably `Matrix::dgCMatrix`) with the same number of rows as `expr`.
`gene_pair`	Character scalar like `"GENE1_GENE2"` indicating the two gene column names in `expr`.
`fisher_transform`	Logical; if `TRUE` (default), apply the Fisher atanh transform to each correlation. Values with $\lvert r \rvert \geq 1$ or non-finite are returned as `NA_real_`.
`drop_zero_weight`	Logical; if `TRUE` (default), restrict each kernel's computation to rows with strictly positive weight for that kernel. If `FALSE`, all rows are considered (zeros contribute nothing).

Value

A named numeric vector of length k (number of kernels), where names are the kernel column names in `weight_matrix`. Entries are Fisher-z values if `fisher_transform=TRUE`, otherwise raw correlations. Kernels with fewer than 2 positively weighted points (or undefined variance) return `NA_real_`.

Examples

```
set.seed(1)
expr <- cbind(g1 = rnorm(60), g2 = rnorm(60))
coords <- cbind(runif(60), runif(60))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
W <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "gaussian")
zvec <- get_local_correlation_vector_pair(expr, W, "g1_g2")
head(zvec)
```

`get_sigma_matrix` *Covariance matrix of kernel-wise Fisher z coefficients*

Description

Given a kernel–kernel correlation matrix `corr` (for the Fisher coefficients) and effective sample sizes `neff`, returns the covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{D}\,\mathrm{corr}\,\boldsymbol{D}$ where $\boldsymbol{D} = \mathrm{diag}(\sqrt{\boldsymbol{v}})$ and $v_k = 1/(n_{\mathrm{eff},k} - 3)$ (variance of Fisher z for a correlation).

Usage

```
get_sigma_matrix(corr, neff)
```

Arguments

`corr`	A symmetric $n{\times}n$ correlation matrix between kernels (e.g., from `approximate_between_coefficie` Preferably a `Matrix` object; a base matrix is also accepted. **Pass the unattenuated correlation** (call with `neff = NULL`) so you don't double-apply any scaling.
`neff`	Numeric vector of length n with effective sample sizes per kernel (e.g., from `get_effective_sample_sizes(W)`).

Details

For kernels with $n_{\mathrm{eff}} \leq 3$, the Fisher z variance is undefined. In that case, we set the corresponding diagonal multiplier to zero while forming $\boldsymbol{D}\,\mathrm{corr}\,\boldsymbol{D}$, and then mark the affected rows/columns of $\boldsymbol{\Sigma}$ as `NA`.

Value

A `Matrix` object of size $n \times n$ with row/column names taken from `corr`. Entries corresponding to kernels with $n_{\mathrm{eff}} \leq 3$ are returned as `NA_real_`.

Examples

```
set.seed(1)
coords  <- cbind(runif(60), runif(60))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
W       <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "gaussian")
neff    <- get_effective_sample_sizes(W)
corr    <- approximate_between_coefficient_correlations_effective(W)  # unattenuated
Sigma   <- get_sigma_matrix(corr, neff)
Matrix::diag(Sigma)[1:5]
```

`scHOTTER_pipeline` *End-to-end pipeline: local co-expression p-values (stores intermediates)*

Description

Takes a gene expression matrix whose **rownames encode 2D coordinates** (e.g. "12.3x45.6", "12.3_45.6", "12.3,45.6", or "12.3 45.6") and returns one-sided upper-tail p-values for the standardized sample-variance statistic of kernel-wise Fisher z correlations, *plus* all intermediate objects.

Usage

```
scHOTTER_pipeline(
  expr,
  span = NULL,
  kernel_type = c("block", "gaussian"),
  fisher_transform = TRUE,
  drop_zero_weight = TRUE
)
```

Arguments

`expr`	Numeric matrix (points $\times$ genes). **Row names must contain** "x y" coordinates separated by `x`, `_`, `,`, or whitespace (e.g. `"10.2_5.7"`). Column names are gene symbols.
`span`	Optional numeric in (0,1]; target proportion of points per kernel (passed to kernel generation and weighting). Default uses the package heuristic if `NULL`.
`kernel_type`	`"block"` or `"gaussian"`; passed to `generate_weight_matrix_euclidean()`.
`fisher_transform`	Logical; Fisher-transform local correlations (`atanh`). Must be `TRUE` to match downstream variance formulas (default `TRUE`).
`drop_zero_weight`	Logical; restrict each kernel's correlation to rows with `w>0` for that kernel (default `TRUE`).

Value

A list with:

- `p_values`: named numeric vector of p-values (one per gene pair).
- `summary`: data.frame with columns `pair`, `s2`, `z`, `p`.
- `intermediates`: a list containing `coords`, `centres`, `weight_matrix_initial`, `membership`, `overlaps`, `light_kernels`, `weight_matrix`, `local_corr_matrix`, `neff`, `corr`, `sigma`, `expectation`, `variance`, `s2`, `z_scores`, `p_values`.
- `params`: the input parameters actually used.

Examples

```
set.seed(1)
n <- 80
coords <- cbind(runif(n, 0, 10), runif(n, 0, 5))
rn <- apply(coords, 1, function(v) paste0(v[1], "_", v[2]))
expr <- cbind(g1 = rnorm(n), g2 = rnorm(n), g3 = rnorm(n))
rownames(expr) <- rn
out <- scHOTTER_pipeline(expr, span = 0.2, kernel_type = "gaussian")
head(out$summary)
```

---

scHOTTER_pipeline_1d *End-to-end pipeline (1D trajectory): local co-expression p-values*

Description

A 1D analogue of `scHOTTER_pipeline()` for trajectory-like data where *exactly one* coordinate axis varies across points. Row names of `expr` must encode coordinates (e.g. "xxy", "x_y", "x,y" or "x y"). The varying axis is used to place kernel centres and compute 1D weights; all subsequent steps (local Fisher z, analytic null, Z, p) are identical to the 2D pipeline.

Usage

```
scHOTTER_pipeline_1d(
  expr,
  span = NULL,
  kernel_type = c("block", "gaussian"),
  fisher_transform = TRUE,
  drop_zero_weight = TRUE
)
```

Arguments

`expr`	Numeric matrix (points $\times$ genes). Row names must encode coordinates. Column names are gene symbols.
`span`	Optional numeric in (0, 1]; target proportion of points per kernel. If `NULL`, a heuristic $13/n$ capped at 0.5 is used.
`kernel_type`	`"block"` or `"gaussian"`; passed to `generate_weight_matrix_euclidean_1d`.
`fisher_transform`	Logical; apply Fisher `atanh` to local correlations (default `TRUE`). Keep `TRUE` to match the null variance model.
`drop_zero_weight`	Logical; restrict each kernel's computation to rows with weight > `0` for that kernel (default `TRUE`).

Value

A list with:

- `p_values`: named numeric vector of p-values (one per gene pair).
- `summary`: data.frame with columns `pair`, `s2`, `z`, `p`.

- `intermediates`: a list containing `coords`, `centres`, `weight_matrix_initial`, `membership`, `overlaps`, `light_kernels`, `weight_matrix`, `local_corr_matrix`, `neff`, `corr`, `sigma`, `expectation`, `variance`, `s2`, `z_scores`, `p_values`.
- `params`: the input parameters actually used.

Examples

```
set.seed(1)
n <- 80
t  <- sort(runif(n, 0, 10))
y0 <- rep(0, n)                          # constant second axis
coords <- cbind(t = t, y = y0)
rn <- apply(coords, 1, function(v) paste0(v[1], "_", v[2]))
expr <- cbind(g1 = rnorm(n), g2 = rnorm(n), g3 = rnorm(n))
rownames(expr) <- rn

out1d <- scHOTTER_pipeline_1d(expr, span = 0.2, kernel_type = "gaussian")
head(out1d$summary)
```

`trim_weight_matrix` *Trim kernels flagged as "light" from a weight matrix*

Description

Removes columns (kernels) listed in `light_kernels` from a point×kernel weight matrix. Returns the original matrix if no kernels are specified.

Usage

```
trim_weight_matrix(weight_matrix, light_kernels)
```

Arguments

`weight_matrix`	A numeric matrix or `Matrix::dgCMatrix` with rows = points, cols = kernels (e.g., from `generate_weight_matrix_euclidean()`). Must have column names.
`light_kernels`	Character vector of kernel names to remove. If empty, the input matrix is returned unchanged.

Value

A `Matrix::dgCMatrix` with the specified columns removed (possibly with zero columns if all kernels are removed).

Examples

```
set.seed(1)
coords  <- cbind(runif(40), runif(40))
centres <- generate_kernel_centres_by_density(coords, span = 0.2)
W      <- generate_weight_matrix_euclidean(coords, centres, span = 0.2, type = "gaussian")
Memb    <- generate_membership_matrix(W)
ovl     <- determine_overlaps(Memb)
light   <- find_light_kernels(ovl, Memb)
```

```
W_trim  <- trim_weight_matrix(W, light)
dim(W_trim)
```