Package 'wSIR'

August 1, 2026

Type Package

Title Weighted Sliced Inverse Regression (wSIR) for supervised dimension reduction of spatial transcriptomics and single cell gene expression data

Version 0.99.9

Description Weighted Sliced Inverse Regression (wSIR) is a supervised dimension reduction algorithm for spatial transcriptomics gene expression data. For a provided gene expression matrix and dataframe of each cell's spatial coordinates, wSIR creates a low-dimensional representation of the gene expression data that preserves the ability to predict spatial coordinates that was present in the gene expression data. Furthermore, wSIR provides interpretable loadings which allow for projection of new single-cell gene expression data into a low-dimensional space which preserves the spatial information present in the gene expression data.

License MIT + file LICENSE

Encoding UTF-8

URL https://sydneybiox.github.io/wSIR, https://github.com/SydneyBioX/wSIR

BugReports https://github.com/sydneybiox/wSIR/issues

biocViews DimensionReduction, Software, GeneExpression, SingleCell, Spatial, Transcriptomics, Regression, CellBasedAssays

Roxygen list(markdown = TRUE)

Depends R (>= 4.4.0),

Imports magrittr (>= 2.0) , ggplot2 (>= 3.5.1), umap (>= 0.2.10), vctrs (>= 0.6), stringr (>= 1.5.1), distances (>= 0.1.11), Rcpp (>= 1.0.11), doBy (>= 4.6.0), BiocParallel (>= 1.38.0), rlang, methods, BiocGenerics (>= 0.50.0), SummarizedExperiment (>= 1.34.0), SingleCellExperiment (>= 1.26.0), SpatialExperiment (>= 1.14.0)

Suggests knitr, BiocStyle, class, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation yes

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/wSIR

git_branch devel

git_last_commit 7d58fcb

git_last_commit_date 2026-07-13

Repository Bioconductor 3.24

Date/Publication 2026-07-31

Author Max Woollard [aut, cre] (ORCID:
<https://orcid.org/0009-0000-6319-6926>),
Pratibha Panwar [ctb] (ORCID: <https://orcid.org/0000-0002-7437-7084>),
Linh Nghiem [aut] (ORCID: <https://orcid.org/0000-0003-2874-9067>),
Shila Ghazanfar [aut] (ORCID: <https://orcid.org/0000-0001-7861-6997>)

Maintainer Max Woollard <mwoo5086@uni.sydney.edu.au>

Contents

`wSIR-package` *wSIR: Weighted Sliced Inverse Regression (wSIR) for supervised dimension reduction of spatial transcriptomics and single cell gene expression data*

Description

Weighted Sliced Inverse Regression (wSIR) is a supervised dimension reduction algorithm for spatial transcriptomics gene expression data. For a provided gene expression matrix and dataframe of each cell's spatial coordinates, wSIR creates a low-dimensional representation of the gene expression data that preserves the ability to predict spatial coordinates that was present in the gene expression data. Furthermore, wSIR provides interpretable loadings which allow for projection of new single-cell gene expression data into a low-dimensional space which preserves the spatial information present in the gene expression data.

Author(s)

Maintainer: Max Woollard <mwoo5086@uni.sydney.edu.au> (ORCID)

Authors:

- Linh Nghiem (ORCID)
- Shila Ghazanfar <shazanfar@gmail.com> (ORCID)

Other contributors:

- Pratibha Panwar (ORCID) [contributor]

See Also

Useful links:

- https://sydneybiox.github.io/wSIR
- https://github.com/SydneyBioX/wSIR
- Report bugs at https://github.com/sydneybiox/wSIR/issues

`calculateWSIR` *calculateWSIR*

Description

Perform wSIR on cells, based on the expression data and a reducedDim in a SingleCellExperiment or SpatialExperiment object

Usage

```
calculateWSIR(
  x,
  assay_type = "logcounts",
  dim_red = NULL,
  colData_columns = NULL,
  spatialCoords = FALSE,
  ...
)
```

Arguments

x	A numeric matrix of normalised gene expression data where rows are features and columns are cells. Alternatively, a SingleCellExperiment or SpatialExperiment containing such a matrix
assay_type	if x is a SingleCellExperiment of SpatialExperiment then this is the assay for which wSIR will be calculated. Default "logcounts".
dim_red	String or integer scalar specifying the dimensionality reduction slot for which to use for the slicing mechanism. Ignored if coords given.
colData_columns	character vector specifying the subset of colData columns to be used for the wSIR slicing mechanism. Ignored if coords or dim_red given
spatialCoords	logical indicating if spatialCoords should be used for the wSIR slicing mechanism. Ignored if coords, dim_red, or colData_columns given, or if x is not a SpatialExperiment object.
...	arguments passing to wSIR

Value

A wSIR object

---

createWeightMatrix *createWeightMatrix*

Description

A function to create the weight matrix given the location of the cells, tile allocations and desired spatial weighting strength. Weight matrix entries represent level of spatial correlation between all pairs of tiles.

Usage

```
createWeightMatrix(coords, labels, alpha = 4)
```

Arguments

coords	dataframe of dimension n * 2. Column names c("x", "y"). Spatial position of each cell.
labels	dataframe of dimension n * 1, column name c("coordinate"). Tile allocation of each cell. This is automatically created in the wSIR function.
alpha	numeric value giving strength of spatial weighting matrix. alpha = 0 returns identity matrix and equals SIR. Large alpha values tend all entries towards 1. Default is 4.

Value

matrix containing the weight value for all pairs of tiles. Each value is between 0 and 1, with 1 always on the diagonal.

exploreWSIRParams *exploreWSIRParams function*

Description

This function is used to select the optimal values for parameters slices and alpha in weighted sliced inverse regression based on your provided gene expression data and corresponding spatial coordinates. For a given evaluation metric, it will visualise the performance of WSIR with varying function parameters based on your data, and return the optimal pair. This pair of slices and alpha can be used for your downstream tasks.

Usage

```
exploreWSIRParams(
  X,
  coords,
  samples = rep(1, nrow(coords)),
  optim_alpha = c(0, 2, 4, 10),
  optim_slices = c(5, 10, 15, 20),
  metric = "DC",
  n_rep = 50,
  plot = TRUE,
  verbose = TRUE,
  BPPARAM = BiocParallel::SerialParam(RNGseed = .Random.seed[1]),
  ...
)
```

Arguments

X	matrix containing normalised gene expression data including n cells and p genes, dimension n * p.
coords	dataframe containing spatial positions of n cells in 2D space. Dimension n * 2. Column names must be c("x", "y").
samples	sample ID of each cell. In total, must have length equal to the number of cells. For example, if your dataset has 10000 cells, the first 5000 from sample 1 and the remaining 5000 from sample 2, you would write samples = c(rep(1, 5000), rep(2, 5000)) to specify that the first 5000 cells are sample 1 and the remaining are sample 2. Default is that all cells are from sample 1. Sample IDs can be of any format: for the previous example, you could write samples = c(rep("sample 1", 5000), rep("sample 2", 5000)), and the result would be the same.
optim_alpha	vector of numbers as the values of parameter alpha to use in WSIR. 0 gives Sliced Inverse Regression (SIR) implementation, and larger values represent stronger spatial correlation. Suggest to use integers for interpretability, but can use non-integers. Values must be non-negative.
optim_slices	vector of integers as the values of parameter slices to use in WSIR. Suggest maximum value in the vector to be no more than around $\sqrt{n/20}$, as this upper bound ensures an average of at least 10 cells per tile in the training set.
metric	character single value. Evaluation metric to use for parameter tuning to select optimal parameter combination. String, use "DC" to use distance correlation,

	"CD" to use correlation of distances, or "ncol" for the number of dimensions in the low-dimensional embedding. Default is "DC".
n_rep	integer for the number of train/test splits of the data to perform.
plot	logical whether a dotplot of parameters and metrics should be produced, default TRUE
verbose	default TRUE
BPPARAM	Optional parallel computing instance as in BiocParallelParam to be used in BiocParallel::bplapply. Default is BiocParallelParam instance with one core.
...	arguments passed on to wSIROptimisation

Value

List with five slots, named "plot", "message", "best_alpha", "best_slices" and "results_dataframe".

1. "plot" shows the average metric value across the n_rep iterations for every combination of parameters slices and alpha. Larger circles for a slices/alpha combination indicates better performance for that pair of values. There is one panel per evaluation metric selected in "metrics" argument.
2. "message" tells you the parameter combination with highest metric value according to selected metric.
3. "best_alpha" returns the integer for the best alpha values among the values that were tested according to selected metric.
4. "best_slices" returns the integer for the best slices value among the values that were tested according to selected metric.
5. "results_dataframe" returns the results dataframe used to create "plot". This dataframe has length(optim_alpha)*length(optim_slices) rows, where one is for each combination of parameters slices and alpha. There is one column for "alpha", one for "slices" and one for each of the evaluation metrics selected in "metrics" argument. Column "alpha" includes the value for parameter alpha, column "slices" includes the value for parameter slices, and each metric column includes the value for the specified metric, which is either Distance Correlation ("DC"), Correlation of Distances ("CD"), or number of columns in low-dimensional embedding ("ncol").

Examples

```
data(MouseData)
explore_params = exploreWSIRParams(X = sample1_exprs,
  coords = sample1_coords,
  optim_alpha = c(0,4),
  optim_slices = c(3,6))
explore_params$plot
explore_params$message
best_alpha = explore_params$best_alpha
best_slices = explore_params$best_slices
wsir_obj = wSIR(X = sample1_exprs,
    coords = sample1_coords,
    optim_params = FALSE,
    alpha = best_alpha,
    slices = best_slices)
```

findTopGenes *findTopGenes*

Description

A function to find and visualise the genes with the highest (in absolute value) loading in WSIR1. These genes contribute the most to the first low-dimensional direction.

Usage

```
findTopGenes(wsir, highest = 10, dirs = 1)
```

Arguments

wsir	wsir object as output of wSIR function. To analyse a different DR method, ensure the slot named 'directions' contains the loadings as a matrix with the gene names as the rownames.
highest	integer for how many of the top genes you would like to see. Recommend no more than 20 for ease of visualisation. Default is 10.
dirs	integer or vector for which direction / directions you want to show/find the top genes from.

Value

List containing two slots. First is named "plot" and shows a barplot with the top genes and their corresponding loadings. Second is named "genes" and is a dataframe of the genes with highest (in absolute value) loading in the specified low-dimensional directions. There are three columns: gene name, loading value, and which direction it is in. The number of columns is the product of parameters highest and dirs.

Examples

```
data(MouseData)

wsir_obj = wSIR(X = sample1_exprs,
  coords = sample1_coords,
  optim_params = FALSE,
  alpha = 4,
  slices = 6) # create wsir object
top_genes_obj = findTopGenes(wsir = wsir_obj, highest = 8)
top_genes_plot = top_genes_obj$plot # select plot
top_genes_plot # print plot
```

generateUmapFromWSIR *generateUmapfromWSIR*

Description

A function to generate UMAP coordinates from the low-dimensional embedding of the gene expression data. These coordinates can later be plotted with the plotUmapFromWSIR function. Those two functions are separate so that you can generate the UMAP points only once using this function (which may take a long time), then modify the resulting plot as much as desired with the plotUmapFromWSIR function.

Usage

```
generateUmapFromWSIR(wsir)
```

Arguments

wsir	wsir object that is output of wSIR function. If you wish to generate UMAP plots based on other DR methods, ensure that the slot named "scores" in WSIR parameter contains the low-dimensional representation of exprs.

Value

matrix of UMAP coordinates of dimension nrow(coords) * 2. Output of this function can be directly used as the input to the plot_umap function.

Examples

```
data(MouseData)
wsir_obj <- wSIR(X = sample1_exprs,
    coords = sample1_coords,
    optim_params = FALSE,
    alpha = 4,
    slices = 6) # create wsir object
umap_coords <- generateUmapFromWSIR(wsir = wsir_obj)
top_genes_obj <- findTopGenes(wsir = wsir_obj, highest = 4)
umap_plot <- plotUmapFromWSIR(umap_coords = umap_coords,
    X = sample1_exprs,
    highest_genes = top_genes_obj,
    n_genes = 4)
umap_plot
```

MouseData *Mouse Gastrulation Data*

Description

Data set consists of spatial transcriptomics data from a mouse embryo. There are three samples, for each we have gene expression data (351 genes), spatial coordinates on the two-dimensional spatial plane, and cell type labels. Sample 1 contains 19451 cells, sample 2 contains 14891 cells and sample 3 contains 23194 cells. We have randomly sampled 20% of the cells from each of these datasets for the vignette and examples to stop the data being too large. After the random sampling, we have 3848 cells in sample 1, 2986 cells in sample 2 and 4607 cells in sample 3.

Format

`sample1_exprs` has a row for each cell in sample 1 and a column for the expression level of each gene. `sample1_coords` has a row for each cell in sample 1 and a column for its position in each of the two spatial axes. `sample1_cell_types` a vector whose i'th entry contains the cell type of the i'th cell of sample 1. `sample2_exprs` has a row for each cell in sample 2 and a column for the expression level of each gene. `sample2_coords` has a row for each cell in sample 2 and a column for its position in each of the two spatial axes. `sample2_cell_types` a vector whose i'th entry contains the cell type of the i'th cell of sample 2. `sample3_exprs` has a row for each cell in sample 3 and a column for the expression level of each gene. `sample3_coords` has a row for each cell in sample 3 and a column for its position in each of the two spatial axes. `sample3_cell_types` a vector, whose i'th entry contains the cell type of the i'th cell of sample 3.

Source

Integration of spatial and single-cell transcriptomic data elucidates mouse organogenesis, *Nature Biotechnology*, 2022. Webpage: https://www.nature.com/articles/s41587-021-01006-2

`plotUmapFromWSIR` *plotUmapFromWSIR*

Description

A function to plot a UMAP generated on the low-dimensional embedding of the gene expression data. The points are coloured by their value for the genes with highest (in absolute value) loading in a selected WSIR direction, by default WSIR1.

Usage

```
plotUmapFromWSIR(
  X,
  umap_coords,
  highest_genes = NULL,
  genes = NULL,
  n_genes,
  ...
)
```

Arguments

`X`	matrix containing normalised gene expression data including n cells and p genes, dimension n * p.

umap_coords	UMAP coordinates for each cell that is output of generateUmapFromWSIR function. The UMAP coordinates can be based on any dimension reduction method, e.g they could be the UMAP coordinates computed on the WSIR dimension reduction of the gene expression data, or on the PCs (principal components), or on any other low-dimensional matrix. Must be a matrix of dimension nrow(X) * 2.
highest_genes	output from findTopGenes function. Default is NULL so an error message can easily be thrown if genes and highest_genes are both not provided.
genes	vector with gene names (must all be in colnames(X)) you wish to show in the UMAP plot. The cells in those plots will be coloured by their expression values for the genes you provide here. Must provide either genes or highest_genes parameter (not both): provide genes if you want to visualise a few specific genes, provide highest_genes if you want to visualise the genes that are found to be the most important to the WSIR directions. Default is NULL so an error message can easily be thrown if genes and highest_genes are both not provided.
n_genes	integer for the number of genes you would like to show. Default is the number of unique genes in the highest_genes parameter or the number of genes in the (vector) parameter genes. Use this parameter if you want to show only a few of the most important genes (e.g select the top 4 with n_genes = 4).
...	additional parameters for ggplot functions, e.g size (for size of points).

Value

Grid of umap plots with n_genes number of plots. Each shows the cells in a UMAP generated on the low-dimensional gene expression data, coloured by their value for each of the genes found by top_genes.

Examples

```
data(MouseData)
wsir_obj <- wSIR(X = sample1_exprs,
    coords = sample1_coords,
    optim_params = FALSE,
    alpha = 4,
    slices = 6) # create wsir object
umap_coords <- generateUmapFromWSIR(wsir = wsir_obj)
top_genes_obj <- findTopGenes(wsir = wsir_obj, highest = 4)
umap_plot <- plotUmapFromWSIR(umap_coords = umap_coords,
    X = sample1_exprs,
    highest_genes = top_genes_obj,
    n_genes = 4)
umap_plot
```

---

projectWSIR *projectWSIR*

Description

function to project new gene expression data into low-dimensional space

Usage

```
projectWSIR(wsir, new_data)
```

Arguments

wsir	wsir object that is usually the output of wSIR function. If you want to project new data into low-dim space following a different DR method, at param wsir use a list with matrix of loadings in slot 2 (e.g PCA loadings) of dimension p * d
new_data	matrix of new gene expression data to project into low-dimensional space. Must have the same p columns as the columns in X argument used to generate wsir.

Value

matrix of low-dimensional representation of newdata gene expression data

Examples

```
data(MouseData)
wsir_obj <- wSIR(X = sample1_exprs,
    coords = sample1_coords,
    optim_params = FALSE,
    alpha = 4,
    slices = 6)
sample2_low_dim_exprs <- projectWSIR(wsir = wsir_obj,
    new_data = sample2_exprs)
```

---

runwSIR *runwSIR*

Description

Perform wSIR on cells, based on the expression data and a reducedDim in a SingleCellExperiment or SpatialExperiment object

Usage

```
runwSIR(x, name = "wSIR", scores_only = FALSE, ...)
```

Arguments

x	A numeric matrix of normalised gene expression data where rows are features and columns are cells. Alternatively, a SingleCellExperiment or SpatialExperiment containing such a matrix
name	string to specify the name to store the result in the reducedDims of the output. Default is "wSIR"
scores_only	logical whether only the wSIR scores should be calculated. If FALSE additional information about the wSIR model will be stored in the attributes of the object. Default FALSE.
...	arguments passing to `calculateWSIR`

Value

If x is matrix-like, a list containing wSIR scores, loadings, etc. If x is a SingleCellExperiment or SpatialExperiment, the same object is returned with an additional slot in `reducedDims(..., name)` corresponding to the wSIR scores matrix. If `scores_only = FALSE`, then the attributes of the wSIR scores contain the following elements:

- directions
- estd
- W
- evalues

Examples

```
data(MouseData)
library(SingleCellExperiment)
library(SpatialExperiment)

sce <- SingleCellExperiment(assays = list(logcounts = t(sample1_exprs)),
reducedDims = list(spatial = sample1_coords))

sce <- runwSIR(x = sce, dim_red = "spatial")

spe <- SpatialExperiment(assays = list(logcounts = t(sample1_exprs)),
spatialCoords = as.matrix(sample1_coords))

spe <- runwSIR(x = spe, spatialCoords = TRUE)
```

`sirCategorical` *sirCategorical*

Description

This function performs WSIR based on provided gene expression data, tile allocations, and weight matrix.

Usage

```
sirCategorical(X, Y, W = NULL, ...)
```

Arguments

X	matrix of normalised gene expression data for n genes across p cells.
Y	dataframe with 1 column named "coordinate" that is the tile allocation for each cell. There should be up to slices^2 unique tile IDs in this column.
W	Weight matrix created by createWeightMatrix. Entry (i,j) represents the spatial correlation level between tiles i and j. The diagonal values should be all 1. If not provided, SIR implementation will be used.
...	arguments passed to sirPCA

Value

list of outputs with 5 named slots. They are the same as the output of the wSIR function: this is the final step in the wSIR function.

`sirPCA` *sirPCA*

Description

This function performs eigendecomposition on `(X^H)^t %*% W %*% (X^H)`, where `X^H` is matrix of scaled slice means.

Usage

```
sirPCA(
  sliced_data,
  maxDirections = 50,
  W = diag(nrow(sliced_data)),
  varThreshold = 0.99
)
```

Arguments

`sliced_data`	matrix of scaled slice means
`maxDirections`	integer (default 50), number of directions we want in our final low-dimensional Z.
`W`	matrix of slice weights. Output of createWeightMatrix function.
`varThreshold`	numeric proportion of eigenvalues of variance in `t(X_H) %*% W %*% X_H` to retain. Default is 0.99. Select higher threshold to include more dimensions, lower threshold to include less dimensions.

Value

list containing: 1) "eigenvectors" matrix of eigenvectors of `(X^H)^t %*% W %*% (X^H)` 2) "d" integer, selected number of dimensions 3) "evalues" vector of eigenvalues of `(X^H)^t %*% W %*% (X^H)`

`slicerCategorical` *slicerCategorical*

Description

This function averages all columns in the X matrix for each category in the single-column Y dataframe. It is used to find the average position within each tile to create the weight matrix, as well as to create the tile means that are used in the eigendecomposition step of SIR/wSIR.

Usage

```
slicerCategorical(X, Y)
```

Arguments

X	matrix or dataframe containing the columns to average
Y	dataframe with a single column named 'coordinate'. This column contains the categorical variable that is the tile ID for each cell.

Value

dataframe containing the averages for each column within each category in Y. Dimension h * p, where h is the number of categories in Y and p is the number of columns in X.

---

spatialAllocator *spatialAllocator*

Description

This function allocates each cell to a tile based on a specified number of tiles.

Usage

```
spatialAllocator(coords, slices = 3)
```

Arguments

coords	dataframe contains the spatial position of each cell. Column names c("x", "y'). Must include row names as integer from 1 to nrow(coords). This is automatically included in the wSIR function prior to this function.
slices	integer the number of slices along each of the two spatial axes. The number of tiles will be slices^2 since there are two spatial axes. E.g set slices = 4 to use 4^2 = 16 tiles. Default value is 3.

Value

output by itself is a matrix containing slice belonging for each axis in long format. When used in lapply(coords_split, spatialAllocator, slices) as in wSIR the output is a dataframe with each cell's tile allocation in the "coordinate" column.

---

visualiseWSIRDirections *visualiseWSIRDirections*

Description

A function to easily visualise the low-dimensional gene expression data. This function plots each cell at its true spatial coordinates, coloured by their values for WSIR1 / WSIR2 / ... . The plots give an intuition about what biological signals are contained in the WSIR directions.

Usage

```
visualiseWSIRDirections(
  coords,
  wsir,
  dirs = 6,
  mincol = "blue",
  maxcol = "red"
)
```

Arguments

coords	dataframe containing spatial positions of n cells in 2D space. Dimension n * 2. Column names must be c("x", "y").
wsir	wsir object as output of wSIR function. To analyse a different DR method, ensure the slot named 'directions' contains the loadings as a matrix with the gene names as the rownames. Must have used the same coords parameter as in coords parameter for this function.
dirs	integer for how many of the low-dimensional directions you would like to visualise. Recommend no more than 10 for ease of visualisation. Default is 6.
mincol	String for the colour of low values of low-dimensional directions. Personal choice for user, default is "blue".
maxcol	String for the colour of high values of low-dimensional directions. Personal choice for user, default is "red".

Value

Grid of plots with dirs number of plots. Each shows the cells at their spatial positions coloured by their value for each of the first 'dirs' WSIR directions.

Examples

```
data(MouseData)
wsir_obj <- wSIR(X = sample1_exprs,
    coords = sample1_coords,
    optim_params = FALSE,
    alpha = 4,
    slices = 6) # create wsir object
vis_obj <- visualiseWSIRDirections(coords = sample1_coords,
wsir = wsir_obj, dirs = 8) # create visualisations
vis_obj
```

---

wSIR *wSIR*

Description

A function to perform supervised dimension reduction of a gene expression matrix using coordinates dataframe as response value. Incorporates weighting mechanism into SIR algorithm to generate low-dimensional representation of the data and allow for projection of new single-cell gene expression data into low-dimensional space.

Usage

```
wSIR(
  X,
  coords,
  optim_params = FALSE,
  optim_alpha = c(0, 1, 2, 4, 8, 12),
  optim_slices = c(3, 5, 7, 10, 15, 20),
  metric = "DC",
  n_rep = 50,
  verbose = FALSE,
  ...
)
```

Arguments

X	matrix containing normalised gene expression data including n cells and p genes, dimension n * p.
coords	dataframe containing spatial positions of n cells in 2D space. Dimension n * 2. Column names must be c("x", "y").
optim_params	logical default FALSE. If you would like wSIR to automatically optimise parameters slices and alpha based on either distance correlation or correlation of distances as evaluation metric. If your downstream task is quite different to either of those metrics, then we suggest you tune those two parameters yourself using your specific task and evaluation metric. In that case, determine your optimal slices and alpha values and then use them in the relevant function, then setting optim_params = FALSE.
optim_alpha	If you have optim_params = TRUE, then this is the values of alpha to optimise over in wSIR. 0 gives Sliced Inverse Regression (SIR) implementation, and larger values represent stronger spatial correlation. Suggest to use integers for interpretability, but can use non-integers. Values must be non-negative.
optim_slices	If you have optim_params = TRUE, then this is the values of slices to optimise over in wSIR. Suggest maximum value in the vector to be no more than around $\sqrt{n/20}$, as this upper bound ensures an average of at least 10 cells per tile in the training set.
metric	If optim_params = TRUE, this is the evaluation metric to use for parameter tuning. String, either "DC" to use distance correlation or "CD" to use correlation of distances. Default is "DC".
n_rep	If optim_params = TRUE, this is the integer for the number of train/test splits of the data to perform during optimisation of parameters slices and alpha.
verbose	logical (default FALSE) whether progress messages should be printed
...	arguments passing to `exploreWSIRParams` and `wSIRSpecifiedParams`

Value

wSIR object which is a list containing 5 (named) positions.

1. scores matrix containing low-dimensional representation of X from wSIR algorithm. Dimension `n * d`, where `d` is chosen to include at least varThreshold proportion of variance.
2. directions matrix containing wSIR directions, dimension `p * d`. Use this to project new data into low-dimensional space via X_new %*% directions.

3. estd integer stating how many dimensions in the computed low-dimensional space are needed to account for varThreshold proportion of variance. Same as number of dimensions in scores.
4. W matrix weight matrix from cells_weight_matrix2 function
5. evalues vector containing p eigenvalues of `t(X_H) %*% W %*% X_H`. varThreshold parameter works on these evalues, such that e.g the first j directions are included if the sum of the first j evalues equals 0.95% of the sum of all evalues.

Examples

```
data(MouseData)
wsir_obj <- wSIR(X = sample1_exprs,
    coords = sample1_coords,
    optim_params = TRUE,
    optim_alpha = c(0,2,4),
    optim_slices = c(3,6,10),
    metric = "DC",
    n_rep = 1) # create wsir object
```

`wSIROptimisation` *wSIROptimisation*

Description

This function is used to calculate the validation metric on which the best tuning parameters are selected

Usage

```
wSIROptimisation(
  exprs_train,
  coords_train,
  exprs_test,
  coords_test,
  samples_train,
  slices,
  alpha,
  eval_metrics = c("CD", "DC", "ncol"),
  ...
)
```

Arguments

`exprs_train`	matrix of normalised gene expression on the training data.
`coords_train`	dataframe containing spatial positions of cells in 2D space on the validation data. Dimension n * 2. Column names must be c("x", "y").
`exprs_test`	matrix of normalised gene expression on the validation data.
`samples_train`	sample ID of each cell in the training data. In total, must have length equal to the number of cells. For example, if your dataset has 10000 cells, the first 5000 from sample 1 and the remaining 5000 from sample 2, you would write samples = c(rep(1, 5000), rep(2, 5000)) to specify that the first 5000 cells are sample

1 and the remaining are sample 2. Default is that all cells are from sample 1. Sample IDs can be of any format: for the previous example, you could write samples = c(rep("sample 1", 5000), rep("sample 2", 5000)), and the result would be the same.

`slices` integer for number of slices on each axis of tissue. For example, slices = 4 creates 4 slices along each spatial axis, yielding 4^2 = 16 tiles. Default is 8, suggested minimum of 3. Suggest to tune this parameter using exploreWSIRParams() function.

`alpha` integer to specify desired strength of spatial correlation. alpha = 0 gives SIR implementation. Larger values give stronger spatial correlations. Default value is 4, at which weight matrix value for a pair of tiles is inversely proportional to the physical distance between them. Suggest to tune this parameter using exploreWSIRParams() function.

`eval_metrics` evaluation metrics to use for parameter tuning. String, options are any or all of: "DC" to use distance correlation; "CD" to use correlation of distances; "ncol" to use number of columns in low-dimensional embedding. Default is all three, specified by metrics = c("DC", "CD", "ncol").

`...` arguments passed to wSIRSpecifiedParams

Value

Average metric value for the selected metric(s) over each train/test split.

`wSIRSpecifiedParams` *wSIR_specified_params*

Description

wSIR function for specified parameters alpha, slices. To be used internally in wSIROptimisation and wSIR functions, each for specific values of alpha and slices.

Usage

```
wSIRSpecifiedParams(
  X,
  coords,
  group = NULL,
  samples = rep(1, nrow(coords)),
  slices = 8,
  alpha = 4,
  ...
)
```

Arguments

`X` matrix containing normalised gene expression data including n cells and p genes, dimension `n * p`.

`coords` dataframe containing spatial positions of n cells in 2D space. Dimension `n * 2`. Column names must be c("x", "y").

group	a factor indicating group level information for cells within and across samples (e.g. cell type).
samples	sample ID of each cell. In total, must have length equal to the number of cells. For example, if your dataset has 10000 cells, the first 5000 from sample 1 and the remaining 5000 from sample 2, you would write samples = c(rep(1, 5000), rep(2, 5000)) to specify that the first 5000 cells are sample 1 and the remaining are sample 2. Default is that all cells are from sample 1. Sample IDs can be of any format: for the previous example, you could write samples = c(rep("sample 1", 5000), rep("sample 2", 5000)), and the result would be the same.
slices	integer for number of slices on each axis of tissue. For example, slices = 4 creates 4 slices along each spatial axis, yielding 4^2 = 16 tiles. Default is 8, suggested minimum of 3. Suggest to tune this parameter.
alpha	integer to specify desired strength of spatial correlation. Suggest to tune this parameter on testing dataset among e.g values c(0,2,4,8). alpha = 0 gives SIR implementation. Larger values give stronger spatial correlations.
...	arguments passed to sirCategorical

Value

wSIR object which is a list containing 5 (named) positions. 1) scores matrix containing low-dimensional representation of X from wSIR algorithm. Dimension `n * d`, where `d` is chosen to include at least varThreshold proportion of variance. 2) directions matrix containing WSIR directions, dimension `p * d`. Use this to project new data into low-dimensional space via X_new %*% directions. 3) estd integer stating how many dimensions in the computed low-dimensional space are needed to account for varThreshold proportion of variance. Same as number of dimensions in scores. 4) W matrix weight matrix from createWeightMatrix function 5) evalues vector containing p eigenvalues of `t(X_H) %*% W %*% X_H`. varThreshold parameter works on these evalues, such that e.g the first j directions are included if the sum of the first j evalues equals 0.95% of the sum of all evalues.

Index