

Package: PhiSpace (via r-universe)

October 13, 2026

Title Continuous cell state annotation for single-cell and spatial omics data

Version 0.99.0

Description Continuous phenotyping of bulk, single-cell and spatial omics data using partial least squares regression. The methods are described in Mao et al. (2025) [<doi:10.1186/s13059-025-03755-8>](https://doi.org/10.1186/s13059-025-03755-8) and Mao et al. (2026) [<doi:10.1016/j.crmeth.2026.101483>](https://doi.org/10.1016/j.crmeth.2026.101483).

License AGPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.5.0), SummarizedExperiment, SingleCellExperiment, SpatialExperiment

Imports Matrix, methods, BiocParallel, ComplexHeatmap, dplyr, ggplot2, ggpubr, irlba, magrittr, plyr, stats, scan, FNN, S4Vectors, kerndwd, umap, withr

Suggests AnnotationDbi, BiocStyle, celldex, cluster, knitr, org.Mm.eg.db, pls, rmarkdown, scRNAseq, scuttle, STexampleData, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

biocViews Software, SingleCell, Spatial, Sequencing, Transcriptomics, GeneExpression, CellBiology, Classification, Regression, DimensionReduction, Clustering

URL <https://mao-lab-org.github.io/PhiSpace/>,
<https://github.com/mao-lab-org/PhiSpace>

BugReports <https://github.com/mao-lab-org/PhiSpace/issues>

Config/pak/sysreqs cmake libglpk-dev make libmagick++-dev gsfonts libicu-dev libpng-dev libxml2-dev libssl-dev perl python3 zlib1g-dev

Repository <https://biocstaging.r-universe.dev>

Date/Publication 2026-10-13 02:36:28 UTC

RemoteUrl <https://github.com/BiocStaging/PhiSpace>

RemoteRef HEAD

RemoteSha 684adb8ae077922f2406cbee6a43ffe17908e3a2

Contents

as.sparse.matrix	3
basicPlotFormat	3
cellTypeCorMat	5
classErr	6
CLRnorm	7
clusterPhiSpace	7
codeY	11
codeY_vec	11
compute_kernel_weights	12
computeUMAP	12
CVTune_ncomp	13
CVTune_nfeat	14
doubleCent	16
findNiches	16
getErr	18
getErr_nfeat	19
getLabels	20
getPC	21
KeepCommonGenes	22
logNormTransf	23
logTransf	24
mvr	25
normPhiScores	26
phenotype	27
PhiSpace	27
PhiSpaceR_1ref	30
piScore	33
plot.PhiSpaceClustering	35
plotPhiSpaceHeatMap	35
predict.PhiSpaceModel	37
print.PhiSpaceClustering	38
pseudoBulk	39
rankFeatures	40
RankTransf	42
reNameCols	43
RTassay	43
saveCellTypeMaps	44
Score2Prob	46
scoreCells	47

Usage

```

basicPlotFormat(
  p,
  theme_overall = "bw",
  base_size = 8,
  legend.position = "top",
  x.text.blank = FALSE,
  y.text.blank = FALSE,
  axis.title.bold = TRUE,
  legend.title.bold = TRUE,
  axis.title.x.blank = FALSE,
  axis.title.y.blank = FALSE,
  legend.title.blank = FALSE,
  legend.spacing.x = 0,
  legend.key.spacing = 0,
  legend.key.spacing.y = 0,
  legend.box.margin = 0,
  legend.box.spacing = 0,
  legend.key.size = 8
)

```

Arguments

<code>p</code>	A ggplot2 object.
<code>theme_overall</code>	Theme of p.
<code>base_size</code>	base_size of theme.
<code>legend.position</code>	Legend position.
<code>x.text.blank</code>	Logical. Set x axis text to blank.
<code>y.text.blank</code>	Logical. Set y axis text to blank.
<code>axis.title.bold</code>	Logical. Set title to be bold.
<code>legend.title.bold</code>	Logical. Set legend title to be bold.
<code>axis.title.x.blank</code>	Logical. Set x axis title to be bold.
<code>axis.title.y.blank</code>	Logical. Set y axis title to be bold.
<code>legend.title.blank</code>	Logical. Set legend title to be blank.
<code>legend.spacing.x</code>	Legend spacing.
<code>legend.key.spacing</code>	Legend key spacing.
<code>legend.key.spacing.y</code>	Legend key spacing on vertical direction.

`legend.box.margin`
Legend box margin.

`legend.box.spacing`
Legend box spacing.,

`legend.key.size`
Legend key size.

Value

A ggplot2 object.

<code>cellTypeCorMat</code>	<i>Create cell type correlation matrices</i>
-----------------------------	--

Description

A generic function to compute a correlation matrix for different input types, including `matrix`, `data.frame` and `SingleCellExperiment` objects.

Usage

```
cellTypeCorMat(input, mask = NULL, reducedDimName = "PhiSpace", groupBy = NULL)

## S4 method for signature 'matrix'
cellTypeCorMat(input, mask = NULL, reducedDimName = "PhiSpace", groupBy = NULL)

## S4 method for signature 'data.frame'
cellTypeCorMat(input, mask = NULL, reducedDimName = "PhiSpace", groupBy = NULL)

## S4 method for signature 'SingleCellExperiment'
cellTypeCorMat(input, mask = NULL, reducedDimName = "PhiSpace", groupBy = NULL)
```

Arguments

`input` A `matrix` containing numeric data for correlation computation.

`mask` A zero-one matrix with the same number of rows as the input. Optional. If provided, each column of `mask` defines a subset of rows used to compute a separate correlation matrix.

`reducedDimName` A character string specifying the name of the reduced dimension to use for `SingleCellExperiment` objects. Defaults to "PhiSpace".

`groupBy` A column of `colData(input)` when input is an SCE object. If provided, mask will be defined according to `colData(input)[,groupBy]`.

Details

This method calculates the correlation matrix directly for the input `matrix`.

This method first converts the `data.frame` to a `matrix` and then calculates the correlation matrix.

This method uses the reduced dimension "PhiSpace" of the `SingleCellExperiment` object to compute the correlation matrix. If "PhiSpace" is not available, an error is raised.

Value

If `mask` is provided, returns a list of correlation matrices (one per column of `mask`). Otherwise, returns a single correlation matrix.

Examples

```
set.seed(1)
scores <- matrix(rnorm(200), 50, 4, dimnames = list(NULL, LETTERS[1:4]))
cellTypeCorMat(scores)
# One correlation matrix per group of rows.
mask <- cbind(first = rep(c(1, 0), each = 25), second = rep(c(0, 1), each = 25))
names(cellTypeCorMat(scores, mask = mask))
```

`classErr`

Calculate classification errors.

Description

Calculate classification errors.

Usage

```
classErr(classQuery, classOriginal, labPerSample = NULL)
```

Arguments

`classQuery` Character vector.
`classOriginal` Character vector.
`labPerSample` Integer.

Value

A list containing:

`err` Numeric vector: the overall error rate and the balanced error rate (the mean of the per-class error rates).
`byClassErrors` Named numeric vector of per-class error rates.

CLRnorm	<i>Center log ratio normalisation</i>
---------	---------------------------------------

Description

Center log ratio normalisation

Usage

```
CLRnorm(sce, assayName = "counts", targetAssay = "data", offset = 1)
```

Arguments

sce	SCE object
assayName	Assay to normalise
targetAssay	Name of normalised assay
offset	Offset value added to raw counts

Value

SCE object

Examples

```
sce <- scuttle::mockSCE(ncells = 20, ngenes = 10)
sce <- CLRnorm(sce)
assay(sce, "data")[1:3, 1:3]
```

clusterPhiSpace	<i>K-means clustering on PhiSpace principal components</i>
-----------------	--

Description

Performs k-means clustering on principal components derived from PhiSpace scores or any other matrix-like data. This is useful for identifying spatial niches or clusters based on cell state compositions or gene expression patterns.

Usage

```
clusterPhiSpace(
  x,
  k = NULL,
  k_range = NULL,
  select_k_method = c("silhouette", "elbow"),
  ncomp = NULL,
  reducedDimName = "PhiSpace",
```

```

    use_assay = NULL,
    nstart = 20,
    iter.max = 500,
    algorithm = c("Lloyd", "Hartigan-Wong", "MacQueen", "Forgy"),
    center = TRUE,
    scale = FALSE,
    seed = NULL,
    silhouette_max_cells = Inf,
    return_pca = TRUE,
    store_in_colData = FALSE,
    cluster_name = "PhiClust"
  )

```

Arguments

x	Either a SpatialExperiment/SingleCellExperiment object containing PhiSpace scores in reducedDim, OR a matrix-like object (matrix, sparse matrix, data frame) with features in columns and observations in rows.
k	Integer specifying the number of clusters. Either provide k or k_range but not both.
k_range	Integer vector of length 2 specifying range of k values to test (e.g., c(5, 15)). Will use elbow method or silhouette to select optimal k. Either provide k or k_range but not both.
select_k_method	Character string specifying method to select optimal k when k_range is provided. Options: "elbow" (total within-cluster SS) or "silhouette" (average silhouette width). Default is "silhouette".
ncomp	Integer specifying number of principal components to use for clustering. If NULL (default), uses min(30, nfeatures - 1). If "all", uses nfeatures - 1 components.
reducedDimName	Character string specifying which reducedDim slot contains the PhiSpace scores. Only used when x is a SingleCellExperiment or SpatialExperiment. Default is "PhiSpace".
use_assay	Character string specifying which assay to use if extracting data from a SingleCellExperiment/SpatialExperiment object instead of using reducedDim. If NULL (default), uses reducedDim specified by reducedDimName. Common options: "logcounts", "counts", "normcounts".
nstart	Integer specifying number of random starts for k-means. Default is 20.
iter.max	Integer specifying maximum number of iterations. Default is 500.
algorithm	Character string specifying k-means algorithm. Options are "Hartigan-Wong", "Lloyd", "Forgy", "MacQueen". Default is "Lloyd".
center	Logical indicating whether to center data before PCA. Default is TRUE.
scale	Logical indicating whether to scale data before PCA. Default is FALSE.

seed	Integer seed for reproducibility. Default is NULL (no seed set). The function sets the seed with <code>withr::local_seed()</code> , so the session's random-number state is restored on exit.
silhouette_max_cells	Number or Inf . With the silhouette method, the silhouette widths need a distance matrix whose size grows with the square of the number of cells (about 0.4 GB for 10,000 cells and 40 GB for 100,000). If set, and there are more cells than this, the widths are computed on a random subsample of this many cells; k-means still uses every cell. Default is Inf : every cell is used.
return_pca	Logical indicating whether to return PCA results. Default is TRUE.
store_in_colData	Logical indicating whether to store cluster assignments in the colData of the input object (only applicable when x is an SCE/SPE). Default is FALSE.
cluster_name	Character string specifying the column name for cluster assignments if store_in_colData is TRUE. Default is "PhiClust".

Details

This function implements a common workflow for spatial clustering:

1. Extract data (from reducedDim, assay, or use directly if matrix)
2. Perform PCA to reduce dimensionality
3. Select top principal components
4. Apply k-means clustering

The function can either:

- Use a fixed k value (specify k parameter)
- Automatically select k from a range (specify k_range parameter)

When k_range is provided, the function tests all k values in the range and selects the optimal k using either:

- **Silhouette method** (default): Maximizes average silhouette width
- **Elbow method**: Identifies elbow point in total within-cluster sum of squares

Value

A list with class "PhiSpaceClustering" containing:

clusters	Factor vector of cluster assignments
cluster_centers	Matrix of cluster centers in PC space
kmeans_result	Full kmeans object from stats::kmeans
pca_result	PCA results (if return_pca = TRUE)
pc_scores	Matrix of PC scores used for clustering

<code>optimal_k</code>	Selected k value (relevant when <code>k_range</code> is used)
<code>k_selection</code>	List with selection metrics (if <code>k_range</code> was used)
<code>parameters</code>	List of parameters used
<code>spe</code>	Updated object (if <code>store_in_colData = TRUE</code> and <code>x</code> is SCE/SPE)

Input Types

The function accepts multiple input types:

- **SingleCellExperiment/SpatialExperiment:** Uses `reducedDim` (default) or `assay`
- **Matrix:** Standard R matrix with observations in rows
- **Sparse matrix:** `dgCMatrix` or similar sparse formats
- **Data frame:** Coerced to matrix

PCA Parameters

The number of PCs to use affects clustering resolution:

- Fewer PCs (e.g., 10-15): Broader, more general clusters
- More PCs (e.g., 30-50): Finer, more specific clusters
- Default (30): Good balance for most applications

K-means Algorithm

- **Lloyd** (default): Standard algorithm, good balance of speed and quality
- **Hartigan-Wong:** Often better results but slower
- **MacQueen:** Faster but may converge to local optima
- **Forgy:** Similar to Lloyd

Examples

```
set.seed(1)
scores <- matrix(rnorm(150 * 5), 150, 5, dimnames = list(NULL, LETTERS[1:5]))
scores[1:50, 1] <- scores[1:50, 1] + 3
scores[51:100, 2] <- scores[51:100, 2] + 3
res <- clusterPhiSpace(scores, k = 3, seed = 1)
res
table(res$clusters)
# Choose k from a range with the silhouette method.
res <- clusterPhiSpace(scores, k_range = 2:5, seed = 1)
summary(res)
plot(res, type = "silhouette")
```

codeY	<i>Convert discrete annotations to dummy variable matrices.</i>
-------	---

Description

Convert discrete annotations to dummy variable matrices.

Usage

```
codeY(sce, phenotypes, method = c("-1,1", "0,1"))
```

Arguments

sce	SingleCellExperiment object.
phenotypes	Which phenotype annotations (e.g. cell types) to convert.
method	Which coding to use, either (-1,1) or (0,1).

Value

YY	A dummy variable matrix encoding discrete annotation
----	--

Examples

```
sce <- scuttle::mockSCE(ncells = 20, ngenes = 10)
YY <- codeY(sce, phenotypes = c("Cell_Cycle", "Treatment"))
head(YY)
```

codeY_vec	<i>Convert discrete annotations to dummy variable matrices.</i>
-----------	---

Description

Convert discrete annotations to dummy variable matrices.

Usage

```
codeY_vec(vec, rowNames = NULL, method = c("-1,1", "0,1"))
```

Arguments

vec	Factor containing class labels.
rowNames	Optional. Row names for output dummy matrix (eg cell barcode).
method	Which coding to use, either (-1,1) or (0,1).

Value

YY A dummy variable matrix encoding discrete annotation

Examples

```
codeY_vec(c("B", "T", "T", "NK"), rowNames = paste0("cell", 1:4))
codeY_vec(c("B", "T", "T", "NK"), method = "0,1")
```

```
compute_kernel_weights
```

Compute kernel weights based on distances

Description

Compute kernel weights based on distances

Usage

```
compute_kernel_weights(distances, kernel, sigma = NULL)
```

Arguments

distances Matrix of distances (cells x neighbors)
kernel Kernel type: "gaussian", "uniform", or "linear"
sigma Bandwidth parameter (auto-computed if NULL)

Value

Matrix of weights

```
computeUMAP
```

Compute UMAP.

Description

Compute UMAP.

Usage

```
computeUMAP(
  dat,
  computPC = TRUE,
  ncomp = 30,
  center = TRUE,
  scale = FALSE,
  config = NULL
)
```

Arguments

<code>dat</code>	Input data.
<code>computPC</code>	Whether to compute PC first.
<code>ncomp</code>	Number of PC components
<code>center</code>	Center or not.
<code>scale</code>	Scale or not.
<code>config</code>	Configuration of UMAP (see <code>umap</code> package).

Value

UMAP coordinates.

Examples

```
set.seed(1)
X <- matrix(rnorm(100 * 10), 100, 10, dimnames = list(paste0("cell", 1:100), NULL))
emb <- computeUMAP(X, ncomp = 5)
dim(emb)
```

CVTune_ncomp	<i>Tune ncomp using cross-validation.</i>
--------------	---

Description

Tune ncomp using cross-validation.

Usage

```
CVTune_ncomp(
  reference,
  assayName = "logcounts",
  phenotypes = NULL,
  YY = NULL,
  center = TRUE,
  scale = FALSE,
  ncompLimits = c(1, 100),
  gridSize = 20,
  ncompGrid = NULL,
  selectedFeat = NULL,
  Kfolds = 5,
  regMethod = "PCA",
  labelCode = "-1,1",
  Ncores = 1,
  seed = 5202056,
  ncomp = NULL
)
```

Arguments

<code>reference</code>	SingleCellExperiment object.
<code>assayName</code>	Character.
<code>phenotypes</code>	Character.
<code>YY</code>	Matrix.
<code>center</code>	Logic.
<code>scale</code>	Logic.
<code>ncompLimits</code>	Vector.
<code>gridSize</code>	Integer.
<code>ncompGrid</code>	Vector.
<code>selectedFeat</code>	Vector.
<code>Kfolds</code>	Integer.
<code>regMethod</code>	Character.
<code>labelCode</code>	Character.
<code>Ncores</code>	Integer.
<code>seed</code>	Integer. The function sets the seed with <code>withr::local_seed()</code> , so the session's random-number state is restored on exit.
<code>ncomp</code>	Chosen number of components, or NULL to choose after the cross-validation: in an interactive session the function plots the loss and asks; otherwise it takes the grid value with the lowest loss.

Value

A list.

CVTune_nfeat	<i>Tune nfeat using cross-validation.</i>
--------------	---

Description

Tune nfeat using cross-validation.

Usage

```
CVTune_nfeat(
  reference,
  assayName = "logcounts",
  phenotypes = NULL,
  YY = NULL,
  ncomp = NULL,
  impScores = NULL,
  center = TRUE,
```

```

    scale = FALSE,
    nfeatLimits = c(10, 15000),
    gridSize = 30,
    nfeatV = NULL,
    Kfolds = 5,
    regMethod = "PCA",
    labelCode = "-1,1",
    seed = 5202056,
    Ncores = 1,
    nfeat = NULL
  )

```

Arguments

<code>reference</code>	SingleCellExperiment object.
<code>assayName</code>	Character.
<code>phenotypes</code>	Vector.
<code>YY</code>	Matrix.
<code>ncomp</code>	Integer.
<code>impScores</code>	Matrix.
<code>center</code>	Logic.
<code>scale</code>	Logic.
<code>nfeatLimits</code>	Vector.
<code>gridSize</code>	Integer.
<code>nfeatV</code>	Vector.
<code>Kfolds</code>	Integer.
<code>regMethod</code>	Character.
<code>labelCode</code>	Character.
<code>seed</code>	Integer. The function sets the seed with <code>withr::local_seed()</code> , so the session's random-number state is restored on exit.
<code>Ncores</code>	Integer.
<code>nfeat</code>	Chosen number of features per phenotype, or <code>NULL</code> to choose after the cross-validation. Only an interactive session can choose: the function plots the loss and asks.

Value

A list.

<code>doubleCent</code>	<i>Double center a matrix by column and row means, resulting in a new matrix with zero row and column means.</i>
-------------------------	--

Description

Double center a matrix by column and row means, resulting in a new matrix with zero row and column means.

Usage

```
doubleCent(X)
```

Arguments

`X` A matrix or an object convertible to one.

Value

A double-centred matrix.

Examples

```
X <- matrix(1:12, 3, 4)
dc <- doubleCent(X)
round(rowMeans(dc), 10)
round(colMeans(dc), 10)
```

<code>findNiches</code>	<i>Find spatial niches by clustering PhiSpace scores</i>
-------------------------	--

Description

Identifies spatial niches by performing k-means clustering on PhiSpace cell type scores. Optionally performs PCA before clustering to reduce dimensionality and improve clustering performance. Can test multiple numbers of niches and return all results for comparison.

Usage

```
findNiches(
  spe,
  reducedDimName = "PhiSpace",
  n_niches,
  use_pca = TRUE,
  ncomp = NULL,
  center = TRUE,
  scale = FALSE,
```

```

    kmeans_iter = 500L,
    kmeans_nstart = 20L,
    kmeans_algorithm = "Lloyd",
    seed = 94863,
    store_pca = FALSE,
    pca_name = "PhiSpace_PCA",
    verbose = TRUE
  )

```

Arguments

spe	SpatialExperiment or SingleCellExperiment object containing PhiSpace scores.
reducedDimName	Name of the reducedDim containing PhiSpace scores (default: "PhiSpace").
n_niches	Integer or vector of integers. Number(s) of niches (clusters) to identify. If a vector is provided, multiple clustering results will be generated.
use_pca	Logical. Whether to perform PCA before clustering (default: TRUE).
ncomp	Integer. Number of principal components to use if use_pca = TRUE. If NULL, defaults to half the number of cell types in PhiSpace scores.
center	Logical. Whether to center data for PCA (default: TRUE).
scale	Logical. Whether to scale data for PCA (default: FALSE).
kmeans_iter	Integer. Maximum number of iterations for k-means (default: 500).
kmeans_nstart	Integer. Number of random starts for k-means (default: 20).
kmeans_algorithm	Character. Algorithm for k-means clustering (default: "Lloyd").
seed	Integer. Random seed for reproducibility (default: 94863). The function sets the seed with <code>withr::local_seed()</code> , so the session's random-number state is restored on exit.
store_pca	Logical. Whether to store PCA results in the object (default: FALSE).
pca_name	Character. Name for stored PCA results if store_pca = TRUE (default: "PhiSpace_PCA").
verbose	Logical. Whether to report progress and niche sizes with <code>message()</code> (default: TRUE). Use <code>verbose = FALSE</code> or <code>suppressMessages()</code> to silence them.

Details

This function identifies spatial niches by clustering cells/spots based on their PhiSpace cell type composition scores. The workflow consists of:

1. Extract PhiSpace scores from the specified reducedDim
2. Optionally perform PCA to reduce dimensionality and focus on major patterns
3. Apply k-means clustering to identify niches (for each k in n_niches)

4. Store niche assignments in colData

PCA is recommended when you have many cell types (>20) as it can improve clustering by focusing on the main axes of variation in cell type composition.

When multiple values of `n_niches` are provided, the function will test each one and store all results, allowing you to compare different numbers of niches and choose the most appropriate one for your analysis.

Value

The input object with niche assignments added to `colData`. If a single `n_niches` is provided, results are stored as `"spatial_niches"`. If multiple `n_niches` are provided, results are stored as `"spatial_niches_k7"`, `"spatial_niches_k8"`, etc. If `store_pca = TRUE`, PCA results are also stored in `reducedDims`.

Examples

```
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
query <- logNormTransf(scuttle::mockSCE(ncells = 100, ngenes = 300))
spe <- SpatialExperiment::SpatialExperiment(
  assays = list(logcounts = logcounts(query)),
  spatialCoords = cbind(x = runif(100), y = runif(100))
)
spe <- PhiSpace(ref, spe, phenotypes = "Cell_Cycle", refAssay = "logcounts",
  ncomp = 3)
spe <- findNiches(spe, n_niches = 3, ncomp = 3, verbose = FALSE)
table(spe$spatial_niches)
```

`getErr`

Compute different errors for cross-validation.

Description

Compute different errors for cross-validation.

Usage

```
getErr(
  ncompGrid,
  XXtrain,
  YYtrain,
  XXtest,
  YYtest,
  center = TRUE,
  scale = FALSE,
  selectedFeat = NULL,
  regMethod = "PLS",
```

```

    assayName = "rank",
    loss = c("Frobenius", "exponential")
  )

```

Arguments

ncompGrid	Vector.
XXtrain	Matrix.
YYtrain	Matrix.
XXtest	Matrix.
YYtest	Matrix.
center	Logic.
scale	Logic.
selectedFeat	Vector.
regMethod	Character.
assayName	Character.
loss	Type of loss function.

Value

A list

getErr_nfeat	<i>Calculate errors for the cross-validation for selecting nfeat.</i>
--------------	---

Description

Calculate errors for the cross-validation for selecting nfeat.

Usage

```

getErr_nfeat(
  nfeatV,
  impScores,
  ncomp,
  XXtrain,
  YYtrain,
  XXtest,
  YYtest,
  center = TRUE,
  scale = FALSE,
  regMethod = "PLS",
  assayName = "logcounts",
  loss = c("Frobenius", "exponential")
)

```

Arguments

nfeatV	Vector.
impScores	Matrix.
ncomp	Integer.
XXtrain	Matrix.
YYtrain	Matrix.
XXtest	Matrix.
YYtest	Matrix.
center	Logic.
scale	Logic.
regMethod	Character.
assayName	Character.
loss	Type of loss function.

Value

A list.

getLabels	<i>Turn scores into labels</i>
------------------	--------------------------------

Description

Gives each row of a score matrix the name of the column with the highest score, or the names of the **labPerSample** highest-scoring columns. This function was called **getClass()** before PhiSpace 0.99.0; it was renamed because that name masked **methods::getClass()**.

Usage

```
getLabels(X, labPerSample = NULL)
```

Arguments

X	Matrix of scores, one row per sample (for example per cell) and one named column per label.
labPerSample	Positive integer: the number of labels per row. The default NULL gives one label.

Value

With one label per row, a character vector of column names, one per row of **X**. With more, a character matrix with one row per row of **X**, labels in decreasing order of score.

Examples

```
set.seed(1)
scores <- matrix(rnorm(20), 5, 4, dimnames = list(NULL, LETTERS[1:4]))
getLabels(scores)
getLabels(scores, labPerSample = 2)
```

getPC

Principal component analysis (PCA) based on partial singular value decomposition (SVD).

Description

getPC() uses irlba's partial SVD. When `ncomp` is at least half of the smaller dimension of `X`, where irlba can be inaccurate, or when the smaller dimension is at most 100, it uses a full SVD of a dense copy of `X` instead. On such small matrices irlba may not converge, and its result then depends on its random start; the full SVD does not.

Usage

```
getPC(X, ncomp, center = TRUE, scale = FALSE, sparse = FALSE)
```

Arguments

<code>X</code>	Matrix.
<code>ncomp</code>	Integer.
<code>center</code>	Logic.
<code>scale</code>	Logic.
<code>sparse</code>	Use sparse matrix or not.

Value

A list containing

<code>scores</code>	Score matrix for <code>X</code> .
<code>loadings</code>	Laoding matrix for <code>X</code> .
<code>sdev</code>	Standard deviations of the principal components.
<code>totVar</code>	Total variance of <code>X</code> after the requested centring and scaling.
<code>props</code>	Proportion of <code>totVar</code> explained by each component.
<code>accuProps</code>	Cumulative sum of <code>props</code> .
<code>ncomp</code>	Number of components.
<code>Xmeans</code>	Column means used for centring, or <code>NULL</code> .
<code>Xscals</code>	Column standard deviations used for scaling, or <code>NULL</code> .

Examples

```

set.seed(1)
X <- matrix(rnorm(100 * 10), 100, 10)
pc <- getPC(X, ncomp = 3)
dim(pc$scores)
pc$props

```

KeepCommonGenes	<i>Keep common genes of two SingleCellExperiment objects.</i>
-----------------	---

Description

Keep common genes of two SingleCellExperiment objects.

Usage

```
KeepCommonGenes(reference, query, dots2hyphen = FALSE)
```

Arguments

reference	SingleCellExperiment object.
query	SingleCellExperiment object.
dots2hyphen	Logic.

Value

A list containing updated SingleCellExperiment objects with matching features.

Examples

```

ref <- scuttle::mockSCE(ncells = 20, ngenes = 10)
query <- scuttle::mockSCE(ncells = 20, ngenes = 10)
query <- query[3:10, ]
res <- KeepCommonGenes(ref, query)
nrow(res$reference)
nrow(res$query)

```

logNormTransf

Log-normalise counts with size factors

Description

Divides the counts of each cell by its size factor and takes $\log_2(x + 1)$. By default the size factors are the library sizes, centred to mean 1. This gives the same values as the defaults of `scuttle::logNormCounts()`, which Bioconductor 3.24 deprecates.

Usage

```
logNormTransf(
  sce,
  assayName = "counts",
  targetAssay = "logcounts",
  sizeFactors = NULL
)
```

Arguments

<code>sce</code>	A <code>SingleCellExperiment</code> or <code>SummarizedExperiment</code> object.
<code>assayName</code>	Name of the count assay.
<code>targetAssay</code>	Name of the new assay.
<code>sizeFactors</code>	Optional positive size factors, one per cell. The default <code>NULL</code> uses the library sizes (column sums of the counts).

Value

`sce` with the log-normalised assay `targetAssay` and the size factors, centred to mean 1, in `colData(sce)$sizeFactor`. A sparse or `DelayedArray` count matrix gives a `dgCMatrix`, a dense one a dense matrix.

See Also

[scrnTransf\(\)](#) for pooled size factors, [logTransf\(\)](#) for `log1p()` of counts or relative counts.

Examples

```
sce <- scuttle::mockSCE(ncells = 20, ngenes = 10)
sce <- logNormTransf(sce)
assay(sce, "logcounts")[1:3, 1:3]
```

logTransf	<i>Relative counts normalisation.</i>
-----------	---------------------------------------

Description

Do log1p normalisation by default.

Usage

```
logTransf(
  sce,
  scaleFactor = 10000,
  assayName = "counts",
  targetAssay = "log1p",
  use_log1p = TRUE,
  sparse = TRUE
)
```

Arguments

sce	SingleCellExperiment object.
scaleFactor	Scale factor to inflate relative counts.
assayName	Character.
targetAssay	Character.
use_log1p	Logical. Use shifted log (log1p) transform, using raw counts instead of relative counts.
sparse	Store the transformed assay as sparse matrix or not.

Value

A SingleCellExperiment object containing a new assay after relative counts normalisation.

Examples

```
sce <- scuttle::mockSCE(ncells = 20, ngenes = 10)
sce <- logTransf(sce, targetAssay = "log1p")
assay(sce, "log1p")[1:3, 1:3]
```

mvr	<i>Multivariate regression via principal component analysis (PCA) or partial least squares (PLS).</i>
------------	---

Description

Simplified version of `pls::mvr`, using computationally faster versions of PCA and PLS.

Usage

```
mvr(
  X,
  Y,
  ncomp,
  method = c("PCA", "PLS"),
  center = TRUE,
  sparse = FALSE,
  scale = FALSE,
  DRinfo = FALSE,
  keepComps = seq_len(ncomp)
)
```

Arguments

X	Matrix (dense or sparse).
Y	Matrix.
ncomp	Integer.
method	Character.
center	Logic.
sparse	Not used; kept for backward compatibility.
scale	Logic.
DRinfo	Logic. Whether to return dimension reduction information from PCA or PLS. Disable to save memory.
keepComps	Integer vector. Numbers of components for which to return regression coefficients. The default returns all, <code>1:ncomp</code> . Use <code>keepComps = ncomp</code> to return only the final coefficients and save memory.

Details

A sparse **X** (for example a `dgCMatrix`) stays sparse: centring and scaling are applied implicitly, without a dense copy of **X**. PLS uses the kernel algorithm of `pls::kernelpls.fit`.

Value

A list containing

coefficients Array of regression coefficients with dimensions features x responses x `length(keepComps)`. The third dimension is named "<k> comps", so `coefficients[, , paste(ncomp, "comps")]` selects the coefficients for `ncomp` components.

Xmeans

Ymeans

ncomp

method

Examples

```
set.seed(1)
X <- matrix(rnorm(100 * 20), 100, 20)
Y <- X[, 1:3] %*% matrix(rnorm(9), 3, 3) + matrix(rnorm(300, sd = 0.1), 100, 3)
fit <- mvr(X, Y, ncomp = 3, method = "PLS")
dim(fit$coefficients)
```

normPhiScores

Normalise PhiSpace phenotypic scores by row or by column.

Description

The by-row normalisation is the same as `SingleR:::trim_byLabel_and_normalize_scores`.

Usage

```
normPhiScores(X, method = c("col", "row"))
```

Arguments

X Matrix.

method Character.

Value

A matrix normalised by row or by column.

Examples

```
set.seed(1)
scores <- matrix(rnorm(20), 5, 4, dimnames = list(NULL, LETTERS[1:4]))
normPhiScores(scores)
normPhiScores(scores, method = "row")
```

phenotype	<i>Soft phenotyping of query assay.</i>
-----------	---

Description

Soft phenotyping of query assay.

Usage

```
phenotype(  
  phenoAssay,  
  atlas_re,  
  assayName = "rank",  
  scaleMethod = c("byQuery", "byRef")  
)
```

Arguments

phenoAssay	Matrix. Query assay to phenotype.
atlas_re	List.
assayName	Character.
scaleMethod	Character.

Value

A list containing

Yhat

Proj

PhiSpace	<i>PhiSpace continuous phenotyping</i>
----------	--

Description

PhiSpace annotates a query dataset or a list of query datasets, given an annotated bulk or single-cell RNA-seq references. PhiSpace can simultaneously model multiple layers of cell phenotypes, e.g. cell type and disease condtion.

Usage

```
PhiSpace(
  reference,
  query,
  phenotypes = NULL,
  response = NULL,
  refAssay = "rank",
  queryAssay = NULL,
  regMethod = c("PLS", "PCA"),
  reducedDimName = "PhiSpace",
  ncomp = NULL,
  nfeat = NULL,
  selectedFeat = NULL,
  center = TRUE,
  scale = FALSE,
  DRinfo = FALSE,
  storeUnNorm = FALSE,
  updateRef = FALSE,
  cellTypeThreshold = NULL,
  fallback = c("none", "scoreCells"),
  fallback_min_classes = 2,
  fallback_score = c("signature", "correlation")
)
```

Arguments

reference	The references. A <code>SingleCellExperiment</code> (SCE) object, a <code>PhiSpaceModel</code> from <code>trainPhiSpace()</code> , or a list of either (models and SCE objects can be mixed). Each SCE must contain an assay named by <code>refAssay</code> . A model is applied as it is: the training arguments (<code>phenotypes</code> , <code>response</code> , <code>refAssay</code> , <code>regMethod</code> , <code>ncomp</code> , <code>nfeat</code> , <code>selectedFeat</code> , <code>center</code> , <code>scale</code> , <code>DRinfo</code> , <code>cellTypeThreshold</code> and the fallback arguments) are used only for SCE references. If the queries lack some model genes, the model is restricted to the genes shared by all queries, as described in <code>predict.PhiSpaceModel()</code> .
query	The queries. An SCE object or a list of SCE object. Each must contain an assay named by <code>queryAssay</code> .
phenotypes	Which phenotypes (e.g. "cell type") to predict. If <code>NULL</code> , then have to specify <code>response</code> .
response	Named matrix. Rows correspond to cells (columns) in reference; columns correspond to phenotypes. If not <code>NULL</code> , then will override <code>phenotypes</code> .
refAssay	Character. Which assay in reference to use to train <code>PhiSpace</code> .
queryAssay	Character. Which assay in query to use to predict. If <code>NULL</code> , <code>refAssay</code> for an SCE reference and the model's training assay for a <code>PhiSpaceModel</code> .
regMethod	Character. Regression method: one of "PLS" and "PCA".
reducedDimName	Name of reducedDim layer to store <code>PhiSpace</code> scores, i.e. <code>PhiSpace</code> scores

	will be stored in <code>reducedDim(query, reducedDimName)</code> ; default is "PhiSpace".
<code>ncomp</code>	Integer. Number of components. If <code>NULL</code> , will use the default, i.e. same as the total number of phenotypes.
<code>nfeat</code>	Integer. Number of features to choose to predict each phenotype. See details.
<code>selectedFeat</code>	Character. Alternatively, can provide a vector of pre-selected features.
<code>center</code>	Logic. Whether to perform centering. See details.
<code>scale</code>	Logic. Whether to perform scaling. See details.
<code>DRinfo</code>	Logic. Whether to return dimension reduction information from PCA or PLS. By default disabled to save memory.
<code>storeUnNorm</code>	Store unnormalised raw PhiSpace scores or not. Default is <code>FALSE</code> .
<code>updateRef</code>	Update reference (store reference PhiSpace scores in reference sce object) or not.
<code>cellTypeThreshold</code>	Integer or <code>NULL</code> . If a positive integer, cell types with fewer than this many cells in the reference will be removed before model fitting. Only used when <code>phenotypes</code> is provided. Default is <code>NULL</code> (no filtering).
<code>fallback</code>	Character. Optional fallback for references with too few classes. " <code>none</code> " preserves standard PhiSpace behaviour. " <code>scoreCells</code> " calls <code>scoreCells()</code> through <code>PhiSpaceR_1ref()</code> when the number of classes is less than <code>fallback_min_classes</code> .
<code>fallback_min_classes</code>	Integer. Minimum number of classes required before fitting the standard PhiSpace model when <code>fallback = "scoreCells"</code> .
<code>fallback_score</code>	Character. Which <code>scoreCells()</code> output to use as the fallback PhiSpace score: " <code>signature</code> " or " <code>correlation</code> ".

Details

- Parameter tuning** By default, PhiSpace takes `ncomp` equal to the total number of phenotypes. For example, if there are 20 cell types and 5 sample sources (e.g. *in vivo* and *in vitro*) defined in the reference, `ncomp` will be set to be 25. By default, PhiSpace doesn't do any feature selection. However, feature selection is very straightforward in PhiSpace. The partial least squares model automatically rank all features according to their contribution to predicting each phenotype. For example, if there are 25 phenotypes then PLS will return 25 rankings of all features. Based on this, customised feature selection can be done in two ways. The user either specify `nfeat`, i.e. the number of features (or markers) used to predict each phenotype; or specify `selectedFeat`, i.e. a subset of features used to predict all phenotypes (e.g. highly variable genes). If `nfeat` is provided, then `selectedFeat` will be automatically defined as the union of the `nfeat` markers of all phenotypes.
- Center and scale** By default, PhiSpace only center the data but do not scale. This is because we often use more features than standard single-cell pipelines (e.g. Seurat selects 2,000 highly variable genes) for a better prediction. This means that we might have included genes with small variance and scaling would greatly inflate the expression levels of these genes.

Value

- If `updateRef = FALSE` (default): An updated query SCE object with PhiSpace annotation results stored in reducedDim slot "PhiSpace";
- If `updateRef = TRUE`: A list of updated reference and query SCE objects with PhiSpace annotation results stored in reducedDim slot "PhiSpace".

References

Mao J., Deng Y. and Lê Cao K.-A. (2025). Φ -Space: continuous phenotyping of single-cell multi-omics data. *Genome Biology* 26, 323. doi:10.1186/s13059025037558

Mao J., Choi J. and Lê Cao K.-A. (2026). Φ -Space ST: a platform-agnostic method to identify cell states in spatial transcriptomics studies. *Cell Reports Methods* 6(8), 101483. doi:10.1016/j.crmeth.2026.101483

Examples

```
# Simulated reference and query; Cell_Cycle is the reference phenotype.
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
query <- logNormTransf(scuttle::mockSCE(ncells = 100, ngenes = 300))
query <- PhiSpace(ref, query, phenotypes = "Cell_Cycle",
                  refAssay = "logcounts", ncomp = 3)
head(reducedDim(query, "PhiSpace"))
```

PhiSpaceR_1ref

PhiSpace annotation using a single reference

Description

Core single-reference implementation for PhiSpace continuous phenotyping. Given an annotated reference dataset and one or more query datasets, this function builds a PLS or PCA regression model from the reference and projects the query onto the learned phenotype space. Both raw and normalised PhiSpace scores are returned. For a higher-level wrapper that handles multiple references, see [PhiSpace\(\)](#).

Usage

```
PhiSpaceR_1ref(
  reference,
  query,
  phenotypes = NULL,
  response = NULL,
  refAssay = "log1p",
  queryAssay = NULL,
  regMethod = c("PLS", "PCA"),
  ncomp = NULL,
  nfeat = NULL,
```

```

    selectedFeat = NULL,
    center = TRUE,
    scale = FALSE,
    DRinfo = FALSE,
    cellTypeThreshold = NULL,
    fallback = c("none", "scoreCells"),
    fallback_min_classes = 2,
    fallback_score = c("signature", "correlation")
  )

```

Arguments

reference	A SingleCellExperiment (SCE) object. The annotated reference dataset. Must contain an assay named by refAssay .
query	An SCE object or a list of SCE objects. The query dataset(s) to annotate. Each must contain an assay named by queryAssay .
phenotypes	Character vector. Column name(s) in colData(reference) specifying which phenotypes (e.g. cell type, disease condition) to predict. If NULL , response must be provided. Currently only supports categorical phenotypes; for continuous phenotypes, specify response directly.
response	Named matrix. Rows correspond to cells (columns) in reference ; columns correspond to phenotypes. If not NULL , overrides phenotypes . Can contain continuous values such as age or BMI.
refAssay	Character. Name of the assay in reference to use for model training (default: "log1p").
queryAssay	Character. Name of the assay in query to use for prediction. If NULL (default), uses the same value as refAssay .
regMethod	Character. Regression method: one of "PLS" (partial least squares, default) or "PCA" (principal component regression).
ncomp	Integer. Number of PLS/PCA components. If NULL (default), set to the total number of phenotype levels.
nfeat	Integer. Number of top-ranked features to select per phenotype for model fitting. If NULL (default), all features are used. See Details.
selectedFeat	Character vector. A set of pre-selected feature names to use for model fitting. If provided, overrides nfeat .
center	Logical. Whether to center the feature matrix before regression (default: TRUE). See Details.
scale	Logical. Whether to scale the feature matrix before regression (default: FALSE). See Details.
DRinfo	Logical. Whether to return dimension reduction information from PCA or PLS. Disable to save memory (default: FALSE).
cellTypeThreshold	Integer or NULL . If a positive integer, cell types with fewer than this many cells in the reference will be removed before model fitting. Only used when phenotypes is provided. Default is NULL (no filtering).

fallback Character. Optional fallback method for references with too few classes. "none" preserves the standard PhiSpace behaviour. "scoreCells" uses `scoreCells()` when the number of classes is less than `fallback_min_classes`.

fallback_min_classes Integer. Minimum number of classes required before fitting the standard PhiSpace model when `fallback = "scoreCells"`.

fallback_score Character. Which `scoreCells()` output to use as the fallback PhiSpace score: "signature" or "correlation".

Details

- **Parameter tuning** By default, PhiSpace takes `ncomp` equal to the total number of phenotypes. For example, if there are 20 cell types and 5 sample sources (e.g. *in vivo* and *in vitro*) defined in the reference, `ncomp` will be set to be 25. By default, PhiSpace doesn't do any feature selection. However, feature selection is very straightforward in PhiSpace. The partial least squares model automatically rank all features according to their contribution to predicting each phenotype. For example, if there are 25 phenotypes then PLS will return 25 rankings of all features. Based on this, customised feature selection can be done in two ways. The user either specify `nfeat`, i.e. the number of features (or markers) used to predict each phenotype; or specify `selectedFeat`, i.e. a subset of features used to predict all phenotypes (e.g. highly variable genes). If `nfeat` is provided, then `selectedFeat` will be automatically defined as the union of the `nfeat` markers of all phenotypes.
- **Center and scale** By default, PhiSpace only center the data but do not scale. This is because we often use more features than standard single-cell pipelines (e.g. Seurat selects 2,000 highly variable genes) for a better prediction. This means that we might have included genes with small variance and scaling would greatly inflate the expression levels of these genes.

Value

A list with the following components:

ncomp Integer. Number of components used. NA when the `scoreCells` fallback is used.

impScores Matrix. Feature importance scores (regression coefficients at the final component). NULL when the `scoreCells` fallback is used.

phenoDict Data frame mapping phenotype labels to their categories, or NULL if `response` was provided directly.

selectedFeat Character vector. Features used in the final model. With the `scoreCells` fallback, the union of the signature features, or all centroid features if there are no signatures.

YrefHat Matrix. Raw (unnormalised) predicted scores for the reference cells. NULL when the `scoreCells` fallback is used.

YrefHatNorm Matrix. Normalised predicted scores for the reference cells (via `normPhiScores()`). NULL when the `scoreCells` fallback is used.

PhiSpaceScore Matrix or list of matrices. Raw (unnormalised) PhiSpace scores for the query dataset(s). A single matrix if one query was provided; a list of matrices if multiple queries.

PhiSpaceNorm Matrix or list of matrices. Normalised PhiSpace scores for the query dataset(s) (via `normPhiScores()`). Same structure as **PhiSpaceScore**.

center Logical. Whether centering was applied. NA when the `scoreCells` fallback is used.

scale Logical. Whether scaling was applied. NA when the `scoreCells` fallback is used.

atlas_re List. Internal model object from `SuperPC()`, containing regression results and preprocessing parameters needed for prediction. NULL when the `scoreCells` fallback is used.

model A `PhiSpaceModel` (see `trainPhiSpace()`) trained on the genes shared with the queries. It can be saved and used with `predict.PhiSpaceModel()`. Absent when the `scoreCells` fallback is used.

fallback List. Present only when the `scoreCells` fallback is used. It records the method, the score type (`fallback_score`), the phenotype column and its number of classes, `fallback_min_classes`, and the `scoreCells()` metadata.

References

Mao J., Deng Y. and Lê Cao K.-A. (2025). Φ -Space: continuous phenotyping of single-cell multi-omics data. *Genome Biology* 26, 323. doi:10.1186/s13059025037558

See Also

`PhiSpace()` for the main user-facing wrapper that handles multiple references, `normPhiScores()` for score normalisation, `tunePhiSpace()` for parameter tuning.

Examples

```
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
query <- logNormTransf(scuttle::mockSCE(ncells = 100, ngenes = 300))
res <- PhiSpaceR_iref(ref, query, phenotypes = "Cell_Cycle",
                     refAssay = "logcounts", ncomp = 3)

names(res)
head(res$PhiSpaceNorm)
```

piScore

Compute pi-score for ranking differentially expressed genes

Description

Combines log fold change and p-value into a single significance score (Xiao et al. 2014). By default returns the signed variant $\log_2\text{FC} * -\log_{10}(p)$ for directional ranking; set `signed = FALSE` for the paper's original $|\log_2\text{FC}| * -\log_{10}(p)$.

Usage

```
piScore(
  de_results,
  logfc_col = "avg_log2FC",
  pval_col = "p_val_adj",
  signed = TRUE,
  pval_floor = .Machine$double.xmin,
  add_column = TRUE,
  rank = FALSE
)
```

Arguments

de_results	A data.frame-like object containing differential expression results.
logfc_col	Character. Column containing log2 fold change. If the default is absent, common alternatives are detected automatically.
pval_col	Character. Column containing p-values. If the default is absent, common alternatives are detected automatically.
signed	Logical. If TRUE , preserve the direction of the log fold change. If FALSE , use the unsigned Xiao et al. definition.
pval_floor	Numeric. Minimum p-value used before applying $-\log_{10}(p)$, avoiding infinite scores when p-values are zero.
add_column	Logical. If TRUE , append pi_score to de_results . If FALSE , return a numeric vector.
rank	Logical. If TRUE and add_column = TRUE , append pi_rank where rank 1 is the largest pi-score.

Value

If **add_column = TRUE**, a data.frame with **pi_score** and optionally **pi_rank** appended. Otherwise, a numeric vector aligned to the input rows.

References

Xiao Y, Hsiao T-H, Suresh U, et al. (2014) A novel significance score for gene selection and ranking. *Bioinformatics* 30(6):801-807. doi:10.1093/bioinformatics/btr671

Examples

```
de <- data.frame(gene = c("g1", "g2", "g3"), avg_log2FC = c(2, -1, 0.5),
  p_val_adj = c(1e-10, 0.01, 0.5))
piScore(de, rank = TRUE)
```

plot.PhiSpaceClustering

Plot method for PhiSpaceClustering objects

Description

Plot method for PhiSpaceClustering objects

Usage

```
## S3 method for class 'PhiSpaceClustering'
plot(x, type = c("pca", "elbow", "silhouette", "variance"), ...)
```

Arguments

<code>x</code>	A PhiSpaceClustering object
<code>type</code>	Character string specifying plot type. Options: "pca" (PC1 vs PC2), "elbow" (elbow plot if <code>k_range</code> was used), "silhouette" (silhouette plot if <code>k_range</code> was used), "variance" (PCA variance explained). Default is "pca".
<code>...</code>	Additional arguments passed to plotting functions

Value

A ggplot object.

plotPhiSpaceHeatMap *Plot PhiSpace scores as a heatmap*

Description

Plot PhiSpace scores as a heatmap

Usage

```
plotPhiSpaceHeatMap(
  PhiSpaceScore,
  phenoDict = NULL,
  phenotypes = NULL,
  queryLabs = NULL,
  refLvls = NULL,
  queryLvls = NULL,
  cluster_columns = FALSE,
  cluster_rows = FALSE,
  show_row_names = FALSE,
```

```

    show_column_names = TRUE,
    row_title_rot = 0,
    ...
)

```

Arguments

PhiSpaceScore	Matrix.
phenoDict	A named list, categorising columns (phenotypes) in PhiSpaceScore . Total number of elements have to be identical to number of columns of PhiSpaceScore .
phenotypes	Optional. Types of phenotypes. If provided, has to be of the same length as phenoDict .
queryLabs	Character vector.
refLvls	Character vector.
queryLvls	Character vector.
cluster_columns	Logic.
cluster_rows	Logic.
show_row_names	Logic.
show_column_names	Logic.
row_title_rot	Rotation (angle) of row titles, inherited from ComplexHeatmap::Heatmap
...	Additional parameters adjusting the ComplexHeatmap object

Value

A heatmap of PhiSpace scores.

Examples

```

set.seed(1)
scores <- matrix(rnorm(60 * 5), 60, 5,
                dimnames = list(NULL, c("T", "B", "NK", "ctrl", "stim")))
plotPhiSpaceHeatMap(
  scores,
  phenoDict = list(cellType = c("T", "B", "NK"), treatment = c("ctrl", "stim")),
  queryLabs = rep(c("cluster1", "cluster2"), each = 30)
)

```

predict.PhiSpaceModel

Score a query with a stored PhiSpace model

Description

Score a query with a stored PhiSpace model

Usage

```
## S3 method for class 'PhiSpaceModel'
predict(object, newdata, assay = NULL, nfeat = NULL, ...)
```

Arguments

object	A PhiSpaceModel from <code>trainPhiSpace()</code> .
newdata	The query: a SummarizedExperiment (for example a SingleCellExperiment) or a gene by cell matrix, with gene names as row names. It must share at least one gene with the model.
assay	Character. Assay of newdata to use. Defaults to the model's refAssay . With "rank", the genes are re-ranked within each cell over the model genes, as in <code>PhiSpaceR_1ref()</code> .
nfeat	Integer or NULL. For a model trained with nfeat and statistics over candidate genes (nfeatPool): the number of genes per phenotype to select among the candidate genes that the query has. If NULL, the training nfeat is used, and genes are selected again only if the query lacks model genes.
...	Not used.

Details

If the query lacks some of the model genes, the model is first restricted to the genes that the query has:

- A model trained with **nfeat** and statistics over candidate genes (**nfeatPool** in `trainPhiSpace()`) selects **nfeat** genes per phenotype again, among its candidate genes that the query has, with a message. It refits the model on these candidate genes, selects the top genes of that fit and refits the model on them (see `trainPhiSpace()`).
- Another model with stored statistics (see `trainPhiSpace()`) is refitted exactly on these genes, with a message. The scores equal those of a model trained on the reference with **selectedFeat** set to these genes.
- A model without statistics (a "rank" model, a model trained with **keepStats** = FALSE, or `PhiSpaceR_1ref()`\$model) keeps the coefficients of these genes and drops the others, with a warning. These scores are approximate: the error grows with the share of the coefficients that the missing genes hold, which the warning reports.

The query must be preprocessed as the reference was. Before scoring, the query values over the model genes are compared with the reference values recorded in the model (in up to 2,000 evenly spaced cells), and a warning is given if

- the non-zero values are integers in one and not in the other (for example raw counts against log-transformed values);
- one has negative values and the other has none (for example centred or scaled values);
- the 99th percentiles of the absolute non-zero values differ by more than a factor of 10;
- the model was trained on the "**rank**" assay and the query assay is not "**rank**", or the reverse. Only an assay named "**rank**" is ranked again within each cell.

The check cannot detect every difference: for example, values normalised to different library sizes can pass it. Changes that act on all genes alike, such as the base of the logarithm, do not change the normalised scores, because the query is centred and each score is scaled by `normPhiScores()`.

Value

A cell by phenotype matrix of raw PhiSpace scores. Use `normPhiScores()` to normalise them.

Examples

```
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
query <- logNormTransf(scuttle::mockSCE(ncells = 100, ngenes = 300))
model <- trainPhiSpace(ref, phenotypes = "Cell_Cycle", refAssay = "logcounts",
                      ncomp = 3)
raw <- predict(model, query)
# Normalised scores, as PhiSpace() stores them.
head(normPhiScores(raw))
```

```
print.PhiSpaceClustering
```

Print method for PhiSpaceClustering objects

Description

Print method for PhiSpaceClustering objects

Usage

```
## S3 method for class 'PhiSpaceClustering'
print(x, ...)
```

Arguments

<code>x</code>	A PhiSpaceClustering object
<code>...</code>	Additional arguments (not used)

Value

`x`, invisibly. Called for the printed summary.

<code>pseudoBulk</code>	<i>Converting single cells to pseudo-bulks.</i>
-------------------------	---

Description

Converting single cells to pseudo-bulks.

Usage

```
pseudoBulk(
  sce,
  phenotypes = NULL,
  response = NULL,
  clusterid = NULL,
  assayName = "counts",
  resampSizes = 100,
  proportion = NULL,
  calcMean = FALSE,
  nPool = 15,
  seed = 904800
)
```

Arguments

<code>sce</code>	<code>SingleCellExperiment</code> object.
<code>phenotypes</code>	Phenotypes to predict. Can be multiple.
<code>response</code>	Response matrix. Can be continuous.
<code>clusterid</code>	Column name of <code>colData</code> for specifying groups for pseudobulking.
<code>assayName</code>	Assay used for pseudobulking.
<code>resampSizes</code>	How many pseudo-bulk resamples to generate from each cluster.
<code>proportion</code>	Used to determine unequal <code>resampSizes</code> .
<code>calcMean</code>	Logical. Calculate mean of sum of expression for pseudobulking.
<code>nPool</code>	How many cells to use for pseudobulking.
<code>seed</code>	Random seeds. The function sets the seed with <code>withr::local_seed()</code> , so the session's random-number state is restored on exit.

Value

An updated `SingleCellExperiment` object with a pseudobulk assay.

Examples

```
sce <- scuttle::mockSCE(ncells = 200, ngenes = 50)
pb <- pseudoBulk(sce, phenotypes = "Cell_Cycle", resampSizes = 10, nPool = 5)
dim(pb$reference)
```

rankFeatures
Rank Features by Discriminative Power

Description

Uses supervised learning methods (PLS-DA, PLS, or DWD) to rank features by their ability to discriminate between groups or predict a continuous response. Based on PhiSpace's `correlatePhiSpace` but generalized to work with any numeric data (not just PhiSpace scores).

Usage

```
rankFeatures(
  data,
  response,
  method = c("PLSDA", "PLS", "DWD"),
  source = c("reducedDim", "assay"),
  assay_name = "logcounts",
  reducedDim_name = "PhiSpace",
  ncomp = NULL,
  center = TRUE,
  scale = FALSE,
  dwf_params = list(),
  seed = NULL
)
```

Arguments

data	A matrix or data.frame where rows are observations and columns are features. Can also be a <code>SpatialExperiment</code> , <code>SingleCellExperiment</code> , or <code>SummarizedExperiment</code> object.
response	Either a character string specifying a column name in <code>colData</code> (when <code>data</code> is an experiment object), or a vector of response values. For classification (PLSDA, DWD), should be factor or character. For regression (PLS), should be numeric.
method	Character string specifying the method: "PLSDA" for classification, "PLS" for regression, or "DWD" for binary classification. Default is "PLSDA".
source	Character string specifying data source when <code>data</code> is an experiment object: "assay" or "reducedDim". Default is "reducedDim".
assay_name	Character string specifying which assay to use when <code>source = "assay"</code> . Default is "logcounts".

<code>reducedDim_name</code>	Character string specifying which reduced dimension to use when <code>source = "reducedDim"</code> . Default is "PhiSpace".
<code>ncomp</code>	Integer specifying number of components. If NULL (default), set automatically: number of classes for PLSDA, $\min(10, \text{ncol}(\text{data})/2)$ for PLS. Not used for DWD.
<code>center</code>	Logical indicating whether to center features. Default is TRUE.
<code>scale</code>	Logical indicating whether to scale features. Default is FALSE.
<code>dwd_params</code>	List of parameters for DWD method (see Details).
<code>seed</code>	Integer seed for reproducibility. Default is NULL. The function sets the seed with <code>withr::local_seed()</code> , so the session's random-number state is restored on exit.

Details

Methods:

- **PLSDA**: Partial Least Squares Discriminant Analysis for multi-class classification. Features ranked by coefficient magnitude.
- **PLS**: Partial Least Squares regression for continuous response. Features ranked by coefficient magnitude.
- **DWD**: Distance Weighted Discrimination for binary classification. Features ranked by discriminant weights.

DWD Parameters (in `dwd_params` list):

- `kernel`: Kernel function (default: `vanilladot()`)
- `qval`: q-value parameter (default: 1)
- `lambda`: Regularization parameter or sequence (default: auto-tuned)
- `cv_folds`: Cross-validation folds (default: 5)
- `max_cells`: Optional limit on the number of cells with a non-linear kernel (default: `Inf`, no limit). A non-linear kernel needs a cells by cells kernel matrix (8 bytes per entry: 0.8 GB for 10,000 cells, 80 GB for 100,000). If set, `rankFeatures()` stops before fitting when there are more cells than this.

Value

A list with class "FeatureRanking" containing:

<code>method</code>	The method used
<code>importance_scores</code>	Matrix of feature importance scores
<code>scores</code>	For PLSDA and PLS, the PLS component scores (observations x <code>ncomp</code>). For DWD, the discriminant score of each observation.
<code>model</code>	The fitted model object
<code>feature_ranking</code>	Data frame with features ranked by importance

`response_summary` Summary of the response variable

`parameters` List of parameters used

Examples

```
set.seed(1)
X <- matrix(rnorm(100 * 20), 100, 20,
            dimnames = list(NULL, paste0("gene", 1:20)))
group <- rep(c("a", "b"), each = 50)
X[group == "b", 1:3] <- X[group == "b", 1:3] + 2
res <- rankFeatures(X, response = group, method = "PLSDA")
head(res$feature_ranking)
```

RankTransf	<i>Rank transform of a SingleCellExperiment object.</i>
------------	---

Description

Rank transform of a `SingleCellExperiment` object.

Usage

```
RankTransf(sce, assayname = "counts", targetAssay = "rank", sparse = TRUE)
```

Arguments

`sce` `SingleCellExperiment` object.

`assayname` Character.

`targetAssay` Character.

`sparse` Logic.

Value

An updated `SingleCellExperiment` object with a rank transformed assay.

Examples

```
sce <- scuttle::mockSCE(ncells = 20, ngenes = 10)
sce <- RankTransf(sce)
assay(sce, "rank")[1:3, 1:3]
```

reNameCols	<i>Turn a matrix to a data.frame withe newly defined colnames.</i>
------------	--

Description

Turn a matrix to a data.frame withe newly defined colnames.

Usage

```
reNameCols(mat, key = "comp")
```

Arguments

mat	Matrix.
key	String.

Value

Data frame.

Examples

```
mat <- matrix(1:6, 2, 3)
reNameCols(mat)
reNameCols(mat, key = "PC")
```

RTassay	<i>Apply rank transform to a gene by cell matrix.</i>
---------	---

Description

Ranks the values within each column (the genes within each cell), with ties given the minimum rank, and rescales the ranks to 0 to 1: $(\text{rank} - 1) / (\text{nrow}(\mathbf{X}) - 1)$. A sparse matrix with no negative values stays sparse, because its zeros have rank 1 and therefore stay 0.

Usage

```
RTassay(X)
```

Arguments

X	A gene by cell matrix.
---	------------------------

Value

Rank transformed gene by cell matrix (sparse).

saveCellTypeMaps	<i>Save spatial cell type annotation heatmaps</i>
------------------	---

Description

Creates and optionally saves spatial heatmaps for all cell types from PhiSpace continuous annotations. This function generates publication-ready visualizations showing the spatial distribution of cell type scores across tissue sections.

Usage

```
saveCellTypeMaps(
  sce,
  methodName = "PhiSpace",
  tissueName = "SpatialSample",
  coordNames = c("x", "y"),
  freeColScale = FALSE,
  quants = c(0.1, 1),
  psize = 0.5,
  savePlots = TRUE,
  returnPlots = FALSE,
  legendPosition = "none",
  censQuant = 1,
  ctypes = NULL,
  outputDir = "figs/spatialCellTypeHeatmaps",
  width = 10,
  height = 10,
  fignrow = 4,
  figncol = 4,
  fsize_title = 10,
  plot_margin = c(0, 0, 0, 0),
  fileFormat = "png",
  dpi = 300
)
```

Arguments

sce	A SingleCellExperiment or SpatialExperiment object containing spatial coordinates and PhiSpace annotations in reducedDim slot.
methodName	Character. Name of the reduced dimension containing cell type scores (default: "PhiSpace").
tissueName	Character. Name identifier for the tissue/sample, used in output filenames when saving plots (default: "SpatialSample").
coordNames	Character vector of length 2. Column names in colData(sce) containing x and y spatial coordinates. For SpatialExperiment objects, this will be automatically detected if not specified (default: c("x", "y")).

freeColScale	Logical. Whether to use independent color scales for each cell type (TRUE) or a unified scale across all cell types (FALSE). Unified scaling facilitates comparison between cell types (default: FALSE).
quants	Numeric vector of length 2. Quantiles used to set color scale limits when freeColScale = FALSE (default: c(0.1, 1)).
psize	Numeric. Point size for spatial coordinates (default: 0.5).
savePlots	Logical. Whether to save plots to disk (default: TRUE).
returnPlots	Logical. Whether to return the list of ggplot objects (default: FALSE). Note: renamed from returnPlot for clarity.
legendPosition	Character. Position of color legend. One of "none", "left", "right", "bottom", "top" (default: "none").
censQuant	Numeric between 0 and 1. Quantile threshold for censoring low values to reduce noise in visualization (default: 1, no censoring).
ctypes	Character vector. Specific cell types to plot. If NULL, all cell types from the annotation will be plotted (default: NULL).
outputDir	Character. Directory path where plots will be saved. Directory will be created if it doesn't exist (default: "figs/spatialCellTypeHeatmaps").
width	Numeric. Plot width in inches when saving (default: 10).
height	Numeric. Plot height in inches when saving (default: 10).
fignrow	Integer. Number of rows in multi-panel figures (default: 4).
figncol	Integer. Number of columns in multi-panel figures (default: 4).
fsize_title	Numeric. Font size for plot titles (default: 10).
plot_margin	Numeric vector of length 4. Plot margins in points (top, right, bottom, left) (default: c(0,0,0,0)).
fileFormat	Character. File format for saved plots. One of "png", "pdf", "jpeg", "tiff" (default: "png").
dpi	Numeric. Resolution for raster formats (default: 300).

Details

This function creates spatial heatmaps showing the distribution of PhiSpace cell type scores across tissue sections. Each cell type is visualized as a separate heatmap where color intensity represents the confidence score for that cell type at each spatial location.

Key features:

- Automatic handling of both SingleCellExperiment and SpatialExperiment objects
- Flexible color scaling options (unified vs. independent scales)
- Multi-panel output for efficient visualization of many cell types
- Customizable censoring to reduce visualization noise
- Multiple output formats supported

Color scaling:

- `freeColScale = FALSE`: Uses the same color scale across all cell types, facilitating direct comparison of scores between different cell types
- `freeColScale = TRUE`: Each cell type uses its own optimal color scale, maximizing contrast within each individual plot

Output organization: When saving plots, multiple cell types are arranged in multi-panel figures (e.g., 4x4 grids). If there are more cell types than can fit in one figure, multiple files will be created with sequential numbering.

Value

If `returnPlots = TRUE`, returns a named list of ggplot objects, one for each cell type. If `savePlots = TRUE`, plots are saved to the specified directory. Invisible NULL otherwise.

See Also

[VizSpatial](#) for single cell type visualization [PhiSpace](#) for generating cell type annotations

Examples

```
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
query <- logNormTransf(scuttle::mockSCE(ncells = 100, ngenes = 300))
spe <- SpatialExperiment::SpatialExperiment(
  assays = list(logcounts = logcounts(query)),
  spatialCoords = cbind(x = runif(100), y = runif(100))
)
spe <- PhiSpace(ref, spe, phenotypes = "Cell_Cycle", refAssay = "logcounts",
  ncomp = 3)
# Return the maps instead of writing files.
maps <- saveCellTypeMaps(spe, savePlots = FALSE, returnPlots = TRUE)
names(maps)
```

Score2Prob

Convert PhiSpace Scores to Probabilities

Description

Converts a cell by cell type likelihood matrix to a probability matrix by setting negative values to zero and normalizing each row to sum to 1.

Usage

```
Score2Prob(PhiSc)
```

Arguments

PhiSc A numeric matrix containing values ranging from -1 to 1. Rows typically represent cells and columns represent cell types.

Details

The function performs the following steps:

1. Sets all negative similarity values to zero
2. Calculates the sum of each row
3. Divides each element in a row by the row sum to create probabilities
4. Handles edge cases where rows sum to zero after removing negatives

For rows that contain all negative values (resulting in zero sums after transformation), the function will issue a warning and set those rows to all zeros. Alternative behavior could be implemented to assign uniform probabilities.

Value

A numeric matrix of the same dimensions as the input, where:

- All negative values are set to zero
- Each row sums to 1 (forms a probability distribution)
- Row and column names are preserved from the input

Examples

```
set.seed(1)
scores <- matrix(runif(20, -1, 1), 5, 4, dimnames = list(NULL, LETTERS[1:4]))
round(Score2Prob(scores), 2)
```

scoreCells

Score query cells against reference classes

Description

`scoreCells()` provides a lightweight correlation- and signature-based fallback for regimes where a full PhiSpace model is not meaningful, such as single-class references or references with very few classes. It stores score matrices in `reducedDim()` slots with one column per reference class.

Usage

```
scoreCells(
  reference,
  query,
  class_col = NULL,
  assay_ref = "logcounts",
  assay_query = "logcounts",
  signature_genes = NULL,
  signature_method = c("mean", "findMarkers"),
  marker_test = c("t", "wilcox", "binom"),
```

```

marker_pval_type = "any",
marker_direction = c("up", "down", "any"),
n_top_genes = 300,
n_signature_genes = 100,
remove_housekeeping = TRUE,
housekeeping_patterns = c("^Rpl", "^Rps", "^mt-", "^MT-", "^Mrpl", "^Mrps"),
housekeeping_genes = NULL,
shared_genes_only = TRUE,
scoring = c("both", "correlation", "signature"),
cor_method = c("spearman", "pearson"),
zscore_signature = TRUE,
score_prefix = "scoreCells",
BPPARAM = BiocParallel::SerialParam(),
verbose = TRUE
)

```

Arguments

reference	A <code>SingleCellExperiment</code> or <code>SpatialExperiment</code> object.
query	A <code>SingleCellExperiment</code> or <code>SpatialExperiment</code> object to score.
class_col	Character or <code>NULL</code> . Column in <code>colData(reference)</code> defining reference classes. If <code>NULL</code> , all reference cells are treated as one class.
assay_ref	Character. Assay in reference used for scoring.
assay_query	Character. Assay in query used for scoring.
signature_genes	<code>NULL</code> , a character vector, or a named list of character vectors. If <code>NULL</code> , signatures are generated from reference . A character vector is accepted for a single-class reference. A multi-class reference requires a named list whose names exactly match the reference classes. Supplied signatures are used as-is after removing duplicates and genes unavailable in either assay; marker-selection and housekeeping-removal arguments are ignored.
signature_method	Character. Signature selection method. "mean" (default) uses mean-expression ranking for every class. "findMarkers" uses <code>scrn::findMarkers()</code> for multi-class references and falls back to mean-expression ranking for single-class references. <code>scrn</code> deprecates <code>findMarkers()</code> in Bioconductor 3.24, so this option will be removed when <code>scrn</code> removes the function.
marker_test	Character. Test type passed to <code>scrn::findMarkers()</code> .
marker_pval_type	Character. <code>pval.type</code> passed to <code>scrn::findMarkers()</code> .
marker_direction	Character. Direction used when selecting markers: "up", "down", or "any".
n_top_genes	Integer. Candidate gene pool per class before removing housekeeping genes.
n_signature_genes	Integer. Number of final signature genes per class.

remove_housekeeping	Logical. Whether to remove housekeeping genes from class signatures.
housekeeping_patterns	Character vector of regex patterns identifying housekeeping genes.
housekeeping_genes	Character vector of additional housekeeping genes. If <code>NULL</code> , a small built-in list is used.
shared_genes_only	Logical. If <code>TRUE</code> , use the intersection of reference and query genes. If <code>FALSE</code> , reference and query row names must be identical.
scoring	Character. Which score matrices to compute: <code>"both"</code> , <code>"correlation"</code> , or <code>"signature"</code> .
cor_method	Character. Correlation method: <code>"spearman"</code> or <code>"pearson"</code> .
zscore_signature	Logical. Whether to z-score signature scores. For multi-class references, z-scoring is across classes within each cell. For a single-class reference, z-scoring is across cells.
score_prefix	Character. Prefix for output <code>reducedDim()</code> names.
BPPARAM	BiocParallel parameter object passed to <code>scan::findMarkers()</code> .
verbose	Logical. Whether to emit progress messages.

Details

A correlation score measures similarity between a query cell and a reference class centroid in the relative expression pattern across that class's effective signature genes. Different classes can therefore be scored using different genes. Correlations are `NA` for query cells with fewer than two complete observations or zero expression variance across the relevant signature.

A signature score is the average query expression across the same effective signature genes, optionally z-scored. When `scoring = "both"`, the correlation and signature branches use exactly the same genes for each class.

Value

The modified `query` object with score matrices stored in `reducedDim()` slots and run details stored in `metadata(query)$scoreCells`.

Examples

```
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
query <- logNormTransf(scuttle::mockSCE(ncells = 100, ngenes = 300))
query <- scoreCells(ref, query, class_col = "Cell_Cycle",
                    signature_method = "mean", n_signature_genes = 20,
                    verbose = FALSE)
reducedDimNames(query)
```

scransTransf	<i>Run scrans normalisation</i>
--------------	---------------------------------

Description

Computes pooled size factors with `scrans::computeSumFactors()` and log-normalises the counts with `logNormTransf()`.

Usage

```
scransTransf(sce, assayName = "counts", clusts = NULL)
```

Arguments

<code>sce</code>	SCE object.
<code>assayName</code>	Assay name.
<code>clusts</code>	Cluster labels (optional).

Value

SCE object with the assay `logcounts` and the size factors, centred to mean 1, in `colData(sce)$sizeFactor`.

Examples

```
sce <- scuttle::mockSCE(ncells = 200, ngenes = 200)
# Pool cells within known groups to compute the size factors.
sce <- scransTransf(sce, clusts = sce$Cell_Cycle)
assayNames(sce)
```

selectFeat	<i>Select nfeat top features from each column (label) of impScores</i>
------------	--

Description

Select nfeat top features from each column (label) of impScores

Usage

```
selectFeat(impScores, nfeat = 5, absVal = TRUE)
```

Arguments

<code>impScores</code>	Matrix. Each column corresponds to a phenotype (eg a cell type) and contains importance scores of features for predicting that phenotype.
<code>nfeat</code>	Number of top features to select from each column of impScores.
<code>absVal</code>	Take absolute value or not.

Value

Vector of selected features.

Examples

```
set.seed(1)
imp <- matrix(rnorm(40), 10, 4,
              dimnames = list(paste0("gene", 1:10), LETTERS[1:4]))
selectFeat(imp, nfeat = 2)$selectedFeat
```

spatial_smooth_expression

Spatial smoothing of gene expression using KNN with kernel weighting

Description

Spatial smoothing of gene expression using KNN with kernel weighting

Usage

```
spatial_smooth_expression(
  expression_matrix,
  coordinates,
  k = 10,
  kernel = "linear",
  sigma = NULL,
  include_self = TRUE,
  verbose = TRUE
)
```

Arguments

<code>expression_matrix</code>	Gene x Cell matrix of expression values
<code>coordinates</code>	Data frame with columns 'x' and 'y' for cell coordinates
<code>k</code>	Number of nearest neighbors (default: 10)
<code>kernel</code>	Kernel function: "gaussian", "uniform", or "linear" (default: "gaussian")
<code>sigma</code>	Bandwidth parameter for Gaussian kernel (default: auto-computed)
<code>include_self</code>	Whether to include the cell itself in smoothing (default: TRUE)
<code>verbose</code>	Whether to report progress with <code>message()</code> and show a progress bar (default: TRUE).

Value

Smoothed gene x cell expression matrix

spatialSampler

Spatially-aware subsampling of SpatialExperiment objects

Description

Spatially-aware subsampling of SpatialExperiment objects

Usage

```
spatialSampler(
  spe,
  prop = 0.1,
  method = "grid",
  grid_size = NULL,
  seed = 123,
  min_cells_per_region = 1,
  balance_regions = FALSE,
  verbose = TRUE
)
```

Arguments

<code>spe</code>	SpatialExperiment object
<code>prop</code>	Proportion of cells to sample (between 0 and 1)
<code>method</code>	Sampling method: "grid", "kmeans", or "random" (default: "grid")
<code>grid_size</code>	Number of grid cells per dimension for grid method (default: auto-calculated)
<code>seed</code>	Random seed for reproducibility (default: 123). The function sets the seed with <code>withr::local_seed()</code> , so the session's random-number state is restored on exit.
<code>min_cells_per_region</code>	Minimum cells required per spatial region (default: 1)
<code>balance_regions</code>	Whether to balance sampling across spatial regions (default: FALSE; if true, might create duplicated spatial units)
<code>verbose</code>	Whether to report progress with <code>message()</code> (default: TRUE)

Value

Subsampled SpatialExperiment object

Examples

```
set.seed(1)
sce <- logNormTransf(scuttle::mockSCE(ncells = 300, ngenes = 50))
spe <- SpatialExperiment::SpatialExperiment(
  assays = list(logcounts = logcounts(sce)),
```

```

    spatialCoords = cbind(x = runif(300), y = runif(300))
  )
  sub <- spatialSampler(spe, prop = 0.2, method = "grid")
  ncol(sub)

```

spatialSmoother	<i>Spatial smoother wrapper for SpatialExperiment and SingleCellExperiment objects</i>
------------------------	--

Description

Spatial smoother wrapper for SpatialExperiment and SingleCellExperiment objects

Usage

```

spatialSmoother(
  object,
  reducedDim2smooth = "PhiSpace",
  smoothReducedDim = TRUE,
  assay2smooth = "logcounts",
  smoothedReducedDim = NULL,
  smoothedAssay = NULL,
  x_coord = NULL,
  y_coord = NULL,
  k = 10,
  kernel = "linear",
  sigma = NULL,
  include_self = TRUE,
  verbose = TRUE
)

```

Arguments

object	SpatialExperiment or SingleCellExperiment object
reducedDim2smooth	Name of reduced dimension to smooth (default: "PhiSpace")
smoothReducedDim	Logical, whether to smooth reduced dimensions instead of gene expression
assay2smooth	Name of assay to smooth (for gene expression smoothing)
smoothedReducedDim	Name for the new smoothed reduced dimension (default: paste0(reducedDim2smooth, "_smoothed"))
smoothedAssay	Name for the new smoothed assay (default: paste0(assay2smooth, "_smoothed"))
x_coord	Column name in colData for x coordinates (required for SCE objects)
y_coord	Column name in colData for y coordinates (required for SCE objects)

k	Number of nearest neighbors (default: 10)
kernel	Kernel function: "gaussian", "uniform", or "linear" (default: "gaussian")
sigma	Bandwidth parameter for Gaussian kernel (default: auto-computed)
include_self	Whether to include the cell itself in smoothing (default: TRUE)
verbose	Whether to report progress with <code>message()</code> and show a progress bar (default: TRUE).

Value

Modified object with new smoothed assay or reduced dimension

Examples

```
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
query <- logNormTransf(scuttle::mockSCE(ncells = 100, ngenes = 300))
spe <- SpatialExperiment::SpatialExperiment(
  assays = list(logcounts = logcounts(query)),
  spatialCoords = cbind(x = runif(100), y = runif(100))
)
spe <- PhiSpace(ref, spe, phenotypes = "Cell_Cycle", refAssay = "logcounts",
  ncomp = 3)
spe <- spatialSmoother(spe, k = 5, verbose = FALSE)
reducedDimNames(spe)
```

split2	<i>Create partitions of data for cross-validation.</i>
---------------	--

Description

Create partitions of data for cross-validation.

Usage

```
split2(x, n)
```

Arguments

x	Vector.
n	Integer.

Value

A partition of index vector **x** to **n** folds.

subsample	<i>Subsample from a sce object</i>
-----------	------------------------------------

Description

Subsample from a sce object

Usage

```
subsample(sce, key = NULL, proportion = 0.1, minCellNum = 50, seed = 5202056)
```

Arguments

sce	SCE object to be sampled from
key	Required for stratified subsampling
proportion	Proportion of cells to sample
minCellNum	Minimum number of cells in each category
seed	Random seed. The function sets the seed with <code>withr::local_seed()</code> , so the session's random-number state is restored on exit.

Value

A downsized SCE object

Examples

```
sce <- scuttle::mockSCE(ncells = 200, ngenes = 10)
small <- subsample(sce, proportion = 0.2)
ncol(small)
# Stratified by a colData column, keeping at least 20 cells per class.
strat <- subsample(sce, key = "Cell_Cycle", proportion = 0.1, minCellNum = 20)
table(strat$Cell_Cycle)
```

summary.PhiSpaceClustering	<i>Summary method for PhiSpaceClustering objects</i>
----------------------------	--

Description

Summary method for PhiSpaceClustering objects

Usage

```
## S3 method for class 'PhiSpaceClustering'
summary(object, ...)
```

Arguments

<code>object</code>	A <code>PhiSpaceClustering</code> object
<code>...</code>	Additional arguments (not used)

Value

`object`, invisibly. Called for the printed summary.

SuperPC

Compute regression model using selected features.

Description

Compute regression model using selected features.

Usage

```
SuperPC(
  reference,
  YY,
  ncomp,
  selectedFeat = NULL,
  assayName = "logcounts",
  regMethod = c("PCA", "PLS"),
  center = TRUE,
  scale = FALSE,
  sparse = TRUE,
  DRinfo = FALSE
)
```

Arguments

<code>reference</code>	SingleCellExperiment object (the reference), or a cell by gene matrix of the <code>assayName</code> assay.
<code>YY</code>	Resonse matrix
<code>ncomp</code>	Integer. Number of components
<code>selectedFeat</code>	Character. Vector containing selected feature names
<code>assayName</code>	Which assay in reference to use for prediction
<code>regMethod</code>	Character. Regression method to use, either PCA or PLS.
<code>center</code>	Logic.
<code>scale</code>	Logic. Scale the predictor matrix or not
<code>sparse</code>	Use sparse matrices or not.
<code>DRinfo</code>	Logic. Whether to return dimension reduction information from PCA or PLS. Disable to save memory.

Value

A list containing regression input and output.

<code>trainPhiSpace</code>	<i>Train a PhiSpace model from a reference</i>
----------------------------	--

Description

Fits the PhiSpace regression model on an annotated reference and returns it as a `PhiSpaceModel` object. The model can be saved (for example with `saveRDS()`) and used later to score queries with `predict()` or `PhiSpace()`, without the reference cells.

Usage

```
trainPhiSpace(
  reference,
  phenotypes = NULL,
  response = NULL,
  refAssay,
  regMethod = c("PLS", "PCA"),
  ncomp = NULL,
  nfeat = NULL,
  selectedFeat = NULL,
  center = TRUE,
  scale = FALSE,
  DRinfo = FALSE,
  cellTypeThreshold = NULL,
  genes = NULL,
  keepStats = TRUE,
  nfeatPool = 2000,
  referenceName = NULL,
  species = NULL,
  geneIdType = NULL,
  preprocessing = NULL
)
```

Arguments

reference	A <code>SingleCellExperiment</code> (or <code>SummarizedExperiment</code>) object. The annotated reference dataset.
phenotypes	Character vector. Column name(s) in <code>colData(reference)</code> with the phenotypes to predict. If <code>NULL</code> , response must be provided.
response	Named matrix. Rows correspond to cells (columns) in reference ; columns correspond to phenotypes. If not <code>NULL</code> , overrides phenotypes .
refAssay	Character. Name of the assay in reference used for training. Queries must be preprocessed in the same way, so there is no default.

regMethod	Character. "PLS" (default) or "PCA".
ncomp	Integer. Number of components. If NULL, the total number of phenotype levels.
nfeat	Integer. Number of top features per phenotype. If NULL, all genes are used. See PhiSpaceR_1ref() .
selectedFeat	Character vector. Pre-selected features; overrides nfeat .
center, scale	Logical. Centring and scaling of the features.
DRinfo	Logical. Whether to keep the PLS or PCA scores and loadings.
cellTypeThreshold	Integer or NULL. Cell types with fewer cells than this are removed before training. Only used with phenotypes .
genes	Character vector or NULL. If given, training uses only these genes (those present in the reference).
keepStats	Logical. Whether to store the cross-product statistics that refit the model on a subset of its genes. Ignored for the "rank" assay. See Details.
nfeatPool	Number or NULL. With nfeat and keepStats = TRUE, the statistics are stored for the top nfeatPool genes per phenotype (at least nfeat), the candidates for selecting genes again when a query lacks model genes. Inf keeps all genes; NULL keeps only the selected genes, which are then refitted without a new selection. See Details.
referenceName, species, geneIdType	Character or NULL. Optional descriptions of the reference, stored in the model.
preprocessing	Character or NULL. Optional description of how refAssay was computed, for example "logTransf(use_log1p = TRUE)". Queries must be preprocessed in the same way; the description is printed with the model and in the warnings of the query check (see predict.PhiSpaceModel()).

Details

[PhiSpaceR_1ref\(\)](#) and [PhiSpace\(\)](#) fit the same model on the genes that the reference shares with the query. [trainPhiSpace\(\)](#) uses all reference genes, or the genes given in **genes**.

A query may lack some of the model genes, for example a targeted spatial panel. With **keepStats** = TRUE (the default), the model stores the cross-products $X'X$ and $X'Y$ of the centred (and scaled) reference over the model genes. [predict\(\)](#) then refits the model on the genes that the query has, without the reference cells. The refitted model equals a model trained on the reference with **selectedFeat** set to those genes.

With **nfeat**, the genes are selected by a fit on all reference genes, whereas [PhiSpace\(\)](#) selects them among the genes that the reference shares with the query. A model trained with **nfeat** therefore stores the statistics of a larger set of candidate genes: the top **nfeatPool** genes per phenotype. If the query lacks some model genes, [predict\(\)](#) refits the model on the candidate genes that the query has, selects **nfeat** genes per phenotype from that fit and refits the model on them. The result approximates the [PhiSpace\(\)](#) fit on the reference cells; with all genes as candidates (**nfeatPool** = **Inf**), the two are equal. When the query

genes are known before training, for example a targeted panel, set **genes** to them: the model then selects its genes among the query genes, as [PhiSpace\(\)](#) does.

The statistics are not kept for the "rank" assay. Ranks are computed within each cell over the model genes, so removing genes changes the reference values and an exact refit needs the reference cells. A "rank" model therefore scores a query that lacks genes approximately, with a warning (see [predict.PhiSpaceModel\(\)](#)).

$X'X$ has one row and one column per model gene: it takes $8G^2$ bytes for G genes, for example 72 MB for 3,000 genes and 3.2 GB for 20,000 genes. Use **nfeat** (with a smaller **nfeatPool**), **selectedFeat** or **genes** to train on fewer genes, or **keepStats = FALSE**. A larger **nfeatPool** gives a selection closer to that of [PhiSpace\(\)](#), and a larger model.

Value

A **PhiSpaceModel** object: a list with

atlas_re The fitted model, as returned in [PhiSpaceR_1ref\(\)](#)\$**atlas_re**.

selectedFeat Genes used by the model, in model order.

impScores Feature importance scores (see [PhiSpaceR_1ref\(\)](#)).

phenoDict Phenotype labels and their categories, or NULL if **response** was given.

responseNames Names of the score columns.

refAssay The training assay.

nCells Number of reference cells used for training.

classSizes Number of reference cells in each phenotype label, in the order of **phenoDict**, or NULL if **response** was given. Labels with few cells give less reliable scores.

stats NULL, or a list with **XtX** (genes x genes) and **XtY** (genes x phenotypes): the cross-products of the centred (and scaled) reference over the model genes (or over the candidate genes, see **nfeatPool**), with the uncoded response, and the column means **Xmeans** and standard deviations **Xscals** used.

nfeat, **nfeatPool** The **nfeat** used for gene selection, or NULL; the **nfeatPool** of the stored candidate genes, or NULL if the statistics cover only the selected genes.

referenceName, **species**, **geneIdType**, **preprocessing** As supplied, or NULL.

valueProfile A summary of the reference values over the model genes, used to check queries: whether all non-zero values are integers, the share of negative values and the 99th percentile of the absolute non-zero values, in up to 2,000 evenly spaced cells.

format_version, **phispace_version**, **created** Model format version, the PhiSpace version that trained the model, and the date.

See Also

[predict.PhiSpaceModel\(\)](#), [PhiSpace\(\)](#), [PhiSpaceR_1ref\(\)](#).

Examples

```
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
query <- logNormTransf(scuttle::mockSCE(ncells = 100, ngenes = 300))
model <- trainPhiSpace(ref, phenotypes = "Cell_Cycle", refAssay = "logcounts",
                       ncomp = 3, referenceName = "mock reference")

model
scores <- predict(model, query)
head(scores)
# A model can replace the reference in PhiSpace().
query <- PhiSpace(model, query)
```

<code>translateLabel</code>	<i>Translate cell type labels from finer to coarser according to a dictionary.</i>
-----------------------------	--

Description

Translate cell type labels from finer to coarser according to a dictionary.

Usage

```
translateLabel(text, dictionary, dFrom = "label.fine", dTo = "label.main")
```

Arguments

<code>text</code>	Character vector.
<code>dictionary</code>	Data frame.
<code>dFrom</code>	Character.
<code>dTo</code>	Character.

Value

A vector containing translated labels.

Examples

```
dictionary <- data.frame(
  label.fine = c("CD4 T", "CD8 T", "Naive B"),
  label.main = c("T cell", "T cell", "B cell")
)
translateLabel(c("CD8 T", "Naive B", "CD4 T"), dictionary)
```

tunePhiSpace*Tune ncomp and nfeat by cross-validation*

Description

Computes the K-fold cross-validation loss of a PhiSpace model over a grid of numbers of components (**ncomp**) and then over a grid of numbers of features per phenotype (**nfeat**), and returns the chosen values with the selected features. In an interactive session the function plots each loss curve and asks for a value unless **ncomp** or **nfeat** is given. In a non-interactive session it chooses the **ncomp** with the lowest loss, and **nfeat** must be given when **tune_nfeat** = **TRUE**.

Usage

```
tunePhiSpace(  
  reference,  
  assayName = "logcounts",  
  phenotypes = NULL,  
  YY = NULL,  
  ncomp = NULL,  
  impScores = NULL,  
  center = TRUE,  
  scale = FALSE,  
  tune_ncomp = TRUE,  
  tune_nfeat = TRUE,  
  ncompLimits = c(1, 100),  
  ncompGridSize = 20,  
  ncompGrid = NULL,  
  selectedFeat = NULL,  
  nfeatLimits = c(10, 15000),  
  nfeatGridSize = 30,  
  nfeatV = NULL,  
  Kfolds = 5,  
  regMethod = "PLS",  
  labelCode = "-1,1",  
  Ncores = 1,  
  seed = 5202056,  
  cellTypeThreshold = NULL,  
  nfeat = NULL  
)
```

Arguments

reference	SingleCellExperiment object.
assayName	Character.
phenotypes	Character.

YY	Matrix.
ncomp	Integer. With <code>tune_ncomp = TRUE</code> , the value to use after the cross-validation instead of asking or taking the lowest loss. With <code>tune_ncomp = FALSE</code> , the value to use; <code>NULL</code> uses the number of columns of <code>YY</code> .
impScores	Matrix.
center	Logical.
scale	Logical.
tune_ncomp	Logical: compute the cross-validation loss for each value in the <code>ncomp</code> grid.
tune_nfeat	Logical: compute the cross-validation loss for each value in the <code>nfeat</code> grid.
ncompLimits	Vector.
ncompGridSize	Integer.
ncompGrid	Vector.
selectedFeat	Vector.
nfeatLimits	Vector.
nfeatGridSize	Integer.
nfeatV	Vector.
Kfolds	Integer.
regMethod	Character.
labelCode	Character.
Ncores	Integer.
seed	Integer. The function sets the seed with <code>withr::local_seed()</code> , so the session's random-number state is restored on exit.
cellTypeThreshold	Integer or <code>NULL</code> . If a positive integer, cell types with fewer than this many cells in the reference will be removed before tuning. Only used when <code>phenotypes</code> is provided. Default is <code>NULL</code> (no filtering).
nfeat	Integer. With <code>tune_nfeat = TRUE</code> , the number of features per phenotype to use after the cross-validation instead of asking; required in a non-interactive session.

Value

A list with `ncomp`, `nfeat`, `impScores` (feature importance scores), `selectedFeat` (selected features), and `tuneNcompErrs` and `tuneNfeatErrs` (data frames of the cross-validation loss for each grid value, or `NULL` when that parameter was not tuned).

Examples

```
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
# Outside an interactive session, nfeat must be given.
tuned <- tunePhiSpace(ref, assayName = "logcounts", phenotypes = "Cell_Cycle",
                      ncompGrid = 1:4, nfeatV = c(20, 50), Kfolds = 3,
```

```

                                nfeat = 50)
tuned$ncomp
tuned$tuneNcompErrs

```

VizSpatial	<i>Visualising features (eg gene expression), metadata (eg cell type) or reducedDim component (eg PhiSpace cell type score) in 2D tissue space.</i>
------------	---

Description

The 2D spatial coordinates (x and y) should be stored as two columns in colData.

Usage

```

VizSpatial(
  obj,
  x_coord = "x",
  y_coord = "y",
  ptSize = 2,
  ptShape = 16,
  colBy = NULL,
  feature = NULL,
  assay2use = NULL,
  reducedDim = NULL,
  reducedDim2use = "PhiSpace",
  censor = FALSE,
  quant = 0.5,
  legend.position = "right",
  legend.symb.size = NULL,
  legend.title = NULL,
  fsize = 14,
  reOrder = FALSE,
  ...
)

```

Arguments

obj	Input object; either <code>SingleCellExperiment</code> or <code>SpatialExperiment</code> . If <code>SingleCellExperiment</code> , then <code>colData(obj)</code> has to contain columns named by <code>x_coord</code> and <code>y_coord</code> below.
x_coord	x coordinate name of a column of colData (if <code>type(obj) == SingleCellExperiment</code>).
y_coord	y coordinate name of a column of colData (if <code>type(obj) == SingleCellExperiment</code>).
ptSize	Size of point, representing spatial cell like objects (segmented cells, spots etc).
ptShape	Shape of point.

<code>colBy</code>	Name of a metadata column to colour points by.
<code>feature</code>	Name of a feature (eg gene) to colour points by.
<code>assay2use</code>	Which assay to use when feature is specified.
<code>reducedDim</code>	Name of reducedDim column to colour points by.
<code>reducedDim2use</code>	Name of reducedDim layer to use when reducedDim is specified.
<code>censor</code>	Logical. Whether to set smaller colour values to zero. Default is <code>FALSE</code> . (Recommend to set to <code>TRUE</code> when visualise PhiSpace scores,)
<code>quant</code>	A value between 0 and 1. Colour values below <code>quant</code> will be set to zero if <code>censor=TRUE</code> .
<code>legend.position</code>	Position of legened, one of "right", "bottome", "left" and "top".
<code>legend.symb.size</code>	Legend symbol size (for discrete legend).
<code>legend.title</code>	Alternative legend name to replace the original.
<code>fsize</code>	Base font size of figure.
<code>reOrder</code>	Logical. Whether to reorder points according to their values (ascending) or not. Set to be <code>TRUE</code> to avoid overplotting.
<code>...</code>	Other arguments to pass to <code>basicPlotFormat</code> .

Value

A `ggplot2` object.

Examples

```
set.seed(1)
ref <- logNormTransf(scuttle::mockSCE(ncells = 200, ngenes = 300))
query <- logNormTransf(scuttle::mockSCE(ncells = 100, ngenes = 300))
spe <- SpatialExperiment::SpatialExperiment(
  assays = list(logcounts = logcounts(query)),
  spatialCoords = cbind(x = runif(100), y = runif(100))
)
spe <- PhiSpace(ref, spe, phenotypes = "Cell_Cycle", refAssay = "logcounts",
  ncomp = 3)
VizSpatial(spe, reducedDim = "G1")
VizSpatial(spe, feature = "Gene_0001", assay2use = "logcounts")
```

zeroFeatQC

Minimal quality control (QC): remove features with all zero values.

Description

Minimal quality control (QC): remove features with all zero values.

Usage

```
zeroFeatQC(sce, assayName = "counts", verbose = TRUE)
```

Arguments

<code>sce</code>	SingleCellExperiment object.
<code>assayName</code>	Character.
<code>verbose</code>	Logical. Whether to report the result with <code>message()</code> (default: TRUE).

Value

An updated SingleCellExperiment object after minimal QC.

Examples

```
sce <- scuttle::mockSCE(ncells = 20, ngenes = 10)
counts(sce)[1:2, ] <- 0
sce <- zeroFeatQC(sce)
dim(sce)
```

Index

`as.sparse.matrix`, 3

`basicPlotFormat`, 3

`cellTypeCorMat`, 5

`cellTypeCorMat`, `data.frame`-method
(*cellTypeCorMat*), 5

`cellTypeCorMat`, `matrix`-method
(*cellTypeCorMat*), 5

`cellTypeCorMat`, `SingleCellExperiment`-method
(*cellTypeCorMat*), 5

`classErr`, 6

`CLRnorm`, 7

`clusterPhiSpace`, 7

`codeY`, 11

`codeY_vec`, 11

`compute_kernel_weights`, 12

`computeUMAP`, 12

`CVTune_ncomp`, 13

`CVTune_nfeat`, 14

`doubleCent`, 16

`findNiches`, 16

`getError`, 18

`getError_nfeat`, 19

`getLabels`, 20

`getPC`, 21

`KeepCommonGenes`, 22

`logNormTransf`, 23

`logNormTransf()`, 50

`logTransf`, 24

`logTransf()`, 23

`message()`, 17, 51, 52, 54, 65

`mvr`, 25

`normPhiScores`, 26

`normPhiScores()`, 32, 33, 38

`phenotype`, 27

`PhiSpace`, 27, 46

`PhiSpace()`, 30, 33, 57–59

`PhiSpaceR_1ref`, 30

`PhiSpaceR_1ref()`, 29, 37, 58, 59

`piScore`, 33

`plot.PhiSpaceClustering`, 35

`plotPhiSpaceHeatMap`, 35

`predict()`, 57, 58

`predict.PhiSpaceModel`, 37

`predict.PhiSpaceModel()`, 28, 33, 58, 59

`print.PhiSpaceClustering`, 38

`pseudoBulk`, 39

`rankFeatures`, 40

`RankTransf`, 42

`reNameCols`, 43

`RTassay`, 43

`saveCellTypeMaps`, 44

`saveRDS()`, 57

`Score2Prob`, 46

`scoreCells`, 47

`scoreCells()`, 29, 32

`scransTransf`, 50

`scransTransf()`, 23

`selectFeat`, 50

`spatial_smooth_expression`, 51

`spatialSampler`, 52

`spatialSmoother`, 53

`split2`, 54

`subsample`, 55

`summary.PhiSpaceClustering`, 55

`SuperPC`, 56

`suppressMessages()`, 17

`trainPhiSpace`, 57

`trainPhiSpace()`, 28, 33, 37

`translateLabel`, [60](#)
`tunePhiSpace`, [61](#)
`tunePhiSpace()`, [33](#)
`VizSpatial`, [46](#), [63](#)
`withr::local_seed()`, [9](#), [14](#), [15](#), [17](#), [39](#),
[41](#), [52](#), [55](#), [62](#)
`zeroFeatQC`, [64](#)