Package: StabMap (via r-universe)

October 16, 2026

Type Package

Title Stabilised mosaic single cell data integration using unshared features

Version 1.6.0

Description StabMap performs single cell mosaic data integration by first building a mosaic data topology, and for each reference dataset, traverses the topology to project and predict data onto a common embedding. Mosaic data should be provided in a list format, with all relevant features included in the data matrices within each list object. The output of stabMap is a joint low-dimensional embedding taking into account all available relevant features. Expression imputation can also be performed using the StabMap embedding and any of the original data matrices for given reference and query cell lists.

License GPL-2

Encoding UTF-8

URL https://sydneybiox.github.io/StabMap, https://sydneybiox.github.io/StabMap/

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

biocViews SingleCell, DimensionReduction, Software

Depends R (>= 4.4.0),

Imports igraph, slam, BiocNeighbors, Matrix, MASS, abind, SummarizedExperiment, methods, MatrixGenerics, BiocGenerics, BiocSingular, BiocParallel

Suggests scran, scater, knitr, UpSetR, gridExtra, SingleCellMultiModal, BiocStyle, magrittr, testthat (>= 3.0.0), purrr, sparseMatrixStats

LazyData false

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libglpk-dev libxml2-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 13:03:28 UTC

RemoteUrl https://github.com/bioc/StabMap

RemoteRef RELEASE_3_23

RemoteSha 2c983548d049d33966b014d142a74b45bc058aab

Contents

adaptiveKNN 2
classifyEmbedding 3
getAdaptiveK 5
imputeEmbedding 6
mockMosaicData 8
mosaicDataTopology 9
mosaicDataUpSet 9
reWeightEmbedding 10
stabMap 11

Index **14**

`adaptiveKNN` *Adaptive k-Nearest Neighbour Classification*

Description

Adaptive k-Nearest Neighbour Classification for a k-nearest neighbour matrix, given class labels and local k values for the training data

Usage

```
adaptiveKNN(knn, class, k_local)
```

Arguments

`knn`	Is a k-nearest neighbour matrix, giving the indices of the training set that the query is closest to. Rows are the query cells, columns are the NNs. Typically output using BiocNeighbors::queryKNN(,k = max(k_local)).
`class`	Is the labels associated with the training set.
`k_local`	Is an integer vector length of the training set, giving the local k to use if k_local is given as a single integer, then that value is used as k for all observations.

Value

A character vector of of classifications for the test set.

Examples

```
# Generate example data
data <- matrix(rpois(10 * 20, 10), 10, 20) # 10 genes, 20 cells
data_2 <- matrix(rpois(10 * 30, 10), 10, 30) # 10 genes, 30 cells

# Generate error matrix for k_local
E <- matrix(runif(100), 20, 5)
colnames(E) <- paste0("K_", 1:5)

# Define training class labels and adaptive k-values
class <- factor(rep(letters[1:2], each = 10))
k_local <- getAdaptiveK(E, labels = class)

knn <- BiocNeighbors::queryKNN(
  t(data), t(data_2),
  k = max(as.numeric(gsub("K_", "", k_local)))
)$index

# Adaptive KNN classification
test <- adaptiveKNN(
  knn, class, as.numeric(gsub("K_", "", k_local))
)
```

`classifyEmbedding` *Adaptive k-Nearest Neighbour Classification using the StabMap joint embedding*

Description

Performs adaptive k-nearest neighbour classification of discrete labels for a training set from a query set, leveraging the StabMap joint embedding. The training labels are defined in ‘labels‘, with all other rows of the embedding treated as the testing set.

Usage

```
classifyEmbedding(
  coords,
  labels,
 type = c("uniform_fixed", "adaptive_labels", "adaptive_local", "uniform_optimised"),
  k_values = 5,
  error_measure = c("simple_error", "balanced_error"),
  adaptive_nFold = 2,
  adaptive_nRep = 5,
  adaptive_local_nhood = 100,
  adaptive_local_smooth = 10,
  verbose = TRUE
)
```

Arguments

coords	A cells (rows) x dimensions data matrix, on which euclidean distances are to be calculated for KNN classification. Must have rownames. Typically, output from ‘stabMap()‘.
labels	A named character vector of labels for the training set.
type	A character of the type of adaptive KNN classification to be used. Must be one of "adaptive_local", "adaptive_labels", "uniform_optimised", or "uniform_fixed". Default is "uniform_fixed".
k_values	A numeric vector of potential k values. If type is "uniform_fixed", then the first value of k_values is used. Default is 5.
error_measure	Is the error type to use for selection of the best k. Must be one of "simple_error" or "balanced_error". "simple_error" weights all cells equally. "balanced_error" weights error by ‘labels‘ factors. Only affects error type for type == "uniform_optimised".
adaptive_nFold	Is the number of folds for adaptive selection cross-validation.
adaptive_nRep	Is the number of repetitions of adaptive selection cross-validation.
adaptive_local_nhood	Is the neighbourhood size for optimising locally.
adaptive_local_smooth	Is the number of neighbours to use for smoothing locally.
verbose	Logical whether to print repetition and fold number for adaptive selection cross-validation.

Value

Is a dataframe with rows the same as coords, and same rownames. Columns are: input_labels is the training labels that were provided in ‘labels‘ (NA is used as labels for the testing set), resubstituted_labels is predicted labels for all rows (including for the training data), predicted_labels is predicted labels for the testing set but true labels as provided in ‘labels‘ for the training set, k is the adaptive k value used for that each row of the training set.

Examples

```
set.seed(100)
# Simulate coordinates
coords <- matrix(rnorm(1000), 100, 10)
rownames(coords) <- paste0("cell_", seq_len(nrow(coords)))

# Define labels of the first 50 cells
labels <- rep(paste0("type_", letters[1:5]), 10)
names(labels) <- rownames(coords)[seq_along(labels)]

# Uniform fixed KNN classification
knn_out <- classifyEmbedding(
  coords, labels,
  type = "uniform_fixed", k_values = 5
)
```

```
table(knn_out$predicted_labels)

# Adaptive KNN classification using local error
knn_out <- classifyEmbedding(
coords, labels,
type = "adaptive_local",
k_values = 2:3,
adaptive_nFold = 5,
adaptive_nRep = 10
)
table(knn_out$predicted_labels)

knn_out <- classifyEmbedding(
  coords, labels,
  type = "adaptive_labels",
  k_values = 2:3,
  adaptive_nFold = 5,
  adaptive_nRep = 10
)
table(knn_out$predicted_labels)

# Adaptive KNN classification using uniform optimised with balanced error
knn_out <- classifyEmbedding(
  coords, labels,
  type = "uniform_optimised",
  k_values = 2:3,
  adaptive_nFold = 3,
  adaptive_nRep = 10,
  error_measure = "balanced_error"
)
table(knn_out$predicted_labels)
```

`getAdaptiveK` *Adaptive k selection for KNN classification*

Description

Given an error matrix, identify the k that maximises the accuracy for cells belonging to a provided labelling/grouping. If no labelling given, expect a cell-cell similarity network to identify the k that maximises the accuracy for cells within that neighbourhood. If neither are given, simply treat all cells as if they have the same labelling/grouping

Usage

```
getAdaptiveK(E, labels = NULL, local = NULL, outputPerCell = TRUE, ...)
```

Arguments

E	An error matrix with rows corresponding to cells and columns corresponding to candidate k values, with values themselves corresponding to error values (either binary for single classification, or continuous after multiple classification).
labels	Group labels for cells.
local	A neighbourhood index representation, as typically output using BiocNeighbors::findKNN().
outputPerCell	Logical whether to return adaptive k for each cell, not just for each label type (used for when labels is given).
...	Includes return_colnames, whether to give the colnames of the best selected, or just the index, which is default TRUE.

Value

Vector of adaptive k values.

Examples

```
E <- matrix(runif(100), 20, 5)
colnames(E) <- paste0("K_", 1:5)

# generate cell labels
labels <- factor(rep(letters[1:2], each = 10))

# generate nearest neighbourhood index representation
data <- matrix(rpois(10 * 20, 10), 10, 20) # 10 genes, 20 cells
local <- BiocNeighbors::findKNN(t(data), k = 5, get.distance = FALSE)$index

best_k_labels <- getAdaptiveK(E,
  labels = labels
)
best_k_local <- getAdaptiveK(E,
  local = local
)
```

---

imputeEmbedding *Impute values using StabMap joint embedding*

Description

Performs naive imputation of values from the list of mosaic data and joint embedding from StabMap.

Usage

```
imputeEmbedding(
  assay_list,
  embedding,
  reference = Reduce(union, lapply(assay_list, colnames)),
  query = Reduce(union, lapply(assay_list, colnames)),
  neighbours = 5,
  fun = mean
)
```

Arguments

assay_list	List of mosaic data from which to perform imputation.
embedding	Joint embedding from which to extract nearest neighbour relationships.
reference	Character vector of cell names to treat as reference cells.
query	Character vector of cell names to treat as query cells.
neighbours	Number of nearest neighbours to consider (default 5).
fun	function (default 'mean') to aggregate nearest neighbours' imputed values.

Value

List containing imputed values from each assay_list data matrix which contains reference cells.

Examples

```
set.seed(2021)
assay_list <- mockMosaicData()
lapply(assay_list, dim)

# stabMap
out <- stabMap(assay_list,
  ncomponentsReference = 20,
  ncomponentsSubset = 20
)

# impute values
imp <- imputeEmbedding(assay_list, out)

# inspect the imputed values
lapply(imp, dim)
imp[[1]][1:5, 1:5]
```

mockMosaicData *mockMosaicData*

Description

Mock up a mosaic data list using simulated data, for use in documentation examples.

Usage

```
mockMosaicData(
  names = c("D1", "D2", "D3"),
  ncells = c(50, 50, 50),
  ngenes = list(1:150, 76:225, 151:300),
  fun = "rnorm",
  ...
)
```

Arguments

names	character vector of mock datasets.
ncells	integer vector of cells in each mock dataset.
ngenes	list containing integer vectors of features measured in each mock dataset.
fun	name of function to simulate data, default "rnorm".
...	further arguments passed to ‘fun‘.

Value

assay_list a list of data matrices with rownames (features) specified.

Examples

```
set.seed(2021)
assay_list <- mockMosaicData()
lapply(assay_list, dim)

# simulate data from another distribution
assay_list <- mockMosaicData(fun = "rnbinom", size = 5, prob = 0.5)
lapply(assay_list, dim)
```

mosaicDataTopology *mosaicDataTopology*

Description

Generate mosaic data topology network as an igraph object.

Usage

```
mosaicDataTopology(assay_list)
```

Arguments

assay_list	a list of data matrices with rownames (features) specified.

Value

igraph weighted network with nodes corresponding to assay_list elements, and edges present if the matrices share at least one rowname. Edge weights correspond to the number of shared rownames among data matrices.

Examples

```
set.seed(2021)
assay_list <- mockMosaicData()
mdt <- mosaicDataTopology(assay_list)
mdt
plot(mdt)
```

mosaicDataUpSet *mosaicDataUpSet*

Description

Plots feature overlaps of mosaic data as an UpSet plot.

Usage

```
mosaicDataUpSet(assay_list, plot = FALSE, ...)
```

Arguments

assay_list	a list of data matrices with rownames (features) specified.
plot	logical (default FALSE) whether the UpSet plot should be printed.
...	further arguments passed to ‘upset‘ from the ‘UpSetR‘ package.

Value

UpSet object displaying degree of overlap of rownames (features) among each of the data matrices in `assay_list`. Set bars correspond to the number of cells/samples present in each data matrix.

Examples

```
set.seed(2021)
assay_list <- mockMosaicData()
lapply(assay_list, dim)
mosaicDataUpSet(assay_list)

# additional arguments from UpSetR::upset()
mosaicDataUpSet(assay_list, empty.intersections = TRUE)
```

---

reWeightEmbedding *Re-weight StabMap embedding*

Description

Re-weights embedding according to given weights for each reference dataset. This gives more or less weighting to each contributing dataset and method (PCA or LDA),

Usage

```
reWeightEmbedding(embedding, weights = NULL, factor = 1e+06)
```

Arguments

`embedding`	Joint embedding as output from stabMap.
`weights`	(optional) named numeric vector giving relative weights for each reference dataset.
`factor`	numeric multiplicative value to offset near-zero values.

Value

matrix of same dimensions as ‘embedding‘.

Examples

```
set.seed(2021)
assay_list <- mockMosaicData()
lapply(assay_list, dim)

# specify which datasets to use as reference coordinates
reference_list <- c("D1", "D3")

# specify some sample labels to distinguish using linear discriminant
# analysis (LDA)
```

```
labels_list <- list(
  D1 = rep(letters[1:5], length.out = ncol(assay_list[["D1"]]))
)

# stabMap
out <- stabMap(assay_list,
  reference_list = reference_list,
  labels_list = labels_list,
  ncomponentsReference = 20,
  ncomponentsSubset = 20
)

# look at the scale of each component and discriminant
boxplot(out, las = 2, outline = FALSE)

# re-weight embedding for less contribution from LDs and equal contribution
# from PCs of both references
out_reweighted <- reWeightEmbedding(
  out,
  weights = c("D1_LD" = 0.5, "D1_PC" = 1, "D3_PC" = 1)
)

# look at the new scale of each component and discriminant
boxplot(out_reweighted, las = 2, outline = FALSE)
```

---

stabMap *Stabilised mosaic single cell data integration using unshared features*

Description

stabMap performs mosaic data integration by first building a mosaic data topology, and for each reference dataset, traverses the topology to project and predict data onto a common principal component (PC) or linear discriminant (LD) embedding.

Usage

```
stabMap(
  assay_list,
  labels_list = NULL,
  reference_list = NULL,
  reference_features_list = lapply(assay_list, rownames),
  reference_scores_list = NULL,
  ncomponentsReference = 50,
  ncomponentsSubset = 50,
  suppressMessages = TRUE,
  projectAll = FALSE,
  restrictFeatures = FALSE,
  maxFeatures = 1000,
```

```
  plot = TRUE,
  scale.center = TRUE,
  scale.scale = TRUE,
  SE_assay_names = "logcounts",
  BPPARAM = SerialParam(),
  verbose = TRUE
)
```

Arguments

assay_list	A list of data matrices with rownames (features) specified.
labels_list	(optional) named list containing cell labels
reference_list	Named list containing logical values whether the data matrix should be considered as a reference dataset, alternatively a character vector containing the names of the reference data matrices. If NULL, defaults to: sapply(names(assay_list), function(x) TRUE, simplify = FALSE)
reference_features_list	List of features to consider as reference data (default is all available features).
reference_scores_list	Named list of reference scores (default NULL). If provided, matrix of cells (rows with rownames given) and dimensions (columns with colnames given) are used as the reference low-dimensional embedding to target, as opposed to performing PCA or LDA on the input reference data.
ncomponentsReference	Number of principal components for embedding reference data, given either as an integer or a named list for each reference dataset.
ncomponentsSubset	Number of principal components for embedding query data prior to projecting to the reference, given either as an integer or a named list for each reference dataset.
suppressMessages	Logical whether to suppress messages (default TRUE).
projectAll	Logical whether to re-project reference data along with query (default FALSE).
restrictFeatures	logical whether to restrict to features used in dimensionality reduction of reference data (default FALSE). Overall it's recommended that this be FALSE for single-hop integrations and TRUE for multi-hop integrations.
maxFeatures	Maximum number of features to consider for predicting principal component scores (default 1000).
plot	Logical whether to plot mosaic data UpSet plot and mosaic data topology networks (default TRUE).
scale.center	Logical whether to re-center data to a mean of 0 (default FALSE).
scale.scale	Logical whether to re-scale data to standard deviation of 1 (default FALSE).
SE_assay_names	Either a string indicating the name of the assays for the SummarizedExperiment objects in assay_list or a named list of assay names, where the names corrispond to the names SE objects in assay_list (default "logcounts")

BPPARAM	a BiocParallelParam object specifying how parallelisation should be performed
verbose	Logical whether console output is provided (default TRUE)

Value

matrix containing common embedding with rows corresponding to cells, and columns corresponding to PCs or LDs for reference dataset(s).

Examples

```
set.seed(2021)
assay_list <- mockMosaicData()
lapply(assay_list, dim)

# specify which datasets to use as reference coordinates
reference_list <- c("D1", "D3")

# specify some sample labels to distinguish using linear discriminant
# analysis (LDA)
labels_list <- list(
  D1 = rep(letters[1:5], length.out = ncol(assay_list[["D1"]]))
)

# examine the topology of this mosaic data integration
mosaicDataUpSet(assay_list)
plot(mosaicDataTopology(assay_list))

# stabMap
out <- stabMap(assay_list,
  reference_list = reference_list,
  labels_list = labels_list,
  ncomponentsReference = 20,
  ncomponentsSubset = 20
)

head(out)
```

Index

adaptiveKNN, 2

classifyEmbedding, 3

getAdaptiveK, 5

imputeEmbedding, 6

mockMosaicData, 8
mosaicDataTopology, 9
mosaicDataUpSet, 9

reWeightEmbedding, 10

stabMap, 11