Package ‘lemur’

February 19, 2024

Type Package

Title Latent Embedding Multivariate Regression

Version 1.0.5

Description Fit a latent embedding multivariate regression (LEMUR) model to multi-condition single-cell data. The model provides a parametric description of single-cell data measured with complex experimental designs. The parametric model is used to (1) align conditions, (2) predict log fold changes between conditions for all cells, and (3) identify cell neighborhoods with consistent log fold changes. For those neighborhoods, a pseudobulked differential expression test is conducted to assess which genes are significantly changed.

URL https://github.com/const-ae/lemur

BugReports https://github.com/const-ae/lemur/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Imports stats, utils, irlba, methods, SingleCellExperiment, SummarizedExperiment, rlang, vctrs, glmGamPoi (>= 1.12.0), BiocGenerics, S4Vectors, Matrix, DelayedMatrixStats, HDF5Array, MatrixGenerics, matrixStats, Rcpp, harmony, limma, BiocNeighbors

Suggests testthat (>= 3.0.0), tidyverse, uwot, dplyr, edgeR, knitr, markdown, BiocStyle

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 4.3)

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

biocViews Transcriptomics, DifferentialExpression, SingleCell, DimensionReduction, Regression

VignetteBuilder knitr
R topics documented:

.DollarNames.lemur_fit ........................................... 3
align_harmony ......................................................... 3
align_impl ........................................................... 5
find_de_neighborhoods ............................................ 5
fold_left ............................................................. 8
glioblastoma_example_data ....................................... 9
grassmann_geodesic_regression .................................. 9
grassmann_lm ........................................................ 10
harmony_new_object ................................................ 10
lemur ................................................................. 11
lemur_fit-class ..................................................... 12
mply_dbl ............................................................. 14
one_hot_encoding .................................................... 14
predict.lemur_fit ................................................... 15
project_on_lemur_fit ............................................... 16
pseudo-inverse ...................................................... 17
recursive_least_squares .......................................... 18
reexports ............................................................ 18
residuals.lemur_fit-method ....................................... 19
ridge_regression .................................................... 20
stack_slice .......................................................... 20
test_de ............................................................... 21
test_global .......................................................... 22
%zero_dom_mat_mult% ............................................. 23

Index ............................................................... 24
\textit{Access values from a \texttt{lemur\_fit}}

\section*{Description}

Access values from a \texttt{lemur\_fit}

\section*{Usage}

\begin{verbatim}
## S3 method for class 'lemur_fit'
.DollarNames(x, pattern = "")

## S4 method for signature 'lemur_fit'
x$name

## S4 replacement method for signature 'lemur_fit'
x$name <- value
\end{verbatim}

\section*{Arguments}

\begin{itemize}
  \item \texttt{x} \hspace{2cm} the \texttt{lemur\_fit}
  \item \texttt{pattern} \hspace{2cm} the pattern from looking up potential values interactively
  \item \texttt{name} \hspace{2cm} the name of the value behind the dollar
  \item \texttt{value} \hspace{2cm} the replacement value. This only works for \texttt{colData} and \texttt{rowData}.
\end{itemize}

\section*{Value}

The respective value stored in the \texttt{lemur\_fit} object.

\section*{See Also}

\texttt{lemur\_fit} for more documentation on the accessor functions.

\section*{align_harmony}

\textit{Enforce additional alignment of cell clusters beyond the direct differential embedding}

\section*{Description}

Enforce additional alignment of cell clusters beyond the direct differential embedding
align_harmony

align_harmony(
  fit,
  design = fit$alignment_design,
  ridge_penalty = 0.01,
  max_iter = 10,
  ...
  verbose = TRUE
)

align_by_grouping(
  fit,
  grouping,
  design = fit$alignment_design,
  ridge_penalty = 0.01,
  preserve_position_of_NAs = FALSE,
  verbose = TRUE
)

Arguments

- **fit**: a lemur_fit object
- **design**: a specification of the design (matrix or formula) that is used for the transformation. Default: fit$design_matrix
- **ridge_penalty**: specification how much the flexibility of the transformation should be regularized. Default: 0.01
- **max_iter**: argument specific for align_harmony. The number of iterations. Default: 10
- **...**: additional parameters that are passed on to relevant functions
- **verbose**: Should the method print information during the fitting. Default: TRUE.
- **grouping**: argument specific for align_by_grouping. Either a vector which assigns each cell to one group or a matrix with ncol(fit) columns where the rows are a soft-assignment to a cluster (i.e., columns sum to 1). NA's are allowed.
- **preserve_position_of_NAs**: argument specific for align_by_grouping. Boolean flag to decide if NAs in the grouping mean that these cells should stay where they are (if possible) or if they are free to move around. Default: FALSE

Value

The fit object with the updated fit$embedding and fit$alignment_coefficients.

Examples

data(glioblastoma_example_data)
fit <- lemur(glioblastoma_example_data, design = ~ patient_id + condition,
  n_emb = 5, verbose = FALSE)
# Creating some grouping for illustration
cell_types <- sample(c("tumor cell", "neuron", "leukocyte"), size = ncol(fit), replace = TRUE)
fitted <- align_by_grouping(fit, grouping = cell_types)

# Alternatively, use harmony to automatically group cells
fitted2 <- align_harmony(fit)
fitted2

# The alignment coefficients are a 3D array
fitted2$alignment_coefficients

---

### align_impl

**Align the points according to some grouping**

**Description**

Align the points according to some grouping

**Usage**

```r
align_impl(
    embedding,
    grouping,
    design_matrix,
    ridge_penalty = 0.01,
    preserve_position_of_NAs = FALSE,
    calculate_new_embedding = TRUE
)
```

**Value**

A list with the new embedding and the coefficients

---

### find_de_neighborhoods

**Find differential expression neighborhoods**

**Description**

Find differential expression neighborhoods
Usage

```r
find_de_neighborhoods(
  fit,
  group_by,
  contrast = fit$contrast,
  selection_procedure = c("zscore", "contrast"),
  directions = c("random", "contrast", "axis_parallel"),
  min_neighborhood_size = 50,
  de_mat = SummarizedExperiment::assays(fit)[["DE"]],
  test_data = fit$test_data,
  test_data_col_data = NULL,
  test_method = c("glmGamPoi", "edgeR", "limma", "none"),
  continuous_assay_name = fit$use_assay,
  count_assay_name = "counts",
  size_factor_method = NULL,
  design = fit$design,
  alignment_design = fit$alignment_design,
  add_diff_in_diff = TRUE,
  make_neighborhoods_consistent = FALSE,
  skip_confounded_neighborhoods = FALSE,
  control_parameters = NULL,
  verbose = TRUE
)
```

Arguments

- **fit**: the lemur_fit generated by `lemur()`
- **group_by**: If the independent_matrix is provided, `group_by` defines how the pseudobulks are formed.
- **contrast**: a specification which contrast to fit. This defaults to the contrast argument that was used for `test_de` and is stored in `fit$contrast`.
- **selection_procedure**: specify the algorithm that is used to select the neighborhoods for each gene. Broadly, `selection_procedure = "zscore"` is faster but less precise than `selection_procedure = "contrast"`.
- **directions**: a string to define the algorithm to select the direction onto which the cells are projected before searching for the neighborhood. `directions = "random"` produces denser neighborhoods, whereas `directions = "contrast"` has usually more power. Alternatively, this can also be a matrix with one direction for each gene (i.e., a matrix of size `nrow(fit) * fit$n_embedding`).
- **min_neighborhood_size**: the minimum number of cells per neighborhood. Default: 50.
- **de_mat**: the matrix with the differential expression values and is only relevant if `selection_procedure = "zscore"` or `directions = "random"`. Defaults to an assay called "DE" that is produced by `lemur::test_de()`.
**find_de_neighborhoods**

- **test_data**
  a SummarizedExperiment object or a named list of matrices. The data is used to test if the neighborhood inferred on the training data contain a reliable significant change. If `test_method` is "glmGamPoi" or "edgeR" a test using raw counts is conducted and two matching assays are needed: (1) the continuous assay (with `continuous_assay_name`) is projected onto the LEMUR fit to find the latent position of each cell and (2) the count assay (count_assay_name) is used for forming the pseudobulk. If `test_method` == "limma", only the continuous assay is needed.
  
  The arguments defaults to the test data split of when calling `lemur()`.

- **test_data_col_data**
  additional column data for the `test_data` argument.

- **test_method**
  choice of test for the pseudobulked differential expression. `glmGamPoi` and `edgeR` work on a count assay. `limma` works on the continuous assay.

- **continuous_assay_name, count_assay_name**
  the assay or list names of independent_data.

- **size_factor_method**
  Set the procedure to calculate the size factor after pseudobulking. This argument is only relevant if `test_method` is "glmGamPoi" or "edgeR". If fit is subsetted, using a vector with the sequencing depth per cell ensures reasonable results. Default: `NULL` which means that `colSums(assay(fit$test_data, count_assay_name))` is used.

- **design, alignment_design**
  the design to use for the fit. Default: `fit$design`

- **add_diff_in_diff**
  a boolean to specify if the log-fold change (plus significance) of the DE in the neighborhood against the DE in the complement of the neighborhood is calculated. If `TRUE`, the result includes three additional columns starting with "did_" short for difference-in-difference. Default: `TRUE`.

- **make_neighborhoods_consistent**
  Include cells from outside the neighborhood if they are at least 10 times in the k-nearest neighbors of the cells inside the neighborhood. Secondly, remove cells from the neighborhood which are less than 10 times in the k-nearest neighbors of the other cells in the neighborhood. Default `FALSE`.

- **skip_confounded_neighborhoods**
  Sometimes the inferred neighborhoods are not limited to a single cell state; this becomes problematic if the cells of the conditions compared in the contrast are unequally distributed between the cell states. Default: `FALSE`.

- **control_parameters**
  named list with additional parameters passed to underlying functions.

- **verbose**
  Should the method print information during the fitting. Default: `TRUE`.

**Value**

- a data frame with one entry per gene

- **name**
  The gene name.
neighborhood A list column where each element is a vector with the cell names included in that neighborhood.
n_cells the number of cells in the neighborhood (lengths(neighborhood)).

sel_statistic The statistic that is maximized by the selection_procedure.
pval, adj_pval, t_statistic, lfc The p-value, Benjamini-Hochberg adjusted p-value (FDR), the t-statistic, and the log2 fold change of the differential expression test defined by contrast for the cells inside the neighborhood (calculated using test_method). Only present if test_data is not NULL.
did_pval, did_adj_pval, did_lfc The measurement if the differential expression of the cells inside the neighborhood is significantly different from the differential expression of the cells outside the neighborhood. Only present if add_diff_in_diff = TRUE.

Examples

data(glioblastoma_example_data)
fit <- lemur(glioblastoma_example_data, design = ~ patient_id + condition,
  n_emb = 5, verbose = FALSE)

# Optional alignment
# fit <- align_harmony(fit)
fit <- test_de(fit, contrast = cond(condition = "panobinostat") - cond(condition = "ctrl"))
nei <- find_de_neighborhoods(fit, group_by = vars(condition, patient_id))
head(nei)

fold_left

fold_left(init)
fold_right(init)

Description
Fold left over a sequence
Fold right over a sequence

Usage
fold_left(init)
fold_right(init)

Arguments
init initial value. If not specified NULL
x the sequence to iterate over
FUN a function with first argument named elem and second argument named accum

Value
The final value of accum.
glioblastoma_example_data

Examples

```r
## Not run:
# This produces ...
fold_left(0)(1:10, (elem, accum) accum + elem)
# ... the same as
sum(1:10)
## End(Not run)
```

glioblastoma_example_data

The `glioblastoma_example_data` dataset

Description

The dataset is a `SingleCellExperiment` object subset to 5,000 cells and 300 genes. The `colData` contain an entry for each cell from which patient it came and to which treatment condition it belonged ("ctrl" or "panobinostat").

Details

The original data was collected by Zhao et al. (2021).

Value

A `SingleCellExperiment` object.

References


grassmann_geodesic_regression

`Solve \(d(P, \exp_p(V \cdot x))^2\)` for \(V\)

Description

Solve \(d(P, \exp_p(V \cdot x))^2\) for \(V\)
Usage

\[
\text{grassmann\_geodesic\_regression(}
\text{coordsystems, design, base\_point, weights = 1, tangent\_regression = FALSE)}
\]

Value

A three-dimensional array with the coefficients \( V \).

\[
\text{grassmann\_lm} \quad \text{Solve } ||Y - \exp_p(V \ast x) Y||^2_2 \text{ for } V
\]

Description

Solve \( ||Y - \exp_p(V \ast x) Y||^2_2 \) for \( V \)

Usage

\[
\text{grassmann\_lm(data, design, base\_point, tangent\_regression = FALSE)}
\]

Value

A three-dimensional array with the coefficients \( V \).

\[
\text{harmony\_new\_object} \quad \text{Create an arbitrary Harmony object so that I can modify it later}
\]

Description

Create an arbitrary Harmony object so that I can modify it later

Usage

\[
\text{harmony\_new\_object()}
\]

Value

The full \texttt{harmony} object (R6 reference class type).
lemur

Main function to fit the latent embedding multivariate regression (LEMUR) model

Description

Main function to fit the latent embedding multivariate regression (LEMUR) model

Usage

lemur(
  data,
  design = ~1,
  col_data = NULL,
  n_embedding = 15,
  linear_coefficient_estimator = c("linear", "cluster_median", "zero"),
  use_assay = "logcounts",
  test_fraction = 0.2,
  ...,
  verbose = TRUE
)

Arguments

data a matrix with observations in the columns and features in the rows. Or a SummarizedExperiment / SingleCellExperiment object

design a formula referring to global objects or column in the colData of data and col_data argument

col_data an optional data frame with ncol(data) rows.

n_embedding the dimension of the $k$-plane that is rotated through space.

linear_coefficient_estimator specify which estimator is used to center the conditions. "linear" runs simple regression it works well in many circumstances but can produce poor results if the composition of the cell types changes between conditions (e.g., one cell type disappears). "cluster_median" works similar as "linear" but is robust against compositional changes. "zero" skips the centering step which is also robust against compositional changes. However, expression changes affecting all cells equally are not regressed out.

use_assay if data is a SummarizedExperiment / SingleCellExperiment object, which assay should be used.

test_fraction the fraction of cells that are split of before the model fit to keep an independent set of test observations. Alternatively, a logical vector of length ncol(data). Default: 20% (0.2).

... additional parameters that are passed on to the internal function lemur_impl.

verbose Should the method print information during the fitting. Default: TRUE.
Value
An object of class lemur_fit which extends SingleCellExperiment. Accordingly, all functions that work for sce’s also work for lemur_fit’s. In addition, we give easy access to the fitted values using the dollar notation (e.g., fit$embedding). For details see the lemur_fit help page.

References

See Also
align_by_grouping, align_harmony, test_de, find_de_neighborhoods

Examples
data(glioblastoma_example_data)
fit <- lemur(glioblastoma_example_data, design = ~ patient_id + condition, n_emb = 5)
fit

Description
The lemur_fit class extends SingleCellExperiment and provides additional accessors to get the values of the values produced by lemur.

Usage
```r
## S4 method for signature 'lemur_fit,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'lemur_fit'
design(object)
```

Arguments
```r
x, i, j, ..., drop
   the lemur_fit object and indices for the [ subsetting operator
object
   the lemur_fit object for the BiocGenerics::design generic
```
Details

To access the values produced by lemur, use the dollar notation ($):

- `fit$n_embedding`: the number of embedding dimensions.
- `fit$design`: the specification of the design in `lemur`. Usually this is a `stats::formula`.
- `fit$base_point`: a matrix (nrow(fit) * fit$n_embedding) with the base point for the Grassmann exponential map.
- `fit$coefficients`: a three-dimensional tensor (nrow(fit) * fit$n_embedding * ncol(fit$design_matrix)) with the coefficients for the exponential map.
- `fit$embedding`: a matrix (fit$n_embedding * ncol(fit)) with the low dimensional position for each cell.
- `fit$design_matrix`: a matrix with covariates for each cell (ncol(fit) * ncol(fit$design_matrix)).
- `fit$linear_coefficients`: a matrix (nrow(fit) * ncol(fit$design_matrix)) with the coefficients for the linear regression.
- `fit$alignment_coefficients`: a 3D tensor with the coefficients for the alignment (fit$n_embedding * fit$n_embedding * ncol(fit$design_matrix))
- `fit$alignment_design`: an alternative design specification for the alignment. This is typically a `stats::formula`.
- `fit$alignment_design_matrix`: an alternative design matrix specification for the alignment.
- `fit$contrast`: a parsed version of the contrast specification from the `test_de` function or `NULL`.
- `fit$colData`: the column annotation `DataFrame`.
- `fit$rowData`: the row annotation `DataFrame`.

Value

An object of class `lemur_fit`.

See Also

- `lemur`, `predict`, `residuals`

Examples

```r
# The easiest way to make a lemur_fit object, is to call `lemur``
data(glioblastoma_example_data)
fit <- lemur(glioblastoma_example_data, design = ~ patient_id + condition,
             n_emb = 5, verbose = FALSE)

fit$n_embedding
fit$embedding[,1:10]
fit$n_embedding
fit$embedding[,1:10]
fit$design_matrix[1:10,,]
fit$coefficients[1:3,,]
```
### mply_dbl

*Iterating function that returns a matrix*

#### Description
The length of `x` determines the number of rows. The length of `FUN(x[i])` determines the number of columns. Must match `ncol`.

#### Usage
```r
cmply_dbl(x, FUN, ncol = 1, ...)
stack_rows(x)
stack_cols(x)
```

#### Arguments
- `x`: the sequence that is mapped to a matrix
- `FUN`: the function that returns a vector of length `ncol`
- `ncol`: the length of the output vector
- `...`: additional arguments that are passed to `FUN`

#### Value
A matrix with `length(x) / nrow(x)` rows and `ncol` columns. For `mply_dbl` the number of columns depends on the output of `FUN`.

#### Functions
- `stack_rows()`: Each list element becomes a row in a matrix
- `stack_cols()`: Each list element becomes a row in a matrix

---

### one_hot_encoding

*Take a vector and convert it to a one-hot encoded matrix*

#### Description
Take a vector and convert it to a one-hot encoded matrix

#### Usage
```r
one_hot_encoding(groups)
```

#### Value
A matrix with `length(unique(groups))` rows and `length(groups)` columns.
**predict.lemur_fit**  
*Predict values from lemur_fit object*

---

### Description

Predict values from `lemur_fit` object

### Usage

```r
## S3 method for class 'lemur_fit'
predict(
  object,
  newdata = NULL,
  newdesign = NULL,
  newcondition = NULL,
  embedding = object$embedding,
  with_linear_model = TRUE,
  with_embedding = TRUE,
  with_alignment = TRUE,
  ...
)
```

### Arguments

- **object**: an `lemur_fit` object
- **newdata**: a data.frame which passed to `model.matrix` with design to make the `newdesign` matrix
- **newdesign**: a matrix with the covariates for which the output is predicted. If `NULL`, the `object$design_matrix` is used. If it is a vector it is repeated `ncol(embedding)` times to create a design matrix with the same entry for each cell.
- **newcondition**: an unquoted expression with a call to `cond()` specifying the covariates of the prediction. See the contrast argument in `test_de` for more details. Note that combinations of multiple calls to `cond()` are not allowed (e.g., `cond(a = 1) - cond(a = 2)`). If specified, `newdata` and `newdesign` are ignored.
- **embedding**: the low-dimensional cell position for which the output is predicted.
- **with_linear_model**: a boolean to indicate if the linear regression offset is included in the prediction.
- **with_embedding**: a boolean to indicate if the embedding contributes to the output.
- **with_alignment**: a boolean to indicate if the alignment effect is removed from the output.
- **...**: additional parameters passed to `predict_impl`.

### Value

A matrix with the same dimension `nrow(object) * nrow(newdesign)`. 

project_on_lemur_fit

See Also

residuals

Examples

data(glioblastoma_example_data)
fit <- lemur(glioblastoma_example_data, design = ~ patient_id + condition,
            n_emb = 5, verbose = FALSE)
pred <- predict(fit)
pred_ctrl <- predict(fit, newdesign = c(1, 0, 0, 0, 0, 0))
pred_trt <- predict(fit, newdesign = c(1, 0, 0, 0, 0, 1))
# This is the same as the test_de result
fit <- test_de(fit, cond(condition = "panobinostat") - cond(condition = "ctrl"))
all.equal(SummarizedExperiment::assay(fit, "DE"), pred_trt - pred_ctrl,
          check.attributes = FALSE)

project_on_lemur_fit

Project new data onto the latent spaces of an existing lemur fit

Description

Project new data onto the latent spaces of an existing lemur fit

Usage

project_on_lemur_fit(
  fit,  # an lemur_fit object
  data,  # a matrix with observations in the columns and features in the rows. Or a SummarizedExperiment
          # /SingleCellExperiment object. The features must match the features in fit.
  col_data = NULL,  # col_data an optional data frame with ncol(data) rows.
  use_assay = "logcounts",  # if data is a SummarizedExperiment / SingleCellExperiment object, which
  design = fit$design,  # assay should be used.
  alignment_design = fit$alignment_design,
  return = c("matrix", "lemur_fit")
)

Arguments

fit

data

col_data

use_assay
design, alignment_design

the design formulas or design matrices that are used to project the data on the correct latent subspace. Both default to the designs from the fit object.

return

which data structure is returned.

Value

Either a matrix with the low-dimensional embeddings of the data or an object of class lemur_fit wrapping that embedding.

Examples

data(glioblastoma_example_data)

subset1 <- glioblastoma_example_data[,1:2500]
subset2 <- glioblastoma_example_data[,2501:5000]

fit <- lemur(subset1, design = ~ patient_id + condition, n_emb = 5,
             test_fraction = 0, verbose = FALSE)

# Returns a `lemur_fit` object with the projection of `subset2`
fit2 <- project_on_lemur_fit(fit, subset2, return = "lemur_fit")
fit2

pseudoinverse

Moore-Penrose pseudoinverse calculated via SVD

Description

In the simplest case, the pseudoinverse is

\[ X^+ = (X^T X)^{-1} X^T. \]

Usage

pseudoinverse(X)

Arguments

- **x**
  - a matrix \( X \)

Details

To handle the more general case, the pseudoinverse can expressed using a SVD \( X = UDV^T \):

\[ X^+ = VD^{-1}U^T \]
Value

The matrix $X^+$. 

---

**recursive_least_squares**

*Iteratively calculate the least squares solution*

---

**Description**

Both functions are for testing purposes. There is a faster implementation called cum_brls_which_abs_max.

**Usage**

```r
recursive_least_squares(y, X)
bulked_recursive_least_squares_contrast(
  y,
  X,
  group,
  contrast,
  ridge_penalty = 1e-06
)
```

**Arguments**

- `y` a vector with observations
- `X` a design matrix

**Value**

a matrix where column $i$ is the solution to $y[1:i] \sim X[1:i,]$.

---

**reexports**

*Objects exported from other packages*

---

**Description**

These objects are imported from other packages. Follow the links below to see their documentation.

- `glmGamPoi vars`

**Value**

see `glmGamPoi::vars`. 
Examples

```r
# `vars` quotes expressions (just like in dplyr)
vars(condition, sample)
```

Description

Predict values from `lemur_fit` object

Usage

```r
## S4 method for signature 'lemur_fit'
residuals(object, with_linear_model = TRUE, with_embedding = TRUE, ...)
```

Arguments

- `object`: an `lemur_fit` object
- `with_linear_model`: a boolean to indicate if the linear regression offset is included in the prediction.
- `with_embedding`: a boolean to indicate if the embedding contributes to the output.
- `...`: ignored.

Value

A matrix with the same dimension `dim(object)`.

See Also

`predict.lemur_fit`

Examples

```r
data(glioblastoma_example_data)
fit <- lemur(glioblastoma_example_data, design = ~ patient_id + condition,
            n_emb = 5, verbose = FALSE)

resid <- residuals(fit)
dim(resid)
```
Ridge regression

Description

The function does not treat the intercept special.

Usage

```r
ridge_regression(Y, X, ridge_penalty = 0, weights = rep(1, nrow(X)))
```

Arguments

- **Y**: the observations matrix (features x samples)
- **X**: the design matrix (samples x covariates)
- **ridge_penalty**: a numeric vector or matrix of size (covariates or covariates x covariates respectively)
- **weights**: a vector of observation weights

Value

The matrix of coefficients.

---

Stack slice

Make a cube from a list of matrices

Description

The length of the list will become the third dimension of the cube.

Usage

```r
stack_slice(x)
destack_slice(x)
```

Arguments

- **x**: a list of vectors/matrices that are stacked

Value

A three-dimensional array.

Functions

- `destack_slice()`: Make a list of matrices from a cube
test_de  Predict log fold changes between conditions for each cell

Description

Predict log fold changes between conditions for each cell

Usage

test_de(
  fit,
  contrast,
  embedding = NULL,
  consider = c("embedding+linear", "embedding", "linear"),
  new_assay_name = "DE"
)

Arguments

fit  the result of calling lemur()
contrast  Specification of the contrast: a call to cond() specifying a full observation (e.g. cond(treatment = "A", sex = "male") - cond(treatment = "C", sex = "male") to compare treatment A vs C for male observations). Unspecified factors default to the reference level.
embedding  matrix of size n_embedding × n that specifies where in the latent space the differential expression is tested. It defaults to the position of all cells from the original fit.
consider  specify which part of the model are considered for the differential expression test.
new_assay_name  the name of the assay added to the fit object. Default: "DE".

Value

If is.null(embedding) the fit object with a new assay called "DE". Otherwise return a matrix with the differential expression values.

See Also

find_de_neighborhoods

Examples

library(SummarizedExperiment)
library(SingleCellExperiment)
data(glioblastoma_example_data)
fit <- lemur(glioblastoma_example_data, design = ~ patient_id + condition,
test_global

Differential embedding for each condition

Description
Differential embedding for each condition

Usage

```r
test_global(
  fit,
  contrast,
  reduced_design = NULL,
  consider = c("embedding+linear", "embedding", "linear"),
  variance_est = c("analytical", "resampling", "none"),
  verbose = TRUE,
  ...)
```

Arguments

- `fit`: the result of calling `lemur()`
- `contrast`: Specification of the contrast: a call to `cond()` specifying a full observation (e.g. `cond(treatment = "A", sex = "male") - cond(treatment = "C", sex = "male")` to compare treatment A vs C for male observations). Unspecified factors default to the reference level.
- `reduced_design`: an alternative specification of the null hypothesis.
- `consider`: specify which part of the model are considered for the differential expression test.
- `variance_est`: How or if the variance should be estimated. 'analytical' is only compatible with `consider = "linear"`. 'resampling' is the most flexible (to adapt the number of resampling iterations, set `n_resampling_iter`). Default: 100
- `verbose`: should the method print information during the fitting. Default: TRUE.
- `...`: additional arguments.
Value

a data.frame

**%zero_dom_mat_mult%**  
*Helper function that makes sure that NA * 0 = 0 in matrix multiply*

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helper function that makes sure that NA * 0 = 0 in matrix multiply</td>
</tr>
</tbody>
</table>

**Usage**

X %zero_dom_mat_mult% Y

**Arguments**

<table>
<thead>
<tr>
<th>X</th>
<th>a matrix of size n*m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>a matrix of size m*p</td>
</tr>
</tbody>
</table>

**Value**

a matrix of size n*p
Index

* internal

%zero_dom_mat_mult%, 23
align_impl, 5
fold_left, 8
gioblastoma_example_data, 9
glmGamPoi::vars, 18
grassmann_geodesic_regression, 9
gassmann_lm, 10
harmony_new_object, 10
mply_dbl, 14
one_hot_encoding, 14
pseudoinverse, 17
recursive_least_squares, 18
reexports, 18
ridge_regression, 20
stack_slice, 20
.DollarNames.lemur_fit, 3
.lemur_fit (lemur_fit-class), 12
[,]lemur_fit,ANY,ANY,ANY-method
(lemur_fit-class), 12
$,lemur_fit-method
(.DollarNames.lemur_fit), 3
$<-,lemur_fit-method
(.DollarNames.lemur_fit), 3
%zero_dom_mat_mult%, 23

align_by_grouping, 12
align_by_grouping (align_harmony), 3
align_harmony, 3, 12
align_impl, 5

BiocGenerics::design, 12
bulked_recursive_least_squares_contrast
(recursive_least_squares), 18
design,lemur_fit-method
(lemur_fit-class), 12
destack_slice(stack_slice), 20
dollar_methods
(.DollarNames.lemur_fit), 3
find_de_neighborhoods, 5, 12, 21
fold_left, 8
fold_right (fold_left), 8
gioblastoma_example_data, 9
glmGamPoi::vars, 18
grassmann_geodesic_regression, 9
gassmann_lm, 10
harmony, 10
harmony_new_object, 10
lemur, 11, 12, 13
lemur(), 21, 22
lemur_fit,3, 12
lemur_fit (lemur_fit-class), 12
lemur_fit-class, 12
model.matrix, 15
mply_dbl, 14
one_hot_encoding, 14
predict, 13
predict.lemur_fit, 15, 19
project_on_lemur_fit, 16
pseudoinverse, 17
recursive_least_squares, 18
reexports, 18
residuals, 13, 16
residuals,lemur_fit-method, 19
ridge_regression, 20
SingleCellExperiment, 9, 12
stack_cols (mply_dbl), 14
stack_rows (mply_dbl), 14
stack_slice, 20
stats::formula, 13
test_de, 12, 15, 21
test_global, 22
vars, 18
vars (reexports), 18

24