Package 'anglemania'

November 12, 2026

URL https://github.com/BIMSBbioinfo/anglemania/

Title Feature Extraction for scRNA-seq Dataset Integration

Version 1.3.0

Description anglemania extracts genes from multi-batch scRNA-seq experiments for downstream dataset integration. It shows improvement over the conventional usage of highly-variable genes for many integration tasks. We leverage gene-gene correlations that are stable across batches to identify biologically informative genes which are less affected by batch effects. Currently, its main use is for single-cell RNA-seq dataset integration, but it can be applied for other multi-batch downstream analyses such as NMF.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

LinkingTo Rcpp, rmio, bigstatsr

Depends R (>= 4.5.0)

Imports bigparallelr, bigstatsr, checkmate, digest, dplyr, Matrix, pbapply, S4Vectors, SingleCellExperiment, stats, SummarizedExperiment, tidyr, withr

Suggests batchelor, BiocStyle, bluster, knitr, magick, matrixStats, patchwork, RcppArmadillo, rmarkdown, scater, scran, Seurat, splatter, testthat (>= 3.0.0), UpSetR

VignetteBuilder knitr

biocViews SingleCell, BatchEffect, MultipleComparison, FeatureExtraction

BiocType Software

Config/testthat/edition 3

BugReports https://github.com/BIMSBbioinfo/anglemania/issues

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

git_branch devel

git_last_commit 097e328

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-11-12

Author Aaron Kollotzek [aut, cre] (ORCID:
<https://orcid.org/0009-0009-7142-4015>),
Vedran Franke [aut] (ORCID: <https://orcid.org/0000-0003-3606-6792>),
Artem Baranovskii [aut],
Altuna Akalin [aut],
SFB1588 [fnd] (Funded by the DFG – Deutsche Forschungsgemeinschaft)

Maintainer Aaron Kollotzek <aaron.kollotzek@mdc-berlin.de>

Contents

adapted_reexports 2
anglemania 4
anglemania_utils 7
extract_angles 9
factorise 10
permute_nonzero 12
sce_example 12
select_genes 13
statistics 15

Index **17**

`adapted_reexports` *Adapted Reexports from bigstatsr*

Description

These are functions that were adapted from the **bigstatsr** package, modified to suppress certain warnings and errors (e.g., from zero variance in scaling).

Usage

```
CutBySize(m, block.size, nb = ceiling(m/block.size))

big_crossprodSelf_no_warning(
  X,
  fun.scaling = big_scale_no_warning(center = FALSE, scale = FALSE),
  ind.row = bigstatsr::rows_along(X),
  ind.col = bigstatsr::cols_along(X),
  block.size = bigstatsr::block_size(nrow(X)),
  backingfile = tempfile(tmpdir = getOption("FBM.dir"))
)

big_cor_no_warning(
  X,
  ind.row = bigstatsr::rows_along(X),
  ind.col = bigstatsr::cols_along(X),
  block.size = bigstatsr::block_size(nrow(X)),
  backingfile = tempfile(tmpdir = getOption("FBM.dir"))
)

big_scale_no_warning(center = TRUE, scale = TRUE)
```

Arguments

`m`	An integer specifying the length of the input to split into intervals.
`block.size`	An integer specifying the maximum length of each block.
`nb`	Number of blocks. Default is `ceiling(m / block.size)`.
`X`	An object of class FBM.
`fun.scaling`	A function with parameters `X`, `ind.row` and `ind.col`, and that returns a data.frame with `$center` and `$scale` for the columns corresponding to `ind.col`, to scale each of their elements such as followed: $$\frac{X_{i,j} - center_j}{scale_j}.$$ Default doesn't use any scaling. You can also provide your own `center` and `scale` by using `as_scaling_fun()`.
`ind.row`	An optional vector of the row indices that are used. If not specified, all rows are used. **Don't use negative indices.**
`ind.col`	An optional vector of the column indices that are used. If not specified, all columns are used. **Don't use negative indices.**
`backingfile`	Path to the file storing the FBM data on disk. **An extension ".bk" will be automatically added.** Default stores in the temporary directory, which you can change using global option "FBM.dir".
`center`	A logical value: whether to return means or 0s.
`scale`	A logical value: whether to return standard deviations or 1s. **You can't use scale without using center.**

Value

Intervals from the input length.

The self crossproduct of an FBM.

The Pearson correlation matrix from an FBM.

A new function that returns a data.frame with two vectors, `center` and `scale`, both of length `ind.col`.

Functions

- `CutBySize()`: Copied version of the unexported `bigstatsr:::CutBySize` function.
- `big_crossprodSelf_no_warning()`: Adapted version of `bigstatsr::big_crossprodSelf` that suppresses warnings and errors related to zero scaling.
- `big_cor_no_warning()`: Adapted version of `bigstatsr::big_cor` that suppresses warnings and errors.
- `big_scale_no_warning()`: Adapted version of `bigstatsr::big_scale` that suppresses warnings and errors.

Examples

```
m = 1000
intervals = CutBySize(m, 100)
intervals
mat <- matrix(rnorm(400), 20, 20)
```

```
X = bigstatsr::FBM(20, 20, init = mat)
crossp <- big_crossprodSelf_no_warning(X)
crossp_base <- crossprod(mat)
all.equal(crossp[], crossp_base)
mat <- matrix(rnorm(400), 20, 20)
X = bigstatsr::FBM(20, 20, init = mat)
cor <- big_cor_no_warning(X)
cor_base <- cor(mat)
all.equal(cor[], cor_base)
set.seed(123)
mat <- matrix(rnorm(200), 20, 10)
X <- bigstatsr::FBM(20, 10, init = mat)
bs_ns <- big_scale_no_warning(center = TRUE, scale = TRUE)
scale_stats <- bs_ns(X)
scale_stats
bigstatsr::big_apply(X, function(X, ind) {
  X.sub <- X[, ind, drop = FALSE]
  X.sub <- t((t(X.sub) - scale_stats$center[ind]) / scale_stats$scale[ind])
  X[, ind] <- X.sub
  NULL
}, block.size = 3)
scaled_mat <- scale(mat)
all.equal(X[], scaled_mat, check.attributes = FALSE)
X[1:5, 1:5]
scaled_mat[1:5, 1:5]
```

---

anglemania *anglemania*

Description

anglemania computes critical angles between genes across all samples provided in an SingleCellExperiment object. It calculates angles, transforms them to z-scores, computes statistical measures, and selects the top genes based on mean and standard deviation of z-scores. These genes are biologically informative and invariant to batch effects.

This function adds a unique batch identifier to the metadata of a SingleCellExperiment object by combining specified dataset and batch keys. This is useful for distinguishing samples during integration or analysis.

Usage

```
anglemania(
  sce,
  batch_key,
  dataset_key = NA_character_,
  max_n_genes = 2000,
  min_cells_per_gene = 1,
  min_samples_per_gene = 2,
  allow_missing_features = FALSE,
  method = "cosine",
  permute_row_or_column = "column",
  permutation_function = "sample",
  prefilter_threshold = 0.5,
```

```
  normalization_method = "divide_by_total_counts",
  verbose = TRUE
)

check_params(
  sce,
  batch_key,
  dataset_key,
  max_n_genes,
  method,
  min_cells_per_gene,
  min_samples_per_gene,
  allow_missing_features,
  permute_row_or_column,
  permutation_function,
  prefilter_threshold,
  normalization_method,
  verbose
)

add_unique_batch_key(sce, dataset_key = NA_character_, batch_key)

get_intersect_genes(
  matrix_list,
  allow_missing_features = FALSE,
  min_samples_per_gene = 1,
  verbose = TRUE
)
```

Arguments

sce	A SingleCellExperiment object.
batch_key	A character string specifying the column name in the metadata that identifies the batch.
dataset_key	A character string specifying the column name in the metadata that identifies the dataset. If NA, only the batch_key is used.
max_n_genes	Integer specifying the maximum number of genes to select.
min_cells_per_gene	Integer specifying the minimum number of cells per gene. Default is 1.
min_samples_per_gene	Integer indicating the minimum number of samples per gene.
allow_missing_features	Logical indicating whether to allow missing features.
method	Character string specifying the method to use for calculating the relationship between gene pairs. Default is "cosine". Other options include "spearman"
permute_row_or_column	Character "row" or "column", whether permutations should be executed row-wise or column wise. Default is "column"
permutation_function	Character "sample" or "permute_nonzero". If sample,then sample is used for constructing background distributions. If permute_nonzero, then only non-zero values are permuted. Default is "sample"

prefilter_threshold
: Numeric value specifying the threshold prefiltering genes. Speeds up gene selection.

normalization_method
: Character "divide_by_total_counts" or "scale_by_total_counts". Default is "divide_by_total_coun

verbose
: Logical indicating whether to print messages.

matrix_list
: A list of bigstatsr::FBM objects.

Details

This function performs the following steps:

1. Computes angles between genes for each batch in the SingleCellExperiment using the specified method, via factorise.
2. Transforms the angles to z-scores.
3. Computes statistical measures (mean z-score, signal-to-noise ratio) across batches using get_list_stats.
4. Selects the top n genes based on mean and standard deviation of z-scores using select_genes.

The computed statistics and selected genes are added to the SingleCellExperiment object, which is returned.

Value

An updated SingleCellExperiment object with computed statistics and selected genes. The results are stored in the metadata of the SingleCellExperiment object.

A list of validated parameters

A SingleCellExperiment object with an additional metadata column containing the unique batch key.

A character vector of intersected genes.

Functions

- check_params(): Check Parameters provided to the anglemania function
- add_unique_batch_key(): Temporarily add a unique batch key to the dataset
- get_intersect_genes(): Extract the intersected genes from a list of matrices (count matrices from different batches/datasets). It also allows for missing features in individual matrices, so that a feature does not have to be present in every single batch.

See Also

get_list_stats, select_genes, factorise, big_apply, https://arxiv.org/abs/1306.0256

Examples

```
# Set seed (optional)
set.seed(1)
sce <- sce_example()
sce <- anglemania(
  sce,
  batch_key = "batch",
  method = "cosine"
)
```

```
# Access the selected genes
selected_genes <- get_anglemania_genes(sce)
selected_genes[1:10]
sce <- sce_example()
params <- check_params(
  sce,
  batch_key = "batch",
  dataset_key = "dataset",
  max_n_genes = 2000,
  method = "cosine",
  min_cells_per_gene = 1,
  min_samples_per_gene = 2,
  allow_missing_features = FALSE,
  permute_row_or_column = "column",
  permutation_function = "sample",
  prefilter_threshold = 0.5,
  normalization_method = "divide_by_total_counts",
  verbose = TRUE
)
sce <- sce_example()
head(SummarizedExperiment::colData(sce))
sce <- add_unique_batch_key(
    sce,
    batch_key = "batch",
    dataset_key = "dataset"
)
head(SummarizedExperiment::colData(sce))
library(SingleCellExperiment)
sce <- sce_example()
barcodes_by_batch <- split(colnames(sce), colData(sce)$batch)
matrix_list <- lapply(barcodes_by_batch, function(barcodes) {
    SingleCellExperiment::counts(sce)[, barcodes]
})
intersect_genes <- get_intersect_genes(matrix_list)
head(intersect_genes)
```

`anglemania_utils` *Utility Functions for the anglemania Package*

Description

A collection of utility functions used within the **anglemania** package for manipulating FBMs, calculating statistics, and selecting genes.

Replaces all NaN and Inf values in a numeric vector with NA.

Usage

```
sparse_to_fbm(s_mat)

replace_with_na(v)

normalize_matrix(
  x_mat,
```

```
  normalization_method = "divide_by_total_counts",
  verbose = TRUE
)

get_anglemania_genes(sce)

get_anglemania_stats_df(sce)
```

Arguments

s_mat	A sparse matrix.
v	A numeric vector.
x_mat	A bigstatsr::FBM object containing the matrix to normalize (typically genes x cells).
normalization_method	A character string specifying the normalization method to use. One of "divide_by_total_counts" (default) or "find_residuals". • "divide_by_total_counts" normalizes each cell by its total expression count and applies log1p. • "find_residuals" computes log1p-transformed residuals after regressing out total expression.
sce	A SingleCellExperiment or SummarizedExperiment object

Value

An FBM object from the **bigstatsr** package.

A numeric vector with NaN and Inf values replaced with NA.

The input FBM object with normalized values written back in place. This function modifies the input x_mat by reference.

A character vector of gene names that have been selected by the anglemania algorithm

A data frame of gene pairs from which the anglemania genes were selected

Functions

- sparse_to_fbm(): Convert a sparse matrix into a file-backed matrix (FBM) with efficient memory usage.
- replace_with_na(): replace Nan and Inf values with NA
- normalize_matrix(): normalize matrix Normalize a Filebacked Big Matrix (FBM) using either total-count scaling or residuals from a linear model. Intended for use with single-cell RNA-seq gene expression data.
- get_anglemania_genes(): Utility function to extract the genes that have been selected by the anglemania algorithm.
- get_anglemania_stats_df(): Utility function to extract the stats of the gene pairs from which the anglemania genes were selected.

Examples

```
s_mat <- Matrix::rsparsematrix(nrow = 10, ncol = 5, density = 0.3)
fbm_mat <- sparse_to_fbm(s_mat)
fbm_mat
v <- c(1, 2, 3, 4, 5, 6, 7, Inf, 9, NA)
v <- replace_with_na(v)
v
library(bigstatsr)
set.seed(42)
mat <- matrix(rpois(1000, lambda = 5), nrow = 100, ncol = 10)
fbm <- as_FBM(mat)

normalize_matrix(
  fbm,
  normalization_method = "divide_by_total_counts"
)[1:5, 1:5]
normalize_matrix(
  fbm,
  normalization_method = "find_residuals"
)[1:5, 1:5]

sce <- sce_example()
sce <- anglemania(sce, batch_key = "batch")
anglemania_genes <- get_anglemania_genes(sce)
head(anglemania_genes)
length(anglemania_genes)
sce <- sce_example()
sce <- anglemania(sce, batch_key = "batch")
anglemania_stats_df <- get_anglemania_stats_df(sce)
head(anglemania_stats_df)
length(anglemania_stats_df)
```

---

extract_angles *Calculate cosine angle between genes*

Description

Constructs a matrix of gene-gene relationships based on distance metrics.

Usage

```
extract_angles(x_mat, method = "cosine")
```

Arguments

x_mat	An FBM object containing raw gene expression data, where rows correspond to genes and columns to samples. The data will be normalized and scaled within the function.
method	A character string specifying the method to compute the gene-gene relationships. Options are: • "cosine" (default): Computes the cosine angle between genes. • "spearman": Computes the Spearman rank correlation coefficient by rank-transforming the data before computing the correlation.

Details

The function returns the gene-gene angle matrix as an FBM object.

Value

An FBM object containing the gene-gene correlation matrix. The matrix is square with dimensions equal to the number of genes and contains the pairwise correlations between genes. The diagonal elements are set to NA.

See Also

big_apply, big_cor, FBM, factorise

Examples

```
mat <- matrix(
 c(
     5, 3, 0, 0,
     0, 0, 0, 3,
     2, 1, 3, 4,
     0, 0, 1, 0,
     1, 2, 1, 2,
     3, 4, 3, 4
   ),
   nrow = 6, # 6 genes
   ncol = 4, # 4 cells
   byrow = TRUE
)

mat <- bigstatsr::FBM(nrow = nrow(mat), ncol = ncol(mat), init = mat)

angle_mat <- extract_angles(mat)
angle_mat[]
```

---

factorise *Factorize Angle Matrices into Z-Scores*

Description

factorise computes the angle matrix of the input gene expression matrix using the specified method, performs permutation to create a null distribution, and transforms the correlations into z-scores. This function is optimized for large datasets using the **bigstatsr** package.

Usage

```
factorise(
  x_mat,
  method = "cosine",
  seed = 1,
  permute_row_or_column = "column",
  permutation_function = "sample",
  normalization_method = "divide_by_total_counts"
)
```

Arguments

x_mat	A FBM object representing the normalized and scaled gene expression matrix.
method	A character string specifying the method for calculating the relationship between gene pairs. Default is "cosine". Other options include "spearman"
seed	An integer value for setting the seed for reproducibility during permutation. Default is 1.
permute_row_or_column	Character "row" or "column", whether permutations should be executed row-wise or column wise. Default is "column"
permutation_function	Character "sample" or "permute_nonzero". If sample, then sample is used for constructing background distributions. If permute_nonzero, then only non-zero values are permuted. Default is "sample"
normalization_method	Character "divide_by_total_counts" or "scale_by_total_counts". Default is "divide_by_total_coun

Details

The function performs the following steps:

1. **Permutation**: The input matrix is permuted column-wise to disrupt existing angles, creating a null distribution.
2. **Angle Computation**: Computes the angle matrix for both the original and permuted matrices using extract_angles.
3. **Method-Specific Processing**:
   - For other methods ("cosine", "spearman"), statistical measures are computed from the permuted data.
4. **Statistical Measures**: Calculates mean, variance, and standard deviation using get_dstat.
5. **Z-Score Transformation**: Transforms the original angle matrix into z-scores.

This process allows for the identification of invariant gene-gene relationships by comparing them to a null distribution derived from the permuted data.

Value

An FBM object containing the z-score-transformed angle matrix.

See Also

extract_angles, get_dstat, big_apply, FBM

Examples

```
mat <- matrix(
 c(
     5, 3, 0, 0,
     0, 0, 0, 3,
     2, 1, 3, 4,
     0, 0, 1, 0,
     1, 2, 1, 2,
     3, 4, 3, 4
   ),
```

```
   nrow = 6, # 6 genes
   ncol = 4, # 4 cells
   byrow = TRUE
)

mat <- bigstatsr::FBM(nrow = nrow(mat), ncol = ncol(mat), init = mat)

# Run factorise with method "cosine" and a fixed seed
result_fbm <- factorise(mat, method = "cosine", seed = 1)
result_fbm[]
```

`permute_nonzero` *permute non-zero elements of vectors*

Description

Permutes the non-zero elements of a numeric vector.

Usage

```
permute_nonzero(v)
```

Arguments

v	A numeric vector.

Value

A numeric vector with non-zero elements permuted.

Examples

```
v <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) # put in some zeroes
v = c(v, 0, 0, 0)
permute_nonzero(v)
```

`sce_example` *Generate example SingleCellExperiment object*

Description

This function generates a SingleCellExperiment object with 2 batches and 2 datasets. The object contains 300 genes and 600 cells. The counts matrix is generated using the `rpois` function with different lambda values for the two batches.

Usage

```
sce_example(seed = 42)
```

Arguments

seed	The seed for the random number generator. Because this function is only used locally, we allow to set a seed within the function.

Value

A SingleCellExperiment object

Examples

```
sce <- sce_example()
sce
```

---

select_genes *Select genes from an anglemania-processed SCE*

Description

Select genes from a SingleCellExperiment object based on mean z-score and the signal-to-noise ratio of angles between gene pairs across batches.

Usage

```
prefilter_angl(
  sce,
  zscore_mean_threshold = 1,
  zscore_sn_threshold = 1,
  verbose = TRUE
)

select_genes(
  sce,
  max_n_genes = 2000,
  score_weights = c(0.4, 0.6),
  verbose = TRUE
)

extract_rows_for_unique_genes(dt, max_n_genes)
```

Arguments

sce	A SingleCellExperiment object.
zscore_mean_threshold	Numeric value specifying the threshold for the absolute mean z-score. Default is 1.
zscore_sn_threshold	Numeric value specifying the threshold for the SNR z-score. Default is 1.
verbose	Logical value indicating whether to print progress messages. Default is TRUE.
max_n_genes	An integer specifying the maximum number of unique genes to return.

score_weights	A vector of two numeric values specifying the weights for the mean z-score and standard deviation of z-score, respectively. Default is c(0.4, 0.6) for a greater emphasis on the standard deviation of z-score.
dt	A data frame containing gene pairs, with columns geneA and geneB.

Details

The function performs the following steps:

1. Identifies gene pairs where both the mean z-score and SNR z-score exceed the specified thresholds.

Selects the top n genes based on the weighted sum of the ranked mean and standard deviation of the z-score of the correlations between gene pairs.

The function combines the geneA and geneB columns, extracts unique gene names, and returns the first max_n_genes genes. If max_n_genes exceeds the number of unique genes available, all unique genes are returned.

Value

A data frame containing the prefiltered gene pairs.

The input SingleCellExperiment object with the anglemania_genes slot updated to include the selected genes and their statistical information.

A vector of unique gene identifiers.

Functions

- prefilter_angl(): Prefilter gene pairs from the mean and SNR z-scores based on thresholds, to simplify downstream filtering.
- select_genes(): Select the top n genes on the weighted sum of the ranks of the mean z-score and SNR z-score of the gene pairs.
- extract_rows_for_unique_genes(): Extract unique gene identifiers from gene pairs, returning up to a specified maximum number.

See Also

extract_rows_for_unique_genes, get_intersect_genes, get_list_stats

select_genes

Examples

```
library(SingleCellExperiment)
sce <- sce_example()
sce <- anglemania(sce, batch_key = "batch")
prefiltered_df <- prefilter_angl(
  sce,
  zscore_mean_threshold = 1,
  zscore_sn_threshold = 1
)
head(prefiltered_df)
sce <- sce_example()
sce <- anglemania(
    sce,
```

```
    batch_key = "batch",
    max_n_genes = 20
)
anglemania_genes <- get_anglemania_genes(sce)
# View the selected genes and use for integration
head(anglemania_genes)
length(anglemania_genes)
sce <- select_genes(
    sce,
    max_n_genes = 10
)
anglemania_genes <- get_anglemania_genes(sce)
head(anglemania_genes)
length(anglemania_genes)
gene_pairs <- data.frame(
  geneA = c("Gene1", "Gene2", "Gene3", "Gene4"),
  geneB = c("Gene3", "Gene4", "Gene5", "Gene6")
)
unique_genes <- extract_rows_for_unique_genes(
  gene_pairs,
  max_n_genes = 3
)
print(unique_genes)
```

`statistics` *Statistics functions for the anglemania package*

Description

A collection of utility functions used within the **anglemania** package for calculating statistics based on the results created during the anglemania function.

Usage

```
get_dstat(corr_matrix)

big_mat_list_mean(matrix_list, weights, verbose = TRUE)

get_list_stats(matrix_list, weights, verbose = TRUE)
```

Arguments

`corr_matrix`	An `FBM` object.
`matrix_list`	A list of `bigstatsr::FBM` objects.
`weights`	A numeric vector of weights for each dataset or batch.

Value

A list with statistical measures including `mean`, `sd`, `var`, `min`, and `max`.

A new `bigstatsr::FBM` object containing the mean values.

A list containing three matrices: `mean_zscore`, `sds_zscore`, and `sn_zscore` which are later used to filter gene pairs based on the absolute mean z-score and signal-to-noise ratio of the angles.

Functions

- `get_dstat()`: Compute mean, standard deviation, variance, min, and max of a correlation matrix stored as an FBM.
- `big_mat_list_mean()`: Calculates the element-wise mean from a list of `bigstatsr::FBM` objects.
- `get_list_stats()`: Calculate mean, standard deviation, and SNR across a list of FBMs.

See Also

big_apply, FBM

big_apply, FBM

Examples

```
s_mat <- Matrix::rsparsematrix(nrow = 10, ncol = 5, density = 0.3)
fbm_mat <- sparse_to_fbm(s_mat)
result <- get_dstat(fbm_mat)
str(result)
result
# Create FBMs
mat1 <- matrix(1:9, nrow = 3)
mat2 <- matrix(1:9, nrow = 3)

fbm1 <- bigstatsr::FBM(nrow = nrow(mat1), ncol = ncol(mat1), init = mat1)
fbm2 <- bigstatsr::FBM(nrow = nrow(mat2), ncol = ncol(mat2), init = mat2)

# Create weights
weights <- c(batch1 = 0.5, batch2 = 0.5)

# Create the list of FBMs
fbm_list <- list(batch1 = fbm1, batch2 = fbm2)

big_mat_list_mean(fbm_list, weights)
library(SingleCellExperiment)
library(S4Vectors)
sce <- sce_example()
sce <- anglemania(sce, batch_key = "batch")
matrix_list <- metadata(sce)$anglemania$matrix_list
weights <- setNames(
  S4Vectors::metadata(sce)$anglemania$params$dataset_weights$weight,
  S4Vectors::metadata(sce)$anglemania$params$dataset_weights$anglemania_batch
)
list_stats <- get_list_stats(matrix_list, weights)
names(list_stats)
list_stats$mean_zscore[1:5, 1:5]
```

Index

∗ **internal**
    adapted_reexports, 2
    anglemania_utils, 7
    select_genes, 13
    statistics, 15

adapted_reexports, 2
add_unique_batch_key *(*anglemania*)*, 4
anglemania, 4
anglemania_utils, 7
as_scaling_fun(), *3*

big_apply, *6*, *10, 11*, *16*
big_cor, *10*
big_cor_no_warning *(*adapted_reexports*)*, 2
big_crossprodSelf_no_warning *(*adapted_reexports*)*, 2
big_mat_list_mean *(*statistics*)*, 15
big_scale_no_warning *(*adapted_reexports*)*, 2

check_params *(*anglemania*)*, 4
CutBySize *(*adapted_reexports*)*, 2

extract_angles, 9, *11*
extract_rows_for_unique_genes, *14*
extract_rows_for_unique_genes *(*select_genes*)*, 13

factorise, *6*, *10*, 10
FBM, *3*, *8–11*, *15, 16*

get_anglemania_genes *(*anglemania_utils*)*, 7
get_anglemania_stats_df *(*anglemania_utils*)*, 7
get_dstat, *11*
get_dstat *(*statistics*)*, 15
get_intersect_genes, *14*
get_intersect_genes *(*anglemania*)*, 4
get_list_stats, *6*, *14*
get_list_stats *(*statistics*)*, 15

normalize_matrix *(*anglemania_utils*)*, 7

permute_nonzero, 12
prefilter_angl *(*select_genes*)*, 13

replace_with_na *(*anglemania_utils*)*, 7

sce_example, 12
select_genes, *6*, 13, *14*
SingleCellExperiment, *4*
sparse_to_fbm *(*anglemania_utils*)*, 7
statistics, 15