

# Package ‘nipalsMCIA’

August 8, 2025

**Title** Multiple Co-Inertia Analysis via the NIPALS Method

**Version** 1.7.0

**Description** Computes Multiple Co-Inertia Analysis (MCIA), a dimensionality reduction (jDR) algorithm, for a multi-block dataset using a modification to the Nonlinear Iterative Partial Least Squares method (NIPALS) proposed in (Hanafi et. al, 2010). Allows multiple options for row- and table-level preprocessing, and speeds up computation of variance explained. Vignettes detail application to bulk- and single cell- multi-omics studies.

**License** GPL-3

**URL** <https://github.com/Muunraker/nipalsMCIA>

**BugReports** <https://github.com/Muunraker/nipalsMCIA/issues>

**Depends** R (>= 4.3.0)

**Imports** ComplexHeatmap, dplyr, fgsea, ggplot2 (>= 3.0.0), graphics, grid, methods, MultiAssayExperiment, SummarizedExperiment, pracma, rlang, RSpectra, scales, stats

**Suggests** BiocFileCache, BiocStyle, circlize, ggpubr, KernSmooth, knitr, piggyback, reshape2, rmarkdown, rpart, Seurat (>= 4.0.0), spatstat.explore, stringr, survival, tidyverse, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**biocViews** Software, Clustering, Classification, MultipleComparison, Normalization, Preprocessing, SingleCell

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/nipalsMCIA>

**git\_branch** devel

**git\_last\_commit** 55354e7

**git\_last\_commit\_date** 2025-04-15

**Repository** Bioconductor 3.22

**Date/Publication** 2025-08-07

**Author** Maximilian Mattessich [cre] (ORCID: <https://orcid.org/0000-0002-1233-1240>),  
 Joaquin Reyna [aut] (ORCID: <https://orcid.org/0000-0002-8468-2840>),  
 Edel Aron [aut] (ORCID: <https://orcid.org/0000-0002-8683-4772>),  
 Ferhat Ay [aut] (ORCID: <https://orcid.org/0000-0002-0708-6914>),  
 Steven Kleinstein [aut] (ORCID: <https://orcid.org/0000-0003-4957-1544>),  
 Anna Konstorum [aut] (ORCID: <https://orcid.org/0000-0003-4024-2058>)

**Maintainer** Maximilian Mattessich <maximilian.mattessich@northwestern.edu>

## Contents

|                                          |    |
|------------------------------------------|----|
| nipalsMCIA-package . . . . .             | 3  |
| block_preproc . . . . .                  | 3  |
| block_weights_heatmap . . . . .          | 4  |
| cc_preproc . . . . .                     | 5  |
| col_preproc . . . . .                    | 5  |
| data_blocks . . . . .                    | 6  |
| deflate_block_bl . . . . .               | 7  |
| deflate_block_gs . . . . .               | 7  |
| extract_from_mae . . . . .               | 8  |
| get_colors . . . . .                     | 9  |
| get_metadata_colors . . . . .            | 9  |
| get_tv . . . . .                         | 10 |
| global_scores_eigenvalues_plot . . . . . | 11 |
| global_scores_heatmap . . . . .          | 11 |
| gsea_report . . . . .                    | 12 |
| metadata_NCI60 . . . . .                 | 13 |
| NipalsResult-class . . . . .             | 13 |
| nipals_iter . . . . .                    | 14 |
| nipals_multiblock . . . . .              | 14 |
| nmb_get_bl . . . . .                     | 16 |
| nmb_get_bs . . . . .                     | 17 |
| nmb_get_bs_weights . . . . .             | 18 |
| nmb_get_eigs . . . . .                   | 18 |
| nmb_get_gl . . . . .                     | 19 |
| nmb_get_gs . . . . .                     | 19 |
| nmb_get_metadata . . . . .               | 20 |
| ord_loadings . . . . .                   | 20 |
| predict_gs . . . . .                     | 21 |
| projection_plot . . . . .                | 22 |
| simple_mae . . . . .                     | 23 |
| vis_load_ord . . . . .                   | 24 |
| vis_load_plot . . . . .                  | 25 |

## Index

26

nipalsMCIA-package

*nipalsMCIA: Multiple Co-Inertia Analysis via the NIPALS Method***Description**

Computes Multiple Co-Inertia Analysis (MCIA), a dimensionality reduction (jDR) algorithm, for a multi-block dataset using a modification to the Nonlinear Iterative Partial Least Squares method (NIPALS) proposed in (Hanafi et. al, 2010). Allows multiple options for row- and table-level preprocessing, and speeds up computation of variance explained. Vignettes detail application to bulk- and single cell- multi-omics studies.

**Author(s)**

**Maintainer:** Maximilian Mattessich <maximilian.mattessich@northwestern.edu> ([ORCID](#))

Authors:

- Joaquin Reyna <joreyna@live.com> ([ORCID](#))
- Edel Aron <edel.aron@yale.edu> ([ORCID](#))
- Ferhat Ay <ferhatay@lji.org> ([ORCID](#))
- Steven Kleinstein <steven.kleinstein@yale.edu> ([ORCID](#))
- Anna Konstorum <konstorum.anna@gmail.com> ([ORCID](#))

**See Also**

Useful links:

- <https://github.com/Muunraker/nipalsMCIA>
- Report bugs at <https://github.com/Muunraker/nipalsMCIA/issues>

block\_preproc

*Block-level preprocessing***Description**

A function that normalizes an input dataset (data block) according to a variety of options. Intended to be used after column/row-level normalization.

**Usage**

```
block_preproc(df, block_preproc_method)
```

**Arguments**

df                      dataset to preprocess (must be in data matrix form)

block\_preproc\_method

method which is used to normalize blocks, with options:

- ‘unit\_var’ FOR CENTERED MATRICES ONLY - divides each block by the square root of its variance
- ‘num\_cols’ divides each block by the number of variables in the block.
- ‘largest\_sv’ divides each block by its largest singular value.
- ‘none’ performs no preprocessing

**Value**

the preprocessed dataset

**Examples**

```
df <- matrix(rbinom(15, 1, prob = 0.3), ncol = 3)
preprocessed_dataframe <- block_preproc(df, "unit_var")
```

---

block\_weights\_heatmap *block\_weights\_heatmap*

---

**Description**

Function to plot heatmap of block score weights

**Usage**

```
block_weights_heatmap(mcia_results)
```

**Arguments**

`mcia_results` MCIA results object returned from ‘nipals\_multiblock’

**Details**

Plotting function for heatmap of block score weights

**Value**

heatmap object containing the block weights as a heatmap

**Examples**

```
data(NCI60)
data_blocks_mae <- simple_mae(data_blocks, row_format="sample",
                              colData=metadata_NCI60)
mcia_results <- nipals_multiblock(data_blocks_mae, num_PCs = 10,
                                 plots = "none", tol = 1e-12)
block_weights_heatmap(mcia_results)
```

---

`cc_preproc`*Centered Column Profile Pre-processing*

---

**Description**

Converts data blocks into centered column profiles where each block has unit variance. Mimics the pre-processing in the Omicade4 package (Meng et al. 2014)

**Usage**

```
cc_preproc(df)
```

**Arguments**

`df` the data frame to apply pre-processing to, in "sample" x "variable" format

**Details**

Performs the following steps on a given data frame:

- Offsets data to make whole matrix non-negative
- Divides each column by its sum
- Subtracts (row sum/total sum) from each row
- Multiplies each column by  $\sqrt{\text{column sum/total sum}}$
- Divides the whole data frame by its total variance (the sqrt of the sum of singular values)

**Value**

the processed data frame

**Examples**

```
df <- matrix(rbinom(15, 1, prob = 0.3), ncol = 3)
preprocessed_dataframe <- cc_preproc(df)
```

---

`col_preproc`*Centered Column Profile Pre-processing*

---

**Description**

Converts data blocks into centered column profiles where each block has unit variance. Mimics the pre-processing in the Omicade4 package (Meng et al. 2014)

**Usage**

```
col_preproc(df, col_preproc_method)
```

## Arguments

- `df` the data frame to apply pre-processing to, in "sample" x "variable" format
- `col_preproc_method` denotes the type of column-centered preprocessing. Options are:
- ‘colprofile’ Performs the following steps on a given data frame:
    - Offsets data to make whole matrix non-negative
    - Divides each column by its sum
    - Subtracts (row sum/total sum) from each row
    - Multiplies each column by  $\sqrt{\text{column sum/total sum}}$
  - ‘standardized’ centers each column and divides by its standard deviation.
  - ‘centered\_only’ ONLY centers data

## Details

Performs preprocessing on a sample/variable (row/column) level according to the parameter given.

## Value

the processed data frame

## Examples

```
df <- matrix(rbinom(15, 1, prob = 0.3), ncol = 3)
preprocessed_dataframe <- col_preproc(df, col_preproc_method = 'colprofile')
```

---

|             |                                |
|-------------|--------------------------------|
| data_blocks | <i>NCI-60 Multi-Omics Data</i> |
|-------------|--------------------------------|

---

## Description

A dataset of measurements of 12,895 mRNA, 537 miRNA, and 7,016 protein variables (columns) on 21 cancer cell lines (rows) from the NCI-60 cancer cell line database.

## Value

Large list with 3 elements (one for each omic)

## Source

Meng et. al, 2016 supplementary materials <https://doi.org/10.1093/bib/bbv108>

## References

<https://github.com/aedin/NCI60Example>

---

|                  |                                     |
|------------------|-------------------------------------|
| deflate_block_bl | <i>Deflation via block loadings</i> |
|------------------|-------------------------------------|

---

**Description**

Removes data from a data frame in the direction of a given block loadings vector.

**Usage**

```
deflate_block_bl(df, bl)
```

**Arguments**

|    |                                              |
|----|----------------------------------------------|
| df | a data frame in "sample" x "variable" format |
| bl | a block loadings vector in variable space    |

**Details**

Subtracts the component of each row in the direction of a given block loadings vector to yield a 'deflated' data matrix.

**Value**

the deflated data frame

**Examples**

```
df <- matrix(rbinom(15, 1, prob = 0.3), ncol = 3)
block_loading <- rbinom(3, 1, prob = 0.3)
deflated_data <- deflate_block_bl(df, block_loading)
```

---

|                  |                                    |
|------------------|------------------------------------|
| deflate_block_gs | <i>Deflation via global scores</i> |
|------------------|------------------------------------|

---

**Description**

Removes data from a data frame in the direction of a given global scores vector.

**Usage**

```
deflate_block_gs(df, gs)
```

**Arguments**

|    |                                              |
|----|----------------------------------------------|
| df | a data frame in "sample" x "variable" format |
| gs | a global scores vector in sample space       |

**Details**

Subtracts the component of each column in the direction of a given global scores vector to yield a ‘deflated’ data matrix.

**Value**

the deflated data frame

**Examples**

```
df <- matrix(rbinom(15, 1, prob = 0.3), ncol = 3)
global_score <- rbinom(5, 1, prob = 0.3)
deflated_data <- deflate_block_gs(df, global_score)
```

---

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| extract_from_mae | <i>Extract a list of harmonized data matrices from an MAE object</i> |
|------------------|----------------------------------------------------------------------|

---

**Description**

Extract a list of harmonized data matrices for input into `nipals_multiblock()` from an MAE object

**Usage**

```
extract_from_mae(MAE_object, subset_data = "all", harmonize = TRUE)
```

**Arguments**

|             |                                                                                                                                                                                                                                                                                                         |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MAE_object  | an MAE object containing experiment data for extraction colData field optional experiments should either be SummarizedExperiment, SingleCellExperiment, or RangedSummarizedExperiment classes                                                                                                           |
| subset_data | <ul style="list-style-type: none"> <li>• ‘all’ use all experiments in MAE object</li> <li>• ‘c(omic1,omic2,...)’ list of omics from names(MAE_object)</li> </ul>                                                                                                                                        |
| harmonize   | A boolean whether samples should be checked for duplicates <ul style="list-style-type: none"> <li>• ‘TRUE’ (default) merges duplicate samples via the ‘MultiAssayExperiment::mergeReplicates’ function</li> <li>• ‘FALSE’ skips sample duplicate check - USE THIS FOR LARGE-SAMPLE DATASETS.</li> </ul> |

**Value**

List of harmonized data matrices for input into ‘`nipals_multiblock()`’

**Examples**

```
data(NCI60)
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",
                              colData=metadata_NCI60)
NCI60_input = extract_from_mae(data_blocks_mae,subset='all')
```

---

|            |                                            |
|------------|--------------------------------------------|
| get_colors | <i>Assigning colors to different omics</i> |
|------------|--------------------------------------------|

---

**Description**

Creates a list of omics and associated colors for plotting. The default palette was chosen to be color-blindness friendly.

**Usage**

```
get_colors(
  mcia_results,
  color_pal = scales::viridis_pal,
  color_pal_params = list()
)
```

**Arguments**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| mcia_results     | object returned from nipals_multiblock() function     |
| color_pal        | a function which returns color palettes (e.g. scales) |
| color_pal_params | list of parameters for the corresponding function     |

**Value**

List of omics with assigned colors

**Examples**

```
data(NCI60)
data_blocks_mae <- simple_mae(data_blocks, row_format="sample",
                              colData=metadata_NCI60)
mcia_results <- nipals_multiblock(data_blocks_mae, num_PCs = 10,
                                  plots = "none", tol = 1e-12)
colors_omics <- get_colors(mcia_results)
```

---

|                     |                                                                  |
|---------------------|------------------------------------------------------------------|
| get_metadata_colors | <i>Assigning colors to different values of a metadata column</i> |
|---------------------|------------------------------------------------------------------|

---

**Description**

Creates a list of metadata columns and associated colors for plotting. The default palette was chosen to be color-blindness friendly.

**Usage**

```
get_metadata_colors(
  mcia_results,
  color_col,
  color_pal = scales::viridis_pal,
  color_pal_params = list()
)
```

**Arguments**

`mcia_results`     object returned from `nipals_multiblock()` function  
`color_col`         an integer or string specifying the column that will be used for `color_col`  
`color_pal`         a function which returns color palettes (e.g. `scales`)  
`color_pal_params`     list of parameters for the corresponding function

**Value**

List of metadata columns with assigned colors

**Examples**

```

data(NCI60)
data_blocks_mae <- simple_mae(data_blocks, row_format="sample",
                              colData=metadata_NCI60)
mcia_results <- nipals_multiblock(data_blocks_mae, num_PCs = 10,
                                 plots = "none", tol = 1e-12)
colors_omics <- get_metadata_colors(mcia_results, "cancerType",
                                   color_pal_params = list(option = "E"))

```

---

`get_tv`

*Computes the total variance of a multi-omics dataset*

---

**Description**

Computes the total variances of all data blocks in a multi-omics dataset, intended for datasets that do not use ‘CCpreproc’

**Usage**

```
get_tv(ds)
```

**Arguments**

`ds`                     a list of multi-omics dataframes/matrices in "sample x variable" format

**Value**

the total variance of the dataset (i.e. sum of block variances)

**Examples**

```

data(NCI60)
tot_var <- get_tv(data_blocks)

```

---

```
global_scores_eigenvalues_plot
      global_scores_eigenvalues_plot
```

---

**Description**

Function to plot eigenvalues of scores up to num\_PCs

**Usage**

```
global_scores_eigenvalues_plot(mcia_results)
```

**Arguments**

mcia\_results      MCIA results object returned from ‘nipals\_multiblock’

**Details**

Plotting function for eigenvalues of scores up to num\_PCs

**Value**

Displays the contribution plot using eigenvalues

**Examples**

```
data(NCI60)
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",
                              colData=metadata_NCI60)
mcia_results <- nipals_multiblock(data_blocks_mae, num_PCs = 10,
                                 plots = "none", tol=1e-12)
global_scores_eigenvalues_plot(mcia_results)
```

---

```
global_scores_heatmap    Plotting a heatmap of global factors scores (sample v. factors)
```

---

**Description**

Plots a heatmap of MCIA global scores

**Usage**

```
global_scores_heatmap(
  mcia_results,
  color_col = NULL,
  color_pal = scales::viridis_pal,
  color_pal_params = list(option = "D")
)
```

**Arguments**

|                  |                                                                                                               |
|------------------|---------------------------------------------------------------------------------------------------------------|
| mcia_results     | the mcia object matrix after running MCIA, must also contain metadata with columns corresponding to color_col |
| color_col        | an integer or string specifying the column that will be used for color_col                                    |
| color_pal        | a list of colors or function which returns a list of colors                                                   |
| color_pal_params | a list of parameters for the color function                                                                   |

**Value**

ComplexHeatmap object

---

|             |                                                       |
|-------------|-------------------------------------------------------|
| gsea_report | <i>Perform biological annotation-based comparison</i> |
|-------------|-------------------------------------------------------|

---

**Description**

Runs fgsea for the input gene vector

**Usage**

```
gsea_report(
  metagenes,
  path.database,
  factors = NULL,
  pval.thr = 0.05,
  nproc = 4
)
```

**Arguments**

|               |                                                            |
|---------------|------------------------------------------------------------|
| metagenes     | Vector of gene scores where the row names are HUGO symbols |
| path.database | path to a GMT annotation file                              |
| factors       | vector of factors which should be analyzed                 |
| pval.thr      | p-value threshold (default to 0.05)                        |
| nproc         | number of processors to utilize                            |

**Value**

data frame with the most significant p-value number of significant pathways  
the selectivity scores across the given factors

---

|                |                                    |
|----------------|------------------------------------|
| metadata_NCI60 | <i>NCI-60 Multi-Omics Metadata</i> |
|----------------|------------------------------------|

---

**Description**

Metadata for the included multi-omics dataset, denoting the cancer type associated with each of the 21 cell lines.

**Value**

List with 21 elements

**Source**

Meng et. al, 2016 supplementary materials <https://doi.org/10.1093/bib/bbv108>

**References**

<https://github.com/aedin/NCI60Example>

---

|                    |                                                                           |
|--------------------|---------------------------------------------------------------------------|
| NipalsResult-class | <i>An S4 class to contain results computed with 'nipals_multiblock()'</i> |
|--------------------|---------------------------------------------------------------------------|

---

**Description**

An S4 class to contain results computed with 'nipals\_multiblock()'

**Value**

A NipalsResult object.

**Slots**

global\_scores A matrix containing global scores as columns.

global\_loadings A matrix containing global loadings as columns.

block\_score\_weights A matrix containing block weights as columns.

block\_scores A list of matrices. Each matrix contains the scores as columns for a given block.

block\_loadings A list of matrices. Each matrix contains the loadings as columns for a given block.

eigvals A list of singular values of the data matrix at each deflation step.

col\_preproc\_method character for the column-level preprocessing method used. See 'col\_preproc()'.

block\_preproc\_method character for the block-level preprocessing method used. See 'block\_preproc()'.

block\_variances A list of variances for each block.

metadata A data frame of metadata originally passed into 'nipals\_multiblock()'.

---

|             |                         |
|-------------|-------------------------|
| nipals_iter | <i>NIPALS Iteration</i> |
|-------------|-------------------------|

---

### Description

Applies one iteration stage/loop of the NIPALS algorithm.

### Usage

```
nipals_iter(ds, tol = 1e-12, maxIter = 1000)
```

### Arguments

|         |                                                                 |
|---------|-----------------------------------------------------------------|
| ds      | a list of data matrices, each in "sample" x "variable" format   |
| tol     | a number for the tolerance on the stopping criterion for NIPALS |
| maxIter | a number for the maximum number of times NIPALS should iterate  |

### Details

Follows the NIPALS algorithm as described by Hanafi et. al. (2010). Starts with a random vector in sample space and repeatedly projects it onto the variable vectors and block scores to generate block and global loadings/scores/weights. The loop stops when either the stopping criterion is low enough, or the maximum number of iterations is reached. Intended as a utility function for 'nipals\_multiblock' to be used between deflation steps.

### Value

a list containing the global/block scores, loadings and weights for a given order

### Examples

```
data(NCI60)
data_blocks <- lapply(data_blocks, as.matrix)
nipals_results <- nipals_iter(data_blocks, tol = 1e-7, maxIter = 1000)
```

---

|                   |                                     |
|-------------------|-------------------------------------|
| nipals_multiblock | <i>Main NIPALS computation loop</i> |
|-------------------|-------------------------------------|

---

### Description

Applies the full adjusted NIPALS algorithm to generate block and global scores/loadings with the desired deflation method.

**Usage**

```
nipals_multiblock(
  data_blocks_mae,
  col_preproc_method = "colprofile",
  block_preproc_method = "unit_var",
  num_PCs = 10,
  tol = 1e-09,
  max_iter = 1000,
  color_col = NULL,
  deflationMethod = "block",
  plots = "all",
  harmonize = TRUE
)
```

**Arguments**

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data_blocks_mae      | a MultiAssayExperiment class object (with sample metadata as a dataframe in the colData attribute).                                                                                                                                                                                                                                                                                                                                 |
| col_preproc_method   | <p>an option for the desired column-level data pre-processing, either:</p> <ul style="list-style-type: none"> <li>• ‘colprofile’ applies column-centering, row and column weighting by contribution to variance.</li> <li>• ‘standardized’ centers each column and divides by its standard deviation.</li> <li>• ‘centered_only’ ONLY centers data</li> </ul>                                                                       |
| block_preproc_method | <p>an option for the desired block-level data pre-processing, either:</p> <ul style="list-style-type: none"> <li>• ‘unit_var’ FOR CENTERED MATRICES ONLY - divides each block by the square root of its variance</li> <li>• ‘num_cols’ divides each block by the number of variables in the block.</li> <li>• ‘largest_sv’ divides each block by its largest singular value.</li> <li>• ‘none’ performs no preprocessing</li> </ul> |
| num_PCs              | the maximum order of scores/loadings                                                                                                                                                                                                                                                                                                                                                                                                |
| tol                  | a number for the tolerance on the stopping criterion for NIPALS                                                                                                                                                                                                                                                                                                                                                                     |
| max_iter             | a number for the maximum number of times NIPALS should iterate                                                                                                                                                                                                                                                                                                                                                                      |
| color_col            | Optional argument with the column name of the ‘metadata’ data frame used to define plotting colors                                                                                                                                                                                                                                                                                                                                  |
| deflationMethod      | <p>an option for the desired deflation method, either:</p> <ul style="list-style-type: none"> <li>• ‘block’ deflation via block loadings (for MCIA, default)</li> <li>• ‘global’ deflation via global scores (for CPCA)</li> </ul>                                                                                                                                                                                                  |
| plots                | <p>an option to display various plots of results:</p> <ul style="list-style-type: none"> <li>• ‘all’ displays plots of block scores, global scores, and eigenvalue scree plot</li> <li>• ‘global’ displays only global score projections and eigenvalue scree plot</li> <li>• ‘none’ does not display plots</li> </ul>                                                                                                              |
| harmonize            | boolean whether or not samples should be checked for duplicates and re-ordered so that each row corresponds to the same sample across datasets. Set to FALSE to greatly reduce computation time on many samples (default = TRUE).                                                                                                                                                                                                   |

## Details

Follows the NIPALS algorithm as described by Hanafi et. al. (2010). For each order of scores/loadings, the vectors are computed via the 'nipals\_iter' function, then used to deflate the data matrix according to the desired deflation method. This process is repeated up to the desired maximum order of scores/loadings.

## Value

a 'nipalsResult' object with the following fields:

- 'global\_scores' a matrix containing global scores as columns (NOT normalized to unit variance)
- 'global\_loadings' a matrix containing global loadings as columns
- 'global\_score\_weights' a matrix of weights to express global scores as a combination of block scores. Has dimensions "num\_Blocks" by "num\_PCs"
- 'eigvals' a matrix containing the eigenvalue for each computed global score.
- 'block scores' a list of matrices, each contains the scores for one block
- 'block loadings' a list of matrices, each contains the loadings for one block (w/ unit length)
- 'block score weights' a matrix containing weights for each block score of each order used to construct the global scores.
- 'col\_preproc\_method' the column preprocessing method used on the data.
- 'block\_variances' a list of variances of each block AFTER NORMALIZATION OPTION APPLIED
- 'metadata' the metadata dataframe supplied with the 'metadata' argument. Note: overrides metadata present in any MAE class object.

## Examples

```
data(NCI60)
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",
                              colData=metadata_NCI60)
NIPALS_results <- nipals_multiblock(data_blocks_mae, num_PCs = 10,
                                    tol = 1e-12, max_iter = 1000,
                                    col_preproc_method = "colprofile",
                                    deflationMethod = "block")
MCIA_results <- nipals_multiblock(data_blocks_mae, num_PCs = 4)
CPCA_results <- nipals_multiblock(data_blocks_mae, num_PCs = 4,
                                  deflationMethod = 'global')
```

---

nmb\_get\_bl

---

*Accessor function for block loadings*


---

## Description

Retrieves the block loadings as a list of matrices from a 'NipalsResult' object, typically output from 'nipals\_multiblock()'.

**Usage**

```
nmb_get_bl(nmb_object)
```

**Arguments**

nmb\_object      A ‘NipalsResult’ object.

**Value**

a list of matrices containing block loadings.

**Examples**

```
data("NCI60")
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",
                             colData=metadata_NCI60)
mcia_out <- nipals_multiblock(data_blocks_mae, num_PCs = 10)
block_loadings<- nmb_get_bl(mcia_out)
```

---

nmb\_get\_bs

*Accessor function for block scores*


---

**Description**

Retrieves the block scores as a list of matrices from a ‘NipalsResult’ object, typically output from ‘nipals\_multiblock()’.

**Usage**

```
nmb_get_bs(nmb_object)
```

**Arguments**

nmb\_object      A ‘NipalsResult’ object.

**Value**

a list of matrices containing block scores.

**Examples**

```
data("NCI60")
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",
                             colData=metadata_NCI60)
mcia_out <- nipals_multiblock(data_blocks_mae, num_PCs = 10)
block_scores <- nmb_get_bs(mcia_out)
```

---

|                    |                                                  |
|--------------------|--------------------------------------------------|
| nmb_get_bs_weights | <i>Accessor function for block score weights</i> |
|--------------------|--------------------------------------------------|

---

**Description**

Retrieves the block score weights from a ‘NipalsResult’ object, typically output from ‘nipals\_multiblock()’.

**Usage**

```
nmb_get_bs_weights(nmb_object)
```

**Arguments**

nmb\_object      A ‘NipalsResult’ object.

**Value**

a matrix containing the block score weights.

**Examples**

```
data("NCI60")
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",
                              colData=metadata_NCI60)
mcia_out <- nipals_multiblock(data_blocks_mae, num_PCs = 10)
block_score_weights <- nmb_get_bs_weights(mcia_out)
```

---

|              |                                          |
|--------------|------------------------------------------|
| nmb_get_eigs | <i>Accessor function for eigenvalues</i> |
|--------------|------------------------------------------|

---

**Description**

Retrieves the eigenvalues from a ‘NipalsResult’ object, typically output from ‘nipals\_multiblock()’.

**Usage**

```
nmb_get_eigs(nmb_object)
```

**Arguments**

nmb\_object      A ‘NipalsResult’ object.

**Value**

a matrix containing the eigenvalues for all global scores.

**Examples**

```
data("NCI60")
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",
                              colData=metadata_NCI60)
mcia_out <- nipals_multiblock(data_blocks_mae, num_PCs = 10)
nipals_eigvals <- nmb_get_eigs(mcia_out)
```

---

`nmb_get_gl`*Accessor function for global loadings*

---

**Description**

Retrieves the global loadings as a matrix from a ‘NipalsResult’ object, typically output from ‘nipals\_multiblock()’.

**Usage**

```
nmb_get_gl(nmb_object)
```

**Arguments**

`nmb_object`      A ‘NipalsResult’ object.

**Value**

a matrix containing global loadings.

**Examples**

```
data("NCI60")
data_blocks_mae <- simple_mae(data_blocks, row_format="sample",
                              colData=metadata_NCI60)
mcia_out <- nipals_multiblock(data_blocks_mae, num_PCs = 10)
global_loadings <- nmb_get_gl(mcia_out)
```

---

`nmb_get_gs`*Accessor function for global scores*

---

**Description**

Retrieves the global scores as a matrix from a ‘NipalsResult’ object, typically output from ‘nipals\_multiblock()’.

**Usage**

```
nmb_get_gs(nmb_object)
```

**Arguments**

`nmb_object`      A ‘NipalsResult’ object.

**Value**

a matrix containing global scores.

**Examples**

```
data("NCI60")
data_blocks_mae <- simple_mae(data_blocks, row_format="sample",
                              colData=metadata_NCI60)
mcia_out <- nipals_multiblock(data_blocks_mae, num_PCs = 10)
global_scores <- nmb_get_gs(mcia_out)
```

---

|                  |                                       |
|------------------|---------------------------------------|
| nmb_get_metadata | <i>Accessor function for metadata</i> |
|------------------|---------------------------------------|

---

**Description**

Retrieves the metadata from a ‘NipalsResult’ object, typically output from ‘nipals\_multiblock()’.

**Usage**

```
nmb_get_metadata(nmb_object)
```

**Arguments**

nmb\_object      A ‘NipalsResult’ object.

**Value**

a dataframe containing metadata associated with the ‘NipalsResult’ object.

**Examples**

```
data("NCI60")
data_blocks_mae <- simple_mae(data_blocks, row_format="sample",
                              colData=metadata_NCI60)
mcia_out <- nipals_multiblock(data_blocks_mae, num_PCs = 10)
nipals_metadata <- nmb_get_metadata(mcia_out)
```

---

|              |                                         |
|--------------|-----------------------------------------|
| ord_loadings | <i>Ranked global loadings dataframe</i> |
|--------------|-----------------------------------------|

---

**Description**

Creates a dataframe with ranked loadings for a given factor

**Usage**

```
ord_loadings(
  mcia_results,
  omic = "all",
  factor = 1,
  absolute = FALSE,
  descending = TRUE
)
```

**Arguments**

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| mcia_results | object returned from nipals_multiblock() function                                       |
| omic         | choose an omic to rank, or choose 'all' for all, ((omic = "all", omic = "miRNA", etc.)) |
| factor       | choose a factor (numeric value from 1 to number of factors in mcia_results)             |
| absolute     | whether to rank by absolute value                                                       |
| descending   | whether to rank in descending or ascending order                                        |

**Value**

ranked dataframe

**Examples**

```
data(NCI60)
data_blocks_mae <- simple_mae(data_blocks, row_format="sample",
                              colData=metadata_NCI60)
mcia_results <- nipals_multiblock(data_blocks_mae, num_PCs = 10,
                                plots = "none", tol = 1e-12)
all_pos_1 <- ord_loadings(mcia_results = mcia_results, omic = "all",
                        absolute = FALSE, descending = TRUE, factor = 1)
```

---

predict\_gs

*Prediction of new global scores based on block loadings and weights*

---

**Description**

Uses previously-computed block scores and weights to compute a global score for new data. Only validated for MCIA results, as CPCA loadings aren't compatible with un-deflated data.

**Usage**

```
predict_gs(mcia_results, test_data)
```

**Arguments**

|              |                                                                                                                       |
|--------------|-----------------------------------------------------------------------------------------------------------------------|
| mcia_results | an mcia object output by nipals_multiblock() containing block scores, weights, and pre-processing identifier.         |
| test_data    | an MAE object with the same block types and features as the training dataset. Feature and omic order must match 'bl'. |

**Details**

Projects the new observations onto each block loadings vector, then weights the projection according to the corresponding block weights.

**Value**

a matrix of predicted global scores for the training data

**Examples**

```
data(NCI60)
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",
                              colData=metadata_NCI60)
mcia_results <- nipals_multiblock(data_blocks_mae, num_PCs = 2)
new_data <- data_blocks_mae # should update with a truly new dataset
preds <- predict_gs(mcia_results, new_data)
```

---

|                 |                        |
|-----------------|------------------------|
| projection_plot | <i>projection_plot</i> |
|-----------------|------------------------|

---

**Description**

Function to generate a projection plot of MCIA results.

**Usage**

```
projection_plot(
  mcia_results,
  projection,
  orders = c(1, 2),
  block_name = NULL,
  color_col = NULL,
  color_pal = scales::viridis_pal,
  color_pal_params = list(option = "E"),
  legend_loc = "bottomleft",
  color_override = NULL,
  cex = 0.5
)
```

**Arguments**

|                  |                                                                                                                                                                                                                                                                                                                                                                |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| mcia_results     | MCIA results object returned from ‘nipals_multiblock’                                                                                                                                                                                                                                                                                                          |
| projection       | of plot, with the following options <ul style="list-style-type: none"> <li>• ‘all’ - scatter plot of two orders of global and block scores (aka factors).</li> <li>• ‘global’ - scatter plot of two orders of global scores only (aka factors).</li> <li>• ‘block’ - scatter plot of two orders of block scores only (aka factors) for given block.</li> </ul> |
| orders           | Option to select orders of factors to plot against each other (for projection plots)                                                                                                                                                                                                                                                                           |
| block_name       | Name of the block to be plotted (if ‘projection = block’ argument used).                                                                                                                                                                                                                                                                                       |
| color_col        | an integer or string specifying the column that will be used for color_col                                                                                                                                                                                                                                                                                     |
| color_pal        | a list of colors or function which returns a list of colors                                                                                                                                                                                                                                                                                                    |
| color_pal_params | a list of parameters for the color function                                                                                                                                                                                                                                                                                                                    |
| legend_loc       | Option for legend location, or "none" for no legend.                                                                                                                                                                                                                                                                                                           |
| color_override   | Option to override colors when necessary, helpful for projection = "global" or "block"                                                                                                                                                                                                                                                                         |
| cex              | Resizing parameter for drawing the points                                                                                                                                                                                                                                                                                                                      |



---

|              |                                  |
|--------------|----------------------------------|
| vis_load_ord | <i>Visualize ranked loadings</i> |
|--------------|----------------------------------|

---

## Description

Visualize a scree plot of loadings recovered from nipalsMCIA() output loadings matrix ranked using the ord\_loadings() functions

## Usage

```
vis_load_ord(
  mcia_results,
  omic,
  factor = 1,
  n_feat = 15,
  absolute = TRUE,
  descending = TRUE,
  color_pal = scales::viridis_pal,
  color_pal_params = list()
)
```

## Arguments

|                  |                                                                             |
|------------------|-----------------------------------------------------------------------------|
| mcia_results     | object returned from nipals_multiblock() function                           |
| omic             | name of the given omic dataset                                              |
| factor           | choose a factor (numeric value from 1 to number of factors in mcia_results) |
| n_feat           | number of features to visualize                                             |
| absolute         | whether to rank by absolute value                                           |
| descending       | whether to rank in descending or ascending order                            |
| color_pal        | a list of colors or function which returns a list of colors                 |
| color_pal_params | a list of parameters for the color function                                 |

## Value

Plot in features for a factor by rank

## Examples

```
data(NCI60)
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",
                              colData=metadata_NCI60)
mcia_results <- nipals_multiblock(data_blocks_mae, num_PCs = 10,
                                  plots = "none", tol = 1e-12)
vis_load_ord(mcia_results, omic="mrna")
```

---

|               |                                                  |
|---------------|--------------------------------------------------|
| vis_load_plot | <i>Visualize all loadings on two factor axes</i> |
|---------------|--------------------------------------------------|

---

**Description**

Visualize all loadings recovered from nipalsMCIA() output loadings matrix ranked using across two factor axes

**Usage**

```
vis_load_plot(  
  mcia_results,  
  axes = c(1, 2),  
  color_pal = scales::viridis_pal,  
  color_pal_params = list()  
)
```

**Arguments**

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| mcia_results     | object returned from nipals_multiblock() function            |
| axes             | list of two numbers associated with two factors to visualize |
| color_pal        | a list of colors or function which returns a list of colors  |
| color_pal_params | a list of parameters for the color function                  |

**Value**

Plot of MCIA feature loadings for chosen axes

**Examples**

```
data(NCI60)  
data_blocks_mae <- simple_mae(data_blocks,row_format="sample",  
                             colData=metadata_NCI60)  
mcia_results <- nipals_multiblock(data_blocks_mae, num_PCs = 10,  
                                plots = "none", tol = 1e-12)  
vis_load_plot(mcia_results, axes = c(1, 4))
```

# Index

- \* **data**
  - data\_blocks, [6](#)
  - metadata\_NCI60, [13](#)
- \* **internal**
  - nipalsMCIA-package, [3](#)
- \* **multi-omics**
  - data\_blocks, [6](#)
  - metadata\_NCI60, [13](#)
- block\_preproc, [3](#)
- block\_weights\_heatmap, [4](#)
- cc\_preproc, [5](#)
- col\_preproc, [5](#)
- data\_blocks, [6](#)
- deflate\_block\_bl, [7](#)
- deflate\_block\_gs, [7](#)
- extract\_from\_mae, [8](#)
- get\_colors, [9](#)
- get\_metadata\_colors, [9](#)
- get\_tv, [10](#)
- global\_scores\_eigenvalues\_plot, [11](#)
- global\_scores\_heatmap, [11](#)
- gsea\_report, [12](#)
- metadata\_NCI60, [13](#)
- nipals\_iter, [14](#)
- nipals\_multiblock, [14](#)
- nipalsMCIA (nipalsMCIA-package), [3](#)
- nipalsMCIA-package, [3](#)
- NipalsResult (NipalsResult-class), [13](#)
- NipalsResult-class, [13](#)
- nmb\_get\_bl, [16](#)
- nmb\_get\_bs, [17](#)
- nmb\_get\_bs\_weights, [18](#)
- nmb\_get\_eigs, [18](#)
- nmb\_get\_gl, [19](#)
- nmb\_get\_gs, [19](#)
- nmb\_get\_metadata, [20](#)
- ord\_loadings, [20](#)
- predict\_gs, [21](#)
- projection\_plot, [22](#)
- simple\_mae, [23](#)
- vis\_load\_ord, [24](#)
- vis\_load\_plot, [25](#)