Package 'slideimp'

April 16, 2026

Type Package

Title Numeric Matrices K-NN and PCA Imputation

Version 1.0.0

Description Fast k-nearest neighbors (K-NN) and principal component analysis (PCA) imputation algorithms for missing values in high-dimensional numeric matrices, i.e., epigenetic data. For extremely high-dimensional data with ordered features, a sliding window approach for K-NN or PCA imputation is provided. Additional features include group-wise imputation (e.g., by chromosome), hyperparameter tuning with repeated cross-validation, multi-core parallelization, and optional subset imputation. The K-NN algorithm is described in: Hastie, T., Tibshirani, R., Sherlock, G., Eisen, M., Brown, P. and Botstein, D. (1999) ``Imputing Missing Data for Gene Expression Arrays". The PCA imputation is an optimized version of the imputePCA() function from the 'missMDA' package described in: Josse, J. and Husson, F. (2016) <doi:10.18637/jss.v070.i01> ``missMDA: A Package for Handling Missing Values in Multivariate Data Analysis".

License GPL (>= 2)

URL https://github.com/hhp94/slideimp, https://hhp94.github.io/slideimp/

BugReports https://github.com/hhp94/slideimp/issues

Depends R (>= 4.1.0)

Imports bigmemory, carrier, checkmate, cli, collapse, mirai, Rcpp, stats

Suggests knitr, missMDA, RhpcBLASctl, rmarkdown, testthat (>= 3.0.0)

LinkingTo mlpack, Rcpp, RcppArmadillo, RcppEnsmallen

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

NeedsCompilation yes

Author Hung Pham [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-8271-9355>)

Maintainer Hung Pham <amser.hoanghung@gmail.com>

Repository CRAN

Date/Publication 2026-04-16 21:50:11 UTC

Contents

col_vars 2
compute_metrics 3
group_imp 4
knn_imp 8
mean_imp_col 10
pca_imp 11
prep_groups 12
print.slideimp_results 14
print.slideimp_sim 15
print.slideimp_tbl 15
register_group_resolver 16
sample_na_loc 17
sim_mat 18
slide_imp 20
tune_imp 23

Index **28**

`col_vars` *Calculate Matrix Column Variance*

Description

`col_vars` computes the sample variance for each column of a numeric matrix.

Usage

```
col_vars(mat, cores = 1)
```

Arguments

`mat`	A numeric matrix.
`cores`	Number of cores to use for parallel computation. Defaults to 1.

Details

Variances for columns with one unique value after dropping NA are set to NA.

Value

col_vars returns a named numeric vector of column variances.

Examples

```
mat <- matrix(rnorm(4 * 10), ncol = 4)
mat[1, 1] <- NA
mat[1:8, 2] <- NA
mat[1:9, 3] <- NA
mat[, 4] <- NA
mat
col_vars(mat)
apply(mat, 2, var, na.rm = TRUE)
```

---

compute_metrics *Compute Prediction Accuracy Metrics*

Description

Computes prediction accuracy metrics for results from tune_imp().

Usage

```
compute_metrics(results, metrics = c("mae", "rmse"))

## S3 method for class 'data.frame'
compute_metrics(results, metrics = c("mae", "rmse"))

## S3 method for class 'slideimp_tune'
compute_metrics(results, metrics = c("mae", "rmse"))
```

Arguments

results	A slideimp_tune data.frame from tune_imp(). Must contain a result list-column with data.frames that have truth and estimate columns.
metrics	A character vector of metric names to compute. Defaults to c("mae", "rmse"). Also available: "mape", "bias", "rsq", and "rsq_trad".

Details

For alternative or faster metrics, see the {yardstick} package.

Value

A data.frame with the original parameters along with unnested metrics: `.metric`, `.estimator`, and `.estimate`.

Examples

```
obj <- sim_mat(100, 100)$input

set.seed(1234)
results <- tune_imp(
  obj = obj,
  parameters = data.frame(k = 10),
  .f = "knn_imp",
  n_reps = 1,
  num_na = 20
)

compute_metrics(results)
```

---

group_imp *Grouped K-NN or PCA Imputation*

Description

Perform K-NN or PCA imputation independently on feature groups (e.g., by chromosomes, flanking probes, or clustering-based groups).

Usage

```
group_imp(
  obj,
  group,
  subset = NULL,
  allow_unmapped = FALSE,
  k = NULL,
  ncp = NULL,
  method = NULL,
  cores = 1,
  .progress = TRUE,
  min_group_size = NULL,
  colmax = NULL,
  post_imp = NULL,
  dist_pow = NULL,
  tree = NULL,
  max_cache = NULL,
  scale = NULL,
  coeff.ridge = NULL,
```

```
  threshold = NULL,
  row.w = NULL,
  seed = NULL,
  nb.init = NULL,
  maxiter = NULL,
  miniter = NULL,
  pin_blas = FALSE,
  na_check = TRUE,
  on_infeasible = c("error", "skip", "mean")
)
```

Arguments

`obj`	A numeric matrix with **samples in rows** and **features in columns**.
`group`	Specification of how features should be grouped for imputation. Accepts three formats: • `character`: string naming a supported Illumina platform; see the Note section. • `data.frame` (Long format): – `group`: Column identifying the group for each feature. – `feature`: Character column of individual feature names. • `data.frame` (List-column format): – `feature`: List-column of character vectors to impute. A row is a group. – `aux`: (Optional) List-column of auxiliary names used for context. – `parameters`: (Optional) List-column of group-specific parameter lists.
`subset`	Character vector of feature names to impute (default `NULL` means impute all features). Must be a subset of `obj_cn` (`colnames(obj)`) and must appear in at least one group's `feature`. Features in a group but not in `subset` are demoted to auxiliary columns for that group. Groups left with zero features after demotion are dropped with a message.
`allow_unmapped`	Logical. If `FALSE`, every column in `colnames(obj)` must appear in `group`. If `TRUE`, columns with no group assignment are left untouched (neither imputed nor used as auxiliary columns) and a message is issued instead of an error.
`k`	Integer. Number of nearest neighbors for imputation. 10 is a good starting point.
`ncp`	Integer. Number of components used to predict the missing entries.
`method`	For K-NN imputation: distance metric to use (`"euclidean"` or `"manhattan"`). For PCA imputation: regularization imputation algorithm (`"regularized"` or `"EM"`).
`cores`	The number of OpenMP cores for K-NN imputation **only**. For PCA or mirai-based parallelism, use `mirai::daemons()` instead.
`.progress`	Show imputation progress (default `TRUE`).
`min_group_size`	Integer or `NULL`. Minimum column count (features + aux) per group. Groups smaller than this are padded with randomly sampled columns from `obj`. Passed to `prep_groups()` internally.

colmax	Numeric. A number from 0 to 1. Threshold of column-wise missing data rate above which imputation is skipped.
post_imp	Boolean. Whether to impute remaining missing values (those that failed imputation) using column means.
dist_pow	Numeric. The amount of penalization for further away nearest neighbors in the weighted average. dist_pow = 0 (default) is the simple average of the nearest neighbors.
tree	Logical. FALSE (default) uses brute-force K-NN. TRUE uses mlpack BallTree.
max_cache	Numeric. Maximum allowed cache size in GB (default 4). When greater than 0, pairwise distances between columns with missing values are pre-computed and cached, which is faster for moderate-sized data but uses O(m^2) memory where m is the number of columns with missing values. Set to 0 to disable caching and trade speed for lower memory usage.
scale	Logical. If TRUE (default), variables are scaled to have unit variance.
coeff.ridge	Numeric. Ridge regularization coefficient (default is 1). Only used if method = "regularized". Values < 1 regularize less (closer to EM); values > 1 regularize more (closer to mean imputation).
threshold	Numeric. The threshold for assessing convergence.
row.w	Row weights (internally normalized to sum to 1). Can be one of: • NULL (default): All rows weighted equally. • A numeric vector: Custom positive weights of length nrow(obj). • "n_miss": Rows with more missing values receive lower weight.
seed	Numeric or NULL. Random seed for reproducibility.
nb.init	Integer. Number of random initializations. The first initialization is always mean imputation.
maxiter	Integer. Maximum number of iterations for the algorithm.
miniter	Integer. Minimum number of iterations for the algorithm.
pin_blas	Logical. If TRUE, pin BLAS threads to 1 to reduce contention when using parallel PCA on systems linked with multi-threaded BLAS.
na_check	Boolean. Check for leftover NA values in the results or not (internal use).
on_infeasible	Character, one of "error" (default on group_imp()), "skip", or "mean" (default on slide_imp()). Controls behaviour when a group is infeasible for imputation, e.g., k/ncp exceeds the number of usable columns after applying colmax, or all subset columns in the group exceed colmax.

Details

Performs K-NN or PCA imputation on groups of features independently, which significantly reduces imputation time for large datasets.

Specify k and related arguments to use K-NN, or ncp and related arguments for PCA imputation. If both k and ncp are NULL, group$parameters must supply either k or ncp for every group.

Parameter resolution:

Group-wise parameters (in `group$parameters`) take priority; global arguments (`k`, `ncp`, `method`, etc.) fill in any gaps. All groups must use the same imputation method. Per-group `k` is capped at `group_size - 1` and `ncp` at `min(nrow(group) - 2L, ncol(group) - 1L)`, with a warning when capping occurs.

Grouping strategies:

- Chromosomal grouping to break down the search space.
- Flanking-probe groups for spatially local imputation.
- Column-clustering to form correlation-based groups.

Value

A numeric matrix of the same dimensions as `obj` with missing values imputed.

Parallelization

- **K-NN**: use the `cores` argument (requires OpenMP). If `mirai::daemons()` are active, `cores` is automatically set to 1 to avoid nested parallelism.
- **PCA**: use `mirai::daemons()` instead of `cores`.

On macOS, OpenMP is typically unavailable and `cores` falls back to

1. Use `mirai::daemons()` for parallelization instead.

On Linux with OpenBLAS or MKL, set `pin_blas = TRUE` when running parallel PCA to prevent BLAS threads and `mirai` workers competing for cores.

Note

A `character` string can be passed to `group` to name a supported Illumina platform (e.g., `"EPICv2"`, `"EPICv2_deduped"`), which fetches the manifest automatically. This requires the `slideimp.extra` package (available on GitHub; see its README for installation instructions). Supported platforms are listed in `slideimp.extra::slideimp_arrays`.

See Also

`prep_groups()`

Examples

```
# Generate example data with missing values
set.seed(1234)
to_test <- sim_mat(10, 20, perc_total_na = 0.05, perc_col_na = 1)
obj <- to_test$input
group <- to_test$col_group # metadata that maps `colnames(obj)` to groups
head(group)

# Simple grouped K-NN imputation
results <- group_imp(obj, group = group, k = 2)
```

```
# Impute only a subset of features
subset_features <- sample(to_test$col_group$feature, size = 10)
knn_subset <- group_imp(obj, group = group, subset = subset_features, k = 2)

# Use prep_groups() to inspect and tweak per-group parameters
prepped <- prep_groups(colnames(obj), group)
prepped$parameters <- lapply(seq_len(nrow(prepped)), \(i) list(k = 2))
prepped$parameters[[2]]$k <- 4
knn_grouped <- group_imp(obj, group = prepped, cores = 2)

# PCA imputation with mirai parallelism
mirai::daemons(2)
pca_grouped <- group_imp(obj, group = group, ncp = 2)
mirai::daemons(0)
pca_grouped
```

---

knn_imp *K-Nearest Neighbor Imputation for Numeric Matrices*

Description

Impute missing values in a numeric matrix using k-nearest neighbors (K-NN).

Usage

```
knn_imp(
  obj,
  k,
  colmax = 0.9,
  method = c("euclidean", "manhattan"),
  cores = 1,
  post_imp = TRUE,
  subset = NULL,
  dist_pow = 0,
  tree = FALSE,
  max_cache = 4,
  na_check = TRUE
)
```

Arguments

obj	A numeric matrix with **samples in rows** and **features in columns**.
k	Integer. Number of nearest neighbors for imputation. 10 is a good starting point.
colmax	Numeric. A number from 0 to 1. Threshold of column-wise missing data rate above which imputation is skipped.
method	Character. Either "euclidean" (default) or "manhattan". Distance metric for nearest neighbor calculation.

cores	Integer. Number of cores for K-NN parallelization (OpenMP). On macOS, OpenMP may need additional compiler configuration.
post_imp	Boolean. Whether to impute remaining missing values (those that failed imputation) using column means.
subset	Character. Vector of column names or integer vector of column indices specifying which columns to impute.
dist_pow	Numeric. The amount of penalization for further away nearest neighbors in the weighted average. dist_pow = 0 (default) is the simple average of the nearest neighbors.
tree	Logical. FALSE (default) uses brute-force K-NN. TRUE uses mlpack BallTree.
max_cache	Numeric. Maximum allowed cache size in GB (default 4). When greater than 0, pairwise distances between columns with missing values are pre-computed and cached, which is faster for moderate-sized data but uses O(m^2) memory where m is the number of columns with missing values. Set to 0 to disable caching and trade speed for lower memory usage.
na_check	Boolean. Check for leftover NA values in the results or not (internal use).

Details

This function performs imputation **column-wise** (using rows as observations).

When dist_pow > 0, imputed values are computed as distance-weighted averages where weights are inverse distances raised to the power of dist_pow.

The tree parameter (when TRUE) uses a BallTree for faster neighbor search via {mlpack} but **requires pre-filling** missing values with column means. This can introduce a small bias when missingness is high.

Value

A numeric matrix of the same dimensions as obj with missing values imputed.

Performance Optimization

- tree = FALSE (default, brute-force K-NN): Always safe and usually faster for small to moderate data or high-dimensional cases.
- tree = TRUE (BallTree K-NN): Only use when imputation run time becomes prohibitive and missingness is low (<5% missing).
- **Subset imputation**: Use the subset parameter for efficiency when only specific columns need imputation (e.g., epigenetic clock CpGs).

References

Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman RB (2001). Missing value estimation methods for DNA microarrays. Bioinformatics 17(6): 520-525.

Examples

```
# Basic K-NN imputation
obj <- sim_mat(20, 20, perc_col_na = 1)$input
sum(is.na(obj))
result <- knn_imp(obj, k = 10)
result
```

`mean_imp_col` *Column Mean Imputation*

Description

Impute missing values in a matrix by replacing them with the mean of their respective columns.

Usage

```
mean_imp_col(obj, subset = NULL, cores = 1)
```

Arguments

`obj`	A numeric matrix with **samples in rows** and **features in columns**.
`subset`	Character. Vector of column names or integer vector of column indices specifying which columns to impute.
`cores`	Integer. Number of cores for K-NN parallelization (OpenMP). On macOS, OpenMP may need additional compiler configuration.

Value

A numeric matrix of the same dimensions as `obj` with missing values in the specified columns replaced by column means.

Examples

```
# Create example matrix with missing values
mat <- matrix(c(1, 2, NA, 4, NA, 6, NA, 8, 9), nrow = 3)
colnames(mat) <- c("A", "B", "C")
mat

# Impute missing values with column means
imputed_mat <- mean_imp_col(mat)
imputed_mat

# Impute only specific columns by name
imputed_subset <- mean_imp_col(mat, subset = c("A", "C"))
imputed_subset

# Impute only specific columns by index
imputed_idx <- mean_imp_col(mat, subset = c(1, 3))
imputed_idx
```

pca_imp *Impute Numeric Matrix with PCA Imputation*

Description

Impute missing values in a numeric matrix using (regularized) iterative PCA.

Usage

```
pca_imp(
  obj,
  ncp = 2,
  scale = TRUE,
  method = c("regularized", "EM"),
  coeff.ridge = 1,
  row.w = NULL,
  threshold = 1e-06,
  seed = NULL,
  nb.init = 1,
  maxiter = 1000,
  miniter = 5,
  colmax = 0.9,
  post_imp = TRUE,
  na_check = TRUE
)
```

Arguments

obj	A numeric matrix with **samples in rows** and **features in columns**.
ncp	Integer. Number of components used to predict the missing entries.
scale	Logical. If TRUE (default), variables are scaled to have unit variance.
method	Character. Either "regularized" (default) or "EM".
coeff.ridge	Numeric. Ridge regularization coefficient (default is 1). Only used if method = "regularized". Values < 1 regularize less (closer to EM); values > 1 regularize more (closer to mean imputation).
row.w	Row weights (internally normalized to sum to 1). Can be one of: • NULL (default): All rows weighted equally. • A numeric vector: Custom positive weights of length nrow(obj). • "n_miss": Rows with more missing values receive lower weight.
threshold	Numeric. The threshold for assessing convergence.
seed	Integer. Random number generator seed.
nb.init	Integer. Number of random initializations. The first initialization is always mean imputation.
maxiter	Integer. Maximum number of iterations for the algorithm.

miniter	Integer. Minimum number of iterations for the algorithm.
colmax	Numeric. A number from 0 to 1. Threshold of column-wise missing data rate above which imputation is skipped.
post_imp	Boolean. Whether to impute remaining missing values (those that failed imputation) using column means.
na_check	Boolean. Check for leftover NA values in the results or not (internal use).

Details

This algorithm is based on the original missMDA::imputePCA function and is optimized for tall or wide numeric matrices.

Value

A numeric matrix of the same dimensions as obj with missing values imputed.

Author(s)

Francois Husson and Julie Josse (original missMDA implementation).

References

Josse, J. & Husson, F. (2013). Handling missing values in exploratory multivariate data analysis methods. Journal de la SFdS. 153 (2), pp. 79-99.

Josse, J. and Husson, F. (2016). missMDA: A Package for Handling Missing Values in Multivariate Data Analysis. Journal of Statistical Software, 70 (1), pp 1-31. doi:10.18637/jss.v070.i01

Examples

```
obj <- sim_mat(10, 10)$input
sum(is.na(obj))
obj[1:4, 1:4]
# Randomly initialize missing values 5 times (1st time is mean).
pca_imp(obj, ncp = 2, nb.init = 5)
```

---

prep_groups *Prepare Groups for Imputation*

Description

Normalize and validate a grouping specification for use with group_imp(). Converts long-format or canonical list-column input into a validated slideimp_tbl, enforcing set relationships, pruning dropped columns, and optionally padding small groups.

Usage

```
prep_groups(
  obj_cn,
  group,
  subset = NULL,
  min_group_size = 0,
  allow_unmapped = FALSE,
  seed = NULL
)
```

Arguments

obj_cn	Character vector of column names from the data matrix (e.g., colnames(obj)). Every element must appear in group$feature unless allow_unmapped = TRUE.
group	Specification of how features should be grouped for imputation. Accepts three formats: • character: string naming a supported Illumina platform; see the Note section. • data.frame (Long format): – group: Column identifying the group for each feature. – feature: Character column of individual feature names. • data.frame (List-column format): – feature: List-column of character vectors to impute. A row is a group. – aux: (Optional) List-column of auxiliary names used for context. – parameters: (Optional) List-column of group-specific parameter lists.
subset	Character vector of feature names to impute (default NULL means impute all features). Must be a subset of obj_cn (colnames(obj)) and must appear in at least one group's feature. Features in a group but not in subset are demoted to auxiliary columns for that group. Groups left with zero features after demotion are dropped with a message.
min_group_size	Integer or NULL. Minimum column count (features + aux) per group. Groups smaller than this are padded with randomly sampled columns from obj. Passed to prep_groups() internally.
allow_unmapped	Logical. If FALSE, every column in colnames(obj) must appear in group. If TRUE, columns with no group assignment are left untouched (neither imputed nor used as auxiliary columns) and a message is issued instead of an error.
seed	Numeric or NULL. Random seed for reproducibility.

Details

Set Validation:

Let A = obj_cn and B = the union of all feature and auxiliary names in group. The function enforces $A \subseteq B$: every column in the matrix must appear somewhere in the manifest.

- Pruning: Elements in B but not in A (e.g., QC-dropped probes) are silently pruned from each group.

- `Dropping:` Groups left with zero features after pruning are removed entirely with a diagnostic message.

Value

A `data.frame` of class `slideimp_tbl` containing:

- `group`: Original group labels (if provided) or sequential group labels.
- `feature`: A list-column of character vectors (feature names).
- `aux`: A list-column of character vectors (auxiliary names).
- `parameters`: A list-column of per-group configuration lists.

See Also

`group_imp()`

`print.slideimp_results` *Print a* `slideimp_results` *Object*

Description

Print the output of `knn_imp()`, `pca_imp()`, `group_imp()`, `slide_imp()`.

Usage

```
## S3 method for class 'slideimp_results'
print(x, n = 6L, p = 6L, ...)
```

Arguments

`x`	A `slideimp_results` object.
`n`	Number of rows to print.
`p`	Number of cols to print.
`...`	Not used.

Value

Invisible `x`.

Examples

```
set.seed(1234)
mat <- sim_mat(n = 10, p = 10)
result <- knn_imp(mat$input, k = 5)
class(result)
print(result, n = 6, p = 6)
```

print.slideimp_sim *Print a* slideimp_sim *Object*

Description

Print the output of sim_mat().

Usage

```
## S3 method for class 'slideimp_sim'
print(x, n = 6L, p = 6L, ...)
```

Arguments

x	A slideimp_sim object.
n	Number of rows of each component to show.
p	Number of columns of input to show.
...	Not used.

Value

Invisible x.

Examples

```
set.seed(123)
sim_data <- sim_mat(n = 50, p = 10, rho = 0.5)
class(sim_data)
print(sim_data)
```

print.slideimp_tbl *Print a* slideimp_tbl *Object*

Description

Print slideimp_tbl objects (which inherit data.frame) with nicer looking list-columns (similar to tibble).

Usage

```
## S3 method for class 'slideimp_tbl'
print(x, n = NULL, ...)
```

Arguments

x	A slideimp_tbl object.
n	Number of rows to show. Defaults to 10.
...	Not used.

Value

Invisible x.

Examples

```
mat <- sim_mat(n = 10, p = 500)
set.seed(1234)
results <- tune_imp(mat$input, parameters = data.frame(k = 5), .f = "knn_imp")
class(results)
print(results)
```

---

register_group_resolver

Register a Group Resolver

Description

Called by the companion package slideimp.extra from their .onLoad() to register a callback that turns character manifest into a data.frame for group_imp()/prep_groups().

Usage

```
register_group_resolver(resolver)
```

Arguments

resolver	A function that takes the manifest character and returns a data.frame.

Value

Invisibly returns NULL. The function is called only for its side effect of registering the resolver.

Examples

```
# Typically called from `.onLoad` by `slideimp.extra` package, not by users directly.
dummy_resolver <- function(x) data.frame(feature = character(), group = character())
register_group_resolver(dummy_resolver)
```

sample_na_loc *Sample Missing Value Locations with Constraints*

Description

Samples indices for NA injection into a matrix while maintaining row/column missing value budgets and avoiding zero-variance columns.

Usage

```
sample_na_loc(
  obj,
  n_cols = NULL,
  n_rows = 2L,
  num_na = NULL,
  n_reps = 1L,
  rowmax = 0.9,
  colmax = 0.9,
  na_col_subset = NULL,
  max_attempts = 100
)
```

Arguments

obj	A numeric matrix with **samples in rows** and **features in columns**.
n_cols	Integer. The number of columns to receive injected NA per repetition. Ignored when num_na is supplied (in which case n_cols is derived as num_na %/% n_rows). Must be provided if num_na is NULL. Ignored in tune_imp() when na_loc is supplied.
n_rows	Integer. The target number of NA values per column (default 2L). • When num_na is supplied: used as the base size. Most columns receive exactly n_rows missing values; num_na %% n_rows columns receive one extra. If there's only one column, it receives all the remainder. • When num_na is NULL: every selected column receives exactly n_rows NA. Ignored in tune_imp() when na_loc is supplied.
num_na	Integer. Total number of missing values to inject per repetition. If supplied, n_cols is computed automatically and missing values are distributed as evenly as possible, using n_rows as the base size (num_na must be at least n_rows). If omitted but n_cols is supplied, the total injected is n_cols * n_rows. If num_na, n_cols, and na_loc are all NULL, tune_imp() defaults to roughly 5% of total cells, capped at 500. sample_na_loc() has no default. Ignored in tune_imp() when na_loc is supplied.
n_reps	Integer. Number of repetitions for random NA injection (default 1).
rowmax, colmax	Numbers between 0 and 1. NA injection cannot create rows/columns with a higher proportion of missing values than these thresholds.

`na_col_subset` Optional integer or character vector restricting which columns of `obj` are eligible for NA injection.

- If `NULL` (default): all columns are eligible.
- If character: values must exist in `colnames(obj)`.
- If integer/numeric: values must be valid 1-based column indices. The vector must be unique and must contain at least `n_cols` columns (or the number derived from `num_na`). Ignored in `tune_imp()` when `na_loc` is supplied.

`max_attempts` Integer. Maximum number of resampling attempts per repetition before giving up due to row-budget exhaustion (default `100`).

Details

The function uses a greedy stochastic search for valid `NA` locations. It ensures that:

- Total missingness per row and column does not exceed `rowmax` and `colmax`.
- At least two distinct observed values are preserved in every column to ensure the column maintains non-zero variance.

Value

A list of length `n_reps`. Each element is a two-column integer matrix (`row`, `col`) representing the coordinates of the sampled `NA` locations.

Examples

```
mat <- matrix(runif(100), nrow = 10)

# Sample 5 `NA` across 5 columns (1 per column)
locs <- sample_na_loc(mat, n_cols = 5, n_rows = 1)
locs

# Inject the `NA` from the first repetition
mat[locs[[1]]] <- NA
mat
```

---

sim_mat *Simulate Matrix with Metadata*

Description

Generates a matrix of random normal data, then optionally scales values between 0 and 1 column-wise. It also creates corresponding data frames for feature (column) and sample (row) metadata and can optionally introduce `NA` values into a specified proportion of rows. A correlation between columns `rho` (before scaling) can be added.

Usage

```
sim_mat(
  n = 100,
  p = 100,
  rho = 0.5,
  n_col_groups = 2,
  n_row_groups = 1,
  perc_total_na = 0.1,
  perc_col_na = 0.5,
  beta = TRUE
)
```

Arguments

n	An integer specifying the number of rows (samples). Default is 100.
p	An integer specifying the number of columns (features). Default is 100.
rho	Columns correlation before scaling (compound symmetry). Default is 0.5.
n_col_groups	An integer for the number of groups to assign to features/columns. Default is 2.
n_row_groups	An integer for the number of groups to assign to samples/rows. Default is 1.
perc_total_na	Proportion of all cells to set to NA. Default is 0.1.
perc_col_na	Proportion of columns across which those NAs are spread. Default is 0.5.
beta	If TRUE (default) scale values between 0 and 1 column wise.

Value

An object of class slideimp_sim. This is a list containing:

- input: A numeric matrix of dimension $n \times p$ containing the simulated values and injected NAs.
- col_group: A data frame with p rows mapping each feature to a group.
- row_group: A data frame with n rows mapping each sample to a group.

Examples

```
set.seed(123)
sim_data <- sim_mat(n = 50, p = 10, rho = 0.5)
sim_data
```

`slide_imp` *Sliding Window K-NN or PCA Imputation*

Description

Performs sliding window K-NN or PCA imputation of large numeric matrices column-wise. This method assumes that columns are meaningfully sorted by `location`.

Usage

```
slide_imp(
  obj,
  location,
  window_size,
  overlap_size = 0,
  flank = FALSE,
  min_window_n,
  subset = NULL,
  dry_run = FALSE,
  k = NULL,
  cores = 1,
  dist_pow = 0,
  max_cache = 4,
  ncp = NULL,
  scale = TRUE,
  coeff.ridge = 1,
  seed = NULL,
  row.w = NULL,
  nb.init = 1,
  maxiter = 1000,
  miniter = 5,
  method = NULL,
  .progress = TRUE,
  colmax = 0.9,
  post_imp = TRUE,
  na_check = TRUE,
  on_infeasible = c("skip", "error", "mean")
)
```

Arguments

`obj`	A numeric matrix with **samples in rows** and **features in columns**.
`location`	A sorted numeric vector of length `ncol(obj)` giving the position of each column (e.g., genomic coordinates). Used to define sliding windows.
`window_size`	Window width in the same units as `location`.

`overlap_size` Overlap between consecutive windows in the same units as `location`. Must be less than `window_size`. Default is `0`. Ignored when `flank = TRUE`.

`flank` Logical. If `TRUE`, instead of sliding windows across the whole matrix, one window of width `window_size` is created flanking each feature listed in `subset`. In this mode `overlap_size` is ignored. Requires `subset` to be provided. Default = `FALSE`.

`min_window_n` Minimum number of columns a window must contain to be imputed. Windows smaller than this are not imputed. `k` and `ncp` must also be smaller than `min_window_n`.

`subset` Character. Vector of column names or integer vector of column indices specifying which columns to impute.

`dry_run` Logical. If `TRUE`, skip imputation and return a `slideimp_tbl` object of the windows that *would* be used (after all dropping rules). `k`/`ncp` are not required in this mode. Columns: `start`, `end`, `window_n`, plus `subset_local` (list-column of local subset indices) when `flank = FALSE`, or `target` and `subset_local` when `flank = TRUE`. Default = `FALSE`.

`k` Integer. Number of nearest neighbors for imputation. 10 is a good starting point.

`cores` Integer. Number of cores for K-NN parallelization (OpenMP). On macOS, OpenMP may need additional compiler configuration.

`dist_pow` Numeric. The amount of penalization for further away nearest neighbors in the weighted average. `dist_pow = 0` (default) is the simple average of the nearest neighbors.

`max_cache` Numeric. Maximum allowed cache size in GB (default `4`). When greater than `0`, pairwise distances between columns with missing values are pre-computed and cached, which is faster for moderate-sized data but uses O(m^2) memory where m is the number of columns with missing values. Set to `0` to disable caching and trade speed for lower memory usage.

`ncp` Integer. Number of components used to predict the missing entries.

`scale` Logical. If `TRUE` (default), variables are scaled to have unit variance.

`coeff.ridge` Numeric. Ridge regularization coefficient (default is 1). Only used if `method = "regularized"`. Values < 1 regularize less (closer to EM); values > 1 regularize more (closer to mean imputation).

`seed` Integer. Random number generator seed.

`row.w` Row weights (internally normalized to sum to 1). Can be one of:

- `NULL` (default): All rows weighted equally.
- A numeric vector: Custom positive weights of length `nrow(obj)`.
- `"n_miss"`: Rows with more missing values receive lower weight.

`nb.init` Integer. Number of random initializations. The first initialization is always mean imputation.

`maxiter` Integer. Maximum number of iterations for the algorithm.

`miniter` Integer. Minimum number of iterations for the algorithm.

`method` For K-NN imputation: distance metric to use (`"euclidean"` or `"manhattan"`). For PCA imputation: regularization imputation algorithm (`"regularized"` or `"EM"`).

.progress	Show progress bar (default = TRUE).
colmax	Numeric. A number from 0 to 1. Threshold of column-wise missing data rate above which imputation is skipped.
post_imp	Boolean. Whether to impute remaining missing values (those that failed imputation) using column means.
na_check	Boolean. Check for leftover NA values in the results or not (internal use).
on_infeasible	Character, one of "error" (default on group_imp()), "skip", or "mean" (default on slide_imp()). Controls behaviour when a group is infeasible for imputation, e.g., k/ncp exceeds the number of usable columns after applying colmax, or all subset columns in the group exceed colmax.

Details

The sliding window approach divides the input matrix into smaller segments based on location values and applies imputation to each window independently. Values in overlapping areas are averaged across windows to produce the final imputed result.

Two windowing modes are supported:

- flank = FALSE (default): Greedily partitions the location vector into windows of width window_size with the requested overlap_size between consecutive windows.
- flank = TRUE: Creates one window per feature in subset that exactly flanks that specific feature using the supplied window_size.

Specify k and related arguments to use knn_imp(), ncp and related arguments for pca_imp().

Value

A numeric matrix of the same dimensions as obj with missing values imputed. When dry_run = TRUE, returns a data.frame of class slideimp_tbl with columns start, end, window_n, plus subset_local (and target when flank = TRUE).

Examples

```
# Generate sample data with missing values with 20 samples and 100 columns
# where the column order is sorted (i.e., by genomic position)
set.seed(1234)
beta_matrix <- sim_mat(20, 100)$input
location <- 1:100

# It's very useful to first perform a dry run to examine the calculated windows
windows_statistics <- slide_imp(
  beta_matrix,
  location = location,
  window_size = 50,
  overlap_size = 10,
  min_window_n = 10,
  dry_run = TRUE
)
windows_statistics
```

```
# Sliding Window K-NN imputation by specifying `k` (sliding windows)
imputed_knn <- slide_imp(
  beta_matrix,
  location = location,
  k = 5,
  window_size = 50,
  overlap_size = 10,
  min_window_n = 10,
  scale = FALSE # This argument belongs to PCA imputation and will be ignored
)
imputed_knn

# Sliding Window PCA imputation by specifying `ncp` (sliding windows)
pca_knn <- slide_imp(
  beta_matrix,
  location = location,
  ncp = 2,
  window_size = 50,
  overlap_size = 10,
  min_window_n = 10
)
pca_knn

# Sliding Window K-NN imputation with flanking windows (flank = TRUE)
# Only the columns listed in `subset` are imputed; each uses its own
# centered window of width `window_size`.
imputed_flank <- slide_imp(
  beta_matrix,
  location = location,
  k = 2,
  window_size = 30,
  flank = TRUE,
  subset = c(10, 30, 70),
  min_window_n = 5
)
imputed_flank
```

---

tune_imp *Tune Parameters for Imputation Methods*

Description

Tunes hyperparameters for imputation methods such as slide_imp(), knn_imp(), pca_imp(), or user-supplied custom functions by repeated cross-validation. For group_imp(), tune knn_imp() or pca_imp() on a single group.

Usage

```
tune_imp(
  obj,
  parameters = NULL,
  .f,
  na_loc = NULL,
  num_na = NULL,
  n_reps = 1,
  n_cols = NULL,
  n_rows = 2,
  rowmax = 0.9,
  colmax = 0.9,
  na_col_subset = NULL,
  max_attempts = 100,
  .progress = TRUE,
  cores = 1,
  location = NULL,
  pin_blas = FALSE
)
```

Arguments

obj	A numeric matrix with **samples in rows** and **features in columns**.
parameters	A data.frame specifying parameter combinations to tune, where each column represents a parameter accepted by .f (excluding obj). List columns are supported for complex parameters. Duplicate rows are automatically removed. NULL is treated as tuning the function with its default parameters.
.f	Either "knn_imp", "pca_imp", "slide_imp", or a custom function specifying the imputation method to tune.
na_loc	Optional. Pre-defined missing value locations to bypass random NA injection with sample_na_loc(). Accepted formats include: • A two-column integer matrix (row, column indices). • A numeric vector of linear locations. • A list where each element is one of the above formats (one per repetition).
num_na	Integer. Total number of missing values to inject per repetition. If supplied, n_cols is computed automatically and missing values are distributed as evenly as possible, using n_rows as the base size (num_na must be at least n_rows). If omitted but n_cols is supplied, the total injected is n_cols * n_rows. If num_na, n_cols, and na_loc are all NULL, tune_imp() defaults to roughly 5% of total cells, capped at 500. sample_na_loc() has no default. Ignored in tune_imp() when na_loc is supplied.
n_reps	Integer. Number of repetitions for random NA injection (default 1).
n_cols	Integer. The number of columns to receive injected NA per repetition. Ignored when num_na is supplied (in which case n_cols is derived as num_na %/% n_rows). Must be provided if num_na is NULL. Ignored in tune_imp() when na_loc is supplied.

`n_rows` Integer. The target number of `NA` values per column (default `2L`).

- When `num_na` is supplied: used as the base size. Most columns receive exactly `n_rows` missing values; `num_na %% n_rows` columns receive one extra. If there's only one column, it receives all the remainder.
- When `num_na` is `NULL`: every selected column receives exactly `n_rows` `NA`. Ignored in `tune_imp()` when `na_loc` is supplied.

`rowmax, colmax` Numbers between 0 and 1. NA injection cannot create rows/columns with a higher proportion of missing values than these thresholds.

`na_col_subset` Optional integer or character vector restricting which columns of `obj` are eligible for NA injection.

- If `NULL` (default): all columns are eligible.
- If character: values must exist in `colnames(obj)`.
- If integer/numeric: values must be valid 1-based column indices. The vector must be unique and must contain at least `n_cols` columns (or the number derived from `num_na`). Ignored in `tune_imp()` when `na_loc` is supplied.

`max_attempts` Integer. Maximum number of resampling attempts per repetition before giving up due to row-budget exhaustion (default `100`).

`.progress` Logical. Show a progress bar during tuning (default `TRUE`).

`cores` Controls the number of cores to parallelize over for K-NN and sliding-window K-NN imputation with OpenMP. For other methods, use `mirai::daemons()` instead.

`location` Required only for `slide_imp`. Numeric vector of column locations.

`pin_blas` Logical. Pin BLAS threads to 1 during parallel tuning (default `FALSE`).

Details

The function supports tuning for built-in methods (`"slide_imp"`, `"knn_imp"`, `"pca_imp"`) or custom functions provided via `.f`.

When `.f` is a character string, the columns in `parameters` are validated against the chosen method's requirements:

- `"knn_imp"`: requires `k` in `parameters`
- `"pca_imp"`: requires `ncp` in `parameters`
- `"slide_imp"`: requires `window_size`, `overlap_size`, and `min_window_n`, plus exactly one of `k` or `ncp`

When `.f` is a custom function, the columns in `parameters` must correspond to the arguments of `.f` (excluding the `obj` argument). The custom function must accept `obj` (a numeric matrix) as its first argument and return a numeric matrix of identical dimensions.

Tuning results can be evaluated using the `yardstick` package or `compute_metrics()`.

Value

A `data.frame` of class `slideimp_tune` containing:

- `...`: All columns originally provided in `parameters`.
- `param_set`: An integer ID for the unique parameter combination.
- `rep_id`: An integer indicating the repetition index.
- `result`: A nested list-column where each element is a data.frame containing `truth` (original values) and `estimate` (imputed values).
- `error`: A character column containing the error message if the iteration failed, otherwise `NA`.

Parallelization

- **K-NN**: use the `cores` argument (requires OpenMP). If `mirai::daemons()` are active, `cores` is automatically set to 1 to avoid nested parallelism.
- **PCA**: use `mirai::daemons()` instead of `cores`.

On macOS, OpenMP is typically unavailable and `cores` falls back to

1. Use `mirai::daemons()` for parallelization instead.

On Linux with OpenBLAS or MKL, set `pin_blas = TRUE` when running parallel PCA to prevent BLAS threads and `mirai` workers competing for cores.

Examples

```
# Setup example data. Increase `num_na` (500) and `n_reps` (10-30) in real
# analyses
obj <- sim_mat(10, 50)$input

# 1. Tune K-NN imputation with random NA injection
params_knn <- data.frame(k = c(2, 4))
results <- tune_imp(obj, params_knn, .f = "knn_imp", n_reps = 1, num_na = 10)
compute_metrics(results)

# 2. Tune with fixed NA positions
na_positions <- list(
  matrix(c(1, 2, 3, 1, 1, 1), ncol = 2),
  matrix(c(2, 3, 4, 2, 2, 2), ncol = 2)
)

results_fixed <- tune_imp(
  obj,
  data.frame(k = 2),
  .f = "knn_imp",
  na_loc = na_positions
)

# 3. Custom imputation function
custom_fill <- function(obj, val = 0) {
  obj[is.na(obj)] <- val
  obj
```

```
}
tune_imp(obj, data.frame(val = c(0, 1)), .f = custom_fill, num_na = 10)

# 4. Parallel tuning (requires mirai package)
mirai::daemons(2)
parameters_custom <- data.frame(mean = c(0, 1), sd = c(1, 1))

# Define a simple custom function for illustration
custom_imp <- function(obj, mean, sd) {
  na_pos <- is.na(obj)
  obj[na_pos] <- stats::rnorm(sum(na_pos), mean = mean, sd = sd)
  obj
}

results_p <- tune_imp(
  obj, parameters_custom, .f = custom_imp, n_reps = 1, num_na = 10
)
mirai::daemons(0) # Close workers
```

Index

col_vars, 2
compute_metrics, 3
compute_metrics(), *25*

group_imp, 4
group_imp(), *12*, *14*, *23*

knn_imp, 8
knn_imp(), *14*, *22, 23*

mean_imp_col, 10

pca_imp, 11
pca_imp(), *14*, *22, 23*
prep_groups, 12
prep_groups(), *5*, *7*, *13*
print.slideimp_results, 14
print.slideimp_sim, 15
print.slideimp_tbl, 15

register_group_resolver, 16

sample_na_loc, 17
sample_na_loc(), *17*, *24*
sim_mat, 18
sim_mat(), *15*
slide_imp, 20
slide_imp(), *14*, *23*

tune_imp, 23
tune_imp(), *3*, *17, 18*, *24, 25*