

Package ‘rMIDAS2’

September 3, 2026

Title Multiple Imputation with 'MIDAS2' Denoising Autoencoders

Version 0.2.0

Description Fits 'MIDAS' denoising autoencoder models for multiple imputation of missing data, generates multiply-imputed datasets, computes imputation means, and runs Rubin's rules regression analysis. Wraps the 'MIDAS2' 'Python' engine via a local 'FastAPI' server over 'HTTP', so no 'reticulate' dependency is needed at runtime. Methods are described in Lall and Robinson (2022) <[doi:10.1017/pan.2020.49](https://doi.org/10.1017/pan.2020.49)> and Lall and Robinson (2023) <[doi:10.18637/jss.v107.i09](https://doi.org/10.18637/jss.v107.i09)>.

License MIT + file LICENSE

URL <https://github.com/MIDASverse/MIDAS2>

BugReports <https://github.com/MIDASverse/MIDAS2/issues>

Depends R (>= 4.1.0)

Encoding UTF-8

SystemRequirements Python (>= 3.9) with the 'midasverse-midas-api' package

Imports curl, httr2 (>= 1.0.0), processx (>= 3.8.0), rlang (>= 1.1.0)

Suggests arrow, jsonlite, reticulate, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Thomas Robinson [aut, cre],
Ranjit Lall [aut]

Maintainer Thomas Robinson <t.robinson7@lse.ac.uk>

Repository CRAN

Date/Publication 2026-09-03 11:50:08 UTC

Contents

combine	2
ensure_server	3
imp_mean	4
install_backend	4
midas	5
midas_fit	6
midas_transform	8
overimpute	9
start_server	10
stop_server	11
uninstall_backend	11
update_backend	12
Index	13

combine	<i>Combine results using Rubin’s rules</i>
---------	--

Description

Runs a GLM across all stored imputations and combines the results using Rubin’s combination rules for multiple imputation inference.

Usage

```
combine(  
  model_id,  
  y,  
  ind_vars = NULL,  
  dof_adjust = TRUE,  
  incl_constant = TRUE,  
  ...  
)
```

Arguments

model_id	A character model ID, or a fitted model object (list with a \$model_id element) as returned by midas_fit() or midas() .
y	Character. Name of the outcome variable.
ind_vars	Character vector of independent variable names, or NULL for all non-outcome columns.
dof_adjust	Logical. Apply Barnard-Rubin degrees-of-freedom adjustment (default TRUE).
incl_constant	Logical. Include an intercept (default TRUE).
...	Arguments forwarded to ensure_server() .

Value

A data frame with columns term, estimate, std.error, statistic, df, and p.value.

Examples

```
## Not run:
df <- data.frame(Y = rnorm(200), X1 = rnorm(200), X2 = rnorm(200))
df$X1[sample(200, 40)] <- NA
fit <- midas_fit(df, epochs = 10L)
midas_transform(fit, m = 10)
results <- combine(fit, y = "Y")
results

## End(Not run)
```

ensure_server	<i>Ensure the server is running</i>
---------------	-------------------------------------

Description

Starts the server if it is not already running. Called internally by every client function so users never have to manage the server manually.

Usage

```
ensure_server(...)
```

Arguments

... Arguments forwarded to [start_server\(\)](#).

Value

Invisibly returns the base URL of the running server.

Examples

```
## Not run:
ensure_server()

## End(Not run)
```

imp_mean	<i>Compute mean imputation</i>
----------	--------------------------------

Description

Calculates the element-wise mean across all stored imputations for a model.

Usage

```
imp_mean(model_id, ...)
```

Arguments

model_id	A character model ID, or a fitted model object (list with a \$model_id element) as returned by <code>midas_fit()</code> or <code>midas()</code> .
...	Arguments forwarded to <code>ensure_server()</code> .

Value

A data frame with the mean imputed values.

Examples

```
## Not run:
df <- data.frame(X1 = rnorm(200), X2 = rnorm(200))
df$X1[sample(200, 40)] <- NA
fit <- midas_fit(df, epochs = 10L)
midas_transform(fit, m = 10)
mean_df <- imp_mean(fit)

## End(Not run)
```

install_backend	<i>Install the MIDAS2 Python backend</i>
-----------------	--

Description

Creates an isolated Python environment and installs the `midasverse-midas-api` package (which pulls in `midasverse-midas` as a dependency).

Usage

```
install_backend(
  method = c("pip", "conda", "uv"),
  envname = "midas2_env",
  package = "midasverse-midas-api"
)
```

Arguments

method	Character. One of "pip", "conda", or "uv".
envname	Character. Name of the virtual environment to create (default "midas2_env").
package	Character. Package specifier to install (default "midasverse-midas-api").

Details

This is the **only** function in the package that uses reticulate, and only for environment creation. It is never used at runtime.

Value

No return value, called for side effects.

Examples

```
## Not run:
install_backend()
install_backend(method = "conda")

## End(Not run)
```

midas	<i>Multiple imputation (all-in-one)</i>
-------	---

Description

Convenience function that fits a MIDAS model and generates imputations in a single call. Equivalent to calling `midas_fit()` followed by `midas_transform()`.

Usage

```
midas(
  data,
  m = 5L,
  hidden_layers = c(256L, 128L, 64L),
  dropout_prob = 0.5,
  epochs = 75L,
  batch_size = 64L,
  lr = 0.001,
  corrupt_rate = 0.8,
  num_adj = 1,
  cat_adj = 1,
  bin_adj = 1,
  pos_adj = 1,
  omit_first = FALSE,
  seed = 89L,
  ...
)
```

Arguments

data	A data frame (may contain NA for missing values).
m	Integer. Number of imputations (default 5).
hidden_layers	Integer vector of hidden layer sizes (default c(256, 128, 64)).
dropout_prob	Numeric. Dropout probability (default 0.5). See "Choosing dropout_prob" above.
epochs	Integer. Number of training epochs (default 75).
batch_size	Integer. Mini-batch size (default 64).
lr	Numeric. Learning rate (default 0.001).
corrupt_rate	Numeric. Corruption rate for denoising (default 0.8).
num_adj	Numeric. Loss multiplier for numeric columns (default 1).
cat_adj	Numeric. Loss multiplier for categorical columns (default 1).
bin_adj	Numeric. Loss multiplier for binary columns (default 1).
pos_adj	Numeric. Loss multiplier for positive columns (default 1).
omit_first	Logical. Omit first column from encoder input (default FALSE).
seed	Integer. Random seed (default 89).
...	Arguments forwarded to ensure_server() .

Value

A list with `model_id` and `imputations` (a list of data frames). Columns keep the classes of data, and cells that were observed in data are returned exactly (see the "Observed values" section of [midas_transform\(\)](#)).

Examples

```
## Not run:
df <- data.frame(X1 = rnorm(200), X2 = rnorm(200))
df$X1[sample(200, 40)] <- NA
result <- midas(df, m = 5, epochs = 10)
head(result$imputations[[1]])

## End(Not run)
```

midas_fit

Fit a MIDAS model

Description

Sends data to the server and fits a MIDAS denoising autoencoder.

Usage

```
midas_fit(
  data,
  hidden_layers = c(256L, 128L, 64L),
  dropout_prob = 0.5,
  epochs = 75L,
  batch_size = 64L,
  lr = 0.001,
  corrupt_rate = 0.8,
  num_adj = 1,
  cat_adj = 1,
  bin_adj = 1,
  pos_adj = 1,
  omit_first = FALSE,
  seed = 89L,
  ...
)
```

Arguments

<code>data</code>	A data frame (may contain NA for missing values).
<code>hidden_layers</code>	Integer vector of hidden layer sizes (default <code>c(256, 128, 64)</code>).
<code>dropout_prob</code>	Numeric. Dropout probability (default 0.5). See "Choosing <code>dropout_prob</code> " above.
<code>epochs</code>	Integer. Number of training epochs (default 75).
<code>batch_size</code>	Integer. Mini-batch size (default 64).
<code>lr</code>	Numeric. Learning rate (default 0.001).
<code>corrupt_rate</code>	Numeric. Corruption rate for denoising (default 0.8).
<code>num_adj</code>	Numeric. Loss multiplier for numeric columns (default 1).
<code>cat_adj</code>	Numeric. Loss multiplier for categorical columns (default 1).
<code>bin_adj</code>	Numeric. Loss multiplier for binary columns (default 1).
<code>pos_adj</code>	Numeric. Loss multiplier for positive columns (default 1).
<code>omit_first</code>	Logical. Omit first column from encoder input (default FALSE).
<code>seed</code>	Integer. Random seed (default 89).
<code>...</code>	Arguments forwarded to <code>ensure_server()</code> .

Value

A list with `model_id`, `n_rows`, `n_cols`, `col_types` and `col_meta` (the column types of data, used by `midas_transform()` to return imputations with the classes you supplied).

Column types

Numeric columns are sent as numbers. Factor, character and logical columns are categorical: they are sent via Parquet, which preserves their type so that the engine fits a softmax head over the levels of a multi-level factor and a binary head over a two-level one. Character and logical columns are converted to factors on the way out. The Parquet transport requires the arrow package; without it, data containing categorical columns raise an error rather than being silently mangled.

Choosing dropout_prob

The default of 0.5 suits wide data with many columns. On narrow frames – fewer than roughly 15 columns – it removes most of the signal available to reconstruct each cell, and a smaller value (around 0.2) can substantially improve reconstruction accuracy. `corrupt_rate` is generally best left at its default. If imputations on a narrow data set look poor, lower `dropout_prob` before changing anything else.

Examples

```
## Not run:
df <- data.frame(X1 = rnorm(200), X2 = rnorm(200), X3 = rnorm(200))
df$X2[sample(200, 40)] <- NA
fit <- midas_fit(df, epochs = 10L)
fit$model_id

# categorical columns are supported directly
df$grp <- factor(sample(c("a", "b", "c"), 200, TRUE))
df$grp[sample(200, 30)] <- NA
fit2 <- midas_fit(df, epochs = 10L)
fit2$col_types

## End(Not run)
```

midas_transform	<i>Generate multiple imputations</i>
-----------------	--------------------------------------

Description

Generates `m` imputed datasets from a fitted MIDAS model.

Usage

```
midas_transform(model_id, m = 5L, data = NULL, ...)
```

Arguments

<code>model_id</code>	A character model ID, or a fitted model object (list with a <code>\$model_id</code> element) as returned by <code>midas_fit()</code> or <code>midas()</code> .
<code>m</code>	Integer. Number of imputations (default 5).

data	Optional. The data frame passed to <code>midas_fit()</code> . When supplied, every cell that was observed in data is restored to its original value, so only missing cells differ from the input. Must have the same number of rows as the data the model was fitted to.
...	Arguments forwarded to <code>ensure_server()</code> .

Value

A list of m data frames, each with imputed values. Columns are returned with the classes supplied to `midas_fit()`, including factor levels and ordering.

Observed values

The imputation engine ingests data as float32, and returns observed cells from that downcast copy rather than re-estimating them. An observed numeric value therefore comes back as the binary32 round-trip of what you supplied – a relative change of roughly $6e-8$, which is invisible on standardised covariates but is a change of about $1e-4$ on data of order 1000. Pass the original frame as data to have observed cells restored exactly. `midas()` does this automatically, since it is given the data.

Examples

```
## Not run:
df <- data.frame(X1 = rnorm(200), X2 = rnorm(200))
df$X1[sample(200, 40)] <- NA
fit <- midas_fit(df, epochs = 10L)
imps <- midas_transform(fit, m = 10)
head(imps[[1]])

# restore observed cells exactly
imps <- midas_transform(fit, m = 10, data = df)
identical(imps[[1]]$X2[!is.na(df$X2)], df$X2[!is.na(df$X2)])

## End(Not run)
```

overimpute

Overimputation diagnostic

Description

Masks a fraction of observed values, re-imputes them, and computes RMSE to assess imputation quality.

Usage

```
overimpute(model_id, mask_frac = 0.1, m = 5L, seed = NULL, ...)
```

Arguments

model_id	A character model ID, or a fitted model object (list with a \$model_id element) as returned by <code>midas_fit()</code> or <code>midas()</code> .
mask_frac	Numeric. Fraction of observed values to mask (default 0.1).
m	Integer. Number of imputations for the diagnostic (default 5).
seed	Integer or NULL. Random seed.
...	Arguments forwarded to <code>ensure_server()</code> .

Value

A list with rmse (named numeric vector) and mean_rmse.

Examples

```
## Not run:
df <- data.frame(X1 = rnorm(200), X2 = rnorm(200))
df$X1[sample(200, 40)] <- NA
fit <- midas_fit(df, epochs = 10L)
diag <- overimpute(fit, mask_frac = 0.1)
diag$mean_rmse

## End(Not run)
```

start_server

Start the MIDAS2 API server

Description

Launches python -m midas2_api as a background process and waits for the /health endpoint to respond.

Usage

```
start_server(python = "python3", port = NULL, venv = NULL, max_wait = 120L)
```

Arguments

python	Path to the Python interpreter (default "python3").
port	Port to bind to. If NULL, a free port is chosen automatically.
venv	Path to a Python virtual environment. If supplied, the interpreter is taken from <venv>/bin/python (or <venv>/Scripts/python.exe on Windows).
max_wait	Maximum number of 0.5-second polling attempts (default 120, i.e. 60 seconds). The first launch may be slower due to Python import caching.

Value

Invisibly returns the port number.

Examples

```
## Not run:
start_server()
start_server(venv = "~/virtualenvs/midas2_env")

## End(Not run)
```

stop_server	<i>Stop the MIDAS2 API server</i>
-------------	-----------------------------------

Description

Kills the background Python process and clears the internal state.

Usage

```
stop_server()
```

Value

No return value, called for side effects.

Examples

```
## Not run:
stop_server()

## End(Not run)
```

uninstall_backend	<i>Uninstall the MIDAS2 Python backend</i>
-------------------	--

Description

Stops the running server (if any), removes the Python environment created by `install_backend()`, and clears the saved configuration.

Usage

```
uninstall_backend(method = c("pip", "conda", "uv"), envname = "midas2_env")
```

Arguments

- | | |
|---------|--|
| method | Character. One of "pip", "conda", or "uv". Must match the method used during installation. |
| envname | Character. Name of the virtual environment to remove (default "midas2_env"). |

Value

No return value, called for side effects.

Examples

```
## Not run:
uninstall_backend()
uninstall_backend(method = "conda")

## End(Not run)
```

update_backend	<i>Update the MIDAS2 Python backend</i>
----------------	---

Description

Upgrades the midasverse-midas-api package (and its dependencies) in the existing Python environment. Stops the running server first so that the new version is loaded on next use.

Usage

```
update_backend(
  method = c("pip", "conda", "uv"),
  envname = "midas2_env",
  package = "midasverse-midas-api"
)
```

Arguments

- method Character. One of "pip", "conda", or "uv". Must match the method used during installation.
- envname Character. Name of the virtual environment (default "midas2_env").
- package Character. Package specifier to upgrade (default "midasverse-midas-api").

Value

No return value, called for side effects.

Examples

```
## Not run:
update_backend()

## End(Not run)
```

Index

combine, [2](#)

ensure_server, [3](#)

ensure_server(), [2](#), [4](#), [6](#), [7](#), [9](#), [10](#)

imp_mean, [4](#)

install_backend, [4](#)

install_backend(), [11](#)

midas, [5](#)

midas(), [2](#), [4](#), [8–10](#)

midas_fit, [6](#)

midas_fit(), [2](#), [4](#), [5](#), [8–10](#)

midas_transform, [8](#)

midas_transform(), [5–7](#)

overimpute, [9](#)

start_server, [10](#)

start_server(), [3](#)

stop_server, [11](#)

uninstall_backend, [11](#)

update_backend, [12](#)