Package 'citestR'

May 8, 2026

Title Conditional Independence of Missingness Test

Version 0.1.1

Description Tests whether missingness in explanatory variables is conditionally independent of the outcome, given observed data. Uses multiply-imputed datasets and cross-validated classifiers to produce a test statistic and p-value, with a sensitivity parameter (kappa) for calibrating interpretation. Wraps the 'citest' 'Python' engine via a local 'FastAPI' server over 'HTTP', so no 'reticulate' dependency is needed at runtime.

License MIT + file LICENSE

URL https://github.com/midasverse/citest

BugReports https://github.com/midasverse/citest/issues

Depends R (>= 4.1.0)

Encoding UTF-8

RoxygenNote 7.3.3

SystemRequirements Python (>= 3.9) with the 'midasverse-citest-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

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-03-23 17:40:19 UTC

Contents

`calibration_pivot` *Generate a calibration pivot table*

Description

Rows are R-squared values, columns are gamma_x values, for a fixed beta_yx.

Usage

```
calibration_pivot(
  beta_yx = 0.3,
  r2_grid = NULL,
  beta_grid = NULL,
  gamma_grid = NULL,
  ...
)
```

Arguments

`beta_yx`	Numeric. Fixed beta_yx value (default 0.3).
`r2_grid`	Numeric vector, or `NULL`.
`beta_grid`	Numeric vector, or `NULL`.
`gamma_grid`	Numeric vector, or `NULL`.
`...`	Arguments forwarded to `ensure_server()`.

Value

A data frame (pivot table).

Examples

```
calibration_pivot(beta_yx = 0.3)
```

---

ci_test *Run the conditional independence test*

Description

All-in-one convenience function: creates a dataset on the server, builds a `CIMissTest`, runs it, and returns the results.

Usage

```
ci_test(
  data,
  y,
  expl_vars = NULL,
  onehot = TRUE,
  imputer = "midas",
  classifier = "rf",
  m = 10L,
  n_folds = 10L,
  classifier_args = list(),
  imputer_args = list(),
  random_state = 42L,
  target_level = "variable",
  variance_method = "mi_crossfit",
  subsample_cap = 2000L,
  ...
)
```

Arguments

`data`	A data frame (may contain `NA`).
`y`	Character. Name of the outcome variable.
`expl_vars`	Character vector of explanatory variable names, or `NULL`.
`onehot`	Logical. One-hot encode categoricals (default `TRUE`).
`imputer`	Character. Imputer backend: `"midas"` (default), `"iterative"`, `"iterative2"`, `"complete"`, or `"null"`.
`classifier`	Character. Classifier backend: `"rf"` (default), `"et"`, or `"logistic"`.

m	Integer. Number of multiply-imputed datasets (default 10).
n_folds	Integer. Number of cross-validation folds (default 10).
classifier_args	Named list of extra classifier arguments.
imputer_args	Named list of extra imputer arguments.
random_state	Integer. Random seed (default 42).
target_level	Character. "variable" or "column".
variance_method	Character. "mi_crossfit" or "legacy_fold".
subsample_cap	Integer or NULL. Maximum rows to subsample.
...	Arguments forwarded to ensure_server().

Value

A list with elements model_id, dataset_id, and results. The results element contains m, B, W_bar, T, t_k, p_k, p_2s, and optionally df.

Examples

```
df <- data.frame(Y = rnorm(200), X1 = rnorm(200), X2 = rnorm(200))
df$X1[sample(200, 20)] <- NA
result <- ci_test(df, y = "Y")
result$results$p_2s
```

---

compute_kappa *Compute theoretical imputation bias kappa*

Description

Compute theoretical imputation bias kappa

Usage

```
compute_kappa(r2_x_z, beta_yx, gamma_x, ...)
```

Arguments

r2_x_z	Numeric. R-squared of X on observed covariates Z.
beta_yx	Numeric. Coefficient of X in the Y equation.
gamma_x	Numeric. Loading of X in the missingness equation.
...	Arguments forwarded to ensure_server().

Value

A single numeric value (kappa).

Examples

```
compute_kappa(r2_x_z = 0.5, beta_yx = 0.3, gamma_x = 0.2)
```

---

ensure_server *Ensure the server is running*

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

```
ensure_server()
```

---

get_summary *Get a summary of test results*

Description

Retrieves a structured summary for a previously fitted model.

Usage

```
get_summary(model_id, ...)
```

Arguments

`model_id`	Character. UUID returned by `ci_test()`.
`...`	Arguments forwarded to `ensure_server()`.

Value

A list with elements `outcome`, `imputer`, `classifier`, `variance_method`, `mean_difference`, `t_statistic`, `df`, `p_value`, and `p_value_two_sided`.

Examples

```
result <- ci_test(df, y = "Y")
get_summary(result$model_id)
```

---

has_server *Check whether the citest server is running*

Description

Returns `TRUE` if the package's background server process is alive. Used as the guard for `@examplesIf` so that examples requiring the Python backend are skipped when no server is available.

Usage

```
has_server()
```

Value

Logical.

---

imputer_r2 *Estimate imputer out-of-sample R-squared*

Description

Runs a mask-and-impute diagnostic on the server.

Usage

```
imputer_r2(model_id, mask_frac = 0.2, m_eval = 1L, ...)
```

Arguments

`model_id`	Character. UUID returned by `ci_test()`.
`mask_frac`	Numeric. Fraction of observed cells to hold out (default 0.2).
`m_eval`	Integer. Number of imputations to average over (default 1).
`...`	Arguments forwarded to `ensure_server()`.

Value

A list with `mean_r2` and `per_variable` (named numeric vector).

Examples

```
result <- ci_test(df, y = "Y")
imputer_r2(result$model_id)
```

`install_backend` *Install the citest Python backend*

Description

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

Usage

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

Arguments

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

```
install_backend()
install_backend(method = "conda")
```

kappa_calibration_table *Generate a kappa calibration table*

Description

Generate a kappa calibration table

Usage

```
kappa_calibration_table(
  r2_grid = NULL,
  beta_grid = NULL,
  gamma_grid = NULL,
  ...
)
```

Arguments

r2_grid	Numeric vector of R-squared values, or NULL for defaults.
beta_grid	Numeric vector of beta values, or NULL for defaults.
gamma_grid	Numeric vector of gamma values, or NULL for defaults.
...	Arguments forwarded to ensure_server().

Value

A data frame with columns r2_x_z, beta_yx, gamma_x, kappa, abs_kappa.

Examples

```
kappa_calibration_table(r2_grid = c(0.3, 0.5, 0.7))
```

make_dataset *Create a dataset on the server*

Description

Sends a data frame to the citest API server and creates a Dataset object.

Usage

```
make_dataset(data, y, expl_vars = NULL, onehot = TRUE, ...)
```

Arguments

data	A data frame (may contain NA for missing values).
y	Character. Name of the outcome variable.
expl_vars	Character vector of explanatory variable names, or NULL for all non-outcome columns.
onehot	Logical. One-hot encode categorical columns (default TRUE).
...	Arguments forwarded to ensure_server().

Value

A list with elements dataset_id, n, columns, y_name, expl_vars, and pct_missing.

Examples

```
df <- data.frame(Y = rnorm(100), X1 = rnorm(100))
ds <- make_dataset(df, y = "Y")
ds$dataset_id
```

---

make_dataset_parquet *Create a dataset from a Parquet file*

Description

Uploads a Parquet file to the citest API server.

Usage

```
make_dataset_parquet(file, y, expl_vars = NULL, onehot = TRUE, ...)
```

Arguments

file	Path to a .parquet file.
y	Character. Name of the outcome variable.
expl_vars	Character vector of explanatory variable names, or NULL.
onehot	Logical. One-hot encode categorical columns (default TRUE).
...	Arguments forwarded to ensure_server().

Value

A list with elements dataset_id, n, columns, y_name, expl_vars, and pct_missing.

Examples

```
ds <- make_dataset_parquet("data.parquet", y = "Y")
```

print.citest_result *Print a citest result*

Description

Displays a concise summary of the conditional independence test result, including the test statistic, degrees of freedom, p-value, and a plain language interpretation.

Usage

```
## S3 method for class 'citest_result'
print(x, ...)
```

Arguments

x	A citest_result object returned by ci_test().
...	Additional arguments (currently ignored).

Value

Invisibly returns x.

Examples

```
result <- structure(list(
  model_id = "example-id",
  dataset_id = "example-ds",
  results = list(m = 0.12, t_k = 2.5, df = 9, p_2s = 0.034)
), class = "citest_result")
print(result)
```

print.citest_summary *Print a citest summary*

Description

Displays a formatted summary of a fitted conditional independence test, including model configuration and key results.

Usage

```
## S3 method for class 'citest_summary'
print(x, ...)
```

Arguments

x	A citest_summary object returned by get_summary().
...	Additional arguments (currently ignored).

Value

Invisibly returns x.

Examples

```
smry <- structure(list(
  outcome = "Y",
  imputer = "midas",
  classifier = "rf",
  variance_method = "mi_crossfit",
  mean_difference = 0.12,
  t_statistic = 2.5,
  df = 9,
  p_value_two_sided = 0.034
), class = "citest_summary")
print(smry)
```

---

simulate_data *Generate a simulated dataset*

Description

Calls one of the built-in data-generating processes on the Python server.

Usage

```
simulate_data(
  dgp,
  n = 1000L,
  ci = TRUE,
  missing_mech = "linear",
  beta_y = NULL,
  mcar_prop = NULL,
  k = NULL,
  ...
)
```

Arguments

dgp	Character. Name of the DGP (e.g. "single_mar", "adult").
n	Integer. Number of observations.
ci	Logical. Conditional independence holds (TRUE) or not.
missing_mech	Character. Missingness mechanism ("linear" or "xor").
beta_y	Numeric or NULL. Outcome effect size (for DGPs that use it).
mcar_prop	Numeric or NULL. Proportion of MCAR missingness.
k	Integer or NULL. Number of columns (for the adult DGP).
...	Arguments forwarded to ensure_server().

Value

A list with dataset_id, n, columns, pct_missing.

Examples

```
sim <- simulate_data("single_mar", n = 500, ci = TRUE)
```

---

start_server *Start the citest API server*

Description

Launches python -m citest_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

```
start_server()
start_server(venv = "~/.virtualenvs/citest_env")
```

`stop_server` *Stop the citest API server*

Description

Kills the background Python process and clears the internal state.

Usage

```
stop_server()
```

Value

No return value, called for side effects.

Examples

```
stop_server()
```

`uninstall_backend` *Uninstall the citest Python backend*

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 = "citest_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 `"citest_env"`).

Value

No return value, called for side effects.

Examples

```
uninstall_backend()
uninstall_backend(method = "conda")
```

`update_backend` *Update the citest Python backend*

Description

Upgrades the `midasverse-citest-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 = "citest_env",
  package = "midasverse-citest-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 `"citest_env"`).
`package`	Character. Package specifier to upgrade (default `"midasverse-citest-api"`).

Value

No return value, called for side effects.

Examples

```
update_backend()
```

Index