Package: smriti (via r-universe)

October 4, 2026

Title Automated Routing Engine for Longitudinal Missing Data

Version 0.2.0

Description Provides an automated routing engine for longitudinal missing data. It utilizes a Lagrange-constrained Random Forest based on sample size, missingness rate, and skew to preserve structural variance.

License MIT + file LICENSE

SystemRequirements C++17

Encoding UTF-8

VignetteBuilder knitr

Imports Rcpp (>= 1.0.0)

Suggests missForest, missRanger, ranger, mice, lavaan, ggplot2, tidyr, dplyr, knitr, rmarkdown, MASS

LinkingTo Rcpp, RcppArmadillo

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Xiyuan Guo [aut, cre]

Maintainer Xiyuan Guo <tommyguo039@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-06-05 21:00:02 UTC

RemoteUrl https://github.com/cran/smriti

RemoteRef HEAD

RemoteSha ed473ff47fe930017cb4089a64a2c5bce8d4eb47

Contents

smriti_fiml 2
smriti_forest 2
smriti_impute 3
smriti_mi 4
smriti_mice 6
smriti_ranger 6

Index **8**

`smriti_fiml` *FIML-Smriti Refinement Wrapper*

Description

This function serves as an agnostic refinement layer, combining Full Information Maximum Likelihood (FIML) with the structural covariance preservation of `smriti_impute()`.

Usage

```
smriti_fiml(data, model, initial_imputation = NULL, lambda = 1)
```

Arguments

`data`	A data frame containing missing values.
`model`	A lavaan model syntax string.
`initial_imputation`	A data frame or matrix of the same dimensions as the subset of `data` defined by the model, containing initial imputed values. If `NULL` (default), `smriti_impute` handles initial imputation.
`lambda`	A numeric value specifying the per-observation penalty weight for covariance matching. Defaults to 1.0.

Value

A data frame with FIML-consistent, covariance-projected imputed values.

`smriti_forest` *missForest-Smriti Refinement Wrapper*

Description

This function serves as an agnostic refinement layer, combining the predictive power of Random Forests via `missForest` with the structural covariance preservation of `smriti_impute()`.

Usage

```
smriti_forest(data, time_cols, robust = TRUE, ...)
```

Arguments

`data`	A data frame containing missing values.
`time_cols`	A character vector or numeric vector specifying the longitudinal columns.
`robust`	A logical value. If TRUE (default), uses robust covariance estimation.
`...`	Additional arguments passed directly to `missForest::missForest()`.

Value

A data frame with covariance-projected imputed values.

Examples

```
## Not run:
df <- data.frame(T1 = c(1, NA, 3, 4), T2 = c(NA, 2, 3, 4))
smriti_forest(df, time_cols = 1:2)

## End(Not run)
```

`smriti_impute` *Smriti Automated Longitudinal Imputation*

Description

This function performs an automated routing and refinement for longitudinal missing data. It establishes a target covariance manifold from observed data, and then projects an initial imputation back toward the structural manifold using a Lagrangian constraint. Only originally-missing values are updated; observed data is held fixed.

Usage

```
smriti_impute(
  data,
  time_cols,
  initial_imputation = NULL,
  lambda = 1,
  learning_rate = 0.001,
  tol = 1e-06,
  max_iter = 2000,
  robust = FALSE,
  custom_target = NULL
)
```

Arguments

`data`	A data frame containing missing values.
`time_cols`	A character vector or numeric vector specifying the longitudinal columns.
`initial_imputation`	A matrix or data frame of the same dimensions as `data[, time_cols]`, containing initial imputed values. If `NULL`, a simple column-mean imputation is performed as a fallback.
`lambda`	A numeric value or "auto" specifying the per-observation penalty weight on the covariance-constraint term. If "auto", a simple heuristic is used based on the Frobenius norm between initial and target covariance. The covariance gradient is deliberately un-normalised (no division by n-1) so the constraint remains

effective at any sample size. Defaults to 1.0 (tuned via simulation for general use). Use lower values (e.g. 0.01) for clean Normal data; increase for stronger constraint enforcement.

`learning_rate` A numeric value for the gradient descent step size. Defaults to 0.001.

`tol` A numeric value for the internal convergence tolerance (Frobenius norm). Defaults to 1e-6.

`max_iter` An integer specifying the maximum number of iterations for the gradient descent projection. Defaults to 2000.

`robust` A logical value. Setting it to TRUE uses a robust target constructed from pairwise Spearman correlations and column-wise MAD, projected to the nearest positive-semidefinite matrix. This protects against outliers and heavy-tailed noise. Defaults to FALSE (pairwise Pearson covariance). Use TRUE when the data contains explicit outlier contamination (e.g. sensor artifacts).

`custom_target` An optional p x p covariance matrix (where p is the number of longitudinal columns) to be used as the structural manifold. If provided, `robust` and `sigma_target` calculations are bypassed. This allows users to supply MAR-valid covariance targets from FIML or EM algorithms.

Value

A data frame with imputed and structurally refined values. Only the originally-missing cells are modified; observed values are returned unchanged.

Examples

```
# Simulated longitudinal data with scattered missingness
df <- data.frame(
  T1 = c(1.2, NA, 2.8, 3.1),
  T2 = c(2.1, 2.5, NA, 4.0),
  T3 = c(3.0, 3.3, 4.1, NA)
)
smriti_impute(df, time_cols = 1:3, robust = FALSE)
```

`smriti_mi` *Smriti Multiple Imputation Wrapper*

Description

This function generates multiple imputed datasets to capture the uncertainty inherent in missing data imputation. It uses bootstrapping to create distinct datasets, applies the Smriti Lagrangian refinement to each, and returns a list of completed datasets suitable for pooling.

Usage

```
smriti_mi(data, time_cols, m = 5, initial_imputation = NULL, ...)
```

Arguments

`data`	A data frame containing missing values.
`time_cols`	A character vector or numeric vector specifying the longitudinal columns.
`m`	An integer specifying the number of imputations. Defaults to 5.
`initial_imputation`	A matrix or data frame of the same dimensions as `data[, time_cols]`, containing initial imputed values. If `NULL`, the `smriti_impute` function's internal fallback (column-mean) is used.
`...`	Additional arguments passed to `smriti_impute()`, e.g. `lambda`, `learning_rate`, `max_iter`, `robust`.

Details

Approximate (bootstrap) multiple imputation. Proper multiple imputation (Rubin 1987) draws imputations from the Bayesian posterior predictive distribution of the missing data given the observed data and the imputation model. This function instead bootstraps the rows and then deterministically applies `smriti_impute()` to each bootstrap sample. The between-imputation variance therefore captures sampling variability but *not* the full imputation-model uncertainty. The resulting standard errors may be moderately anti-conservative. Users needing valid Rubin's-rules pooling should treat the output as approximate MI and consider adding a stochastic residual-draw step, or use a fully Bayesian engine such as `mice`.

Pooling the returned list can be performed manually via Rubin's rules:

```
mi_list <- smriti_mi(df, time_cols = 2:5, m = 10)
# Within-imputation estimates
estimates <- lapply(mi_list, function(d) coef(lm(y ~ x, data = d)))
# Pool with Rubin's rules (point estimate = mean, SE = sqrt(W + (1+1/m)*B))
```

Value

A list of `m` completed data frames, each with an `"imputation"` attribute giving its index (1..m). The list has class `"smriti_mi_list"`.

See Also

`smriti_impute()` for the single-imputation engine.

Examples

```
## Not run:
df <- data.frame(
  T1 = c(1, 2, NA, 4),
  T2 = c(NA, 2, 3, 4),
  T3 = c(1, NA, 3, 4)
)
mi_list <- smriti_mi(df, time_cols = c("T1", "T2", "T3"), m = 5)

## End(Not run)
```

smriti_mice *mice-Smriti Refinement Wrapper*

Description

This function serves as an agnostic refinement layer, combining multivariate imputation by chained equations (MICE) with the structural covariance preservation of `smriti_impute()`.

Usage

```
smriti_mice(data, time_cols, robust = TRUE, ...)
```

Arguments

`data`	A data frame containing missing values.
`time_cols`	A character vector or numeric vector specifying the longitudinal columns.
`robust`	A logical value. If TRUE (default), uses robust covariance estimation.
`...`	Additional arguments passed directly to `mice::mice()`.

Value

A data frame with covariance-projected imputed values.

Examples

```
## Not run:
df <- data.frame(T1 = c(1, NA, 3, 4), T2 = c(NA, 2, 3, 4))
smriti_mice(df, time_cols = 1:2)

## End(Not run)
```

smriti_ranger *missRanger-Smriti Refinement Wrapper*

Description

This function serves as an agnostic refinement layer, combining the fast predictive imputation of `missRanger` with the structural covariance preservation of `smriti_impute()`.

Usage

```
smriti_ranger(data, time_cols, robust = TRUE, ...)
```

Arguments

`data`	A data frame containing missing values.
`time_cols`	A character vector or numeric vector specifying the longitudinal columns.
`robust`	A logical value. If TRUE (default), uses robust covariance estimation.
`...`	Additional arguments passed directly to `missRanger::missRanger()`.

Value

A data frame with covariance-projected imputed values.

Examples

```
## Not run:
df <- data.frame(T1 = c(1, NA, 3, 4), T2 = c(NA, 2, 3, 4))
smriti_ranger(df, time_cols = 1:2)

## End(Not run)
```

Index

mice::mice(), *6*
missForest::missForest(), *2*
missRanger::missRanger(), *7*

smriti_fiml, 2
smriti_forest, 2
smriti_impute, 3
smriti_impute(), *2*, *5*, *6*
smriti_mi, 4
smriti_mice, 6
smriti_ranger, 6