imputeTestbench GSOC Update: Multivariate Imputation, Performance & Metrics

Project notes & demo

Contents

1 Overview

This vignette documents the GSOC updates implemented for the *imputeTestbench* evolution, focusing on:

- **Multivariate support** via `impute_errors_multi()` with shared missingness masks across variables.
- **Improved core**: removal of `eval(parse())`, pre-resolved function objects, and **parallelization hooks** (`future`/`foreach`).

- **Input class support**: accepts `ts`, `zoo/xts`, and routes multivariate `ts/data.frame/matrix` (and `tsibble`) automatically.
- **Method plugin system**: registry to **register/list/get** imputation methods with capabilities (univariate vs multivariate).
- **Built-in methods** (R + optional Python via `reticulate`) and graceful skipping of unavailable methods.
- **Advanced metrics**: structure-preserving comparisons (`corr_preservation`, `cov_frobenius`, `struct_preservation_index`) and an optional `downstream_forecast_rmse`.
- **Plotting**: updated `plot_errors()` that reads detail from `attr(errprof, "errall")`, with optional faceting by variable.

2 Quick start

If you are running this vignette outside the package (e.g., in a folder of `.R` files), ensure these files are present in the working directory and source them below.

```
library(ggplot2)
#> Warning: package 'ggplot2' was built under R version 4.4.3
# Source local dev files if functions are not already available
need <- function(f) !exists(f, mode = "function")

if (need("sample_dat"))            source("sample_dat.R")
if (need("impute_errors"))         source("impute_errors.R")
if (need("impute_errors_multi"))   source("impute_errors_multi.R")
if (need("plot_errors"))           source("plot_errors.R")
if (need("register_impute_method")) source("methods_registry.R")
if (need("register_builtin_methods")) source("methods_builtin.R")
if (!exists("corr_preservation"))  source("error_functions.R")  # advanced metrics
if (!exists("globalVariables", mode = "function")) {
  # harmless in rendering; only needed for R CMD check in a package
  if (file.exists("globalVariables.R")) source("globalVariables.R")
}

# Register built-ins (idempotent)
if (exists("register_builtin_methods")) register_builtin_methods()
```

2.0.1 Optional dependencies

Some built-in methods require additional packages. Install as needed:

```
install.packages(c("zoo","imputeTS","VIM","missForest","mice","future.apply","future","foreach","doPara
# For Python KNN: reticulate + scikit-learn (one-time)
# install.packages("reticulate")
# reticulate::conda_create("r-sklearn", packages = c("python=3.10"))
# reticulate::conda_install("r-sklearn", packages = c("scikit-learn","numpy"), pip = TRUE)
# reticulate::use_condaenv("r-sklearn", required = TRUE)
```

3 API highlights

3.1 `impute_errors()` — univariate benchmark

- Accepts numeric vectors and `ts` objects; auto-routes multivariate inputs to `impute_errors_multi()`.
- Key args: `methods`, `error`, `missPercent`, `repetition`, `simtype` = `"mcar"`/`"mar"`, `parallel`, `workers`.
- Returns a data.frame with class `errprof`. Attributes:
  - `attr(x, "errall")`: detailed rows (`Method`, `Percent`, `Repetition`, `Error` [, `Variable`]).
  - `attr(x, "metric")`, `attr(x, "simtype")`, `attr(x, "repetition")`.

3.2 `impute_errors_multi()` — multivariate benchmark

- Accepts `matrix`, `data.frame` (numeric columns), multivariate `ts`, and `tsibble`.
- Uses **shared masks** across variables per repetition/percent.
- Supports **multivariate-capable methods** (called **once** per repetition) and univariate methods (applied **per column**).
- Structural metrics are computed on the **entire imputed matrix** per repetition and stored with `Variable = "STRUCT"`.

3.3 Method registry

- `register_impute_method(name, fun, mode = "univariate"|"multivariate", pkg = NULL, notes = NULL)`
- `list_impute_methods()`
- `get_impute_method(name)`
- Built-ins are registered by `register_builtin_methods()`.

3.4 Plotting

- `plot_errors(errprof, plotType = "boxplot" | "bar" | "line", facet_by_variable = FALSE)`
- Reads `attr(errprof, "errall")` and adapts automatically.
- Prints a note if some methods were **skipped** (missing deps).

4 Examples

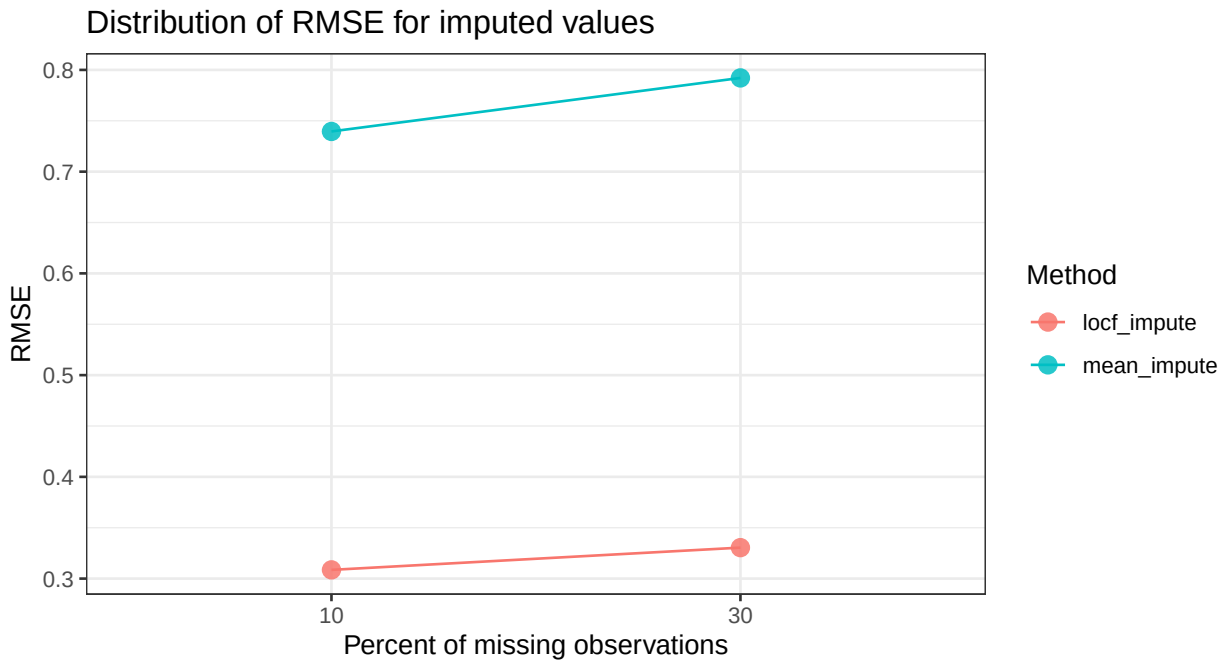
4.1 1. Univariate benchmark (toy series)

```r
# simple imputation methods
mean_impute <- function(x, ...) { x[is.na(x)] <- mean(x, na.rm = TRUE); x }
locf_impute <- function(x, ...) {
  if (!requireNamespace("zoo", quietly = TRUE)) stop("Please install 'zoo' for locf_impute")
  zoo::na.locf(x, na.rm = FALSE)
}
# metric
rmse <- function(truth, estimate) sqrt(mean((truth - estimate)^2, na.rm = TRUE))

set.seed(123)
```

```r
ts_u <- sin(seq(0, 8*pi, length.out = 500)) + stats::rnorm(500, sd = 0.1)

res_u <- impute_errors(
  dataIn      = ts_u,
  methods     = c("mean_impute", "locf_impute"),
  error       = "rmse",
  missPercent = c(10, 30),
  repetition  = 3,
  simtype     = "mar",
  blck        = 10
)
res_u
#>        Method Percent      rmse
#> 1 locf_impute      10 0.3085799
#> 2 mean_impute      10 0.7395062
#> 3 locf_impute      30 0.3304486
#> 4 mean_impute      30 0.7921206
head(attr(res_u, "errall"))
#>        Method Percent Repetition     Error
#> 1 mean_impute      10          1 0.7036809
#> 2 mean_impute      10          2 0.8760853
#> 3 mean_impute      10          3 0.6387523
#> 4 locf_impute      10          1 0.3615065
#> 5 locf_impute      10          2 0.2526623
#> 6 locf_impute      10          3 0.3115710
```

4.1.1 Plot (averages)

```r
plot_errors(res_u, plotType = "line")
```



4.2 2. Multivariate benchmark (two correlated series)

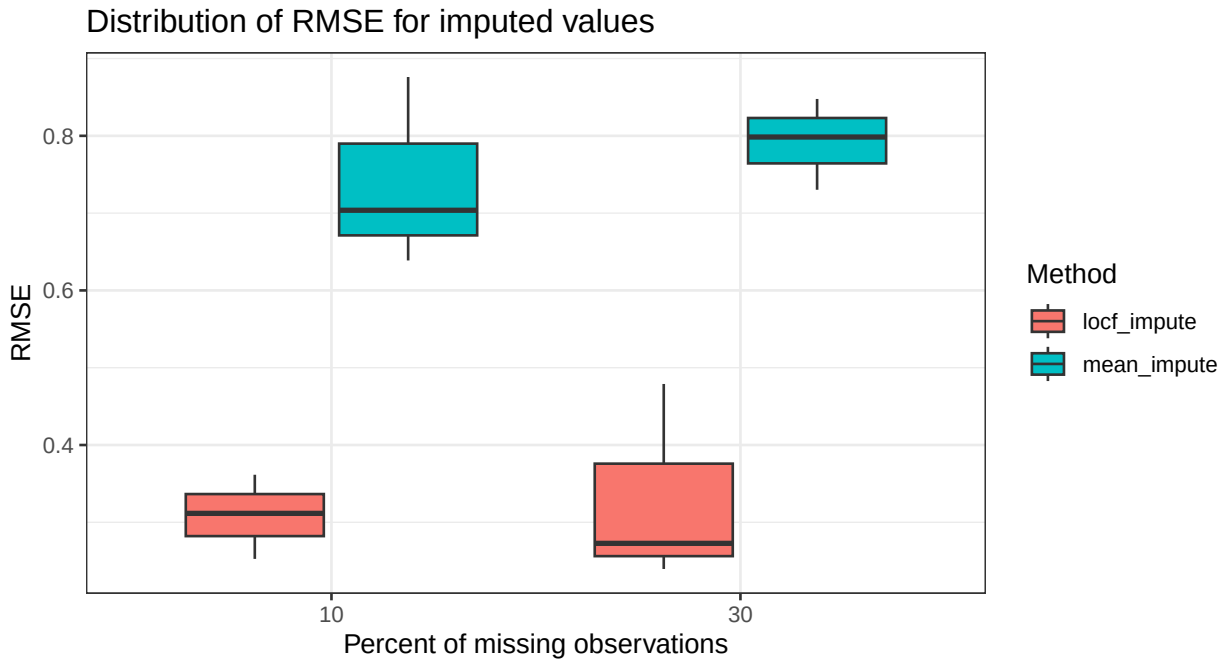
```
ts_v1 <- ts_u
ts_v2 <- ts_u + stats::rnorm(500, sd = 0.05)
X <- cbind(A = ts_v1, B = ts_v2)

res_m <- impute_errors_multi(
  dataIn      = X,
  methods     = c("mean_impute", "locf_impute"),
  error       = "rmse",
  missPercent = c(10, 30),
  repetition  = 3,
  simtype     = "mcar"
)
res_m
#>        Method Percent      rmse
#> 1 locf_impute      10 0.1567294
#> 2 mean_impute      10 0.7199516
#> 3 locf_impute      30 0.1658192
#> 4 mean_impute      30 0.7170948
head(attr(res_m, "by_variable"))
#>        Method Percent Variable     Error
#> 1 locf_impute      10        A 0.1518660
#> 2 mean_impute      10        A 0.7199118
#> 3 locf_impute      30        A 0.1590589
#> 4 mean_impute      30        A 0.7174939
#> 5 locf_impute      10        B 0.1615929
#> 6 mean_impute      10        B 0.7199914
```

4.2.1 Faceted boxplot

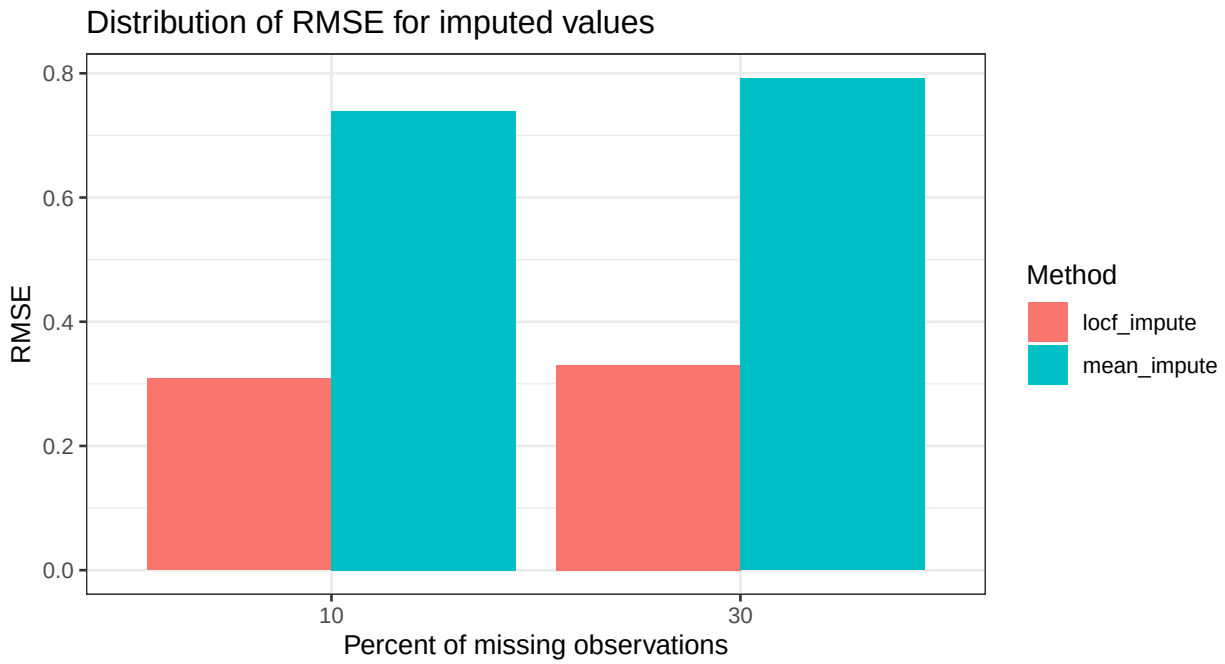
```
library(ggplot2)
errall_m <- attr(res_m, "errall")
ggplot(errall_m, aes(x = factor(Percent), y = Error, fill = Method)) +
  geom_boxplot() +
  facet_wrap(~ Variable, scales = "free_y") +
  labs(x = "% Missing", y = "RMSE", title = "Multivariate imputation benchmark")
```

4.3 3. Built-in plotter

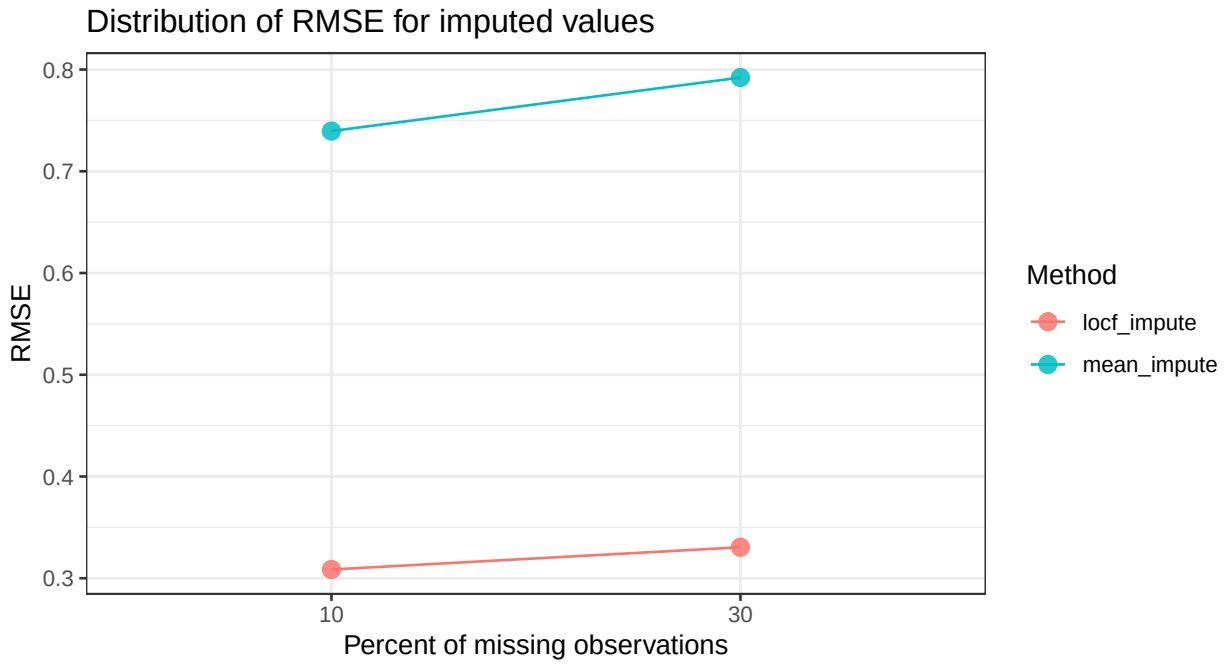
```
plot_errors(res_u, plotType = "boxplot")
```



```
plot_errors(res_u, plotType = "bar")
```




```
plot_errors(res_u, plotType = "line")
```



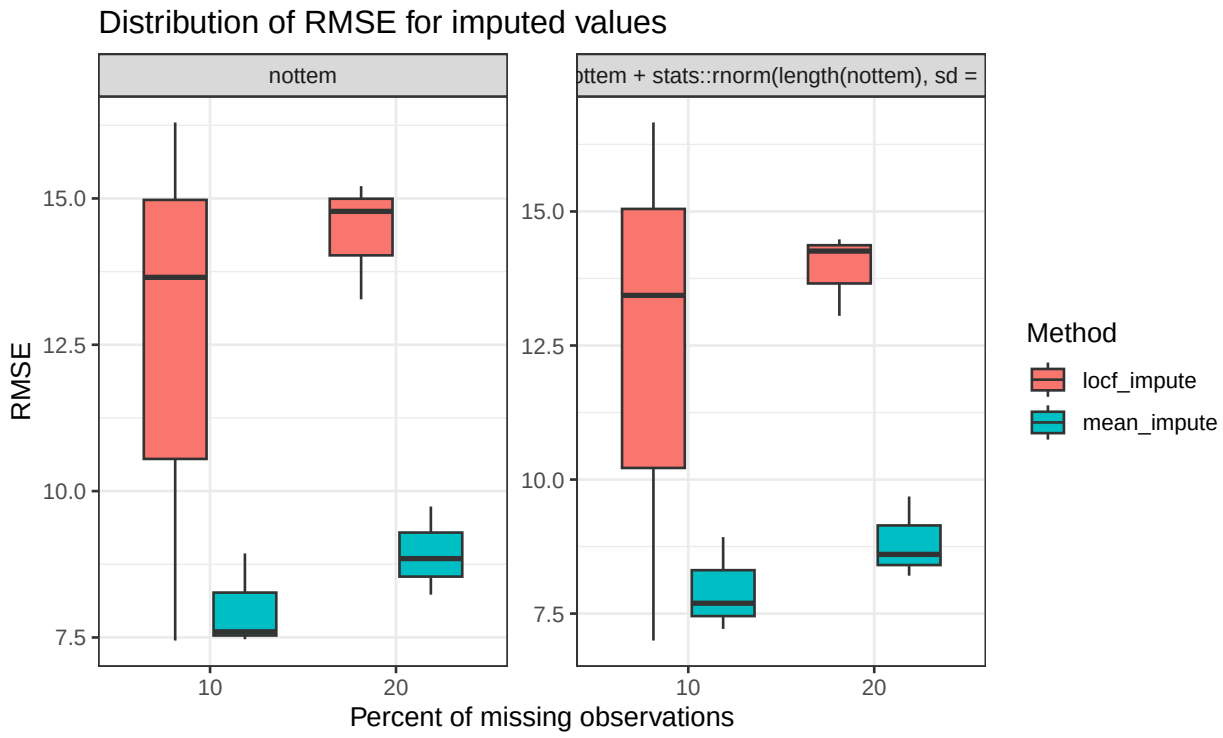
4.4 4. Parallelization

```r
# install.packages(c("future.apply","future","doParallel","foreach"))
res_u_par <- impute_errors(
  dataIn      = ts_u,
  methods     = c("mean_impute", "locf_impute"),
  error       = "rmse",
  missPercent = c(10, 30),
  repetition  = 8,
  simtype     = "mcar",
  parallel    = "future",   # or "foreach"
  workers     = 4
)
print(res_u_par)
```

4.5 5. Accepting ts inputs (univariate & multivariate)

```r
# Univariate ts
res_nottem <- impute_errors(
  dataIn      = nottem,   # ts object accepted
  methods     = c("mean_impute"),
  error       = "rmse",
  missPercent = c(10, 20),
  repetition  = 3,
  simtype     = "mcar"
)
res_nottem
#>        Method Percent     rmse
#> 1 mean_impute      10 8.128926
#> 2 mean_impute      20 8.364593
head(attr(res_nottem, "errall"))
#>        Method Percent Repetition    Error
#> 1 mean_impute      10          1 9.448221
#> 2 mean_impute      10          2 7.938028
#> 3 mean_impute      10          3 7.000529
#> 4 mean_impute      20          1 8.430376
#> 5 mean_impute      20          2 8.315451
#> 6 mean_impute      20          3 8.347952

# Multivariate ts: auto-routed to impute_errors_multi()
m   <- cbind(nottem, nottem + stats::rnorm(length(nottem), sd = 1))
mts <- ts(m, start = start(nottem), frequency = stats::frequency(nottem))

res_multi <- impute_errors(
  dataIn      = mts,
  methods     = c("mean_impute", "locf_impute"),
  error       = "rmse",
  missPercent = c(10, 20),
  repetition  = 3,
  simtype     = "mar",
  blck        = 6
)
res_multi
```

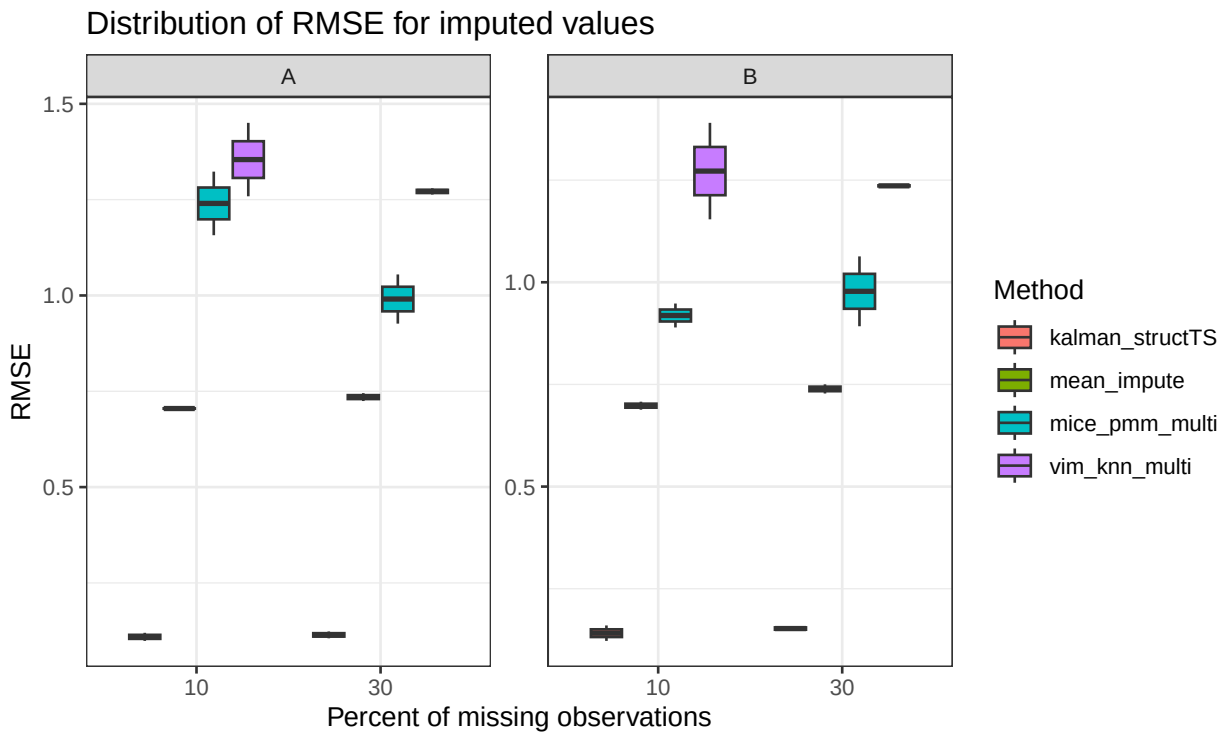
```
#>        Method Percent      rmse
#> 1 locf_impute      10 12.414916
#> 2 mean_impute      10  7.972563
#> 3 locf_impute      20 14.176790
#> 4 mean_impute      20  8.885026
head(attr(res_multi, "by_variable"))
#>        Method Percent                                          Variable     Error
#> 1 locf_impute      10                                            nottem 12.465733
#> 2 mean_impute      10                                            nottem  8.000626
#> 3 locf_impute      20                                            nottem 14.421976
#> 4 mean_impute      20                                            nottem  8.938386
#> 5 locf_impute      10 nottem + stats::rnorm(length(nottem), sd = 1) 12.364100
#> 6 mean_impute      10 nottem + stats::rnorm(length(nottem), sd = 1)  7.944499
plot_errors(res_multi, plotType = "boxplot", facet_by_variable = TRUE)
```



4.6 6. Method registry & mixed uni/multi methods

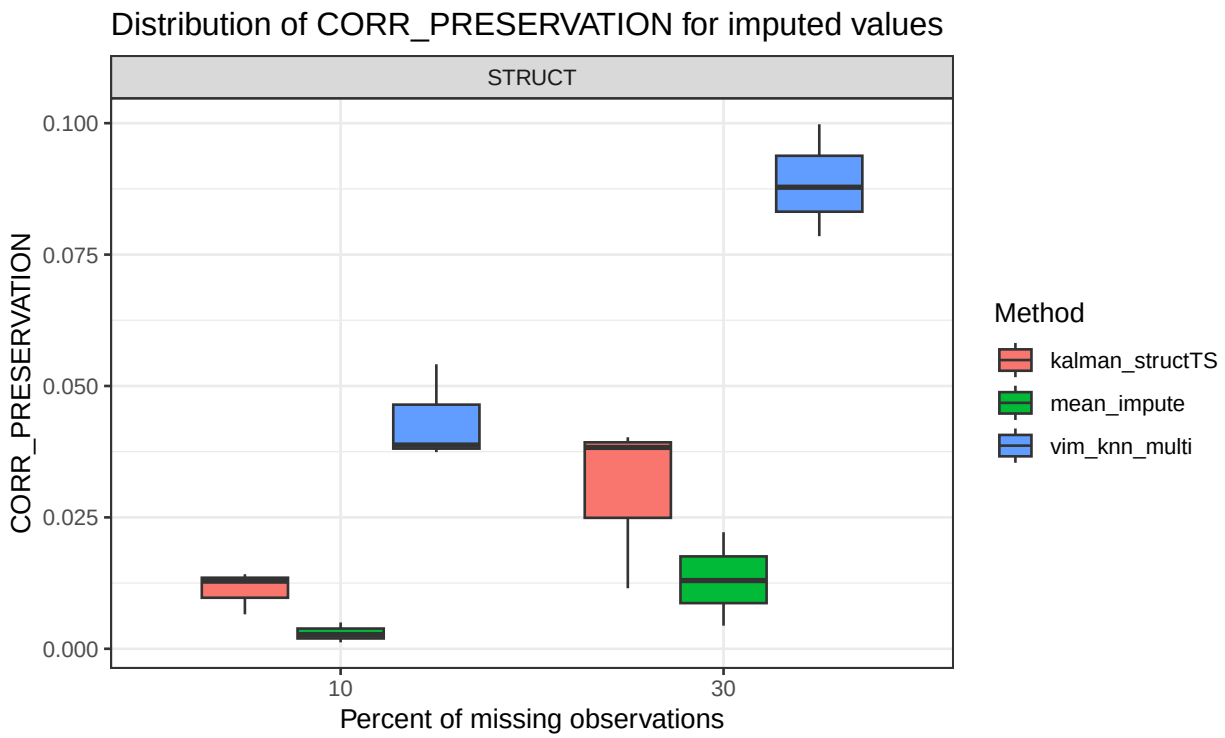
```
list_impute_methods()
#> [1] "interp_linear"           "interp_spline"
#> [3] "kalman_structTS"         "locf_impute"
#> [5] "mean_impute"             "mice_pmm_multi"
#> [7] "missForest_impute_multi" "py_sklearn_knn_multi"
#> [9] "vim_knn_multi"
# Expect: mean_impute, locf_impute, interp_linear, interp_spline, kalman_structTS,
#         missForest_impute_multi, mice_pmm_multi, vim_knn_multi, py_sklearn_knn_multi

set.seed(42)
```

```r
n <- 300
u <- sin(seq(0, 6*pi, length.out = n)) + stats::rnorm(n, sd = 0.1)
X2 <- cbind(A = u, B = u + stats::rnorm(n, sd = 0.1))

meths <- c("mean_impute", "kalman_structTS", "vim_knn_multi", "mice_pmm_multi")
rmse  <- function(truth, estimate) sqrt(mean((truth - estimate)^2, na.rm = TRUE))

res_mix <- impute_errors_multi(
  dataIn      = X2,
  methods     = meths,     # multivariate ones will run jointly; others per column
  error       = "rmse",
  missPercent = c(10, 30),
  repetition  = 2,
  simtype     = "mcar"
)
#> Warning: Number of logged events: 10
#> Warning: Number of logged events: 10
#> Warning: Number of logged events: 10
res_mix
#>            Method Percent      rmse
#> 1 kalman_structTS      10 0.1252577
#> 2     mean_impute      10 0.7014729
#> 3  mice_pmm_multi      10 1.0793988
#> 4   vim_knn_multi      10 1.3133144
#> 5 kalman_structTS      30 0.1334971
#> 6     mean_impute      30 0.7369129
#> 7  mice_pmm_multi      30 0.9842636
#> 8   vim_knn_multi      30 1.2537137
head(attr(res_mix, "by_variable"))
#>            Method Percent Variable     Error
#> 1 kalman_structTS      10        A 0.1092795
#> 2     mean_impute      10        A 0.7049885
#> 3  mice_pmm_multi      10        A 1.2401355
#> 4   vim_knn_multi      10        A 1.3545442
#> 5 kalman_structTS      30        A 0.1144405
#> 6     mean_impute      30        A 0.7348272
plot_errors(res_mix, plotType = "boxplot", facet_by_variable = TRUE)
```



4.7 7. Structural metrics (joint matrix metrics)

```r
set.seed(123)
n <- 300
u <- sin(seq(0, 6*pi, length.out = n)) + stats::rnorm(n, 0, 0.15)
X3 <- cbind(A = u, B = 0.8*u + stats::rnorm(n, 0, 0.15), C = 0.2*u + stats::rnorm(n, 0, 0.15))

meths2 <- c("mean_impute","kalman_structTS","vim_knn_multi")

# Corr preservation (lower is better); results include a "STRUCT" panel in errall
res_corr <- impute_errors_multi(
  dataIn      = X3,
  methods     = meths2,
  error       = "corr_preservation",
  missPercent = c(10, 30),
  repetition  = 3,
  simtype     = "mar",
  blck        = 8
)
head(attr(res_corr, "errall"))
#>            Method Percent Repetition Variable       Error
#> 1     mean_impute      10          1   STRUCT 0.005004990
#> 2 kalman_structTS      10          1   STRUCT 0.014181068
#> 3   vim_knn_multi      10          1   STRUCT 0.054134165
#> 4     mean_impute      10          2   STRUCT 0.002697006
#> 5 kalman_structTS      10          2   STRUCT 0.006558702
#> 6   vim_knn_multi      10          2   STRUCT 0.037414241
plot_errors(res_corr, plotType = "boxplot", facet_by_variable = TRUE)
```



5 Notes & caveats

- Some Kalman/StructTS fits may emit convergence warnings on noisy data; this does **not** affect the benchmark flow.
- If optional dependencies are missing, those methods are **auto-skipped** and reported (e.g., `VIM`, `reticulate` + `sklearn`).
- Structural metrics return a single scalar per method & repetition (stored under `Variable = "STRUCT"`).

6 Session info

```
sessionInfo()
#> R version 4.4.1 (2024-06-14 ucrt)
#> Platform: x86_64-w64-mingw32/x64
#> Running under: Windows 11 x64 (build 26100)
#>
#> Matrix products: default
#>
#>
#> locale:
#> [1] LC_COLLATE=English_India.utf8  LC_CTYPE=English_India.utf8
#> [3] LC_MONETARY=English_India.utf8 LC_NUMERIC=C
#> [5] LC_TIME=English_India.utf8
#>
```

```
#> time zone: Asia/Dubai
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] ggplot2_3.5.2
#>
#> loaded via a namespace (and not attached):
#>  [1] tidyselect_1.2.1    timeDate_4041.110   dplyr_1.1.4
#>  [4] farver_2.1.2        fastmap_1.2.0       rpart_4.1.23
#>  [7] digest_0.6.36       lifecycle_1.0.4     survival_3.6-4
#> [10] magrittr_2.0.3      compiler_4.4.1      rlang_1.1.4
#> [13] tools_4.4.1         utf8_1.2.4          yaml_2.3.8
#> [16] data.table_1.15.4   knitr_1.48          labeling_0.4.3
#> [19] sp_2.1-4            curl_5.2.1          xml2_1.3.6
#> [22] TTR_0.24.4          abind_1.4-8         withr_3.0.1
#> [25] purrr_1.0.2         imputeTS_3.3        nnet_7.3-19
#> [28] grid_4.4.1          fansi_1.0.6         xts_0.14.0
#> [31] jomo_2.7-6          e1071_1.7-14        colorspace_2.1-0
#> [34] mice_3.18.0         scales_1.3.0        iterators_1.0.14
#> [37] MASS_7.3-60.2       tinytex_0.53        cli_3.6.3
#> [40] rmarkdown_2.29      reformulas_0.4.1    generics_0.1.3
#> [43] rstudioapi_0.16.0   robustbase_0.99-4-1 minqa_1.2.8
#> [46] proxy_0.4-27        splines_4.4.1       forecast_8.24.0
#> [49] parallel_4.4.1      urca_1.3-4          vctrs_0.6.5
#> [52] boot_1.3-30         glmnet_4.1-10       Matrix_1.7-0
#> [55] carData_3.0-5       car_3.1-3           tseries_0.10-58
#> [58] stinepack_1.5       mitml_0.4-5         Formula_1.2-5
#> [61] vcd_1.4-13          foreach_1.5.2       tidyr_1.3.1
#> [64] quantmod_0.4.26     glue_1.7.0          pan_1.9
#> [67] nloptr_2.2.1        DEoptimR_1.1-4      codetools_0.2-20
#> [70] ggtext_0.1.2        gtable_0.3.5        shape_1.4.6.1
#> [73] quadprog_1.5-8      lme4_1.1-37         lmtest_0.9-40
#> [76] munsell_0.5.1       tibble_3.2.1        pillar_1.9.0
#> [79] htmltools_0.5.8.1   VIM_6.2.2           R6_2.5.1
#> [82] Rdpack_2.6.4        evaluate_1.0.1      lattice_0.22-6
#> [85] highr_0.11          rbibutils_2.3       backports_1.5.0
#> [88] gridtext_0.1.5      broom_1.0.6         fracdiff_1.5-3
#> [91] class_7.3-22        Rcpp_1.0.12         nlme_3.1-164
#> [94] laeken_0.5.3        ranger_0.17.0       xfun_0.45
#> [97] zoo_1.8-12          pkgconfig_2.0.3
```