Package 'ctreeMI'

July 25, 2026

Title Conditional Inference Trees with Stacked Multiple Imputation

Version 0.3.0

Description Implements the stacked-imputation workflow for conditional inference trees ('ctree') described in Sherlock et al. (2026) <doi:10.1080/00273171.2026.2661244>. When data contain missing values, multiply imputed datasets (e.g., from 'mice') are stacked vertically and a single 'ctree' is fit on the combined data. To correct for the artificially inflated sample size introduced by stacking, each node-level test statistic is divided by the number of imputations M and its p-value recomputed before nodes are pruned (the Stack/M correction), producing a conservative but interpretable single tree that incorporates imputation uncertainty without requiring pooling of structurally different trees. Also exports stack_imputations(), rescale_statistic() and prune_stackM() as standalone utilities. The underlying 'ctree' algorithm is provided by 'partykit' (Hothorn & Zeileis, 2015; Hothorn, Hornik & Zeileis, 2006 <doi:10.1198/106186006X133933>).

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Imports partykit (>= 1.2-0), mice (>= 3.0.0), stats, methods

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 4.0.0)

URL https://github.com/Phillip-Sherlock/ctreeMI

BugReports https://github.com/Phillip-Sherlock/ctreeMI/issues

NeedsCompilation no

Author Phillip Sherlock [aut, cre] (ORCID: <https://orcid.org/0000-0003-0433-3681>)

Maintainer Phillip Sherlock <phillip.sherlock@ufl.edu>

Repository CRAN

Date/Publication 2026-07-25 21:50:01 UTC

Contents

`ctreeMI-package` *Conditional Inference Trees with Stacked Multiple Imputation*

Description

Implements the stacked-imputation workflow for conditional inference trees ('ctree') described in Sherlock et al. (2026) <doi:10.1080/00273171.2026.2661244>. When data contain missing values, multiply imputed datasets (e.g., from 'mice') are stacked vertically and a single 'ctree' is fit on the combined data. To correct for the artificially inflated sample size introduced by stacking, each node-level test statistic is divided by the number of imputations M and its p-value recomputed before nodes are pruned (the Stack/M correction), producing a conservative but interpretable single tree that incorporates imputation uncertainty without requiring pooling of structurally different trees. Also exports stack_imputations(), rescale_statistic() and prune_stackM() as standalone utilities. The underlying 'ctree' algorithm is provided by 'partykit' (Hothorn & Zeileis, 2015; Hothorn, Hornik & Zeileis, 2006 <doi:10.1198/106186006X133933>).

Details

The main function is `ctree_stacked`, which accepts a `mids` object from **mice**, a list of imputed data frames, or a plain data frame. It returns a `ctreeMI` object that inherits from **partykit**'s `constparty` class, so all standard methods (`plot`, `predict`, `nodeids`, etc.) work without modification.

Two utility functions are also exported: `stack_imputations` (stack a list of data frames) and `rescale_alpha` (compute `alpha / M`).

Author(s)

Phillip Sherlock [aut, cre] (<https://orcid.org/0000-0003-0433-3681>)

Maintainer: Phillip Sherlock <phillip.sherlock@ufl.edu>

References

Sherlock, P., Mansolf, M., Hofheimer, J., Hockett, C. W., O'Connor, T. G., Roubinov, D., Graff, J. C., Lai, J.-S., Bush, N. R., Wright, R. J., & Chiu, Y.-H. M. (2026). Beyond linear risk: A machine learning approach to understanding perinatal depression in context. *Multivariate Behavioral Research*, 1–16. doi:10.1080/00273171.2026.2661244

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3), 651–674. doi:10.1198/106186006X133933

Hothorn, T., & Zeileis, A. (2015). **partykit**: A modular toolkit for recursive partitioning in R. *Journal of Machine Learning Research*, 16, 3905–3909.

See Also

`ctree_stacked`, `stack_imputations`, `rescale_alpha`, `ctree`, `mice`

`ctree_stacked` *Conditional Inference Tree on Stacked Multiply Imputed Data*

Description

Fits a conditional inference tree (`ctree`) on stacked multiply imputed datasets using the Stack / M rescaling procedure described in Sherlock et al. (2026). Multiply imputed datasets are concatenated vertically ("stacked"), and the significance threshold used for node-level pruning is divided by the number of imputations M to counteract the artificially inflated sample size. This yields a single, coherent, interpretable tree that incorporates imputation variability without requiring the pooling of structurally different trees.

Usage

```
ctree_stacked(formula, data, m = NULL, alpha = 0.05, verbose = TRUE, ...)
```

Arguments

`formula`	A model formula, passed to `ctree`.
`data`	A `mids` object from `mice`, a list of imputed data frames, or a single complete data frame. If a single complete data frame is supplied, the function falls back to a standard `ctree` call with a warning.
`m`	Integer. Number of imputations to use. Defaults to all available datasets. Ignored when `data` is a plain data frame.
`alpha`	Numeric. Nominal significance threshold for node-level splitting (default `0.05`). The corrected threshold actually applied is `alpha / m`. Must be strictly between 0 and 1.
`verbose`	Logical. If `TRUE` (default), prints a message summarizing the stacking and correction applied.
`...`	Additional arguments passed to `ctree_control`.

Details

Methodological background

Conditional inference trees (Hothorn, Hornik & Zeileis, 2006) use permutation-based significance tests to select splits, providing built-in protection against spurious partitioning. When data are multiply imputed, pooling trees fitted to separate imputations is infeasible: structurally different trees define different subgroups, making the targets of inference incomparable across imputations.

Rodgers et al. (2021) proposed stacking the M imputed datasets and fitting a single tree to the combined data. This produces one coherent, interpretable tree. The complication is that stacking inflates the nominal sample size by M, causing test statistics at each node to be similarly inflated.

Sherlock et al. (2026) proposed and validated the **Stack / M correction**: use a significance threshold of `alpha / M`. Monte Carlo simulations under MCAR confirmed sub-nominal (conservative) type-I error and acceptable power, making this approach well-suited for exploratory analyses where interpretability is prioritised.

Value

An object of class `c("ctreeMI", "constparty", "party")`. All **partykit** methods (`plot`, `predict`, etc.) work on this object. An additional `ctreeMI_info` attribute carries:

`m` Number of imputations used.

`n_original` Rows in one imputed dataset.

`n_stacked` Total rows in the stacked dataset ($M \times n$).

`alpha_nominal` Nominal alpha supplied by the user.

`alpha_applied` Corrected alpha applied (`alpha / m`).

`formula` The model formula.

`call` The matched call.

Node-level sample sizes reported by `print()` and `plot()` reflect the stacked dataset. Divide by M to obtain effective per-node counts in the original data.

References

Sherlock, P., Mansolf, M., Hofheimer, J., Hockett, C. W., O'Connor, T. G., Roubinov, D., Graff, J. C., Lai, J.-S., Bush, N. R., Wright, R. J., & Chiu, Y.-H. M. (2026). Beyond linear risk: A machine learning approach to understanding perinatal depression in context. *Multivariate Behavioral Research*, 1–16. doi:10.1080/00273171.2026.2661244

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3), 651–674. doi:10.1198/106186006X133933

Hothorn, T., & Zeileis, A. (2015). **partykit**: A modular toolkit for recursive partitioning in R. *Journal of Machine Learning Research*, 16, 3905–3909.

Rodgers, J., Khoo, S.-T., & Ludtke, O. (2021). Handling missing data in structural equation models using multiple imputation and stacking. *Structural Equation Modeling*, 28(6), 915–930.

See Also

ctree, ctree_control, mice, stack_imputations, rescale_alpha, print.ctreeMI, summary.ctreeMI

Examples

```
  library(mice)

  # Introduce missingness into the airquality dataset
  set.seed(42)
  aq <- airquality
  aq$Ozone[sample(nrow(aq), 20)]   <- NA
  aq$Solar.R[sample(nrow(aq), 15)] <- NA

  # Impute (M = 20)
  imp <- mice(aq, m = 20, printFlag = FALSE)

  # Fit ctree with Stack/M correction
  fit <- ctree_stacked(Ozone ~ Solar.R + Wind + Temp + Month,
                       data  = imp,
                       alpha = 0.05)
  print(fit)
  plot(fit)


# Example using a list of data frames (no mice required)
set.seed(1)
make_df <- function(i) {
  set.seed(i)
  n  <- 100
  x1 <- rnorm(n)
  y  <- x1 + rnorm(n)
  data.frame(y = y, x1 = x1)
}
imp_list <- lapply(1:10, make_df)
fit <- ctree_stacked(y ~ x1, data = imp_list, alpha = 0.05, verbose = FALSE)
print(fit)
```

---

node_table *Terminal-Node Table with Effective Sample Sizes*

Description

Summarizes the nodes of a fitted ctree_stacked tree as a data frame, giving for each node the sequence of splits that defines it, its size in the stacked data, and its *effective* sample size on the original scale (stacked size divided by the number of imputations M).

Because a stacked dataset repeats every observation M times, node sizes printed by `print()` and `plot()` are M times larger than the number of observations they represent. Reporting those numbers directly overstates precision. This function performs the division so that node sizes can be reported and interpreted correctly.

Usage

```
node_table(object, terminal_only = TRUE, max_levels = NULL, digits = 3)
```

Arguments

`object`	An object of class `"ctreeMI"` as returned by `ctree_stacked`.
`terminal_only`	Logical. If `TRUE` (default) only terminal nodes are returned. If `FALSE`, inner nodes are included as well, which is useful for describing the tree structure.
`max_levels`	Integer or `NULL`. For splits on unordered factors with many levels, the number of levels to print before truncating (e.g. `{A, B, +3 more}`). `NULL` (default) prints the full level set.
`digits`	Integer. Number of decimal places for outcome summaries.

Details

Note that `effective_n` is exact at the root and is an observation-scale rescaling elsewhere. Because imputed values differ across imputations, an individual's M rows are not guaranteed to follow the same path down the tree, so `effective_n` should be read as the node's size expressed on the original scale rather than as a count of distinct individuals.

Value

A data frame of class `"ctreeMI_nodes"` with one row per node and the following columns:

`node_id` Node identifier, matching `print()` and `plot()` output.

`depth` Number of splits from the root to this node.

`n_stacked` Number of rows in the stacked data.

`effective_n` `n_stacked / M`: the node size on the scale of the original data. This is the number to report.

outcome summaries Node means for numeric responses, or the proportion in the second level for binary factor responses. One column per response variable, so multivariate outcomes are supported.

`path` The split conditions leading to the node, joined by `" & "`, suitable for pasting into a manuscript table.

`conditions` A list-column holding the same conditions as a character vector, for programmatic use.

References

Sherlock, P., Mansolf, M., Hofheimer, J., Hockett, C. W., O'Connor, T. G., Roubinov, D., Graff, J. C., Lai, J.-S., Bush, N. R., Wright, R. J., & Chiu, Y.-H. M. (2026). Beyond linear risk: A machine learning approach to understanding perinatal depression in context. *Multivariate Behavioral Research*, 1–16. doi:10.1080/00273171.2026.2661244

See Also

`ctree_stacked`, `report_ctreeMI`

Examples

```
set.seed(1)
make_df <- function(i) {
  set.seed(i)
  n  <- 200
  x1 <- rnorm(n)
  x2 <- factor(sample(c("A", "B", "C"), n, TRUE))
  y  <- 2 * (x1 > 0) + 1.5 * (x2 == "A") + rnorm(n)
  data.frame(y = y, x1 = x1, x2 = x2)
}
imp_list <- lapply(1:10, make_df)
fit <- ctree_stacked(y ~ x1 + x2, data = imp_list, verbose = FALSE)

# Terminal nodes with effective sample sizes
node_table(fit)

# Truncate long factor level sets
node_table(fit, max_levels = 2)

# Include inner nodes
node_table(fit, terminal_only = FALSE)
```

---

print.ctreeMI *Print Method for ctreeMI Objects*

Description

Prints a header summarizing the stacked-imputation settings, followed by the standard **partykit** tree output.

Usage

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

Arguments

x	An object of class "ctreeMI" as returned by ctree_stacked.
...	Further arguments passed to the **partykit** print method.

Value

x, invisibly.

See Also

ctree_stacked, summary.ctreeMI

Examples

```
set.seed(1)
imp_list <- lapply(1:5, function(i) {
  set.seed(i)
  data.frame(y = rnorm(80), x = rnorm(80))
})
fit <- ctree_stacked(y ~ x, data = imp_list, verbose = FALSE)
print(fit)
```

`report_ctreeMI` *Methods Paragraph for a Stacked-Imputation Tree*

Description

Generates a short methods paragraph describing how a `ctree_stacked` model was fit, populated with the actual values used: the number of imputations, the original and stacked sample sizes, the nominal and applied significance thresholds, and the size and depth of the resulting tree. The paragraph is intended to be edited and pasted into a manuscript, and it standardizes how the Stack/M correction is described across studies.

Usage

```
report_ctreeMI(object, digits = 3)
```

Arguments

`object`	An object of class `"ctreeMI"` as returned by `ctree_stacked`.
`digits`	Integer. Significant digits used when reporting the applied significance threshold.

Details

The generated text describes the imputation and stacking workflow and the Stack/M threshold correction, and states that node sizes are reported on the original scale. It does not describe the imputation model itself, which is specific to the analysis and must be added by the author: the number of iterations, the variables included, and the imputation methods used should be reported alongside this paragraph.

Value

An object of class `"ctreeMI_report"`: a list whose `text` element is the methods paragraph as a single character string, alongside the underlying quantities (`m`, `n_original`, `n_stacked`, `alpha_nominal`, `alpha_applied`, `n_terminal`, `max_depth`, `formula`) so they can be used programmatically. The print method wraps and displays the paragraph.

References

Sherlock, P., Mansolf, M., Hofheimer, J., Hockett, C. W., O'Connor, T. G., Roubinov, D., Graff, J. C., Lai, J.-S., Bush, N. R., Wright, R. J., & Chiu, Y.-H. M. (2026). Beyond linear risk: A machine learning approach to understanding perinatal depression in context. *Multivariate Behavioral Research*, 1–16. doi:10.1080/00273171.2026.2661244

See Also

ctree_stacked, node_table

Examples

```
set.seed(1)
make_df <- function(i) {
  set.seed(i)
  n  <- 200
  x1 <- rnorm(n)
  y  <- 2 * (x1 > 0) + rnorm(n)
  data.frame(y = y, x1 = x1)
}
imp_list <- lapply(1:10, make_df)
fit <- ctree_stacked(y ~ x1, data = imp_list, verbose = FALSE)

# Print the methods paragraph
report_ctreeMI(fit)

# Access the raw text or the underlying quantities
rep <- report_ctreeMI(fit)
rep$n_terminal
substr(rep$text, 1, 60)
```

---

rescale_alpha *Rescale Significance Threshold for the Stack / M Correction*

Description

Computes the adjusted significance threshold `alpha / M` used in the Stack / M correction of Sherlock et al. (2026). Dividing the nominal alpha by the number of imputations M counteracts the inflated test statistics that arise from stacking M copies of the data.

Usage

```
rescale_alpha(alpha = 0.05, m)
```

Arguments

`alpha`	Numeric. Nominal significance level (default `0.05`). Must be strictly between 0 and 1.
`m`	A single positive integer. Number of imputations.

Value

A single numeric value: `alpha / m`.

References

Sherlock, P., Mansolf, M., Hofheimer, J., Hockett, C. W., O'Connor, T. G., Roubinov, D., Graff, J. C., Lai, J.-S., Bush, N. R., Wright, R. J., & Chiu, Y.-H. M. (2026). Beyond linear risk: A machine learning approach to understanding perinatal depression in context. *Multivariate Behavioral Research*, 1–16. doi:10.1080/00273171.2026.2661244

See Also

`ctree_stacked`, `stack_imputations`

Examples

```
rescale_alpha(0.05, 30)   # 0.001666...
rescale_alpha(0.05, 10)   # 0.005
rescale_alpha(0.01, 5)    # 0.002
```

`stack_imputations` *Stack Multiply Imputed Datasets*

Description

Concatenates a list of imputed data frames into a single stacked data frame. An imputation-index column (`.imp`) is added to identify which imputed dataset each row originated from.

Usage

```
stack_imputations(data_list, imp_col = ".imp")
```

Arguments

`data_list`	A list of data frames, all with the same dimensions and column names, representing M imputed versions of the same dataset.
`imp_col`	Character string. Name of the imputation-index column added to the stacked data (default `".imp"`). Set to `NULL` to suppress the column.

Value

A single data frame with $M \times n$ rows, where n is the number of rows in each imputed dataset. If `imp_col` is not `NULL`, an integer column recording the imputation index is appended.

References

Sherlock, P., Mansolf, M., Hofheimer, J., Hockett, C. W., O'Connor, T. G., Roubinov, D., Graff, J. C., Lai, J.-S., Bush, N. R., Wright, R. J., & Chiu, Y.-H. M. (2026). Beyond linear risk: A machine learning approach to understanding perinatal depression in context. *Multivariate Behavioral Research*, 1–16. doi:10.1080/00273171.2026.2661244

Rodgers, J., Khoo, S.-T., & Ludtke, O. (2021). Handling missing data in structural equation models using multiple imputation and stacking. *Structural Equation Modeling*, 28(6), 915–930.

See Also

`ctree_stacked`, `rescale_alpha`

Examples

```
df1 <- data.frame(x = 1:5, y = c(2, 4, 6, 8, 10))
df2 <- data.frame(x = 1:5, y = c(2, 3, 6, 9, 10))
df3 <- data.frame(x = 1:5, y = c(1, 4, 5, 8, 11))

stacked <- stack_imputations(list(df1, df2, df3))
nrow(stacked)           # 15
table(stacked$.imp)     # 5 rows per imputation
```

---

summary.ctreeMI *Summary Method for ctreeMI Objects*

Description

Prints and returns a summary of the stacked-imputation fit and the resulting tree structure (number of nodes, maximum depth).

Usage

```
## S3 method for class 'ctreeMI'
summary(object, ...)
```

Arguments

`object`	An object of class `"ctreeMI"` as returned by `ctree_stacked`.
`...`	Currently unused.

Value

A list (returned invisibly) with components:

`ctreeMI_info` Stacking metadata from `ctree_stacked`.

`n_terminal_nodes` Number of terminal nodes.

`depth` Maximum depth of the fitted tree.

See Also

ctree_stacked, print.ctreeMI

Examples

```
set.seed(1)
imp_list <- lapply(1:5, function(i) {
  set.seed(i)
  data.frame(y = rnorm(80), x = rnorm(80))
})
fit <- ctree_stacked(y ~ x, data = imp_list, verbose = FALSE)
summary(fit)
```

Index