Package 'ctreeMI'

August 26, 2026

Title Conditional Inference Trees with Stacked Multiple Imputation

Version 1.0.1

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, every node-level test statistic is divided by the number of imputations M, the node-level p-values are recomputed from the chi-squared reference distribution 'ctree' uses (including its multiplicity adjustment across candidate splitting variables), and the tree is compressed bottom-up (the Stack/M correction). Degrees of freedom are derived for each node and each candidate variable, so univariate, bivariate and higher-dimensional outcomes are all handled, as are unordered factor predictors, whose degrees of freedom depend on how many levels remain in a node. The result is a single interpretable tree that incorporates imputation uncertainty without requiring pooling of structurally different trees. Also exports stack_imputations(), rescale_statistic(), prune_stackM(), node_table() and report_ctreeMI() 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 2026-08-26 13:10:02 UTC

Contents

`check_stackM_extraction` *Check That Node Statistics Can Be Extracted*

Description

Diagnostic. Fits small trees under both of the multiplicity settings the correction supports, and verifies that node-level statistics and degrees of freedom can be recovered from the fitted objects. Run it once after installing, or after upgrading 'partykit'.

Usage

```
check_stackM_extraction(verbose = TRUE)
```

Arguments

`verbose`	Logical. Print a per-node report.

Value

Invisibly, 'TRUE' if extraction succeeded everywhere it was tried.

Examples

```
check_stackM_extraction(verbose = FALSE)
```

`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 one tree is grown on the combined data. Each node-level chi-square statistic is then divided by the number of imputations (M) to counteract the artificially inflated sample size, the node-level p-values are recomputed, and the tree is compressed bottom-up. 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,
  scale_minsize = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`formula`	A model formula, passed to [partykit::ctree()].
`data`	A ‘mids‘ object from [mice::mice()], a list of imputed data frames, or a single complete data frame. If a single complete data frame is supplied (no missing data handling needed), the function falls back to a standard [partykit::ctree()] call with a warning.
`m`	Integer. Number of imputations to use. If ‘data‘ is a ‘mids‘ object or a list, defaults to the number of datasets available. Ignored when ‘data‘ is a plain data frame.
`alpha`	Numeric. The nominal significance threshold for node-level splitting (default 0.05). It is applied to p-values recomputed from node statistics that have been divided by ‘m‘; ‘alpha‘ itself is not rescaled. Must be strictly between 0 and 1.
`scale_minsize`	Logical. If ‘TRUE‘ (default), ‘minsplit‘ and ‘minbucket‘ are multiplied by ‘m‘ so they refer to original rather than stacked observations. The ‘partykit‘ defaults would otherwise permit terminal nodes holding fewer than one original observation.

verbose	Logical. If 'TRUE' (default), prints a message summarising the stacking and correction applied.
...	Additional arguments passed to [partykit::ctree_control()]. Note: 'alpha' in '...' is ignored in favour of the 'alpha' argument above.

Details

Methodological background

When data contain missing values, a common and principled approach is multiple imputation (Rubin, 1987), wherein M completed datasets are generated by drawing from the posterior predictive distribution of the missing values. For most statistical models, results from M datasets are pooled using Rubin's rules.

Conditional inference trees (ctree; Hothorn, Hornik & Zeileis, 2006) are not straightforward to pool across imputations because the tree structure itself – which variables are split, at what values, and in what order – can differ across imputations. Pooling structurally different trees produces inconsistent subgroup definitions and uninterpretable results.

Rodgers et al. (2021) proposed stacking the M imputed datasets vertically and fitting a single tree to the combined data. This circumvents the structural-mismatch problem and produces one interpretable tree. However, stacking inflates the nominal sample size by a factor of M, causing node-level test statistics to be similarly inflated and the tree to split more aggressively than warranted.

Sherlock et al. (2026) proposed and validated the **Stack / M** correction: each node-level test statistic computed on the stacked data is divided by M, the p-value is recomputed from the rescaled statistic, the Bonferroni adjustment for the number of candidate splitting variables is reapplied, and nodes that no longer meet 'alpha' are pruned. See "Scope" and "Node-level calibration" below for the conditions under which the procedure has been examined and for the behaviour of its node-level test.

Correction applied to the statistic, not to alpha

Rescaling the statistic is **not** equivalent to dividing the significance threshold by M. Writing 'q(p, df)' for the chi-square quantile function, the two rules are:

- statistic rescaling (correct): reject when 'X > M * q(1 - alpha, df)'
- threshold rescaling (incorrect): reject when 'X > q(1 - alpha / M, df)'

These coincide only at 'M = 1'. At 'df = 1', 'alpha = 0.05', 'M = 30' the first requires 'X > 115.2' and the second only 'X > 9.9', so threshold rescaling under-corrects by an order of magnitude and grows substantially larger trees than the published method.

Versions 0.1.0 and 0.2.0 of this package implemented threshold rescaling. Trees fitted with those versions are under-corrected and should be refitted.

When to use this procedure

The problem it solves is specific. Multiple imputation produces M completed datasets, and for most models their results are combined by Rubin's rules; a tree offers no estimand that can be averaged, since the M trees may split on different variables in a different order. Stacking avoids that, and the correction addresses the inflated sample size that stacking introduces. If a single interpretable tree is wanted from multiply imputed data, that is what this package is for.

In the simulations archived at the DOI given below, the procedure recovered known structure more often than listwise deletion, surrogate splits, missingness incorporated in attributes, or single imputation, and produced the most stable partitions across independent sets of imputations. Its accuracy held across missingness rates of 15 while each of those four declined, listwise deletion most steeply. Tree sizes tracked the true size closely wherever real structure was present, and were stable from M = 5 to M = 50.

Two settings matter in practice. The outcome should be included in the imputation model, which ‘mice()‘ does by default: omitting it attenuates the associations a tree is meant to find, at a cost described under "Node-level calibration" below. And M should follow the usual guidance for the fraction of missing information; M = 30 was used throughout these simulations.

The procedure suits least the exploratory case in which no structure may be present, since the miscalibration described below is concentrated under a true null. A single unreplicated subgroup found in data with no prior reason to expect one warrants the checks given below rather than the node-level p-value.

Scope

The procedure assumes the data are missing at random in the sense of Rubin (1976), as does the multiple imputation it is built on. In the simulations archived at the DOI given below it was examined under MCAR and MAR across a range of missingness rates, sample sizes and outcome types, and it is intended for use where missingness is plausibly at random. Under MNAR, recovery of the true structure degrades for every imputation-based approach, this one included, and a tree fitted in that setting should be treated as provisional. The mechanism is rarely known in applied work, and MNAR cannot be distinguished from MAR using the observed data, so this is a statement about where the assumption is defensible rather than a condition that can be checked.

Node-level calibration

The node-level test is not calibrated to its nominal level when the imputation model conditions on the outcome. That is recommended practice, and it is what ‘mice()‘ does by default when run on a data frame containing the outcome. Under a true null the test rejects more often than ‘alpha‘ implies, increasingly so as the missingness rate rises. Omitting the outcome from the imputation model reverses this into conservatism rather than restoring calibration, and is not recommended, since it attenuates the associations a tree is meant to detect.

Node-level p-values should therefore be read as approximate rather than nominal. Two checks are available in their place. [node_table()] reports the effective sample size behind each terminal node, in original rather than stacked observations, so a node resting on few original cases can be identified. And refitting on an independent set of imputations, then comparing the resulting partitions, indicates whether the structure is stable.

Simulations characterising this behaviour are archived at doi:10.5281/zenodo.21939940.

Documentation prior to version 1.0.1 described the correction as sub-nominal under MCAR. That characterisation comes from the simulations in Sherlock et al. (2026), which used a marginal imputation model conditioning on neither the outcome nor the remaining predictors. It does not hold under outcome-conditioned imputation.

Usage with ‘mice‘

“‘r library(mice) imp <- mice(my_data, m = 30, printFlag = FALSE) fit <- ctree_stacked(outcome ~ ., data = imp, alpha = 0.05) print(fit) plot(fit) “‘

Usage with a list of data frames

“‘r imp_list <- lapply(1:30, function(i) complete(imp, i)) fit <- ctree_stacked(outcome ~ ., data = imp_list, alpha = 0.05) “‘

Value

An object of class ‘c("ctreeMI", "constparty", "party")‘. This inherits all methods from [partykit::ctree()] output and additionally carries a ‘ctreeMI_info‘ attribute with the following elements:

‘m‘ Number of imputations used.

‘n_original‘ Number of rows in a single imputed dataset.

‘n_stacked‘ Total rows in the stacked dataset.

‘alpha‘ The significance threshold applied to the rescaled p-values.

‘correction‘ Character, ‘"statistic/M"‘.

‘node_stats‘ Data frame of per-node raw statistics, degrees of freedom, rescaled statistics, p-values, and retention.

‘n_splits_before‘, ‘n_splits_after‘ Splits before and after the correction was applied.

‘outcome_dim‘ Rank of the influence function of the response, i.e. the degrees of freedom carried by a numeric or ordered predictor.

‘minsplit‘, ‘minbucket‘ Minimum node sizes actually used, in stacked rows.

‘formula‘ The model formula.

‘call‘ The matched call.

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, D. M., Jacobucci, R., & Grimm, K. J. (2021). A multiple imputation approach for handling missing data in classification and regression trees. *Journal of Behavioral Data Science*, 1(1), 127-153. doi:10.35566/jbds/v1n1/p6

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys.* Wiley.

See Also

[partykit::ctree()], [partykit::ctree_control()], [mice::mice()], [stack_imputations()], [rescale_statistic()], [prune_stackM()], [node_table()]

Examples

```
## Not run:
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
imp <- mice(aq, m = 10, 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)

## End(Not run)
```

---

node_table *Terminal-Node Summary With Effective Sample Sizes*

Description

Summarises the nodes of a fitted tree: the split path leading to each node, its size in the stacked data, its effective size in original observations (stacked size divided by ‘M‘), and node-level outcome summaries.

Usage

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

Arguments

object	A ‘ctreeMI‘ object from [ctree_stacked()], or any ‘party‘ object (in which case ‘M‘ is taken to be 1).
terminal_only	Logical. Report terminal nodes only (default), or every node including the root.
max_levels	Integer or ‘NULL‘. Truncate factor level sets longer than this in the printed split path.
digits	Number of digits for the outcome summaries.

Details

Terminal-node sample sizes printed by ‘partykit‘ refer to the stacked data, in which every observation appears ‘M‘ times. ‘effective_n‘ divides by ‘M‘ to give the number of original observations behind each node. Node-level means and proportions need no such adjustment: they are already averages over the imputations.

Value

A data frame of class ‘"ctreeMI_nodes"‘ with one row per node: ‘node_id‘, ‘depth‘, ‘n_stacked‘, ‘effective_n‘, any outcome summaries, ‘path‘, and a ‘conditions‘ list column holding the split conditions individually.

See Also

[ctree_stacked()], [report_ctreeMI()]

Examples

```
set.seed(1)
imps <- lapply(1:5, function(i) {
  x <- stats::rnorm(200)
  data.frame(x = x, y = stats::rnorm(200) + 1.5 * (x > 0))
})
fit <- ctree_stacked(y ~ x, data = imps, verbose = FALSE)
node_table(fit)
```

---

print.ctreeMI *Print Method for ctreeMI Objects*

Description

Prints a summary of the stacked-imputation settings used to fit the tree, followed by the standard [partykit::ctree()] 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.

print.ctreeMI_nodes *Print a Node Table*

Description

Print a Node Table

Usage

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

Arguments

x	A ‘"ctreeMI_nodes"‘ object from [node_table()].
...	Passed to [print.data.frame()].

Value

‘x‘, invisibly.

print.ctreeMI_report *Print a Methods Paragraph*

Description

Print a Methods Paragraph

Usage

```
## S3 method for class 'ctreeMI_report'
print(x, width = 76, ...)
```

Arguments

x	A ‘"ctreeMI_report"‘ object from [report_ctreeMI()].
width	Wrapping width in characters.
...	Currently unused.

Value

‘x‘, invisibly.

`prune_stackM` *Prune a Stacked Tree Using the Stack / M Correction*

Description

Applies the Stack / M correction of Sherlock et al. (2026) to a conditional inference tree fitted on stacked multiply imputed data. Every node-level test statistic is divided by ‘M‘, the p-values are recomputed from the chi-square reference distribution ‘ctree‘ uses, the multiplicity adjustment over candidate splitting variables is reapplied, and the tree is compressed bottom-up.

Usage

```
prune_stackM(tree, m, alpha = 0.05, verbose = TRUE)
```

Arguments

`tree`	A ‘party‘/‘constparty‘ object fitted by [partykit::ctree()] on stacked data, with ‘teststat = "quadratic"‘ and ‘testtype‘ either ‘"Univariate"‘ or ‘"Bonferroni"‘. Both are handled: whether the stored p-values already carry ‘partykit‘’s multiplicity adjustment is read from the fitted control rather than assumed.
`m`	Integer. Number of imputations.
`alpha`	Numeric. Nominal significance threshold.
`verbose`	Logical. Report how many splits were retained.

Details

What is tested

‘ctree‘ splits a node when *any* candidate variable meets ‘alpha‘, so that is the rule reapplied here: all candidates tested at the node are rescaled and the smallest corrected p-value decides. Testing only the variable that was split on would not be equivalent, because candidates carry different degrees of freedom and rescaling can reorder them.

How the tree is compressed

Pruning is bottom-up, as in the analysis for the paper. An internal node is collapsed only once all of its own internal descendants have been collapsed, so a node whose descendant survives the correction keeps its split.

Degrees of freedom

With ‘teststat = "quadratic"‘ the node-level statistic is chi-square with ‘df‘ equal to the rank of the covariance matrix of the linear statistic: the outcome dimension for a numeric or ordered predictor (1 for a univariate outcome, 2 for a bivariate outcome, and so on), and ‘(L - 1)‘ times the outcome dimension for an unordered factor with ‘L‘ levels present in the node. These are derived for every node and every candidate variable, by inverting the (statistic, p-value) pairs ‘partykit‘ stored and falling back on the structural rule where the stored p-value has underflowed to zero, which happens routinely on stacked data.

Value

A list with the pruned ‘tree‘ and a ‘node_stats‘ data frame with one row per internal node of the unpruned tree, recording the variable split on, its raw and rescaled statistics, the degrees of freedom and how they were obtained, the p-values before and after correction, the smallest corrected p-value over all candidates at that node, and whether the split was retained. The full per-candidate table is attached as ‘attr(node_stats, "candidates")‘.

References

Sherlock, P., et al. (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

Examples

```
set.seed(1)
n <- 300
d <- data.frame(x = stats::rnorm(n))
d$y1 <- stats::rnorm(n) + 1.2 * (d$x > 0)
d$y2 <- stats::rnorm(n) + 0.9 * (d$x > 0)
grown <- partykit::ctree(y1 + y2 ~ x, data = d)
out <- prune_stackM(grown, m = 5, verbose = FALSE)
out$node_stats
```

---

report_ctreeMI *A Methods Paragraph For a ctreeMI Analysis*

Description

Generates a paragraph describing the analysis in the form usually required by a methods section: the number of imputations, the size of the stacked dataset, the model, the Stack / M correction and the threshold applied, how many candidate splits survived it, and the shape of the resulting tree.

Usage

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

Arguments

`object`	A ‘ctreeMI‘ object from [ctree_stacked()].
`digits`	Number of digits used in the node summaries.

Value

An object of class ‘"ctreeMI_report"‘: a list whose ‘text‘ element is the paragraph, alongside the quantities it reports.

See Also

[ctree_stacked()], [node_table()]

Examples

```
set.seed(1)
imps <- lapply(1:5, function(i) {
  x <- stats::rnorm(200)
  data.frame(x = x, y = stats::rnorm(200) + 1.5 * (x > 0))
})
fit <- ctree_stacked(y ~ x, data = imps, verbose = FALSE)
report_ctreeMI(fit)
```

---

rescale_statistic *Rescale a Node-Level Test Statistic for Stacking*

Description

Divides a node-level chi-square test statistic by the number of imputations ‘M‘ and recomputes the corresponding p-value. This is the Stack / M correction of Sherlock et al. (2026).

Usage

```
rescale_statistic(statistic, m, df)
```

Arguments

`statistic`	Numeric. The raw test statistic computed on the stacked data.
`m`	Integer. Number of imputations.
`df`	Numeric. Degrees of freedom of the reference chi-square distribution: the outcome dimension for a numeric or ordered predictor, ‘(L - 1)‘ times the outcome dimension for an unordered factor with ‘L‘ levels. See [prune_stackM()], which derives this for you.

Details

Stacking ‘M‘ imputed datasets produces ‘M * n‘ rows, and a chi-square statistic computed on the stacked data scales approximately linearly with ‘M‘. Dividing the statistic by ‘M‘ returns it to the scale of a single imputed dataset before the p-value is computed.

This is **not** the same as dividing the significance threshold by ‘M‘. The two rules coincide only at ‘M = 1‘:

- Statistic rescaling rejects when ‘X > M * qchisq(1 - alpha, df)‘.
- Threshold rescaling rejects when ‘X > qchisq(1 - alpha / M, df)‘.

For ‘df = 1‘, ‘alpha = 0.05‘ and ‘M = 30‘, the first requires ‘X > 115.2‘ and the second only ‘X > 9.9‘. Versions 0.1.0 and 0.2.0 implemented the second rule.

Value

A list with ‘statistic_rescaled‘ and ‘p_value‘.

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

Examples

```
rescale_statistic(115.2, m = 30, df = 1)
```

---

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. This is the first step of the stacked-imputation workflow described in Rodgers et al. (2021) and applied to conditional inference trees in Sherlock et al. (2026).

Usage

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

Arguments

`data_list`	A list of data frames, each the same dimensions, 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 this column.

Value

A single data frame with ‘M * n‘ rows, where ‘n‘ is the number of rows in each imputed dataset.

References

Sherlock, P., et al. (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, D. M., Jacobucci, R., & Grimm, K. J. (2021). A multiple imputation approach for handling missing data in classification and regression trees. *Journal of Behavioral Data Science*, 1(1), 127-153. doi:10.35566/jbds/v1n1/p6

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))
stacked <- stack_imputations(list(df1, df2))
nrow(stacked) # 10
table(stacked$.imp)
```

`summary.ctreeMI` *Summary Method for ctreeMI Objects*

Description

Returns a named list with details about the stacked-imputation fit and the resulting tree structure (number of terminal nodes, tree 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 (invisibly) with components:

‘ctreeMI_info‘ Stacking metadata from [ctree_stacked()].

‘n_terminal_nodes‘ Number of terminal nodes in the fitted tree.

‘depth‘ Maximum depth of the fitted tree.

Index