Package ‘cjbart’

March 2, 2022

Title Heterogeneous Effects Analysis of Conjoint Experiments

Version 0.2.2

Description A tool for analyzing conjoint experiments using Bayesian Additive Regression Trees ('BART'), a machine learning method developed by Chipman, George and McCulloch (2010) <doi:10.1214/09-AOAS285>. This tool focuses specifically on estimating, identifying, and visualizing the heterogeneity within marginal component effects, at the observation- and individual-level. It uses a variable importance measure ('VIMP') with delete-d jackknife variance estimation, following Ishwaran and Lu (2019) <doi:10.1002/sim.7803>, to obtain bias-corrected estimates of which variables drive heterogeneity in the predicted individual-level effects.

License Apache License (>= 2.0)

Encoding UTF-8

RoxygenNote 7.1.2

Depends R (>= 3.6.0), BART

Imports stats, rlang, tidyr, ggplot2, randomForestSRC, Rdpack

Suggests testthat, knitr, cregg, rmarkdown

VignetteBuilder knitr

URL https://github.com/tsrobinson/cjbart

BugReports https://github.com/tsrobinson/cjbart/issues

RdMacros Rdpack

NeedsCompilation no

Author Thomas Robinson [aut, cre, cph]
(<https://orcid.org/0000-0001-7097-1599>),
Raymond Duch [aut, cph] (<https://orcid.org/0000-0002-1166-7674>)

Maintainer Thomas Robinson <ts.robinson1994@gmail.com>

Repository CRAN

Date/Publication 2022-03-02 14:20:18 UTC

R topics documented:

`cjbart` *Generate Conjoint Model Using BART*

Description

A wrapper for the `BART::pbart()` function.

Usage

```
cjbart(data, Y, id = NULL, round = NULL, use_round = TRUE, cores = 1, ...)
```

Arguments

`data`	A data.frame, containing all attributes, controls, the outcome and id variables to analyze.
`Y`	Character string – the outcome variable
`id`	Character string – variable identifying individual respondents (optional)
`round`	Character string – variable identifying rounds of the conjoint experiment
`use_round`	Boolean – whether to include the round indicator column when training the BART model (default = `TRUE`)
`cores`	Integer – number of CPU cores used in model training
`...`	Other arguments passed to `BART::pbart()`

Details

Please note, `cjbart` currently only works for a binary outcome.

Value

A trained `BART::pbart()` model that can be passed to `IMCE()`

See Also

`BART::pbart()`

Examples

```
subjects <- 5
rounds <- 2
profiles <- 2
obs <- subjects*rounds*profiles

fake_data <- data.frame(A = sample(c("a1","a2"), obs, replace = TRUE),
                        B = sample(c("b1","b2"), obs, replace = TRUE),
                        id1 = rep(1:subjects, each=rounds),
                        stringsAsFactors = TRUE)

fake_data$Y <- sample(c(0,1), obs, replace = TRUE)

cj_model <- cjbart(data = fake_data,
                   Y = "Y",
                   id = "id1")
```

`het_vimp` *Estimate Variable Importance Metrics for* `cjbart` *Object*

Description

Estimates random forest variable importance scores for multiple attribute-levels of a conjoint experiment.

Usage

```
het_vimp(model, outcomes = NULL, covars = NULL)
```

Arguments

`model`	Object of class `cjbart`, the result of running `IMCE()`
`outcomes`	An optional vector of attribute levels to generate importance metrics for. By default, all attribute-levels are analyzed.
`covars`	An optional vector of covariates to include in the importance metric check. By default, all covariates are included in each importance model.

Details

Having generated a schedule of individual-level marginal component effect estimates, this function fits a random forest model for each attribute-level using the supplied covariates as predictors. It then calculates a variable importance measure (VIMP) for each covariate. The VIMP method assesses how important each covariate is in terms of partitioning the predicted individual-level effects distribution, and can thus be used as an indicator of which variables drive heterogeneity in the IMCEs.

To recover a VIMP measure, we used permutation-based importance metrics recovered from random forest models estimated using `randomForestSRC::rfsrc()`. To permute the data, this function uses random node assignment, whereby cases are randomly assigned to a daughter node whenever a tree splits on the target variable (see Ishwaran et al. 2008). Importance is defined in terms of how random node assignment degrades the performance of the forest. Higher degradation indicates a variable is more important to prediction.

Variance estimates of each variable's importance are subsequently recovered using the delete-d jackknife estimator developed by Ishwaran and Lu (2019). The jackknife method has inherent bias correction properties, making it particularly effective for variable selection exercises such as identifying drivers of heterogeneity.

Value

A "long" data.frame of variable importance scores for each combination of covariates and attribute-levels, as well as the estimated 95% confidence intervals for each metric.

References

Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS (2008). "Random survival forests." *The annals of applied statistics*, **2**(3), 841–860.

Ishwaran H, Lu M (2019). "Standard errors and confidence intervals for variable importance in random forest regression, classification, and survival." *Statistics in medicine*, **38**(4), 558–582.

See Also

`randomForestSRC::rfsrc()` and `randomForestSRC::subsample()`

`IMCE` *Heterogeneous Effects Analysis of Conjoint Results*

Description

`IMCE` calculates the individual-level marginal component effects from a BART-estimated conjoint model.

Usage

```
IMCE(
  data,
  model,
  attribs,
  ref_levels,
  method = "bayes",
  alpha = 0.05,
  keep_omce = FALSE,
  cores = 1,
  skip_checks = FALSE
)
```

Arguments

`data`	A data.frame, containing all attributes, covariates, the outcome and id variables to analyze.
`model`	A model object, the result of running `cjbart()`
`attribs`	Vector of attribute names
`ref_levels`	Vector of reference levels, used to calculate marginal effects
`method`	Character string, setting the variance estimation method to use. When method is "parametric", a typical combined variance estimate is employed; when `method = "bayes"`, the 95% posterior interval is calculated; and when `method = "rubin"`, combination rules are used to combine the variance analogous to in multiple imputation analysis.
`alpha`	Number between 0 and 1 – the significance level used to compute confidence/posterior intervals. When `method = "bayes"`, the posterior interval is calculated by taking the alpha/2 and (1-alpha/2) quantiles of the posterior draws. When `method = "rubin"`, the confidence interval equals the IMCE +/- `qnorm(alpha/2)`. By default, alpha is 0.05 i.e. generating a 95% confidence/posterior interval.
`keep_omce`	Boolean, indicating whether to keep the OMCE-level results (default = `FALSE`)
`cores`	Number of CPU cores used during prediction phase
`skip_checks`	Boolean, indicating whether to check the structure of the data (default = `FALSE`). Only set this to `TRUE` if you are confident that the data is structured appropriately

Details

The OMCE estimates are the result of subtracting the predicted value of each observation under the reference-level category from the predicted value of each observation under the given attribute level. If an attribute has *k* levels, then this will yield *k-1* estimates per observation. The IMCE is the average of the OMCEs for each individual within the data.

Value

`IMCE` returns an object of type "cjbart", a list object.

`omce`	A data.frame containing the observation-level marginal effects
`imce`	A data.frame containing the individual-level marginal effects
`imce_upper`	A data.frame containing the upper bound of the IMCE confidence/credible interval
`imce_lower`	A data.frame containing the lower bound of the IMCE confidence/credible interval
`att_levels`	A vector containing the attribute levels

See Also

`cjbart()`

Examples

```
subjects <- 5
rounds <- 2
profiles <- 2
obs <- subjects*rounds*profiles

fake_data <- data.frame(A = sample(c("a1","a2"), obs, replace = TRUE),
                        B = sample(c("b1","b2"), obs, replace = TRUE),
                        id1 = rep(1:subjects, each=rounds),
                        stringsAsFactors = TRUE)

fake_data$Y <- sample(c(0,1), obs, replace = TRUE)

cj_model <- cjbart(data = fake_data,
                   Y = "Y",
                   id = "id1")

## Skip if not Unix due to longer CPU time
if (.Platform$OS.type=='unix') {

  het_effects <- IMCE(data = fake_data,
                      model = cj_model,
                      attribs = c("A","B"),
                      ref_levels = c("a1","b1"),
                      cores = 1)

  summary(het_effects)
}
```

`plot.cjbart` *Plot Marginal Component Effects of a* `cjbart` *Object*

Description

Plots observation-level or individual-level marginal component effects (OMCE and IMCE respectively). By default, all attribute-levels in the model are plotted.

Usage

```
## S3 method for class 'cjbart'
plot(x, covar = NULL, plot_levels = NULL, se = TRUE, ...)
```

Arguments

`x`	Object of class `cjbart`, the result of running `IMCE()`
`covar`	Character string detailing the covariate over which to analyze heterogeneous effects

plot_levels	Optional vector of conjoint attribute names to plot. If not supplied, all attributes within the conjoint model will be plotted.
se	Boolean determining whether to show an estimated 95% confidence interval
...	Additional arguments for plotting the marginal component effects (see below).

Value

Plot of marginal component effects.

---

plot.cjbart.vimp *Plot Variable Importance Matrix for Heterogeneity Analysis*

Description

Plots a heatmap of variable importance, across predicted IMCEs. By default, all attribute-levels and covariates in the model are plotted.

Usage

```
## S3 method for class 'cjbart.vimp'
plot(x, covars = NULL, att_levels = NULL, ...)
```

Arguments

x	Object of class `cjbart`, the result of running `IMCE()`
covars	Optional vector of covariate names to plot. By default, all included covariates are shown.
att_levels	Optional vector of attribute-levels to plot. By default, all attribute-levels are shown.
...	Additional arguments (not currently used)

Value

Plot of covariate importance scores

rf_vimp — *Estimate a Single Variable Importance Metric for* `cjbart` *Object*

Description

Estimates random forest variable importance scores for a single attribute-level of a conjoint experiment. This function is for advanced use. Users should typically use the `het_vimp()` function.

Usage

```
rf_vimp(model, outcome, covars = NULL)
```

Arguments

`model`	Object of class `cjbart`, the result of running `IMCE()`
`outcome`	Character string detailing the covariate over which to analyze heterogeneous effects
`covars`	An optional vector of covariates to include in the importance metric check. When `covars = NULL` (the default), all covariates are included in the importance model.

Value

Data.frame of variable importance scores for each covariate in the model, as well as values for the estimated 95% confidence interval for each importance score.

RMCE — *Inspect Round-Level Marginal Component Effect (RMCE)*

Description

`RMCE` calculates the round-level marginal component effects from a cjbart model.

Usage

```
RMCE(imces)
```

Arguments

`imces`	An object of class "cjbart", the result of calling the `IMCE` function

Details

The RMCE estimates are the result of averaging the OMCEs within each round, for each subject in the experiment. The RMCE is the intermediate causal quantity between OMCEs and IMCEs, and can be useful for inspecting whether there are any carryover or stability issues across rounds.

Value

IMCE returns a data frame of RMCEs.

See Also

cjbart() and IMCE()

---

summary.cjbart *Summarizing* cjbart *Marginal Component Effect Estimates*

Description

summary method for class "cjbart"

Usage

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

Arguments

object	Object of class cjbart, the result of running IMCE()
...	Further arguments (not currently used)

Value

Data frame summarizing the average marginal component effect, the minimum and maximum values, and standard deviations for each attribute-level.

Examples

```
subjects <- 5
rounds <- 2
profiles <- 2
obs <- subjects*rounds*profiles

fake_data <- data.frame(A = sample(c("a1","a2"), obs, replace = TRUE),
                        B = sample(c("b1","b2"), obs, replace = TRUE),
                        id1 = rep(1:subjects, each=rounds),
                        stringsAsFactors = TRUE)

fake_data$Y <- sample(c(0,1), obs, replace = TRUE)

cj_model <- cjbart(data = fake_data,
                   Y = "Y",
                   id = "id1")

## Skip if not Unix due to longer CPU time
```

```
if (.Platform$OS.type=='unix') {

  het_effects <- IMCE(data = fake_data,
                      model = cj_model,
                      attribs = c("A","B"),
                      ref_levels = c("a1","b1"),
                      cores = 1)

  summary(het_effects)
}
```

Index