# Package ‘vimp’

August 10, 2018

**Type** Package

**Title** Nonparametric Variable Importance

**Version** 1.1.0

**Author** Brian D. Williamson [aut, cre],
Noah Simon [aut],
Marco Carone [aut]

**Maintainer** Brian D. Williamson <brianw26@uw.edu>

**Description** Calculate point estimates of and valid confidence intervals for nonparametric variable importance measures in high and low dimensions, using flexible estimators of the underlying regression functions. For more information about the methods, please see Williamson et al. (2017) <https://biostats.bepress.com/uwbiostat/paper422/>.

**License** MIT + file LICENSE

**Depends** R (>= 3.1.0)

**Imports** SuperLearner, stats, graphics

**LazyData** TRUE

**RoxygenNote** 6.0.1

**Suggests** knitr, rmarkdown, MASS, gam, glmnet, xgboost, covr, testthat

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2018-08-09 23:00:07 UTC

## R topics documented:

---

`average_vim` *Average multiple independent importance estimates*

---

### Description

Average the output from multiple calls to `vimp_regression`, for different independent groups, into a single estimate with a corresponding standard error and confidence interval.

### Usage

```
average_vim(..., weights = rep(1/length(list(...)), length(list(...))))
```

### Arguments

| | |
|---|---|
| `...` | an arbitrary number of `vim` objects |
| `weights` | how to average the vims together, and must sum to 1; defaults to 1/(number of vims) for each vim, corresponding to the arithmetic mean |

### Value

an object of class `vim` containing the (weighted) average of the individual importance estimates, as well as the appropriate standard error and confidence interval. This results in a list containing:

- call - the call to `average_vim()`
- s - a list of the column(s) to calculate variable importance for
- SL.library - a list of the libraries of learners passed to `SuperLearner`
- full_fit - a list of the fitted values of the chosen method fit to the full data
- red_fit - a list of the fitted values of the chosen method fit to the reduced data
- est- a vector with the corrected estimates
- naive- a vector with the naive estimates
- update- a list with the influence curve-based updates
- mat - a matrix with the estimated variable importance, the standard error, and the $(1-\alpha)\times 100\%$ confidence interval
- full_mod - a list of the objects returned by the estimation procedure for the full data regression (if applicable)
- red_mod - a list of the objects returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation

### Examples

```
library(SuperLearner)
library(gam)
## generate the data
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- smooth + stats::rnorm(n, 0, 1)

## set up a library for SuperLearner
learners <- "SL.gam"

## get estimates on independent splits of the data
samp <- sample(1:n, n/2, replace = FALSE)

## using Super Learner
est_2 <- vimp_regression(Y = y[samp], X = x[samp, ], indx = 2,
           run_regression = TRUE, alpha = 0.05,
           SL.library = learners, cvControl = list(V = 10))

est_1 <- vimp_regression(Y = y[-samp], X = x[-samp, ], indx = 2,
           run_regression = TRUE, alpha = 0.05,
           SL.library = learners, cvControl = list(V = 10))

ests <- average_vim(est_1, est_2, weights = c(1/2, 1/2))
```

---

## cv_vim *Nonparametric Variable Importance Estimates using Cross-validation*

---

### Description

Compute estimates and confidence intervals for the nonparametric variable importance parameter of interest, using cross-validation in the updating procedure. This essentially involves splitting the data into V train/test splits; train the learners on the training data, evaluate importance on the test data; and average over these splits.

### Usage

```
cv_vim(Y, X, f1, f2, f3, f4, indx = 1, V = 10, folds = NULL,
  type = "regression", run_regression = TRUE, SL.library = c("SL.glmnet",
  "SL.xgboost", "SL.mean"), alpha = 0.05, na.rm = FALSE,
  update_denom = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `Y` | the outcome. |
| `X` | the covariates. |
| `f1` | the fitted values from a flexible estimation technique regressing Y on X; a list of length V, built on the training data. |
| `f2` | the fitted values from a flexible estimation technique regressing the fitted values in `f1` on X withholding the columns in `indx`; a list of length V, built on the training data. |
| `f3` | the fitted values from a flexible estimation technique regressing Y on X; a list of length V, built on the test data. |
| `f4` | the fitted values from a flexible estimation technique regressing the fitted values in `f1` on X withholding the columns in `indx`; a list of length V, built on the test data. |
| `indx` | the indices of the covariate(s) to calculate variable importance for; defaults to 1. |
| `V` | the number of folds for cross-validation, defaults to 10 |
| `folds` | the folds to use, if f1 and f2 are supplied |
| `type` | the type of parameter (e.g., ANOVA-based is `"regression"`) |
| `run_regression` | if outcome Y and covariates X are passed to `vimp_regression`, and `run_regression` is `TRUE`, then Super Learner will be used; otherwise, variable importance will be computed using the inputted fitted values. |
| `SL.library` | a character vector of learners to pass to `SuperLearner`, if `f1` and `f2` are Y and X, respectively. Defaults to `SL.glmnet`, `SL.xgboost`, and `SL.mean`. |
| `alpha` | the level to compute the confidence interval at. Defaults to 0.05, corresponding to a 95% confidence interval. |
| `na.rm` | should we remove NA's in the outcome and fitted values in computation? (defaults to `FALSE`) |
| `update_denom` | logical; was smoothing used to estimate the denominator of the parameter of interest, if standardized? (defaults to `TRUE`) |
| `...` | other arguments to the estimation tool, see "See also". |

## Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function, and the validity of the confidence intervals. In the interest of transparency, we return most of the calculations within the `vim` object. This results in a list containing:

- call - the call to `cv_vim`
- s - the column(s) to calculate variable importance for
- SL.library - the library of learners passed to `SuperLearner`
- full_fit - the fitted values of the chosen method fit to the full data (a list, for train and test data)
- red_fit - the fitted values of the chosen method fit to the reduced data (a list, for train and test data)

- est - the estimated variable importance
- naive - the naive estimator of variable importance
- naives - the naive estimator on each fold
- updates - the influence curve-based update for each fold
- se - the standard error for the estimated variable importance
- ci - the $(1 - \alpha) \times 100\%$ confidence interval for the variable importance estimate
- full_mod - the object returned by the estimation procedure for the full data regression (if applicable)
- red_mod - the object returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation

### Value

An object of class `vim`. See Details for more information.

### See Also

`SuperLearner` for specific usage of the `SuperLearner` function and package.

### Examples

```
library(SuperLearner)
library(gam)
n <- 100
p <- 2
## generate the data
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- smooth + stats::rnorm(n, 0, 1)

## set up a library for SuperLearner
learners <- c("SL.mean", "SL.gam")


## using Super Learner
est <- cv_vim(Y = y, X = x, indx = 2, V = 5,
type = "regression", run_regression = TRUE,
SL.library = learners, alpha = 0.05)
```

## format.vim — *Format a `vim` object*

### Description

Nicely formats the output from a `vim` object for printing.

### Usage

```
## S3 method for class 'vim'
format(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | the `vim` object of interest. |
| `...` | other options, see the generic `format` function. |

## merge_vim — *Merge multiple `vim` objects into one*

### Description

Take the output from multiple different calls to `vimp_regression` and merge into a single `vim` object; mostly used for plotting results.

### Usage

```
merge_vim(...)
```

### Arguments

| | |
|---|---|
| `...` | an arbitrary number of `vim` objects, separated by commas. |

### Value

an object of class `vim` containing all of the output from the individual `vim` objects. This results in a list containing:

- call - the call to `merge_vim()`
- s - a list of the column(s) to calculate variable importance for
- SL.library - a list of the libraries of learners passed to `SuperLearner`
- full_fit - a list of the fitted values of the chosen method fit to the full data
- red_fit - a list of the fitted values of the chosen method fit to the reduced data
- est- a vector with the corrected estimates

- naive- a vector with the naive estimates
- update- a list with the influence curve-based updates
- se- a vector with the standard errors
- ci- a matrix with the CIs
- mat - a matrix with the estimated variable importance, the standard errors, and the $(1 - \alpha) \times 100\%$ confidence intervals
- full_mod - a list of the objects returned by the estimation procedure for the full data regression (if applicable)
- red_mod - a list of the objects returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - a list of the levels, for confidence interval calculation

## Examples

```
library(SuperLearner)
library(gam)
## generate the data
## generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- smooth + stats::rnorm(n, 0, 1)

## set up a library for SuperLearner
learners <- "SL.gam"

## using Super Learner
est_2 <- vimp_regression(Y = y, X = x, indx = 2,
           run_regression = TRUE, alpha = 0.05,
           SL.library = learners, cvControl = list(V = 10))

est_1 <- vimp_regression(Y = y, X = x, indx = 1,
           run_regression = TRUE, alpha = 0.05,
           SL.library = learners, cvControl = list(V = 10))

ests <- merge_vim(est_1, est_2)
```

---

`onestep_based_estimator` *Estimate variable importance using a one-step estimator-based approach*

---

## Description

Compute nonparametric estimates of the variable importance parameter interpreted as the proportion of variability explained by including a group of covariates in the estimation technique.

## Usage

```
onestep_based_estimator(full, reduced, y, type = "regression",
  na.rm = FALSE)
```

## Arguments

| | |
|---|---|
| `full` | fitted values from a regression of the outcome on the full set of covariates. |
| `reduced` | fitted values from a regression of the fitted values from the full regression on the reduced set of covariates. |
| `y` | the outcome. |
| `type` | which parameter are you estimating (defaults to `regression`, for ANOVA-based variable importance)? |
| `na.rm` | logical; should NA's be removed in computation? (defaults to `FALSE`) |

## Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

## Value

The estimated variable importance for the given group of left-out covariates.

---

`plot.vim` *Plot* `vim` *objects*

---

## Description

Plot the estimates and standard errors for a set of `vim` objects.

## Usage

```
## S3 method for class 'vim'
plot(x, y, ...)
```

## Arguments

| | |
|---|---|
| `x` | an `vim` object. |
| `y` | a vector of names, given in the same order as the estimates in the `vim` object. |
| `...` | other options, see the generic `plot` function. |

---

`print.vim` *Print a `vim` object*

---

### Description

Prints out the table of estimates, confidence intervals, and standard errors for a `vim` object.

### Usage

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

### Arguments

| | |
|---|---|
| `x` | the `vim` object of interest. |
| `...` | other options, see the generic `print` function. |

---

`vimp` *vimp: Nonparametric variable importance assessment*

---

### Description

The vimp package provides one major function: vim. This function calculates an estimate of the variable importance parameter of interest developed by Williamson, Gilbert, Simon, and Carone. The parameter is defined as the additional variability in the outcome explained by including the covariates of interest in the estimating procedure.

### vimp Functions

The function vimp_regression() computes the estimates, standard error estimates, and confidence intervals for the ANOVA-based variable importance measure. It is an object of class "vim", and has its own print method.

The function merge_vim() takes the output of multiple calls to vimp_regression(), and combines the results into a single vim object.

The function format() formats a vim object for printing; print() prints the results; and plot() plots the estimates and standard errors.

## vimp_ci *Confidence intervals for variable importance*

### Description

Compute confidence intervals for the true variable importance parameter interpreted as the proportion of variability explained by including a group of covariates in the estimation technique.

### Usage

```
vimp_ci(est, se, level = 0.95)
```

### Arguments

| | |
|---|---|
| `est` | estimate of variable importance from a call to `variableImportance`. |
| `se` | estimate of the standard error of `est`, from a call to `variableImportanceSE` |
| `level` | confidence interval type (defaults to 0.95). |

### Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

### Value

The Wald-based confidence interval for the true importance of the given group of left-out covariates.

## vimp_regression *Nonparametric Variable Importance Estimates*

### Description

Compute estimates of and confidence intervals for nonparametric ANOVA-based variable importance.

### Usage

```
vimp_regression(Y, X, f1 = NULL, f2 = NULL, indx = 1,
  run_regression = TRUE, SL.library = c("SL.glmnet", "SL.xgboost",
  "SL.mean"), alpha = 0.05, na.rm = FALSE, ...)
```

### Arguments

| | |
|---|---|
| Y | the outcome |
| X | the covariates |
| f1 | the fitted values from a flexible estimation technique regressing Y on X. |
| f2 | the fitted values from a flexible estimation technique regressing the fitted values in f1 on X withholding the columns in indx. |
| indx | the indices of the covariate(s) to calculate variable importance for; defaults to 1. |
| run_regression | if outcome Y and covariates X are passed to vimp_regression, and run_regression is TRUE, then Super Learner will be used; otherwise, variable importance will be computed using the inputted fitted values. |
| SL.library | a character vector of learners to pass to SuperLearner, if f1 and f2 are Y and X, respectively. Defaults to SL.glmnet, SL.xgboost, and SL.mean. |
| alpha | the level to compute the confidence interval at. Defaults to 0.05, corresponding to a 95% confidence interval. |
| na.rm | should we remove NA's in the outcome and fitted values in computation? (defaults to FALSE) |
| ... | other arguments to the estimation tool, see "See also". |

### Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function, and the validity of the confidence intervals. In the interest of transparency, we return most of the calculations within the vim object. This results in a list containing:

- call - the call to vim
- s - the column(s) to calculate variable importance for
- SL.library - the library of learners passed to SuperLearner
- full_fit - the fitted values of the chosen method fit to the full data
- red_fit - the fitted values of the chosen method fit to the reduced data
- est - the estimated variable importance
- naive - the naive estimator of variable importance
- update - the influence curve-based update
- se - the standard error for the estimated variable importance
- ci - the $(1 - \alpha) \times 100\%$ confidence interval for the variable importance estimate
- full_mod - the object returned by the estimation procedure for the full data regression (if applicable)
- red_mod - the object returned by the estimation procedure for the reduced data regression (if applicable)
- alpha - the level, for confidence interval calculation

### Value

An object of classes vim and vim_regression. See Details for more information.

## See Also

SuperLearner for specific usage of the SuperLearner function and package.

## Examples

```
library(SuperLearner)
library(gam)
## generate the data
## generate X
p <- 2
n <- 100
x <- data.frame(replicate(p, stats::runif(n, -5, 5)))

## apply the function to the x's
smooth <- (x[,1]/5)^2*(x[,1]+7)/5 + (x[,2]/3)^2

## generate Y ~ Normal (smooth, 1)
y <- smooth + stats::rnorm(n, 0, 1)

## set up a library for SuperLearner
learners <- "SL.gam"

## using Y and X
est <- vimp_regression(y, x, indx = 2,
           alpha = 0.05, run_regression = TRUE,
           SL.library = learners, cvControl = list(V = 10))

## using pre-computed fitted values
full <- SuperLearner(Y = y, X = x,
SL.library = learners, cvControl = list(V = 10))
full.fit <- predict(full)$pred
reduced <- SuperLearner(Y = full.fit, X = x[, 2, drop = FALSE],
SL.library = learners, cvControl = list(V = 10))
red.fit <- predict(reduced)$pred

est <- vimp_regression(Y = y, f1 = full.fit, f2 = red.fit,
            indx = 2, run_regression = FALSE, alpha = 0.05)
```

---

`vimp_se` *Estimate standard errors*

---

## Description

Compute standard error estimates for estimates of the variable importance parameter interpreted as the proportion of variability explained by including a group of covariates in the estimation technique.

### Usage

```
vimp_se(update, n = length(update), na.rm = FALSE)
```

### Arguments

| | |
|---|---|
| `update` | the influence curve-based update |
| `n` | the sample size |
| `na.rm` | logical; should NA's be removed in computation? (defaults to `FALSE`) |

### Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

### Value

The standard error for the estimated variable importance for the given group of left-out covariates.

---

## `vimp_update` *Estimate the influence function*

---

### Description

Compute the value of the influence function for the given group of left-out covariates.

### Usage

```
vimp_update(full, reduced, y, type = "regression", na.rm = FALSE)
```

### Arguments

| | |
|---|---|
| `full` | fitted values from a regression of the outcome on the full set of covariates. |
| `reduced` | fitted values from a regression either (1) of the outcome on the reduced set of covariates, or (2) of the fitted values from the full regression on the reduced set of covariates. |
| `y` | the outcome. |
| `type` | which parameter are you estimating (defaults to `regression`, for ANOVA-based variable importance)? |
| `na.rm` | logical; should NAs be removed in computation? (defaults to `FALSE`) |

### Details

See the paper by Williamson, Gilbert, Simon, and Carone for more details on the mathematics behind this function and the definition of the parameter of interest.

### Value

The influence function values for the given group of left-out covariates.

# Index