Package 'domir'

July 11, 2022

Title Tools to Support Relative Importance Analysis

Version 1.0.0

Date 2022-7-3

Description Methods to apply decomposition-based relative importance analysis for R functions. This package supports the application of decomposition methods by providing 'lapply'- or 'Map'-like meta-functions that compute dominance analysis (Azen, R., & Budescu, D. V. (2003) <doi:10.1037/1082-989X.8.2.129>; Grömping, U. (2007) <doi:10.1198/000313007X188252>) or Shapley value regression (Lipovetsky, S., & Conklin, M. (2001) <doi:10.1002/asmb.446>) based on the values returned from other functions.

Imports stats, utils

Suggests dplyr, forcats, Formula, ggplot2, knitr, parameters, performance, purrr, relaimpo, rlang, rmarkdown, stringr, testthat (>= 3.0.0), tidyr

License GPL (>= 3)

URL https://github.com/jluchman/domir

BugReports https://github.com/jluchman/domir/issues

Encoding UTF-8

RoxygenNote 7.2.0

Config/testthat/edition 3

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Joseph Luchman [aut, cre] (<https://orcid.org/0000-0002-8886-9717>)

Maintainer Joseph Luchman <jluchman@gmail.com>

Repository CRAN

Date/Publication 2022-07-11 07:40:16 UTC

R topics documented:

`domir-package` *Tools to Support Relative Importance Analysis*

Description

Methods to apply decomposition-based relative importance analysis for R functions.

Details

Determining the relative importance of inputs to (i.e., independent variables, predictors, features) to a predictive model is topic of interest to scientists and analysts. Decomposing a returned value, such as a model fit metric or statistic, into parts attributable to each input is a commonly applied method for determining relative importance in predictive models.

This package supports applying decomposition methods using `lapply`- or `Map`-like functions that compute dominance analysis (Azen & Budescu, 2004; Budescu, 1993)/Shapley value decomposition (Grömping, 2007; Lipovetsky & Conklin, 2001) based on the values returned from other, predictive modeling, functions.

The user interface is structured such that domir automates the decomposition of the returned value and comparisons between model inputs and the user provides the analysis pipeline including model inputs, the predictive modeling function into which they are entered, and returned value from the model to decompose.

This package's user interface accepts inputs as names on the right hand side of a `formula` which can be passed on to the predictive model directly or further processed in the analysis pipeline. The interface is also planned to be extended to `Formula` from the package `{Formula}` as well as `list` types as inputs.

Author(s)

Joseph Luchman `<jluchman@gmail.com>`

References

- Azen, R., & Budescu, D. V. (2003). The dominance analysis approach for comparing predictors in multiple regression. Psychological Methods, 8(2), 129-148. doi:10.1037/1082-989X.8.2.129

- Budescu, D. V. (1993). Dominance analysis: A new approach to the problem of relative importance of predictors in multiple regression. Psychological Bulletin, 114(3), 542-551. doi:10.1037/0033-2909.114.3.542
- Grömping, U. (2007). Estimators of relative importance in linear regression based on variance decomposition. The American Statistician, 61(2), 139-147. doi:10.1198/000313007X188252
- Lipovetsky, S, & and Conklin, M. (2001). Analysis of regression in game theory approach. Applied Stochastic Models in Business and Industry, 17(4), 319-330. doi:10.1002/asmb.446

`domin` *Dominance analysis supporting* `formula`*-based modeling functions*

Description

Computes dominance statistics for predictive modeling functions that accept a `formula`.

Usage

```
domin(
  formula_overall,
  reg,
  fitstat,
  sets = NULL,
  all = NULL,
  conditional = TRUE,
  complete = TRUE,
  consmodel = NULL,
  reverse = FALSE,
  ...
)
```

Arguments

`formula_overall`
: An object of class `formula` or that can be coerced to class `formula` for use in the modeling function in `reg`. The `terms` on the right hand side of this formula are used as separate entries to the dominance analysis.

  A valid `formula_overall` entry is necessary, even if only submitting entries in `sets`, to define a valid left hand side of the prediction equation (see examples). The function called in `reg` must accept one or more responses on the left hand side.

`reg`
: A function implementing the predictive (or "reg"ression) model called.

  String function names (e.g., "lm"), function names (e.g., `lm`), or anonymous functions (e.g., `function(x) lm(x)`) are acceptable entries. This argument's contents are passed to `do.call` and thus any function call `do.call` would accept is valid.

  The predictive model in `reg` must accept a `formula` object as its first argument or must be adapted to do so with a wrapper function.

`fitstat` List providing arguments to call a fit statistic extracting function (see details). The `fitstat` list must be of at least length two.

The first element of `fitstat` must be a function implementing the fit statistic extraction. String function names (e.g., "summary"), function names (e.g., `summary`), or anonymous functions (e.g., `function(x) summary(x)`) are acceptable entries. This element's contents are passed to `do.call` and thus any function call `do.call` would accept is valid.

The second element of `fitstat` must be the named element of the list or vector produced by the fit extractor function called in the first element of `fitstat`. This element must be a string (e.g., "r.squared").

All list elements beyond the second are submitted as additional arguments to the fit extractor function call.

The fit statistic extractor function in the first list element of `fitstat` must accept the model object produced by the predictive modeling function in `reg` as its first argument or be adapted to do so with a wrapper function.

The fit statistic produced must be scalar valued (i.e., vector of length 1).

`sets` A list with each element comprised of vectors containing variable/factor names or `formula` coercible strings.

Each separate list element-vector in `sets` is concatenated (when the list element-vector is of length > 1) and used as an entry to the dominance analysis along with the terms in `formula_overall`.

`all` A vector of variable/factor names or `formula` coercible strings. The entries in this vector are concatenated (when of length > 1) but are not used in the dominance analysis. Rather the value of the fit statistic associated with these terms is removed from the dominance analysis; this vector is used like a set of covariates.

The entries in `all` are removed from and considered an additional component that explains the fit metric. As a result, the general dominance statistics will no longer sum to the overall fit metric and the standardized vector will no longer sum to 1.

`conditional` Logical. If `FALSE` then conditional dominance matrix is not computed.

If conditional dominance is not desired as an importance criterion, avoiding computing the conditional dominance matrix can save computation time.

`complete` Logical. If `FALSE` then complete dominance matrix is not computed.

If complete dominance is not desired as an importance criterion, avoiding computing complete dominance designations can save computation time.

`consmodel` A vector of variable/factor names, `formula` coercible strings, or other formula terms (i.e., 1 to indicate an intercept). The entries in this vector are concatenated (when of length > 1) and, like the entries of `all`, are not used in the dominance analysis; this vector is used as an adjustment to the baseline value of the overall fit statistic.

The use of `consmodel` changes the interpretation of the the general and conditional dominance statistics. When `consmodel` is used, the general and conditional dominance statistics are reflect the difference between the constant model and the overall fit statistic values.

Typical usage of `consmodel` is to pass "1" to set the intercept as the baseline and control for its value when the baseline model's fit statistic value is not 0 (e.g., if using the AIC or BIC as a fit statistic; see examples).

As such, this vector is used to set a baseline for the fit statistic when it is non-0.

`reverse` Logical. If `TRUE` then standardized vector, ranks, and complete dominance Designations are reversed in their interpretation.

This argument should be changed to `TRUE` if the fit statistic used decreases with better fit to the data (e.g., AIC, BIC).

`...` Additional arguments passed to the function call in the `reg` argument.

Details

`domin` automates the computation of all possible combination of entries to the dominance analysis (DA), the creation of `formula` objects based on those entries, the modeling calls/fit statistic capture, and the computation of all the dominance statistics for the user.

`domin` accepts only a "deconstructed" set of inputs and "reconstructs" them prior to formulating a coherent predictive modeling call.

One specific instance of this deconstruction is in generating the number of entries to the DA. The number of entries is taken as all the `terms` from `formula_overall` and the separate list element vectors from `sets`. The entries themselves are concatenated into a single formula, combined with the entries in `all`, and submitted to the predictive modeling function in `reg`. Each different combination of entries to the DA forms a different `formula` and thus a different model to estimate.

For example, consider this `domin` call:

```
domin(y ~ x1 + x2, lm, list(summary, "r.squared"), sets = list(c("x3", "x4")), all = c("c1",
"c2"), data = mydata))
```

This call records three entries and results in seven (i.e., $2^3 - 1$) different combinations:

1. x1
2. x2
3. x3, x4
4. x1, x2
5. x1, x3, x4
6. x2, x3, x4
7. x1, x2, x3, x4

`domin` parses `formula_overall` to obtain all the terms in it and combines them with `sets`. When parsing `formula_overall`, only the processing that is available in the `stats` package is applied. Note that `domin` is not programmed to process terms of order > 1 (i.e., interactions/products) appropriately (i.e., only include in the presence of lower order component terms).

From these combinations, the predictive models are constructed and called. The predictive model call includes the entries in `all`, applies the appropriate formula, and reconstructs the function itself. The seven combinations above imply the following series of predictive model calls:

1. `lm(y ~ x1 + c1 + c2, data = mydata)`

2. `lm(y ~ x2 + c1 + c2, data = mydata)`
3. `lm(y ~ x3 + x4 + c1 + c2, data = mydata)`
4. `lm(y ~ x1 + x2 + c1 + c2, data = mydata)`
5. `lm(y ~ x1 + x3 + x4 + c1 + c2, data = mydata)`
6. `lm(y ~ x2 + x3 + x4 + c1 + c2, data = mydata)`
7. `lm(y ~ x1 + x2 + x3 + x4 + c1 + c2, data = mydata)`

It is possible to use a `domin` with only sets (i.e., no IVs in `formula_overall`; see examples below). There must be at least two entries to the DA for `domin` to run.

All the called predictive models are submitted to the fit extractor function implied by the entries in `fitstat`. Again applying the example above, all seven predictive models' objects would be individually passed as follows:

```
summary(lm_obj)["r.squared"]
```

where `lm_obj` is the model object returned by `lm`.

The entries to `fitstat` must be as a list and follow a specific structure: `list(fit_function, element_name, ...)`

`fit_function` First element and function to be applied to the object produced by the `reg` function

`element_name` Second element and name of the element from the object returned by `fit_function` to be used as a fit statistic. The fit statistic must be scalar-valued/length 1

`...` Subsequent elements and are additional arguments passed to `fit_function`

In the case that the model object returned by `reg` includes its own fit statistic without the need for an extractor function, the user can apply an anonymous function following the required format to extract it.

Value

Returns an object of `class` "domin". An object of class "domin" is a list composed of the following elements:

`General_Dominance` Vector of general dominance statistics.

`Standardized` Vector of general dominance statistics normalized to sum to 1.

`Ranks` Vector of ranks applied to the general dominance statistics.

`Conditional_Dominance` Matrix of conditional dominance statistics. Each row represents a term; each column represents an order of terms.

`Complete_Dominance` Logical matrix of complete dominance designations. The term represented in each row indicates dominance status; the terms represented in each columns indicates dominated-by status.

`Fit_Statistic_Overall` Value of fit statistic for the full model.

`Fit_Statistic_All_Subsets` Value of fit statistic associated with terms in `all`.

`Fit_Statistic_Constant_Model` Value of fit statistic associated with terms in `consmodel`.

`Call` The matched call.

`Subset_Details` List containing the full model and descriptions of terms in the full model by source.

Notes

domin has been superseded by domir.

Examples

```
## Basic linear model with r-square

domin(mpg ~ am + vs + cyl,
  lm,
  list("summary", "r.squared"),
  data = mtcars)


## Linear model including sets

domin(mpg ~ am + vs + cyl,
  lm,
  list("summary", "r.squared"),
  data = mtcars,
  sets = list(c("carb", "gear"), c("disp", "wt")))


## Multivariate linear model with custom multivariate r-square function
## and all subsets variable

Rxy <- function(obj, names, data) {
   return(list("r2" = cancor(predict(obj),
       as.data.frame(mget(names, as.environment(data))))[["cor"]][1]^2))
       }

domin(cbind(wt, mpg) ~ vs + cyl + am,
  lm,
  list("Rxy", "r2", c("mpg", "wt"), mtcars),
  data = mtcars,
  all = c("carb"))


## Sets only

domin(mpg ~ 1,
  lm,
  list("summary", "r.squared"),
  data = mtcars,
  sets = list(c("am", "vs"), c("cyl", "disp"), c("qsec", "carb")))

## Constant model using AIC

domin(mpg ~ am + carb + cyl,
  lm,
  list(function(x) list(aic = extractAIC(x)[[2]]), "aic"),
  data = mtcars,
  reverse = TRUE, consmodel = "1")
```

domir *Dominance analysis methods*

Description

Parses input object to obtain valid elements, determines all required combinations/subsets of elements (depends on input type), submits subsets to a function, and computes dominance decomposition statistics based on the returned values from the function.

Usage

```
domir(.obj, ...)

## S3 method for class 'formula'
domir(
  .obj,
  .fct,
  .set = NULL,
  .wst = NULL,
  .all = NULL,
  .adj = NULL,
  .cdl = TRUE,
  .cpt = TRUE,
  .rev = FALSE,
  ...
)

## S3 method for class 'Formula'
domir(...)

## S3 method for class 'list'
domir(...)
```

Arguments

.obj	A formula, Formula, or list. Parsed to produce subsets of elements to submit to .fct. Always submits subsets of .obj the same type to .fct as the same type or class as the input object.
...	Passes arguments to other methods; passes arguments to function in .fct.
.fct	A function/closure or string function name. Applied to all subsets of elements as received from .obj. Must return a length 1 (scalar), numeric, atomic vector.
.set	A list. Must be comprised of elements of the same type or class as .obj. Elements of the list can be named.

.wst	Not yet used.
.all	A formula, Formula, or list. Must be the same type or class as .obj.
.adj	A formula, Formula, or list. Must be the same type or class as .obj.
.cdl	Logical. If FALSE then conditional dominance matrix is not computed and method to produce general dominance statistics changes.
.cpt	Logical. If FALSE then complete dominance matrix is not computed.
.rev	Logical. If TRUE then standardized vector, ranks, and complete dominance designations are reversed in their interpretation.

Details

Element Parsing:

.objs elements are parsed and used to determine the required number of subsets included the dominance analysis. How the elements are parsed is determined depends on .objs type or class method.

formula*:*
The formula method applies the standard terms function parsing which separates term names on the right hand side of the formula. All terms separated by + are considered a separate element for generating subsets.
Any terms on the left hand side of .obj are retained and passed through to all subsets.

Formula *and* list*:*
The Formula and list methods are not yet implemented.

Additional Details:
By default, each parsed element in .obj will be used as a separate element to generate subsets and will obtain a separate contribution to the returned value.

Changing Element Parsing:

All methods' default behavior of using all elements to generate subsets can be overriden using .set, .all, and .adj arguments.

Elements in .set, .all, and .adj must also be present in .obj. The entries in three arguments change .objs parsing behavior but still depend on .obj as the primary input object.

.set*:*
.set binds together elements in .obj such that they form a single new element. The elements in .obj bound together contribute jointly to the returned value.
If elements in .set are named, the .set element's name will be used in the "domir" object returned and all printed results.
The formula method for .set does not allow any element to have a left hand side.

.all*:*
.all binds elements in .obj to all subsets. The elements in .obj bound together by .all are given precedence in ascribing the returned value and contribute jointly to Value_All. Value_All

is determined prior to conducting the dominance analysis and its value is removed from the returned values for all subsets.
The `formula` method for `.all` does not allow a left hand side.

`.adj`*:*
`.adj` binds elements in `.obj` to all subsets. The elements in `.obj` bound together by `.adj` are considered external to the dominance analysis but are adjusted for given they affect the returned value. Elements in `.adj` contribute jointly to `Value_Adjust` and have a higher precedence than those bound in `.all`. `Value_Adjust` is determined prior to conducting the dominance analysis and its value is removed from the returned values for all subsets as well as from `Value_All`.
The `formula` method for `.adj` does not allow a left hand side but allows constants/intercepts (i.e., `~ 1`) to be included as a valid element in the right hand size even when not explicitly included in `.obj`.

Additional Details:
All element parsing methods for `domir` will submit subsets generated as an object of the same type as `.obj` (i.e., a `formula` in `.obj` will be submitted as a `formula`) to the `.fct` as the first, unnamed argument.

`.fct` **as Analysis Pipeline:**

The function in `.fct` will be called repeatedly, once for each subset of elements created from `.obj`.

`.fct` is expected to be a complete analysis pipeline that receives a subset of elements from `.obj`, uses the subset of elements from `.obj` in the type/class as received to generate a predictive model, and extracts a returned value of the appropriate type to dominance analyze.

At current, only atomic (i.e., non-`list`), numeric scalars (i.e., vectors of length 1) are allowed as returned values.

`domir` is intended to be strict about input and output requirements for functions in `.fct` and applies a series of checks to ensure the input and output adhere to these requirements. In most circumstances, the user will have to make their own named or anonymous function to supply to `.fct` to meet `domir`s requirements.

Value

Returns an object of `class` "domir" which is a composed of the following elements:

`General_Dominance` Vector of general dominance values.

`Standardized` Vector of general dominance values normalized to sum to 1.

`Ranks` Vector of ranks applied to the general dominance values.

`Conditional_Dominance` Matrix of conditional dominance values. Each row represents an element in `.obj`; each column represents a number of elements from `.obj` in a subset.

`Complete_Dominance` Logical matrix of complete dominance designations. The `.obj` elements represented in each row indicates dominance status; the `.obj` elements represented in each column indicates dominated-by status.

`Value` Value returned by `.fct` with all elements (i.e., from `.obj`, `.all`, and `.adj`.

`Value_All` Value of `.fct` associated with elements included in `.all`; when elements are in `.adj`, will be adjusted for `Value_Adjust`.

`Value_Adjust` Value of `.fct` associated with elements in `.adj`.

`Call` The matched call.

Examples

```
## Basic linear model with r-square

lm_r2 <- function(fml, data) {
  lm_res <- lm(fml, data = data)
  r2 <- summary(lm_res)[["r.squared"]]
  return(r2) }

domir(mpg ~ am + vs + cyl,
  lm_r2,
  data = mtcars)


## Linear model including set

domir(mpg ~ am + vs + cyl + carb + gear + disp + wt,
 lm_r2,
 .set = list(~ carb + gear, ~ disp + wt),
 data = mtcars)


## Multivariate linear model with multivariate r-square
## and all subsets variable

mlm_rxy <- function(fml, data, dvnames) {
  mlm_res <- lm(fml, data = data)
  mlm_pred <- predict(mlm_res)
  mlm_rxy <- cancor(mlm_pred, data[dvnames])$cor[[1]]^2
  return(mlm_rxy)
  }

domir(cbind(wt, mpg) ~ vs + cyl + am + carb,
  mlm_rxy,
  .all = ~ carb,
  data = mtcars,
  dvnames = c("wt", "mpg"))


## Named sets

domir(mpg ~ am + gear + cyl + vs + qsec + drat,
  lm_r2,
  data = mtcars,
  .set = list(trns = ~ am + gear,
    eng = ~ cyl + vs, misc = ~ qsec + drat))

## Linear model using AIC

lm_aic <- function(fml, data) {
  lm_res <- lm(fml, data = data)
  aic <- AIC(lm_res)
  return(aic) }
```

```
domir(mpg ~ am + carb + cyl,
  lm_aic,
  .adj = ~ 1,
  .rev = TRUE,
  data = mtcars)
```

`print.domin` *Print method for* `domin`

Description

Reports formatted results from `domin` class object.

Usage

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

Arguments

`x`	an object of class "domin".
`...`	further arguments passed to or from other methods. Not used currently.

Details

The print method for class `domin` objects reports out the following results:

- Fit statistic for the full model. The fit statistic for the all subsets model is reported here if there are any entries in `all`. The fit statistic for the constant model is reported here if there are any entries in `consmodel`.
- Matrix describing general dominance statistics, standardized general dominance statistics, and the ranking of the general dominance statistics
- If `conditional` is `TRUE`, matrix describing the conditional dominance designations
- If `complete` is `TRUE`, matrix describing the complete dominance designations
- If following `summary.domin`, matrix describing the strongest dominance designations between all independent variables
- If there are entries in `sets` and/or `all` the terms included in each set as well as the terms in all subsets are reported

The `domin` print method alters dimension names for readability and they do not display as stored in the original `domin` object.

Value

The "domin" object with altered column and row names for conditional and complete dominance results as displayed in the console.

`print.domir` *Print method for* `domir`

Description

Reports formatted results from `domir` class object.

Usage

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

Arguments

`x`	an object of class "domir".
`...`	further arguments passed to `print.default`.

Details

The print method for class `domir` objects reports out the following results:

- Value when all elements are included in `obj`.
- Value for the elements included in `.all` if any.
- Value for the elements included in `.adj` if any.
- Matrix describing general dominance values, standardized general dominance values, and the ranking of the general dominance values.
- Matrix describing the conditional dominance values; when computed
- Matrix describing the complete dominance designations; when determined
- If following `summary.domir`, matrix describing the strongest dominance designations between all elements.

The `domir` print method alters dimension names for readability and they do not display as stored in the `domir` object.

Value

The submitted "domir" object, invisibly.

summary.domin *Summary method for* domin

Description

Reports dominance designation results from the domin class object.

Usage

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

Arguments

object	an object of class "domin".
...	further arguments passed to or from other methods. Not used currently.

Details

The summary method for class domin is used for obtaining the strongest dominance designations (i.e., general, conditional, or complete) among the independent variables.

Value

The originally submitted "domin" object with an additional Strongest_Dominance element added.

Strongest_Dominance Matrix comparing the independent variable in the first row to the independent variable in the third row. The second row denotes the strongest designation between the two independent variables.

summary.domir *Summary method for* domir

Description

Reports dominance designation results from the domir class object.

Usage

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

Arguments

object	an object of class "domir".
...	further arguments passed to or from other methods. Not used currently.

Details

The summary method for class `domir` is used for obtaining the strongest dominance designations (i.e., general, conditional, or complete) among the elements.

Value

The submitted "domir" object with an additional `Strongest_Dominance` element added.

`Strongest_Dominance` Matrix comparing the element in the first row to the element in the third row. The second row denotes the strongest designation between the two elements.

Index