Package 'domir'

July 22, 2025

Title Tools to Support Relative Importance Analysis

Version 1.2.0

Date 2024-5-4

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>) an extension of Shapley value regression (Lipovetsky, S., & Conklin, M. (2001) <doi:10.1002/asmb.446>) based on the values returned from other functions.

Imports parallel, stats, utils

Suggests dplyr, dominanceanalysis, forcats, Formula, ggplot2, knitr, lme4, parameters, performance, pscl, purrr, relaimpo, rlang, rmarkdown, stringr, systemfit, testthat (>= 3.0.0), tidyr

License GPL (>= 3)

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

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

Encoding UTF-8

RoxygenNote 7.3.1

Config/testthat/edition 3

VignetteBuilder knitr

Language en-US

NeedsCompilation no

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

Maintainer Joseph Luchman <jluchman@gmail.com>

Repository CRAN

Date/Publication 2024-05-04 23:20:02 UTC

Contents

`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). domin also does not allow offset 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` is an R port of the Stata command with the same name (see Luchman, 2021).

`domin` has been superseded by `domir`.

References

Luchman, J. N. (2021). Relative importance analysis in Stata using dominance analysis: domin and domme. The Stata Journal, 21, 2. doi: 10.1177/1536867X211025837.

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 list of names, determines all required combinations of subsets of the name list, submits name list subsets to a function as the input type, 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 = FALSE,
  .cdl = TRUE,
  .cpt = TRUE,
  .rev = FALSE,
  .cst = NULL,
  .prg = FALSE,
  ...
)
```

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

Arguments

.obj	A formula or formula_list. Parsed to produce list of names. Combinations of subsets the name list are sapply-ed to .fct. The name list subsets submitted to .fct are formatted to be of the same class as .obj and are submitted to .fct as the first, unnamed argument.
...	Passes arguments to other methods during method dispatch; passes arguments to the function in .fct during function execution.
.fct	A function 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 class as .obj. Elements of the list can be named.
.wst	Not yet used.
.all	A formula or formula_list. Must be the same class as .obj.
.adj	Logical. If TRUE then a model including only an intercept is submitted to .fct and the value returned is subtracted from the values returned from all subsets in the dominance analysis.
.cdl	Logical. If FALSE then conditional dominance matrix is not computed.
.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.

`.cst` Object of class c("SOCKcluster", "cluster") from `parallel-package`.
When non-`NULL`, will alter the method for collecting values from all combinations of names from using `sapply` to `parallel::parSapply`.

`.prg` Logical.
If `TRUE` then a progress bar is displayed during collection of values to indicate progress.

Details

Element Parsing:

`.objs` is parsed into a name list that is used to determine the required number of combinations of subsets of the name list included the dominance analysis. How the name list is obtained depends on `.obj`'s class.

`formula`*:*
The `formula` creates a name list using all terms in the formula. The terms are obtained using `terms.formula`. All processing that is normally applied to the right hand side of a formula is implemented (see `formula`).
A response/left hand side is not required but, if present, is included in all `formulas` passed to `.fct`.

`formula_list`*:*
The `formula_list` creates a name list out of response-term pairs. The terms are obtained using `terms.formula` applied to each individual formula in the list.

Additional Details:
By default, names obtained from `.obj` are all considered separate 'value-generating names' with the same priority. Each value-generating name will be a separate element when computing combination subsets and will be compared to all other value-generating names.
`formulas` and `formula_list` elements are assumed to have an intercept except if explicitly removed with a `- 1` in the `formula`(s) in `.obj`. If removed, the intercept will be removed in all `formula`(s) in each `sapply`-ed subset to `.fct`.
If `offset`s are included, they are passed, like intercepts, while `sapply`-ing subsets to `.fct`.

Changing Element Parsing:

All methods' default behavior that considers all value-generating names to be of equal priority can be overriden using `.set` and `.all` arguments.

Names in `.set` and `.all` must also be present in `.obj`.

`.set`*:*
`.set` binds together value-generating names such that they are of equal priority and are never separated when submitted to `.fct`. Thus, the elements in `.set` bound together contribute jointly to the returned value and are considered, effectively, a single value-generating name.
If list elements in `.set` are named, this name will be used in all returned results as the name of the set of value-generating names bound together.
`.set` thus considers the value-generating names an 'inseparable set' in the dominance analysis and are always included or excluded together.

`.all`*:*
`.all` gives immediate priority to value-generating names. The value-generating names in `.all` are bound together, are ascribed their full amount of the returned value from `.fct`, and are not adjusted for contribution of other value-generating names.

The value of .fct ascribed to the value-generating names bound together in .all is returned separately from, and not directly compared to, the other value-generating names.
The formula method for .all does not allowthe submitted formula to have a left hand side.
.all includes the value-generating names in 'all subsets' submitted to the dominance analysis which effectively removes the value associated with this set of names.

.adj*:*
.adj indicates that an intercept-only model should be supplied to .fct. This intercept-only subset is given most immediate priority and the value of .fct ascribed to it is removed from all other value-generating names and sets including those in .all.
The formula method will submit an intercept-only formula to .fct. The formula_list method creates a separate, intercept-only subset for each of the formulas in the list. Both the formula and formula_list methods will respect the user's removal of an intercept and or inclusion of an offset.
.adj then 'adjusts' the returned value for a non-0 value-returning null model when no value generating names are included. This is often useful when a predictive model's fit metric is not 0 when no predictive factors are included in the model.

Additional Details:
All methods submit combinations of names as an object of the same class as .obj. A formula in .obj will submit all combinations of names as formulas to .fct. A formula_list in .obj will submit all combinations of subsets of names as formula_lists to .fct. In the case that .fct requires a different class (e.g., a character vector of names, a Formula::Formula see fmllst2Fml) the subsets of names will have to be processed in .fct to obtain the correct class. The all subsets of names will be submitted to .fct as the first, unnamed argument.

.fct as Analysis Pipeline:

.fct is expected to be a complete analysis pipeline that receives a subset of names of the same class as .obj and uses these names in the class as submitted to generate a returned value of the appropriate type to dominance analyze. Typically, the returned value is a scalar fit statistic/metric extracted from a predictive model.

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

The .fct argument is strict about names submitted and returned value requirements for functions used. A series of checks to ensure the submitted names and returned value adhere to these requirements. The checks include whether the .obj can be submitted to .fct without producing an error and whether the returned value from .fct is a length 1, atomic, numeric vector. In most circumstances, the user will have to make their own named or anonymous function to supply as .fct to satisfy the checks.

Value

Returns an object of class "domir" composed of:

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` Matrix of proportions of subsets where the name in the row has a larger value than the name in the column. The se proportions determine complete dominance when a value of 1 or 0.

`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.

Notes

`formula` **method:**

Prior to version 1.1.0, the `formula` method allowed a `formula` to be submitted to `.adj`. Submitting an intercept-only `formula` as opposed to a logical has been depreciated and submitting a `formula` with more than an intercept is defunct.

The `formula` and `formula_list` methods can be used to pass responses, intercepts, and `offsets` to all combinations of names. If the user seeks to include other model components integral to estimation (i.e., a random effect term in `lme4::glmer()`) include them as `update` to the submitted `formula` or `formula_list` imbedded in `.fct`.

Second-order or higher terms (i.e., interactions like `~ a*b`) are parsed by default but not used differently from first-order terms for producing subsets. The values ascribed to such terms may not be valid unless the user ensures that second-order and higher terms are used appropriately in `.fct`.

Examples

```
## Linear model returning r-square
lm_r2 <-
  function(fml, data) {
    lm_res <- lm(fml, data = data)
    summary(lm_res)[["r.squared"]]
 }

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 regression 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)
    cancor(mlm_pred, data[dvnames])$cor[[1]]^2
  }

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 returning AIC
lm_aic <-
  function(fml, data) {
    lm_res <- lm(fml, data = data)
    AIC(lm_res)
 }

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


## 'systemfit' with 'formula_list' method returning AIC
if (requireNamespace("systemfit", quietly = TRUE)) {
  domir(
    formula_list(mpg ~ am + cyl + carb, qsec ~ wt + cyl + carb),
    function(fml) {
      res <- systemfit::systemfit(fml, data = mtcars)
      AIC(res)
    },
    .adj = TRUE, .rev = TRUE
```

```
  )
}
```

`fmllst2Fml` *Translate* `formula_list` *into* `Formula::Formula`

Description

Translates `formula_list` objects into a `Formula::Formula`

Usage

```
fmllst2Fml(fmllst, drop_lhs = NULL)
```

Arguments

`fmllst`	A `formula_list` classed object.
`drop_lhs`	An integer vector. Used as a selection vector to remove left hand side names prior to generating the `Formula` object. This vector must be composed of integers (e.g., 1L and not 1). This is useful for some `Formulas` that do not have a separate LHS for each LHS model part (e.g., `pscl::zeroinfl`) but are required to have separte LHS parts by `formula_list`.

Value

A `Formula::Formula` object.

`formula_list` *A* `list` *composed of* `formulas`

Description

Defines a list object composed of `formulas`. The purpose of this class of object is to impose structure of the list to ensure that it can be used to obtain RHS-LHS pairs and will be able to be parsed in `domir`.

Usage

```
formula_list(...)
```

Arguments

`...`	`formulas`, possibly named

Details

The formula_list requires that each element of the list is a formula and that each formula is unique with a different, non-NULL dependent variable/response.

Value

A list of class formula_list.

---

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, if computed
- Matrix describing the complete dominance designations, if evaluated
- 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` objects is used for obtaining the strongest dominance designations (i.e., general, conditional, or complete) among all pairs of dominance analyzed 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