Package 'domir'

September 27, 2021

Title Tools to Support Relative Importance Analysis

Version 0.2.0

Date 2021-9-27

Description Provides tools that support relative importance analysis focusing on dominance analysis. Dominance analysis is a methodology for determining the relative importance of predictors/features/independent variables (Azen, R., & Budescu, D. V. (2003) <doi:10.1037/1082-989X.8.2.129>; Groemping, U. (2007) <doi:10.1198/000313007X188252>) as well as parameter estimates (Luchman, J. N, Lei, X., & Kaplan, S. (2020) <doi:10.47263/JASEM.4(2)02>). These tools are intended to extend relative importance analysis to, effectively, any statistical or machine learning function as defined or desired by the user-especially those where the user wants to use custom importance/fit statistic or modeling function.

Imports methods, stats, utils

Suggests dominanceanalysis, testthat (>= 3.0.0), relaimpo, yhat, knitr, rmarkdown

License GPL (>= 3)

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

Encoding UTF-8

RoxygenNote 7.1.2

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 2021-09-27 14:30:02 UTC

R topics documented:

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

Description

The `domir` package provides flexible wrapper and helper functions for conducting relative importance analysis with a focus on dominance analysis. The intention of this package is to provide tools that allow relative importance analysis across a wide variety of practical data analytic situations.

Details

Relative importance analysis is a methodology focused on comparing independent variables (IVs)/features/predictors as well as parameter estimates to one another in terms of how they predict some dependent variable/response/outcome in the context of a predictive model.

The intention of this package is to focus on what I will call "model evaluation" or post hoc examination of IV predictive utility in the context of a vetted, selected predictive model. That is, the methods to apply here will assume that the user has previously applied model selection methods and that the IVs of the predictive model for relative importance analysis have non-trivial effects in improving model fit. The methods in this package are not intended for model selection - though I will acknowledge that many importance methods are (at least implicitly) focused on model selection/identifying which IVs have a trivial effect and removing them.

The only method implemented at current in `domir` is dominance analysis method `domin`. `domin` is a flexible wrapper function that can be used with many modeling functions. `domin` is an extension of the Stata command by the same name (see Luchman, 2021) to the R environment.

See the *vignettes* for a more extensive discussion of basic concepts for DA and the README for further examples and a discussion of some of the implementation specifics of `domin`.

Author(s)

Joseph Luchman <jluchman_at_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

- Groemping, U. (2007). Estimators of relative importance in linear regression based on variance decomposition. The American Statistician, 61(2), 139-147. doi:10.1198/000313007X188252
- Luchman, J. N., Lei, X., & Kaplan, S. A. (2020). Relative Importance Analysis With Multivariate Models: Shifting the Focus from Independent Variables to Parameter Estimates. Journal of Applied Structural Equation Modeling, 4(2), 1-20. doi:10.47263/JASEM.4(2)02
- Luchman, J. N. (2021). Determining relative importance in Stata using dominance analysis: domin and domme. Stata Journal 21(2), 510-538. doi:10.1177/1536867X211025837

`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,
  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.

`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.

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")
```

`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.

Details

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

- Fit statistic for the full model as well as the fit statistic for the all subsets model if any entries in all as well as consmodel
- Matrix describing general dominance statistics, standardized general dominance statistics, and the ranking of the general dominance statistics
- Matrix describing the conditional dominance statistics.
- If conditional is TRUE, matrix describing the complete dominance designations
- 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 domin object.

Value

None. This method is called only for side-effect of printing to the console.

Index