Package ‘gawdis’

July 22, 2025

Type Package

Title Multi-Trait Dissimilarity with more Uniform Contributions

Version 0.1.5

Author Francesco de Bello [aut],
Zoltan Botta-Dukat [aut],
Jan Leps [aut],
Pavel Fibich [aut, cre]

Maintainer Pavel Fibich <pavel.fibich@prf.jcu.cz>

Description R function gawdis() produces multi-trait dissimilarity with more uniform contributions of different traits. de Bello et al. (2021) <doi:10.1111/2041-210X.13537> presented the approach based on minimizing the differences in the correlation between the dissimilarity of each trait, or groups of traits, and the multi-trait dissimilarity. This is done using either an analytic or a numerical solution, both available in the function.

License GPL-2 | GPL-3

Encoding UTF-8

BugReports https://github.com/pavel-fibich/gawdis/issues/

URL https://github.com/pavel-fibich/gawdis/

Depends FD, GA

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2023-05-03 11:10:02 UTC

Contents

GAgawdis *Internal Genetic Algorithm gawdis function*

Description

Internal part of `gawdis()` function for running genetic algorithm

Usage

```
GAgawdis( tr = NULL, asym.bin = NULL, ord = "podani",gr = NULL,
gr.weight = FALSE, fuzzy = NULL, getSpecDists = NULL,
f = NULL, min.weight = 0.001, max.weight = 1, maxiter = 300,
monitor = FALSE, ... )
```

Arguments

`tr`	Matrix or data frame containing the variables. Variables can be numeric, ordered, or factor. Symmetric or asymmetric binary variables should be numeric and only contain 0 and 1. Character variables will be converted to factor. NAs are tolerated.
`asym.bin`	Vector listing the asymmetric binary variables in x.
`ord`	Character string specifying the method to be used for ordinal variables (i.e. ordered). `podani` refers to Eqs. 2a-b of Podani (1999), while "metric" refers to his Eq. 3 (see 'Details'); both options convert ordinal variables to ranks. "classic" simply treats ordinal variables as continuous variables.
`gr`	Vector for traits grouping, i.e. defining group of traits that are considered to be reflecting similar biological information (e.g. many leaf traits in plants covering similar information). By default each trait is treated separately (`groups=NULL`). In order to define groups use the same values, e.g. `groups = c(1,2,2,2,3,3)` in case of 6 variables attributed to 3 groups, with the length of vector that should be the same as `ncol(x)`.
`gr.weight`	Option to weight traits inside the groups. By default it is set to FALSE, all traits inside the groups have the same weights, meaning that some traits will have a greater contribution within the group; TRUE means that `gawdis()` will determine different weights of traits inside the groups, before combining this group with other traits outside the group.
`fuzzy`	Vector including groups which are defining a single variable, like in the case of fuzzy coding and dummy variables. In this case, use the argument `groups` to define which columns belong to the groups. If `fuzzy` includes group name (from `groups` argument), then the function will transform distances between species within specified group to have maximum value set to 1 (e.g. for `groups=c(1,1,2,2,2),fuzzy=c(2)` only distances of group 2 will be transformed). Default is NULL, not to transform distances of any group. Having both `groups.weight=TRUE, fuzzy=TRUE` is not possible, therefore `!is.null(fuzzy)` leads to overwriting `groups.weight` to FALSE.

`getSpecDists`	Allows to use own code that defines the function `getSpecDists(tr,gr,gr.weight)` for computing distances between species for each trait (traits are passed as tr argument). It can be given, or pre-defined function doing the same things as `gowdis()` is used (it is not necessary to specify it). If groups and groups.weight arguments are given in gawdis, then they are passed to `getSpecDists()` as gr and gr.weight arguments.
`f`	This is the criteria used to equalize the contribution of traits to the multi-trait dissimilarity. It can be specified. Alternative, by default, the approach is minimizing the differences in the correlations between the dissimilarity on individual trait and the multi-trait approach. Specifically the 1/SD of correlations (SD=standard deviation) is used, i.e. all traits will tend to have a similar correlation with the multi-trait dissimilarity. opti.f is fitness function that is maximalized by genetic algorithm.
`min.weight`	Set minimum value for weights of traits.
`max.weight`	Set maximum value for weights of traits.
`maxiter`	Maximum number of iterations to run before the GA search is halted, see `?ga` from GA package. The default is 300 which was found to be quite reliable. The greater numbers increase the computation time.
`monitor`	If to monit progress of genetic algorithm.
`...`	Arguments passed to GA

Value

Returns 'diss' as dissimilarity, weights as solution of GA, ga as GA, spedis as species distance.

Examples

```
#GAgawdis() is not exptected to be run directly, but you can try it by

 library(FD)
 GAgawdis(dummy$trait,maxiter=100)
```

`gawdis` *gawdis function*

Description

`gawdis()`, is an extension of the function gowdis, in the package FD, for Gower distance (Gower 1971) as fully described in de Bello et al. (2021). It provides a solution to the problem of unequal traits contribution when combining different traits in a multi-trait dissimilarity (without care the contribution of some traits can much stronger than others, i.e. the correlation of the dissimilarity of individual trait, with the multi-trait dissimilarity, will be much stronger for some traits, particularly categorical ones). The solution to this problem is based on minimizing the differences in the correlation between the dissimilarity of each individual trait (or type of traits) and the multi-trait one.

Such a task can be resolved analytically or using iterative explorations, depending on the type of data (basically is NA is available only the iterative approach is possible). Both approaches assess ways to provide an equal contribution of traits to the combined trait dissimilarity. Iterative exploration borrows an algorithm from genetic analyses (GA), with the package for genetic algorithms GA, Morrall (2003). This approach is used to minimize standard deviation (SD) of Pearson correlations between the Gower dissimilarity based on single traits and Gower distances combining all traits together, with a proper weight on each variable. GA iteratively explores the space of trait weights by trying several sets of weights (population of candidate solutions), and combines them by processes inspired from the biology (e.g. selection, mutation and crossover) to get new sets of weights (new generation) with better fitness than previous one (Morrall 2003). The best fitness in our case are weights with the minimal SD of correlations. GA is thus doing an optimization, meaning that the more interactions is used the better solution should be found (although still there is a random effect applied), but also greater computing time is necessary. When the `groups` are given, first a combined traits distance between species is computed for each group separately as a distance for all traits inside the group together. The computation of the distance depends also on if the traits should be weighted inside the groups. If so, the weights are at the first found by `gawdis()` applied on the matrix with just traits inside the group (gawdis() founds the best weights for the group). If traits should not be weighted inside the groups, directly just a standard Gower distances is applied for all traits inside the group.

Usage

```
gawdis(x,W = NULL, asym.bin = NULL, ord = c("podani", "metric", "classic"),
w.type = c("analytic", "optimized", "equal", "user"), groups = NULL,
groups.weight = FALSE, fuzzy = NULL, opti.getSpecDists = NULL,
opti.f = NULL,opti.min.weight = 0.01, opti.max.weight = 1,
opti.maxiter = 300, silent = FALSE)
```

Arguments

`x`	Matrix or data frame containing the variables. Variables can be numeric, ordered, or factor. Symmetric or asymmetric binary variables should be numeric and only contain 0 and 1. Character variables will be converted to factor. NAs are tolerated.
`W`	Vector listing the weights for the variables in x. W is considered only if `w.type` is `user`, for `w.type="equal"` all weights having the same value and for other w.type's the weights are computed (see `w.type`).
`asym.bin`	Vector listing the asymmetric binary variables in x.
`ord`	Character string specifying the method to be used for ordinal variables (i.e. ordered). "podani" refers to Eqs. 2a-b of Podani (1999), while "metric" refers to his Eq. 3 (see 'Details'); both options convert ordinal variables to ranks. "classic" simply treats ordinal variables as continuous variables.
`w.type`	Type of used method. `w.type = "analytic"` (default option) – weights optimized by a mathematical algorithm (no NAs are allowed in this option); `w.type = "optimized"` – weights optimized by genetic/optimization algorithm based on iteractions; `w.type = "equal"` – equal weights, `w.type = "user"` – user defined weights are used. Note that is `w.type = "analytic"` in case of NAs, the function will apply `w.type = "equal"`.

`groups`	Vector for traits grouping, i.e. defining group of traits that are considered to be reflecting similar biological information (e.g. many leaf traits in plants covering similar information). By default each trait is treated separately (`groups = NULL`). In order to define groups use the same values, e.g. `groups = c(1,2,2,2,3,3)` in case of 6 variables attributed to 3 groups, with the length of vector that should be the same as `ncol(x)`.
`groups.weight`	Option to weight traits inside the groups. By default it is set to FALSE, all traits inside the groups have the same weights, meaning that some traits will have a greater contribution within the group; TRUE means that gawdis will determine different weights of traits inside the groups, before combining this group with other traits outside the group.
`fuzzy`	Vector including groups which are defining a single variable, like in the case of fuzzy coding and dummy variables. In this case, use the argument `groups` to define which columns belong to the groups. If `fuzzy` includes group name (from `groups` argument), then the function will transform distances between species within specified group to have maximum value set to 1 (e.g. for `groups=c(1,1,2,2,2),fuzzy=c(2)` only distances of group 2 will be transformed). Default is NULL, not to transform distances of any group. Having both `groups.weight=TRUE, fuzzy=TRUE` is not possible, therefore `!is.null(fuzzy)` leads to overwriting `groups.weight` to FALSE.
`opti.getSpecDists`	Allows to use own code that defines the function `getSpecDists(tr,gr,gr.weight)` for computing distances between species for each trait (traits are passed as tr argument). It can be given, or pre-defined function doing the same things as gowdis is used (it is not necessary to specify it). If groups and groups.weight arguments are given in gawdis, then they are passed to `getSpecDists()` as gr and gr.weight arguments.
`opti.f`	This is the criteria used to equalize the contribution of traits to the multi-trait dissimilarity. It can be specified. Alternative, by default, the approach is minimizing the differences in the correlations between the dissimilarity on individual trait and the multi-trait approach. Specifically the 1/SD of correlations (SD=standard deviation) is used, i.e. all traits will tend to have a similar correlation with the multi-trait dissimilarity. opti.f is fitness function that is maximalized by genetic algorithm.
`opti.min.weight`	Set minimum value for weights of traits.
`opti.max.weight`	Set maximum value for weights of traits.
`opti.maxiter`	Maximum number of iterations to run before the GA search is halted, see ?ga from GA package. The default is 300 which was found to be quite reliable. The greater numbers increase the computation time.
`silent`	If to print warnings and detailed information during the computation.

Value

An object of class dist with the following attributes: Labels, Types (the variable types, where 'C' is continuous/numeric, 'O' is ordinal, 'B' is symmetric binary, 'A' is asymmetric binary, and 'N' is nominal), Size, Metric. Including attributes 1) “correls” with the correlations of each trait with the multi-trait dissimilarity, 2) “weights” for the weights of traits, 3) “group.correls” with weights of groups, 4)”components” with between species transformed distances, and 5) “cor.mat” with correlations between traits.

References

de Bello, F. et al. (2021) Towards a more balanced combination of multiple traits when computing functional differences between species. Methods in Ecology and Evolution, doi: https://doi.org/10.1111/2041-210X.13537 .

Gower, J. C. (1971) A general coefficient of similarity and some of its properties. Biometrics 27: 857-871.

Podani, J. (1999) Extending Gower’s general coefficient of similarity to ordinal characters. Taxon 48:331-340.

Morrall, D. (2003) Ecological Applications of Genetic Algorithms. Springer, Berlin, Heidelberg.

Laliberté, E., and Legendre, P. (2010) A distance-based framework for measuring functional diversity from multiple traits. Ecology 91:299-305.

Laliberté, E., Legendre, P., and Shipley, B. (2014). FD: measuring functional diversity from multiple traits, and other tools for functional ecology. R package version 1.0-12. https://cran.r-project.org/package=FD .

See Also

gowdis from FD package.

Examples

```
library(FD) # input data
#the gowdis and gawdis functions provide the same results#
ex1 <- gowdis(dummy$trait)
#using gawdis in the same way as gowdis
ex1.gaw1 <- gawdis(dummy$trait, w.type ="equal")
plot(ex1, ex1.gaw1); abline(0, 1)
#but when doing so, some traits have stronger contribution on the
#multi-trait dissimilarity particularly factorial and binary traits#
attr(ex1.gaw1, "correls")
#correlation of single-trait dissimilarity with multi-trait one#

#the gawdis function finds the best weights to equalize trait
#contributions this can be done in two ways: analytic=using formulas;
#optimized=using iterations both approaches give very similar results
#but only the latter can work with NAs#
#for the sake of comparisons here NAs are removed#
analytical<-gawdis(dummy$trait[,c(2,4,6,8)], w.type ="analytic")
#it is not needed to add the argument w.type, this is the approach
#used by default if not defined#
attr(analytical, "correls")
attr(analytical, "weights") #weights finally given to traits
iters<-gawdis(dummy$trait[,c(2,4,6,8)], w.type ="optimized", opti.maxiter=2)
```

```
#here we used 'only' 2 iterations, to speed up the process of tests and
#because it better to use at least opti.maxiter=100#
attr(iters, "correls")
#correlations are not equal, but enough close to each other
attr(iters, "weights")
plot(analytical, iters); abline(0, 1)

#the function can be used also for fuzzy coded/dummy variables traits#
#let's create some data#
bodysize<-c(10, 20, 30, 40, 50, NA, 70)
carnivory<-c(1, 1, 0, 1, 0,1, 0)
red<-c(1, 0, 0.5, 0, 0.2, 0, 1)
yellow<-c(0, 1, 0, 0, 0.3, 1, 0)
blue<-c(0, 0, 0.5,1, 0.5, 0, 0)
colors.fuzzy<-cbind(red, yellow, blue)
names(bodysize)<-paste("sp", 1:7, sep="")
names(carnivory)<-paste("sp", 1:7, sep="")
rownames(colors.fuzzy)<-paste("sp", 1:7, sep="")
tall<-as.data.frame(cbind(bodysize, carnivory, colors.fuzzy))
tall
#use groups and fuzzy to treat the 3 columns related to traits
#as one traits#
gaw.tall<-gawdis(tall, w.type="equal", groups =c(1, 2, 3,3,3),fuzzy=c(3))
attr(gaw.tall,"weights")
#to get optimized results just change w.type="optimized"
```

Index