Package: optimus (via r-universe)

September 8, 2024

Type Package

Title Model Based Diagnostics for Multivariate Cluster Analysis

Version 0.2.0

Date 2018-01-16

Maintainer Mitchell Lyons <mitchell.lyons@gmail.com>

Description Assessment and diagnostics for comparing competing clustering solutions, using predictive models. The main intended use is for comparing clustering/classification solutions of ecological data (e.g. presence/absence, counts, ordinal scores) to 1) find an optimal partitioning solution, 2) identify characteristic species and 3) refine a classification by merging clusters that increase predictive performance. However, in a more general sense, this package can do the above for any set of clustering solutions for i observations of j variables.

Imports stats, methods, mvabund (>= 3.1), ordinal (>= 2015.1-21)

Depends R (>= 3.1.0)

URL https://github.com/mitchest/optimus/

BugReports https://github.com/mitchest/optimus/issues

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

Repository https://mitchest.r-universe.dev

RemoteUrl https://github.com/mitchest/optimus

RemoteRef HEAD

RemoteSha 8caa20da60c92578acfaf2f835fe528935e75a1d

Contents

`find_optimal` *Find an optimal classification among competing clustering solutions*

Description

`find_optimal` takes a clustering solution, or a set of related clustering solutions, fits models based on the underlying multivariate data, and calculates the sum-of-AIC value for the solution/s. The smallest sum-of-AIC value is the optimal solution.

Usage

```
find_optimal(data, clustering, family, K = 1, cutree = NULL,
  cutreeLevels = 2:10, cutreeOveride = FALSE)
```

Arguments

`data`	a data frame (or object that can be coerced by `as.data.frame` containing the "raw" multivariate data. This is not necessarily the data used by the clustering algorithm - it is the data on which you are testing the predictive ability of the clustering solutions.
`clustering`	either an object on which `cutree` will work, or a list with one or more components, each containing an atomic vector of cluster labels (that can be coerced by `as.factor`). The number of cluster labels (either generated by `cutree` or supplied in each list component) must match the number of rows of the object supplied in the `data` argument.
`family`	a character string denoting the error distribution to be used for model fitting. The options are similar to those in `family`, but are more limited - see Details.
`K`	number of trials in binomial regression. By default, K=1 for presence-absence data (with cloglog link).
`cutree`	logical, but default is `NULL` for auto-detection. Whether `cutree` should be used on the object supplied to the `clustering` argument
`cutreeLevels`	a numerical vector specifying the different partitioning levels to calculate sum-of-AIC for (that is the values of `k` to be supplied to `cutree`). Ignored if `cutree = FALSE`, as the number of partitions will be automatically generated from the number of unique levels in each component of `clustering`.

`cutreeOveride` logical. Ignored if `cutree = FALSE`. Should the checks on whether the object supplied to the `clustering` works with `cutree`? WARNING: only set `cutreeOveride = TRUE` if you are totally sure `cutree` works, but the error message is telling you it doesn't. See Arguments in `cutree` and first consider modifying the object supplied to clustering=.

Details

`find_optimal` is built on the premise that a *good* clustering solution (i.e. a classification) should provide information about the composition and abundance of the multivariate data it is classifying. A natural way to formalize this is with a predictive model, where group membership (clusters) is the predictor, and the multivariate data (site by variables matrix) is the response. `find_optimal` fits linear models to each variable, and calculates the sum of the AIC value (sum-of-AIC) for each model. sum-of-AIC is motivated as an estimate of Kullback-Leibler distance, so we posit that the clustering solution that minimises the sum-of-AIC value is the *best*. So, in context of optimal partitioning, `find_optimal` can be used to automatically and objectively decide which clustering solution is the best among competing solutions. Lyons et al. (2016) provides background, a detailed description of the methodology, and application of sum-of-AIC on both real and simulated ecological multivariate abundance data.

At present, `find_optimal` supports the following error distributions for model fitting:

- Gaussian (LM)
- Negative Binomial (GLM with log link)
- Poisson (GLM with log link)
- Binomial (GLM with cloglog link for binary data, logit link otherwise)
- Ordinal (Proportional odds model with logit link)

Gaussian LMs should be used for 'normal' data. Negative Binomial and Poisson GLMs should be used for count data. Binomial GLMs should be used for binary and presence/absence data (when `K=1`), or trials data (e.g. frequency scores). If Binomial regression is being used with `K>1`, then `data` should be numerical values between 0 and 1, interpreted as the proportion of successful cases, where the total number of cases is given by `K` (see Details in `family`). Ordinal regression should be used for ordinal data, for example, cover-abundance scores. For ordinal regression, data should be supplied as either 1) factors, with the appropriate ordinal level order specified (see `levels`) or 2) numeric, which will be coerced into a factor with levels ordered in numerical order (e.g. cover-abundance/numeric response scores). LMs fit via `manylm`; GLMs fit via `manyglm`; proportional odds model fit via `clm`.

Value

a data frame containing the sum-of-AIC value for each clustering solution, along with the number of clusters the solution had. The object is of class `aicsums`.

Attributes for the data frame are:

`family` which error distribution was used for modelling, see Arguments

`K` number of cases for Binomial regression, see Arguments

`cutree` whether `cutree` was used, see Arguments

`cutreeLevels` number of partitioning levels specified, see Arguments

Author(s)

Mitchell Lyons

References

Lyons et al. 2016. Model-based assessment of ecological community classifications. *Journal of Vegetation Science*, **27 (4)**: 704–715.

See Also

plot.aicsums, get_characteristic, merge_clusters, S3 for residual plots (at some stage)

Examples

```
## Prep the 'swamps' data
## ======================

data(swamps) # see ?swamps
swamps <- swamps[,-1]

## Assess clustering solutions using cutree() method
## =================================================

## perhaps not the best clustering option, but this is base R
swamps_hclust <- hclust(d = dist(x = log1p(swamps), method = "canberra"),
                        method = "complete")

## calculate sum-of-AIC values for 10:25 clusters, using the hclust() output
swamps_hclust_aics <- find_optimal(data = swamps, clustering = swamps_hclust,
family = "poisson", cutreeLevels = 10:25)

## Looks like ~20 clusters is where predictive performance levels off

## Note here that the data passed to find_optimal() was actually NOT the
## data used for clustering (transform/distance), rather it was the
## original abundance (response) data of interest

## plot - lower sum-of-AIC valuea indicate 'better' clustering
plot(swamps_hclust_aics)


## Not run:
## Assess clustering solutions by supplying a list of solutions
## ============================================================

## again, we probably wouldn't do this, but for illustrative purposes
## note that we are generating a list of solutions this time
swamps_kmeans <- lapply(X = 2:40,
FUN = function(x, data) {stats::kmeans(x = data, centers = x)$cluster},
data = swamps)

## calculate sum-of-AIC values for the list of clustering solutions
```

```
swamps_kmeans_aics <- find_optimal(data = swamps, clustering = swamps_kmeans,
family = "poisson") # note cutreeLevels= argument is not needed

plot(swamps_kmeans_aics)

## End(Not run)

## See vignette for more explanation than this example
## ============================================================
```

`get_characteristic` *Determine the characteristic variables (e.g. species) of a clustering solution (e.g. classification)*

Description

`get_characteristic` takes a clustering solution, fits models based on the underlying multivariate data, and determines 'important' variables for the clustering solution. In Ecology, particularly vegetation science, this is the process of determining characteristic (or diagnostic/indicator) species of a classification.

Usage

```
get_characteristic(data, clustering, family, type = "per.cluster",
  signif = TRUE, K = 1)
```

Arguments

`data`	a data frame (or object that can be coerced by `as.data.frame` containing the "raw" multivariate data. This is not necessarily the data used by the clustering algorithm - it is the data on which you are testing the predictive ability of the clustering solution.
`clustering`	a clustering solution for `data`, that is, a vector of cluster labels (that can be coerced by `as.factor`). The number of cluster labels must match the number of rows of the object supplied in the `data` argument. The solution could for example come form a call to `cutree`, see Examples
`family`	a character string denoting the error distribution to be used for model fitting. The options are similar to those in `family`, but are more limited - see Details.
`type`	a character string, one of `"per.cluster"` or `"global"`, denoting the type of characteristic variables (species). See Details.
`signif`	logical, denoting whether *significance* should be returned also when `"type=per.cluster"`. Ignored if `"type=global"`. See Details. Minimal additional overhead is required if `TRUE`.
`K`	number of trials in binomial regression. By default, K=1 for presence-absence data (with cloglog link).

Details

`get_characteristic` is built on the premise that a *good* clustering solution (i.e. a classification) should provide information about the composition and abundance of the multivariate data it is classifying. A natural way to formalize this is with a predictive model, where group membership (clusters) is the predictor, and the multivariate data (site by variables matrix) is the response. `get_characteristic` fits linear models to each variable. If `type = "per.cluster"` the coefficients corresponding to each level of the clustering solution for each variable are used to define the characteristic variables for each cluster level. If `type = "global"`, characteristic variables are determined (via delta AIC - larger values = more important) for the overall classification. If `signif = TRUE`, delta AIC (that is, to the corresponding null model) and the coefficient standard errors are also returned with the per-cluster characteristic variables. We loosely define that the larger the coefficient (with larger delta AIC values and smaller standard errors guiding *significance*), the *more* characteristic that variable (species) is. Lyons et al. (2016) provides background, a detailed description of the methodology, and application of delta AIC on both real and simulated ecological multivariate abundance data.

At present, `get_characteristic` supports the following error distributions for model fitting:

- Gaussian (LM)
- Negative Binomial (GLM with log link)
- Poisson (GLM with log link)
- Binomial (GLM with cloglog link for binary data, logit link otherwise)
- Ordinal (Proportional odds model with logit link)

Gaussian LMs should be used for 'normal' data. Negative Binomial and Poisson GLMs should be used for count data. Binomial GLMs should be used for binary and presence/absence data (when `K=1`), or trials data (e.g. frequency scores). If Binomial regression is being used with `K>1`, then `data` should be numerical values between 0 and 1, interpreted as the proportion of successful cases, where the total number of cases is given by `K` (see Details in `family`). Ordinal regression should be used for ordinal data, for example, cover-abundance scores. For ordinal regression, data should be supplied as either 1) factors, with the appropriate ordinal level order specified (see `levels`) or 2) numeric, which will be coerced into a factor with levels ordered in numerical order (e.g. cover-abundance/numeric response scores). LMs fit via `manylm`; GLMs fit via `manyglm`; proportional odds model fit via `clm`.

Value

either a list of sorted characteristic variables for each cluster (of class `perclustchar`) or a data frame containing the delta AIC values for each variable (of class `globalchar`). If `signif=` is not `"none"`, then the corresponding significance metrics are appended.

Attributes for the object are:

`family` which error distribution was used for modelling, see Arguments

`type` the type of characteristic variables calculated, see Arguments

`K` number of cases for Binomial regression, see Arguments

Author(s)

Mitchell Lyons

References

Lyons et al. 2016. Model-based assessment of ecological community classifications. *Journal of Vegetation Science*, **27 (4)**: 704–715.

See Also

find_optimal, S3 for print 'top-n' variables for each cluster, S3 for residual plots (at some stage)

Examples

```
## Prep the 'swamps' data
## ======================

data(swamps) # see ?swamps
swamps <- swamps[,-1]

## Find characteristic species in a classification of the swamps data
## ==================================================================

## perhaps not the best clustering option, but this is base R
swamps_hclust <- hclust(d = dist(x = log1p(swamps), method = "canberra"),
                        method = "complete")

# calculate per cluster characteristic species
swamps_char <- get_characteristic(data = swamps,
clustering = cutree(tree = swamps_hclust, k = 10), family = "poisson",
type = "per.cluster")

# look at the top 10 characteristic species for cluster 1
head(swamps_char[[1]], 10)

# calculate global characteristic species
swamps_char <- get_characteristic(data = swamps,
clustering = cutree(tree = swamps_hclust, k = 10), family = "poisson",
type = "global")

# top 10 characteristic species for the whole classification
head(swamps_char, 10)

## See vignette for more explanation than this example
## ==========================================================
```

---

merge_clusters *Iteratively merges clusters in a way that improves predictive performance*

Description

merge_clusters takes a clustering solution, generates all possible pairwise combinations of clusters, fits models to each combination, and merges the pair with the lowest delta AIC. The process is repeated iteratively

Usage

```
merge_clusters(data, clustering, family, n.iter = NULL, K = 1,
  quietly = FALSE)
```

Arguments

data	a data frame (or object that can be coerced by as.data.frame containing the "raw" multivariate data. This is not necessarily the data used by the clustering algorithm - it is the data on which you are testing the predictive ability of the clustering solution.
clustering	an initial clustering solution (to be iteratively merged) for data, that is, a vector of cluster labels (that can be coerced by as.factor). The number of cluster labels must match the number of rows of the object supplied in the data argument. The solution could for example come form a call to cutree, see Examples
family	a character string denoting the error distribution to be used for model fitting. The options are similar to those in family, but are more limited - see Details.
n.iter	the number of merging iterations to perform, by default it will merge down to 2 clusters
K	number of trials in binomial regression. By default, K=1 for presence-absence data (with cloglog link)
quietly	suppress messages during merging procedure

Details

merge_clusters is built on the premise that a *good* clustering solution (i.e. a classification) should provide information about the composition and abundance of the multivariate data it is classifying. A natural way to formalize this is with a predictive model, where group membership (clusters) is the predictor, and the multivariate data (site by variables matrix) is the response. merge_clusters fits linear models to each pairwise combination of a given set of clusters, and calculates their delta sum-of-AIC (that is, to the corresponding null model). The smallest delta AIC is taken to be the cluster pair that is *most* similar, so it is merged, and the process is repeated. Lyons et al. (2016) provides background, a detailed description of the methodology, and application of delta AIC on both real and simulated ecological multivariate abundance data.

At present, merge_clusters supports the following error distributions for model fitting:

- Gaussian (LM)
- Negative Binomial (GLM with log link)
- Poisson (GLM with log link)
- Binomial (GLM with cloglog link for binary data, logit link otherwise)
- Ordinal (Proportional odds model with logit link)

Gaussian LMs should be used for 'normal' data. Negative Binomial and Poisson GLMs should be used for count data. Binomial GLMs should be used for binary and presence/absence data (when K=1), or trials data (e.g. frequency scores). If Binomial regression is being used with K>1, then data should be numerical values between 0 and 1, interpreted as the proportion of successful cases, where the total number of cases is given by K (see Details in family). Ordinal regression should be used for ordinal data, for example, cover-abundance scores. For ordinal regression, data should be supplied as either 1) factors, with the appropriate ordinal level order specified (see levels) or 2) numeric, which will be coerced into a factor with levels ordered in numerical order (e.g. cover-abundance/numeric response scores). LMs fit via manylm; GLMs fit via manyglm; proportional odds model fit via clm.

Value

a list containing the clustering solution (vector) at each merge iteration. The object is of class dsumaic, and can be directly passed to find_optimal.

Attributes for the data frame are:

family which error distribution was used for modelling, see Arguments

K number of cases for Binomial regression, see Arguments

Author(s)

Mitchell Lyons

References

Lyons et al. 2016. Model-based assessment of ecological community classifications. *Journal of Vegetation Science*, **27 (4)**: 704–715.

See Also

find_optimal, get_characteristic, S3 print function for 'daic' class, S3 residual plotting function

Examples

```
## Not run:
## Prep the 'swamps' data
## ======================

data(swamps) # see ?swamps
swamps <- swamps[,-1]

## Merge via AIC and compare to hclust heirarchy
## =============================================

## perhaps not the best clustering option, but this is base R
swamps_hclust <- hclust(d = dist(x = log1p(swamps), method = "canberra"),
                        method = "complete")
```

```
## generate iteratively merged clustering solutions, based on sum-of-AIC
clustering_aicmerge <- merge_clusters(swamps, cutree(tree = swamps_hclust, k = 30),
family = "poisson", n.iter = 20)

## compare to hclust heirarchy
optimal_aicmerge <- find_optimal(data = swamps, clustering = clustering_aicmerge,
family = "poisson")

optimal_hclust <- find_optimal(data = swamps, clustering = swamps_hclust,
family = "poisson", cutreeLevels = 10:30))

plot(optimal_aicmerge)
points(optimal_hclust, col = "red", pch = 16)

## End(Not run)
```

`plot.aicsums` *Plot sum-of-AIC results*

Description

S3 `plot` method for sum-of-AIC results from `find_optimal`.

Usage

```
## S3 method for class 'aicsums'
plot(x, col = "black", pch = 16, ...)
```

Arguments

`x`	an object of class `aicsums`, as produced by `find_optimal`.
`col`	point colour
`pch`	point type
`...`	additional arguments to pass to `plot`.

Value

A plot is drawn on the current graphics device

Author(s)

Mitchell Lyons

Examples

```
## see ?find_optimal()
```

points.aicsums *Plot more sum-of-AIC results*

Description

S3 points method for sum-of-AIC results from find_optimal. Implemented to compare multiple outputs from find_optimal.

Usage

```
## S3 method for class 'aicsums'
points(x, col = sample(1:20, 1), pch = sample(c(1:15,
  17:20), 1), ...)
```

Arguments

x	an object of class aicsums, as produced by find_optimal.
col	point colour - random if not specified
pch	point type - random if not specified
...	additional arguments to pass to points.

Value

Points drawn on the current plot

Author(s)

Mitchell Lyons

Examples

```
## see ?find_optimal()
```

swamps *Dharawal National Park Upland Heath Swamps Plot Network*

Description

The Upland Heath Swamps Plot Network Vegetation Structure and Floristics Data Package contains information on the vegetation structure and species present in 60 established swamp monitoring sites in upland swamps scattered throughout the study area (Keith and Myerscough 1993). Each site is sampled in nine combinations of moisture-by-vegetation structure strata. The Upland Heath Swamps Plot Network research plots commenced in 1983 and have been revisited in 2004, 2009 and again in 2014. More detail can be found at http://www.ltern.org.au/ltern-plot-networks/upland-heath-swamps.

Usage

```
swamps
```

Format

A data frame with 54 rows and 171 variables. The first column (`transect`) is the name of the site (0.5 x 0.5 m quadrats). The remaining columns contain frequency counts (out of 30) for all vascular plant taxa at the site (taxonomy in metadata link below).

Source

http://www.ltern.org.au/knb/metacat?action=read&qformat=html&docid=ltern.84.15

Index