Package 'opticut'

December 17, 2016

Type Package

Title Likelihood Based Optimal Partitioning for Indicator Species Analysis

Version 0.1-0

Date 2016-12-16

Author Peter Solymos [cre, aut], Ermias T. Azeria [ctb]

Maintainer Peter Solymos <solymos@ualberta.ca>

Description Likelihood based optimal partitioning for indicator species analysis. Finding the best binary partition for each species based on model selection, possibly controlling for modifying/confounding variables as described in Kemencei et al. (2014) <doi:10.1556/ComEc.15.2014.2.6>. The package also implements various measures of uncertainty based on binary partitions, optimal multinomial partitioning, and exploratory suitability indices, with native support for parallel computations.

URL https://github.com/psolymos/opticut

BugReports https://github.com/psolymos/opticut/issues

Depends R (>= 3.1.0), pbapply (>= 1.3-0)

Imports MASS, pscl, betareg, ResourceSelection (>= 0.3-0), parallel, mefa4

License GPL-2

LazyLoad yes

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-17 10:48:16

R topics documented:

`opticut-package` *Likelihood Based Optimal Partitioning for Indicator Species Analysis*

Description

Likelihood based optimal partitioning for indicator species analysis. Finding the best binary partition for each species based on model selection, possibly controlling for modifying/confounding variables as described in Kemencei et al. (2014) <doi:10.1556/ComEc.15.2014.2.6>. The package also implements various measures of uncertainty based on binary partitions, optimal multinomial partitioning, and exploratory suitability indices, with native support for parallel computations.

Details

The DESCRIPTION file:

Package:	opticut
Type:	Package
Title:	Likelihood Based Optimal Partitioning for Indicator Species Analysis
Version:	0.1-0
Date:	2016-12-16
Author:	Peter Solymos [cre, aut], Ermias T. Azeria [ctb]
Maintainer:	Peter Solymos <solymos@ualberta.ca>
Description:	Likelihood based optimal partitioning for indicator species analysis. Finding the best binary partition for each s
URL:	https://github.com/psolymos/opticut
BugReports:	https://github.com/psolymos/opticut/issues
Depends:	R (>= 3.1.0), pbapply (>= 1.3-0)
Imports:	MASS, pscl, betareg, ResourceSelection (>= 0.3-0), parallel, mefa4
License:	GPL-2
LazyLoad:	yes
LazyData:	true

Index of help topics:

allComb	Finding all possible binary partitions
bestmodel	Best model, partition, and MLE
beta2i	Indicator values
dolina	Land snail data set
lorenz	Lorenz curve bases thresholds and partitions
occolors	Color palettes for the opticut package
ocoptions	Options for the opticut package
opticut	Likelihood based optimal partitioning for indicator species analysis
opticut-package	Likelihood Based Optimal Partitioning for Indicator Species Analysis
optilevels	Optimal number of factor levels
rankComb	Ranking based binary partitions
sindex	Weighted relative suitability index
uncertainty	Quantifying uncertainty for fitted objects

The main user interface is the `opticut` function to find the optimal binary partitioning. Make sure to evaluate `uncertainty`.

`optilevels` goes beyond binary partitions and finds the optimal number of factor levels.

Author(s)

Peter Solymos [cre, aut], Ermias T. Azeria [ctb]

Maintainer: Peter Solymos <solymos@ualberta.ca>

References

Kemencei, Z., Farkas, R., Pall-Gergely, B., Vilisics, F., Nagy, A., Hornung, E. & Solymos, P., 2014. Microhabitat associations of land snails in forested dolinas: implications for coarse filter conservation. Community Ecology 15:180–186. <doi:10.1556/ComEc.15.2014.2.6>

Examples

```
## community data
y <- cbind(
    Sp1=c(4,6,3,5, 5,6,3,4, 4,1,3,2),
    Sp2=c(0,0,0,0, 1,0,0,1, 4,2,3,4),
    Sp3=c(0,0,3,0, 2,3,0,5, 5,6,3,4))

## stratification
g <-    c(1,1,1,1, 2,2,2,2, 3,3,3,3)

## find optimal partitions for each species
oc <- opticut(formula = y ~ 1, strata = g, dist = "poisson")
summary(oc)

## visualize the results
plot(oc, cut = -Inf)

## quantify uncertainty
```

```
uc <- uncertainty(oc, type = "asymp", B = 999)
summary(uc)

## go beyond binary partitions
ol <- optilevels(y[,"Sp2"], as.factor(g))
ol[c("delta", "coef", "rank", "levels")]
```

`allComb` *Finding all possible binary partitions*

Description

These functions are used to find all possible binary partitions. Finding all combinations require a classification vector with K > 1 partitions.

Usage

```
allComb(x, collapse)
kComb(k)
checkComb(x)
```

Arguments

`x`	a vector for `allComb` (can be of any type but treated as factor, must have at least 2 unique values); and a numeric matrix for `checkComb`.
`collapse`	character, what to paste between levels. Defaults to `getOption("ocoptions")$collapse`.
`k`	numeric, number of partitions in a given classification (K > 1).

Value

`kComb` returns a contrast matrix corresponding to all possible binary partitions of the factor with k levels. Complements are not counted twice, i.e. (0,0,1,1) is equivalent to (1,1,0,0). The number of such possible combinations is M = 2^(K-1) - 1.

`allComb` this takes a classification vector with at least 2 levels and returns a model matrix with binary partitions.

`checkComb` checks if combinations are unique and non-complementary (misfits are returned as attributes). Returns a logical value.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

`rankComb` for alternative partitioning algorithm.

`opticut` for the user interface.

Examples

```
kComb(k = 2)
kComb(k = 3)
kComb(k = 4)

## finding all combinations
(f <- rep(LETTERS[1:4], each=2))
(mc <- allComb(f, collapse = "_"))
## checking for complementary entries
checkComb(mc) # TRUE
## adding complementary entries to the matrix
mc2 <- cbind(z = 1 - mc[,1], mc[,c(1:ncol(mc), 1)])
colnames(mc2) <- 1:ncol(mc2)
mc2
checkComb(mc2) # FALSE
```

`bestmodel` *Best model, partition, and MLE*

Description

Accessing best model, best partition, and Maximum Likelihood Estimate from fitted objects.

Usage

```
bestmodel(object, ...)
bestpart(object, ...)
getMLE(object, ...)
```

Arguments

`object`	fitted model object.
`...`	other arguments passed to the underlying functions.

Value

`bestmodel` returns the best supported model for further manipulation (e.g. prediction).

`bestpart` returns a matrix with the best supported partitions for each species (species as columns).

`getMLE` returns a named list corresponding to the best supported model. The list has the following elements: `coef` is the Maximum Likelihood Estimate (MLE), `vcov` is the variance-covariance matrix for the MLE, `dist` is the distribution inherited from input `object`.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

`opticut`, `uncertainty`.

beta2i *Indicator values*

Description

Transformation of estimated coefficients to indicator values.

Usage

```
beta2i(x, scale = 1)
```

Arguments

x	numeric, real valued coefficients.
scale	numeric, scaling constant.

Value

Returns a numeric vector (I = abs(tanh(x * scale))).

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

opticut used the scaled I values as indicator values.

ocoptions for setting value for the default scaling factor.

Examples

```
x <- seq(-5, 5, 0.1)
Col <- occolors(c("red", "blue"))(10)
plot(x, beta2i(x), type = "n")
s <- seq(1, 0.1, -0.1)
for (i in 1:10) {
    lines(x, beta2i(x, scale = s[i]), col = Col[i])
    text(1.5 - 0.2, beta2i(1.5, scale = s[i]), s[i], col = Col[i])
}
```

dolina *Land snail data set*

Description

A comprehensive and micro-scale land snail data set from 16 dolines of the Aggtelek Karst Area, Hungary. Data set containing land snail counts as described in Kemecei et al. 2014.

Usage

```
data("dolina")
```

Format

A list with 3 elements: `xtab` is a sample x species matrix with counts, `samp` is a data frame with sample level attributes, `taxa` is a data frame with scientific names and families for the species.

Land snails were sampled during daylight hours between 16 and 18 of August, 2007. Samples were taken from four microhabitat types (`dolina$samp$microhab`, `dolina$samp$mhab`): litter (LI), trunks of live trees (TL), dead wood (also known as coarse woody debris; DW), and rock (RO). In each of the 16 dolina (`dolina$samp$dolina`), seven samples were collected in the litter microhabitat along a north-south transect. In the case of the other three microhabitat types, samples were collected from three random locations per microhabitat type in each dolina. A total of 256 samples (`dolina$samp$sample`) were collected, each consisting 2 sub-samples collected by 2 sampling methods (`dolina$samp$method`): litter samples (Q) and timed search (T).

One liter of litter samples including topsoil were collected to be examined later in the laboratory. Litter samples were collected adjacent to live wood, dead wood and rocks, and not from the wood or rocks themselves. Litter samples in the litter microhabitat were not collected near wood or rocks (minimum distance of 2 meters). During 5 minutes per site of time-restricted direct search we investigated microhabitats in a 1 meter radius circle around the litter sample location, but also including tree or rock surfaces.

The vertical zone (`dolina$samp$stratum`, bottom, middle or edge of the dolinas), aspect of these sample locations (`dolina$samp$aspect`), along with litter depth (`dolina$samp$lthick`, cm), and litter moisture (`dolina$samp$lmoist`, scored on an ordinal scale: 1=dry, 2=fresh, 3=moist) were also recorded.

Distinction of live animals versus fresh empty shells was not feasible due to the method of sorting dry material and the delay in litter sample processing, so these were combined and constituted the 'fresh' group. Whitened, disintegrating and broken shells constituted the 'broken' group. This 'broken' group was excluded from the data set.

Source

Solymos et al. 2016 and Kemencei et al. 2014.

References

Kemencei, Z., Farkas, R., Pall-Gergely, B., Vilisics, F., Nagy, A., Hornung, E. & Solymos, P., 2014. Microhabitat associations of land snails in forested dolinas: implications for coarse filter conservation. Community Ecology 15:180–186. <doi:10.1556/ComEc.15.2014.2.6>

Solymos, P., Kemencei, Z. Pall-Gergely, B., Farkas, R., Vilisics, F., Nagy, A., Kisfali, M. & Hornung, E., 2016. Public data from the dolina project. Version 1.0. Zenodo, <doi:10.5281/zenodo.53080>

Examples

```
data(dolina)
str(dolina)

## species richness by microhabitat and method
Richness <- rowSums(dolina$xtab > 0)
boxplot(Richness ~ mhab + method, dolina$samp,
    ylab="Species richness", main="Dolina data set",
    col=rep(c("#2C7BB6", "#D7191C"), each=4))
```

---

lorenz *Lorenz curve bases thresholds and partitions*

Description

Lorenz curve bases thresholds and partitions.

Usage

```
lorenz(x, na.last = TRUE)

## S3 method for class 'lorenz'
quantile(x, probs = seq(0, 1, 0.25), ...)
## S3 method for class 'lorenz'
plot(x, type = c("L", "x"),
    tangent = NA, h = NA, v = NA, ...)

## S3 method for class 'summary.lorenz'
print(x, digits, ...)
## S3 method for class 'lorenz'
summary(object, ...)
```

Arguments

x	a vector of nonnegative numbers for lorenz, or an object to plot or summarized.
na.last	for controlling the treatment of NAs. If TRUE, missing values in the data are put last; if FALSE, they are put first; if NA, they are removed (see order).
probs	numeric vector of probabilities with values in [0,1], as in quantile.

type	character, indicating what to plot. "L": the cumulative distribution quantiles; "x": ordered but not-cumulated values.
tangent	color value for the Lorenz-curve tangent when plotted. The default NA value omits the tangent from the plot.
h	color value for the horizontal line for the Lorenz-curve tangent when plotted. The default NA value omits the horizontal line from the plot.
v	color value for the vertical line for the Lorenz-curve tangent when plotted. The default NA value omits the vertical line from the plot.
digits	numeric, number of significant digits in output.
object	object to summarize.
...	other arguments passed to the underlying functions.

Details

The Lorenz curve is a continuous piecewise linear function representing the distribution of income or wealth. Cumulative portion of the population: p_i = 1 / n (i=1,...,n), vs. cumulative portion of wealth: L_i = sum_j=1^i x_j / sum_j=1^n x_j, where x_i are indexed in non-decreasing order (x_i <= x_i+1). By convention, p_0 = L_0 = 0.

The following charactersitics of the Lorenz curve are calculated: "t": index where tangent (slope 1) touches the curve; "x(t)", "p(t)", and "L(t)" are values corresponding to index t. "S": Lorenz asymmetry coefficient (S = p(t) + L(t)), S = 1 indicates symmetry. "G": Gini coefficient. 0 is perfect equality, values close to 1 indicate high inequality. "J": Youden index is the (largest) distance between the anti-diagonal and the curve. Distance is largest at the tangent point (J = max(p(t) - L(t))).

Value

lorenz returns an object of class lorenz. It is a matrix with n+1 rows (n = length(x)) and 3 columns (p, L, x).

The quantile method finds values of x_i corresponding to quantiles p_i. The plot method draws a Lorenz curve. The summary method returns characteristics of the Lorenz curve.

Author(s)

Peter Solymos <solymos@ualberta.ca>

References

Damgaard, C., & Weiner, J. (2000): Describing inequality in plant size or fecundity. Ecology 81:1139–1142. <doi:10.2307/177185>

Schisterman, E. F., Perkins, N. J., Liu, A., & Bondell, H. (2005): Optimal cut-point and its corresponding Youden index to discriminate individuals using pooled blood samples. Epidemiology 16:73–81. <doi:10.1097/01.ede.0000147512.81966.ba>

Youden, W. J. (1950): Index for rating diagnostic tests. Cancer 3:32–5. <doi:10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3>

See Also

quantile, order.

Examples

```
set.seed(1)
x <- c(rexp(100, 10), rexp(200, 1))

l <- lorenz(x)
head(l)
tail(l)
summary(l)

op <- par(mfrow=c(2,1))
plot(l, lwd=2, tangent=2, h=3, v=4)
abline(0, 1, lty=2, col="grey")
abline(1, -1, lty=2, col="grey")
plot(l, type="x", lwd=2, h=3, v=4)
par(op)
```

---

occolors *Color palettes for the opticut package*

Description

A convenient way of setting color palettes for the opticut package.

Usage

```
occolors(theme)
col2gray(col, method="BT.709")
```

Arguments

theme	character value, character vector, or a function used to interpolate the colors. The built-in values are "br" (blue-red divergent palette, colorblind safe), "gr" (green-red divergent palette), "bw" (black and white: grayscale converted "br" settings). See colorRampPalette, gray and the Examples.
col	vector of color specification as described on the help page for the col2rgb function. This is converted to grayscale.
method	character, the method used for grayscale conversion.

Details

Grayscale conversion methods in col2gray calculate gray levels based on red (R), green (G), and blue (B) color channels as follows: "BT.709": 0.2126 * R + 0.7152 * G + 0.0722 * B, luminosity correction following the ITU-R BT.709 recommendation; "BT.601": 0.299 * R + 0.587 * G + 0.114 * B, luminosity correction following the ITU-R BT.601 recommendation; "desaturate": (max(R, G, B) + min(R, G, B)) / 2, also called lightness; "average": (R + G + B) / 3; "maximum": max(R, G, B); "minimum": min(R, G, B); "red": R; "green": G; "blue": B.

Value

occolors returns a function, see colorRampPalette.

col2gray returns a vector of grey colors based on the conversion method and gray.

Author(s)

Peter Solymos <solymos@ualberta.ca>

Hexadecimal values for the built-in palettes are taken from http://colorbrewer2.org/.

See Also

colorRampPalette for a general description of palettes.

ocoptions for setting the color theme options in the opticut package.

Examples

```
## using palettes
plot(1:100, rep(2, 100), pch = 15,
    ylim = c(0, 21), axes = FALSE, ann = FALSE,
    col = occolors()(100)) # default 'bg'
text(50, 1, "theme = 'br'")
points(1:100, rep(5, 100), pch = 15,
    col=occolors("gr")(100))
text(50, 4, "theme = 'gr'")
points(1:100, rep(8, 100), pch = 15,
    col=occolors("bw")(100))
text(50, 7, "theme = 'bw'")
points(1:100, rep(11, 100), pch = 15,
    col=occolors(terrain.colors)(100))
text(50, 10, "theme = terrain.colors")
points(1:100, rep(14, 100), pch = 15,
    col=occolors(c("purple", "pink", "orange"))(100))
text(50, 13, "theme = c('purple', 'pink', 'orange')")
points(1:100, rep(17, 100), pch = 15,
    col=occolors(c("#a6611a", "#ffffbf", "#018571"))(100))
text(50, 16, "theme = c('#a6611a', '#ffffbf', '#018571')")
points(1:100, rep(20, 100), pch = 15,
    col=occolors(c("#7b3294", "#ffffbf", "#008837"))(100))
text(50, 19, "theme = c('#7b3294', '#ffffbf', '#008837')")

## grayscale conversions
```

```
n <- 25
col <- occolors("br")(n)
method <- c("BT.709", "BT.601",
    "desaturate", "average", "maximum", "minimum",
    "red", "green", "blue")
plot(0, type="n", ann=FALSE, axes=FALSE,
    xlim=c(0, n), ylim=c(3*length(method), 0))
for (j in 1:length(method)) {
    for (i in 1:n) {
        polygon(c(i-1, i, i, i-1), c(0, 0, 1, 1)+((j-1)*3),
            col=col[i], border=col[i])
        polygon(c(i-1, i, i, i-1), c(1, 1, 2, 2)+((j-1)*3),
            col=col2gray(col[i], method=method[j]),
            border=col2gray(col[i], method=method[j]))
        text(n/2, 1+((j-1)*3), method[j])
    }
}
```

`ocoptions` *Options for the opticut package*

Description

A convenient way of handling options related to the opticut package.

Usage

```
ocoptions(...)
```

Arguments

`...`	arguments in `tag = value` form, or a list of tagged values. The tags must come from the parameters described below.

Value

When parameters are set by `ocoptions`, their former values are returned in an invisible named list. Such a list can be passed as an argument to `ocoptions` to restore the parameter values. Tags are the following:

`collapse`	character value to be used when merging factor levels, the default is `"+"`.
`cut`	log likelihood ratio value, model/species with lower values are excluded from summaries and plots, the default is `2`.
`sort`	logical value indicating if species/partitions should be meaningfully sorted, the default is `TRUE`. It can take numeric value when only species (`1`) or partitions (`2`) are to be sorted (`1:2` is equivalent to `TRUE`, wile any other numeric value is equivalent to `FALSE`).
`theme`	the color theme to be used based on `occolors`, the default is `"br"`.

check_comb	check the design matrices for complementary partitions using checkComb, the default is TRUE.
try_error	if opticut should try to exclude species where the models failed (TRUE), the default is to stop when an error is encountered (FALSE).
scale	the scaling factor used to calculate indicator value (I) based on the estimated coefficient (b): I = abs(tanh(b*scale)), the default is 0.5.

Author(s)

Peter Solymos <solymos@ualberta.ca>

Examples

```
## simple example from Legendre 2013
## Indicator Species: Computation, in
## Encyclopedia of Biodiversity, Volume 4
## http://dx.doi.org/10.1016/B978-0-12-384719-5.00430-5
gr <- as.factor(paste0("X", rep(1:5, each=5)))
spp <- cbind(Species1=rep(c(4,6,5,3,2), each=5),
    Species2=c(rep(c(8,4,6), each=5), 4,4,2, rep(0,7)),
    Species3=rep(c(18,2,0,0,0), each=5))
rownames(spp) <- gr

## current settings
print(unlist(ocoptions())) # these give identical answers
unlist(getOption("ocoptions"))
summary(ocall <- opticut(spp ~ 1, strata=gr, dist="gaussian", comb="all"))

## resetting pboptions and checking new settings
ocop <- ocoptions(collapse="&", sort=FALSE)
unlist(getOption("ocoptions"))
## running again with new settings
summary(ocall <- opticut(spp ~ 1, strata=gr, dist="gaussian", comb="all"))

## resetting original
ocoptions(ocop)
unlist(getOption("ocoptions"))
```

---

opticut *Likelihood based optimal partitioning for indicator species analysis*

Description

The functions find the best binary partition for each species based on model selection. Possibly controlling for modifying/confounding variables. The general algorithm is described in Kemencei et al. 2014.

Usage

```
opticut1(Y, X, Z, dist = "gaussian", sset=NULL, ...)

opticut(...)
## Default S3 method:
opticut(Y, X, strata, dist = "gaussian",
    comb = c("rank", "all"), sset=NULL, cl = NULL, ...)
## S3 method for class 'formula'
opticut(formula, data, strata, dist = "gaussian",
    comb = c("rank", "all"), sset=NULL, cl = NULL, ...)

fix_levels(x, sep = "_")
strata(object, ...)
## S3 method for class 'opticut'
strata(object, ...)

## S3 method for class 'opticut'
bestmodel(object, which = NULL, ...)
## S3 method for class 'opticut'
bestpart(object, pos_only = FALSE, ...)
## S3 method for class 'opticut'
getMLE(object, which, ...)

wplot(x, ...)
## S3 method for class 'opticut1'
wplot(x, cut, ylim = c(-1, 1),
    las=1, ylab = "Model weight * Association", xlab = "Partitions",
    theme, mar = c(5, 4, 4, 4) + 0.1, bty = "o", ...)
## S3 method for class 'opticut'
wplot(x, which = NULL, cut, sort,
    las = 1, ylab = "Model weight * Association", xlab = "Partitions",
    theme, mar = c(5, 4, 4, 4) + 0.1, bty = "o", ...)
## S3 method for class 'opticut'
plot(x, which = NULL, cut, sort,
    las, ylab = "Relative abundance", xlab = "Strata",
    show_I = TRUE, show_S = TRUE, hr = TRUE, tick = TRUE,
    theme, mar = c(5, 4, 4, 4) + 0.1, bty = "o",
    lower = 0, upper = 1, pos = 0, horizontal=TRUE, ...)

## S3 method for class 'opticut1'
print(x, cut, sort, digits, ...)
## S3 method for class 'opticut'
print(x, digits, ...)
## S3 method for class 'summary.opticut'
print(x, cut, sort, digits, ...)
## S3 method for class 'opticut'
summary(object, ...)
```

```
## S3 method for class 'opticut'
as.data.frame(x,
    row.names = NULL, optional = FALSE, cut, sort, ...)
## S3 method for class 'summary.opticut'
as.data.frame(x,
    row.names = NULL, optional = FALSE, cut, sort, ...)
```

Arguments

formula	two sided model formula, response species data (matrix, or possible a vector for single species case) in the left-hand side, model terms for modifying effects in the right-hand side (its structure depending on the underlying functions). For example, in the most basic Gaussian case it can be `y ~ 1` (mo modifying variables) or `y ~ x` (with modifying variables). Centering the modifying terms (or choosing the origin wisely) is generally recommended (especially for Gaussian distribution where linear predictors are additive on the response scale) because the relative abundance contrast is estimated at the origin (0).
data	an optional data frame, list or environment containing the variables in the model. If not found in data, the variables are taken from `parent.frame()`, typically the environment from which `opticut` is called.
strata	vector (usually a factor), unique values define partitions. It can also be a matrix with rows as observations and binary partitions as columns.
dist	character or function, a distribution to fit. If character, it can follow one of these patterns: `"family"`, or `"family:link"` when appropriate (there is a `link` argument in the underlying function, or the link can be specified via the `family` argument). See Details and Examples.
comb	character, how to define the binary partitions. `"rank"` uses `rankComb`, `"all"` uses `allComb`.
sset	an optional vector specifying a subset of observations to be used in the fitting process. `NULL` means no subset taken.
cl	a cluster object, or an integer for multiple cores in parallel computations (this does nothing on Windows).
Y	numeric vector of observations for `opticut1`, vector or community matrix for `opticut.default`.
X	numeric, design matrix. Can be missing, in which case an intercept-only model is assumed.
Z	factor (must have at least 2 unique levels, this triggers `rankComb`), or a design matrix (custom matrix or as returned by `allComb`.
x, object	object to plot, print, summarize. For `fix_levels` it needs to be a factor.
cut	log likelihood ratio value to be used as a cut-off for showing species whose log likelihood ratio is not less than the cut-off.
sort	logical value indicating if species/partitions should be meaningfully sorted, the default is `TRUE`. It can take numeric value when only species (`1`) or partitions (`2`) are to be sorted (`1:2` is equivalent to `TRUE`, wile any other numeric value is equivalent to `FALSE`).

show_I	logical, if indicator values should be shown.
show_S	logical, if number of indicator species should be shown.
hr, tick	logical, if horizontal rules (hr) and ticks to the axis legends (tick) should be added. Default is TRUE for both.
theme	color theme as defined by occolors.
mar	numeric, graphical parameters for plot margin par.
ylab, xlab, las, ylim	graphical arguments, see plot. By default, las is 1 when horizontal = TRUE and 2 when horizontal = FALSE.
bty	Character, determines the type of box which is drawn about plots, see par.
lower, upper	numeric (between 0 and 1), lower is the minimum and upper is the maximum height for rectangles drawn in the plot. Both need to be in [0, 1] and higher cannot be smaller than lower.
pos	numeric, position of rectangles in the plot relative to the baseline. Value must be in the [-1, 1] range (below vs. above baseline).
horizontal	logical, plot orientation: species as rows (TRUE) or as columns (FALSE).
digits	numeric, number of significant digits in output.
which	numeric or character (can be a vector) defining a subset of species from the fitted object, or or NULL (all species, default).
sep	a character string to separate the sub-strings in factor levels.
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed. See as.data.frame.
optional	logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. See as.data.frame.
pos_only	logical, best partition normally returns the original variable without recognizing the direction of the association. pos_only = TRUE returns values where negative associations are taken into account and 1 indicates strata of positive association. This is only important when comb is not "rank".
...	other arguments passed to the underlying functions.

Details

Currently available distributions:

"gaussian" real valued continuous observations, e.g. biomass, uses lm of the stats package.

"poisson" Poisson count data, uses glm of the stats package.

"binomial" presence-absence (detection-nondetection) type data, uses glm of the stats package.

"negbin" overdispersed Negative Binomial count data, uses glm.nb of the MASS package.

"beta" continuous response in the unit interval, e.g. percent cover, uses betareg of the betareg package.

"zip" zero-inflated Poisson counts, indicative properties are tested as part of the abundance model, uses zeroinfl of the pscl package. The zero-inflation component refers to the probability of 0.

"zinb" zero-inflated Negative Binomial counts, indicative properties are tested as part of the abundance model, uses zeroinfl of the pscl package. The zero-inflation component refers to the probability of 0.

"zip2" zero-inflated Poisson counts, indicative properties are tested as part of the zero-model, uses zeroinfl of the pscl package. The zero-inflation component refers to the probability of 1 to be consistent with other methods regarding positive and negative effects.

"zinb2" zero-inflated Negative Binomial counts, indicative properties are tested as part of the zero-model, uses zeroinfl of the pscl package. The zero-inflation component refers to the probability of 1 to be consistent with other methods regarding positive and negative effects.

"ordered" response measured on ordinal scale, e.g. ordinal vegetation cover, uses polr of the MASS package.

"rsf" presence-only data using resource selection functions (RSF) as explained in rsf in the ResourceSelection package, assuming global availability (m = 0). The "rsf" works only for single species using opticut1 because 'presence-only' type data cannot be kept in a single matrix-like object for multiple species. Intercept only model is accepted for "rsf".

"rspf" presence-only data using resource selection probability functions (RSPF) as explained in rspf in the ResourceSelection package, assuming global availability (m = 0). The "rspf" works only for single species using opticut1 because 'presence-only' type data cannot be kept in a single matrix-like object for multiple species. Intercept only model is not accepted for "rspf".

Custom distributions can be defined, see Examples.

fix_levels is a utility function for replacing characters in factor levels that are identical to the value of the getOption("ocoptions")$collapse value. This case can lead to an error when specifying the strata argument, and the fix_levels can help.

Value

opticut1 returns an object of class opticut1, that is a modified data frame with additional attributes.

opticut returns an object of class opticut, that is a list with the following components:

"call" the function call.

"species" a list of species specific opticut1 objects.

"X" modifying variables as model matrix.

"Y" response, single species vector or matrix.

"strata" defines the partitions.

"nobs" sample size.

"sset" subset, if specified.

"nsplit" number of binary splits considered.

"dist" distribution.

"comb" combination type.

"failed" IDs for failed species models.

"collapse" character used for combining partition labels.

`fix_levels` returns a factor with modified levels.

The `strata` method extracts the `strata` argument as factor. The method finds unique row combinations when custom matrix is supplied for `strata`.

The `print` and `summary` methods are called for their side effects. The summary shows the following information: best supported split, strength and sign of association, indicator value (I), expected values (mu0, mu1), log likelihood ratio (logLR), and model weights(w).

The `plot` method presents the contrasts by species and strata.

The `wplot` (weight plot) shows model weights for partitions.

`bestpart` returns a matrix with the best supported partitions for each species (species as columns).

`bestmodel` returns the best supported model for further manipulation (e.g. prediction). Note: custom distribution functions are designed to return only point estimates, thus the best model cannot be returned. In this case, use the best partition returned by `bestpart` to refit the model. `getMLE` returns a named list corresponding to the best supported model. The list has the following elements: `coef` is the Maximum Likelihood Estimate (MLE), `vcov` is the variance-covariance matrix for the MLE, `dist` is the distribution inherited from input `object`.

The coercion method `as.data.frame` return a data frame.

Warning

The use of the `opticut1` function is generally discouraged: some of the internal checks are not guaranteed to flag issues when the formula-to-model-matrix translation is side-stepped (this is what is happening when the modifier variables are supplied as `X` argument in `opticut1`). Use the `opticut` with a single species instead.

Author(s)

Peter Solymos <solymos@ualberta.ca>

References

Kemencei, Z., Farkas, R., Pall-Gergely, B., Vilisics, F., Nagy, A., Hornung, E. & Solymos, P. (2014): Microhabitat associations of land snails in forested dolinas: implications for coarse filter conservation. Community Ecology 15:180–186. <doi:10.1556/ComEc.15.2014.2.6>

See Also

`allComb`, and `rankComb` for partitioning algorithms.

`beta2i` for indicator value (I) calculations in summaries.

`bestmodel`, `bestpart`, and `uncertainty` for manipulating fitted objects.

`ocoptions` on how to set some of the global options related to the presentation of the results in the package.

`optilevels` for finding the optimal number of factor levels.

Examples

```
## --- Gaussian
## simple example from Legendre 2013
## Indicator Species: Computation, in
## Encyclopedia of Biodiversity, Volume 4
## http://dx.doi.org/10.1016/B978-0-12-384719-5.00430-5
gr <- as.factor(paste0("X", rep(1:5, each=5)))
spp <- cbind(Species1=rep(c(4,6,5,3,2), each=5),
    Species2=c(rep(c(8,4,6), each=5), 4,4,2, rep(0,7)),
    Species3=rep(c(18,2,0,0,0), each=5))
rownames(spp) <- gr
spp

## all partitions
summary(ocall <- opticut(spp ~ 1, strata=gr, dist="gaussian", comb="all"))
summary(opticut(spp, strata=gr, dist="gaussian", comb="all")) # alternative

## rank based partitions
summary(ocrank <- opticut(spp ~ 1, strata=gr, dist="gaussian", comb="rank"))
summary(opticut(spp, strata=gr, dist="gaussian", comb="rank")) # alternative

## --- Binomial
## simulated binary data
set.seed(1234)
n <- 200
x0 <- sample(1:4, n, TRUE)
x1 <- ifelse(x0 %in% 1:2, 1, 0)
x2 <- rnorm(n, 0.5, 1)
p1 <- plogis(-0.5 + 2*x1 + -0.8*x2)
Y1 <- rbinom(n, 1, p1)
p2 <- plogis(-0.1 + 2*ifelse(x0==4,1,0) + -0.8*x2)
Y2 <- rbinom(n, 1, p2)
p3 <- plogis(-0.1 + -0.8*x2)
Y3 <- rbinom(n, 1, p3)
Y <- cbind(SPP1=Y1, SPP2=Y2, SPP3=Y3)
X <- model.matrix(~x2)

## all partitions, single species
Z <- allComb(x0)
opticut1(Y1, X, Z, dist="binomial")

## rank based partitions, single species
opticut1(Y1, X, as.factor(x0), dist="binomial")

## all partitions, multiple species
(m1 <- opticut(Y ~ x2, strata=x0, dist="poisson", comb="all"))
summary(m1)
## show all species
summary(m1, cut=0)
## plot best partitions and indicator values
plot(m1)
## model weights for all species
```

```
wplot(m1)
## different ways of plotting weights for single species
wplot(m1$species[[1]])
wplot(m1, which = 1)

## rank based partitions, multiple species
summary(m2 <- opticut(Y ~ x2, strata=x0, dist="poisson", comb="rank"))

## best partition
head(bestpart(m2))

## best model
mods <- bestmodel(m2)
mods
## explore further
str(predict(mods[[1]]))
confint(mods[[1]])

## MLE and variance-covariance matrix (species 1)
getMLE(m2, which = 1)

## Not run:
## --- Zero-inflated Negative Binomial
## dolina example
data(dolina)
## stratum as ordinal
dolina$samp$stratum <- as.integer(dolina$samp$stratum)
## filter species to speed up things a bit
Y <- dolina$xtab[,colSums(dolina$xtab > 0) >= 20]
## opticut results, note the cloglog link function
dol <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="zinb:cloglog")
summary(dol)
## vertical plot orientation
plot(dol, horizontal=FALSE, pos=1, upper=0.8)

## parallel computing comparisons
library(parallel)
cl <- makeCluster(2)
## sequential, all combinations (2^(K-1) - 1)
system.time(opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="zinb", comb="all", cl=NULL))
## sequential, rank based combinations (K - 1)
system.time(opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="zinb", comb="rank", cl=NULL))
## parallel, all combinations (2^(K-1) - 1)
system.time(opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="zinb", comb="all", cl=cl))
## parallel, rank based combinations (K - 1)
system.time(opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="zinb", comb="rank", cl=cl))
stopCluster(cl)
```

```
## --- Customizing distributions
## we may want to expand the Zero-inflation component in a ZIP model
## see how the return value needs to be structured
fun <- function(Y, X, linkinv, zi_term, ...) {
    X <- as.matrix(X)
    mod <- pscl::zeroinfl(Y ~ X-1 | zi_term, dist = "poisson", ...)
    list(coef=coef(mod),
        logLik=logLik(mod),
        linkinv=mod$linkinv)
}
Xdol <- model.matrix(~ stratum + lmoist + method, data=dolina$samp)
## this fits the null model (i.e. no partitions added)
fun(Y[,"amin"], Xdol, zi_term=dolina$samp$method)
## now we can use dist=fun
opticut1(Y[,"amin"], Xdol, Z=dolina$samp$mhab,
    dist=fun, zi_term=dolina$samp$method)
dol2 <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist=fun, zi_term=dolina$samp$method)
summary(dol2)

## End(Not run)

## current collapse value
getOption("ocoptions")$collapse
## factor levels sometimes need to be manipulated
## before feeding it to opticut
fix_levels(as.factor(c("A b", "C d")), sep=":")
fix_levels(as.factor(c("A b", "C d")), sep="")
```

`optilevels` *Optimal number of factor levels*

Description

Finds the optimal number of factor levels given the data and a model using a likelihood-based agglomerative algorithm.

Usage

```
optilevels(y, x, z = NULL, alpha = 0, dist = "gaussian", ...)

## S3 method for class 'optilevels'
bestmodel(object, ...)
```

Arguments

y	vector of observations.

x	a factor or a matrix of proportions (i.e. the values 0 and 1 should have consistent meaning across the columns, often through a unit sum constraint). It is the user's responsibility to ensure that values supplied for x are sensible. x is not expected to include an intercept.
z	a design matrix with predictor variables besides the one(s) defined via the argument x. It is the user's responsibility to ensure that values supplied for z are sensible and it also makes sense to bind x and z together. Variables in z should be centered (mean 0) (and possibly nomalized by SD), because the design matrix from x is not expected to include an intercept.
alpha	weighting factor for calculating information criteria for model selection (i.e. IC = (1-alpha)*AIC + alpha*BIC, also referred to as CAIC: consistent AIC).
dist	character, distribution argument passed to underlying functions, see listed on the help page of opticut (except for dist = "rsf" and dist = "rspf" which does not have a sound theoretical basis due to non-identifiability of piece-wise constant RSPF models).
object	fitted object.
...	other arguments passed to the underlying functions, see opticut.

Value

An object of class 'optilevels' that is a list with the following elements:

delta delta IC values along the selection path considering best models.

ic IC values along the selection path considering best models.

coef matrix of coefficients (linear predictor scale) corresponding to argument x along the selection path considering best models.

zcoef matrix of coefficients (linear predictor scale) corresponding to argument z when not NULL along the selection path considering best models, or NULL.

rank matrix ranks based on the coefficients along the selection path considering best models. Ranking uses the default ties.method = "average" in rank.

deltalist delta IC values along the selection path considering all competing models.

iclist IC values along the selection path considering all competing models.

coeflist matrix of coefficients (linear predictor scale) corresponding to argument x along the selection path considering all competing models.

zcoeflist matrix of coefficients (linear predictor scale) corresponding to argument z when not NULL along the selection path considering all competing models, or NULL.

ranklist matrix ranks based on the coefficients along the selection path considering all competing models.

levels list of (merged) factor levels along the selection path considering best models.

Y vector of observations (argument y).

X design matrix component corresponding to argument x.

Z design matrix component corresponding to argument z.

alpha weighting argument.

`dist` distribution argument.

`factor` logical, indicating if argument `x` is a factor (`TRUE`) or a matrix (`FALSE`).

`bestmodel` returns the best supported model for further manipulation (e.g. prediction).

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

`opticut` for finding best binary partitions.

`sindex` and `wrsi` for weighted relative suitability index.

Examples

```
## --- Factor levels with Gaussian distribution
## simple example from Legendre 2013
## Indicator Species: Computation, in
## Encyclopedia of Biodiversity, Volume 4
## http://dx.doi.org/10.1016/B978-0-12-384719-5.00430-5
gr <- as.factor(paste0("X", rep(1:5, each=5)))
spp <- cbind(Species1=rep(c(4,6,5,3,2), each=5),
    Species2=c(rep(c(8,4,6), each=5), 4,4,2, rep(0,7)),
    Species3=rep(c(18,2,0,0,0), each=5))
rownames(spp) <- gr
spp

ol <- optilevels(spp[,"Species3"], gr)
ol[c("delta", "coef", "rank", "levels")]

## get the final factor level
gr1 <- gr
levels(gr1) <- ol$level[[length(ol$level)]]
table(gr, gr1)

## compare the models
o0 <- lm(spp[,"Species3"] ~ gr - 1)
o1 <- lm(spp[,"Species3"] ~ gr1 - 1)
data.frame(AIC(o0, o1), delta=AIC(o0, o1)$AIC - AIC(o0))
ol$delta # should be identical

## --- Proportions with Poisson distribution
## simulation
set.seed(123)
n <- 500 # number of observations
k <- 5 # number of habitat types
b <- c(-1, -0.2, -0.2, 0.5, 1)
names(b) <- LETTERS[1:k]
x <- replicate(k, exp(rnorm(n)))
x <- x / rowSums(x) # proportions
X <- model.matrix(~.-1, data=data.frame(x))
```

```
lam <- exp(drop(X %*% b))
y <- rpois(n, lam)

z <- optilevels(y, x, dist="poisson")

## best model refit
bestmodel(z)

## estimates
plogis(z$coef)
plogis(b)
## optimal classification
z$rank

## get the final matrix
x1 <- mefa4::groupSums(x, 2, z$levels[[length(z$levels)]])
head(x)
head(x1)

## compare the models
m0 <- glm(y ~ x - 1, family="poisson")
m1 <- glm(y ~ x1 - 1, family="poisson")
data.frame(AIC(m0, m1), delta=AIC(m0, m1)$AIC - AIC(m0))
z$delta # should be identical

## Not run:
## dolina example with factor
data(dolina)
dolina$samp$stratum <- as.integer(dolina$samp$stratum)
y <- dolina$xtab[dolina$samp$method == "Q", "ppyg"]
x <- dolina$samp$mhab[dolina$samp$method == "Q"]
z <- scale(model.matrix(~ stratum + lmoist - 1,
    dolina$samp[dolina$samp$method == "Q",]))

## without additional covariates
dol1 <- optilevels(y, x, z=NULL, dist="poisson")
dol1$rank
summary(bestmodel(dol1))

## with additional covariates
dol2 <- optilevels(y, x, z, dist="poisson")
dol2$rank
summary(bestmodel(dol2))

## compare the two models
AIC(bestmodel(dol1), bestmodel(dol2))

## End(Not run)
```

---

rankComb *Ranking based binary partitions*

Description

Blindly fitting a model to all possible partitions is wasteful use of resources. Instead, one can rank the K partitions based on expected response values to explore only K-1 binary partitions along the gradient defined by the ranks of the expected values.

Usage

```
oComb(x, collapse)
rankComb(Y, X, Z, dist = "gaussian", collapse, ...)
```

Arguments

`Y`	numeric, vector of observations.
`X`	numeric, design matrix.
`Z`	factor, must have at least 2 unique levels.
`dist`	character, distribution argument passed to underlying functions, see listed on the help page of `opticut`.
`x`	and a numeric vector.
`collapse`	character, what to paste between levels. Defaults to `getOption("ocoptions")$collapse`.
`...`	other arguments passed to the underlying functions, see `opticut`.

Value

`oComb` returns the 'contrast' matrix based on the rank vector as input. Ranked from lowest to highest expected value among the partitions.

The function `rankComb` fits the model with multiple (K > 2) factor levels to find out the ranking, and returns a binary classification matrix as returned by `oComb` corresponding to the ranking.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

`allComb` for alternative partitioning algorithm.

`opticut` for the user interface.

Examples

```
## simulate some data
set.seed(1234)
n <- 200
x0 <- sample(1:4, n, TRUE)
x1 <- ifelse(x0 %in% 1:2, 1, 0)
x2 <- rnorm(n, 0.5, 1)
lam <- exp(0.5 + 0.5*x1 + -0.2*x2)
Y <- rpois(n, lam)
```

```
## binary partitions
head(rc <- rankComb(Y, model.matrix(~x2), as.factor(x0), dist="poisson"))
attr(rc, "est") # expected values in factor levels
aggregate(exp(0.5 + 0.5*x1), list(x0=x0), mean) # true values

## simple example
oComb(1:4, "+")
## using estimates
oComb(attr(rc, "est"))
```

`sindex` *Weighted relative suitability index*

Description

Calculates weighted relative suitability index.

Usage

```
sindex(y, x)
wrsi(y, x)
```

Arguments

`y`	matrix of observations for `sindex`, vector of observations for `wrsi`.
`x`	a matrix of proportions (i.e. the values 0 and 1 should have consistent meaning across the columns, often through a unit sum constraint).

Value

`wrsi` returns a data frame (class 'wrsi') with the following columns:

`WRSI` weighted relative suitability index, range (0- Inf).

`zWRSI` log of `WRSI` (z-transformed), range (-Inf, Inf).

`rWRSI` inverse Fisher z-transformed `zWRSI`, range (-1, 1).

`Pused` **and** `Pavail` total proportion of used (`y > 0`) and available of each feature (column) in `x`.

`Pw` weighted proportions from `y`.

`u` **and** `a` used and available totals for each feature (column) in `x`.

`sindex` returns a data frame (class 'sindex') with one column for each species, and one row for each feature (column) in `x`. Cell values are inverse Fisher z-transformed (`zWRSI`) indices.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

optilevels for finding optimal number of factor levels.

Examples

```
## --- habitat composition matrix
set.seed(1234)
n <- 1000 # sample size
k <- 5 # habitat classes
s <- runif(n, 1, 5)
p <- plogis(rnorm(n*k, 0, rep(s, k)))
p <- p*t(replicate(n, sample(c(10,4,2,1,1))))
x <- p / rowSums(p)
summary(x)
summary(rowSums(x))

## --- observations
## expected abundance in each habitat class
lam <- c(0.8, 0.6, 0.5, 0.4, 0.1)*1
## sample x habitat level abundances
yy <- t(sapply(seq_len(n), function(i) {
    ## intercept and modifier combined
    rpois(k, (x[i,]*lam))
    }))
## total: sum over habitat classes
## this is what we observe
y <- rowSums(yy)
colSums(yy)
table(y)

## --- wrsi calculations
(w <- wrsi(y, x))
op <- par(mfrow=c(1,2))
## habitat level observations are unknown
plot(lam, colSums(yy) / sum(yy), type="b")
## this is approximated by the wrsi
plot(lam, w$rWRSI, type="b")
abline(h=0, lty=2)
par(op)

## --- sindex calculations for multiple species
y2 <- cbind(Spp1=y, Spp2=rev(y), Spp3=sample(y))
(w2 <- sindex(y2, x))
heatmap(t(as.matrix(w2)), scale="none", col=occolors()(25))
```

---

uncertainty *Quantifying uncertainty for fitted objects*

Description

Quantifying uncertainty for fitted objects.

Usage

```
uncertainty(object, ...)
## S3 method for class 'opticut'
uncertainty(object,
    which = NULL, type = c("asymp", "boot", "multi"),
    B = 99, cl = NULL, ...)

check_strata(x, mat)
## S3 method for class 'uncertainty'
strata(object, ...)

## S3 method for class 'uncertainty'
bestpart(object, ...)
## S3 method for class 'uncertainty1'
bestpart(object, ...)

## S3 method for class 'uncertainty1'
print(x, ...)
## S3 method for class 'uncertainty'
print(x, ...)
## S3 method for class 'summary.uncertainty'
print(x, sort, digits, ...)
## S3 method for class 'uncertainty'
summary(object, level = 0.95, ...)

## S3 method for class 'uncertainty'
as.data.frame(x,
    row.names = NULL, optional = FALSE, sort, ...)
## S3 method for class 'summary.uncertainty'
as.data.frame(x,
    row.names = NULL, optional = FALSE, sort, ...)

## S3 method for class 'uncertainty1'
bsmooth(object, ...)
## S3 method for class 'uncertainty'
bsmooth(object, ...)
```

Arguments

object	fitted model object from opticut which does not contain extra arguments as part of ... (or an output from uncertainty for the summary method).
which	numeric or character (can be a vector) defining a subset of species from the fitted object, or or NULL (all species, default).

type	character, describing the type of uncertainty calculation for indicator value I, and expected values mu0, and mu1. "asymp": asymptotic distribution is based on best partition found for the input object (this option is unavailable for custom distribution functions). "boot": non-parametric bootstrap distribution based on best partition found for the input object. "multi": non-parametric bootstrap distribution based on best partition found for the bootstrap data (i.e. the model ranking is re-evaluated each time). "multi" works only if comb = "rank" in the opticut call.
B	numeric, number of iterations. For type = "boot" and type = "multi" it can be a user-supplied matrix with indices for resampling with dimensions length of observations times B.
cl	a cluster object, or an integer for multiple cores in parallel computations (this does nothing on Windows).
x	an object to be printed.
level	the confidence level required.
sort	logical value indicating if species should be meaningfully sorted, the default is TRUE.
digits	numeric, number of significant digits in output.
mat	a matrix with resampling indices (rows as samples, columns as iterations).
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed. See as.data.frame.
optional	logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. See as.data.frame.
...	other arguments passed to the underlying functions.

Value

uncertainty returns an object of class 'uncertainty'. The uncertainty element of the object is a list with species specific output as elements (object class 'uncertainty1'). Each 'uncertainty1' output is a data frame with 4 columns: best partition, indicator value I, and expected values mu0, and mu1.

check_strata returns a logical vector checking if all original strata from the input object are represented by resampling indices. Number of strata are attached as attributes for further diagnostics.

The summary method prints the name of the best supported split, selection frequency (R, reliability), indicator values (I, based on the distribution of values within the best supported split with highest reliability) and conficence interval for I (based on level).

bestpart finds the selection frequencies for strata as best partitions (number of strata times number of species).

The coercion method as.data.frame returns a data frame.

The bsmooth method returns bootstrap smoothed results for each strata.

Warning

Resampling methods can lead to complete exclusion of certain strata when sample size is small. Try revising the stratification of the input object, or provide custom resampling indices via the `B` argument using stratified (block) bootstrap, jackknife (leave-one-out), or similar techniques. Sometimes finding a suitable random seed via `set.seed` can resolve the issue.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

`opticut` for the user interface.

Examples

```
set.seed(2345)
n <- 50
x0 <- sample(1:4, n, TRUE)
x1 <- ifelse(x0 %in% 1:2, 1, 0)
x2 <- rnorm(n, 0.5, 1)
x3 <- ifelse(x0 %in% 2:4, 1, 0)
lam1 <- exp(0.5 + 1*x1 + -0.2*x2)
Y1 <- rpois(n, lam1)
lam2 <- exp(1 + 0.5*x3)
Y2 <- rpois(n, lam2)
Y3 <- rpois(n, exp(0))
Y <- cbind(Spp1=Y1, Spp2=Y2, Spp3=Y3)

oc <- opticut(Y ~ x2, strata=x0, dist="poisson", comb="rank")

## asymptotic confidence intervals
(uc1 <- uncertainty(oc, type="asymp", B=999))
summary(uc1)
## bootstrap-based confidence intervals
(uc2 <- uncertainty(oc, type="boot", B=19))
summary(uc2)

## use user-supplied indices
## multi-model bootstrap based uncertainties
B <- replicate(25, sample.int(n, replace=TRUE))
check_strata(oc, B) # check representation
(uc3 <- uncertainty(oc, type="multi", B=B))
summary(uc3)

## best partitions:
## selection frequencies for strata and species
bestpart(uc3)
heatmap(bestpart(uc3), scale="none", col=occolors()(25))

## bootstrap smoothed predictions per strata
bsmooth(uc3)
```

```
heatmap(bestpart(uc3), scale="none", col=occolors()(25))

## individual species results
uc3$uncertainty
bestpart(uc3$uncertainty[[1]])
bsmooth(uc3$uncertainty[[1]])

## block bootstrap
block_fun <- function()
    unlist(lapply(unique(x0), function(z) if (sum(x0==z) < 2)
        which(x0==z) else sample(which(x0==z), sum(x0==z), replace=TRUE)))
B <- replicate(25, block_fun())
check_strata(oc, B) # check representation
summary(uncertainty(oc, type="multi", B=B))

## jackknife
B <- sapply(1:n, function(i) which((1:n) != i))
check_strata(oc, B) # check representation
summary(uncertainty(oc, type="multi", B=B))

## Not run:
## dolina example
data(dolina)
## stratum as ordinal
dolina$samp$stratum <- as.integer(dolina$samp$stratum)
## filter species to speed up things a bit
Y <- ifelse(dolina$xtab[,colSums(dolina$xtab > 0) >= 20] > 0, 1, 0)
## opticut results, note the cloglog link function
dol <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="binomial:cloglog")

## parallel computing for uncertainty
library(parallel)
cl <- makeCluster(2)
ucdol <- uncertainty(dol, type="multi", B=25, cl=cl)
stopCluster(cl)

bestpart(ucdol)
heatmap(t(bestpart(ucdol)), scale="none", col=occolors()(25),
    distfun=function(x) dist(x, "manhattan"))

## See how indicator value changes with different partitions
## (and why it is the wrong metric to use in this calse)
with(ucdol$uncertainty[["pvic"]],
    boxplot(I ~ best, col="gold", ylab="Indicator value"))
## What we should calculate is the bootstrap smoothed mean of the
## expected value and its confidence intervals
bs <- bsmooth(ucdol$uncertainty[["pvic"]])
boxplot(t(bs), ylab="Expected value")
cbind(Mean=rowMeans(bs), t(apply(bs, 1, quantile, probs=c(0.025, 0.975))))

## End(Not run)
```

Index