Package ‘opticut’

February 1, 2018

Type Package

Title Likelihood Based Optimal Partitioning and Indicator Species Analysis

Version 0.1-2

Date 2018-02-01

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

Maintainer Peter Solymos <solymos@ualberta.ca>

Description Likelihood based optimal partitioning and indicator species analysis. Finding the best binary partition for each species based on model selection, with the possibility to take into account modifying/confounding variables as described in Kemencei et al. (2014) <doi:10.1556/ComEc.15.2014.2.6>. The package implements binary and multi-level response models, various measures of uncertainty, Lorenz-curve based thresholding, 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 (>= 0.3-2), parallel, mefa4

License GPL-2

LazyLoad yes

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-01 16:10:17 UTC

R topics documented:

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

Description

Likelihood based optimal partitioning and indicator species analysis. Finding the best binary partition for each species based on model selection, with the possibility to take into account modifying/confounding variables as described in Kemencei et al. (2014) <doi:10.1556/ComEc.15.2014.2.6>. The package implements binary and multi-level response models, various measures of uncertainty, Lorenz-curve based thresholding, with native support for parallel computations.

Details

The DESCRIPTION file:

Package:	opticut
Type:	Package
Title:	Likelihood Based Optimal Partitioning and Indicator Species Analysis
Version:	0.1-2
Date:	2018-02-01
Author:	Peter Solymos [cre, aut], Ermias T. Azeria [ctb]
Maintainer:	Peter Solymos <solymos@ualberta.ca>
Description:	Likelihood based optimal partitioning and indicator species analysis. Finding the best binary partition for each
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 (>= 0.3-2), 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                  Scaling for the Indicator Potential
birdrec                 Bird Species Detections
dolina                  Land Snail Data Set
lorenz                  Lorenz Curve Based Thresholds and Partitions
multicut                Multi-level Response Model
occolors                Color Palettes for the opticut Package
ocoptions               Options for the opticut Package
opticut                 Optimal Binary Response Model
opticut-package         Likelihood Based Optimal Partitioning and
                        Indicator Species Analysis
optilevels              Optimal Number of Factor Levels
rankComb                Ranking Based Binary Partitions
uncertainty             Quantifying Uncertainty for Fitted Objects
```

The main user interface are the `opticut` and `multicut` functions to find the optimal binary or multi-level response models. Make sure to evaluate `uncertainty`. `optilevels` 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

mc <- multicut(formula = y ~ 1, strata = g, dist = "poisson")
summary(mc)

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 strata.

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 levels (strata) 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` 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

opticut for the user interface.

rankComb and lorenz for alternative partitioning algorithms.

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

Generic functions for 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`, `multicut`, `uncertainty`.

`beta2i` *Scaling for the Indicator Potential*

Description

Transformation of estimated contrasts to indicator potential.

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` and `multicut` use the scaled I values as indicator potential.

`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])
}
```

`birdrec` *Bird Species Detections*

Description

Data set listing 156 species (mostly birds, few amphibians and mammals) detected at 127 sites (367 point locations) in Alberta, Canada in 2015, using autonomous recording technology (ARU; Wildlife Acoustic Song Meter) for sound recordings.

Usage

```
data("birdrec")
```

Format

A list with 3 elements with matching ordering: `xtab` is a sample x species matrix with number of detections, `samp` is a data frame with sample level attributes. `taxa` is a data frame with species level attributes.

Multiple random recordings at each location were selected according to a stratified random design (based on combination of `TOY` and `TOD`). These recordings were listened to by trained analysts and species were identified based on auditory cues.

This data set lists detections from the first 1-minute segment of each recording. Dates for the 3967 1-minute segments range between 2015-03-31 and 2015-07-29. Variables in `birdrec$samp` are the following:

`PKEY`: primary key for location/time combinations.

`POINT`: unique spatial location IDs, each point had its own ARU unit.

`SITE`: site ID (1-4 ARU units deployed per site).

`YEAR`: year, 2015.

`MONTH`: month from 3 (March) to 7 (July).

`MDAY`: day of month, 1-31.

`HOUR`: 24-hour of day, values between 0-12.

`MINUTE`: minute, 0-59.

`YDAY`: ordinal day of the year, 89-209.

`RAIN`, `WIND`, `INDUSTRY`, `NOISE`: level of rain, wind, industrial noise, and background noise. 0 = no; 1 = light; 2 = moderate; 3 = heavy.

`MICROPHONE`: Every recording contains a certain level of background static due to the pre-amplifiers; however, problems, such as, electrostatic discharge on the microphones, faulty wiring, poorly installed microphones and/or missing microphones can occur causing excess static or dead channels. 0 = no microphone related issues; 1 = left microphone cuts out intermittently; 2 = right microphone cuts out intermittently; 3 = both microphones cut out intermittently; 4 = left channel failed; 5 = right channel failed; 6 = both channels failed (no cases in the data set); 7 = left side extra static; 8 = right side extra static; 9 = both sides extra static; 10 = other issues; 11 = unbalanced channels.

TOY: time of year intervals used for stratified random selection of dates. 8 intervals divided into 3 major units (early, mid, and late breeding season; YDAY 140 and 180 were used as threshold between the major units).

TOD: time of day, midnight (HOUR = 0) or morning (HOUR > 0).

Variables in birdrec$taxa are the following: Species, CommonName, ScientificName, Family, Order, Class, and MigratoryBehaviour.

Methodology and metadata is described in ABMI (2016), and Lankau et al. (2015).

Source

Alberta Biodiversity Monitoring Institute (ABMI, www.abmi.ca)

References

Alberta Biodiversity Monitoring Institute (ABMI), 2016. Terrestrial field data collection protocols (abridged version) 2016-05-18. Alberta Biodiversity Monitoring Institute; Edmonton, Alberta, Canada.

Lankau, H.E., MacPhail, A., Knaggs, M. & Bayne, E., 2015. Acoustic recording analysis protocol. Bioacoustic Unit, University of Alberta, and Alberta Biodiversity Monitoring Institute; Edmonton, Alberta, Canada.

Examples

```
data(birdrec)
str(birdrec)

aggregate(rowSums(birdrec$xtab),
    list(TOY=birdrec$samp$TOY, TOD=birdrec$samp$TOD), mean)
boxplot(rowSums(birdrec$xtab) ~ TOD + TOY, birdrec$samp,
    col=c("gold", "tomato"), ylab="# detections")

## Not run:
y <- ifelse(birdrec$xtab > 0, 1, 0)
g <- paste0(gsub("[[:digit:]]", "", as.character(birdrec$samp$TOY)),
    substr(as.character(birdrec$samp$TOD), 1, 4))
g <- factor(g, levels=c("EarlyMorn", "MidMorn", "LateMorn",
    "EarlyMidn", "MidMidn", "LateMidn"))
## binary response model
oc <- opticut(y ~ 1, strata=g, dist="binomial")
## multi-level response model
mc <- multicut(y ~ 1, strata=g, dist="binomial")

## testing equality of labels
splito <- as.character(summary(oc)$summary$split)
splitm <- as.character(summary(mc)$summary$split)
table(splito == splitm)
## seeing how much those differ
bpo <- summary(oc)$bestpart
bpm <- summary(mc)$bestpart
rs <- rowSums(abs(bpo-bpm))
```

```
table(rs)
10 * bpo[rs > 0,] + bpm[rs > 0,]

## End(Not run)
```

`dolina` *Land Snail Data Set*

Description

A comprehensive and micro-scale land snail data set from 16 dolinas 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 for those microhabitats.

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 presented here.

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 Based Thresholds and Partitions*

Description

Lorenz curve based thresholds and partitions.

Usage

```
lorenz(x, n = rep(1, length(x)), na.last = TRUE)

## S3 method for class 'lorenz'
quantile(x, probs = seq(0, 1, 0.25),
    type = c("L", "p"), ...)
iquantile(x, ...)
## S3 method for class 'lorenz'
iquantile(x, values,
    type = c("L", "p"),...)

## 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.
`n`	a vector of frequencies, must be same length as `x`.
`na.last`	logical, for controlling the treatment of `NA`s. 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`.
`values`	numeric vector of values for which the corresponding population quantiles are to be returned.
`type`	character. For the `plot` method it indicates whether to plot the cumulative distribution quantiles (`"L"`) or ordered but not-cumulated values (`"x"`). For the `quantile` and `iquantile` methods it indicates which of the quantiles (`"L"` or `"p"`) to use.
`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 abundance (income, or wealth). Cumulative portion of the population: p_i = i / m (i=1,...,m), vs. cumulative portion of abundance: L_i = sum_j=1^i x_j * n_j / sum_j=1^n x_j * n_j. where x_i are indexed in non-decreasing order (x_i <= x_i+1). By convention, p_0 = L_0 = 0. `n` can represent unequal frequencies.

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, x_t is the unmodified input. `"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 - L) = p_t - L_t).

Value

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

The `quantile` method finds values of x_i corresponding to quantiles L_i or p_i (depending on the `type` argument). The `iquantile` (inverse quantile) method finds quantiles of L_i or p_i corresponding to values of x_i.

The `plot` method draws a Lorenz curve. Because the object is a matrix, `lines` and `points` will work for adding multiple lines.

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)
summary(unclass(l))

(q <- c(0.05, 0.5, 0.95))
(p_i <- quantile(l, probs=q, type="p"))
iquantile(l, values=p_i, type="p")
(p_i <- quantile(l, probs=q, type="L"))
iquantile(l, values=p_i, type="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)

## Lorenz-tangent approach to binarize a multi-level problem
n <- 100
g <- as.factor(sort(sample(LETTERS[1:4], n, replace=TRUE, prob=4:1)))
x <- rpois(n, exp(as.integer(g)))
mu <- aggregate(x, list(g), mean)
```

```
(l <- lorenz(mu$x, table(g)))
(s <- summary(l))

plot(l)
abline(0, 1, lty=2)
lines(rep(s["p[t]"], 2), c(s["p[t]"], s["L[t]"]), col=2)
```

---

multicut *Multi-level Response Model*

Description

The functions fits the multi-level response model for each species, possibly controlling for modifying/confounding variables.

Usage

```
multicut1(Y, X, Z, dist = "gaussian", sset=NULL, ...)

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

## S3 method for class 'multicut'
bestmodel(object, which = NULL, ...)
## S3 method for class 'multicut'
bestpart(object, ...)
## S3 method for class 'multicut'
strata(object, ...)
## S3 method for class 'multicut'
getMLE(object, which, vcov=FALSE, ...)
## S3 method for class 'multicut'
subset(x, subset=NULL, ...)
## S3 method for class 'multicut'
fitted(object, ...)
## S3 method for class 'multicut'
predict(object, gnew=NULL, xnew=NULL, ...)

## S3 method for class 'multicut'
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 'multicut1'
plot(x,
    ylab = "Relative abundance", xlab = "Strata", ...)
lcplot(x, ...)
## S3 method for class 'multicut1'
lcplot(x,
    ylab="Cumulative abundance", xlab="Strata",
    bty = "o", theme, ...)

## S3 method for class 'multicut1'
print(x, digits, ...)
## S3 method for class 'multicut'
print(x, digits, ...)
## S3 method for class 'summary.multicut'
print(x, cut, sort, digits, ...)
## S3 method for class 'multicut'
summary(object, ...)

## S3 method for class 'multicut'
as.data.frame(x,
    row.names = NULL, optional = FALSE, cut, sort, ...)
## S3 method for class 'summary.multicut'
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` (no 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 contrasts are 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 `multicut` is called.
`strata, Z`	a factor, unique values define strata (must have at least 2 unique levels, empty levels are dropped).
`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 on `opticut` page and Examples.
`sset`	an optional vector specifying a subset of observations (rows) 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 (integer value for forking is ignored on Windows).
`Y`	numeric vector of observations for `multicut1`, vector or community matrix for `multicut.default`.
`X`	numeric, design matrix for possible confounding/modifier variables. Can be missing, in which case an intercept-only model is assumed.
`x, object`	object to plot, print, summarize.
`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`).
`show_I`	logical, if indicator potential (I) 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`	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 around 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 `NULL` (all species, default).
`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`.
`subset`	logical, numeric, or character index indicating species to keep, missing values are not accepted. The default `NULL` returns the original object without subsetting.
`vcov`	logical, if variance-covariance matrix is to be returned.
`gnew, xnew`	new values for `strata` and modifiers (right-hand-side of formula) to predict for, or `NULL`.
`...`	other arguments passed to the underlying functions.

Value

`multicut1` returns an object of class 'multicut1'.

`multicut` returns an object of class 'multicut', that is a list with the following components:

`"call"`	the function call.
`"species"`	a list of species specific multicut1 objects.
`"X"`	modifying variables as model matrix.
`"Y"`	response, single species vector or matrix.
`"strata"`	defines the stratification.
`"nobs"`	sample size.
`"sset"`	subset, if specified.
`"dist"`	distribution.
`"failed"`	IDs for failed species models dropped from results list.

The `strata` method extracts the `strata` argument as factor.

The `print` and `summary` methods are called for their side effects showing expected values, and log likelihood ratio (logLR). Optimal binary partitions are determined as part of the summary based on Lorenz-tangent based thresholding, which requires nonnegative expected values. Indicator potential (I) is based on largest the contrast (difference) between the minimum and maximum estimates on the linear predictor (link) scale.

The `subset` method subsets the species in the multicut object.

The `plot` method presents the estimates by species and strata. The `lcplot` method plots the Lorenz curve for a single species 'multicut1' object.

`bestpart` returns a matrix with the best supported partitions for each species (samples and rows, species as columns). Binary partitions are based on Lorenz-tangent based optimal threshold (see `lorenz`). `lorenz` requires nonnegative fitted values which is not guaranteed under `dist = "gaussian"` with identity link, see `fix_fitted` `ocoptions` setting for how to resolve this (choosing a different link function, distribution, or centering modified variables is advised).

`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 or `NULL`, `dist` is the distribution inherited from input `object`.

`fitted` returns expected values on the predictor scale for the observations as a matrix (number of observations by number of species). `predict` returns `fitted` values when both `gnew` and `xnew` are `NULL`, or corresponding point predictions (expected values) on the predictor scale.

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

Warning

The use of the `multicut1` 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 `multicut1`). Use the `multicut` function with a single species instead.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

lorenz Examples for how multi-level partitions are binarized using the Lorenz-tangent approach.

opticut for optimal binary response model, optilevels for finding the optimal number of factor levels.

beta2i for indicator potential (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 and how errors encountered during model fitting are handled.

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
## must add some noise to avoid perfect fit
spp[6, "Species1"] <- 7
spp[1, "Species3"] <- 17
spp

## negative expected values are not good
oco <- ocoptions(fix_fitted=TRUE)
summary(ocall <- multicut(spp ~ 1, strata=gr, dist="gaussian"))
summary(multicut(spp, strata=gr, dist="gaussian")) # alternative
ocoptions(oco) # reset options

## --- Binomial
## simulated binary data
set.seed(1234)
n <- 200
x0 <- sample(1:4, n, TRUE)
x1 <- ifelse(x0 <= 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)

(m0 <- multicut1(Y1, X, as.factor(x0), dist="binomial"))
lcplot(m0)

summary(m1 <- multicut(Y ~ x2, strata=x0, dist="poisson"))
plot(m1)

## subset results
summary(subset(m1, 1:2))

## best partition
head(bestpart(m1))

## best model
mods <- bestmodel(m1)
mods
## explore further
confint(mods[[1]])

## MLE and variance-covariance matrix (species 1)
getMLE(m1, which = 1, vcov=TRUE)

## fitted values
head(fitted(m1))
## prediction for new data
head(predict(m1, gnew=x0, xnew=data.frame(x2=x2)))

## 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 <- multicut(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
library(parallel)
cl <- makeCluster(2)
multicut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="zip",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
multicut1(Y[,"amin"], Xdol, Z=dolina$samp$mhab,
    dist=fun, zi_term=dolina$samp$method)
dol2 <- multicut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist=fun, zi_term=dolina$samp$method)
summary(dol2)

## End(Not run)
```

`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"` (default, 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. See Details.

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 gray 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/.

Converting color to grayscale: https://en.wikipedia.org/wiki/Grayscale

See Also

colorRampPalette for a general description of palettes.

ocoptions for setting the color theme option 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).
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 and multicut 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 potential (I) based on the estimated contrast (x): I = abs(tanh(x*scale)), the default is 0.5.
fix_fitted	bestpart.multicut uses lorenz which requires nonnegative fitted values, however models with identity link can lead to negative expected values. When TRUE the fitted values (x) are adjusted as x' = x + abs(min(x)) to ensure nonnegativity. The default is FALSE.
robust_loglik	if ill-defined models resulting in perfect fit (infinite log likelihood, or NA, NaN) should be allowed. The default TRUE makes such ill-defined log likelihoods a very small real number -(.Machine$double.xmax^(1/3)). FALSE is equivalent to allowing every model to safeguard against such cases or not.

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
## must add some noise to avoid perfect fit
spp[6, "Species1"] <- 7
spp[1, "Species3"] <- 17
spp

## 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 *Optimal Binary Response Model*

Description

The functions fits the multi-level response model for each species by finding the best binary partition 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, vcov=FALSE, ...)
## S3 method for class 'opticut'
subset(x, subset=NULL, ...)
## S3 method for class 'opticut'
fitted(object, ...)
## S3 method for class 'opticut'
predict(object, gnew=NULL, xnew=NULL, ...)

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` (no 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 (must have at least 2 unique levels, empty levels are dropped). 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 (rows) 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 (integer value for forking is ignored 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`).
`show_I`	logical, if indicator potential (I) 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 around 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 `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"`.
`subset`	logical, numeric, or character index indicating species to keep, missing values are not accepted. The default `NULL` returns the original object without subsetting.
`vcov`	logical, if variance-covariance matrix is to be returned.
`gnew, xnew`	new values for `strata` and modifiers (right-hand-side of formula) to predict for, or `NULL`. Predicting for new strata available for `comb = "rank"` models only.
`...`	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. Identity link is assumed. Centering modified variables is generally advised to avoid negative expected values when the response is nonnegative.

`"poisson"` Poisson count data, uses `glm` of the stats package. Exponential (log) link is assumed.

`"binomial"` presence-absence (detection-nondetection) type data, uses `glm` of the stats package. Logistic (logit) link is assumed.

`"negbin"` overdispersed Negative Binomial count data, uses `glm.nb` of the MASS package. Exponential (log) link is assumed.

`"beta"` continuous response in the unit interval (0-1), e.g. percent cover, uses `betareg` of the betareg package. Logistic (logit) link for the mean model is assumed.

`"zip"` zero-inflated Poisson counts, indicative properties are tested as part of the abundance model, uses `zeroinfl` of the pscl package. Exponential (log) link is used for count based analysis, the second part of the `dist` argument following the colon is used as link function for the zero component (logistic link assumed).

`"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. Exponential (log) link is used for count based analysis, the second part of the `dist` argument following the colon is used as link function for the zero component (logistic link assumed).

`"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. Logistic (logit) link is assumed for zero-nonzero based analysis, only symmetric link functions (logit, probit) allowed. Exponential (log) link is used for the count data part which cannot be changed.

`"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. Logistic (logit) link is assumed for zero-nonzero based analysis, only symmetric link functions (logit, probit) allowed. Exponential (log) link is used for the count data part which cannot be changed.

`"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 (i.e. no modifier variables in right-hand-side of the formula) is accepted for `"rsf"`. Exponential (log) link is assumed.

`"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"`, need to have at least one continuous modifier variable for identifiability (see Solymos & Lele 2016). Logistic (logit) link is assumed.

Custom distributions can be defined, see Examples. Note: not all downstream algorithms and methods work with custom distributions.

`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 dropped from results list.

`"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 potential (I), expected values (mu0, mu1), log likelihood ratio (logLR), and model weights(w).

The `subset` method subsets the species in the opticut object.

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 (samples and rows, 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 or `NULL`, `dist` is the distribution inherited from input `object`.

`fitted` returns expected values on the predictor scale for the observations as a matrix (number of observations by number of species). `predict` returns `fitted` values when both `gnew` and `xnew` are `NULL`, or corresponding point predictions (expected values) on the predictor scale (available for `comb = "rank"` models only).

The coercion methods `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> and Ermias T. Azeria

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. & Lele, S. R. (2016): Revisiting resource selection probability functions and single-visit methods: clarification and extensions. Methods in Ecology and Evolution 7:196–205. <doi:10.1111/2041-210X.12432>

See Also

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

`beta2i` for indicator potential (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 and how errors encountered during model fitting are handled.

`multicut` for multinomial-response model, `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
## must add some noise to avoid perfect fit
spp[6, "Species1"] <- 7
spp[1, "Species3"] <- 17
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 <= 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"))
## subset results
summary(subset(m2, 1:2))

## best partition
head(bestpart(m2))

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

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

## fitted values
head(fitted(m2))
## prediction for new data
head(predict(m2, gnew=x0, xnew=data.frame(x2=x2)))

## 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 normalized by SD), because the design matrix from `x` is not expected to include an intercept.
`alpha`	numeric [0-1], 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 = "zip2"`, `dist = "zinb2"` `dist = "rsf"`, and `dist = "rspf"`).
`object`	fitted object.
`...`	other arguments passed to the underlying functions, see `opticut1`.

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 and multicut for fitting best binary and multi-level response models.

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
## must add some noise to avoid perfect fit
spp[6, "Species1"] <- 7
spp[1, "Species3"] <- 17
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(crossprod(t(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 levels (strata) 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"))
```

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

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

## 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 (which should 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. See Details.
`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 (integer value for forking is ignored 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`.

`subset`	logical, numeric, or character index indicating species to keep, missing values are not accepted.
`...`	other arguments passed to the underlying functions.

Details

Uncertainty is calculated for indicator potential `I`, and expected values (`mu0`, and `mu1` for opticut, and `mu_*` for multicut objects).

`"asymp"`: asymptotic distribution is based on best supported model (this option is unavailable for custom distribution functions because it requires the Hessian matrix). This type is available for both opticut and multicut objects.

`"boot"`: non-parametric bootstrap distribution based on best partition found for the input object. This type is available for both opticut and multicut objects.

`"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. This type is not available for multicut objects.

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 columns: `best` partition, indicator potential `I`, and expected values (`mu0`, and `mu1` for opticut, and `mu_*` for multicut objects).

`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 confidence interval for I (based on `level`).

The `subset` method subsets the species in the uncertainty object.

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

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

The `bsmooth` method returns bootstrap smoothed results for each strata (not available for multicut based uncertainty objects, check uncertainty results instead).

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. Finding a suitable random seed via `set.seed` or dropping unsuitable iterations can also resolve the issue.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

opticut and multicut for the user interface of the input objects.

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]])

## Not run:
## 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))

## multicut based uncertainty
mc <- multicut(Y ~ x2, strata=x0, dist="poisson")

## asymptotic confidence intervals
(muc1 <- uncertainty(mc, type="asymp", B=999))
summary(muc1)
bestpart(muc1)

## bootstrap-based confidence intervals
(muc2 <- uncertainty(mc, type="boot", B=19))
summary(muc2)
bestpart(muc2)

## 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))))
```

```
## A more interesting simulated example for bootstrap smoothing
## and comparing opticut vs. multicut
set.seed(1)
n <- 2000
x <- sort(runif(n, -8, 8))
p <- plogis(0.5 + -0.1 * x + -0.2 * x^2)
y <- rbinom(n, 1, p)
d <- diff(range(x))/10
br <- seq(min(x), max(x), by=d)
g <- cut(x, br, include.lowest=TRUE)
levels(g) <- LETTERS[1:nlevels(g)]
o <- opticut(y ~ 1, strata=g, dist="binomial")
m <- multicut(y ~ 1, strata=g, dist="binomial")
library(parallel)
cl <- makeCluster(2)
uo <- uncertainty(o, type="multi", B=99, cl=cl)
um <- uncertainty(m, type="boot", B=99, cl=cl)
stopCluster(cl)
## bootstrap average for opticut
bs <- bsmooth(uo$uncertainty[[1]])
stat <- cbind(Mean=rowMeans(bs),
    t(apply(bs, 1, quantile, probs=c(0.025, 0.975))))
## bootstrap average for multicut
bsm <- as.matrix(um$uncertainty[[1]][,-(1:2)])
statm <- cbind(Mean=colMeans(bsm),
    t(apply(bsm, 2, quantile, probs=c(0.025, 0.975))))

op <- par(mfrow=c(2,1))
plot(p ~ x, type="l", ylim=c(0,1), main="Binary partitions (opticut)")
abline(v=br, col="grey", lty=3)
lines(br[-1]-0.5*d, stat[,1], col=4)
lines(br[-1]-0.5*d, stat[,2], col=4, lty=2)
lines(br[-1]-0.5*d, stat[,3], col=4, lty=2)
lines(br[-1]-0.5*d, bs[,1], col=2)
legend("topright", bty="n", lty=c(1,1,2,1), col=c(1,4,4,2),
    legend=c("True response","bsmooth","0.95 CI","Best partition"))

plot(p ~ x, type="l", ylim=c(0,1), main="Multi-level model (multicut)")
abline(v=br, col="grey", lty=3)
lines(br[-1]-0.5*d, statm[,1], col=4)
lines(br[-1]-0.5*d, statm[,2], col=4, lty=2)
lines(br[-1]-0.5*d, statm[,3], col=4, lty=2)
legend("topright", bty="n", lty=c(1,1,2), col=c(1,4,4),
    legend=c("True response","bsmooth","0.95 CI"))
par(op)

## End(Not run)
```

Index