opticut: likelihood-based optimal partitioning for indicator species analysis (supporting information)

Peter Solymos and Ermias T. Azeria

December 19, 2016

Contents

1 Introduction

Identifying and monitoring indicator species has long been considered a cost-effective way of tracking environmental change or the status of the biota. Examples include the characterization of vegetation types (Chytry et al. 2002), degradation of ecosystems (McGeoch & Chown 1988), or signalling cryptic or rare species (Halme et al. 2009). Throughout these examples, a key attribute of indicator species (also referred to as character or differential species) is that they have strong associations with the environmental variables that they are supposed to indicate.

Approaches to quantify the degree of environmental associations for species (indicator value) traditionally falls into three major types of approaches:

1. contingency table based measures (De Caceres & Legendre 2009);
2. analysis of variance (ANOVA; Wildi & Feldmeyer-Christe 2013); and
3. the widely used non-parametric IndVal method (Dufrene & Legendre 1997).

While the different approaches have strong appeal and applications, they do not always meet the challenges presented by ecological data.

Ecological data come in different forms: binary, ordinal, count, abundance, or presence only data. Some of these data types are suitable for a particular approach, while some formats need 'tweaking'. For example, binarizing abundance or count data for contingency tables leads to information loss. ANOVA, on the other hand, implies normality and homoscedastic errors, which might not always be satisfied by 0/1, ordinal, skewed, or percent cover data. Finally, randomization test for the IndVal approach requires count data, which renders hypothesis testing difficult if not impossible for continuous or ordinal data.

Another staple of observation field studies is the presence of modifying or confounding variables, or the presence of systematic biases (variable sampling effort, imperfect detectability, sample selection bias). Ignoring these effects can lead to erroneous indicator species analysis (Zettler et al. 2013). Controlling for these effects can improve the assessment of species-environment relationships, thus lead to better evaluation of indicator species.

To address these limitations, Kemencei et al. (2014) proposed a model-based indicator species analysis that accounted for the effects of modifying variables, and non-independence in the data due to paired sampling design. This model-based approach has been generalized and made available in the opticut R extension package. The opticut package offers computationally efficient and extensible algorithms for finding indicator species, tools for exploring and visualizing the results, and quantifying uncertainties. This manual showcases the functionality of the package.

1.1 Install

The opticut R package can be installed from the Comprehensive R Archive Network (CRAN) as:

```r
install.packages("opticut")
```

Install development version from GitHub:

```r
library(devtools)
install_github("psolymos/opticut")
```

User visible changes in the package are listed in the NEWS file.

1.2 Report a problem

Use the issue tracker to report a problem.

1.3 License

GPL-2

1.4 Loading the package

To get started, open R and load the **opticut** package as:

```
library(opticut)
```

```
## Loading required package: pbapply
```

```
## opticut 0.1-0     2016-12-16
```

2 Patritioning

Optimal partitioning (optimal cut, or in short: opticut) is found for each species independent of each other. We make observations (y_i) of possibly multiple species at $i = 1, ..., n$ sites. Now let us consider a discrete site descriptor (g_i) with K levels or strata $(k = 1, \dots, K; K > 2)$. This stratification might come from remotely sensed or other geospatial information, field measurements, or from multivariate clustering. We can use g_i to create $m = 1, ..., M$ possible binary partitions based on coding one or more levels as 1s and the rest with 0s. We denote any such possible partition as $z^{(m)}$. The total number of binary partitions is $M = 2^{K-1} - 1$, not counting cases when 0s or 1s are completely missing (which is the null model). The opticut method, as opposed to for example IndVal method is invariant to the coding of 0s and 1s in $z^{(m)}$. This means that complementary cases, such as $z^{(m)}$ and $1 - z^{(m)}$, are treated as interchangeable. The opticut package provides utility functions to create and check binary partitions from multi-level vectors (`kComb`, `allComb`, `checkComb`).

2.1 Manipulating partitions

Finding all combinations does not require a model or observed responses. It only takes a classification vector with $K > 1$ strata. The `kComb` function returns a 'contrast' matrix corresponding to all possible binary partitions of the factor with K levels:

```
kComb(k = 2)
```

```
##      [,1]
## [1,]    1
## [2,]    0
```

```r
kComb(k = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

```r
kComb(k = 4)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]    1    0    0    0    1    1    1
## [2,]    0    1    0    0    1    0    0
## [3,]    0    0    1    0    0    1    0
## [4,]    0    0    0    1    0    0    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):

```r
## finding all combinations
(f <- rep(LETTERS[1:4], each=2))
```

```
## [1] "A" "A" "B" "B" "C" "C" "D" "D"
```

```r
(mc <- allComb(f, collapse = "_"))
```

```
##   A B C D A_B A_C A_D
## A 1 0 0 0   1   1   1
## A 1 0 0 0   1   1   1
## B 0 1 0 0   1   0   0
## B 0 1 0 0   1   0   0
## C 0 0 1 0   0   1   0
## C 0 0 1 0   0   1   0
## D 0 0 0 1   0   0   1
## D 0 0 0 1   0   0   1
## attr(,"collapse")
## [1] "_"
## attr(,"comb")
## [1] "all"
```

```r
## checking for complementary entries
checkComb(mc) # TRUE
```

```
## [1] TRUE
## attr(,"comp")
##      i j
## attr(,"same")
##      i j
```

```r
## adding complementary entries to the matrix
mc2 <- cbind(z = 1 - mc[,1], mc[,c(1:ncol(mc), 1)])
colnames(mc2) <- 1:ncol(mc2)
```

```
mc2
```

```
##   1 2 3 4 5 6 7 8 9
## A 0 1 0 0 0 1 1 1 1
## A 0 1 0 0 0 1 1 1 1
## B 1 0 1 0 0 1 0 0 0
## B 1 0 1 0 0 1 0 0 0
## C 1 0 0 1 0 0 1 0 0
## C 1 0 0 1 0 0 1 0 0
## D 1 0 0 0 1 0 0 1 0
## D 1 0 0 0 1 0 0 1 0
```

```
checkComb(mc2) # FALSE
```

```
## [1] FALSE
## attr(,"comp")
##      i j
## [1,] 1 2
## [2,] 1 9
## attr(,"same")
##      i j
## [1,] 9 2
```

2.2 Choosing a parametric model

A suitable parametric (or semi-parametric) model can be chosen to describe the relationship between the observations for a single species and the site descriptors. The choice of the parametric model depends on the nature of the observations and the goals of the study. The systematic component of the model (also called the linear predictor), $f(\mu_i) = \beta_0^{(m)} + \beta_1^{(m)} z_i^{(m)} + \sum_{j=1}^{p} \alpha_j^{(m)} x_{ij}$ x_ij, is linked to the random component of the model through the link function f. The expected value is given by the inverse link function: $E[Y_i] = \mu_i = f^{-1}(\beta_0^{(m)} + \beta_1^{(m)} z_i^{(m)} + \sum_{j=1}^{p} \alpha_j^{(m)} x_{ij})$. Expected values can then be estimated for each partition. The symbol x_{ij} denotes other site descriptors ($j = 1, ..., p$; number of predictors besides g_i) that can take discrete or continuous values. These variables might describe variation in the observations not fully explained by the partitions, e.g. due to spatially uneven distribution, differences in sampling effort, or environmental variables interfering with the observation process.

We can estimate the parameters in the linear predictor (and possibly other “nuisance”" factors such as variance components in mixed effects models) and calculate expected values. The probability density function for the model $P(Y_i = y_i \mid z_i^{(m)}, x_{ij}, \theta)$ is used to find the maximum likelihood estimates (MLE) of the model parameters $\hat{\theta}^{(m)} = (\hat{\beta}_0^{(m)}, \hat{\beta}_1^{(m)}, \hat{\alpha}_1^{(m)}, \ldots, \hat{\alpha}_p^{(m)})$ that jointly maximize the log-likelihood function. The log-likelihood function evaluated at the MLE is $l(\hat{\theta}^{(m)}; y)$.

The opticut package has several built-in distributions that can be specified though the `dist` argument. Currently available distributions:

- `"gaussian"`: real valued continuous observations, e.g. biomass;
- `"poisson"`: Poisson count data;
- `"binomial"`: presence-absence type data;

- `"negbin"`: overdispersed Negative Binomial count data;
- `"beta"`: continuous response in the unit interval, e.g. percent cover;
- `"zip"`, `"zip2"`: zero-inflated Poisson counts (partitioning in count model: `"zip"`, or in zero model: `"zip2"`);
- `"zinb"`, `"zinb2"`: zero-inflated Negative Binomial counts (partitioning in count model: `"zinb"`, or in zero model: `"zinb2"`);
- `"ordered"`: response measured on ordinal scale, e.g. ordinal vegetation cover (only available for single species because ordinal levels often do not match across different species thus leading to different intercept terms);
- `"rsf"`, `"rspf"`: presence-only data using resource selection and resource selection probability functions (only available for single species because used distribution is unique for all species thus multiple species cannot be combined in a single input matrix).

Other distributions can be specified by user-defined functions as explained later in the manual.

2.3 All combinations

Fitting the model to all the M candidate binary partitions leads to a set of log-likelihood values. One can compare the log-likelihood values $l(\hat{\theta}^{(m)}; y))$ to the log-likelihood value based on the null model $l(\hat{\theta}^{(0)}; y)$. We define the null model the same way as the other M models but without the binary partition: $\beta_1^{(m)} = 0$. The log of the likelihood ratio between the M candidate models and the null model can be calculated as $l(\hat{\theta}^{(m)}; y) - l(\hat{\theta}^{(0)}; y))$.

The best-supported model m' and the corresponding binary partition $z^{(m')}$ is the model with the highest log-likelihood ratio value $l(\hat{\theta}^{(m')}; y)$. Model weights are calculated as $w_m = exp\{l(\hat{\theta}^{(m)}; y) - l(\hat{\theta}^{(m')}; y)\} / \sum_{m=1}^{M} exp\{l(\hat{\theta}^{(m)}; y) - l(\hat{\theta}^{(m')}; y)\}$. These weights sum to 1 and indicate asymptotic probabilities of finding the same best partition when the sampling is replicated. The concentration of asymptotic probabilities among the models can be expressed through the Simpson index $H = \sum_{m=1}^{M} w_m^2$. High values of H indicate high concentration of the model weights for one or few model out of the total number of models compared (Kemencei et al. 2014).

The opticut package provides the `opticut1` function to fit a chosen parametric model to a set of binary partitions and the summary returns the model output for each candidate partition with log likelihood ratios and model weights.

```
## stratification
g <- c(1,1,1,1, 2,2,2,2, 3,3,3,3)
## abundance
y <- c(0,0,3,0, 2,3,0,5, 5,6,3,4)
mods <- opticut1(Y = y, Z = allComb(g), dist = "gaussian")
mods
```

```
## Univariate opticut results, comb = all, dist = gaussian
## I = 0.8932; w = 0.5742; H = 0.4926; logL_null = -25.93
##
## Best supported models with logLR >= 2:
##   assoc      I   mu0  mu1 logLR      w
## 3    ++ 0.8932 1.625 4.50 3.233 0.5742
## 1    -- 0.8798 3.500 0.75 2.879 0.4031
## 3 binary splits (1 model not shown)
```

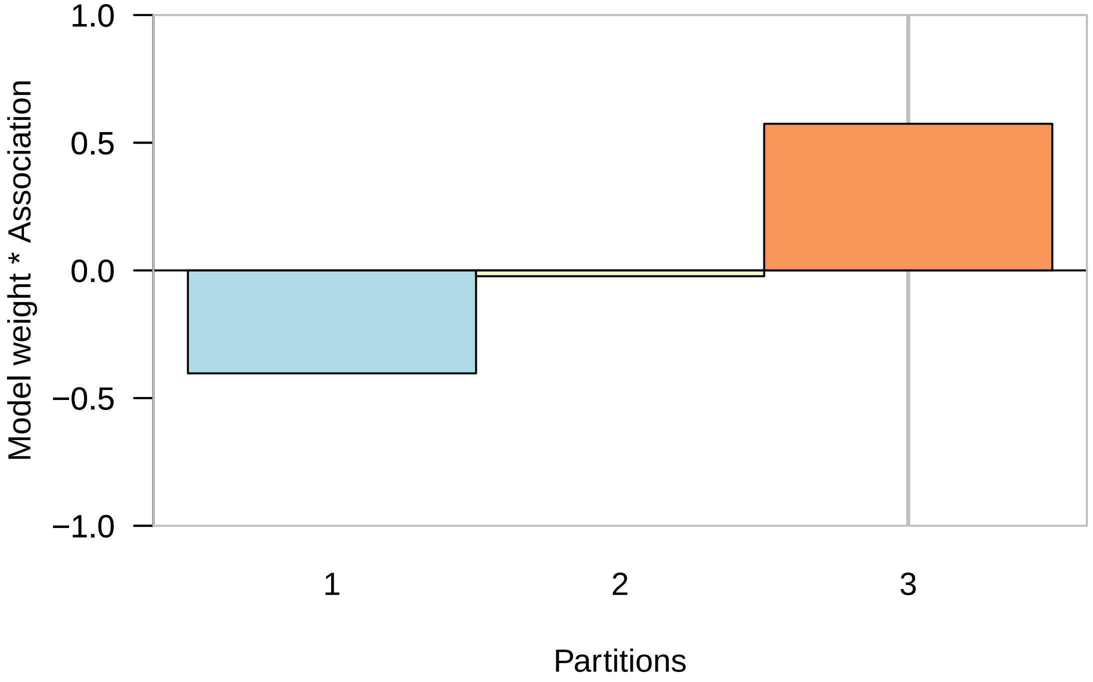
Not all the models are printed, only the ones where the log likelihood ratio (`logLR`) value is ≥ 2 by default. We can change this by explicitly defining the `cut` argument in the `print` method (or by setting the `cut` global option via `ocoptions`, as explained later):

```
print(mods, cut = -Inf)
```

```
## Univariate opticut results, comb = all, dist = gaussian
## I = 0.8932; w = 0.5742; H = 0.4926; logL_null = -25.93
##
## Best supported models with logLR >= -Inf:
##   assoc       I   mu0  mu1    logLR       w
## 3    ++ 0.89319 1.625 4.50 3.232629 0.57415
## 1    -- 0.87983 3.500 0.75 2.878892 0.40309
## 2     - 0.06242 2.625 2.50 0.004726 0.02276
## 3 binary splits
```

Model support across the partitions can be visualized by the model weight plot (`wplot`):

```
wplot(mods, cut = -Inf)
```



The `opticut` function can take matrices or a model formula as its input, and repeats the procedure done by opticut1 for multiple species in a community matrix. The summary shows the best-supported model for each species. It is the preferred way of specifying the model for single species as well. Single species results are part of the `$species` element of the output object:

```
(oc <- opticut(y, strata = g, comb = "all", dist = "gaussian"))
```

```
## Multivariate opticut results, comb = all, dist = gaussian
##
## Call:
```

```
## opticut.default(Y = y, strata = g, dist = "gaussian", comb = "all")
##
## 1 species, 3 binary splits
```

```
summary(oc)
```

```
## Multivariate opticut results, comb = all, dist = gaussian
##
## Call:
## opticut.default(Y = y, strata = g, dist = "gaussian", comb = "all")
##
## Best supported model with logLR >= 2:
##      split assoc      I   mu0 mu1 logLR      w
## Sp 1     3    ++ 0.8932 1.625 4.5 3.233 0.5742
## 3 binary splits
```

```
oc$species
```

```
## $`Sp 1`
## Univariate opticut results, comb = all, dist = gaussian
## I = 0.8932; w = 0.5742; H = 0.4926; logL_null = -25.93
##
## Best supported models with logLR >= 2:
##   assoc      I   mu0  mu1 logLR      w
## 3    ++ 0.8932 1.625 4.50 3.233 0.5742
## 1    -- 0.8798 3.500 0.75 2.879 0.4031
## 3 binary splits (1 model not shown)
```

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, as shown above.

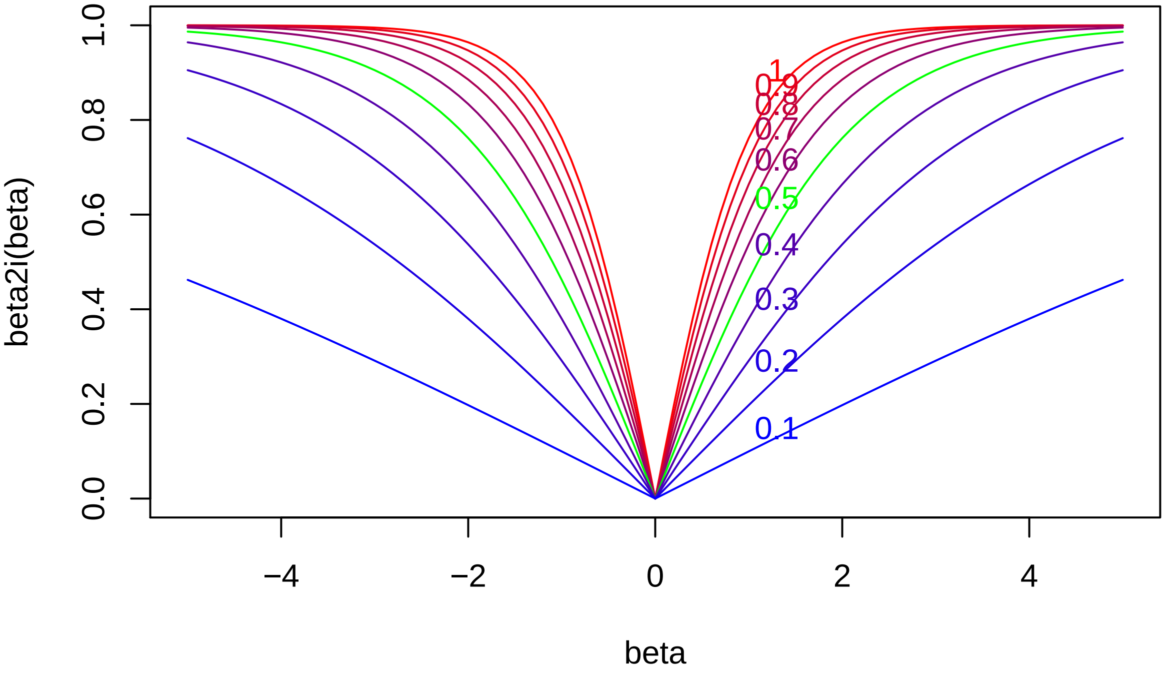
2.4 Indicator value

Once a model is fit a to a given binary partition, we can quantify the indicator value for the species. The indicator value denoted by $I^{(m)}$ describes the contrast between the two subset of the data represented by the binary partition $z^{(m)}$. We define indicator value as a scaled version of the $\beta_1^{(m)}$ coefficient estimate: $I^{(m)} =\mid tanh(c\beta_1^{(m)}) \mid$, where $c = 0.5$. The hyperbolic tangent function is used as an inverse Fisher transformation to scale real valued coefficients into the [-1,1] range. The absolute value then results in a [0, 1] range for the indicator value.

The c is a scaling constant that modifies the shape of the function. We chose 0.5 as the default value that allows the indicator value to change more gradually according to our experience with real-world data sets. The $c = 0.5$ setting is also identical to an inverse logistic function transformed into the [-1, 1] range.

The `beta2i` function is used internally to calculate the indicator value. Here we show the effect of the scaling constant c on the shape of the function, the default $c = 0.5$ is in green:

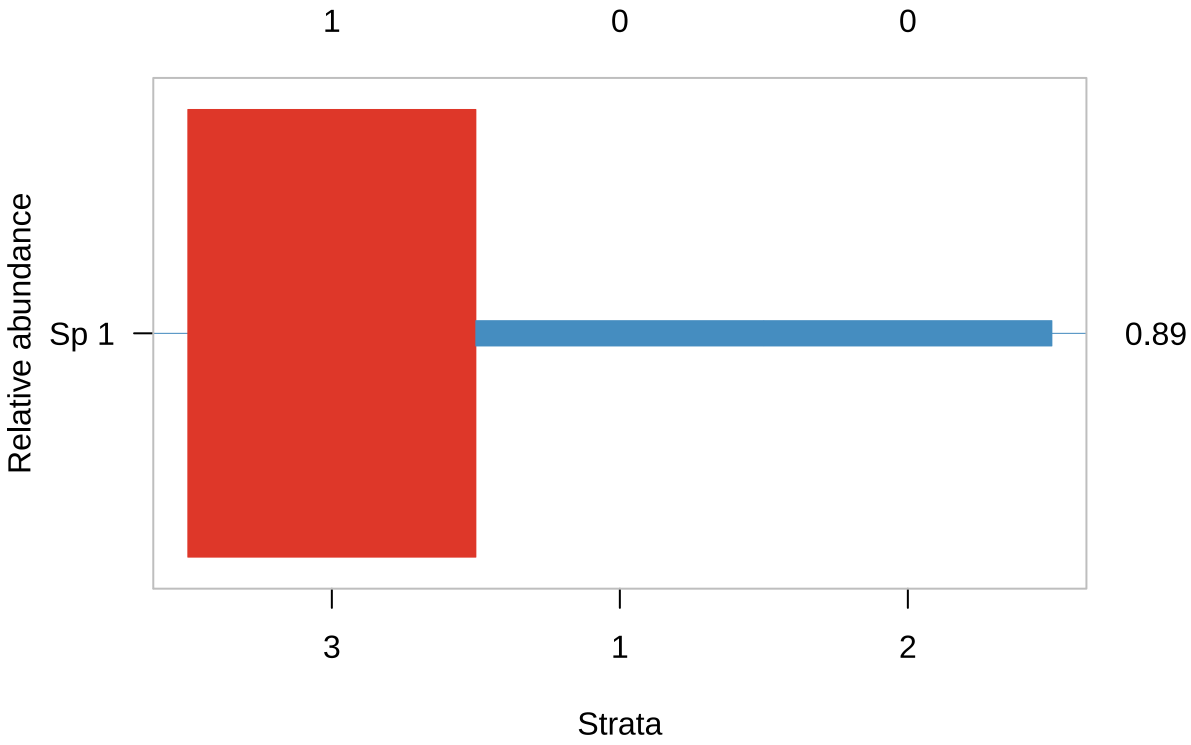
```
beta <- seq(-5, 5, 0.1)
Col <- occolors(c("red", "blue"))(10)
Col[6] <- "#00FF00"
plot(beta, beta2i(beta), type = "n")
s <- seq(1, 0.1, -0.1)
for (i in 1:10) {
    lines(beta, beta2i(beta, scale = s[i]), col = Col[i])
    text(1.5 - 0.2, beta2i(1.5, scale = s[i]), s[i], col = Col[i])
}
```



An alternative way to define the indicator value would be taking the relative difference between the expected values for the 0 and 1 stratum in $z^{(m)}$. This, however, depends on the response scale and the baseline values chosen for possible modifying effects.

Our definition of indicator value is on the linear predictor scale and is more readily compared across species without respect to their relative abundance and values of other modifying factors. Note however, that the meaning of the indicator value might be quite different for studies using different parametric models: it is a difference in the Gaussian case, multiplier in log-linear models, change in log-odds in logistic or ordinal regression. The indicator value given a binary partition is returned by the model summaries, and used in visualization:

```
plot(oc)
```



Use the `ocoptions` function to change the default `scale` (c) setting.

2.5 User defined combinations

It is possible to

```
comb <- cbind(
    A = c(rep(1, 4), rep(0, 8)),
    B = c(rep(0, 4), rep(1, 4), rep(0, 4)))
comb
```

```
##       A B
##  [1,] 1 0
##  [2,] 1 0
##  [3,] 1 0
##  [4,] 1 0
##  [5,] 0 1
##  [6,] 0 1
##  [7,] 0 1
##  [8,] 0 1
##  [9,] 0 0
## [10,] 0 0
## [11,] 0 0
## [12,] 0 0
```

```
print(opticut1(Y = y, Z = comb, dist = "gaussian"), cut = -Inf)
```

```
## Univariate opticut results, comb = NA, dist = gaussian
## I = 0.8798; w = 0.9466; H = 0.8988; logL_null = -25.93
##
## Best supported models with logLR >= -Inf:
##   assoc       I   mu0  mu1    logLR       w
## A    -- 0.87983 3.500 0.75 2.878892 0.94655
## B     - 0.06242 2.625 2.50 0.004726 0.05345
## 2 binary splits
```

If the user happen to define complementary partitions, an error message is thrown:

```
comb <- cbind(comb, 1-comb)
colnames(comb) <- LETTERS[1:4]
comb
```

```
##       A B C D
##  [1,] 1 0 0 1
##  [2,] 1 0 0 1
##  [3,] 1 0 0 1
##  [4,] 1 0 0 1
##  [5,] 0 1 1 0
##  [6,] 0 1 1 0
##  [7,] 0 1 1 0
##  [8,] 0 1 1 0
##  [9,] 0 0 1 1
## [10,] 0 0 1 1
## [11,] 0 0 1 1
## [12,] 0 0 1 1
```

```
try(opticut1(Y = y, Z = comb, dist = "gaussian"))
checkComb(comb)
```

```
## [1] FALSE
## attr(,"comp")
##      i j
## [1,] 1 3
## [2,] 2 4
## attr(,"same")
##      i j
```

The global option `check_comb` can be set to override this default behavior, although there is no real point in duplicating reparametrized but otherwise identical models:

```
op <- ocoptions(check_comb = FALSE, cut = -Inf)
opticut1(Y = y, Z = comb, dist = "gaussian")
```

```
## Univariate opticut results, comb = NA, dist = gaussian
## I = 0.8798; w = 0.4733; H = 0.4494; logL_null = -25.93
##
## Best supported models with logLR >= -Inf:
##   assoc       I   mu0   mu1    logLR       w
## A    -- 0.87983 3.500 0.750 2.878892 0.47328
## C    ++ 0.87983 0.750 3.500 2.878892 0.47328
## B     - 0.06242 2.625 2.500 0.004726 0.02672
## D     + 0.06242 2.500 2.625 0.004726 0.02672
## 4 binary splits
```

```
ocoptions(op)
```

2.6 Rank based combinations

The IndVal method requires the algorithm to evaluate $2^K - 1$ binary partitions. Our opticut approach is parametrization invariant with respect to coding the levels in the binary partitions (it affects the intercept term but not the contrast or the log likelihood ratio). This effectively halves the number of partitions we need to compare ($2^{K-1} - 1$, `comb = "all"` in opticut). Still, the number of partitions increases according to powers of 2. Here we propose an approach that increases linearly with K. This algorithm is based on sorting all the K partitions in g according to increasing order of the linear predictor estimates for K coefficients (as opposed to estimating 2 coefficients for a binary partition). The logic follows from the fact that the optimal binary partitioning tries to find the best split in terms of likelihood ratio with lower estimates on one side, and higher estimates on the other side of the split. As a consequence, we only need to try $K - 1$ binary partitions to find the optimal $z^{(m')}$. This algorithm is implemented in the `rankComb` function that is called by opticut with the argument `comb = "rank"`.

The function `rankComb` evaluates a model with a K-level factor and returns a corresponding partitioning matrix. Attributes hold the estimates for the K levels:

```
rankComb(Y = y, Z = as.factor(g), dist = "gaussian", collapse = "_")
```

```
##   3 2_3
## 1 0   0
## 1 0   0
## 1 0   0
## 1 0   0
## 2 0   1
## 2 0   1
## 2 0   1
## 2 0   1
## 3 1   1
## 3 1   1
## 3 1   1
## 3 1   1
## attr(,"est")
##    1    2    3
## 0.75 2.50 4.50
## attr(,"collapse")
## [1] "_"
## attr(,"comb")
## [1] "rank"
```

The `collapse` argument can be important to make partitions more distinguishable. The global option is `" "` that can be modified via the `ocoptions` function. It can cause problems when the the `collapse` character is part of the factor levels used for g. The `fix_levels` function comes handy for fixing these levels. The function replaces the `collapse` character to something else:

```r
getOption("ocoptions")$collapse
```

```
## [1] "+"
```

```r
fix_levels(as.factor(c("A b", "C d")), sep=":")
```

```
## [1] A b C d
## Levels: A b C d
```

```r
fix_levels(as.factor(c("A b", "C d")), sep="")
```

```
## [1] A b C d
## Levels: A b C d
```

There is an overhead of fitting the model to calculate the ranking first. But computing efficiencies can be still high compared to all partitions. As a consequence of the ranking process, we do not have summaries for all the possible binary partitions, only for the top candidates. Moreover, the partitions produced for each species might not be identical. Therefore the model weights (w) and Simpson index (H) have different interpretation and cannot be that easily compared across species, unless the model weights are highly concentrated for the top models. In this case, the sum of weights for the missing models becomes negligible. Another consequence of the ranking process is that $\beta_1^{(m)}$ estimates are always positive. In the case of comparing all the partitions, the full set of partitions is fixed for all species, and some respond positively while others respond negatively to the same binary variable in terms of the $\beta_1^{(m)}$ values. Thus it is required to store the sign of the relationship as part of the summary.

The `comb = "rank"` is the default setting in `opticut`. It is clear that the 2 approaches lead to identical best partitions. Only the model weights (w) are different:

```r
summary(opticut(y, strata = g, comb = "all", dist = "gaussian"))
```

```
## Multivariate opticut results, comb = all, dist = gaussian
##
## Call:
## opticut.default(Y = y, strata = g, dist = "gaussian", comb = "all")
##
## Best supported model with logLR >= 2:
##      split assoc      I   mu0 mu1 logLR      w
## Sp 1     3    ++ 0.8932 1.625 4.5 3.233 0.5742
## 3 binary splits
```

```r
summary(opticut(y, strata = g, comb = "rank", dist = "gaussian"))
```

```
## Multivariate opticut results, comb = rank, dist = gaussian
##
## Call:
## opticut.default(Y = y, strata = g, dist = "gaussian", comb = "rank")
##
```

```
## Best supported model with logLR >= 2:
##      split assoc      I   mu0 mu1 logLR      w
## Sp 1     3    ++ 0.8932 1.625 4.5 3.233 0.5875
## 2 binary splits
```

Here is how the ranking info is turned into binary partitions internally:

```
## 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"))
```

```
##   2 1+2 1+2+4
## 1 0   1     1
## 3 0   0     0
## 3 0   0     0
## 3 0   0     0
## 4 0   0     1
## 3 0   0     0
```

```
attr(rc, "est") # expected values in factor levels
```

```
##        1        2        3        4
## 2.644132 2.650397 1.738868 1.738892
```

```
aggregate(exp(0.5 + 0.5*x1), list(x0=x0), mean) # true values
```

```
##   x0        x
## 1  1 2.718282
## 2  2 2.718282
## 3  3 1.648721
## 4  4 1.648721
```

```
## simple example
oComb(1:4, "+")
```

```
##   1 1+2 1+2+3
## 1 1   1     1
## 2 0   1     1
## 3 0   0     1
## 4 0   0     0
```

```
## using estimates
oComb(attr(rc, "est"))
```

```
##   3 3+4 1+3+4
## 1 0   0     1
## 2 0   0     0
## 3 1   1     1
## 4 0   1     1
```

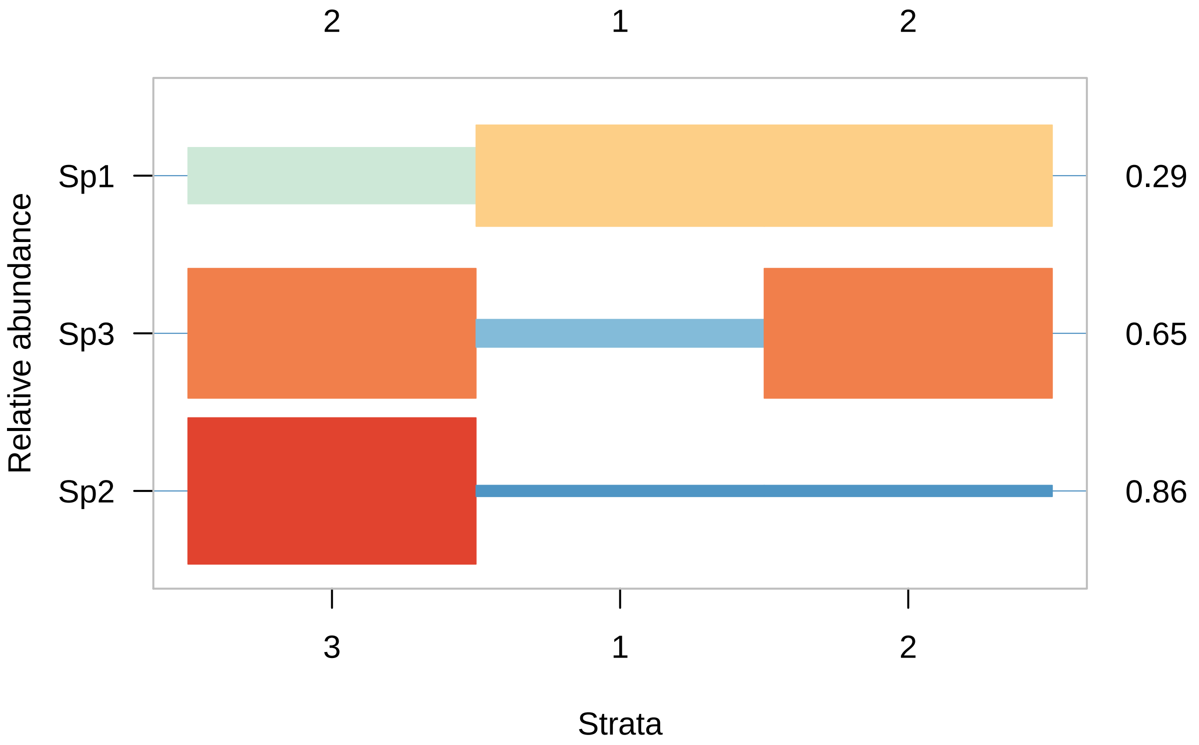
2.7 Partitioning for multiple species

The `opticut` function can take matrices or a model formula as its input, and repeats the procedure done by `opticut1` for multiple species in a community matrix. The summary shows the best-supported model for each species, in this case based on a Poisson count model (`dist = "poisson"`). The `plot` method uses the indicator value (`I` in the summary) to represent the contrast between the two strata of the best supported binary partition:

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

## community matrix
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))

oc <- opticut(formula = y ~ 1, strata = g, dist = "poisson", comb = "all")
summary(oc)
```

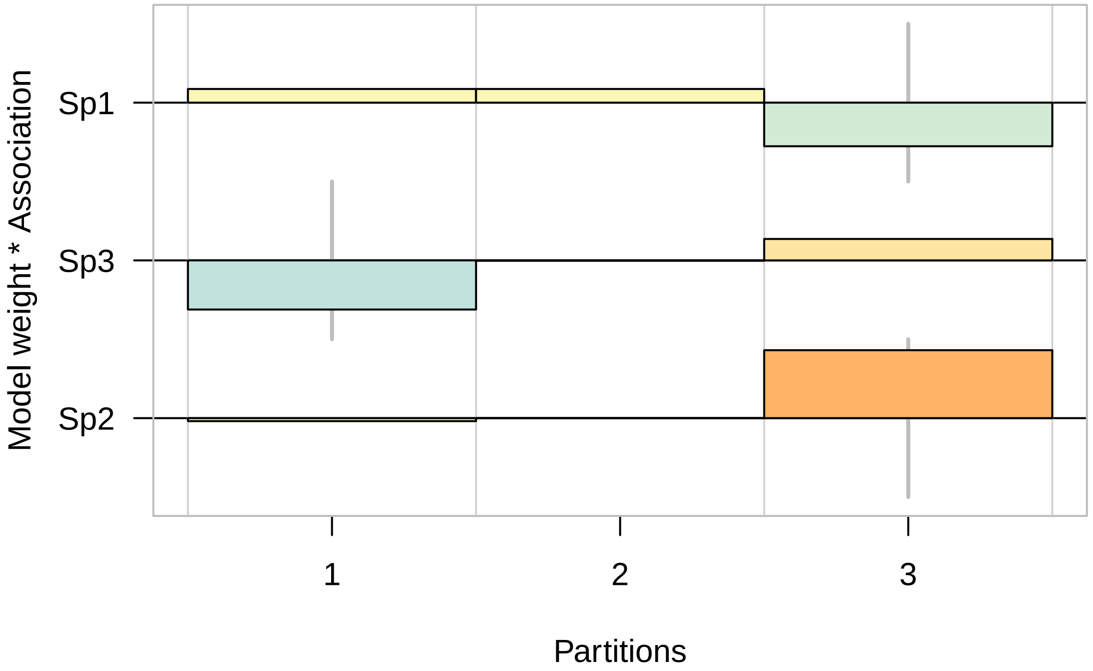
```
## Multivariate opticut results, comb = all, dist = poisson
##
## Call:
## opticut.formula(formula = y ~ 1, strata = g, dist = "poisson",
##     comb = "all")
##
## Best supported models with logLR >= 2:
##     split assoc      I  mu0  mu1 logLR      w
## Sp3     1    -- 0.6471 3.50 0.75 4.793 0.6922
## Sp2     3   +++ 0.8571 0.25 3.25 9.203 0.9573
## 3 binary splits
## 1 species not shown
```

```
plot(oc, cut = -Inf)
```



Model support across the partitions can be visualized by the model weight plot (`wplot`).

```r
wplot(oc, cut = -Inf)
```



Note that the `wplot` uses 'splits' (binary partitions) whereas `plot` uses the K levels. Therefore, `wplot` does not work for multiple species when `comb = "rank"`. In this case the splits can be different across the species, whereas `comb = "all"` uses the same pre-defined partitions across all species.

Compare the "all" and "rank" based combinations: same best partitions putting aside complementaryity

```
op <- ocoptions(cut = -Inf)
summary(opticut(formula = y ~ 1, strata = g, dist = "poisson", comb = "all"))
```

```
## Multivariate opticut results, comb = all, dist = poisson
##
## Call:
## opticut.formula(formula = y ~ 1, strata = g, dist = "poisson",
##     comb = "all")
##
## Best supported models with logLR >= -Inf:
##     split assoc      I  mu0  mu1 logLR      w
## Sp1     3     - 0.2857 4.50 2.50 1.498 0.6144
## Sp3     1    -- 0.6471 3.50 0.75 4.793 0.6922
## Sp2     3   +++ 0.8571 0.25 3.25 9.203 0.9573
## 3 binary splits
```

```
summary(opticut(formula = y ~ 1, strata = g, dist = "poisson", comb = "rank"))
```

```
## Multivariate opticut results, comb = rank, dist = poisson
##
## Call:
## opticut.formula(formula = y ~ 1, strata = g, dist = "poisson",
##     comb = "rank")
##
## Best supported models with logLR >= -Inf:
##     split assoc      I  mu0  mu1 logLR      w
## Sp1   1+2     + 0.2857 2.50 4.50 1.498 0.7611
## Sp3   2+3    ++ 0.6471 0.75 3.50 4.793 0.6962
## Sp2     3   +++ 0.8571 0.25 3.25 9.203 0.9577
## 2 binary splits
```

```
ocoptions(op)
```

2.8 Accessing models and partitions

The `bestmodel` method returns the best-supported model for further model diagnostics and prediction, the `getMLE` prints out the estimated coefficients and the variance-covariance matrix:

```
mods <- bestmodel(oc)
mods
```

```
## $Sp1
##
## Call:  stats::glm(formula = Y ~ . - 1, family = Family, data = XX)
##
## Coefficients:
##      V1       Z1
##  1.5041  -0.5878
##
## Degrees of Freedom: 12 Total (i.e. Null);  10 Residual
## Null Deviance:       63.01
## Residual Deviance: 4.39  AIC: 46.05
##
## $Sp2
##
## Call:  stats::glm(formula = Y ~ . - 1, family = Family, data = XX)
##
## Coefficients:
##     V1      Z1
## -1.386   2.565
##
## Degrees of Freedom: 12 Total (i.e. Null);  10 Residual
## Null Deviance:       25.54
## Residual Deviance: 6.445     AIC: 26.58
##
## $Sp3
##
## Call:  stats::glm(formula = Y ~ . - 1, family = Family, data = XX)
##
## Coefficients:
##     V1      Z1
##  1.253  -1.540
##
## Degrees of Freedom: 12 Total (i.e. Null);  10 Residual
## Null Deviance:       49.33
## Residual Deviance: 18.9  AIC: 48.37
```

```
## explore further
str(predict(mods[[1]]))
```

```
##  Named num [1:12] 1.5 1.5 1.5 1.5 1.5 ...
##  - attr(*, "names")= chr [1:12] "1" "2" "3" "4" ...
```

```
confint(mods[[1]])
```

```
## Waiting for profiling to be done...
```

```
##        2.5 %     97.5 %
## V1  1.158610 1.81387284
## Z1 -1.343213 0.07397014
```

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

```
## $coef
##          V1          Z1
##  1.5040774 -0.5877867
##
```

```
## $vcov
##             V1          Z1
## V1  0.02777777 -0.02777777
## Z1 -0.02777777  0.12777667
##
## $dist
## [1] "poisson"
```

The `bestpart` method returns the binary partition for the best-supported model:

```
bestpart(oc)
```

```
##   Sp1 Sp2 Sp3
## 1   0   0   1
## 1   0   0   1
## 1   0   0   1
## 1   0   0   1
## 2   0   0   0
## 2   0   0   0
## 2   0   0   0
## 2   0   0   0
## 3   1   1   0
## 3   1   1   0
## 3   1   1   0
## 3   1   1   0
```

2.9 Quantifying uncertainty

Uncertainty in the estimated coefficients and uncertainty in the derived indicator value for the best-supported model ($I^{(m')}$) is quantified based on the estimate of the Hessian matrix assuming asymptotic normality of the MLE. The distribution of $I^{(m')}$ in this case is based on parametric bootstrap. This approach (`type = "asymp"` in `uncertainty`) is suitable when asymptotic normality assumption is reasonable, i.e. sample size is large. For small sample situations, a non-parametric bootstrap algorithm is implemented (`type = "boot"`) to estimate uncertainty in $I^{(m')}$. The summary contains lower and upper confidence limits (for a given error rate) representing the asymptotic or bootstrap distribution (based on B number of iterations) given the fixed partition for the best model, $z^{(m')}$.

The output is summarized, the `$uncertainty` component contains individual species results:

```
g <-    c(1,1,1,1, 2,2,2,2, 3,3,3,3)

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))

oc <- opticut(formula = y ~ 1, strata = g, dist = "poisson")
```

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

```
## Multivariate opticut uncertainty results
## type = asymp, B = 999, level = 0.95
##
##     split R      I   Lower  Upper
## Sp1   1+2 1 0.2819 0.01915 0.5611
## Sp3   2+3 1 0.6120 0.23287 0.8668
## Sp2     3 1 0.8280 0.51930 0.9662
```

```r
uc$uncertainty
```

```
## $Sp1
## Univariate opticut uncertainty results, type = asymp, B = 999
##
##    best             I                   mu0             mu1
##  1+2:1000   Min.   :0.0005819   Min.   :1.043   Min.   :2.684
##             1st Qu.:0.1705304   1st Qu.:2.051   1st Qu.:3.996
##             Median :0.2813414   Median :2.506   Median :4.479
##             Mean   :0.2818717   Mean   :2.630   Mean   :4.546
##             3rd Qu.:0.3888490   3rd Qu.:3.056   3rd Qu.:5.003
##             Max.   :0.7179702   Max.   :7.440   Max.   :7.081
##
## $Sp2
## Univariate opticut uncertainty results, type = asymp, B = 999
##
##  best           I                mu0                mu1
##  3:1000   Min.   :0.1822   Min.   :0.02765   Min.   :1.342
##           1st Qu.:0.7750   1st Qu.:0.15502   1st Qu.:2.714
##           Median :0.8621   Median :0.23968   Median :3.267
##           Mean   :0.8280   Mean   :0.31351   Mean   :3.396
##           3rd Qu.:0.9138   3rd Qu.:0.40962   3rd Qu.:3.939
##           Max.   :0.9824   Max.   :2.06166   Max.   :7.864
##
## $Sp3
## Univariate opticut uncertainty results, type = asymp, B = 999
##
##    best             I                 mu0               mu1
##  2+3:1000   Min.   :0.03062   Min.   :0.1238   Min.   :1.768
##             1st Qu.:0.50640   1st Qu.:0.5115   1st Qu.:3.078
##             Median :0.64176   Median :0.7602   Median :3.496
##             Mean   :0.61203   Mean   :0.8888   Mean   :3.539
##             3rd Qu.:0.74693   3rd Qu.:1.1265   3rd Qu.:3.947
##             Max.   :0.93269   Max.   :4.0788   Max.   :6.218
```

The bootstrap can be difficult for small sample sizes, strata can go completely missing:

```
try(uc <- uncertainty(oc, type = "boot", B = 99))
```

A general requirement for the bootstrap approach (`"boot"` and `"multi"`) is that the bootstrap samples contain observations from each stratum. We recommend having at least 5 observations per strata. Possible problems with missing partitions in small data sets can be remedied by supplying pre-defined indices for resampling, for example, based on jackknife (leave-one-out) approach. The resampling scheme can be customized for such needs. Use the `check_strata` function:

```
B <- sapply(1:length(g), function(i) which((1:length(g)) != i))
check_strata(oc, B) # check representation
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## attr(,"nx")
## [1] 3
## attr(,"nmat")
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3
```

```
summary(uncertainty(oc, type = "boot", B = B))
```

```
## Multivariate opticut uncertainty results
## type = boot, B = 12, level = 0.95
##
##     split R      I  Lower  Upper
## Sp1   1+2 1 0.2866 0.2167 0.3644
## Sp3   2+3 1 0.6523 0.5556 0.9053
## Sp2     3 1 0.8572 0.8384 0.9158
```

The reliability of the best partition can also be assessed using the setting `type = "multi"` (as in *multi*ple models). In this case, the model partitions are re-evaluated for each bootstrap sample. Model uncertainty is assessed as the number of times a partition is supported out of the B bootstrap runs ($b = 1, \ldots, B$). The reliability (R) metric in the summary is the proportion for the most frequently supported partition. The corresponding indicator value and confidence interval is conditional on this most commonly supported partition.

```
summary(ucm <- uncertainty(oc, type = "multi", B = B))
```

```
## Multivariate opticut uncertainty results
## type = multi, B = 12, level = 0.95
##
##     split      R      I  Lower  Upper
## Sp1   1+2 1.0000 0.2866 0.2167 0.3644
## Sp3   2+3 0.6923 0.6872 0.6174 0.9368
## Sp2     3 1.0000 0.8572 0.8384 0.9158
```

The `bestpart` method returns the selection frequencies for the best-supported models (based on `type = "multi"` with `comb = "rank"`):

```
bestpart(ucm)
```

```
##   Sp1 Sp2       Sp3
## 1   1   0 0.0000000
## 2   1   0 0.6923077
## 3   0   1 1.0000000
```

The bootstrap averaged ('bagged') expected values for each $k = 1, \ldots, K$ stratum in g can be accessed using the `bsmooth` method that calculates $E[Y_i \mid g_i = k] = \frac{1}{B} \sum_{b=1}^{B} f^{-1}({}^{(b)}\hat{\beta}_0^{(m')} + {}^{(b)}\hat{\beta}_1^{(m')}{}^{(b)}z_i^{(m')})$:

```
bsmooth(ucm)
```

```
##   Sp1  Sp2      Sp3
## 1 4.5 0.25 0.978022
## 2 4.5 0.25 2.956044
## 3 2.5 3.25 3.824176
```

3 Distributions

For most distributions, we will use the `dolina` data set that is part of the **opticut** package. It is 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.

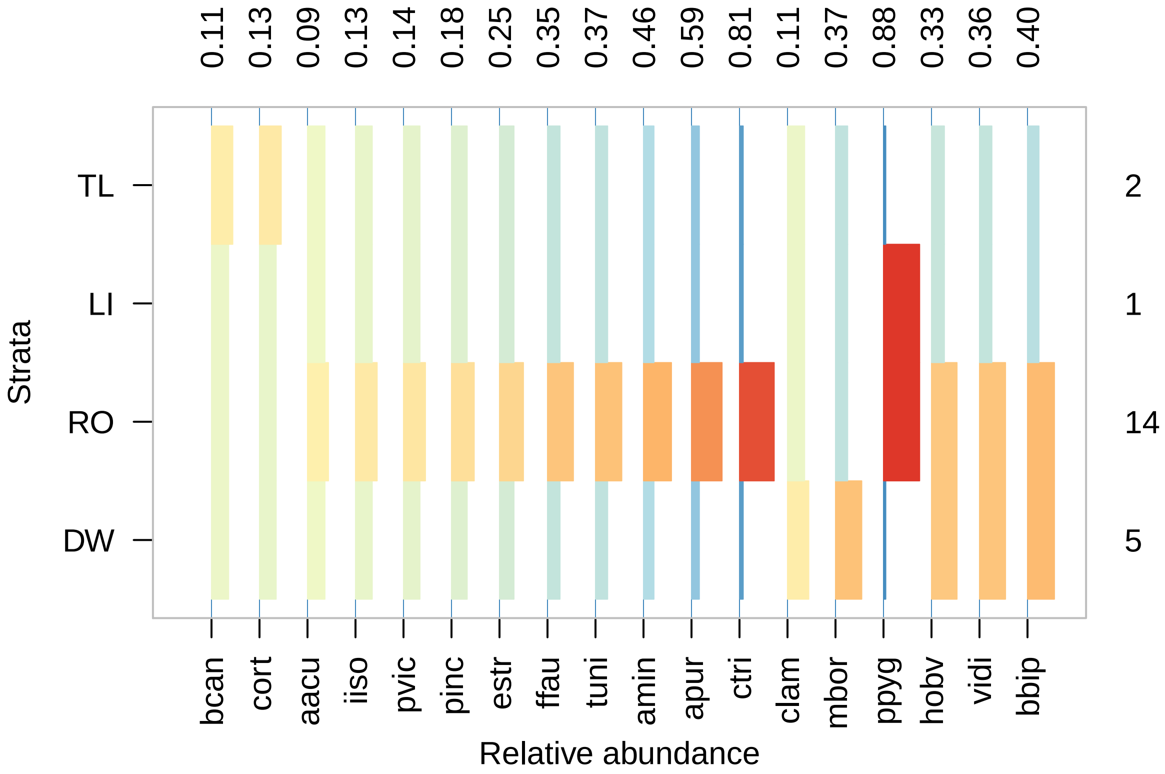
```
data(dolina)
## stratum as ordinal
dolina$samp$stratum <- as.integer(dolina$samp$stratum)
## filter species to speed up things a bit
Y <- dolina$xtab[,colSums(dolina$xtab > 0) >= 20]
```

3.1 Real valued data: Gaussian

```
dol <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="gaussian")
summary(dol)
```

```
## Multivariate opticut results, comb = rank, dist = gaussian
##
## Call:
## opticut.formula(formula = Y ~ stratum + lmoist + method, data = dolina$samp,
##     strata = dolina$samp$mhab, dist = "gaussian")
##
## Best supported models with logLR >= 2:
##      split assoc      I      mu0    mu1  logLR      w
## dper DW+RO   +++ 0.5988  3.03383 4.4165  9.238 0.6600
## bbip DW+RO   +++ 0.4028  0.11875 0.9727 26.906 0.5892
## vidi DW+RO   +++ 0.3552  0.64966 1.3925 10.360 0.9866
## hobv DW+RO   +++ 0.3297 -0.07760 0.6074 23.017 0.6582
## ppyg LI+RO    ++ 0.8822  4.47006 7.2414  4.533 0.8833
## mbor    DW   +++ 0.3662 -0.17269 0.5952 25.555 1.0000
## clam    DW    ++ 0.1078  0.12200 0.3385  5.527 0.8665
## ctri    RO   +++ 0.8063 -0.37944 1.8531  9.223 0.9954
## apur    RO   +++ 0.5912  1.82422 3.1831  9.096 0.9754
## amin    RO   +++ 0.4560  2.17491 3.1595  9.546 0.5919
## tuni    RO   +++ 0.3683  1.01426 1.7872 21.063 1.0000
## ffau    RO   +++ 0.3480  0.02962 0.7560 30.899 1.0000
## estr    RO   +++ 0.2462  0.23962 0.7423 21.892 1.0000
## pinc    RO   +++ 0.1836  0.26944 0.6409 20.890 0.9924
## pvic    RO    ++ 0.1431  0.36807 0.6562  7.107 0.7323
## iiso    RO   +++ 0.1274  0.06840 0.3245 25.005 1.0000
## cort    TL   +++ 0.1321 -0.03773 0.2280 21.583 0.9921
## bcan    TL   +++ 0.1086  0.11177 0.3298 12.425 0.9988
## 3 binary splits
## 1 species not shown
```

```
## vertical plot orientation
plot(dol, horizontal=FALSE, pos=1, upper=0.8)
```



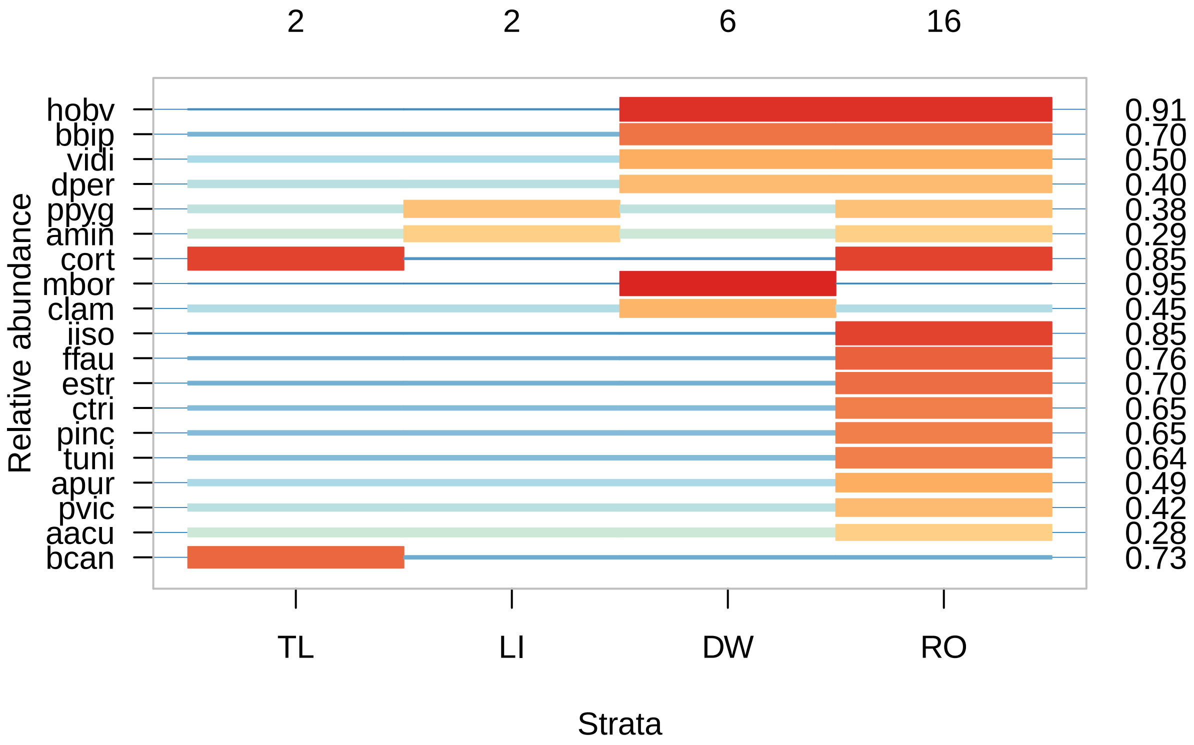
3.2 Nonnegative counts: Poisson and Negative Binomial

```
dol <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="poisson")
summary(dol)
```

```
## Multivariate opticut results, comb = rank, dist = poisson
##
## Call:
```

```
## opticut.formula(formula = Y ~ stratum + lmoist + method, data = dolina$samp,
##     strata = dolina$samp$mhab, dist = "poisson")
##
## Best supported models with logLR >= 2:
##      split assoc      I       mu0      mu1   logLR      w
## hobv DW+RO   +++ 0.9083 0.0384393  0.80014  75.633 1.0000
## bbip DW+RO   +++ 0.6986 0.1843710  1.03897  63.376 1.0000
## vidi DW+RO   +++ 0.4955 0.7994519  2.36967  38.141 1.0000
## dper DW+RO   +++ 0.4029 4.5082800 10.59304  54.686 0.8816
## ppyg LI+RO   +++ 0.3758 3.0963391  6.82425 100.632 1.0000
## amin LI+RO   +++ 0.2884 2.1715347  3.93142  22.048 0.7260
## cort TL+RO   +++ 0.8506 0.0212887  0.26361  22.532 0.4901
## mbor    DW   +++ 0.9489 0.0004436  0.01693  80.850 1.0000
## clam    DW    ++ 0.4461 0.1125752  0.29391   7.122 0.6114
## iiso    RO   +++ 0.8545 0.0504103  0.64254  21.695 1.0000
## ffau    RO   +++ 0.7557 0.1043986  0.75035  49.352 1.0000
## estr    RO   +++ 0.7050 0.1853057  1.07085  29.952 1.0000
## ctri    RO   +++ 0.6495 0.2931486  1.37962 124.868 1.0000
## pinc    RO   +++ 0.6476 0.2513583  1.17509  19.970 0.8020
## tuni    RO   +++ 0.6418 2.3448978 10.74959  42.793 1.0000
## apur    RO   +++ 0.4897 2.6442605  7.72006  53.488 1.0000
## pvic    RO   +++ 0.4150 0.4325312  1.04624   9.007 0.6502
## aacu    RO    ++ 0.2821 0.6989512  1.24819   3.741 0.4915
## bcan    TL   +++ 0.7292 0.1594665  1.01806  15.246 0.9958
## 3 binary splits
```

```
## horizontal plot orientation
plot(dol)
```



Because `opticut` uses the `stats::glm` function to fit the Poisson model, it accepts other arguments, e.g. offsets. Let's subset the `dolina` data set for the litter sampling method (`"Q"`). Pool the abundances in each of the 16 dolines by microhabitat types. By doing this, we make sampling effort uneven. Litter microhabitat was sampled along a North-South transect (7 locations), whereas the other three strata (rock, live trees, dead wood) were sampled at 3 random locations in each dolina.

```
DQ <- dolina$samp[dolina$samp$method == "Q",]
DQ$dol_mhab <- paste0(DQ$dolina, "_", DQ$mhab)
head(DQ)
```

```
##       sample dolina microhab mhab method   aspect stratum lmoist lthick
## 10A1Q   10A1     10   litter   LI      Q southern       4    1.0    2.0
## 10A2Q   10A2     10   litter   LI      Q southern       3    1.0    2.5
## 10A3Q   10A3     10   litter   LI      Q southern       2    1.0    3.0
## 10A4Q   10A4     10   litter   LI      Q     flat       1    1.5    0.5
## 10A5Q   10A5     10   litter   LI      Q northern       2    1.0    1.5
## 10A6Q   10A6     10   litter   LI      Q northern       3    1.0    3.0
##       dol_mhab
## 10A1Q    10_LI
## 10A2Q    10_LI
## 10A3Q    10_LI
## 10A4Q    10_LI
## 10A5Q    10_LI
## 10A6Q    10_LI
```

```
YQ <- dolina$xtab[dolina$samp$method == "Q",]
YQ <- YQ[,colSums(YQ > 0) >= 20]
YQ <- mefa4::groupSums(YQ, 1, DQ$dol_mhab)
```

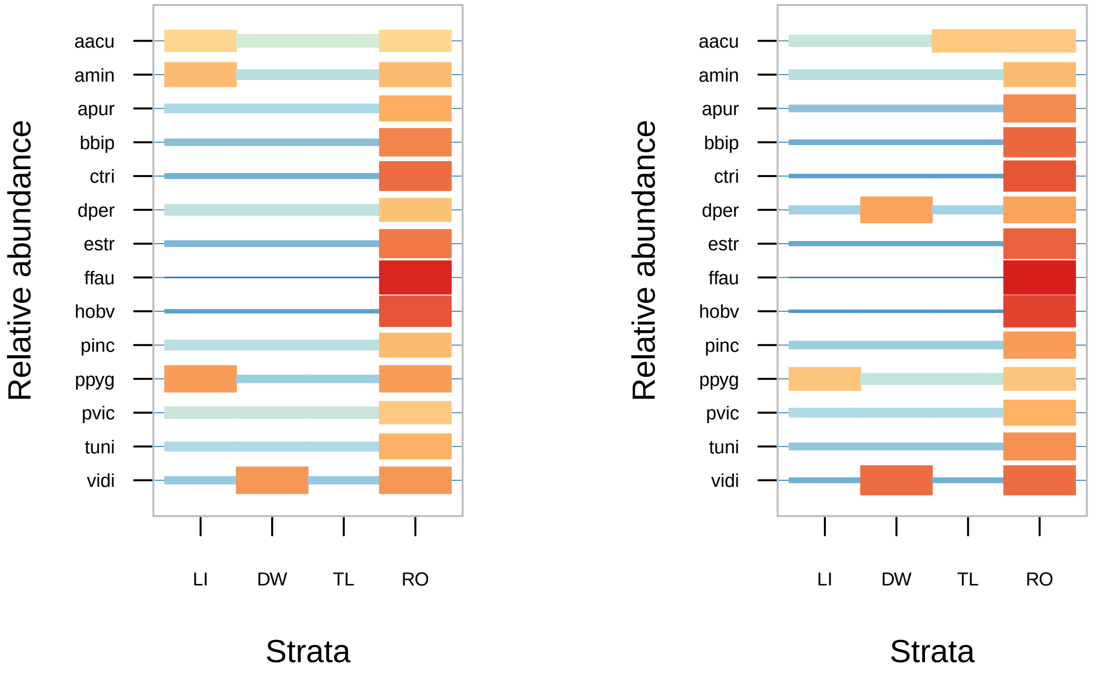
```
DQ <- mefa4::nonDuplicated(DQ, dol_mhab, TRUE)
```

Let's compare the results of ignoring sampling effort differences with a case when we use sampling effort as offset. Offsets are defined as `log(7)` or `log(3)` representing the sampling volume differences (more debris searched leads to more snails found). Using offsets results in less species that is associated with litter:

```
op <- ocoptions(collapse="_", sort=FALSE, cut=-Inf)
dol0 <- opticut(YQ, strata=DQ$mhab, dist="poisson")
off <- ifelse(DQ$mhab == "LI", log(7), log(3))
dol1 <- opticut(YQ, strata=DQ$mhab, dist="poisson", offset=off)
table(wo_offset=summary(dol0)$summary$split,
    with_offset=summary(dol1)$summary$split)
```

```
##           with_offset
## wo_offset DW_RO LI_RO RO TL_RO
##     DW_RO     1     0  0     0
##     LI_RO     0     1  1     1
##     RO        1     0  9     0
```

```
ocoptions(op)
opar <- par(mfrow=c(1,2))
plot(dol0, show_I=FALSE, show_S=FALSE, sort=FALSE, cex.axis=0.6)
plot(dol1, show_I=FALSE, show_S=FALSE, sort=FALSE, cex.axis=0.6)
```



```
par(opar)
```

The Negative Binomial model can be quite picky, as it gives an error somewhere in the middle of the process:

```r
try(dol <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="negbin"))
```

```
## Warning: step size truncated due to divergence
```

Changing the `try_error` global option will allow the process to go on after catching the error:

```r
op <- ocoptions(try_error = TRUE)
dol <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="negbin")
```

```
## Warning: step size truncated due to divergence

## Warning in opticut.default(Y = Y, X = X, strata = strata, dist = dist, comb
## = comb, : Bad news: opticut failed for 1 out of 19 species.
```

```r
dol$failed
```

```
## [1] "mbor"
```

```r
ocoptions(op)
```

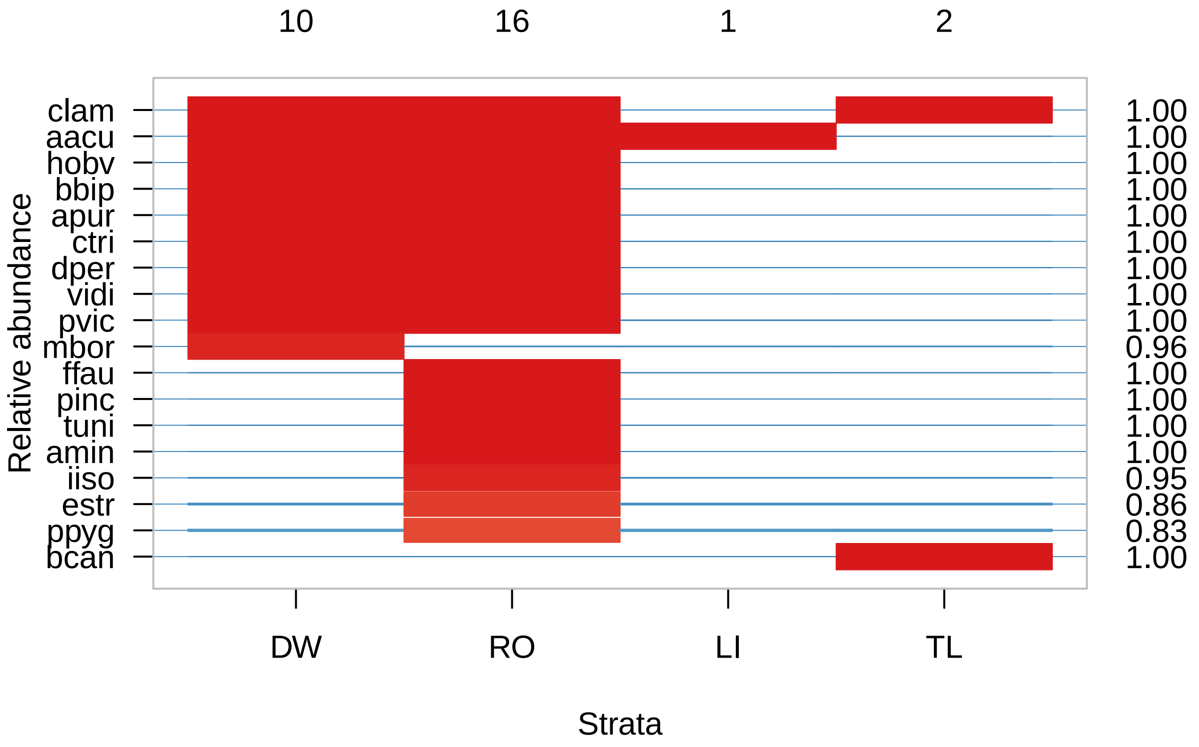
It failed for the `"mbor"` (*Macrogastra borealis*) species, which is now excluded from the output.

3.3 Zero-inflated count distributions

The Zero-inflated Negative Binomial implementation in the **pscl** package seems more robust, no error messages. We use the `dist = "zinb2"` option so that we test for optimal partitioning in the zero-inflation (ZI) component. Using a mixture distribution can be important if 0s can occur not only as a result of the ZI component, but due to low abundance in the count distribution (Poisson or Negative Binomial). Differentiating among these different types of zeros is not possible by binarizing the data and using logistic regression.

In the case of `"zip2"` and `"zinb2"` distributions, the coefficients refer to the probability of non-zero, so that positive and negative effects are properly identified (as opposed to `"zip"` and `"zinb"` where the ZI coefficients refer to probability of zero):

```r
dol <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="zinb2")
plot(dol)
```



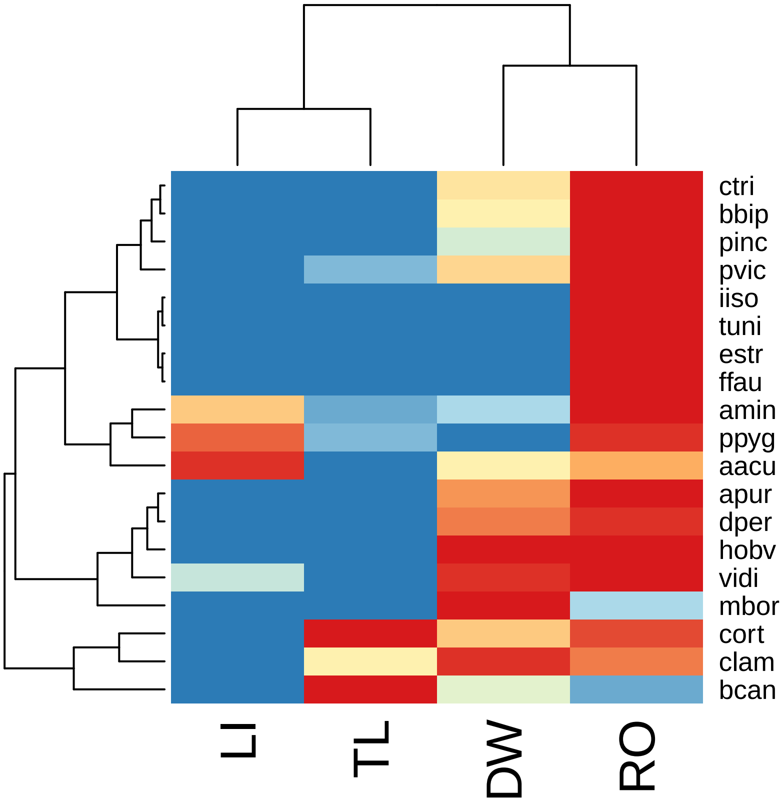
3.4 Presence-absence data: Binomial

The following example uses the dolina data set and illustrates how the link function can be specified in the Binomial case:

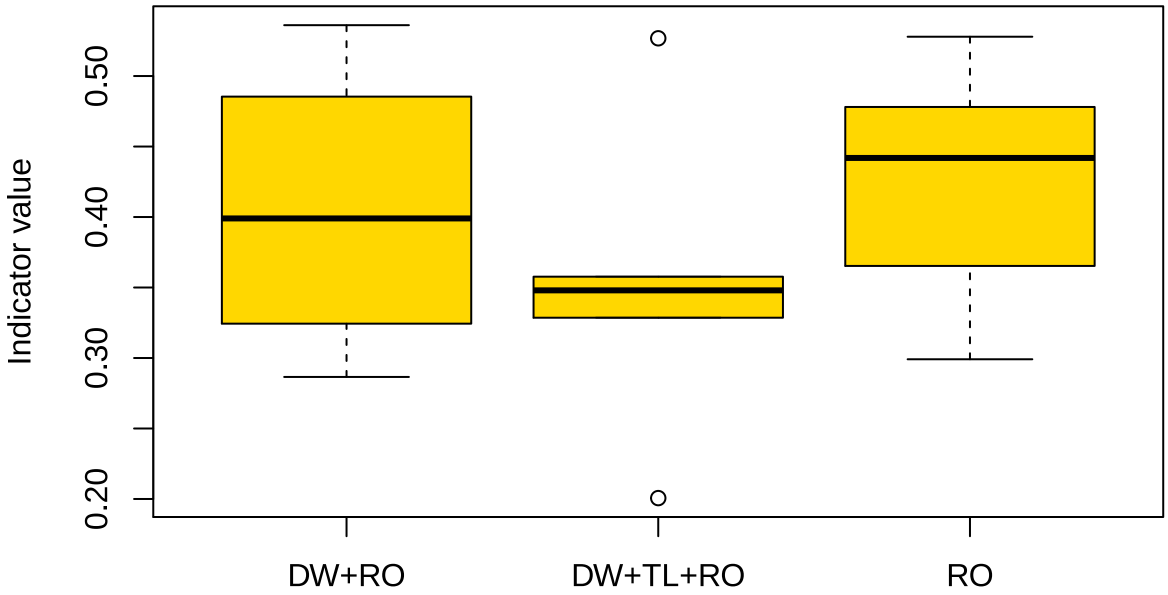
```
## dolina example
data(dolina)
## stratum as ordinal
dolina$samp$stratum <- as.integer(dolina$samp$stratum)
## filter species to speed up things a bit
Y <- ifelse(dolina$xtab[,colSums(dolina$xtab > 0) >= 20] > 0, 1, 0)
## opticut results, note the cloglog link function
dol <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="binomial:cloglog")

## parallel computing for uncertainty
ucdol <- uncertainty(dol, type="multi", B=25)

bestpart(ucdol)
```

```
##          aacu      amin       apur      bbip      bcan      clam      cort
## LI 0.92307692 0.6538462 0.03846154 0.0000000 0.0000000 0.0000000 0.0000000
## DW 0.53846154 0.2692308 0.76923077 0.5384615 0.4230769 0.9230769 0.6538462
## TL 0.03846154 0.1538462 0.03846154 0.0000000 1.0000000 0.5384615 1.0000000
## RO 0.73076923 1.0000000 0.96153846 1.0000000 0.1538462 0.8076923 0.8846154
##          ctri       dper       estr ffau      hobv       iiso       mbor
## LI 0.00000000 0.00000000 0.00000000    0 0.0000000 0.03846154 0.00000000
## DW 0.57692308 0.80769231 0.00000000    0 1.0000000 0.03846154 1.00000000
## TL 0.03846154 0.03846154 0.03846154    0 0.0000000 0.00000000 0.03846154
## RO 1.00000000 0.92307692 1.00000000    1 0.9615385 1.00000000 0.26923077
##         pinc      ppyg      pvic       tuni      vidi
## LI 0.0000000 0.8461538 0.0000000 0.03846154 0.3461538
## DW 0.3846154 0.0000000 0.6153846 0.00000000 0.9230769
## TL 0.0000000 0.1923077 0.1923077 0.00000000 0.0000000
## RO 1.0000000 0.9230769 1.0000000 1.00000000 1.0000000
```

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



```
## See how indicator value changes with different partitions
with(ucdol$uncertainty[["pvic"]],
    boxplot(I ~ best, col="gold", ylab="Indicator value"))
```



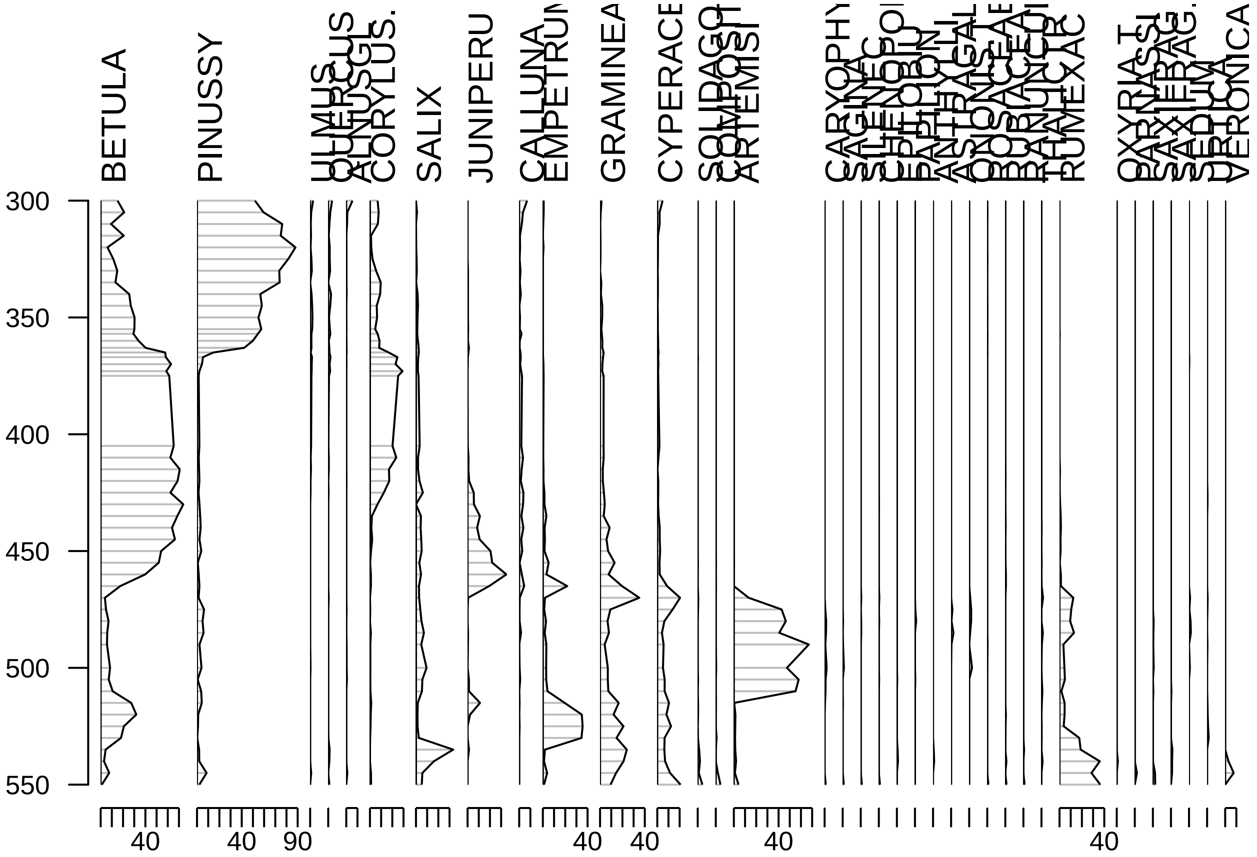
3.5 Percent cover data: Binomial, Beta distribution, and ordinal

The data we are using is from the **rioja** package (Juggins 2015) is pollen stratigraphic data from Abernethy Forest, Scotland, spanning approximately 5500–12100 BP (from Birks & Mathews 1978).

```
library(rioja)
```

```
## This is rioja 0.9-9
```

```
data(aber)
strat.plot(aber$spec, aber$ages$Depth, scale.percent=TRUE, y.rev=TRUE)
```



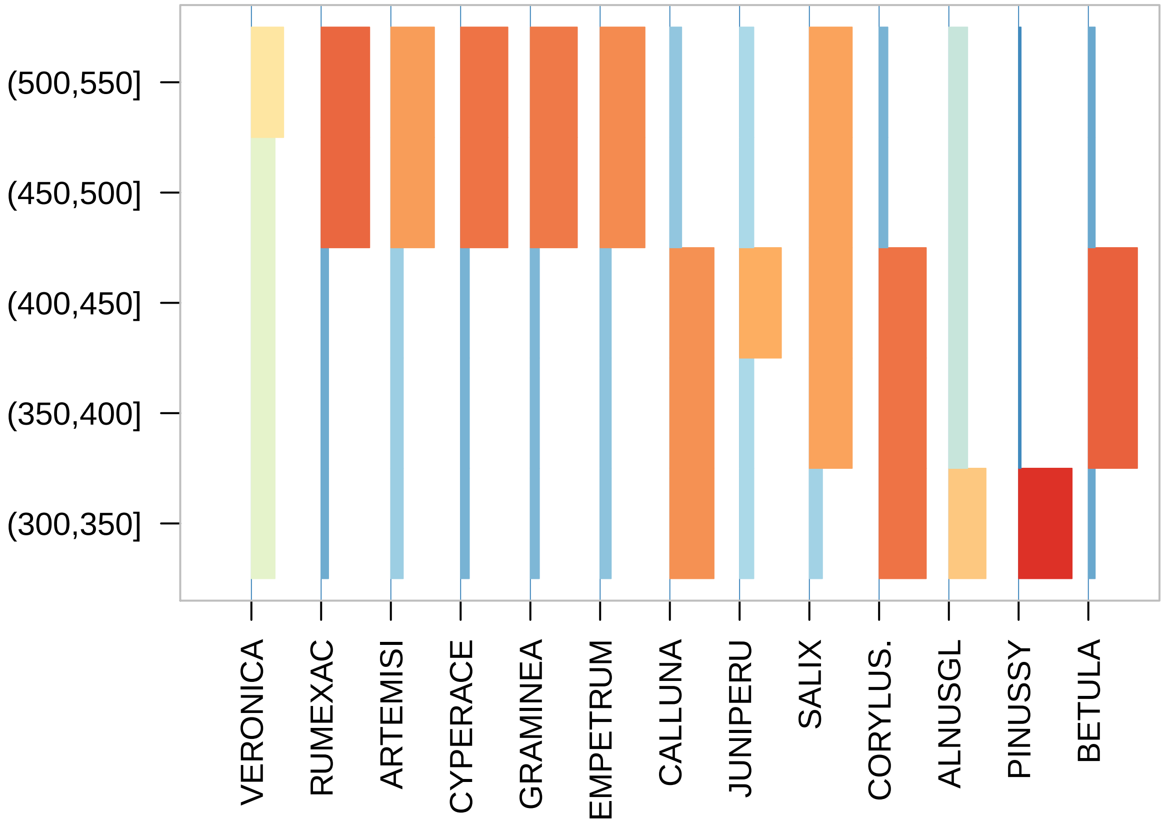
We use the Beta distribution for the stratigraphy example. The only limitation of the Beta distribution in this context is that boundary values (0 and 1) are not part of the support. One can use a tiny value (e.g. 0.0001). We are going to focus on species with at least 5 percent maximum values in the data set.

```
z <- as.factor(cut(aber$ages$Depth, 5))
ab <- as.matrix(aber$spec) / 100
ab[ab == 0] <- 0.0001
ab <- ab[,apply(ab, 2, max) > 0.05]

a <- opticut(ab, strata=z, comb="rank", dist="beta")
summary(a)
```

```
## Multivariate opticut results, comb = rank, dist = beta
##
## Call:
## opticut.default(Y = ab, strata = z, dist = "beta", comb = "rank")
##
## Best supported models with logLR >= 2:
##                                           split assoc      I      mu0
## SALIX    (350,400]+(400,450]+(450,500]+(500,550]   +++ 0.5335 0.014015
## CORYLUS.           (300,350]+(350,400]+(400,450]   +++ 0.6883 0.019366
## CALLUNA            (300,350]+(350,400]+(400,450]   +++ 0.5802 0.005234
## BETULA                       (350,400]+(400,450]   +++ 0.7576 0.153662
## RUMEXAC                      (450,500]+(500,550]   +++ 0.7376 0.015648
## CYPERACE                     (450,500]+(500,550]   +++ 0.6909 0.015826
## GRAMINEA                     (450,500]+(500,550]   +++ 0.6798 0.028003
## EMPETRUM                     (450,500]+(500,550]   +++ 0.6048 0.020863
## ARTEMISI                     (450,500]+(500,550]    ++ 0.5586 0.046111
## PINUSSY                                (300,350]   +++ 0.9142 0.083196
## ALNUSGL                                (300,350]    ++ 0.3289 0.002089
## JUNIPERU                               (400,450]    ++ 0.4933 0.025317
##               mu1  logLR      w
## SALIX    0.044639  8.393 0.5344
## CORYLUS. 0.096616 17.133 0.9940
## CALLUNA  0.019421 12.719 0.5430
## BETULA   0.568254 27.575 1.0000
## RUMEXAC  0.095248 15.892 0.9888
## CYPERACE 0.080857 24.351 1.0000
## GRAMINEA 0.131311 21.569 0.9914
## EMPETRUM 0.079641 10.171 0.6500
## ARTEMISI 0.145790  7.025 0.9708
## PINUSSY  0.669499 26.667 1.0000
## ALNUSGL  0.004128  2.081 0.4201
## JUNIPERU 0.071110  4.017 0.6459
## 4 binary splits
## 1 species not shown
```

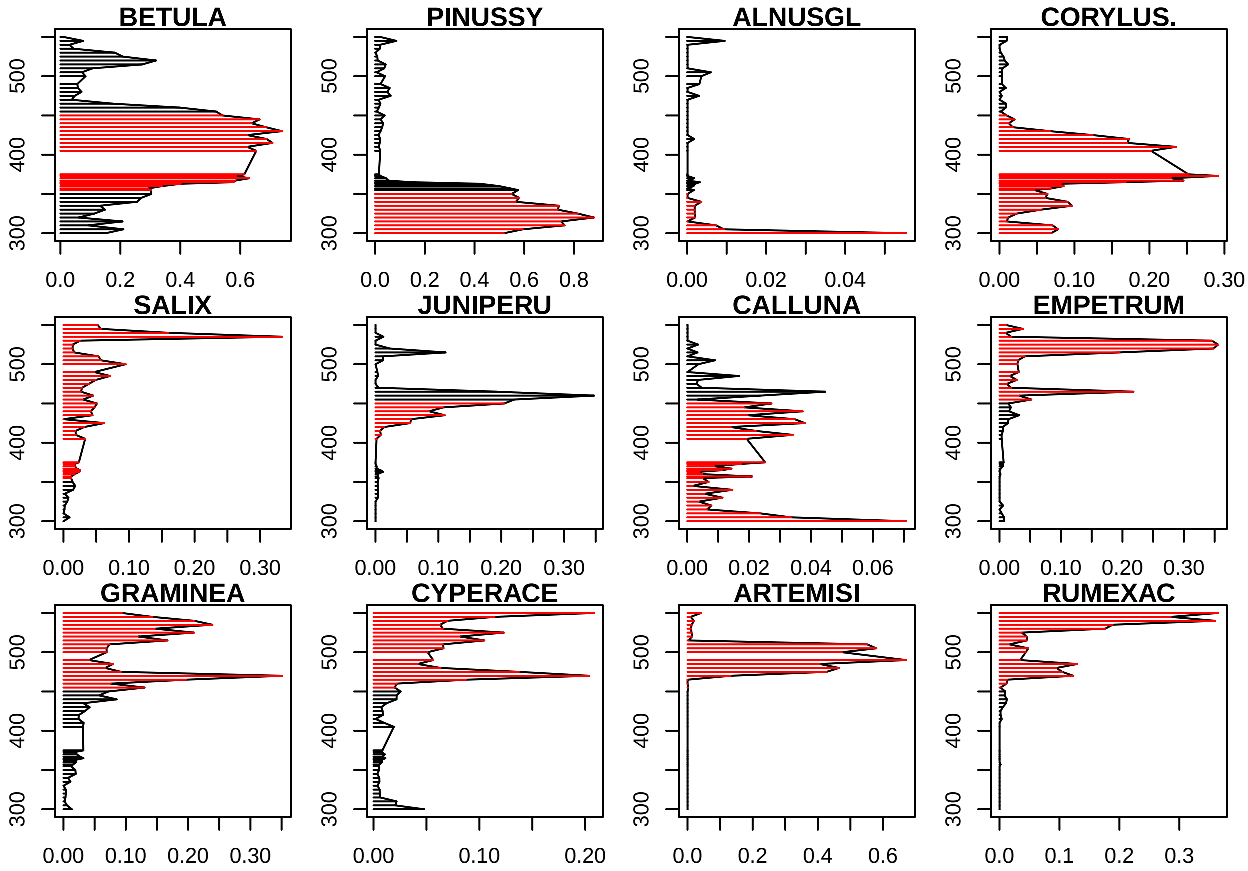
The plot shows results in a vertical orientation, resembling the stratigraphy plot.

```
plot(a, sort=FALSE, horizontal=FALSE, pos=1, upper=0.8,
    show_I=FALSE, show_S=FALSE, mar=c(6,6,1,1), xlab="", ylab="")
```



Let us combine together the best partitions with the raw data:

```
bp <- bestpart(a)
opar <- par(mfrow=c(3,4), mar=c(2,2,1,1))
for (i in 1:12) {
    plot(ab[,i], aber$ages$Depth, type="l", ann=FALSE)
    segments(x0=rep(0, nrow(ab)), y0=aber$ages$Depth, x1=ab[,i],
        col=ifelse(bp[,i] > 0, 2, 1))
    title(main=colnames(ab)[i])
}
```



```r
par(opar)
```

The following data set is from the **optpart** package (Roberts 2016b): vascular plant species cover for the Shoshone National Forest, Wyoming, USA. We will try to find species associated with different elevations.

First we try a Binomial model for 0/1 data. Note that we set the `try_error` option to `TRUE` so the function will not stop when it runs into an error.

```r
library(optpart)
```

```
## Loading required package: cluster

## Loading required package: labdsv

## Loading required package: mgcv

## Loading required package: nlme

## This is mgcv 1.8-15. For overview type 'help("mgcv-package")'.

## Loading required package: MASS

##
## Attaching package: 'labdsv'

## The following object is masked from 'package:stats':
##
```

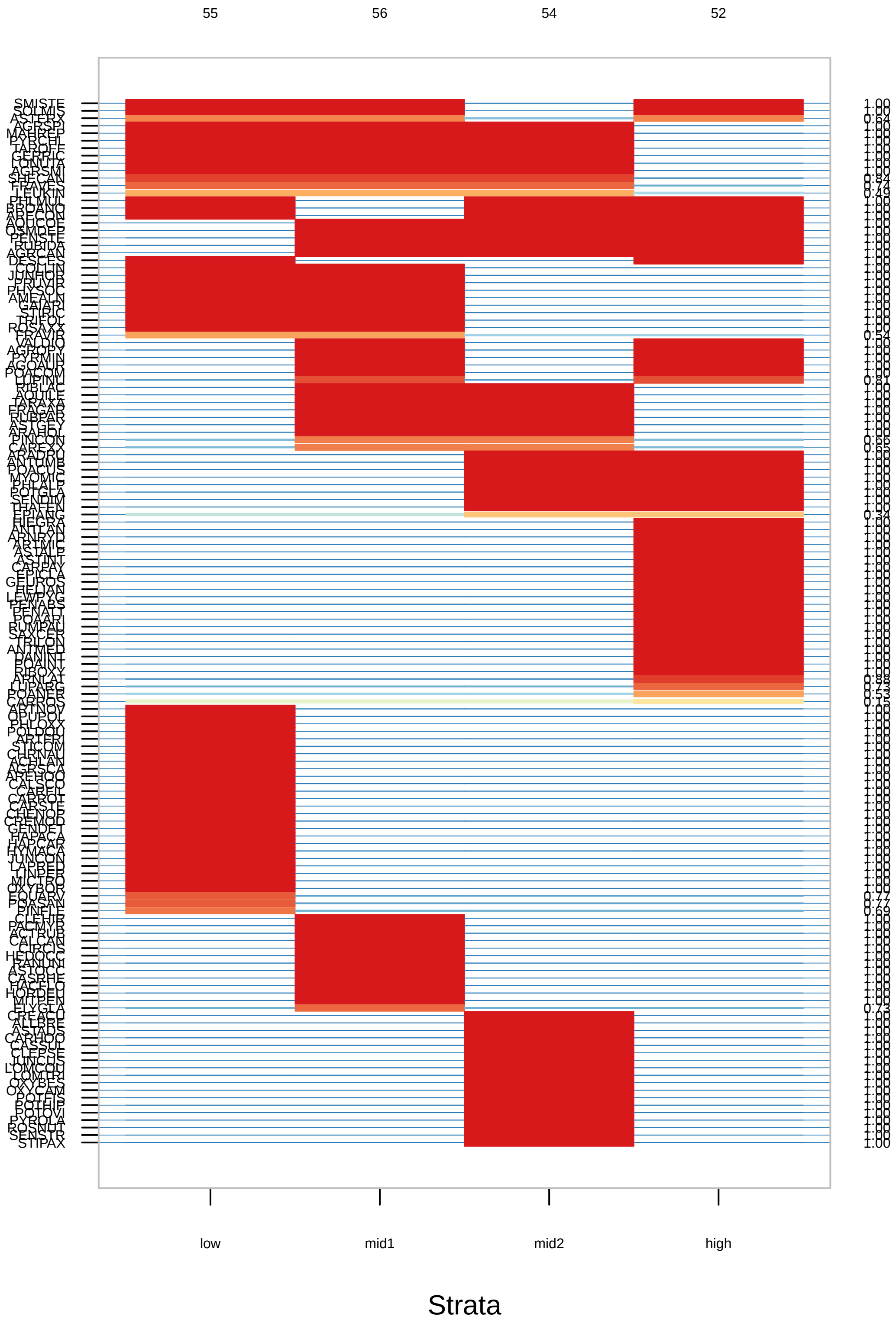
```
##     density
```

```
## Loading required package: plotrix
```

```
##
## Attaching package: 'optpart'
```

```
## The following object is masked from 'package:labdsv':
##
##     clustify
```

```
data(shoshsite)
data(shoshveg)

elev <- cut(shoshsite$elevation, breaks=c(0, 7200, 8000, 9000, 20000))
levels(elev) <- c("low","mid1","mid2", "high")

sveg <- as.matrix(shoshveg)
sveg[sveg > 0] <- 1
o <- opticut(sveg ~ 1, strata=elev, dist="binomial")
plot(o, sort=1, mar=c(4,6,2,2), cex.axis=0.5, ylab="")
```



Next, we use the same data set as percent cover data and specify a Beta distribution:

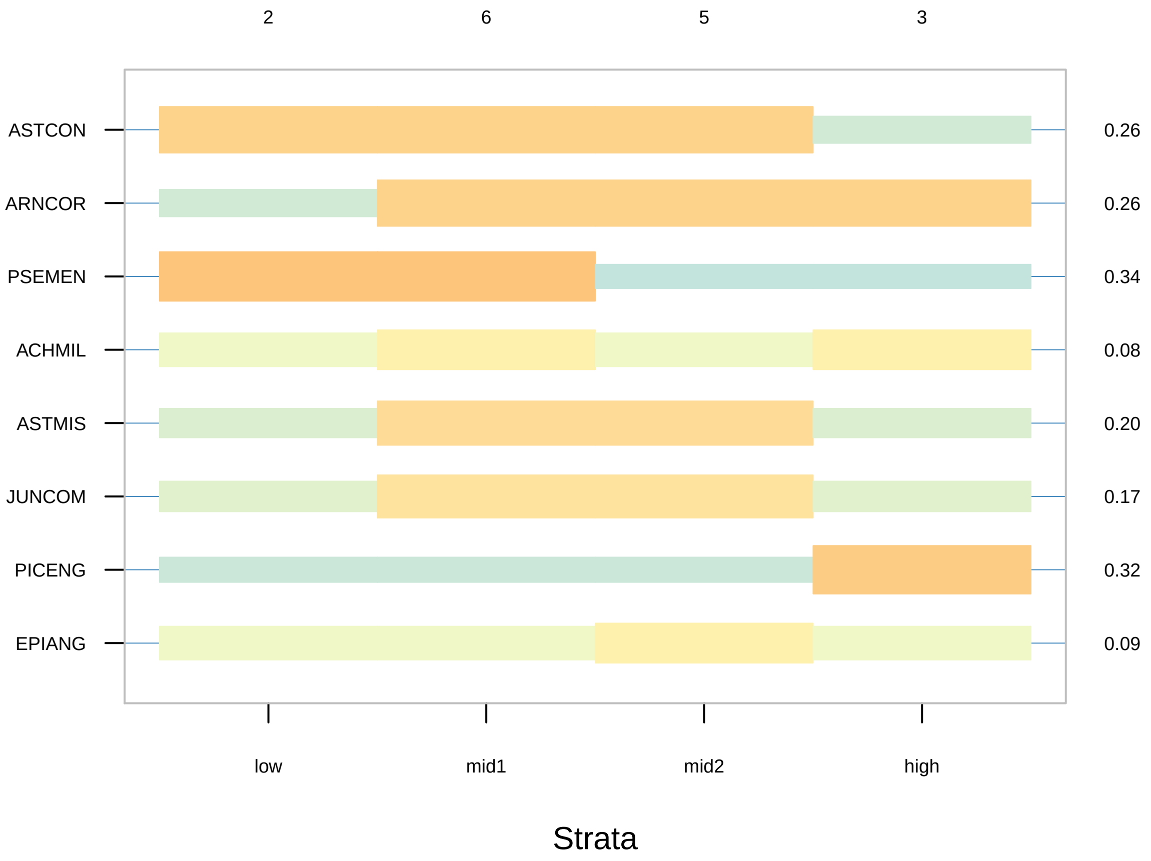
```
sveg <- as.matrix(shoshveg)
sveg <- sveg[,colSums(sveg>0) >= 50]
table(sveg)
```

```
## sveg
##    0 0.1 0.5   1   2   3   4   5   6   7   8
## 792 203 161 142  85  43  30  19  16   7   2
```

```
sveg[sveg==0] <- 0.001
sveg[sveg==0.1] <- 0.01
sveg[sveg==0.5] <- 0.05
sveg[sveg==1] <- 0.15
sveg[sveg==2] <- 0.25
sveg[sveg==3] <- 0.35
sveg[sveg==4] <- 0.45
sveg[sveg==5] <- 0.55
sveg[sveg==6] <- 0.65
sveg[sveg==7] <- 0.75
sveg[sveg==8] <- 0.8
table(sveg)
```

```
## sveg
## 0.001  0.01  0.05  0.15  0.25  0.35  0.45  0.55  0.65  0.75   0.8
##   792   203   161   142    85    43    30    19    16     7     2
```

```
o2 <- opticut(sveg ~ 1, strata=elev, dist="beta")
plot(o2, sort=1, mar=c(4,6,2,2), cex.axis=0.6, ylab="")
```



Finally, here is how an ordered logistic model can be fit to a single species. Note that number of ordered categories and baseline levels might change from one species to the other, therefore this distribution is not available for multiple species.

```
Y <- shoshveg$ASTMIS
table(Y)
```

```
## Y
##   0 0.1 0.5   1   2   3
##  82  29  17  16   5   1
```

```
o4 <- opticut(Y ~ 1, strata=elev, dist="ordered")
o4$species
```

```
## $`Sp 1`
## Univariate opticut results, comb = rank, dist = ordered
## I = 0.5699; w = 0.9902; H = 0.9806; logL_null = -192
##
## Best supported models with logLR >= 2:
##                  assoc      I    mu0    mu1 logLR        w
## mid1+mid2        +++ 0.5699 0.7076 0.8983 8.026 0.990239
## mid1              ++ 0.3991 0.5991 0.7767 2.881 0.005768
## mid1+mid2+high    ++ 0.4426 0.7245 0.8719 2.513 0.003992
## 3 binary splits
```

3.6 Presence-only (use-availability) data

Presence only data is a result from animal movement studies, or based on museum specimens. Here is an example of Mountain Goat telemetry data that were collected in the Coast Mountains of northwest British Columbia, Canada, as described in Lele and Keim (2006).

Multiple species analysis is not possible due to different number of used (presence) points for each species. It leads to a ragged-array format which is not straightforward to supply through the formula interface. The analysis of such presence-only data is also subject to conditions as explained by Solymos & Lele (2016).

According to our expectations, Mountain Goats are associated with steeper slopes:

```
library(ResourceSelection)
```

```
## ResourceSelection 0.3-0   2016-11-04
```

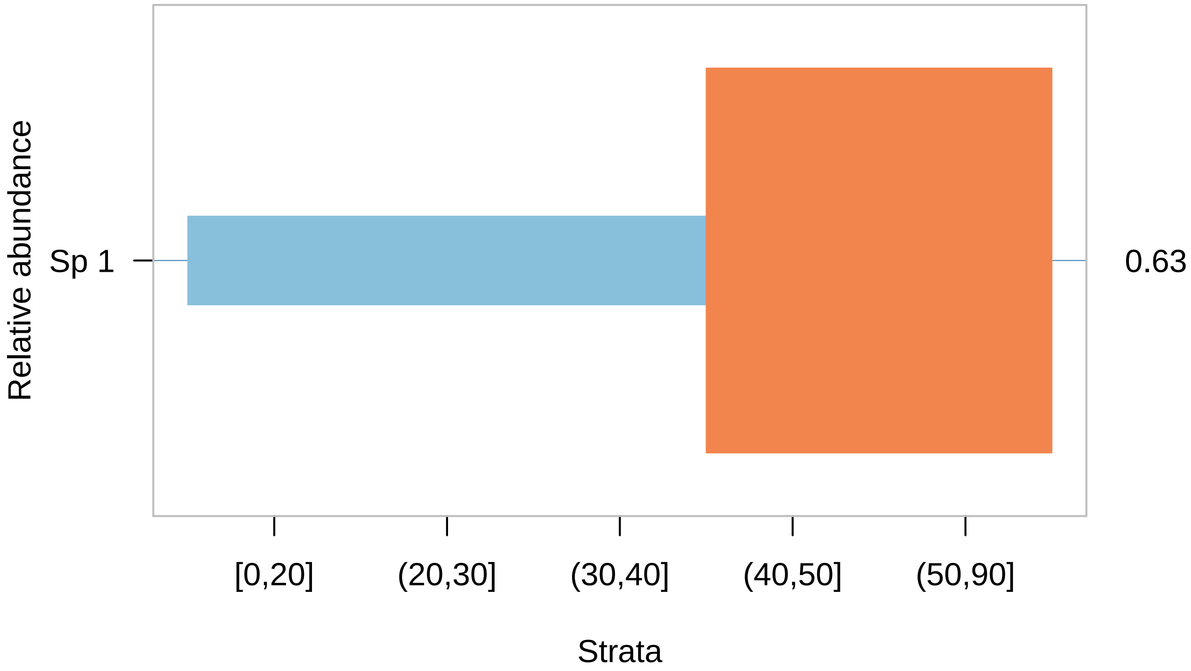
```
data(goats)
slp <- cut(goats$SLOPE,c(0, 20, 30, 40, 50, 90), include.lowest=TRUE)
table(slp, goats$STATUS)
```

```
##
## slp          0    1
##   [0,20]  2572  316
##   (20,30] 3463  691
##   (30,40] 3929 1895
##   (40,50] 2158 2347
##   (50,90]  554 1089
```

```
o <- opticut(STATUS ~ ELEVATION, data=goats, strata=slp, dist="rsf")
o$species
```

```
## $`Sp 1`
## Univariate opticut results, comb = rank, dist = rsf
## I = 0.6258; w = 1; H = 1; logL_null = -12097
##
## Best supported models with logLR >= 2:
##                                  assoc      I    mu0    mu1  logLR
## (40,50]+(50,90]                    +++ 0.6258 0.2607 1.1324 1013.1
## (30,40]+(40,50]+(50,90]            +++ 0.6567 0.1395 0.6730  988.9
## (20,30]+(30,40]+(40,50]+(50,90]   +++ 0.6589 0.1049 0.5102  451.3
## (50,90]                            +++ 0.6392 0.3759 1.7077  409.7
##                                           w
## (40,50]+(50,90]                   1.000e+00
## (30,40]+(40,50]+(50,90]           3.132e-11
## (20,30]+(30,40]+(40,50]+(50,90] 1.057e-244
## (50,90]                         8.707e-263
## 4 binary splits
```

```
plot(o, sort=FALSE, show_S=FALSE)
```



3.7 Customizing the distribution

Me may want to expand the Zero-inflation component in a ZIP model. So instead of a constant zero probability, we may want to include covariate effects to account for some variation in the counts. See how the return value needs to be structured when supplying a function as distribution.

```
data(dolina)
dolina$samp$stratum <- as.integer(dolina$samp$stratum)
Y <- dolina$xtab[,colSums(dolina$xtab > 0) >= 20]

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)
fun(Y[,"amin"], Xdol, zi_term=dolina$samp$method)
```

```
## $coef
## count_X(Intercept)     count_Xstratum      count_Xlmoist
##         1.30243481        -0.14266929         0.02460366
##     count_XmethodT   zero_(Intercept)      zero_zi_termT
##        -0.64335270        -0.56590398         0.61664601
##
## $logLik
## 'log Lik.' -788.9897 (df=6)
##
## $linkinv
## function (eta)
## .Call(C_logit_linkinv, eta)
## <environment: namespace:stats>
```

```
opticut1(Y[,"amin"], Xdol, Z=dolina$samp$mhab,
    dist=fun, zi_term=dolina$samp$method)
```

```
## Univariate opticut results, comb = rank, dist = fun
## I = 0.2652; w = 0.9849; H = 0.9703; logL_null = -789
##
## Best supported models with logLR >= 2:
##          assoc      I    mu0    mu1  logLR         w
## LI+RO      +++ 0.2652 0.7754 0.8560 15.370 9.849e-01
## RO         +++ 0.2358 0.7941 0.8618 11.189 1.505e-02
## LI+DW+RO    ++ 0.2036 0.7409 0.8121  5.328 4.286e-05
## 3 binary splits
```

```
dol2 <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist=fun, zi_term=dolina$samp$method)
summary(dol2)
```

```
## Multivariate opticut results, comb = rank, dist = fun
##
## Call:
## opticut.formula(formula = Y ~ stratum + lmoist + method, data = dolina$samp,
##     strata = dolina$samp$mhab, dist = fun, zi_term = dolina$samp$method)
##
## Best supported models with logLR >= 2:
##         split assoc      I       mu0    mu1  logLR      w
## cort DW+TL+RO   +++ 1.0000 7.247e-10 0.1302 19.933 0.6502
## hobv    DW+RO   +++ 0.9117 9.715e-02 0.6996 49.368 1.0000
## bbip    DW+RO   +++ 0.7037 3.155e-01 0.7260 45.369 1.0000
## vidi    DW+RO   +++ 0.4519 6.570e-01 0.8353 19.411 1.0000
## ctri    LI+RO   +++ 0.3951 5.638e-01 0.7488 27.661 1.0000
## amin    LI+RO   +++ 0.2652 7.754e-01 0.8560 15.370 0.9849
## ppyg    LI+RO   +++ 0.2518 8.781e-01 0.9234 41.925 1.0000
## aacu    TL+RO    ++ 0.3969 5.375e-01 0.7291  6.650 0.9738
## mbor       DW   +++ 0.9423 4.190e-01 0.9605 23.691 0.9803
## clam       DW    ++ 0.4736 3.755e-01 0.6273  5.582 0.7947
## iiso       RO   +++ 0.8562 1.757e-01 0.7334 17.176 0.9998
## ffau       RO   +++ 0.7118 1.149e-01 0.4355 20.143 0.9475
## pinc       RO   +++ 0.6322 2.465e-01 0.5921 14.673 0.5292
## estr       RO   +++ 0.6316 2.133e-01 0.5457 14.059 0.9903
## tuni       RO   +++ 0.4617 8.441e-01 0.9363 13.350 0.9996
## apur       RO   +++ 0.4326 7.859e-01 0.9026 37.612 0.9844
## pvic       RO    ++ 0.4205 5.499e-01 0.7497  7.165 0.4985
## dper       RO   +++ 0.3616 9.152e-01 0.9584 37.337 0.5118
## bcan       TL   +++ 0.7402 3.854e-01 0.8077 11.543 0.9844
## 3 binary splits
```

3.7.1 Mixed-effects models (LMM, GLMM)

Here is an example using mixed models and the package **lme4** (Bates et al. 2015) with the dolina data. Pairs of samples were taken at each sampling location using 2 methods. The `mwthod` is now a fixed factor, while location is a random intercept (see Kemencei et al. 2014 for more explanation).

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'lme4'
```

```
## The following object is masked from 'package:nlme':
##
##     lmList
```

```
lmefun <- function(Y, X, linkinv, gr, ...) {
    X <- as.matrix(X)
    m <- glmer(Y ~ X-1 + (1|gr), family=poisson("log"), ...)
    list(coef=fixef(m),
        logLik=logLik(m),
        linkinv=family(m)$linkinv)
}
```

```
lmefun(Y[,1], X=matrix(1, nrow(Y), 1), gr=dolina$samp$sample)
```

```
## $coef
##         X
## -2.785934
##
## $logLik
## 'log Lik.' -316.9976 (df=2)
##
## $linkinv
## function (eta)
## pmax(exp(eta), .Machine$double.eps)
## <environment: namespace:stats>
```

```
o <- opticut(Y[,"dper"] ~ method, data=dolina$samp, strata=dolina$samp$mhab,
    dist=lmefun, gr=dolina$samp$sample)
print(o$species[[1]], cut=-Inf)
```

```
## Univariate opticut results, comb = rank, dist = lmefun
## I = 0.6174; w = 0.7741; H = 0.6503; logL_null = -765.5
##
## Best supported models with logLR >= -Inf:
##          assoc      I    mu0   mu1  logLR         w
## DW+RO      +++ 0.6174 0.4312 1.822 21.314 7.741e-01
## DW+TL+RO   +++ 0.6222 0.3218 1.382 20.082 2.259e-01
## RO          ++ 0.4979 0.6042 1.803  7.872 1.125e-06
## 3 binary splits
```

3.7.2 Generalized additive models (GAM)

We will use the gam function from the **mgcv** package (Wood 2006) and data of Ovenbird survey counts from Solymos et al. (2012).

First off, we define the function:

```
library(mgcv)
library(detect)
```

```
## Loading required package: Formula

## Loading required package: stats4

## detect 0.4-0      2016-03-02
```

```
data(oven)
oven$veg <- factor(NA, c("open","decid","conif"))
oven$veg[oven$pforest < 0.5] <- "open"
oven$veg[oven$pforest >= 0.5 & oven$pdecid >= 0.5] <- "decid"
oven$veg[oven$pforest >= 0.5 & oven$pdecid < 0.5] <- "conif"
table(oven$veg, useNA="always")
```

```
##
##  open decid conif  <NA>
##   563   217   111     0
```

```
oven$xlat <- scale(oven$lat)
oven$xlong <- scale(oven$long)
gamfun <- function(Y, X, linkinv, Data, ...) {
    X <- as.matrix(X)
    m <- mgcv::gam(Y ~ X-1 + s(xlat) + s(xlong), Data, ...)
    list(coef=coef(m),
        logLik=logLik(m),
        linkinv=family(m)$linkinv)
}
```

Try it on a single partition: agriculture vs. not:

```
x <- ifelse(oven$veg=="agr",1,0)
X <- model.matrix(~x)
gamfun(oven$count, X, Data=oven, family=poisson)
```

```
## $coef
## X(Intercept)           Xx    s(xlat).1    s(xlat).2    s(xlat).3
##  -1.01488395   0.00000000   0.32680912   1.33149247   1.51266561
##    s(xlat).4    s(xlat).5    s(xlat).6    s(xlat).7    s(xlat).8
##   0.01960588  -0.47991711  -1.66934534  -0.02977112   2.22468794
##    s(xlat).9   s(xlong).1   s(xlong).2   s(xlong).3   s(xlong).4
##   1.69609102  -1.47468141   2.55035502   0.17726240  -0.79825825
##   s(xlong).5   s(xlong).6   s(xlong).7   s(xlong).8   s(xlong).9
##   0.40756654   0.69649220  -0.11320791  -1.65408002   1.42234371
##
## $logLik
## 'log Lik.' -777.7468 (df=16.38545)
##
## $linkinv
## function (eta)
## pmax(exp(eta), .Machine$double.eps)
## <environment: namespace:stats>
```

```
print(opticut1(oven$count, X=X[,1,drop=FALSE], oven$veg, dist=gamfun,
    Data=oven, family=poisson), cut=-Inf)
```

```
## Univariate opticut results, comb = rank, dist = gamfun
## I = 0.5801; w = 1; H = 1; logL_null = -777.7
##
## Best supported models with logLR >= -Inf:
##             assoc      I    mu0    mu1 logLR         w
## decid+conif   +++ 0.5801 0.2049 0.7712 40.37 1.000e+00
## decid         +++ 0.3546 0.2952 0.6196 18.66 3.716e-10
## 2 binary splits
```

Poisson count example with GAM:

```
o <- opticut(count ~ 1, oven, strata=veg, dist=gamfun, Data=oven, family=poisson)
summary(o)
```

```
## Multivariate opticut results, comb = rank, dist = gamfun
##
## Call:
## opticut.formula(formula = count ~ 1, data = oven, strata = veg,
##     dist = gamfun, Data = oven, family = poisson)
##
## Best supported model with logLR >= 2:
##             split assoc      I    mu0    mu1 logLR w
## Sp 1 decid+conif   +++ 0.5801 0.2049 0.7712 40.37 1
## 2 binary splits
```

```
o <- opticut(count ~ 1, oven, strata=veg, dist="poisson")
summary(o)
```

```
## Multivariate opticut results, comb = rank, dist = poisson
##
## Call:
## opticut.formula(formula = count ~ 1, data = oven, strata = veg,
##     dist = "poisson")
##
## Best supported model with logLR >= 2:
##             split assoc      I    mu0   mu1 logLR w
## Sp 1 decid+conif   +++ 0.6615 0.2149 1.055 134.1 1
## 2 binary splits
```

Binomial GAM:

```
oven$pa <- ifelse(oven$count > 0, 1, 0)
o <- opticut(pa ~ 1, oven, strata=veg, dist=gamfun, Data=oven, family=binomial)
summary(o)
```

```
## Multivariate opticut results, comb = rank, dist = gamfun
##
## Call:
## opticut.formula(formula = pa ~ 1, data = oven, strata = veg,
##     dist = gamfun, Data = oven, family = binomial)
##
## Best supported model with logLR >= 2:
##             split assoc      I    mu0   mu1 logLR w
## Sp 1 decid+conif   +++ 0.7041 0.1515 0.507 25.92 1
## 2 binary splits
```

```
o <- opticut(pa ~ 1, oven, strata=veg, dist="binomial")
summary(o)
```

```
## Multivariate opticut results, comb = rank, dist = binomial
```

```
##
## Call:
## opticut.formula(formula = pa ~ 1, data = oven, strata = veg,
##     dist = "binomial")
##
## Best supported model with logLR >= 2:
##            split assoc      I    mu0    mu1 logLR w
## Sp 1 decid+conif   +++ 0.7446 0.1528 0.5518 77.78 1
## 2 binary splits
```

3.7.3 Incorporating detectability: N-mixture models

A single-visit based N-mixture is an example where detection error is estimated (Solymos et al. 2012, Solymos & Lele 2016). We will use the Ovenbird data from the **detect** package (Solymos et al. 2016).

```
svfun <- function(Y, X, linkinv, ...) {
    X <- as.matrix(X)
    m <- detect::svabu(Y ~ X-1 | Xdet-1, ...)
    list(coef=coef(m),
        logLik=logLik(m),
        linkinv=poisson()$linkinv)
}
## detection model
oven$xjulian <- scale(oven$julian)
Xdet <- model.matrix(~ xjulian, oven)
svfun(oven$count, X=model.matrix(~ xlat + xlong, oven))
```

```
## $coef
##    sta_X(Intercept)           sta_Xxlat          sta_Xxlong
##          1.87245689          0.38389398          0.20738688
## det_Xdet(Intercept)     det_Xdetxjulian     zif_(Intercept)
##         -1.71818256         -0.37206221          0.06101481
##
## $logLik
## 'log Lik.' 155.8717 (df=6)
##
## $linkinv
## function (eta)
## pmax(exp(eta), .Machine$double.eps)
## <environment: namespace:stats>
```

Let us compare results based on naive GLM and N-mixture:

```
## naive GLM
o1 <- opticut(count ~ xlat + xlong, oven, strata=veg, dist="poisson")
print(o1$species[[1]], cut=-Inf)
```

```
## Univariate opticut results, comb = rank, dist = poisson
## I = 0.6523; w = 1; H = 1; logL_null = -903.1
##
## Best supported models with logLR >= -Inf:
##             assoc      I    mu0   mu1 logLR         w
## decid+conif   +++ 0.6523 0.2152 1.023 99.33 1.000e+00
## decid         +++ 0.5180 0.3267 1.029 58.56 1.958e-18
## 2 binary splits
```

```
## N-mixture
summary(svabu(count ~ xlat + veg | xjulian, data=oven))
```

```
##
## Call:
## svabu(formula = count ~ xlat + veg | xjulian, data = oven)
##
## Single visit Binomial - Zero Inflated Poisson model
## Conditional Maximum Likelihood estimates
##
## Coefficients for abundance (log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.5217     0.6391   2.381  0.01727 *
## xlat          0.1683     0.0632   2.663  0.00774 **
## vegdecid      0.5697     0.1892   3.011  0.00260 **
## vegconif      0.5523     0.2149   2.570  0.01018 *
## Coefficients for detection (logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.71943    0.73521  -2.339 0.019352 *
## xjulian     -0.30786    0.08322  -3.699 0.000216 ***
## Coefficients for zero inflation (logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.1678     0.1098  -1.529    0.126
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-likelihood: 116.5 on 7 Df
## AIC =  -219
```

```
o2 <- opticut(count ~ xlat + xlong, oven, strata=veg, dist=svfun)
print(o2$species[[1]], cut=-Inf)
```

```
## Univariate opticut results, comb = rank, dist = svfun
## I = 0.2576; w = 0.9608; H = 0.9246; logL_null = 155.9
##
## Best supported models with logLR >= -Inf:
##             assoc      I   mu0   mu1  logLR       w
## decid+conif     + 0.2576 4.668 7.908 -34.13 0.96076
## decid           + 0.1256 5.552 7.147 -37.33 0.03924
## 2 binary splits
```

4 Package options

4.1 High performance computing

```r
data(dolina)
dolina$samp$stratum <- as.integer(dolina$samp$stratum)
Y <- ifelse(dolina$xtab > 0, 1, 0)
dol <- opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="binomial")

## 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="binomial", comb="all", cl=NULL))
```

```
##    user  system elapsed
##   1.720   0.040   1.782
```

```r
## sequential, rank based combinations (K - 1)
system.time(opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="binomial", comb="rank", cl=NULL))
```

```
##    user  system elapsed
##   1.068   0.021   1.103
```

```r
## parallel, all combinations (2^(K-1) - 1)
system.time(opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="binomial", comb="all", cl=cl))
```

```
##    user  system elapsed
##   0.012   0.002   2.115
```

```r
## parallel, rank based combinations (K - 1)
system.time(opticut(Y ~ stratum + lmoist + method, data=dolina$samp,
    strata=dolina$samp$mhab, dist="binomial", comb="rank", cl=cl))
```

```
##    user  system elapsed
##   0.010   0.002   0.857
```

```r
## compare timings for uncertainty calculations
system.time(uncertainty(dol, type="asymp", B=999))
```

```
##    user  system elapsed
##   0.561   0.020   0.582
```

```r
system.time(uncertainty(dol, type="asymp", B=999, cl=cl))
```

```
##    user  system elapsed
##   0.027   0.004   0.462
```

```
system.time(uncertainty(dol, type="boot", B=4))
```

```
##    user  system elapsed
##   2.091   0.052   2.150
```

```
system.time(uncertainty(dol, type="boot", B=4, cl=cl))
```

```
##    user  system elapsed
##   0.025   0.003   1.381
```

```
system.time(uncertainty(dol, type="multi", B=4))
```

```
##    user  system elapsed
##   5.605   0.132   5.750
```

```
system.time(uncertainty(dol, type="multi", B=4, cl=cl))
```

```
##    user  system elapsed
##   0.025   0.002   3.937
```

```
stopCluster(cl)
```

4.2 Global options

The ocoptions function provides a convenient way of handling options related to the **opticut** package. The function takes arguments in <tag> = <value> form, or a list of tagged values. The tags must come from the parameters described below. 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.

```
## 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
str(ocoptions()) # these give identical answers
```

```
## List of 7
##  $ collapse  : chr "+"
##  $ cut       : num 2
##  $ sort      : logi TRUE
##  $ theme     : chr "br"
##  $ check_comb: logi TRUE
##  $ try_error : logi FALSE
##  $ scale     : num 0.5
```

```
str(getOption("ocoptions"))
```

```
## List of 7
##  $ collapse  : chr "+"
##  $ cut       : num 2
##  $ sort      : logi TRUE
##  $ theme     : chr "br"
##  $ check_comb: logi TRUE
##  $ try_error : logi FALSE
##  $ scale     : num 0.5
```

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

```
## Multivariate opticut results, comb = all, dist = gaussian
##
## Call:
## opticut.formula(formula = spp ~ 1, strata = gr, dist = "gaussian",
##     comb = "all")
##
## Best supported models with logLR >= 2:
##          split assoc      I mu0  mu1 logLR      w
## Species2 X1+X3   +++ 0.9866 2.0  7.0 14.82 0.4995
## Species1 X2+X3   +++ 0.8483 3.0  5.5 17.33 0.4999
## Species3    X1   +++ 1.0000 0.5 18.0 55.19 1.0000
## 15 binary splits
```

```
## resetting pboptions and checking new settings
ocop <- ocoptions(collapse="&", sort=FALSE)
str(ocoptions())
```

```
## List of 7
##  $ collapse  : chr "&"
##  $ cut       : num 2
##  $ sort      : logi FALSE
##  $ theme     : chr "br"
##  $ check_comb: logi TRUE
##  $ try_error : logi FALSE
##  $ scale     : num 0.5
```

```
## running again with new settings
summary(ocall <- opticut(spp ~ 1, strata=gr, dist="gaussian", comb="all"))
```

```
## Multivariate opticut results, comb = all, dist = gaussian
##
## Call:
## opticut.formula(formula = spp ~ 1, strata = gr, dist = "gaussian",
##     comb = "all")
##
## Best supported models with logLR >= 2:
##          split assoc      I mu0  mu1 logLR      w
## Species1 X2&X3   +++ 0.8483 3.0  5.5 17.33 0.4999
## Species2 X1&X3   +++ 0.9866 2.0  7.0 14.82 0.4995
## Species3    X1   +++ 1.0000 0.5 18.0 55.19 1.0000
## 15 binary splits
```

```
## resetting original
ocoptions(ocop)
str(ocoptions())
```

```
## List of 7
##  $ collapse  : chr "+"
##  $ cut       : num 2
##  $ sort      : logi TRUE
##  $ theme     : chr "br"
##  $ check_comb: logi TRUE
##  $ try_error : logi FALSE
##  $ scale     : num 0.5
```

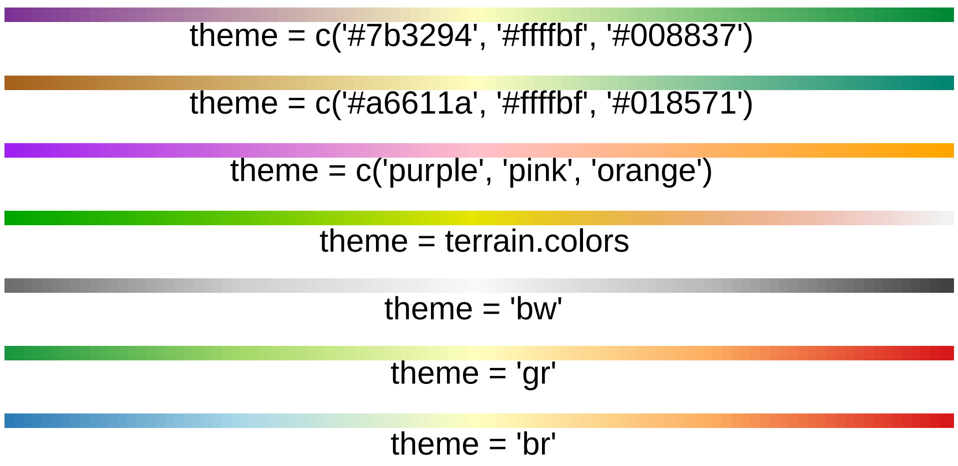
4.3 Color themes

The **opticut** package uses color themes for plotting and provides a convenient way of setting color palettes via the `occolors` function. The function takes a single `theme` argument and returns a function as in `colorRampPalette`.

The `theme` argument can be a 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). Hexadecimal values for the built-in palettes are taken from http://colorbrewer2.org.

```
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'")
```

```r
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')")
```



4.4 Progress bar

The expected completion time of extensive calculations and the progress is shown by the progress bar via the **pbapply** package. Default options with **opticut** are:

```r
str(pboptions())
```

```
## List of 10
##  $ type     : chr "none"
##  $ char     : chr "[=-]"
##  $ txt.width: num 50
##  $ gui.width: num 300
##  $ style    : num 6
##  $ initial  : num 0
##  $ title    : chr "R progress bar"
##  $ label    : chr ""
##  $ nout     : int 100
##  $ min_time : num 2
```

See `?pboptions` for a description of these options. Use `pboptions(type = "none")` to turn off the progress bar in interactive R sessions. The progress bar is automatically turned off during non-interactive sessions.

4.5 Dynamic documents

opticut object summaries come with an `as.data.frame` method that can be used to turn the summary into a data frame, which is what for example the `kable` function from **knitr** package expects. This way, formatting the output is much facilitated, and the user does not have to dig into the structure of the summary object.

The GitHub repository has a minimal Rmarkdown example do demonstrate how to format **opticut** objects for best effects: Rmd source, knitted PDF.

```
library(knitr)

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))
g <-    c(1,1,1,1, 2,2,2,2, 3,3,3,3)
oc <- opticut(formula = y ~ 1, strata = g, dist = "poisson")
uc <- uncertainty(oc, type = "asymp", B = 999)

print(kable(as.data.frame(oc), digits=3))
```

```
##
##
##       split   assoc        I    mu0    mu1   logLR       w
## ----  ------  ------  ------  -----  -----  ------  ------
## Sp3   2+3     ++       0.647   0.75   3.50   4.793   0.696
## Sp2   3       +++      0.857   0.25   3.25   9.203   0.958
```

```
print(kable(oc$species[[1]][,c(1,2,4,5,8,9,10)], digits=3))
```

```
##
##
##        assoc       I   mu0   mu1      logL   logLR       w
## ----  ------  ------  ----  ----  --------  ------  ------
## 1          1   0.125   3.5   4.5   -22.185   0.339   0.239
## 1+2        1   0.286   2.5   4.5   -21.026   1.498   0.761
```

```
print(kable(as.data.frame(uc), digits=3))
```

```
##
##
```

```
##        split     R       I    Lower    Upper
## ----  ------   ---  ------  ------  ------
## Sp1   1+2        1   0.278   0.025   0.554
## Sp3   2+3        1   0.616   0.187   0.878
## Sp2   3          1   0.823   0.468   0.965
```

The `kable` output is rendered as nice tables (without the `print` part):

```
kable(as.data.frame(oc), digits=3)
```

	split	assoc	I	mu0	mu1	logLR	w
Sp3	2+3	++	0.647	0.75	3.50	4.793	0.696
Sp2	3	+++	0.857	0.25	3.25	9.203	0.958

```
kable(oc$species[[1]][,c(1,2,4,5,8,9,10)], digits=3)
```

	assoc	I	mu0	mu1	logL	logLR	w
1	1	0.125	3.5	4.5	-22.185	0.339	0.239
1+2	1	0.286	2.5	4.5	-21.026	1.498	0.761

```
kable(as.data.frame(uc), digits=3)
```

	split	R	I	Lower	Upper
Sp1	1+2	1	0.278	0.025	0.554
Sp3	2+3	1	0.616	0.187	0.878
Sp2	3	1	0.823	0.468	0.965

5 Summary

The likelihood-based optimal partitioning framework implemented in the opticut R package provides a compelling alternative to other available R packages (**indicspecies**, De Caceres & Legendre 2009; **labdsv** and **optpart**, Roberts 2016a, 2016b; **vegan**, Oksanen et al. 2016), especially when the type of data or the presence of modifying effects call for an alternative approach.

The package comes with many parametric models included for binary, count, abundance, percent cover, ordinal, and presence-only data, and the approach can be extended to more complex situations, such as mixed models, additive models. The opticut package leverages other R packages (**MASS**, Venables & Ripley 2002; **betareg**, Cribari-Neto & Zeileis 2010; **pscl**, Zeileis et al. 2008; **ResourceSelection**, Lele et al. 2016) for fitting parametric models. The approach can be extended to linear and generalized linear mixed models (LMM, GLMM; see Kemencei et al. 2014), generalized additive models, N-mixture models.

Computing times are shortened by the application of efficient algorithms and through high performance computing options. The **opticut** package provides progress bars with estimated remaining time for long-running evaluations (through the **pbapply** package, Solymos & Zawadzki 2016), natively supports several parallel back-ends (multicore machines, computing clusters, forking; through the **parallel** package, R Core Team 2016) to speed calculations up, and provides options and methods for dynamic report generation (coercion methods and color themes). Functions in the package can be customized and extended to meet the needs under a wide range of real-world situations.

Please cite the **opticut** package in scholarly publications as:

Peter Solymos and Ermias T. Azeria (2016). opticut: Likelihood Based Optimal Partitioning for Indicator Species Analysis. R package version <insert appropriate version number here>. https://github.com/psolymos/opticut

Use `citation("opticut")` for an up-to-date citation.

6 References

Bates, D., Maechler, M., Bolker, B. & Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software* 67(1): 1–48.

Birks, H. H. & Mathews, R. W. (1978). Studies in the vegetational history of Scotland V. Late Devensian and early Flandrian macrofossil stratigraphy at Abernethy Forest, Invernessshire. *New Phytologist* 80: 455–84.

Chytry, M., Tichy, L., Holt, J. & Botta-Dukat, Z. (2002). Determination of diagnostic species with statistical fidelity measures. *Journal of Vegetation Science* 13: 79–90.

Cribari-Neto, F. & Zeileis, A. (2010). Beta Regression in R. *Journal of Statistical Software* 34(2): 1–24. http://www.jstatsoft.org/v34/i02/.

De Caceres, M. & Legendre, P. (2009). Associations between species and groups of sites: Indices and statistical inference. *Ecology* 90: 3566–3574.

Dufrene, M. & Legendre, P. (1997) Species assemblages and indicator species: The need for a flexible asymmetrical approach. *Ecological Monographs* 67: 345–366.

Halme, P., Monkkonen, M., Kotiaho, J. S, Ylisirnio, A-L. (2009). Quantifying the indicator power of an indicator species. *Conservation Biology* 23: 1008–1016.

Juggins, S. (2015). rioja: Analysis of Quaternary Science Data, R package version 0.9-9. http://cran.r-project.org/package=rioja

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.

Lele, S. R., Keim, J. L. & Solymos, P. (2016). ResourceSelection: Resource Selection (Probability) Functions for Use-Availability Data. R package version 0.3-0. https://CRAN.R-project.org/package=ResourceSelection

McGeoch, M. A. & Chown, S. L. (1998). Scaling up the value of bioindicators. *Trends in Ecology and Evolution* 13: 46–47.

Oksanen, J., Blanchet, F. G., Friendly, M., Kindt, R., Legendre, P., McGlinn, D., Minchin, P. R., O'Hara, R. B., Simpson, G. L., Solymos, P., Stevens, M. H. H., Szoecs, E. & Wagner, H. (2016). vegan: Community Ecology Package. R package version 2.5-0. https://CRAN.R-project.org/package=vegan

R Core Team (2016). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Roberts, D. W. (2016a). labdsv: Ordination and Multivariate Analysis for Ecology. R package version 1.8-0. https://CRAN.R-project.org/package=labdsv

Roberts, D. W. (2016b). optpart: Optimal Partitioning of Similarity Relations. R package version 2.3-0. https://CRAN.R-project.org/package=optpart

Solymos, P., Moreno, M. & Lele, S. R. (2016). detect: Analyzing Wildlife Data with Detection Error. R package version 0.4-0. https://github.com/psolymos/detect

Solymos, P., Lele, S. R. & Bayne, E. (2012). Conditional likelihood approach for analyzing single visit abundance survey data in the presence of zero inflation and detection error. *Environmetrics* 23: 197–205.

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.

Solymos, P. & Zawadzki, Z. (2016). pbapply: Adding Progress Bar to '*apply' Functions. R package version 1.3-2. https://github.com/psolymos/pbapply

Venables, W. N. & Ripley, B. D. (2002). *Modern Applied Statistics with S*. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

Wildi, O. & Feldmeyer-Christe, E. (2013). Indicator values (IndVal) mimic ranking by F-ratio in real-world vegetation data. *Community Ecology* 14(2): 139–143.

Wood, S. N. (2006). *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC.

Zeileis, A., Kleiber, C. & Jackman, S. (2008). Regression Models for Count Data in R. *Journal of Statistical Software*, 27(8), 1-25. http://www.jstatsoft.org/v27/i08/

Zettler, M. L., Proffitt, C. E., Darr, A., Degraer, S., Devriese, L., Greathead, C., Kotta, J., Magni, P., Martin, G., Reiss, H., Speybroeck, J., Tagliapietra, D., Van Hoey, G. & Ysebaert, T. (2013). On the Myths of Indicator Species: Issues and Further Consideration in the Use of Static Concepts for Ecological Applications. *PLoS ONE*, 8(10):e78219.