

opticut: likelihood based optimal partitioning for indicator species analysis

Peter Solymos, Ermias T. Azeria

November 17, 2015

Contents

1	Introduction	1
2	Theory	2
2.1	The quest for optimal binary partitioning	2
2.2	Finding all possible binary partitions	3
2.3	Poisson count model example	5
2.4	Not using all possible partitions	8
3	Distributions	9
3.1	Gaussian	10
3.2	Binomial	12
3.3	Poisson: Mite data set – high performance computing	14
3.4	Percentages	17
3.5	Presence-only data	32
4	Custom distributions	33
4.1	Mixed models	33
4.2	Imperfect detectability: N-mixture case	34
4.3	Sampling differences: using offsets	36
4.4	GAM models	38
5	Finding best partitions	40
6	Uncertainty	41
7	Opticut logo	43

1 Introduction

General problem: find where species abundances are high vs. low in a way which leads to optimal classification by maximizing the contrast between the partitions.

Previous attempts: historical review, highlighting IndVal.

Issues with previous attempts:

- summary statistics & Monte Carlo randomization with p-values, no model,
- data types not always compatible with randomization (i.e. decimals),
- confounding effects to classification can impact power,
- assessing the ranking of partitions without inferential statements.

Goals:

- describe a general and extensible approach that addresses the above limitations,
- implement a computationally efficient algorithm,
- tools for exploring the results (i.e. summaries, plots) in a object oriented framework.

Why opticut?

- HPC is natively supported
- efficient for large number of partitions
- lots of models defined, extensible
- uncertainty and partitioning reliability

2 Theory

2.1 The quest for optimal binary partitioning

Y_i 's are observations for a single species from n locations ($i = 1, \dots, n$). g_i 's are known discrete descriptors of the locations with K levels ($K > 2$). $z_i^{(m)}$ is a binary reclassification of g taking values (0, 1). The superscript $m = 1, \dots, M$ indicates a possible combination of binary reclassification out of the total $M = 2^{K-1} - 1$ total combinations (excluding complements). See below for options for defining binary partitions. There can also be other site descriptors denoted as x_{ij} taking discrete or continuous values ($j = 1, \dots, p$; number of predictors).

A suitable parametric model describes the relationship between the observations and the site descriptors through the probability density function $P(Y_i = y_i \mid z_i^{(m)}, x_{ij}, \theta)$ where θ is the vector of model parameters: $\theta = (\beta_0, \beta_1, \alpha_1, \dots, \alpha_p)$. The choice of the parametric model depends on the nature of the observations. It can be Gaussian, Binomial, Poisson, ordinal, Beta regression, or zero-inflated models, with a suitable link function (f) for the mean: $f(\eta_i) = \beta_0^{(m)} + \beta_1^{(m)} z_i^{(m)} + \sum_{j=1}^p \alpha_j^{(m)} x_{ij}$.

$\widehat{\theta^{(m)}}$ is the maximum likelihood estimate (MLE) of the model parameters given the data and classification m , with corresponding log-likelihood value $l(\widehat{\theta^{(m)}}; y)$. Finding MLEs for all M candidate binary partitions leads to a set of log-likelihood values. One can compare the log-likelihood values to a null model (no binary partition is necessary) where $\beta_1 = 0$ leading to the MLE $\widehat{\theta^{(0)}}$ and corresponding log-likelihood value for the null model: $l(\widehat{\theta^{(0)}}; y)$.

The log-likelihood ratio for each candidate partition can be calculated as $l(\widehat{\theta^{(m)}}; y) - l(\widehat{\theta^{(0)}}; y)$. The best supported binary partition is the model with the highest log-likelihood ratio value.

One way of calculating the indicator value for each candidate partition is based on expected values using the inverse link function as $\mu_0^{(m)} = f^{-1}(\beta_0^{(m)})$ and $\mu_1^{(m)} = f^{-1}(\beta_0^{(m)} + \beta_1^{(m)})$. $I = 1 - \min(\mu_0^{(m)}, \mu_1^{(m)}) / \max(\mu_0^{(m)}, \mu_1^{(m)})$. Where $\mu_0^{(m)} = E[Y_i \mid z_i^{(m)} = 0, x_{ij} = 0]$ and $\mu_1^{(m)} = E[Y_i \mid z_i^{(m)} = 1, x_{ij} = 0]$ are expected values for the observations given the binary partition $z_i^{(m)}$ and at 0 value for all x_{ij} . This approach can be sensitive to the range of values supported by the link function. For example it works nicely with logarithmic or logistic link function where non-negativity of predicted values is ensured by definition. This is, however, not the case for the identity link in the Gaussian case, when negative values can

invalidate the indicator value calculations as described above. (This usually happens when confounding variables are not centered and the intercept then reflects that difference as part of the baseline.)

As an alternative, one can use the estimate $\beta_1^{(m)}$ itself to express the contrast between the two strata. This also makes the index more comparable when different link functions are used. We used the hyperbolic tangent function (or inverse Fisher's z transform) to scale the real valued $\beta_1^{(m)}$ into the unit range (0-1): $I = \tanh(|\beta_1^{(m)}|) = \frac{\exp(2|\beta_1^{(m)}|)-1}{\exp(2|\beta_1^{(m)}|)+1}$. Positive and negative cases are taken as absolute values, so that the index reflects only the contrast between strata, and not the direction of it. Negative value can happen when using all combinations.

2.2 Finding all possible binary partitions

Finding all combinations does not require a model or observed responses. It only takes a classification vector with $K > 1$ partitions.

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 $M = 2^{K-1} - 1$.

Get the package and load it:

```
#devtools::install_github("psolymos/opticut")
#devtools::install("~/repos/opticut")
#devtools::check("~/repos/opticut")
#devtools::build("~/repos/opticut", binary=TRUE)
library(opticut)
```

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

```
## opticut 0.1-2      2016-09-14
```

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

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

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

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

```
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` this takes a classification vector with at least 2 levels and returns a model matrix with binary partitions. `checkComb` checks if combinations are unique and non-complementary (misfits are returned as attributes).

```
(f <- rep(LETTERS[1:4], each=2))
```

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

```
(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"
```

```
checkComb(mc)
```

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

```
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)
```

```
## [1] FALSE
## attr("comp")
##      i j
```

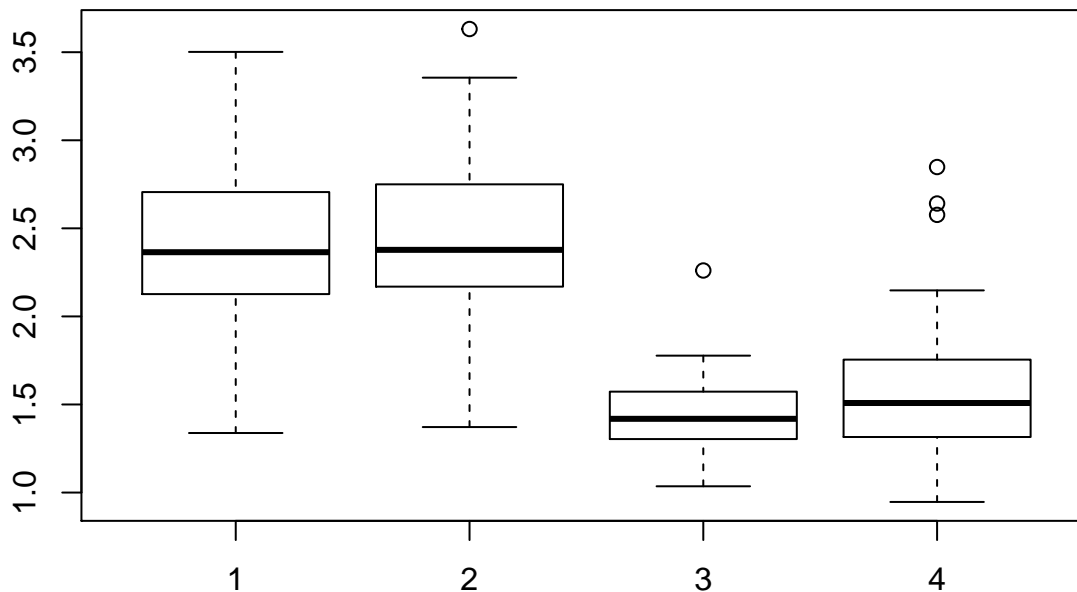
```
## [1,] 1 2
## [2,] 1 9
## attr("same")
##      i j
## [1,] 9 2
```

2.3 Poisson count model example

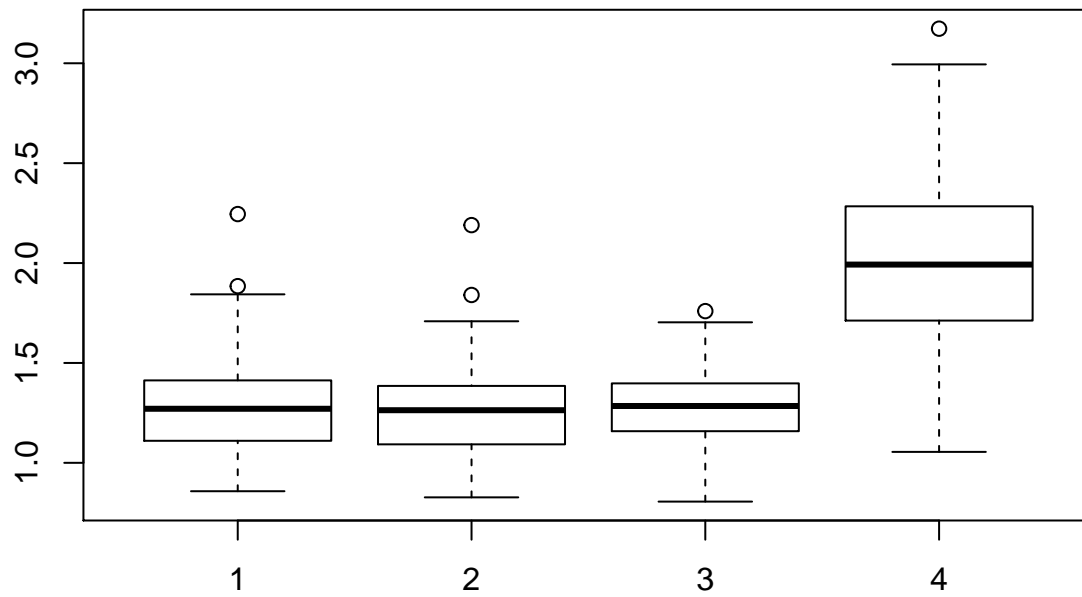
```
set.seed(1234)
n <- 200
x0 <- sample(1:4, n, TRUE)
x1 <- ifelse(x0 %in% 1:2, 1, 0)
x2 <- rnorm(n, 0.5, 1)
table(x0,x1)
```

```
##      x1
## x0    0  1
##   1    0 52
##   2    0 51
##   3   51  0
##   4   46  0
```

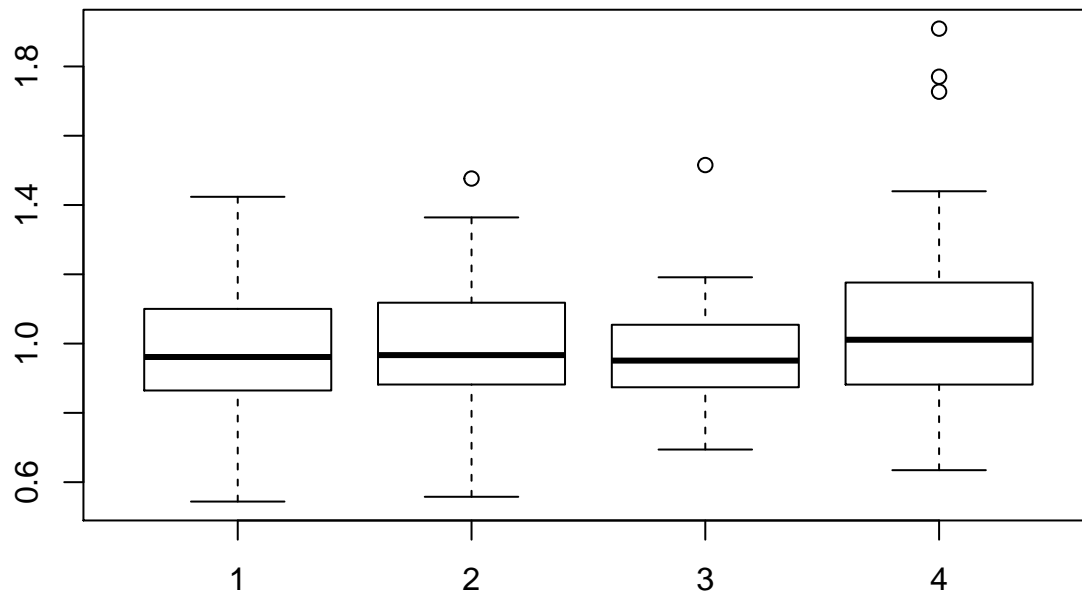
```
lam1 <- exp(0.5 + 0.5*x1 + -0.2*x2)
boxplot(lam1~x0)
```



```
Y1 <- rpois(n, lam1)
lam2 <- exp(0.1 + 0.5*ifelse(x0==4,1,0) + 0.2*x2)
boxplot(lam2~x0)
```



```
Y2 <- rpois(n, lam2)
lam3 <- exp(0.1 + -0.2*x2)
boxplot(lam3~x0)
```



```
Y3 <- rpois(n, lam3)
Y <- cbind(SPP1=Y1, SPP2=Y2, SPP3=Y3)
X <- model.matrix(~x2)
Z <- allComb(x0)
opticut1(Y1, X, Z, dist="poisson")
```

```
## Univariate opticut results, comb = all, dist = poisson
## I = 0.3972; w = 0.9842; H = 0.9687; logL_null = -343.8
##
## Best supported models with logLR >= 2:
```

```
##      assoc      I   mu0   mu1 logLR      w
## 1 2    +++ 0.3972 1.739 2.647 8.396 0.984200
## 3      -- 0.3006 2.337 1.714 3.051 0.004694
## 4      -- 0.3005 2.371 1.739 3.027 0.004583
## 2      ++ 0.2505 2.048 2.645 2.633 0.003092
## 1      ++ 0.2483 2.048 2.639 2.598 0.002987
## 7 binary splits (2 models not shown)
```

```
opticut1(Y2, X, Z, dist="poisson")
```

```
## Univariate opticut results, comb = all, dist = poisson
## I = 0.4326; w = 0.9254; H = 0.8584; logL_null = -315.6
##
## Best supported models with logLR >= 2:
##      assoc      I   mu0   mu1 logLR      w
## 4      ++ 0.4326 1.134 1.803 6.245 0.92545
## 1 3      -- 0.2863 1.486 1.107 3.119 0.04063
## 1 2      -- 0.2342 1.458 1.149 2.064 0.01415
## 7 binary splits (4 models not shown)
```

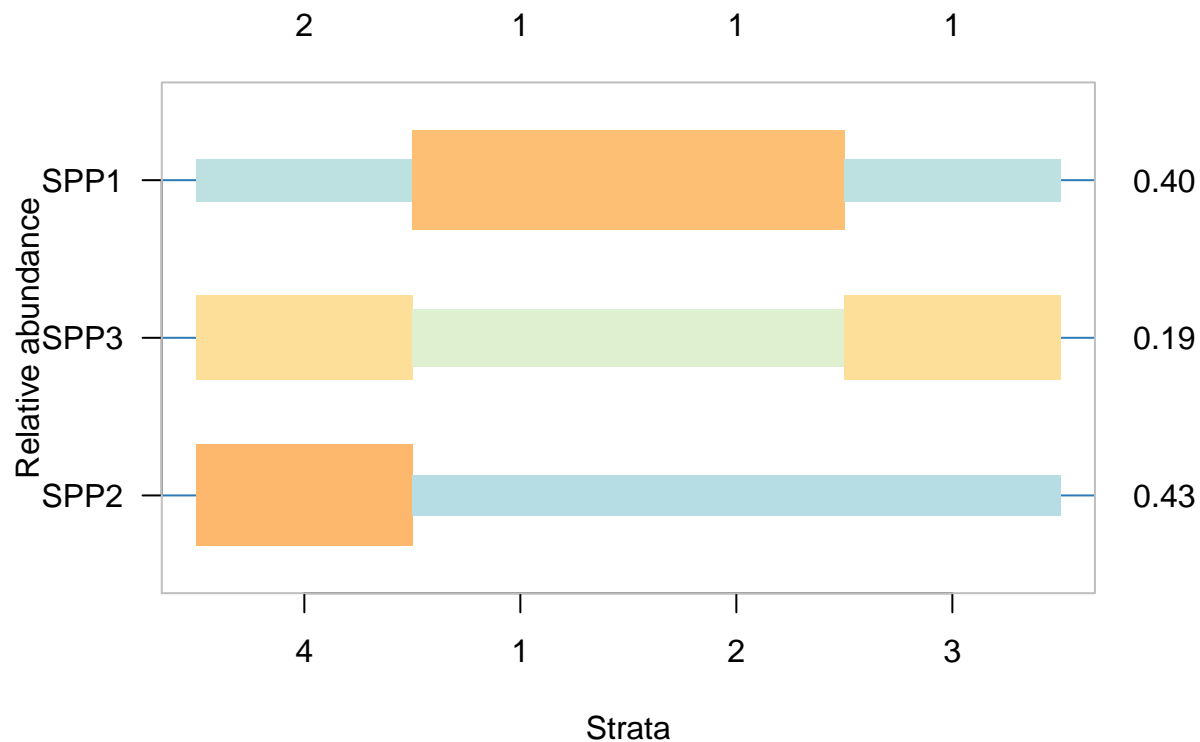
```
opticut1(Y3, X, Z, dist="poisson")
```

```
## Univariate opticut results, comb = all, dist = poisson
## I = 0.1899; w = 0.2205; H = 0.1562; logL_null = -244.4
##
## Best supported model:
##      assoc      I   mu0   mu1 logLR      w
## 1 2      - 0.1899 1.096 0.9047 0.805 0.2205
## 7 binary splits (6 models not shown)
```

```
summary(m <- opticut(Y ~ x2, strata=x0, dist="poisson", comb="all"))
```

```
## Multivariate opticut results, comb = all, dist = poisson
##
## Call:
## opticut(formula = Y ~ x2, strata = x0, dist = "poisson", comb = "all")
##
## Best supported models with logLR >= 2:
##      split assoc      I   mu0   mu1 logLR      w
## SPP1   1 2    +++ 0.3972 1.739 2.647 8.396 0.9842
## SPP2    4      ++ 0.4326 1.134 1.803 6.245 0.9254
## 7 binary splits
## 1 species not shown
```

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



Describe here what is what in the output.

2.4 Not using all possible partitions

Blindly fitting a model to all possible partitions is wasteful use of resources. Instead, one can rank the K partitions based on expected response values ($\mu_1, \dots, \mu_k, \dots, \mu_K$, where $\mu_k = E[Y_i | g_i = k, x_{ij} = 0]$). This way we have to explore only $K - 1$ partitions:

```
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
## attr("rank")
## 1 2 3 4
## 1 2 3 4
```

`oComb` return the 'contrast' matrix based on the rank vector as input. Rank 1 means lowest 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 similarly to `allComb`:

```
head(rc <- rankComb(Y1, 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")
```

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

Note that the ranking varies from species to species, thus it is not possible to supply the resulting matrix as strata definition:

```
summary(opticut(Y ~ x2, strata=x0, dist="poisson", comb="rank"))
```

```
## Multivariate opticut results, comb = rank, dist = poisson
##
## Call:
## opticut(formula = Y ~ x2, strata = x0, dist = "poisson", comb = "rank")
##
## Best supported models with logLR >= 2:
##      split assoc      I   mu0   mu1 logLR      w
## SPP1    1 2    +++ 0.3972 1.739 2.647 8.396 0.9922
## SPP2     4    ++ 0.4326 1.134 1.803 6.245 0.9505
## 3 binary splits
## 1 species not shown
```

There is an overhead of fitting the model to calculate the ranking. But computing efficiencies can be still high compared to all partitions when the number of levels (k) is high.

A downside of this approach is that not all possible partitions are explored, thus the model weights do not represent all possible models, but only the top candidates. Thus model weight interpretation is different (i.e. cannot be used as a reliability matrix, especially when support for the best model is not dominant).

3 Distributions

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", "zinb": 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,
- "rsf", "rspf": presence-only data using resource selection and resource selection probability functions.

3.1 Gaussian

```
Y <- rnorm(n, log(lam1) + 10, 0.5)
(mod <- opticut(Y ~ x2, strata=x0, dist="gaussian"))

## Multivariate opticut results, comb = rank, dist = gaussian
##
## Call:
## opticut(formula = Y ~ x2, strata = x0, dist = "gaussian")
##
## 1 species, 3 binary splits
```

Legendre example

```
gr <- rep(1:5, each=5)
spp <- cbind(Sp1=rep(c(4,6,5,3,2), each=5),
             Sp2=c(rep(c(8,4,6), each=5), 4,4,2, rep(0,7)),
             Sp3=rep(c(18,2,0,0,0), each=5))
rownames(spp) <- gr
spp
```

```
##   Sp1 Sp2 Sp3
## 1   4   8  18
## 1   4   8  18
## 1   4   8  18
## 1   4   8  18
## 1   4   8  18
## 2   6   4   2
## 2   6   4   2
## 2   6   4   2
## 2   6   4   2
## 2   6   4   2
## 3   5   6   0
## 3   5   6   0
## 3   5   6   0
## 3   5   6   0
## 3   5   6   0
## 4   3   4   0
## 4   3   4   0
## 4   3   2   0
## 4   3   0   0
## 4   3   0   0
## 5   2   0   0
## 5   2   0   0
## 5   2   0   0
## 5   2   0   0
## 5   2   0   0
```

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

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

```
##
## Call:
## opticut(formula = spp ~ 1, strata = gr, dist = "gaussian", comb = "all")
##
## Best supported models with logLR >= 2:
##      split assoc      I mu0  mu1 logLR      w
## Sp2    1 3    +++ 0.9999 2.0   7.0 14.82 0.4995
## Sp1    2 3    +++ 0.9866 3.0   5.5 17.33 0.4999
## Sp3     1    +++ 1.0000 0.5  18.0 55.19 1.0000
## 15 binary splits
```

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

```
## Multivariate opticut results, comb = rank, dist = gaussian
##
## Call:
## opticut(formula = spp ~ 1, strata = gr, dist = "gaussian", comb = "rank")
##
## Best supported models with logLR >= 2:
##      split assoc      I mu0  mu1 logLR      w
## Sp2    1 3    +++ 0.9999 2.0   7.0 14.82 0.4996
## Sp1    2 3    +++ 0.9866 3.0   5.5 17.33 0.4999
## Sp3     1    +++ 1.0000 0.5  18.0 55.19 1.0000
## 4 binary splits
```

```
## DeCaceres & Legendre 2013 Oikos example from Fig 1
## Oikos 119: 1674-1684, 2010
## doi: 10.1111/j.1600-0706.2010.18334.x
Y <- c(0, 0, 3, 0, 2, 3, 0, 5, 5, 6, 3, 4)
z <- rep(1:3, each=4)
Z <- allComb(z)
Z <- cbind(Z, 1-Z)
colnames(Z) <- c("1", "2", "3", "2 3", "1 3", "1 2")
Z
```

```
##      1 2 3 2 3 1 3 1 2
## 1 1 0 0 0 1 1
## 1 1 0 0 0 1 1
## 1 1 0 0 0 1 1
## 1 1 0 0 0 1 1
## 2 0 1 0 1 0 1
## 2 0 1 0 1 0 1
## 2 0 1 0 1 0 1
## 2 0 1 0 1 0 1
## 3 0 0 1 1 1 0
## 3 0 0 1 1 1 0
## 3 0 0 1 1 1 0
## 3 0 0 1 1 1 0
```

```
try(opticut1(Y, Z=Z))
oc <- ocoptions(check_comb=FALSE, cut=-Inf) # relax the checks
opticut1(Y, Z=Z) # identical results for complementary partitions
```

```
## Univariate opticut results, comb = NA, dist = gaussian
## I = 0.9937; w = 0.2871; H = 0.2463; logL_null = -25.93
##
## Best supported models with logLR >= -Inf:
##      assoc      I    mu0    mu1    logLR      w
## 3      ++ 0.9937 1.625 4.500 3.232629 0.28708
## 1 2     -- 0.9937 4.500 1.625 3.232629 0.28708
## 1      -- 0.9919 3.500 0.750 2.878892 0.20154
## 2 3     ++ 0.9919 0.750 3.500 2.878892 0.20154
## 2      - 0.1244 2.625 2.500 0.004726 0.01138
## 1 3     + 0.1244 2.500 2.625 0.004726 0.01138
## 6 binary splits
```

```
ocoptions(oc) # restore defaults
```

BCI data

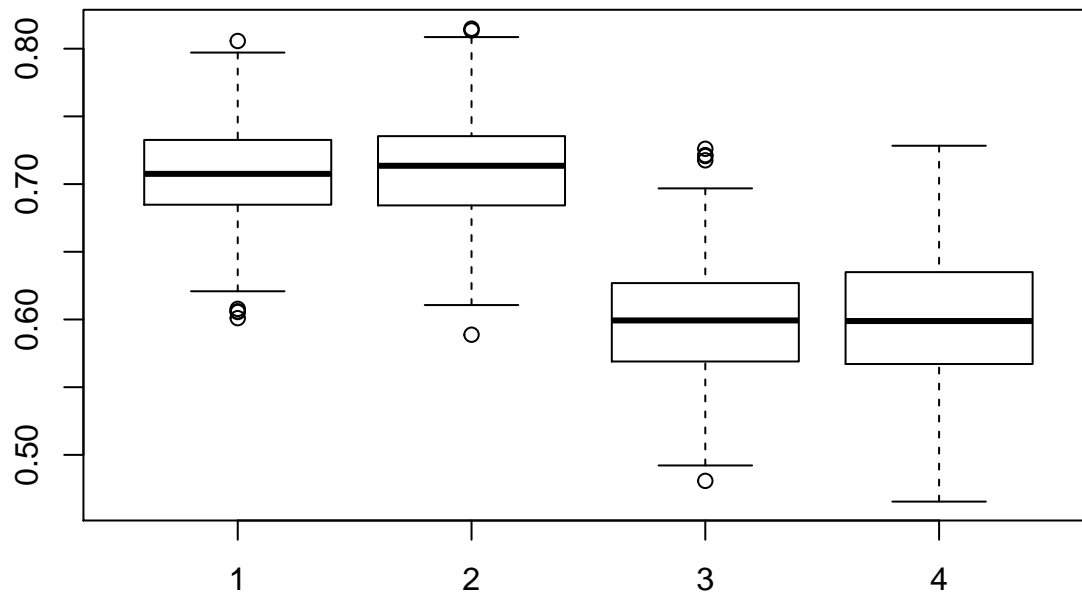
```
library (vegan)
data (BCI)
library (BiodiversityR) # available from R version 2.15.1, not older!
data (BCI.env)
BCI.soil <- read.delim ('http://www.davidzeleny.net/anadat-r/lib/exe/fetch.php?media=data:bci.soil.txt')
###
BCI.hab <- read.table("http://www.kharms.biology.lsu.edu/TORUS_Habitats.txt",
  sep="\t", header=TRUE)
```

3.2 Binomial

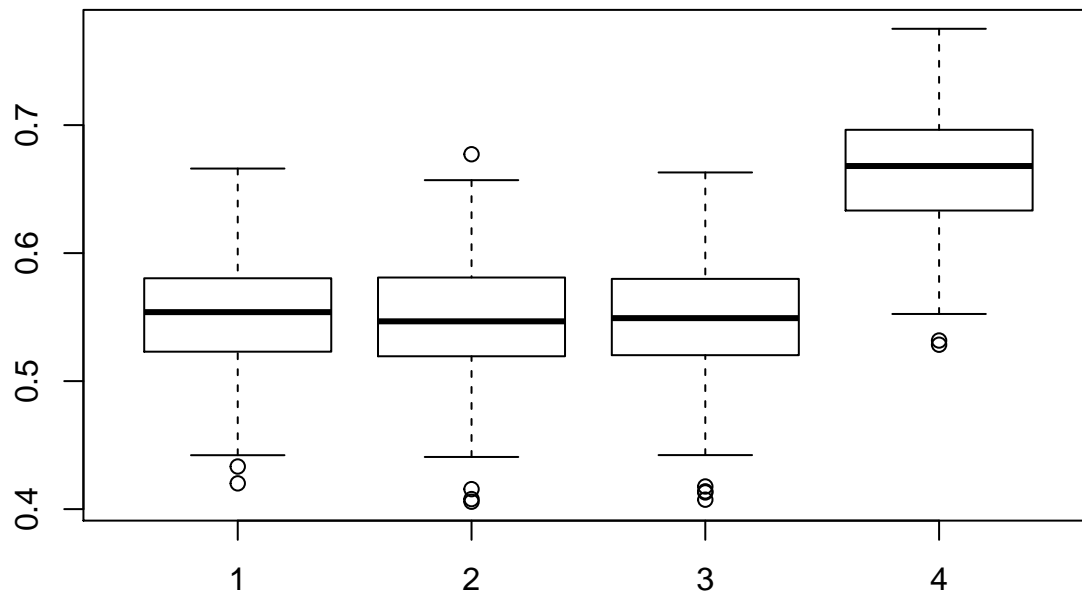
```
set.seed(1234)
n <- 1000
x0 <- sample(1:4, n, TRUE)
x1 <- ifelse(x0 %in% 1:2, 1, 0)
x2 <- rnorm(n, 0.5, 1)
table(x0,x1)
```

```
##      x1
## x0    0    1
## 1     0 240
## 2     0 242
## 3    260    0
## 4    258    0
```

```
p1 <- plogis(0.5 + 0.5*x1 + -0.2*x2)
boxplot(p1~x0)
```



```
Y1 <- rbinom(n, 1, p1)
p2 <- plogis(0.1 + 0.5*ifelse(x0==4,1,0) + 0.2*x2)
boxplot(p2~x0)
```



```
Y2 <- rbinom(n, 1, p2)
Y <- cbind(SPP1=Y1, SPP2=Y2)
X <- model.matrix(~x2)
Z <- allComb(x0)

summary(opticut(Y ~ x2, strata=x0, dist="binomial"))
```

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

```
## opticut(formula = Y ~ x2, strata = x0, dist = "binomial")
##
## Best supported models with logLR >= 2:
##      split assoc      I      mu0      mu1 logLR      w
## SPP1    1 2      ++ 0.4656 0.6301 0.7383 6.954 0.9222
## SPP2     4      ++ 0.4416 0.5415 0.6549 5.046 0.6378
## 3 binary splits
```

3.3 Poisson: Mite data set – high performance computing

See computing time diffs and plotting options.

```
library(vegan)
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.4-1
```

```
data(mite)
data(mite.env)
mite.env$hab <- with(mite.env, interaction(Shrub, Topo, drop=TRUE))
summary(mod0 <- opticut(as.matrix(mite) ~ SubsDens, mite.env,
  strata=mite.env$hab, dist="poisson", comb="all"))
```

```
## Multivariate opticut results, comb = all, dist = poisson
```

```
##
```

```
## Call:
```

```
## opticut(formula = as.matrix(mite) ~ SubsDens, data = mite.env,
```

```
##      strata = mite.env$hab, dist = "poisson", comb = "all")
```

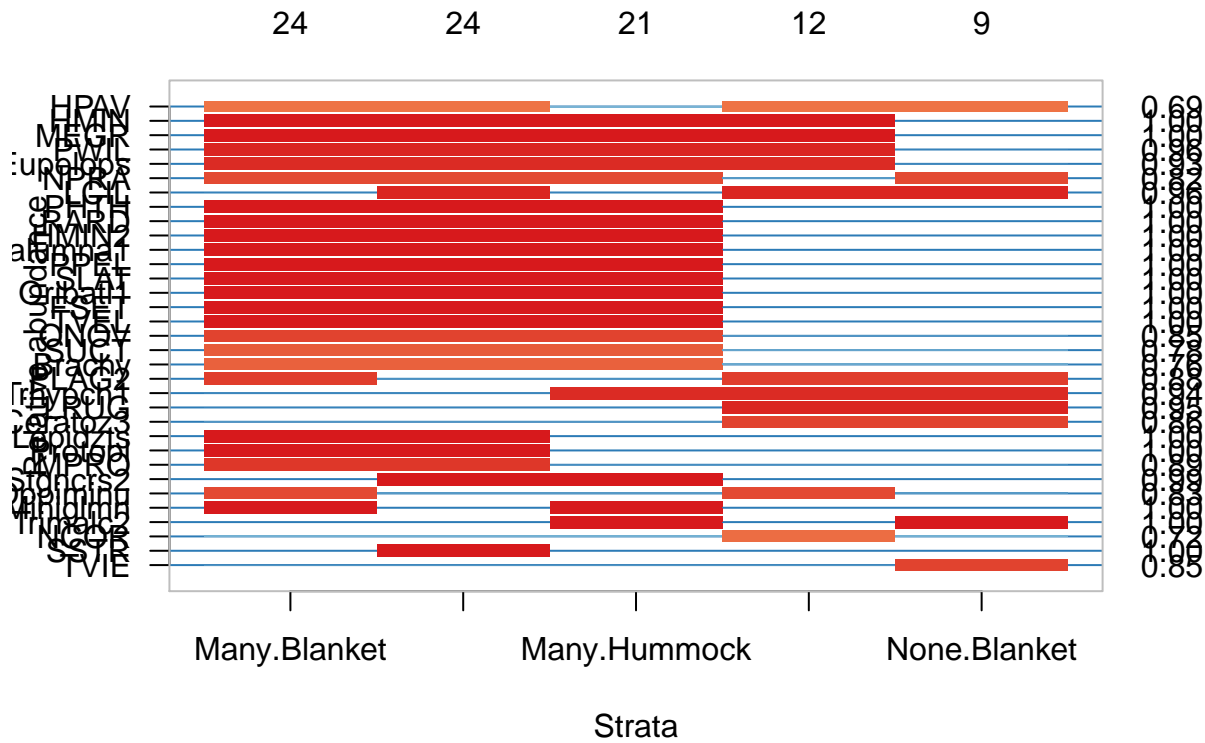
```
##
```

```
## Best supported models with logLR >= 2:
```

```
##      split assoc      I      mu0      mu1
## HPAV      Many.Hummock --- 0.6920 3.500e+01 1.493e+01
## HMIN      None.Blanket --- 0.9990 1.880e+01 4.177e-01
## MEGR      None.Blanket --- 0.9969 9.729e-01 3.818e-02
## PWIL      None.Blanket --- 0.9574 1.637e+00 2.416e-01
## Eupelops   None.Blanket -- 0.9345 1.564e+00 2.878e-01
## NPRA       Few.Blanket --- 0.8206 1.425e+00 4.474e-01
## LCIL    Many.Blanket Many.Hummock --- 0.9577 8.923e+00 1.312e+00
## PPTH      None.Blanket Few.Blanket --- 1.0000 1.463e+00 2.525e-09
## RARD      None.Blanket Few.Blanket --- 1.0000 2.546e+00 1.251e-08
## HMIN2     None.Blanket Few.Blanket --- 1.0000 1.785e+00 5.504e-09
## Galumna1  None.Blanket Few.Blanket --- 1.0000 1.523e+00 3.486e-09
## PPEL      None.Blanket Few.Blanket -- 1.0000 1.349e-01 6.501e-10
## SLAT      None.Blanket Few.Blanket --- 1.0000 7.284e-01 4.062e-09
## Oribatl1  None.Blanket Few.Blanket --- 0.9986 1.311e+00 3.441e-02
## FSET      None.Blanket Few.Blanket --- 0.9986 2.726e+00 7.211e-02
## TVEL      None.Blanket Few.Blanket --- 0.9983 1.297e+01 3.759e-01
## ONOV      None.Blanket Few.Blanket --- 0.8508 4.183e+01 1.188e+01
```

## SUCT	None.Blanket	Few.Blanket	---	0.7799	2.901e+01	1.020e+01
## Brachy	None.Blanket	Few.Blanket	---	0.7561	1.689e+01	6.294e+00
## PLAG2	Few.Hummock	Many.Hummock	---	0.8774	1.683e+01	4.300e+00
## Trhypch1	Many.Blanket	Few.Hummock	---	0.9364	7.493e-01	1.358e-01
## LRUG	None.Blanket	Few.Blanket	+++	0.9458	8.516e+00	5.100e+01
## Ceratoz3	None.Blanket	Few.Blanket	+++	0.8555	7.098e+00	2.543e+01
## Lepidzts	Many.Blanket	Few.Hummock	+++	1.0000	2.333e-09	8.547e-01
## Protopl	Many.Blanket	Few.Hummock	+++	0.9968	7.695e-01	1.916e+01
## MPRO	Many.Blanket	Few.Hummock	++	0.8944	2.034e-01	8.614e-01
## Stgnrcs2	Few.Hummock	Many.Hummock	+++	0.9873	5.786e-03	7.230e-02
## Oppiminu	Few.Blanket	Many.Blanket	+++	0.8348	2.243e+00	7.475e+00
## Miniglmn	Many.Blanket	Many.Hummock	+++	0.9975	2.716e-02	7.745e-01
## Trimalc2	None.Blanket	Many.Hummock	+++	0.9997	6.359e-02	4.929e+00
## NCOR		Few.Blanket	++	0.7181	4.238e+00	1.046e+01
## SSTR		Few.Hummock	+++	0.9971	1.561e-03	4.080e-02
## TVIE		None.Blanket	+++	0.8486	2.320e+00	8.106e+00
##	logLR	w				
## HPAV	26.281	0.9944				
## HMIN	99.969	1.0000				
## MEGR	35.465	1.0000				
## PWIL	12.421	0.7564				
## Eupelops	6.536	0.3863				
## NPRA	8.523	0.8540				
## LCIL	540.701	1.0000				
## PHTH	56.766	1.0000				
## RARD	54.142	1.0000				
## HMIN2	87.196	1.0000				
## Galumna1	42.750	1.0000				
## PPEL	7.620	0.9359				
## SLAT	17.854	0.9399				
## Oribatl1	69.712	1.0000				
## FSET	69.026	1.0000				
## TVEL	331.735	1.0000				
## ONOV	189.632	1.0000				
## SUCT	136.668	1.0000				
## Brachy	63.635	1.0000				
## PLAG2	9.162	0.9210				
## Trhypch1	35.539	0.9998				
## LRUG	218.994	1.0000				
## Ceratoz3	16.217	0.9392				
## Lepidzts	13.044	0.9912				
## Protopl	22.746	0.9936				
## MPRO	2.539	0.2639				
## Stgnrcs2	29.557	0.9147				
## Oppiminu	12.310	0.9245				
## Miniglmn	12.918	0.9445				
## Trimalc2	83.215	1.0000				
## NCOR	6.241	0.4468				
## SSTR	23.076	0.9724				
## TVIE	11.050	0.9240				
##	15 binary splits					
##	2 species not shown					

```
plot(mod0)
```



```
system.time(aa <- opticut(as.matrix(mite) ~ 1, strata=mite.env$hab, dist="poisson", comb="rank"))
```

```
## user system elapsed
## 0.632 0.015 0.650
```

```
system.time(bb <- opticut(as.matrix(mite) ~ 1, strata=mite.env$hab, dist="poisson", comb="all"))
```

```
## user system elapsed
## 1.480 0.042 1.531
```

```
## sequential
```

```
system.time(opticut(as.matrix(mite) ~ 1, strata=mite.env$hab, dist="poisson"))
```

```
## user system elapsed
## 0.584 0.015 0.601
```

```
## parallel -- compare system times
```

```
library(parallel)
```

```
cl <- makeCluster(3)
```

```
system.time(opticut(as.matrix(mite) ~ 1, strata=mite.env$hab, dist="poisson", cl=cl))
```

```
## user system elapsed
## 0.008 0.001 1.645
```



```
stopCluster(cl)
## forking -- will not work on Windows
if (!.Platform$OS.type == "windows")
system.time(opticut(as.matrix(mite) ~ 1, strata=mite.env$hab, dist="poisson", cl=3))

##      user  system elapsed
##    0.556   0.104   0.369
```

3.4 Percentages

3.4.1 Dune data, cover type data as ordinal

See <http://www.davidzeleny.net/anadat-r/doku.php/en:data:dune>

```
library(vegan)
data(dune)
data(dune.env)
dune.env$manure <- as.integer(dune.env$Manure) - 1
dune.env$moisture <- as.integer(dune.env$Moisture) - 1

oc <- ocoptions(collapse="+", cut=-Inf)

## ordinal regr
## (when nlevels() < 3 use logistic regression instead !!!)
#Dune <- as.matrix(dune)
#Dune <- Dune[,apply(Dune, 2, function(z) length(unique(z)))>2]
#x1 <- opticut(Dune ~ 1, dune.env, strata=Management, dist="ordered")
#summary(x1)

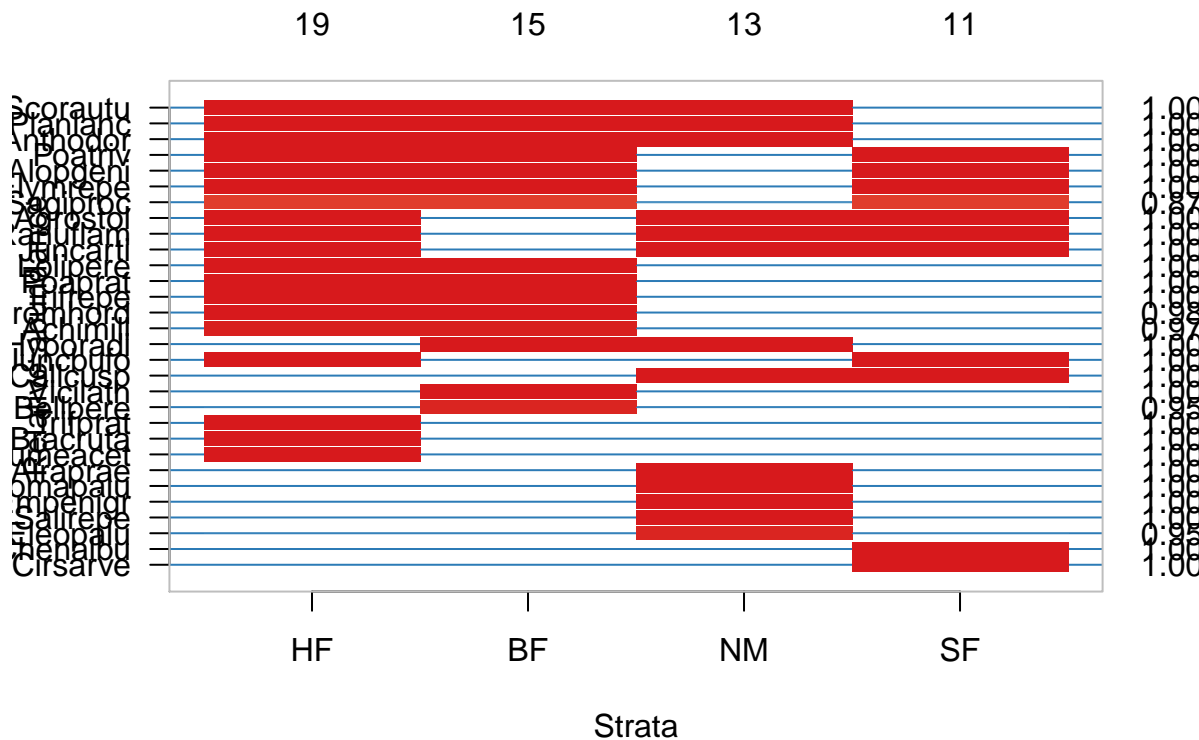
#plot(x1, mar=c(5,5,3,3))

## Binarizing data
Dune01 <- ifelse(as.matrix(dune)>0,1,0)
x2 <- opticut(Dune01 ~ 1, strata=dune.env$Management, dist="binomial")
summary(x2)

## Multivariate opticut results, comb = rank, dist = binomial
##
## Call:
## opticut(formula = Dune01 ~ 1, strata = dune.env$Management, dist = "binomial")
##
## Best supported models with logLR >= -Inf:
##      split assoc      I      mu0      mu1 logLR      w
## Scorautu BF+HF+NM    ++ 1.0000 6.667e-01 1.0000 2.6826 0.7441
## Planlanc BF+HF+NM    ++ 1.0000 8.647e-09 0.5000 3.2449 0.6890
## Anthodor BF+HF+NM    ++ 1.0000 8.647e-09 0.4286 2.6566 0.6314
## Poatriv  BF+HF+SF   +++ 1.0000 3.181e-09 0.9286 9.3465 0.7609
## Alop geni BF+HF+SF   ++ 1.0000 8.647e-09 0.5714 3.8995 0.4732
## Elymrepe BF+HF+SF   ++ 1.0000 8.647e-09 0.4286 2.6566 0.6812
## Sagipro c BF+HF+SF   + 0.8672 1.667e-01 0.4286 0.6849 0.3726
## Agrostol HF+NM+SF   ++ 1.0000 8.647e-09 0.5882 2.3455 0.4403
## Ranuflam HF+NM+SF   + 1.0000 8.647e-09 0.3529 1.1801 0.3916
```

```
## Juncarti HF+NM+SF + 1.0000 8.647e-09 0.2941 0.9481 0.3977
## Lolipere BF+HF ++ 1.0000 3.333e-01 1.0000 5.8221 0.8987
## Poaprat BF+HF ++ 1.0000 5.000e-01 1.0000 3.8995 0.7383
## Trifrepe BF+HF ++ 1.0000 6.667e-01 1.0000 2.3699 0.6700
## Bromhord BF+HF ++ 0.9836 8.333e-02 0.5000 2.2595 0.4345
## Achimill BF+HF ++ 0.9716 1.667e-01 0.6250 2.2497 0.6992
## Hyporadi BF+NM ++ 1.0000 1.170e-09 0.3333 2.7256 0.7635
## Juncbufo HF+SF ++ 1.0000 3.181e-09 0.3636 2.7977 0.7960
## Callcusp NM+SF + 1.0000 3.181e-09 0.2500 1.7062 0.5054
## Vicilath BF ++ 0.9980 5.882e-02 0.6667 2.7414 0.4650
## Bellpere BF + 0.9538 2.353e-01 0.6667 1.0326 0.4286
## Trifprat HF ++ 1.0000 1.170e-09 0.6000 5.0891 0.8579
## Bracruta HF + 1.0000 6.667e-01 1.0000 1.6990 0.6935
## Rumeacet HF ++ 0.9994 6.667e-02 0.8000 5.0707 0.7703
## Airaprae NM ++ 1.0000 1.170e-09 0.3333 2.6826 0.7175
## Comapalu NM ++ 1.0000 1.170e-09 0.3333 2.6826 0.7516
## Empenigr NM + 1.0000 4.305e-10 0.1667 1.2669 0.4870
## Salirepe NM ++ 1.0000 1.170e-09 0.5000 4.2953 0.9007
## Eleopalu NM + 0.9459 1.429e-01 0.5000 1.3462 0.4330
## Chenalbu SF + 1.0000 4.305e-10 0.1667 1.2669 0.5523
## Cirsarve SF + 1.0000 4.305e-10 0.1667 1.2669 0.5387
## 3 binary splits
```

```
plot(x2)
```



```
x3 <- opticut(Dune01 ~ manure + moisture, dune.env, strata=dune.env$Management, dist="binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

[illegible]

[illegible]


```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

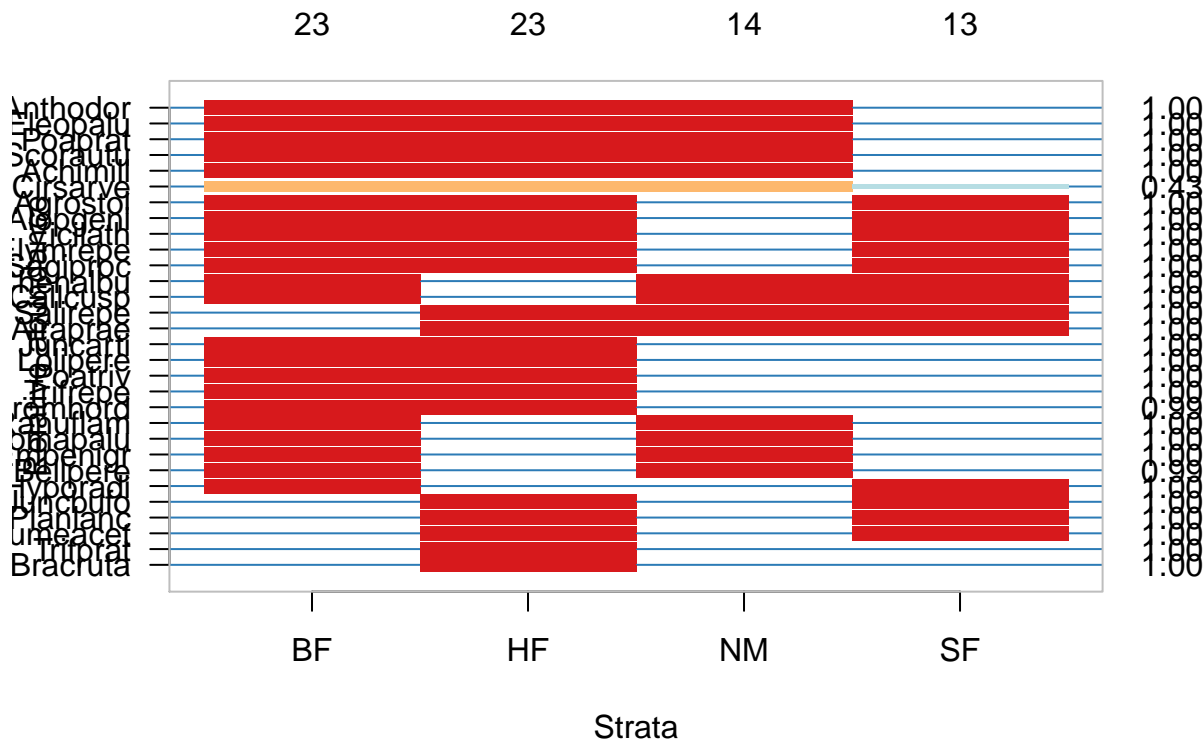
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(x3)
```

```
## Multivariate opticut results, comb = rank, dist = binomial
##
## Call:
## opticut(formula = Dune01 ~ manure + moisture, data = dune.env,
##         strata = dune.env$Management, dist = "binomial")
##
## Best supported models with logLR >= -Inf:
##           split assoc      I      mu0      mu1      logLR      w
## Anthodor BF+HF+NM      + 1.0000 3.690e-09 5.376e-01 1.629e+00 0.3705
## Eleopalu BF+HF+NM      + 1.0000 2.220e-16 2.220e-16 5.232e-01 0.4576
## Poaprat  BF+HF+NM     ++ 1.0000 2.220e-16 1.000e+00 3.253e+00 0.7463
## Scorautu BF+HF+NM      + 1.0000 9.193e-01 1.000e+00 6.534e-01 0.3693
## Achimill BF+HF+NM      + 0.9987 3.591e-02 5.923e-01 9.259e-01 0.5420
## Cirsarve BF+HF+NM      + 0.4266 2.220e-16 2.220e-16 3.386e-11 0.3333
## Agrostol BF+HF+SF      + 1.0000 2.220e-16 2.220e-16 1.341e+00 0.3333
## Alop geni BF+HF+SF     ++ 1.0000 6.724e-13 2.381e-02 3.298e+00 0.6809
## Vicilath BF+HF+SF     ++ 1.0000 1.000e+00 1.000e+00 3.998e+00 0.4954
## Elymrepe BF+HF+SF      + 1.0000 1.186e-08 2.227e-01 3.434e-01 0.4050
## Sagipro  BF+HF+SF      + 0.9993 1.261e-02 4.009e-01 1.782e+00 0.7077
## Chenalbu BF+NM+SF      + 1.0000 2.220e-16 2.220e-16 5.232e-01 0.3857
## Callcusp BF+NM+SF      + 1.0000 2.220e-16 2.220e-16 5.232e-01 0.4576
## Salirepe HF+NM+SF      + 0.9993 3.047e-02 6.221e-01 7.491e-10 0.3333
## Airaprae HF+NM+SF      + 0.9991 1.522e-02 4.159e-01 5.392e-10 0.3333
## Juncarti  BF+HF       ++ 1.0000 2.220e-16 2.220e-16 2.660e+00 0.3857
## Lolipere  BF+HF       ++ 1.0000 1.000e+00 1.000e+00 5.209e+00 0.4966
## Poatriv   BF+HF       + 1.0000 2.220e-16 2.220e-16 1.539e-09 0.3333
## Trifrepe  BF+HF       ++ 1.0000 5.906e-01 1.000e+00 2.284e+00 0.5479
## Bromhord  BF+HF       + 0.9926 2.178e-02 2.672e-01 1.430e+00 0.4426
## Ranuflam  BF+NM       + 1.0000 2.220e-16 2.220e-16 -4.704e-09 0.3333
## Comapalu  BF+NM       + 1.0000 2.220e-16 1.837e-13 2.249e-09 0.3333
## Empenigr  BF+NM       + 1.0000 2.220e-16 2.935e-13 2.062e-09 0.3333
## Bellpere  BF+NM       + 0.9931 5.142e-02 4.786e-01 9.237e-01 0.4039
## Hyporadi  BF+SF       + 1.0000 5.289e-01 1.000e+00 7.921e-01 0.3811
```

```
## Juncbufo    HF+SF    ++ 1.0000 6.852e-10 9.064e-01 3.777e+00 0.8265
## Planlanc    HF+SF    + 1.0000 1.000e+00 1.000e+00 1.294e+00 0.3611
## Rumeacet    HF+SF   +++ 1.0000 4.887e-11 1.000e+00 1.033e+01 0.9961
## Trifprat     HF    ++ 1.0000 5.073e-12 1.000e+00 4.571e+00 0.4655
## Bracruta     HF     + 1.0000 6.414e-01 1.000e+00 1.818e+00 0.5193
## 3 binary splits
```

```
plot(x3)
```



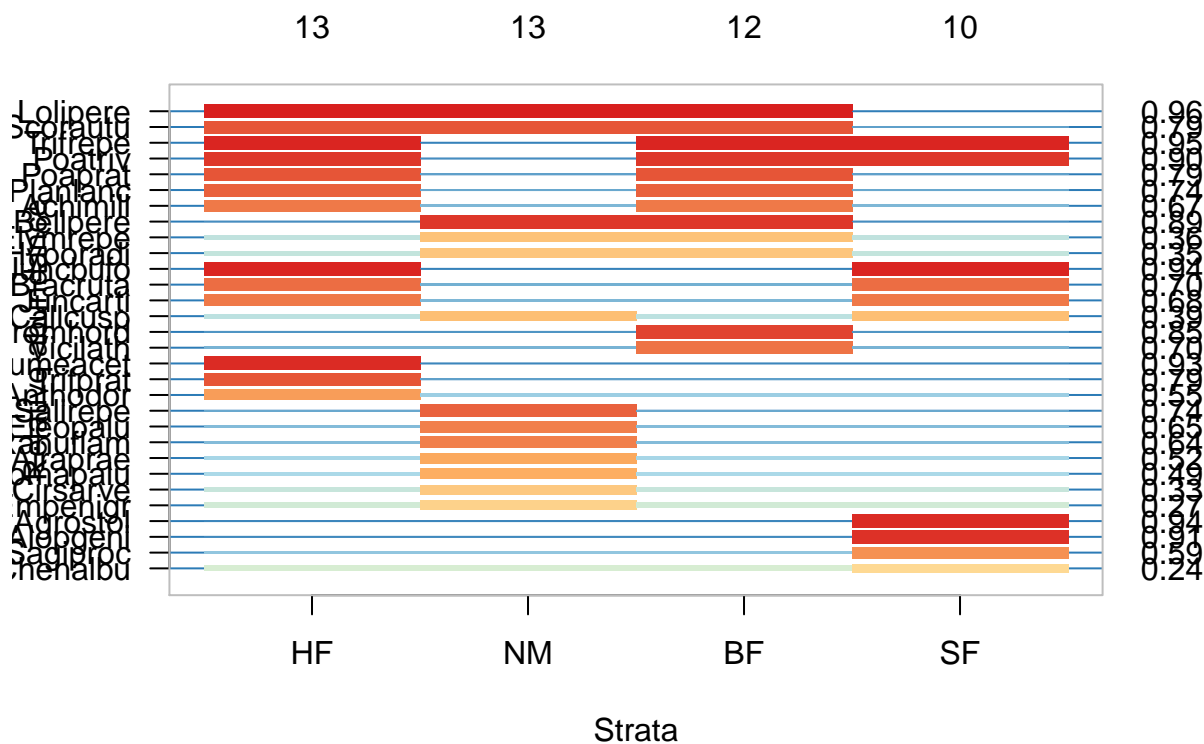
```
## Beta regression
Dune2 <- as.matrix(dune+0.5) / 10
x4 <- opticut(Dune2 ~ 1, strata=dune.env$Management, dist="beta")
summary(x4)
```

```
## Multivariate opticut results, comb = rank, dist = beta
##
## Call:
## opticut(formula = Dune2 ~ 1, strata = dune.env$Management, dist = "beta")
##
## Best supported models with logLR >= -Inf:
##      split assoc      I      mu0      mu1 logLR      w
## Scorautu BF+HF+NM    ++ 0.7794 0.17817 0.38107 5.0745 0.8647
## Anthodor BF+HF+NM     + 0.5760 0.10484 0.18420 1.1481 0.4203
## Poatrive BF+HF+SF    ++ 0.9620 0.12172 0.49912 7.3010 0.8074
## Poaprat  BF+HF+SF    ++ 0.8433 0.13521 0.34908 4.3908 0.4868
## Elymrepe BF+HF+SF     + 0.5986 0.12205 0.21717 1.1230 0.4244
## Lolipere  BF+HF      ++ 0.8840 0.21057 0.51807 4.1041 0.5739
## Trifrepe  BF+HF      ++ 0.7511 0.20483 0.40589 3.3954 0.6065
## Planlanc  BF+HF      ++ 0.7389 0.12810 0.27492 2.4674 0.5496
```



```
##
## Call:
## opticut(formula = Dune2 ~ manure, data = dune.env, strata = dune.env$Management,
##         dist = "beta")
##
## Best supported models with logLR >= -Inf:
##      split assoc      I      mu0      mu1      logLR      w
## Lolipere BF+HF+NM    ++ 0.9628 0.02503 0.15721 4.52994 0.6104
## Scorausu BF+HF+NM    ++ 0.7883 0.17329 0.37858 2.78625 0.6581
## Trifrepe BF+HF+SF    ++ 0.9473 0.21532 0.62509 3.90047 0.4742
## Poatriv  BF+HF+SF    ++ 0.8963 0.11822 0.36436 2.00207 0.6594
## Poaprat   BF+HF      ++ 0.7948 0.11196 0.27161 5.10107 0.6029
## Planlanc  BF+HF      ++ 0.7433 0.14503 0.30657 2.53058 0.4590
## Achimill  BF+HF      ++ 0.6711 0.09329 0.18824 2.58564 0.4552
## Bellpere  BF+NM      ++ 0.8885 0.02647 0.10061 2.12610 0.5576
## Elymrepe  BF+NM      + 0.3640 0.08561 0.12057 0.07552 0.3407
## Hyporadi  BF+NM      + 0.3544 0.09572 0.13293 0.21056 0.3572
## Juncbufo  HF+SF      ++ 0.9446 0.09242 0.37623 3.10402 0.8621
## Bracruta  HF+SF      + 0.7044 0.31627 0.52623 0.92747 0.4554
## Juncarti  HF+SF      + 0.6770 0.15362 0.29256 0.70062 0.3922
## Callcusp  NM+SF      + 0.3886 0.09554 0.13733 0.63669 0.3964
## Bromhord  BF        ++ 0.8499 0.07911 0.23168 3.00961 0.7403
## Vicilath  BF        ++ 0.6987 0.06912 0.14988 4.62302 0.8194
## Rumeacet  HF        ++ 0.9302 0.09680 0.36044 6.26423 0.7311
## Trifprat  HF        ++ 0.7931 0.07088 0.18342 3.76058 0.7219
## Anthodor  HF        + 0.5504 0.16814 0.27289 0.87929 0.3494
## Salirepe  NM        + 0.7439 0.08105 0.18708 1.19959 0.5531
## Eleopalu  NM        + 0.6457 0.13865 0.25755 0.50249 0.3632
## Ranuflam  NM        + 0.6416 0.08310 0.16247 0.83466 0.3694
## Airaprae  NM        + 0.5190 0.06399 0.10832 0.66716 0.4815
## Comapalu  NM        + 0.4882 0.05980 0.09784 0.73272 0.4966
## Cirsarve  NM        + 0.3253 0.04022 0.05547 0.38506 0.3667
## Empenigr  NM        + 0.2717 0.05575 0.07237 0.29145 0.3942
## Agrostol  SF        ++ 0.9356 0.23871 0.63227 2.70445 0.7902
## Alop geni  SF        ++ 0.9124 0.15916 0.46926 2.10605 0.7632
## Sagiproc  SF        + 0.5929 0.14226 0.24704 0.56757 0.4104
## Chenalbu  SF        + 0.2400 0.05312 0.06686 0.71117 0.4983
## 3 binary splits
```

```
plot(x5)
```



```
xx <- data.frame(#ord0=summary(x1)$summary$split,
  bin0=summary(x2)$summary$split,
  binx=summary(x3)$summary$split,
  bet0=summary(x4)$summary$split,
  betx=summary(x5)$summary$split)
rownames(xx) <- rownames(summary(x2)$summary)
xx
```

##	bin0	binx	bet0	betx
## Achimill	BF+HF	BF+HF+NM	BF+HF	BF+HF
## Agrostol	HF+NM+SF	BF+HF+SF	SF	SF
## Airaprae	NM	HF+NM+SF	NM	NM
## Alopogeni	BF+HF+SF	BF+HF+SF	SF	SF
## Anthodor	BF+HF+NM	BF+HF+NM	BF+HF+NM	HF
## Bellpere	BF	BF+NM	BF	BF+NM
## Bromhord	BF+HF	BF+HF	BF	BF
## Chenalbu	SF	BF+NM+SF	SF	SF
## Cirsarve	SF	BF+HF+NM	SF	NM
## Comapalu	NM	BF+NM	NM	NM
## Eleopalu	NM	BF+HF+NM	NM	NM
## Elymrepe	BF+HF+SF	BF+HF+SF	BF+HF+SF	BF+NM
## Empenigr	NM	BF+NM	NM	NM
## Hyporadi	BF+NM	BF+SF	BF+NM	BF+NM
## Juncarti	HF+NM+SF	BF+HF	HF+NM	HF+SF
## Juncbufo	HF+SF	HF+SF	HF+SF	HF+SF
## Lolipere	BF+HF	BF+HF	BF+HF	BF+HF+NM
## Planlanc	BF+HF+NM	HF+SF	BF+HF	BF+HF
## Poaprat	BF+HF	BF+HF+NM	BF+HF+SF	BF+HF
## Poatriv	BF+HF+SF	BF+HF	BF+HF+SF	BF+HF+SF
## Ranuflam	HF+NM+SF	BF+NM	NM	NM

```
## Rumeacet      HF      HF+SF      HF      HF
## Sagiproc BF+HF+SF BF+HF+SF      SF      SF
## Salirepe      NM HF+NM+SF      NM      NM
## Scorausu BF+HF+NM BF+HF+NM BF+HF+NM BF+HF+NM
## Trifprat      HF      HF      HF      HF
## Trifrepe      BF+HF      BF+HF      BF+HF BF+HF+SF
## Vicilath      BF BF+HF+SF      BF      BF
## Bracruta      HF      HF      HF+NM      HF+SF
## Callcusp      NM+SF BF+NM+SF      NM      NM+SF
```

```
ooptions(oc)
```

3.4.2 Varespec data (% cover)

```
library(vegan)
data(varespec)
data(varechem)
y <- as.matrix(varespec / 100)
range(y[y>0])
```

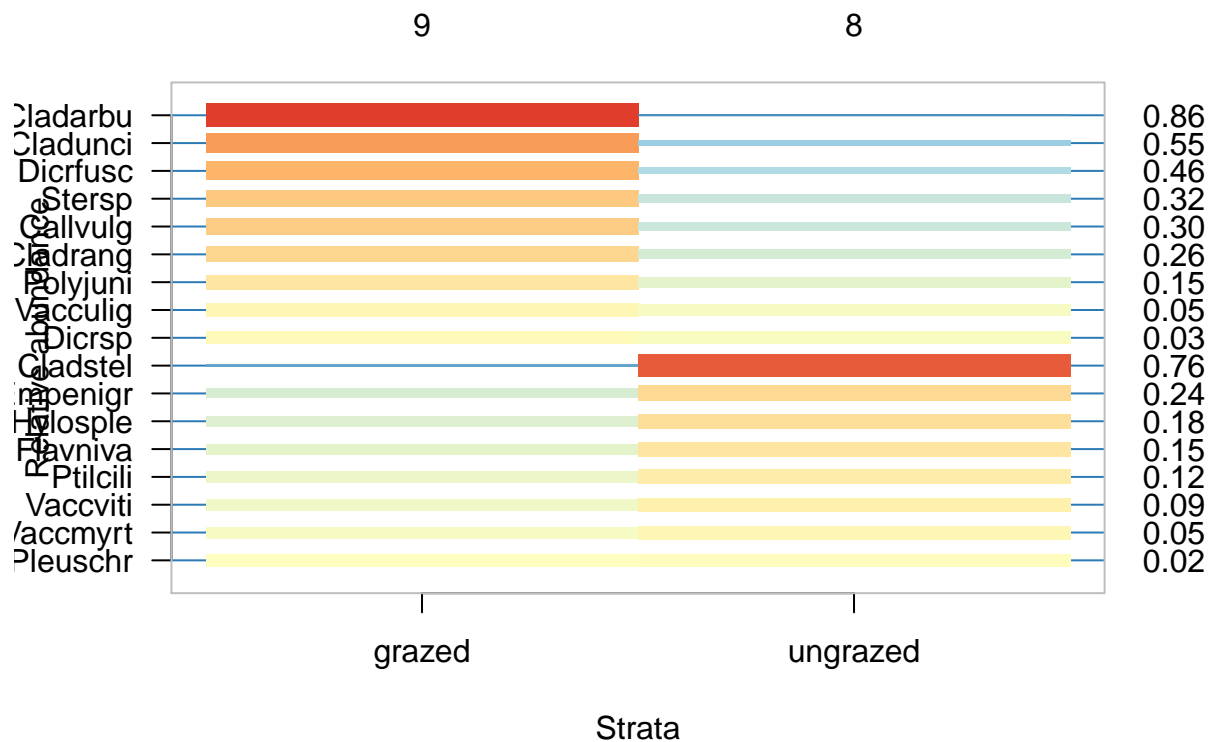
```
## [1] 0.0002 0.8430
```

```
y[y <= 0] <- 0.0001
y <- y[,apply(y, 2, max) > 0.05]
varechem$grazing <- as.factor(ifelse(rownames(varechem) %in% c(5,6,7,8,13,14,15,16,
  18,19,20,22,23,24,26), "grazed", "ungrazed"))

x <- opticut(y ~ 1, varechem, strata=grazing, dist="beta")
summary(x)
```

```
## Multivariate opticut results, comb = rank, dist = beta
##
## Call:
## opticut(formula = y ~ 1, data = varechem, strata = grazing, dist = "beta")
##
## Best supported models with logLR >= 2:
##           split assoc      I      mu0      mu1 logLR w
## Cladarbu   grazed    ++ 0.8616 0.04831 0.157 6.213 1
## Cladstel  ungrazed    ++ 0.7619 0.13055 0.290 2.091 1
## 1 binary split
## 15 species not shown
```

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



Implement ZI-Beta (quite unreliable for such small data set)

```
zi_beta_fun <- function(Y, X, linkinv, ...) {
  kx <- ncol(X)
  id1 <- Y > 0
  id0 <- !id1
  nll_ZIB_ML <- function(parms) {
    mu <- plogis(X %*% parms[1:kx])
    gamma <- exp(parms[kx + 1]) # precision
    phi <- plogis(parms[kx+2])
    alpha <- mu * gamma
    beta <- (1 - mu) * gamma
    loglik0 <- log(phi)
    loglik1 <- log(1 - phi) + suppressWarnings(dbeta(Y,
      alpha, beta, log = TRUE))
    loglik <- sum(loglik0[id0]) + sum(loglik1[id1])
    if (!is.finite(loglik) || is.na(loglik))
      loglik <- -.Machine$double.xmax^(1/3)
    -loglik
  }
  Yv <- Y
  Yv[Y <= 0.001] <- 0.001
  ini <- c(coef(betareg::betareg(Yv ~ .-1, data=X)), zi=-5)
  X <- as.matrix(X)
  res <- optim(ini, nll_ZIB_ML, ...)
  list(coef=res$par,
    logLik=-res$value,
    linkinv=binomial("logit")$linkinv)
}
y <- as.matrix(varespec / 100)
```

3.4.3 Stratigraphy example

```
## Multivariate opticut results, comb = rank, dist = beta
##
## Call:
## opticut(formula = ab ~ 1, strata = z, dist = "beta", comb = "rank")
##
## Best supported models with logLR >= 2:
##
```

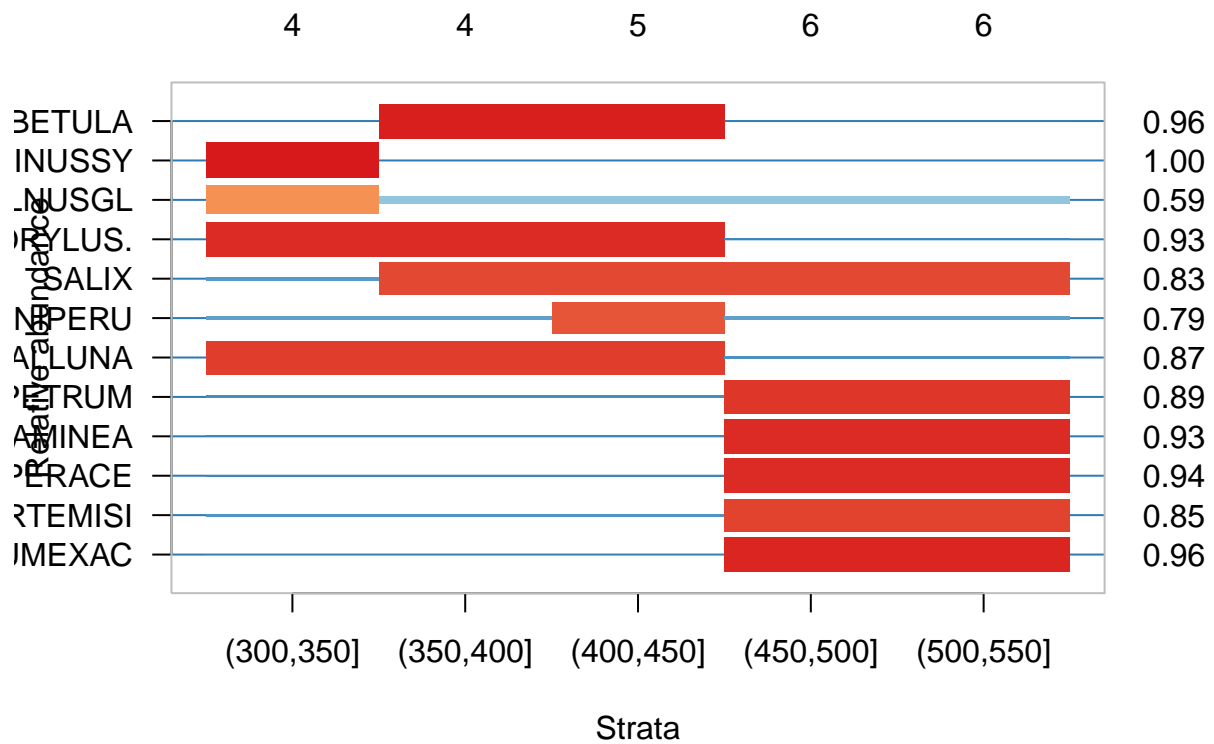
	split	assoc	I	mu0
## SALIX (350,400] (400,450] (450,500] (500,550]	+++	0.8306	0.014015	
## CORYLUS. (300,350] (350,400] (400,450]	+++	0.9341	0.019366	
## CALLUNA (300,350] (350,400] (400,450]	+++	0.8682	0.005234	
## BETULA (350,400] (400,450]	+++	0.9627	0.153662	
## RUMEXAC (450,500] (500,550]	+++	0.9554	0.015648	
## CYPERACE (450,500] (500,550]	+++	0.9353	0.015826	
## GRAMINEA (450,500] (500,550]	+++	0.9299	0.028003	
## EMPETRUM (450,500] (500,550]	+++	0.8857	0.020863	
## ARTEMISI (450,500] (500,550]	++	0.8515	0.046111	
## PINUSSY (300,350]	+++	0.9960	0.083196	
## ALNUSGL (300,350]	++	0.5936	0.002089	
## JUNIPERU (400,450]	++	0.7935	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

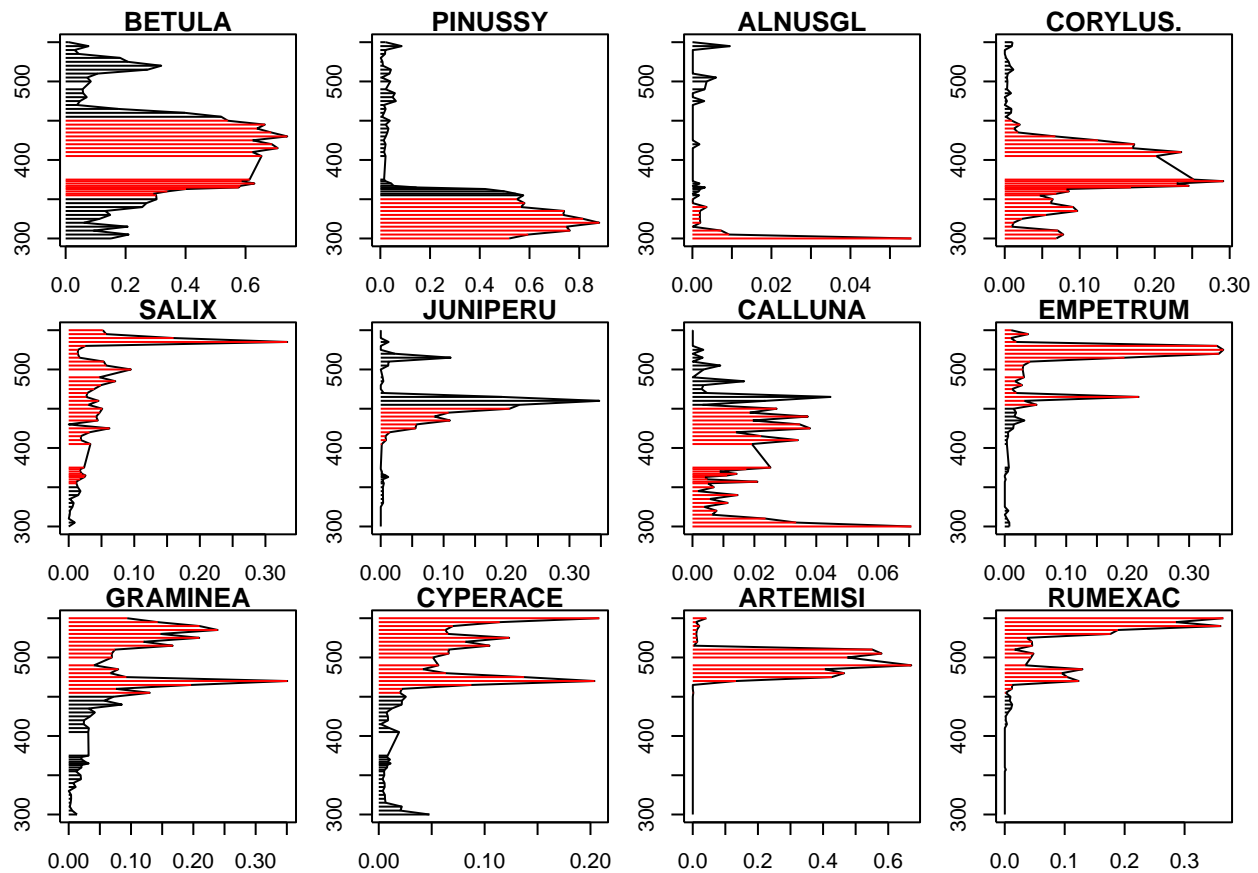
```
## 4 binary splits
## 1 species not shown
```

```
plot(a, sort=FALSE)
```



```
bp <- bestpart(a)

op <- 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])
}
```



```
par(op)
```

3.5 Presence-only data

Describe RSF/RSPF differences especially related to covariates.

```
## presence-only data
## single species model only:
## because the used distr is different for
## each species by definition.

library(ResourceSelection)

## ResourceSelection 0.2-6    2016-02-15

## settings
n.used <- 1000
m <- 10
n <- n.used * m
set.seed(1234)
x <- data.frame(x0=as.factor(sample(1:3, n, replace=TRUE)),
  x1=rnorm(n), x2=runif(n))
cfs <- c(1, -0.5, 0.1, -1, 0.5)
## Logistic RSPF model
```



```
dd <- simulateUsedAvail(x, cfs, n.used, m, link="logit")

Y <- dd$status
X <- model.matrix(~ x1 + x2, dd)
Z <- allComb(as.integer(dd$x0))

mod1 <- opticut(Y ~ x1 + x2, dd, strata=x0, dist="rsf", m=0, B=0)
mod1$species
```

```
## $Species.1
## Univariate opticut results, comb = rank, dist = rsf
## I = 0.2344; w = 0.59; H = 0.5162; logL_null = -9184
##
## Best supported models with logLR >= 2:
##      assoc      I mu0    mu1 logLR    w
## 3      ++ 0.2344    1 1.270 6.649 0.59
## 1 3      ++ 0.2397    1 1.277 6.285 0.41
## 2 binary splits
```

```
mod2 <- opticut(Y ~ x1 + x2, dd, strata=x0, dist="rspf", m=0, B=0)
mod2$species
```

```
## $Species.1
## Univariate opticut results, comb = rank, dist = rspf
## I = 0.5854; w = 0.73; H = 0.6058; logL_null = -9169
##
## Best supported model with logLR >= 2:
##      assoc      I    mu0    mu1 logLR    w
## 3      ++ 0.5854 0.7359 0.8449 2.508 0.73
## 2 binary splits (1 model not shown)
```

4 Custom distributions

The `distr` argument accepts a function, so other parametric models can be supplied which are avoided due to package dependencies.

4.1 Mixed models

Here is an example using mixed models and the package `lme4`:

```
library(lme4)
```

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

```
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)
```

```

ee <- rnorm(n/5)
g <- rep(1:5, each=n/5)
lam1 <- exp(0.5 + 0.5*x1 + -0.2*x2 + ee[g])
Y1 <- rpois(n, lam1)

X <- model.matrix(~x2)
Z <- allComb(x0)

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(Y1, X, gr=g)

```

```

## $coef
## X(Intercept)      Xx2
##    0.6880337   -0.1899153
##
## $logLik
## 'log Lik.' -345.1799 (df=3)
##
## $linkinv
## function (eta)
## pmax(exp(eta), .Machine$double.eps)
## <environment: namespace:stats>

```

```

opticut1(Y1, X, Z, dist=lmefun, gr=g)

```

```

## Univariate opticut results, comb = all, dist = lmefun
## I = 0.4736; w = 0.9985; H = 0.997; logL_null = -345.2
##
## Best supported models with logLR >= 2:
##   assoc      I  mu0  mu1  logLR      w
## 1 2    +++ 0.4736 1.480 2.476 13.582 0.9984969
## 1      ++ 0.4057 1.745 2.683  6.864 0.0012072
## 4     -- 0.3473 2.168 1.509  4.739 0.0001441
## 3     -- 0.3495 2.134 1.482  4.620 0.0001279
## 2     ++ 0.2447 1.858 2.386  2.813 0.0000210
## 7 binary splits (2 models not shown)

```

4.2 Imperfect detectability: N-mixture case

A single-visit based N-mixture is an example where detection error is estimated. Let us compare results based on naive GLM and N-mixture:

```

library(detect)

```

```

## Loading required package: Formula

```

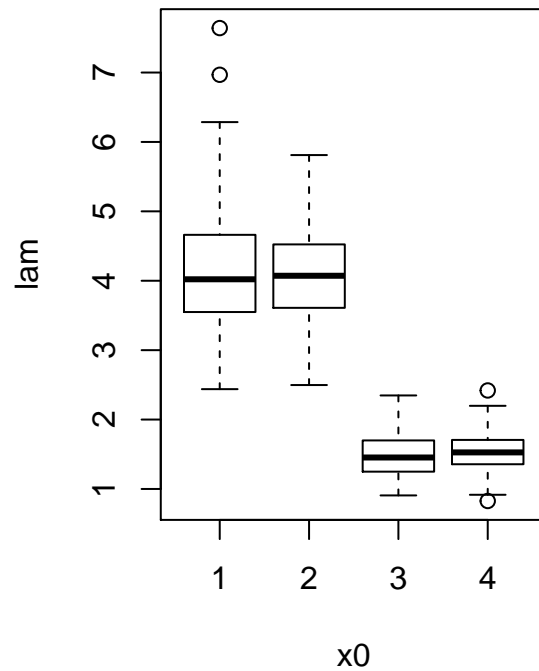
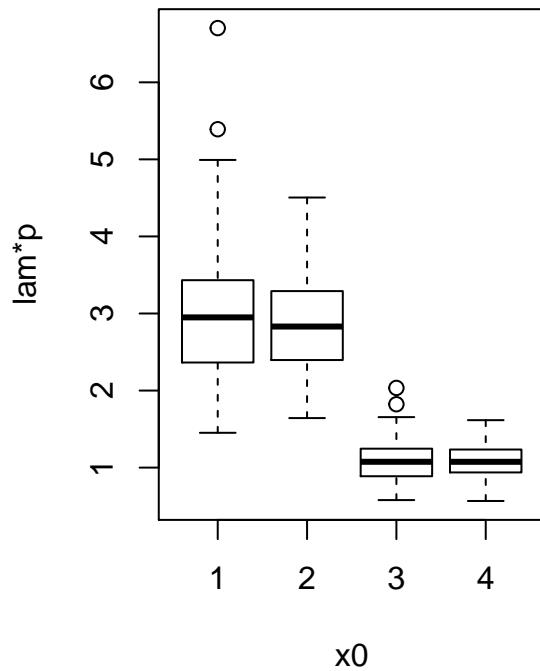
```
## Loading required package: stats4
```

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

```
set.seed(2345)
n <- 500
x0 <- sample(1:4, n, TRUE)
x1 <- ifelse(x0 %in% 1:2, 1, 0)
x2 <- rnorm(n, 0.5, 1)
x3 <- runif(n, 0, 1)
lam <- exp(0.5 + 1*x1 + -0.2*x2)
p <- plogis(2 + -2*x3)
Y <- rpois(n, lam*p)

X <- model.matrix(~x2)

op <- par(mfrow=c(1,2))
boxplot((lam*p) ~ x0, ylab="lam*p", xlab="x0")
boxplot(lam ~ x0, ylab="lam", xlab="x0")
```



```
par(op)

svfun <- function(Y, X, linkinv, ...) {
  X <- as.matrix(X)
  m <- svabu(Y ~ X-1 | x3, ...)
  list(coef=coef(m, "sta"),
       logLik=logLik(m),
       linkinv=poisson()$linkinv)
}
svfun(Y, X)
```

```
## $coef
```

```
## X(Intercept)          Xx2
##      1.6746855      -0.2458261
##
## $logLik
## 'log Lik.' -884.583 (df=5)
##
## $linkinv
## function (eta)
## pmax(exp(eta), .Machine$double.eps)
## <environment: namespace:stats>
```

```
## naive GLM
print(opticut1(Y, X, as.factor(x0), dist="poisson", cut=-Inf)
```

```
## Univariate opticut results, comb = rank, dist = poisson
## I = 0.7741; w = 1; H = 1; logL_null = -930
##
## Best supported models with logLR >= -Inf:
##      assoc      I  mu0  mu1 logLR      w
## 1 2      +++ 0.7741 1.139 3.193 115.16 1.000e+00
## 1 2 4      +++ 0.6998 1.078 2.566  52.38 5.383e-28
## 2      +++ 0.5260 1.808 3.243  38.04 3.211e-34
## 3 binary splits
```

```
## N-mixture
print(opticut1(Y, X, as.factor(x0), dist=svfun, cut=-Inf)
```

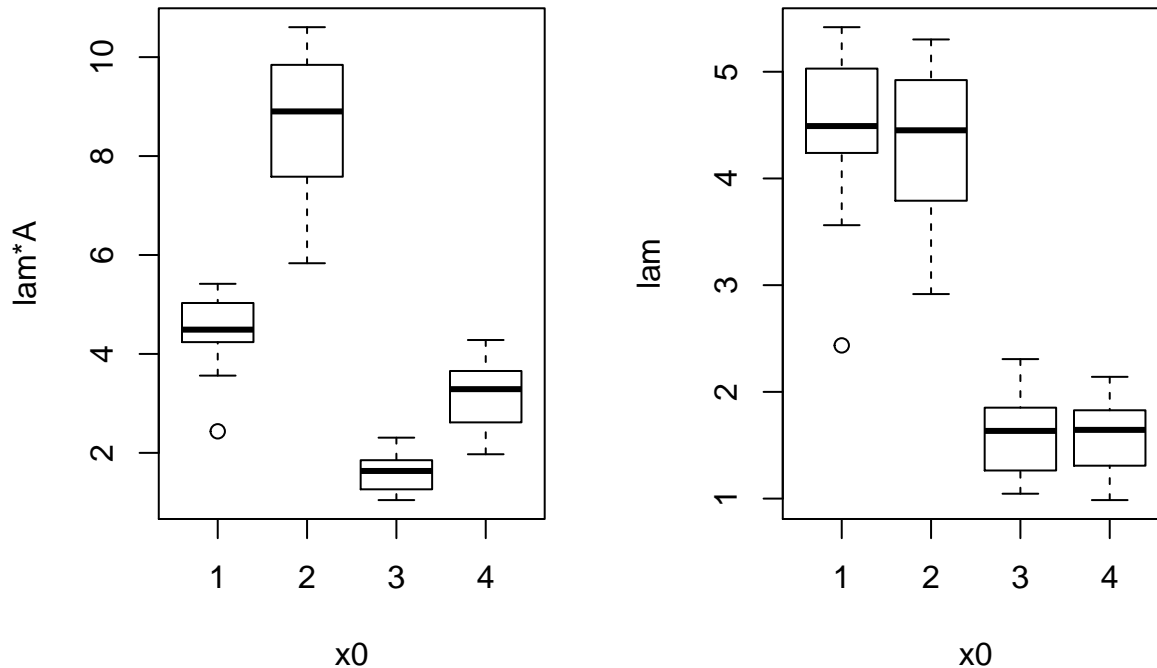
```
## Univariate opticut results, comb = rank, dist = svfun
## I = 0.7876; w = 1; H = 0.9999; logL_null = -884.6
##
## Best supported models with logLR >= -Inf:
##      assoc      I  mu0  mu1 logLR      w
## 1 2      +++ 0.7876 2.409 6.989 26.239 1.000e+00
## 1 2 3      +++ 0.7261 1.936 4.859 15.787 2.888e-05
## 2      ++ 0.4664 4.142 6.866  3.284 1.073e-10
## 3 binary splits
```

4.3 Sampling differences: using offsets

Not accounting for unequal sampling effort can be quite misleading, especially if that is related to habitat classes. This example shows how to take advantage of the other arguments passed to the ... in the `opticut` function.

```
set.seed(1234)
n <- 50
x0 <- sample(1:4, n, TRUE)
x1 <- ifelse(x0 %in% 1:2, 1, 0)
x2 <- rnorm(n, 0.5, 1)
lam <- exp(0.5 + 1*x1 + -0.2*x2)
A <- ifelse(x0 %in% c(1,3), 1, 2)
Y <- rpois(n, lam*A)
```

```
op <- par(mfrow=c(1,2))
boxplot((lam*A) ~ x0, ylab="lam*A", xlab="x0")
boxplot(lam ~ x0, ylab="lam", xlab="x0")
```



```
par(op)
```

```
## no offset: incorrect
opticut(Y ~ x2, strata=x0, dist="poisson", comb="rank")$species
```

```
## $Species.1
## Univariate opticut results, comb = rank, dist = poisson
## I = 0.8354; w = 0.9895; H = 0.9792; logL_null = -154.2
##
## Best supported models with logLR >= 2:
##      assoc      I  mu0  mu1 logLR      w
## 2      +++ 0.8354 3.028 10.110 42.01 9.895e-01
## 1 2      +++ 0.8412 2.192  7.464 37.46 1.049e-02
## 1 2 4     +++ 0.8483 1.766  6.164 24.77 3.243e-08
## 3 binary splits
```

```
## with offsets: log Area
opticut(Y ~ x2, strata=x0, dist="poisson", offset=log(A), comb="rank")$species
```

```
## $Species.1
## Univariate opticut results, comb = rank, dist = poisson
## I = 0.8153; w = 1; H = 1; logL_null = -135.7
##
## Best supported models with logLR >= 2:
##      assoc      I  mu0  mu1 logLR      w
## 1 2      +++ 0.8153 1.572 4.930 32.34 1.000e+00
```

```
## 1 2 3    +++ 0.7780 1.431 4.049 17.29 2.919e-07
## 2        +++ 0.6356 2.388 5.060 16.68 1.584e-07
## 3 binary splits
```

4.4 GAM models

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
##
```

```
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:lme4':
```

```
##
```

```
##      lmList
```

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

```
library(detect)
```

```
data(oven)
```

```
oven$veg <- factor(NA, c("agr", "open", "decid", "conif", "mix"))
```

```
oven$veg[oven$pforest < 0.5] <- "open"
```

```
oven$veg[oven$pagri > 0.5 & oven$pforest < 0.5] <- "agr"
```

```
oven$veg[oven$pforest >= 0.5] <- "mix"
```

```
oven$veg[oven$pforest >= 0.5 & oven$pdecid >= 0.8] <- "decid"
```

```
oven$veg[oven$pforest >= 0.5 & oven$pdecid < 0.2] <- "conif"
```

```
table(oven$veg, useNA="always")
```

```
##
```

```
##      agr  open decid conif  mix  <NA>
```

```
##      530    33    78    30   220    0
```

```
oven$xlat <- scale(oven$lat)
```

```
oven$xleng <- scale(oven$long)
```

```
gamfun <- function(Y, X, linkinv, Data, ...) {
```

```
  X <- as.matrix(X)
```

```
  m <- mgcv::gam(Y ~ X-1 + s(xlat) + s(xleng), Data, ...)
```

```
  list(coef=coef(m),
```

```
        logLik=logLik(m),
```

```
        linkinv=family(m)$linkinv)
```

```
}
```

```
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
```

```
## -0.306661304 -1.382449156 -0.555089095 -0.890381109  0.306380012
```

```
##      s(xlat).4      s(xlat).5      s(xlat).6      s(xlat).7      s(xlat).8
##  1.122776013  0.132117780 -0.052077672  0.036702449 -3.041886993
##      s(xlat).9      s(xlong).1      s(xlong).2      s(xlong).3      s(xlong).4
##  0.955525315 -1.090347031  1.922368963 -0.004724731 -0.428722334
##      s(xlong).5      s(xlong).6      s(xlong).7      s(xlong).8      s(xlong).9
##  0.369410432  0.241421901 -0.228281375 -0.857581486  0.995878919
##
## $logLik
## 'log Lik.' -730.8404 (df=16.34444)
##
## $linkinv
## function (eta)
## pmax(exp(eta), .Machine$double.eps)
## <environment: namespace:stats>
```

```
print(opticcut1(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 = 0.8815; w = 0.9985; H = 0.997; logL_null = -777.7
##
## Best supported models with logLR >= -Inf:
##
##          assoc      I      mu0      mu1 logLR      w
## open decid conif mix +++ 0.8815 0.1847 0.7359 46.906 9.985e-01
## decid conif mix      +++ 0.8681 0.2049 0.7712 40.373 1.451e-03
## decid mix            +++ 0.8196 0.2298 0.7298 37.182 5.973e-05
## decid                ++ 0.4631 0.3462 0.5714  6.942 4.396e-18
## 4 binary splits
```

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

```
## Multivariate opticut results, comb = rank, dist = gamfun
##
## Call:
## opticcut(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
## Species.1 open decid conif mix +++ 0.8815 0.1847 0.7359 46.91 0.9985
## 4 binary splits
```

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

```
## Multivariate opticut results, comb = rank, dist = poisson
##
## Call:
## opticcut(formula = count ~ 1, data = oven, strata = veg, dist = "poisson")
##
## Best supported model with logLR >= 2:
```

```
##                               split assoc      I      mu0      mu1 logLR      w
## Species.1 open decid conif mix    +++ 0.935 0.1868 1.019 142.7 0.9998
## 4 binary splits
```

```
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 = 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
## Species.1 open decid conif mix    +++ 0.9388 0.1421 0.4824 28.84 0.919
## 4 binary splits
```

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

```
## Multivariate opticut results, comb = rank, dist = binomial
##
## Call:
## opticut(formula = pa ~ 1, data = oven, strata = veg, dist = "binomial")
##
## Best supported model with logLR >= 2:
##                               split assoc      I      mu0      mu1 logLR      w
## Species.1 open decid conif mix    +++ 0.9629 0.1377 0.5374 82.37 0.6569
## 4 binary splits
```

5 Finding best partitions

It is useful to access the best binary partition

```
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)
lam <- exp(0.5 + 1*x1 + -0.2*x2)
Y <- rpois(n, lam)
o <- opticut(Y ~ x2, strata=x0, dist="poisson", comb="rank")
summary(o)
```

```
## Multivariate opticut results, comb = rank, dist = poisson
##
## Call:
## opticut(formula = Y ~ x2, strata = x0, dist = "poisson", comb = "rank")
```



```
##
## Best supported model with logLR >= 2:
##      split assoc      I    mu0    mu1 logLR      w
## Species.1   1 2    +++ 0.7037 1.819 4.361 12.46 0.9918
## 3 binary splits
```

```
bp <- bestpart(o)
head(bp)
```

```
## Species.1
## 1         1
## 1         1
## 3         0
## 1         1
## 2         1
## 2         1
```

The model based on the best partition can be returned as:

```
bestmodel(o, which=1)
```

```
## $Species.1
##
## Call: stats::glm(formula = Y ~ . - 1, family = Family, data = XX)
##
## Coefficients:
## `(Intercept)`          Z1          x2
##      0.5981      0.8747     -0.1246
##
## Degrees of Freedom: 50 Total (i.e. Null); 47 Residual
## Null Deviance:      180.3
## Residual Deviance: 46.98    AIC: 184.8
```

the `which` argument can be used to subset the species.

6 Uncertainty

Uncertainty in I values might be of interest. The `type` argument for the `uncertainty` method can take the following values:

- "asympt": asymptotic distribution of I , μ_0 and μ_1 based on best partition found for the input object.
- "boot": non-parametric bootstrap distribution of I , μ_0 and μ_1 based on best partition found for the input object.
- "multi": non-parametric bootstrap distribution of I , μ_0 and μ_1 based on best partition found for the bootstrap data (i.e. the model ranking is re-evaluated each time).

```
uc1 <- uncertainty(o, type="asympt", B=5000)
uc2 <- uncertainty(o, type="boot", B=200)
uc3 <- uncertainty(o, type="multi", B=200)

uc1$uncertainty[[1]]
```

```
## Univariate opticut uncertainty results, type = asymp, B = 5000
##
##      best          I          mu0          mu1
## 1 2:5001  Min.    :0.2381  Min.    :1.120  Min.    :2.863
##          1st Qu.:0.5299  1st Qu.:1.640  1st Qu.:4.022
##          Median :0.5846  Median :1.814  Median :4.356
##          Mean   :0.5764  Mean   :1.835  Mean   :4.387
##          3rd Qu.:0.6319  3rd Qu.:2.001  3rd Qu.:4.727
##          Max.   :0.7970  Max.   :3.401  Max.   :6.587
```

```
uc2$uncertainty[[1]]
```

```
## Univariate opticut uncertainty results, type = boot, B = 200
##
##      best          I          mu0          mu1
## 1 2:201  Min.    :0.3206  Min.    :1.244  Min.    :3.178
##          1st Qu.:0.5356  1st Qu.:1.651  1st Qu.:4.014
##          Median :0.5801  Median :1.801  Median :4.408
##          Mean   :0.5780  Mean   :1.819  Mean   :4.365
##          3rd Qu.:0.6270  3rd Qu.:1.986  3rd Qu.:4.685
##          Max.   :0.7412  Max.   :2.522  Max.   :5.930
```

```
uc3$uncertainty[[1]]
```

```
## Univariate opticut uncertainty results, type = multi, B = 200
##
##      best          I          mu0          mu1
## 1      : 1  Min.    :0.4628  Min.    :0.7104  Min.    :3.137
## 1 2    :183 1st Qu.:0.6463  1st Qu.:1.6341  1st Qu.:4.046
## 1 2 4   : 2  Median :0.7165  Median :1.8144  Median :4.532
## 2      : 15  Mean   :0.7132  Mean   :1.8115  Mean   :4.530
##          3rd Qu.:0.7833  3rd Qu.:1.9957  3rd Qu.:4.915
##          Max.   :0.9322  Max.   :2.6809  Max.   :6.141
```

```
## performance comparisons for 10 species
YYY <- cbind(Y, Y, Y, Y, Y, Y, Y, Y, Y, Y)
colnames(YYY) <- LETTERS[1:10]
o <- opticut(YYY ~ x2, strata=x0, dist="poisson", comb="rank")

library(parallel)
cl <- makeCluster(2)
system.time(uncertainty(o, type="asymp", B=5000))
```

```
##      user  system elapsed
##    0.090   0.001   0.091
```

```
system.time(uncertainty(o, type="asymp", B=5000, cl=cl))
```

```
##      user  system elapsed
##    0.011   0.002   1.038
```

```
system.time(uncertainty(o, type="boot", B=100))
```

```
##      user  system elapsed  
##    2.890    0.065    2.963
```

```
system.time(uncertainty(o, type="boot", B=100, cl=c1))
```

```
##      user  system elapsed  
##    0.009    0.001    1.447
```

```
system.time(uncertainty(o, type="multi", B=100))
```

```
##      user  system elapsed  
##   13.622    0.250   13.898
```

```
system.time(uncertainty(o, type="multi", B=100, cl=c1))
```

```
##      user  system elapsed  
##    0.009    0.001    6.912
```

```
stopCluster(c1)
```

7 Opticut logo

```
y <- matrix(c(  
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
  1, 1, 0, 0, 1, 1, 1, 0, 0, 1,  
  1, 0, 1, 1, 0, 1, 0, 1, 1, 1,  
  1, 0, 1, 1, 0, 1, 0, 1, 1, 1,  
  1, 0, 1, 1, 0, 1, 0, 1, 1, 1,  
  1, 1, 0, 0, 1, 1, 1, 0, 0, 1,  
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1),  
  7, 10, byrow=TRUE)  
yy <- t(1-y*0.85)  
colnames(yy) <- paste0("Spp", 1:7)  
x <- opticut(yy ~ 1, strata=0:9)  
plot(x, sort=FALSE, cut=-Inf, xlab="", ylab="")
```

