

opticut: likelihood based optimal partitioning for indicator species analysis

Peter Solymos, Ermiyas T. Azeria, Bela Tothmeresz

November 17, 2015

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, no model,
- data types not always compatible with randomization (i.e. decimals),
- confounding effects to classification accuracy can impact power.

Goals:

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

Theory

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 describe the relationship between the observations and the site descriptors through the probability density function $P(Y_i = y_i | z_i^{(m)}, x_{ij}, \theta)$ 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.

The indicator value (I) for each candidate partition can be calculated 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 $\mu_0^{(m)} = E[Y_i | z_i^{(m)} = 0, x_{ij} = 0]$ and $\mu_1^{(m)} = E[Y_i | z_i^{(m)} = 1, x_{ij} = 0]$ are expected values for the observations given the binary partition $z_i^{(m)}$ and at 0 value for all x_{ij} .

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

```
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
```

```
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)
```

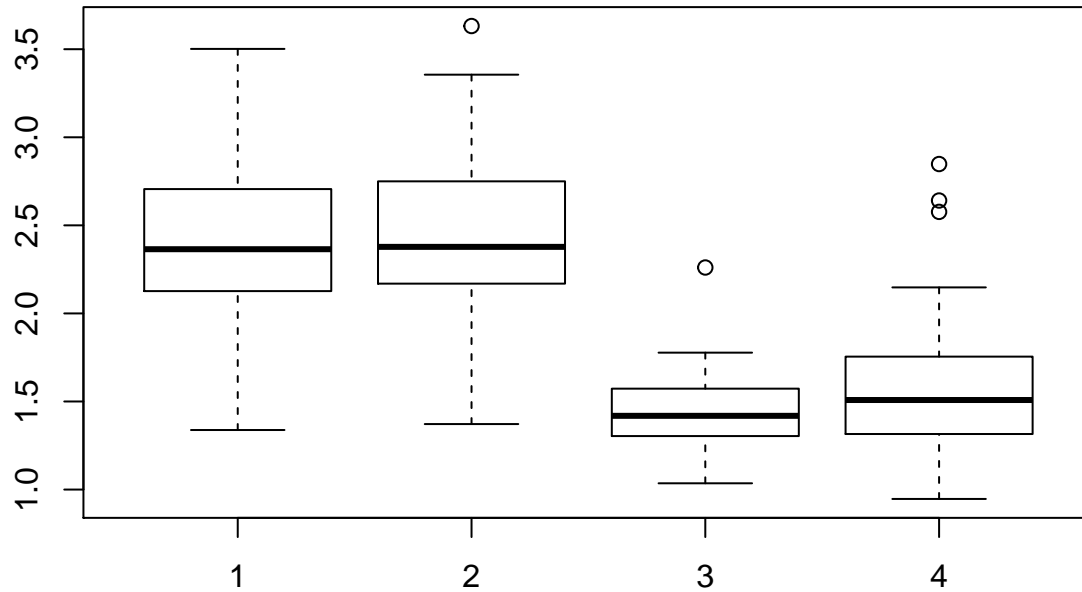
```
## [1] FALSE
## attr("comp")
##      i j
## [1,] 1 2
## [2,] 1 9
## attr("same")
##      i j
## [1,] 9 2
```

Poisson count model example

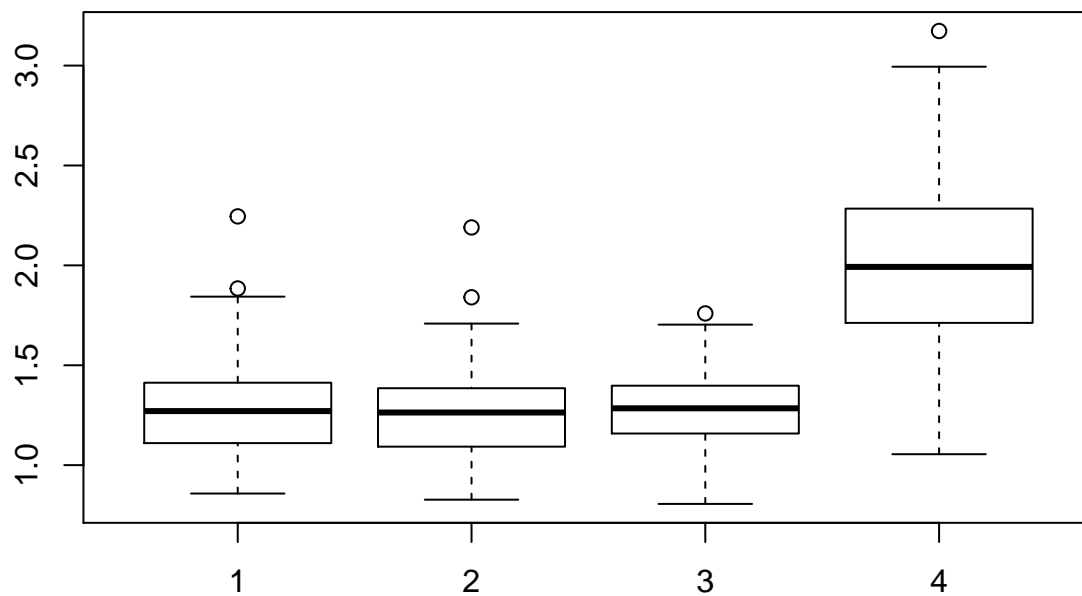
```
set.seed(1234)
n <- 200
x0 <- sample(1:4, n, TRUE)
x1 <- ifelse(x0 %in% 1:2, 1, 0)
x2 <- rnorm(n, 0.5, 1)
table(x0,x1)
```

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

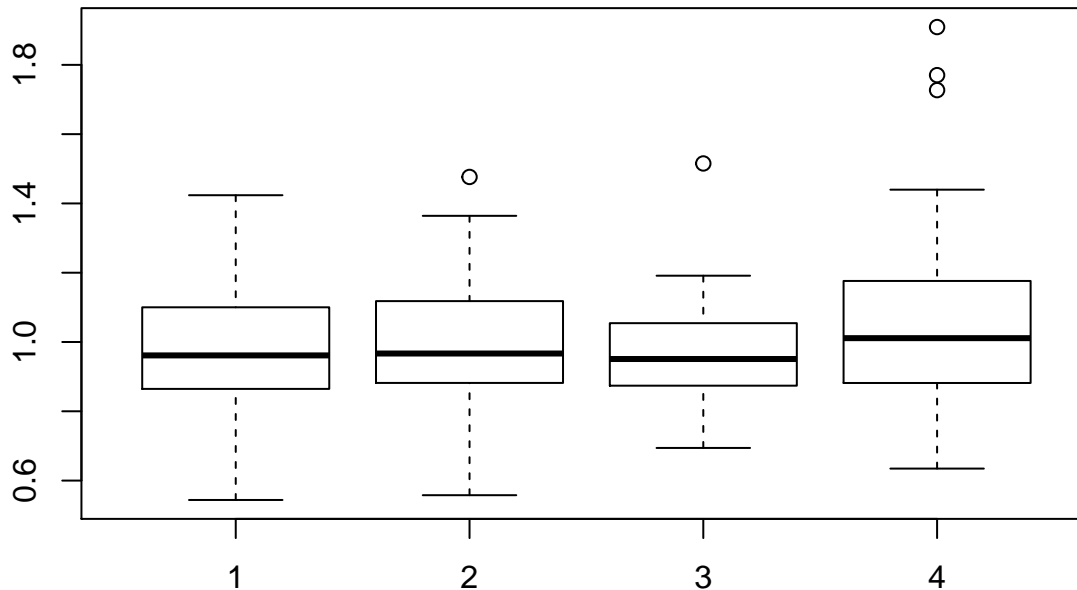
```
lam1 <- exp(0.5 + 0.5*x1 + -0.2*x2)
boxplot(lam1~x0)
```



```
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, dist = poisson
## I = 0.3431; w = 0.9842; H = 0.9687; logL_null = -343.8
##
## Best supported models with logLR >= 2:
##   assoc      I  null  mu0  mu1  logL logLR      w
## 1 2    +++ 0.3431 2.198 1.739 2.647 -335.4 8.396 0.984200
## 3      -- 0.2667 2.198 2.337 1.714 -340.8 3.051 0.004694
## 4      -- 0.2666 2.198 2.371 1.739 -340.8 3.027 0.004583
## 2      ++ 0.2258 2.198 2.048 2.645 -341.2 2.633 0.003092
## 1      ++ 0.2240 2.198 2.048 2.639 -341.2 2.598 0.002987
## 7 binary splits (2 models not shown)
```

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

```
## Univariate opticut results, dist = poisson
## I = 0.3706; w = 0.9254; H = 0.8584; logL_null = -315.6
##
## Best supported models with logLR >= 2:
##   assoc      I  null  mu0  mu1  logL logLR      w
## 4      ++ 0.3706 1.301 1.134 1.803 -309.3 6.245 0.92545
## 1 3      -- 0.2552 1.301 1.486 1.107 -312.5 3.119 0.04063
## 1 2      -- 0.2123 1.301 1.458 1.149 -313.5 2.064 0.01415
## 7 binary splits (4 models not shown)
```

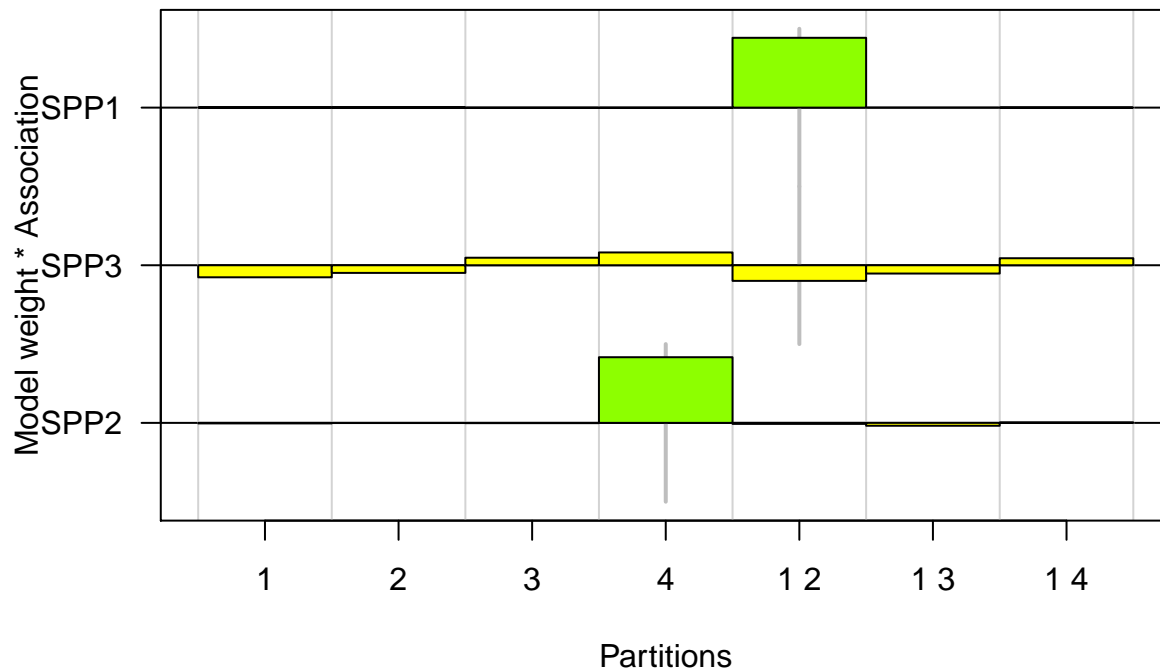
```
opticut1(Y3, X, Z, dist="poisson")
```

```
## Univariate opticut results, dist = poisson
## I = 0.1749; w = 0.2205; H = 0.1562; logL_null = -244.4
##
## Best supported model:
##      assoc      I    null  mu0    mu1  logL logLR      w
## 1 2      - 0.1749 0.9997 1.096 0.9047 -243.6 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, dist = poisson
##
## Call:
## opticut(formula = Y ~ x2, strata = x0, dist = "poisson", comb = "all")
##
##      split assoc      I    null  mu0    mu1 logLR      w
## SPP1    1 2    +++ 0.3431 2.198 1.739 2.647 8.396 0.9842
## SPP2     4    ++ 0.3706 1.301 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.

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 highest expected value among the partitions.

The function `rankComb` fits the model with multiple ($K > 2$) factor levels to find out the ranking, and returns a binary classification matrix 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, dist = poisson
##
## Call:
## opticut(formula = Y ~ x2, strata = x0, dist = "poisson", comb = "rank")
##
##      split assoc      I null  mu0  mu1 logLR      w
## SPP1    1 2    +++ 0.3431 2.198 1.739 2.647 8.396 0.9922
## SPP2     4    ++ 0.3706 1.301 1.134 1.803 6.245 0.9505
## 4 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.

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": zero-inflated Poisson counts,
- "zinb": zero-inflated Negative Binomial counts,
- "ordered": response measured on ordinal scale, e.g. ordinal vegetation cover,
- "rspf": presence-only data using resource selection probability functions.

Gaussian

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

## Multivariate opticut results, dist = gaussian
##
## Call:
## opticut(formula = Y ~ x2, strata = x0, dist = "gaussian")
##
## 1 species, 4 binary splits
```

Legendre example

```
gr <- rep(1:5, each=5)
spp <- cbind(Species1=rep(c(4,6,5,3,2), each=5),
             Species2=c(rep(c(8,4,6), each=5), 4,4,2, rep(0,7)),
             Species3=rep(c(18,2,0,0,0), each=5))
rownames(spp) <- gr
spp
```

```
##   Species1 Species2 Species3
## 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, dist = gaussian
##
## Call:
## opticut(formula = spp ~ 1, strata = gr, dist = "gaussian", comb = "all")
##
##      split assoc      I null mu0  mu1 logLR      w
## Species3      1   +++ 0.9722   4 0.5 18.0 55.19 1.0000
## Species2     1 3   +++ 0.7143   4 2.0  7.0 14.82 0.4995
## Species1     2 3   +++ 0.4545   4 3.0  5.5 17.33 0.4999
## 15 binary splits
```

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

```
## Multivariate opticut results, dist = gaussian
##
## Call:
## opticut(formula = spp ~ 1, strata = gr, dist = "gaussian", comb = "rank")
##
##      split assoc      I null mu0  mu1 logLR      w
## Species3      1   +++ 0.9722   4 0.5 18.0 55.19 1.0000
## Species2     1 3   +++ 0.7143   4 2.0  7.0 14.82 0.4996
## Species1     2 3   +++ 0.4545   4 3.0  5.5 17.33 0.4999
## 5 binary splits
```

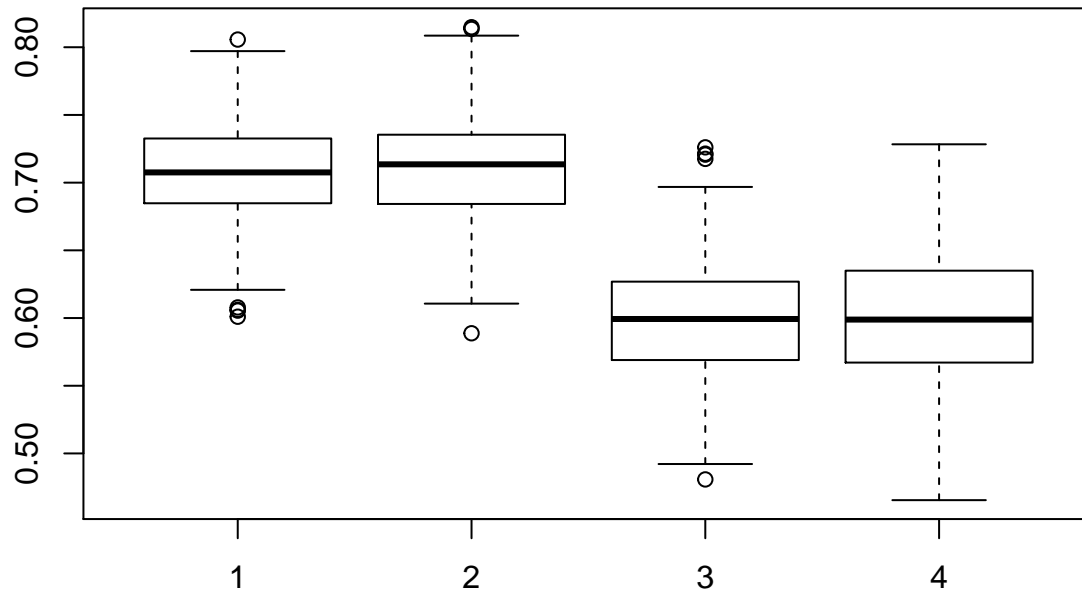
Binomial

```
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)
```

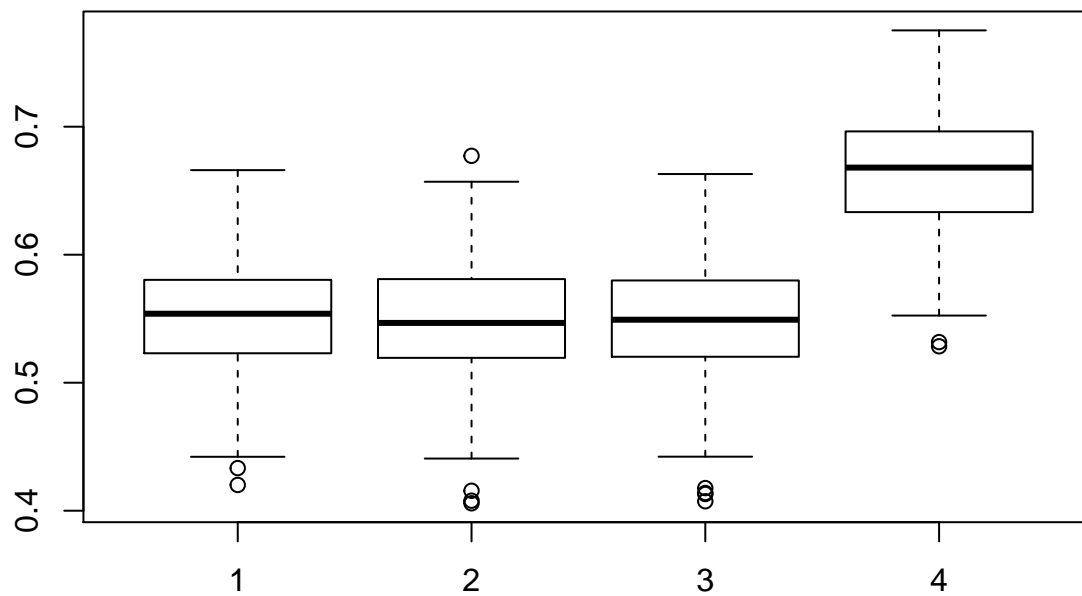
```
##      x1
## x0    0    1
```

```
## 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, dist = binomial
##
## Call:
## opticut(formula = Y ~ x2, strata = x0, dist = "binomial")
##
##      split assoc      I    null    mu0    mu1 logLR      w
## SPP1    1 2      ++ 0.1465 0.6819 0.6301 0.7383 6.954 0.9222
## SPP2     4      ++ 0.1731 0.5713 0.5415 0.6549 5.046 0.6378
## 4 binary splits

```

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 2.3-1

```

```

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"))

```

```

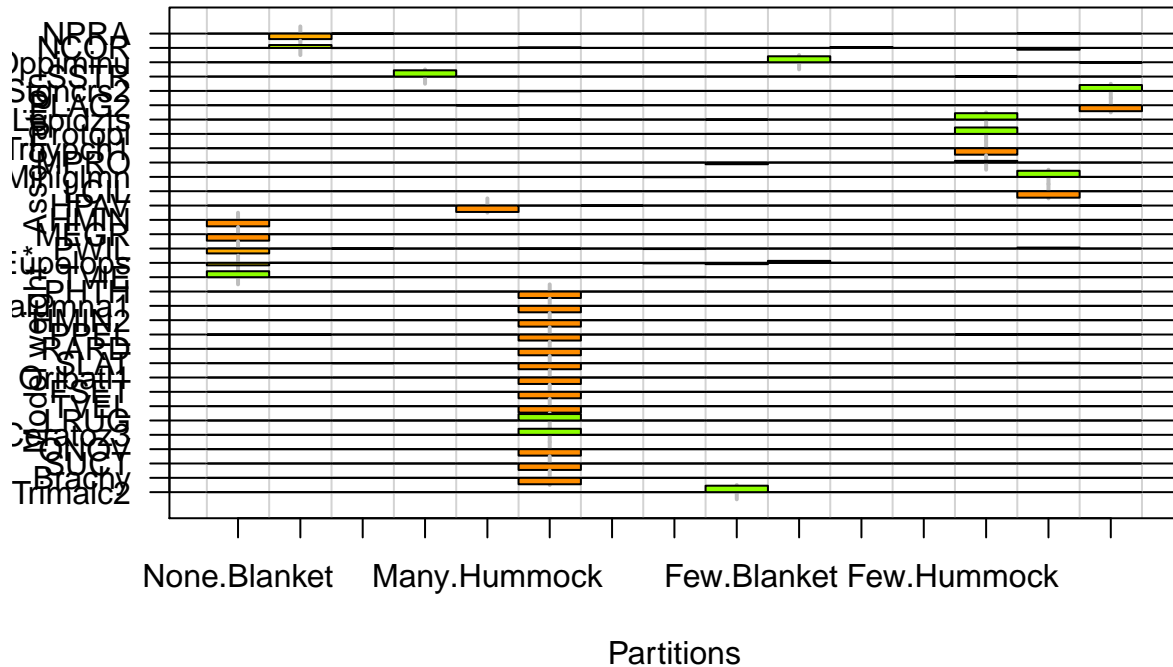
## Multivariate opticut results, dist = poisson
##
## Call:
## opticut(formula = as.matrix(mite) ~ SubsDens, data = mite.env,
##      strata = mite.env$hab, dist = "poisson", comb = "all")
##
##      split assoc      I    null    mu0
## NPRA          Few.Blanket --- 0.6860 1.39950 1.425e+00
## NCOR          Few.Blanket  ++ 0.5949 4.10891 4.238e+00
## Oppiminu  Few.Blanket Many.Blanket +++ 0.7000 1.92927 2.243e+00
## SSTR          Few.Hummock +++ 0.9617 0.02685 1.561e-03
## Stgnrcrs2 Few.Hummock Many.Hummock +++ 0.9200 0.10750 5.786e-03
## PLAG2        Few.Hummock Many.Hummock --- 0.7445 6.12998 1.683e+01
## Lepidzts     Many.Blanket Few.Hummock +++ 1.0000 0.07346 2.333e-09
## Protopl      Many.Blanket Few.Hummock +++ 0.9598 1.20870 7.695e-01
## Trhypch1     Many.Blanket Few.Hummock --- 0.8187 0.96168 7.493e-01
## MPRO         Many.Blanket Few.Hummock  ++ 0.7639 0.19546 2.034e-01
## Miniglmm     Many.Blanket Many.Hummock +++ 0.9649 0.42604 2.716e-02
## LCIL         Many.Blanket Many.Hummock --- 0.8529 8.00019 8.923e+00
## HPAV          Many.Hummock --- 0.5733 20.71728 3.500e+01

```

## HMIN	None.Blanket	---	0.9778	11.55798	1.880e+01
## MEGR	None.Blanket	---	0.9608	0.56493	9.729e-01
## PWIL	None.Blanket	---	0.8524	1.06153	1.637e+00
## Eupelops	None.Blanket	--	0.8159	1.04856	1.564e+00
## TVIE	None.Blanket	+++	0.7138	4.64007	2.320e+00
## PHTH	None.Blanket Few.Blanket	---	1.0000	0.74858	1.463e+00
## Galumna1	None.Blanket Few.Blanket	---	1.0000	0.81936	1.523e+00
## HMIN2	None.Blanket Few.Blanket	---	1.0000	0.87891	1.785e+00
## PPEL	None.Blanket Few.Blanket	--	1.0000	0.06473	1.349e-01
## RARD	None.Blanket Few.Blanket	---	1.0000	1.42609	2.546e+00
## SLAT	None.Blanket Few.Blanket	---	1.0000	0.39983	7.284e-01
## Oribatl1	None.Blanket Few.Blanket	---	0.9738	0.63616	1.311e+00
## FSET	None.Blanket Few.Blanket	---	0.9735	1.48766	2.726e+00
## TVEL	None.Blanket Few.Blanket	---	0.9710	7.06907	1.297e+01
## LRUG	None.Blanket Few.Blanket	+++	0.8330	24.86572	8.516e+00
## Ceratoz3	None.Blanket Few.Blanket	+++	0.7209	12.20564	7.098e+00
## ONOV	None.Blanket Few.Blanket	---	0.7161	29.47746	4.183e+01
## SUCT	None.Blanket Few.Blanket	---	0.6484	20.82412	2.901e+01
## Brachy	None.Blanket Few.Blanket	---	0.6273	12.38763	1.689e+01
## Trimalc2	None.Blanket Many.Hummock	+++	0.9871	10.88546	6.359e-02
##	mul logLR w				
## NPRA	4.474e-01	8.523	0.8540		
## NCOR	1.046e+01	6.241	0.4468		
## Oppiminu	7.475e+00	12.310	0.9245		
## SSTR	4.080e-02	23.076	0.9724		
## Stgnrcs2	7.230e-02	29.557	0.9147		
## PLAG2	4.300e+00	9.162	0.9210		
## Lepidzts	8.547e-01	13.044	0.9912		
## Protopl	1.916e+01	22.746	0.9936		
## Trhypch1	1.358e-01	35.539	0.9998		
## MPRO	8.614e-01	2.539	0.2639		
## Miniglmn	7.745e-01	12.918	0.9445		
## LCIL	1.312e+00	540.701	1.0000		
## HPAV	1.493e+01	26.281	0.9944		
## HMIN	4.177e-01	99.969	1.0000		
## MEGR	3.818e-02	35.465	1.0000		
## PWIL	2.416e-01	12.421	0.7564		
## Eupelops	2.878e-01	6.536	0.3863		
## TVIE	8.106e+00	11.050	0.9240		
## PHTH	2.525e-09	56.766	1.0000		
## Galumna1	3.486e-09	42.750	1.0000		
## HMIN2	5.504e-09	87.196	1.0000		
## PPEL	6.501e-10	7.620	0.9359		
## RARD	1.251e-08	54.142	1.0000		
## SLAT	4.062e-09	17.854	0.9399		
## Oribatl1	3.441e-02	69.712	1.0000		
## FSET	7.211e-02	69.026	1.0000		
## TVEL	3.759e-01	331.735	1.0000		
## LRUG	5.100e+01	218.994	1.0000		
## Ceratoz3	2.543e+01	16.217	0.9392		
## ONOV	1.188e+01	189.632	1.0000		
## SUCT	1.020e+01	136.668	1.0000		
## Brachy	6.294e+00	63.635	1.0000		
## Trimalc2	4.929e+00	83.215	1.0000		

```
## 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
## 0.589 0.016 0.610
```

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

```
## user system elapsed
## 1.373 0.045 1.429
```

```
## sequential
```

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

```
## user system elapsed
## 0.579 0.017 0.599
```

```
## 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
## 0.010 0.001 0.306
```

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

```
##      user  system elapsed
##    0.005   0.004   0.366
```

Percenages

Dune data, cover type data

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

```
library(vegan)
data(dune)
data(dune.env)

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

Loading required package: MASS

```
summary(x)
```

```
## Multivariate opticut results, dist = ordered
##
## Call:
## opticut(formula = Dune ~ 1, strata = dune.env$Management, dist = "ordered")
##
##           split assoc      I null      mu0      mu1 logLR      w
## Vicilath      BF    ++ 0.4873  0.5 5.00e-01 0.9752 3.024 0.5264
## Bromhord      BF    ++ 0.4844  0.5 5.00e-01 0.9697 2.741 0.4712
## Achimill     BF HF    ++ 0.4567  0.5 5.00e-01 0.9204 3.009 0.6036
## Trifrepe     BF HF    ++ 0.4549  0.5 5.00e-01 0.9173 3.457 0.5762
## Planlanc BF HF NM    ++ 0.5000  0.5 5.00e-01 1.0000 3.245 0.4406
## Anthodor BF HF NM    ++ 0.5000  0.5 5.00e-01 1.0000 2.657 0.7082
## Scorausu BF HF NM    ++ 0.5000  0.5 5.00e-01 1.0000 6.034 0.9175
## Poatriv BF HF SF    +++ 0.5000  0.5 5.00e-01 1.0000 9.346 0.9827
## Elymrepe BF HF SF    ++ 0.5000  0.5 5.00e-01 1.0000 2.657 0.6848
## Lolipere BF HF SF    ++ 0.4786  0.5 5.00e-01 0.9590 4.351 0.6845
## Poaprat BF HF SF    ++ 0.4724  0.5 5.00e-01 0.9476 4.309 0.7658
## Hyporadi      BF NM    ++ 0.5000  0.5 5.00e-01 1.0000 2.726 0.7053
## Trifprat      HF    ++ 0.5000  0.5 5.00e-01 1.0000 5.089 0.8579
## Rumeacet      HF    ++ 0.4936  0.5 5.00e-01 0.9873 5.801 0.8744
## Juncbufo      HF SF    ++ 0.5000  0.5 5.00e-01 1.0000 2.798 0.7000
## Comapalu      NM    ++ 1.0000  0.1 1.17e-09 0.3333 2.683 0.7516
## Salirepe      NM    ++ 0.5000  0.5 5.00e-01 1.0000 4.295 0.7984
## Airaprae      NM    ++ 0.5000  0.5 5.00e-01 1.0000 2.683 0.6520
```

```
## Alop geni      SF      ++ 0.4724 0.5 5.00e-01 0.9476 4.135 0.4060
## Agrostol      SF      ++ 0.4627 0.5 5.00e-01 0.9306 3.382 0.5417
## 4 binary splits
## 10 species not shown
```

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

```
## Multivariate opticut results, dist = binomial
##
## Call:
## opticut(formula = Dune ~ 1, strata = dune.env$Management, dist = "binomial")
##
##          split assoc      I null      mu0      mu1 logLR      w
## Vicilath      BF      ++ 0.9118 0.15 5.882e-02 0.6667 2.741 0.4650
## Bromhord      BF HF      ++ 0.8333 0.25 8.333e-02 0.5000 2.259 0.4345
## Achimill      BF HF      ++ 0.7333 0.35 1.667e-01 0.6250 2.250 0.6992
## Lolipere      BF HF      ++ 0.6667 0.60 3.333e-01 1.0000 5.822 0.8987
## Poaprat       BF HF      ++ 0.5000 0.70 5.000e-01 1.0000 3.900 0.7383
## Trifrepe      BF HF      ++ 0.3333 0.80 6.667e-01 1.0000 2.370 0.6700
## Planlanc      BF HF NM      ++ 1.0000 0.35 8.647e-09 0.5000 3.245 0.6890
## Anthodor      BF HF NM      ++ 1.0000 0.30 8.647e-09 0.4286 2.657 0.6314
## Scorautu      BF HF NM      ++ 0.3333 0.90 6.667e-01 1.0000 2.683 0.7441
## Poatriv       BF HF SF      +++ 1.0000 0.65 3.181e-09 0.9286 9.346 0.7609
## Alop geni      BF HF SF      ++ 1.0000 0.40 8.647e-09 0.5714 3.900 0.4732
## Elymrepe      BF HF SF      ++ 1.0000 0.30 8.647e-09 0.4286 2.657 0.6812
## Hyporadi      BF NM      ++ 1.0000 0.15 1.170e-09 0.3333 2.726 0.7635
## Trifprat      HF      ++ 1.0000 0.15 1.170e-09 0.6000 5.089 0.8579
## Rumeacet      HF      ++ 0.9167 0.25 6.667e-02 0.8000 5.071 0.7703
## Agrostol      HF NM SF      ++ 1.0000 0.50 8.647e-09 0.5882 2.346 0.4403
## Juncbufo      HF SF      ++ 1.0000 0.20 3.181e-09 0.3636 2.798 0.7960
## Salirepe      NM      ++ 1.0000 0.15 1.170e-09 0.5000 4.295 0.9007
## Airaprae      NM      ++ 1.0000 0.10 1.170e-09 0.3333 2.683 0.7175
## Comapalu      NM      ++ 1.0000 0.10 1.170e-09 0.3333 2.683 0.7516
## 4 binary splits
## 10 species not shown
```

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

```
## Loading required package: betareg
```

```
summary(x)
```

```
## Multivariate opticut results, dist = beta
##
## Call:
## opticut(formula = Dune ~ 1, strata = dune.env$Management, dist = "beta")
##
```

```
##          split assoc      I      null      mu0      mu1 logLR      w
## Bromhord      BF    ++ 0.6103 0.12983 0.10488 0.2691 2.535 0.6478
## Vicilath      BF    ++ 0.5491 0.07074 0.05974 0.1325 4.369 0.9287
## Lolipere     BF HF    ++ 0.5935 0.33166 0.21057 0.5181 4.104 0.5739
## Planlanc     BF HF    ++ 0.5340 0.18557 0.12810 0.2749 2.467 0.5496
## Achimill     BF HF    ++ 0.5043 0.13251 0.09444 0.1905 2.596 0.6056
## Trifrepe     BF HF    ++ 0.4954 0.28469 0.20483 0.4059 3.395 0.6065
## Scoraustu BF HF NM    ++ 0.5325 0.31799 0.17817 0.3811 5.074 0.8647
## Poatriv     BF HF SF    ++ 0.7561 0.36584 0.12172 0.4991 7.301 0.8074
## Poaprat     BF HF SF    ++ 0.6127 0.28059 0.13521 0.3491 4.391 0.4868
## Rumeacet      HF    ++ 0.7280 0.15147 0.09043 0.3324 6.224 0.9885
## Trifprat      HF    ++ 0.6150 0.10177 0.07273 0.1889 3.846 0.8910
## Salirepe      NM    ++ 0.5668 0.11261 0.08105 0.1871 2.862 0.8470
## Alop geni     SF    ++ 0.6567 0.24799 0.15803 0.4603 4.225 0.8093
## Agrostol      SF    ++ 0.5757 0.29251 0.20803 0.4903 3.096 0.7098
## 4 binary splits
## 16 species not shown
```

Stratigraphy example

```
library(rioja)
```

```
## This is rioja 0.9-5
```

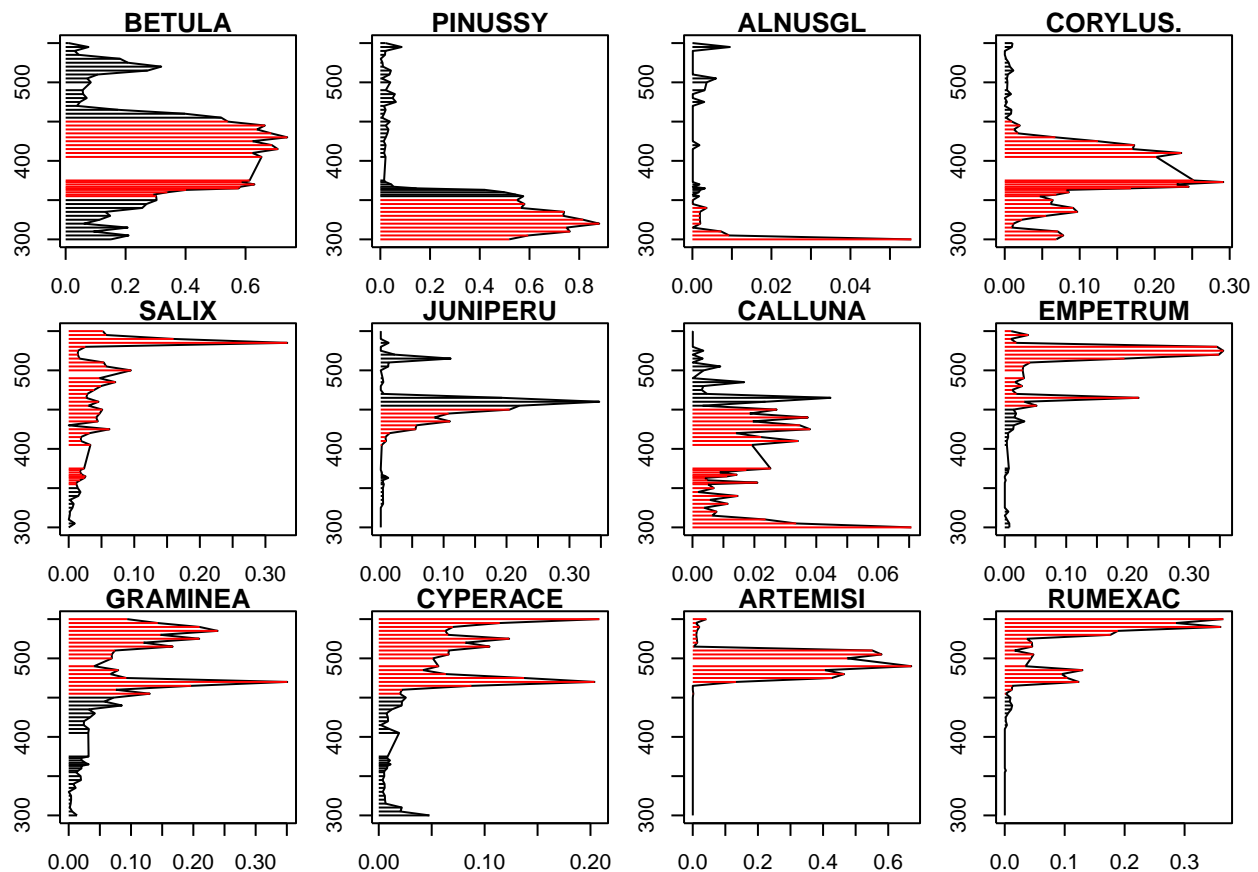
```
data(aber)
strat.plot(aber$spec, aber$ages$Depth, scale.percent=TRUE, y.rev=TRUE)
```



```
##          mu0      mu1 logLR      w
## PINUSSY  0.083196 0.669499 26.667 1.0000
## ALNUSGL  0.002089 0.004128  2.081 0.4201
## CORYLUS. 0.019366 0.096616 17.133 0.9940
## CALLUNA  0.005234 0.019421 12.719 0.5430
## BETULA   0.153662 0.568254 27.575 1.0000
## SALIX    0.014015 0.044639  8.393 0.5344
## JUNIPERU 0.025317 0.071110  4.017 0.6459
## RUMEXAC  0.015648 0.095248 15.892 0.9888
## CYPERACE 0.015826 0.080857 24.351 1.0000
## GRAMINEA 0.028003 0.131311 21.569 0.9914
## EMPETRUM 0.020863 0.079641 10.171 0.6500
## ARTEMISI 0.046111 0.145790  7.025 0.9708
## 5 binary splits
## 1 species not shown
```

```
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)
```

Presence-only data

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

library(ResourceSelection)

## ResourceSelection 0.2-4    2014-05-19

## 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))

(mod <- opticut1(Y, X, dd$x0, dist="rspf", m=0, B=0))

## Univariate opticut results, dist = rspf
## I = 0.129; w = 0.73; H = 0.6058; logL_null = -9169
##
## Best supported model with logLR >= 2:
##   assoc      I    null    mu0    mu1  logL logLR    w
## 3    ++ 0.129 0.8049 0.7359 0.8449 -9167 2.508 0.73
## 2 binary splits (1 model not shown)
```

Custom distributions

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

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, dist = lmefun
## I = 0.4023; w = 0.9985; H = 0.997; logL_null = -345.2
##
## Best supported models with logLR >= 2:
##      assoc      I null  mu0  mu1  logL  logLR      w
## 1 2    +++ 0.4023 1.99 1.480 2.476 -331.6 13.582 0.9984969
## 1      ++ 0.3498 1.99 1.745 2.683 -338.3  6.864 0.0012072
## 3      -- 0.3057 1.99 2.134 1.482 -340.6  4.620 0.0001279
## 4      -- 0.3039 1.99 2.168 1.509 -340.4  4.739 0.0001441
## 2      ++ 0.2210 1.99 1.858 2.386 -342.4  2.813 0.0000210
## 7 binary splits (2 models not shown)
```

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:

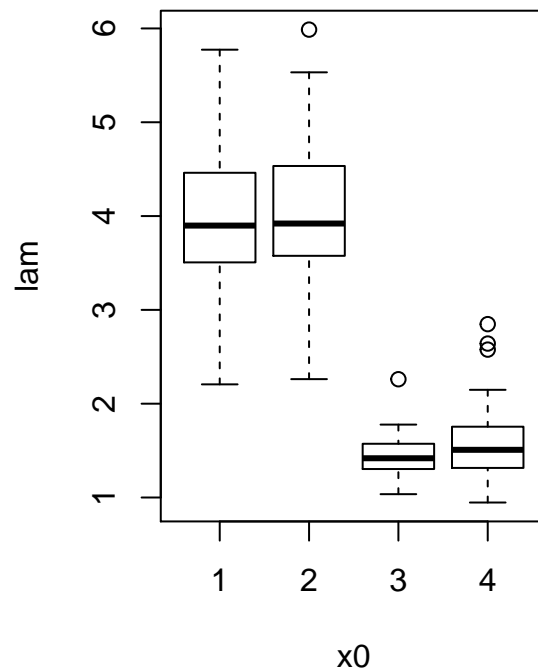
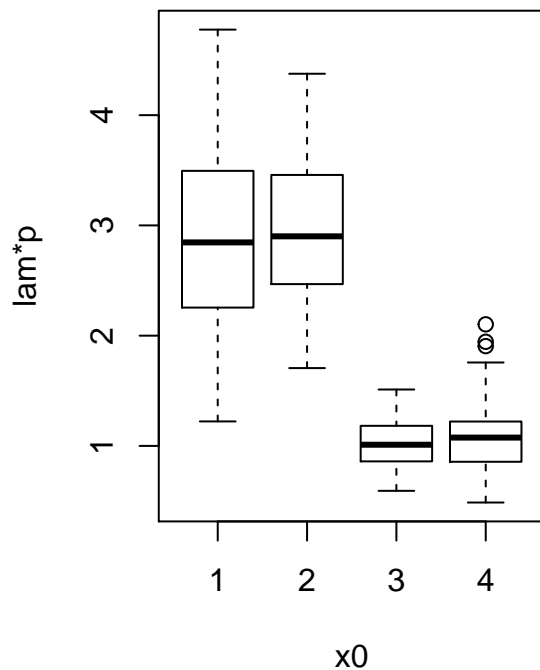
```
library(detect)
```

```
## Loading required package: Formula
## Loading required package: stats4
## detect 0.3-2      2014-05-15
```

```
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)
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
##      1.849369     -0.224221
##
## $logLik
## 'log Lik.' -368.6241 (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, dist = poisson
## I = 0.6251; w = 1; H = 1; logL_null = -383.1
##
## Best supported models with logLR >= -Inf:
##      assoc      I null  mu0  mu1  logL logLR      w
## 1 2      +++ 0.6251 2.272 1.232 3.286 -339.4 43.76 1.000e+00
## 1 2 3      +++ 0.5363 2.272 1.234 2.662 -366.9 16.22 1.101e-12
## 2      +++ 0.4317 2.272 1.906 3.353 -368.5 14.67 2.334e-13
## 3 binary splits

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

## Univariate opticut results, dist = svfun
## I = 0.6339; w = 1.797e-06; H = 1; logL_null = -368.6
##
## Best supported models with logLR >= -Inf:
##      assoc      I null  mu0  mu1  logL logLR      w
## 1 2 3      +++ 0.6339 6.356 2.055 5.612 -359.9 8.735 1.797e-06
## 1 2      +++ 0.6198 6.356 2.133 5.610 -346.7 21.964 1.000e+00
## 2      + 0.3865 6.356 3.020 4.923 -367.3 1.327 1.090e-09
## 3 binary splits

```

Sampling error differences: using offsets

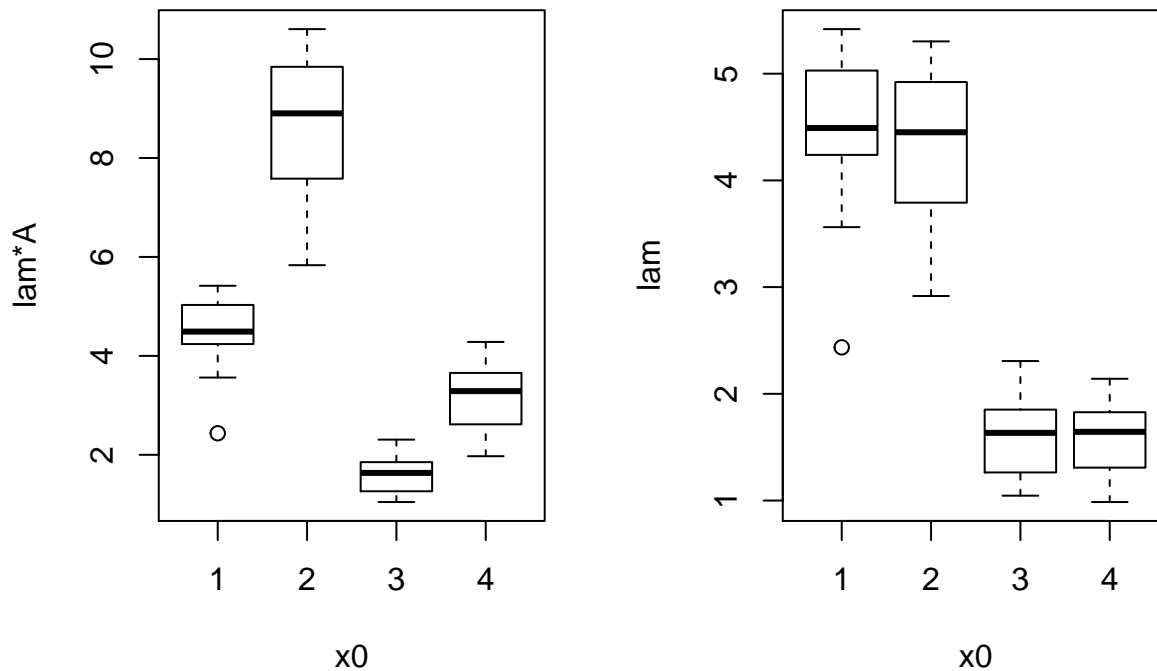
Not accounting for unequal sampling effort can be quite misleading, especially if that is related to habitat classes. This example shows how to take advantage of the other arguments passed to the ... in the `opticut` function.

```

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, dist = poisson
## I = 0.7135; w = 3.243e-08; H = 0.9792; logL_null = -154.2
##
## Best supported models with logLR >= 2:
##      assoc      I  null  mu0  mu1  logL logLR      w
## 1 2 4    +++ 0.7135 4.864 1.766 6.164 -129.4 24.77 3.243e-08
## 1 2      +++ 0.7064 4.864 2.192 7.464 -116.7 37.46 1.049e-02
## 2      +++ 0.7005 4.864 3.028 10.110 -112.2 42.01 9.895e-01
## 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, dist = poisson
## I = 0.6811; w = 1; H = 1; logL_null = -135.7
##
## Best supported models with logLR >= 2:
##      assoc      I null  mu0  mu1  logL logLR      w
## 1 2      +++ 0.6811 3.339 1.572 4.930 -103.3 32.34 1.000e+00
## 1 2 3      +++ 0.6466 3.339 1.431 4.049 -118.4 17.29 2.919e-07
## 2      +++ 0.5280 3.339 2.388 5.060 -119.0 16.68 1.584e-07
## 3 binary splits
```

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, dist = poisson
##
## Call:
## opticut(formula = Y ~ x2, strata = x0, dist = "poisson", comb = "rank")
##
##      split assoc      I null  mu0  mu1 logLR      w
## Species 1  1 2      +++ 0.583 2.855 1.819 4.361 12.46 0.9918
## 4 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:  glm(formula = Y ~ . - 1, family = poisson(link), 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.

Uncertainty

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

- "asymp": 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, 1, type="asymp", B=5000)
```

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

```
uc2 <- uncertainty(o, 1, type="boot", B=200)
uc3 <- uncertainty(o, 1, type="multi", B=200)
```

```
summary(uc1[[1]])
```

```
##    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
```

```
summary(uc2[[1]])
```

```
##    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
```

```
summary(uc3[[1]])
```

```
##      best      I      mu0      mu1
## 1      : 1   Min.    :0.3940   Min.    :0.7104   Min.    :3.137
## 1 2    :183  1st Qu.:0.5365   1st Qu.:1.6341   1st Qu.:4.046
## 1 2 4   : 2   Median :0.5936   Median :1.8144   Median :4.532
## 2      : 15   Mean    :0.5937   Mean    :1.8115   Mean    :4.530
##          3rd Qu.:0.6514   3rd Qu.:1.9957   3rd Qu.:4.915
##          Max.    :0.8127   Max.    :2.6809   Max.    :6.141
```

```
plot(density(uc1[[1]]$I), ylim=c(0,7))
lines(density(uc2[[1]]$I), col=2)
lines(density(uc3[[1]]$I), col=4)
legend("topleft", lty=1, col=c(1,2,4),
       legend=c("asyp", "boot", "multi"))
```

density.default(x = uc1[[1]]\$I)

