# Calibration with opticut

*Peter Solymos and Ermias T. Azeria*

*December 22, 2016*

# Contents

# Chapter 1

# Calibration with single species

This is not necessarily great.

```
library(opticut)
```

```
## Loading required package: pbapply

## opticut 0.0-9     2016-10-30
```

```
library(dclone)
```

```
## Warning: package 'dclone' was built under R version 3.2.5

## Loading required package: coda

## Loading required package: parallel

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 3.2.5

## dclone 2.1-2      2016-03-11
```

```
library(rjags)
```

```
## Linked to JAGS 4.2.0

## Loaded modules: basemod,bugs
```

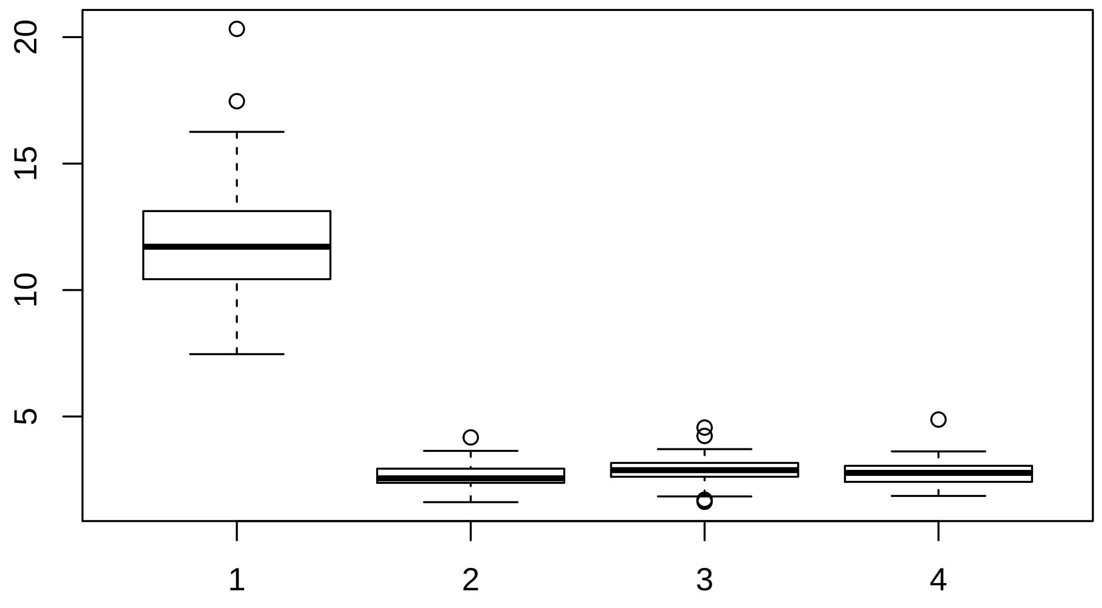
```
source("~/repos/opticut/extras/calibrate.R")

set.seed(234)
n <- 50*4
K <- 4
g <- sample(1:K, n, replace=TRUE)
x <- rnorm(n, 0, 1)
z <- ifelse(g == 1, 1, 0)
```

```
b0 <- 1
b1 <- 1.5
a <- 0.2
mu <- b0 + z*b1 + x*a
#p <- plogis(mu)
#y <- rbinom(n, 1, p)
lam <- exp(mu)
y <- rpois(n, lam)
table(y, g)
```

```
##     g
## y     1  2  3  4
##   0   0  4  5  2
##   1   0 10 12 12
##   2   0  9  9 10
##   3   1  8 13  7
##   4   1  6 11  6
##   5   0  7  8  5
##   6   2  0  3  4
##   7   0  1  0  1
##   8   4  0  2  0
##   9   7  0  0  0
##   10  4  0  0  0
##   11  4  0  0  0
##   12  3  0  0  0
##   13  4  0  0  0
##   14  2  0  0  0
##   15  1  0  0  0
##   16  4  0  0  0
##   17  4  0  0  0
##   18  2  0  0  0
##   20  1  0  0  0
##   21  1  0  0  0
```

```
boxplot(lam ~ g)
```



```
df <- data.frame(y=y, x=x, g=g)
df <- df[-(1:5),]
#o <- opticut(y ~ x, data=df, strata=g, dist="binomial")
o <- opticut(y ~ x, data=df, strata=g, dist="poisson")
summary(o)
```

```
## Multivariate opticut results, comb = rank, dist = poisson
##
## Call:
## opticut.formula(formula = y ~ x, data = df, strata = g, dist = "poisson")
##
## Best supported model with logLR >= 2:
##      split assoc      I   mu0   mu1 logLR w
## Sp 1     1   +++ 0.6293 2.744 12.06 245.8 1
## 3 binary splits
```

```
co <- calibrate(o, cbind("Sp 1"=(y[1:5])), model.matrix(~x, data.frame(x=x[1:5])))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
```

```
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 6
##    Total graph size: 70
##
## Initializing model
```

```
y[1:5]
```

```
## [1]  4  2 10  4 10
```

```
g[1:5]
```

```
## [1] 3 4 1 4 1
```

```
co$gnew
```

```
## [1] "4" "3" "1" "2" "1"
```

```
round(co$pi, 2)
```

```
##         1    2    3    4
## [1,] 0.01 0.33 0.33 0.33
## [2,] 0.00 0.33 0.34 0.34
## [3,] 1.00 0.00 0.00 0.00
## [4,] 0.00 0.34 0.33 0.32
## [5,] 1.00 0.00 0.00 0.00
```

We get g=1 cases correctly classified (this is where Sp1 is to be found), But the rest is just all equally probable. All what we see is that it is not g=1. All makes sense, but not very useful. Luckily for us, we usually have >1 species.

# Chapter 2

# Calibration with multiple species

This should work much better in principle.

```r
set.seed(234)
n <- 50*4
K <- 4
g <- sample(1:K, n, replace=TRUE)
g[1:K] <- 1:K
x <- rnorm(n, 0, 1)
z <- ifelse(g %in% 1:2, 1, 0)
b0 <- 1
b1 <- 1.5
a <- 0.2
mu <- b0 + z*b1 + x*a
lam <- exp(mu)
y1 <- rpois(n, lam)

z <- ifelse(g %in% 2:3, 1, 0)
b0 <- 1
b1 <- 1.5
a <- 0.2
mu <- b0 + z*b1 + x*a
lam <- exp(mu)
y2 <- rpois(n, lam)
y <- cbind(y1=y1, y2=y2)

df <- data.frame(x=x, g=g)
df <- df[-(1:5),]
yy <- y[-(1:5),]
o <- opticut(yy ~ x, data=df, strata=g, dist="poisson")
summary(o)
```

```
## Multivariate opticut results, comb = rank, dist = poisson
##
## Call:
## opticut.formula(formula = yy ~ x, data = df, strata = g, dist = "poisson")
##
## Best supported models with logLR >= 2:
##    split assoc      I    mu0   mu1 logLR w
## y1   1+2   +++ 0.6344 2.823 12.62 331.6 1
## y2   2+3   +++ 0.6502 2.499 11.79 314.5 1
## 3 binary splits
```

```
co <- calibrate(o, y[1:5,], df[1:5,])
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 10
##    Unobserved stochastic nodes: 7
##    Total graph size: 122
##
## Initializing model
```

```
y[1:5,]
```

```
##      y1 y2
## [1,] 14  3
## [2,] 14 17
## [3,]  3  6
## [4,]  4  2
## [5,]  5  1
```

```
g[1:5]
```

```
## [1] 1 2 3 4 1
```

```
co$gnew
```

```
## [1] "1" "2" "3" "4" "4"
```

```
round(co$pi, 2)
```

```
##         1    2    3    4
## [1,] 0.98 0.02 0.00 0.00
## [2,] 0.00 1.00 0.00 0.00
## [3,] 0.00 0.05 0.88 0.07
## [4,] 0.08 0.00 0.01 0.91
## [5,] 0.07 0.00 0.00 0.93
```

# Chapter 3

# Dolina and calibration

```
data(dolina)
dolina$samp$stratum <- as.integer(dolina$samp$stratum)
Y <- dolina$xtab#[dolina$samp$method=="Q",]
X <- dolina$samp#[dolina$samp$method=="Q",]
inew <- 1:nrow(Y) %in% sample(nrow(Y), 10)
Y <- Y[,colSums(Y[!inew,] > 0) >= 20]
#Y <- ifelse(Y > 0, 1, 0)
x <- X[!inew,]
xnew <- dolina$samp[inew,]
y <- Y[!inew,]
ynew <- Y[inew,]
o <- opticut(y ~ stratum + lmoist + method, data=x, strata=mhab, dist="poisson")
#o <- opticut(y ~ 1, data=x, strata=mhab, dist="poisson")

## binary calibration
cal <- calibrate(o, ynew, xnew)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 190
##    Unobserved stochastic nodes: 29
##    Total graph size: 1901
##
## Initializing model
```

```
#cal <- calibrate(o, ynew)
round(cal$pi,2)
```

```
##          LI    DW    TL    RO
## 11R3T 0.05 0.83 0.04 0.08
## 12A4Q 0.00 0.12 0.87 0.00
## 13A2T 0.46 0.05 0.47 0.03
## 1A1T  0.44 0.07 0.45 0.04
## 1A6T  0.43 0.05 0.44 0.08
## 3H2T  0.00 0.09 0.00 0.90
## 4A7T  0.38 0.07 0.38 0.18
## 4R3Q  0.00 0.00 0.00 1.00
## 7R1Q  0.00 0.00 0.00 1.00
## 9L3T  0.81 0.00 0.18 0.00
```

```
round(cor(cal$pi),2)
```

```
##       LI    DW    TL    RO
## LI  1.00 -0.29  0.23 -0.63
## DW -0.29  1.00 -0.18 -0.25
## TL  0.23 -0.18  1.00 -0.70
## RO -0.63 -0.25 -0.70  1.00
```

```
data.frame(predicted=cal$gnew,
    truth=dolina$samp[inew,"mhab"],
    OK=cal$gnew == dolina$samp[inew,"mhab"])
```

```
##    predicted truth    OK
## 1         DW    RO FALSE
## 2         TL    LI FALSE
## 3         TL    LI FALSE
## 4         TL    LI FALSE
## 5         TL    LI FALSE
## 6         RO    DW FALSE
## 7         LI    LI  TRUE
## 8         RO    RO  TRUE
## 9         RO    RO  TRUE
## 10        LI    TL FALSE
```

```
cm <- table(pred=cal$gnew, true=X[inew,"mhab"])
cm <- cm[rownames(cm),rownames(cm)]
cm
```

```
##     true
## pred DW LI RO TL
##   DW  0  0  1  0
##   LI  0  1  0  1
##   RO  1  0  2  0
##   TL  0  4  0  0
```

```
sum(diag(cm)) / sum(cm)
```

```
## [1] 0.3
```

```
## multinomial calibration using calibrate.default
oo <- lapply(1:ncol(y), function(i) {
    glm(y[,i] ~ mhab + stratum + lmoist + method, x, family=poisson)
})
xnew <- model.matrix(~ stratum + lmoist + method, xnew)
cal <- calibrate(oo, ynew, xnew, K=4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 190
##    Unobserved stochastic nodes: 29
##    Total graph size: 2187
##
## Initializing model
```

```
round(cal$pi,2)
```

```
##       [,1] [,2] [,3] [,4]
## 11R3T 0.07 0.66 0.11 0.16
## 12A4Q 0.00 0.02 0.98 0.00
## 13A2T 0.60 0.04 0.35 0.01
## 1A1T  0.64 0.03 0.31 0.02
## 1A6T  0.44 0.04 0.48 0.04
## 3H2T  0.00 0.60 0.00 0.40
## 4A7T  0.49 0.04 0.39 0.09
## 4R3Q  0.00 0.00 0.00 1.00
## 7R1Q  0.00 0.00 0.00 1.00
## 9L3T  0.82 0.01 0.18 0.00
```

```
round(cor(cal$pi),2)
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,]  1.00 -0.41  0.12 -0.63
## [2,] -0.41  1.00 -0.36 -0.04
## [3,]  0.12 -0.36  1.00 -0.62
## [4,] -0.63 -0.04 -0.62  1.00
```

```
data.frame(predicted=levels(x$mhab)[cal$gnew],
    truth=dolina$samp[inew,"mhab"],
    OK=levels(x$mhab)[cal$gnew] == dolina$samp[inew,"mhab"])
```

```
##    predicted truth    OK
## 1         DW    RO FALSE
## 2         TL    LI FALSE
## 3         LI    LI  TRUE
## 4         LI    LI  TRUE
## 5         TL    LI FALSE
## 6         DW    DW  TRUE
## 7         LI    LI  TRUE
## 8         RO    RO  TRUE
## 9         RO    RO  TRUE
## 10        LI    TL FALSE
```

```
cm <- table(pred=levels(x$mhab)[cal$gnew], true=X[inew,"mhab"])
cm <- cm[rownames(cm),rownames(cm)]
cm
```

```
##     true
## pred DW LI RO TL
##   DW  1  0  1  0
##   LI  0  3  0  1
##   RO  0  0  2  0
##   TL  0  2  0  0
```

```
sum(diag(cm)) / sum(cm)
```

```
## [1] 0.6
```

```
library(optpart)
```

```
## Warning: package 'optpart' was built under R version 3.2.5

## Loading required package: cluster

## Warning: package 'cluster' was built under R version 3.2.5

## Loading required package: labdsv

## Loading required package: mgcv

## Warning: package 'mgcv' was built under R version 3.2.5

## Loading required package: nlme

## Warning: package 'nlme' was built under R version 3.2.5

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

## Loading required package: MASS

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

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

```
## Warning: package 'plotrix' was built under R version 3.2.5
```

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

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

```
data(shoshsite)
data(shoshveg)

#elev <- cut(shoshsite$elevation, breaks=c(0, 7200, 8000, 9000, 20000))
#levels(elev) <- c("low", "mid1", "mid2", "high")
elev <- cut(shoshsite$elevation, breaks=c(0, 7800, 8900, 20000))
levels(elev) <- c("low", "mid", "high")

sveg <- as.matrix(shoshveg)
sveg[sveg > 0] <- 1

set.seed(1)
ss <- seq_len(nrow(sveg)) %in% sample.int(nrow(sveg), floor(nrow(sveg) * 0.9))
sveg1 <- sveg[ss,]
sveg1 <- sveg1[,colSums(sveg1) > 5]
elev1 <- elev[ss]
dim(sveg)
```

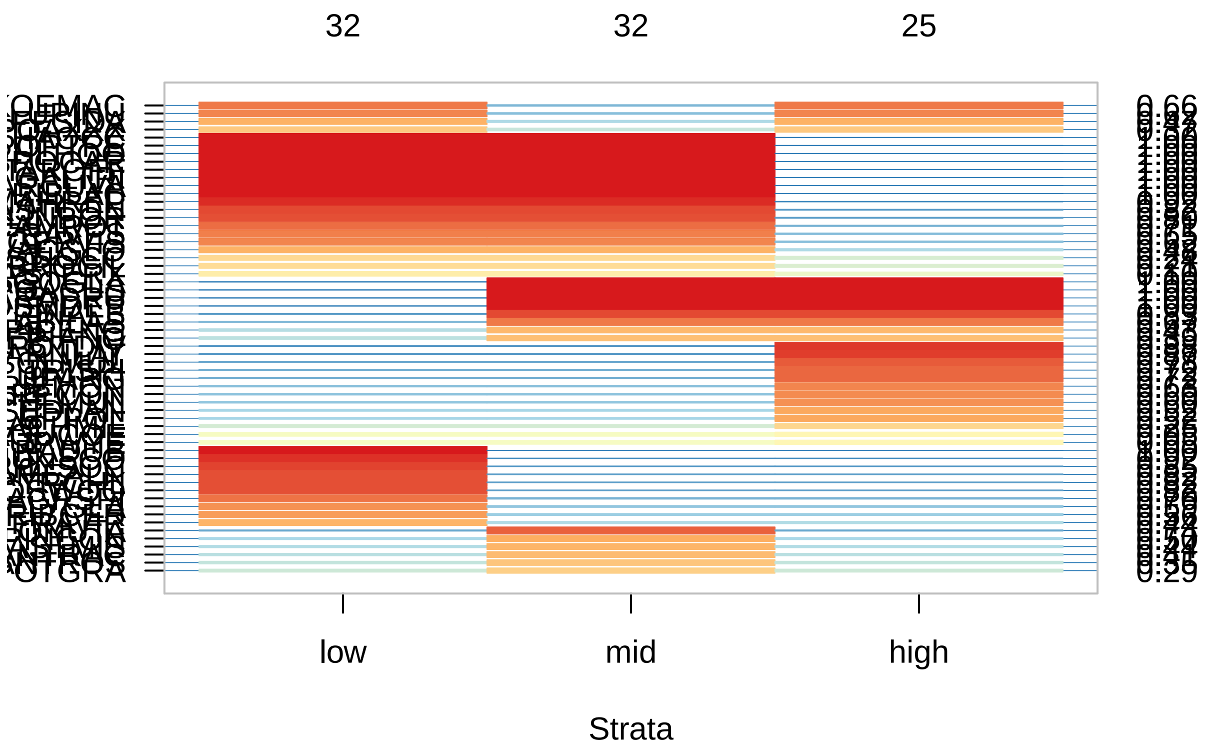
```
## [1] 150 368
```

```
dim(sveg1)
```

```
## [1] 135  85
```

```
sveg2 <- sveg[!ss, colnames(sveg1)]
elev2 <- elev[!ss]


o <- opticut(sveg1 ~ 1, strata=elev1, dist="binomial")
plot(o, sort=1)
```



```
co <- calibrate(o, sveg2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1275
##    Unobserved stochastic nodes: 100
##    Total graph size: 7586
##
## Initializing model
```

```
data.frame(predicted=co$gnew,
    truth=elev2,
    OK=co$gnew == elev2,
    round(co$pi,2))
```

```
##            predicted truth    OK  low  mid high
## X1V90L019      high   mid FALSE 0.00 0.01 0.99
## X1V90P002       low   low  TRUE 0.89 0.11 0.00
## X1V90L031       low   low  TRUE 0.95 0.05 0.00
## X1V891031       low   low  TRUE 0.99 0.01 0.00
## X5V96I296      high  high  TRUE 0.00 0.00 1.00
## X5V96M122      high  high  TRUE 0.00 0.00 1.00
## X4V93R081       low   mid FALSE 1.00 0.00 0.00
## X1V901051      high  high  TRUE 0.00 0.00 1.00
## X1V891022       mid   mid  TRUE 0.01 0.99 0.00
## X4V89P048       mid  high FALSE 0.02 0.93 0.04
## X4V95S079       mid   mid  TRUE 0.01 0.99 0.00
## X2V95S211      high  high  TRUE 0.00 0.00 1.00
## X1V90L024      high  high  TRUE 0.00 0.00 1.00
## X5V96K201      high  high  TRUE 0.00 0.00 1.00
## X5V96I014       mid   mid  TRUE 0.04 0.96 0.00
```

```
sum(co$gnew == elev2) / nrow(sveg2)
```

```
## [1] 0.8
```

## 3.1 Conclusions

- More species, merrier results (see simple simulation)
- Binary partitions might not be the best, especially for count models
- Multinomial models (K levels in g) are more eficient (see dolina example)
- Many species in binomial context work quite well, if K is not unreasonably high.

## 3.2 Todo

- OK use notation: j=1…p & r=1…S
- OK implement test cases for lm/glm
- OK write internal function (.calibrate)
- OK implement user interface: calibrate, calibrate.opticut(object, ynew, xnew, …)
- add NB, beta, ZI cases (ZI: inits!) [drop all the 1-species models!]
- gaussian needs sig^2