Package ‘dga’

May 10, 2021

Type Package

Title Capture-Recapture Estimation using Bayesian Model Averaging

Version 2.0.1

Date 2021-05-04

Description Performs Bayesian model averaging for capture-recapture. This includes code to stratify records, check the strata for suitable overlap to be used for capture-recapture, and some functions to plot the estimated population size.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation yes

ByteCompile yes

Imports chron, Rcpp

RoxygenNote 7.1.1

LinkingTo Rcpp, RcppArmadillo

Suggests testthat

Author James Johndrow [aut],
Kristian Lum [aut],
Patrick Ball [aut],
Olivier Binette [ctb, cre]

Maintainer Olivier Binette <olivier.binette@gmail.com>

Repository CRAN

Date/Publication 2021-05-10 16:22:11 UTC

R topics documented:

`dga-package` *Capture-Recapture Estimation using Bayesian Model Averaging*

Description

Performs Bayesian model averaging over all decomposable graphical models, each representing a different list dependence structure, with the goal of estimating the number of uncounted elements from the universe of elements that could have been recorded or “captured". The code here is an implementation of the model described in Madigan and York (1997).

Details

Package:	dga
Type:	Package
Version:	1.2
Date:	2015-04-16
License:	GPL >=2

Author(s)

James Johndrow <`james.johndrow@gmail.com`>, Kristian Lum <`kl@hrdag.org`>, and Patrick Ball <`pball@hrdag.org`>

Maintainer: Kristian Lum <`kl@hrdag.org`>

References

Madigan, David, and Jeremy C. York. "Bayesian methods for estimation of the size of a closed population." Biometrika 84.1 (1997): 19-31.

Examples

```
### This simulated example goes through the whole process for 3 lists.###
library(chron)
# Simulate some data
N <- 1000 # true population size
num.lists <- 3 # number of lists

# simulate 3 lists independently 'capturing'
l1 <- rbinom(N, 1, .2)
l2 <- rbinom(N, 1, .25)
l3 <- rbinom(N, 1, .3)

overlaps <- data.frame(l1, l2, l3)

# simulate dates of recording
dates <- paste(
  rep(2015, N), "-", sample(1:4, N, replace = TRUE), "-",
  sample(1:28, N, replace = TRUE)
)
dates <- chron(dates, format = c(dates = "y-m-d"))

# save true number in each stratum for comparison later
truth <- table((months(dates)))[1:4]

# remove dates of unrecorded elements
dates <- dates[apply(overlaps, 1, sum) > 0]

# remove individuals not recorded on any list
overlaps <- overlaps[apply(overlaps, 1, sum) > 0, ]

# stratify by date
strata <- make.strata(overlaps, dates = dates, date.defs = "monthly")

# check to make sure that all strata are OK
check <- check.strata(strata)

# look at strata, just to make sure
par(mfrow = c(2, 2), mar = rep(1, 4))
for (i in 1:nrow(strata$overlap.counts)) {
  venn3(strata$overlap.counts[i, ],
    main = rownames(strata$overlap.counts)[i],
    cex.main = 1
  )
}
```

```
# load the graphs to make the estimates
data(graphs3)

# select expansion factor defining the largest number of unrecorded elements.
# this makes Nmissing <- 0:(sum(Y)*fac)
fac <- 5

# set prior
delta <- 1 / 2^num.lists

# loop over strata to calculate posterior distributions of
# the total population size for each stratum
par(mfrow = c(2, 2), mar = rep(1.75, 4))
# if using Rstudio, make sure your plot window is pretty big here!
for (i in 1:nrow(strata$overlap.counts)) {
  Nmissing <- 0:(sum(strata$overlap.counts[i, ]) * fac)
  Y <- array(strata$overlap.counts[i, ], dim = rep(2, num.lists))
  weights <- bma.cr(Y, Nmissing, delta, graphs3)
  plotPosteriorN(weights, Nmissing + sum(strata$overlap.counts[i, ]),
    main = rownames(strata$overlap.counts)[i]
  )
  points(truth[i], .5 * max(weights), col = "red", pch = 16, cex = 2)
}
```

---

bma.cr *Bayesiam Model Averaging for Capture-Recapture*

Description

This function averages over all decomposable graphical models for p lists.

Usage

```
bma.cr(
  Y,
  Nmissing,
  delta,
  graphs,
  logprior = NULL,
  log.prior.model.weights = NULL
)
```

Arguments

Y	a 2^p array of list intersection counts. See details.
Nmissing	A vector of all possible values for the number of individuals that appear on no list.

`delta` The hyper-parameter for the hyper-Dirichlet prior distribution on list intersection probabilities. A smaller value indicates fewer prior observations per cell. A suggested default is `2^-p`

`graphs` A pre-computed list of all decomposable graphical models for `p` lists. These should be loaded using data(graphsp); see example. Currently, this package includes a list of graphs for three, four, or five lists.

`logprior` The log of the prior probability of each value in Nmissing. If left blank, this will default to the -log(Nmissing).

`log.prior.model.weights` Prior weights on the graphs. This should be a vector of length length(graphs).

Details

This is the main function in this package. It performs capture-recapture (or multiple systems estimation) using Bayesian model averaging as outlined in Madigan and York (1997).

Y can be created by the array() command from a vector that is ordered lexigraphically by the cell names, e.g., c(x000, x001, x010, x011, x100, x101, x110, x111).

Value

This function returns a matrix of weights, where rows correspond to models and columns correspond to values of Nmissing. Thus, the `ij`th entry of the matrix is the posterior probability of the `i`th model and the `j`th entry of Nmissing. Row sums return posterior probabilities by model.Column sums return posterior probabilities by value of Nmissing.

Note

This function is pretty robust relative to the more common log-linear model approach to capture-recapture. It will not fail (or issue a numerical warning) even if there are no overlaps among the lists. The user should take care that there is adequate list overlap and that there are sufficient cases in the stratum.

Author(s)

James Johndrow <`james.johndrow@gmail.com`> and Kristian Lum (kl@hrdag.org)

References

Madigan, David, and Jeremy C. York. "Bayesian methods for estimation of the size of a closed population." Biometrika 84.1 (1997): 19-31.

Examples

```
#### 5 list example from M & Y ##########
delta <- .5
Y <- c(0, 27, 37, 19, 4, 4, 1, 1, 97, 22, 37, 25, 2, 1, 3, 5,
       83, 36, 34, 18, 3, 5, 0, 2, 30, 5, 23, 8, 0, 3, 0, 2)
Y <- array(Y, dim = c(2, 2, 2, 2, 2))
```

```
Nmissing <- 1:300
N <- Nmissing + sum(Y)
data(graphs5)
weights <- bma.cr(Y, Nmissing, delta, graphs5)
plotPosteriorN(weights, N)

##### 3 list example from M & Y #######
Y <- c(0, 60, 49, 4, 247, 112, 142, 12)
Y <- array(Y, dim = c(2, 2, 2))

delta <- 1
a <- 13.14
b <- 55.17


Nmissing <- 1:300
N <- Nmissing + sum(Y)

logprior <- N * log(b) - (N + a) * log(1 + b) + lgamma(N + a) - lgamma(N + 1) - lgamma(a)

data(graphs3)
weights <- bma.cr(Y, Nmissing, delta, graphs3, logprior)
plotPosteriorN(weights, N)
```

`cfunction` *A Helper Function for make.strata*

Description

A helper function used in make.strata to make list overlap counts.

Usage

```
cfunction(x, nlist)
```

Arguments

`x`	capture histories, transformed from binary to decimal
`nlist`	the number of lists

Value

a table of the number of records with each capture history

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
## The function is currently defined as
cfunction <- function(x, nlist) {
  out <- table(c(x, 0:(2^nlist - 1))) - 1
}
```

`check.strata` *Checks Each Stratum for Suitability for Capture-Recapture*

Description

Takes in list intersection counts and source list totals as produced by make.strata. It then checks whether there are between three and five lists, whether all of the lists are non-empty, and whether all of the lists overlap with some other list.

Usage

```
check.strata(strata)
```

Arguments

`strata`	A list of list overlaps and source countsin the format of the output of make.strata. list.overlaps contains a data frame of list overlaps by stratum. source.counts contains the number of records by source and stratum.

Value

A boolean indicating whether any serious problems have been found with the strata.

Note

This does not issue a warning for cases where some subset of lists is not connected to the others, e.g. Lists A and B have overlap with each other, lists C and D have overlap with each other, but no records from A or B overlap with lists C or D. We suggest that you examine the list intersection counts manually as well.

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
library(chron)

N <- 1000
overlaps <- data.frame(l1 = rbinom(N, 1, .5), l2 = rbinom(N, 1, .5), l3 = rbinom(N, 1, .5))
dates <- paste(
  rep(2015, N), "-", sample(1:12, N, replace = TRUE), "-",
  sample(1:28, N, replace = TRUE)
)
dates <- chron(dates, format = c(dates = "y-m-d"))
locations <- sample(c("A", "B", "C", "D"), N, replace = TRUE)

# Aggregate only by week:
strata <- make.strata(overlaps, dates, date.def = "weekly")
check <- check.strata(strata)
```

---

circle *A Helper Function Used in venn3*

Description

Takes the parameters of a circle and returns points on its perimeter to be plotted to make circles for a venn diagram.

Usage

```
circle(x, y, r)
```

Arguments

x	the x coordinate of the center of the circle.
y	the y coordinate of the center of the circle.
r	the radius of the circle

Value

inds	the x,y coordinates of the periphery of a circle, to be used in venn3.

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
plot(dga:::circle(0, 0, 1), type = "l")
```

circle.ind *A Helper Function for venn3*

Description

Used in venn3 to tell whether proposed points are inside of the given circle.

Usage

```
circle.ind(ps, x, y, r)
```

Arguments

`ps`	a n x 2 matrix of coordinates.
`x`	the x coordinate of the center of the circle.
`y`	the y coordinate of the center of the circle.
`r`	the radius of the circle.

Value

a length n vector telling whether each row of ps is inside the given circle.

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
ps <- cbind(runif(100), runif(100))
plot(dga:::circle(0, 0, 1), type = "l")
inds <- dga:::circle.ind(ps, 0, 0, 1)
points(inds)
```

CompLogML — *Computes Marginal Likelihoods for Each Clique and Value of Nmissing*

Description

Assembles all of the pieces of the marginal likelihoods to be used to calculate the posterior probability of each model/value of Nmissing.

Usage

```
CompLogML(D, Nmissing, delta)
```

Arguments

D	A marginal array of the list overlap counts.
Nmissing	The vector of possible values for the missing cell.
delta	The prior hyper parameter for the Dirichlet distribution.

Value

The log marginal likelihood of the marginal table.

Author(s)

James Johndrow <james.johndrow@gmail.com> and Kristian Lum <kl@hrdag.org>

References

Madigan, David, and Jeremy C. York. "Bayesian methods for estimation of the size of a closed population." Biometrika 84.1 (1997): 19-31.

Examples

```
Y <- c(0, 27, 37, 19, 4, 4, 1, 1, 97, 22, 37, 25, 2, 1, 3, 5,
       83, 36, 34, 18, 3, 5, 0, 2, 30, 5, 23, 8, 0, 3, 0, 2)
Y <- array(Y, dim = c(2, 2, 2, 2, 2))

# Compute marginal array over lists 1 and 3
D <- apply(Y, c(1, 3), sum)

dga:::CompLogML(D, 1:300, 0.5)
```

`ellipse` *A Helper Function Used by Venn4 to Define the Perimeter of an Ellipse*

Description

Draws the ellipses used in venn4.

Usage

```
ellipse(x, y, a, b, alpha)
```

Arguments

`x`	the x coordinate of the center of the ellipse.
`y`	the y coordinate of the center of the ellipse.
`a`	the x-direction radius.
`b`	the y-direction radius.
`alpha`	the angle of rotation of the ellipse

Value

points that define the perimeter of an ellipse.

Author(s)

Kristian Lum <`kl@hrdag.org`>

Examples

```
plot(dga:::ellipse(0, 0, .5, .2, 1))
```

`ellipse.ind` *A Helper Function Used by Venn4*

Description

Takes potential points to be plotted in the venn diagrams and returns whether the point is inside or outside of the ellipse described by x, y, a, b, and alpha.

Usage

```
ellipse.ind(ps, x, y, a, b, alpha)
```

Arguments

`ps`	a n x 2 matrix of coordinates.
`x`	the x coordinate of the center of the ellipse.
`y`	the y coordinate of the center of the ellipse.
`a`	the x-radius of the ellipse.
`b`	the y-radius of the ellipse.
`alpha`	the angle of rotation of the ellipse

Value

a length n vector indicating whether each point is inside the ellipse.

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
## The function is currently defined as
ps <- cbind(runif(100), runif(100))
plot(dga:::ellipse(0, 0, .5, .3, 0), type = "l")
inds <- dga:::ellipse.ind(ps, 0, 0, .5, .3, 0)
points(inds)
```

---

graphs3 — *All Decomposable Graphical Models on Three Lists*

Description

This dataset contains all of the cliques and separators for each of the decomposable graphical models on three lists. On three lists, this is all of the models.

Usage

```
data(graphs3)
```

Format

A list of lists. graphs3[[i]] is the ith model under consideration. This consists of graphs3[[i]]$C, all of the cliques in that model, and graphs3[[i]]$S, the separators.

graphs4 — *All Decomposable Graphical Models on Four Lists*

Description

This dataset contains all of the cliques and separators for each of the decomposable graphical models on four lists.

Usage

```
data(graphs4)
```

Format

A list of lists. graphs4[[i]] is the `i`th model under consideration. This consists of graphs4[[i]]$C, all of the cliques in that model, and graphs4[[i]]$S, the separators.

graphs5 — *All Decomposable Graphical Models on Five Lists*

Description

This dataset contains all of the cliques and separators for each of the decomposable graphical models on five lists.

Usage

```
data(graphs5)
```

Format

A list of lists. graphs5[[i]] is the `i`th model under consideration. This consists of graphs5[[i]]$C, all of the cliques in that model, and graphs5[[i]]$S, the separators.

integer.base.b *Base Converter*

Description

Takes a decimal number and converts it to base b.

Usage

```
integer.base.b(x, b = 2)
```

Arguments

x	A number.
b	The desired base.

Details

This was harvested from the internet here: https://stat.ethz.ch/pipermail/r-help/2003-September/038978.html. Posted by Spencer Graves.

Value

A number in base b.

Author(s)

Spencer Graves

References

https://stat.ethz.ch/pipermail/r-help/2003-September/038978.html

make.strata *Transforms Records to List Intersection Counts by Stratum*

Description

Helps you to create list overlaps in the correct order to be used in bma.cr. This function also does some of the heavy lifting to stratify records by time (date, etc.) and other variables.

Usage

```
make.strata(
  overlaps,
  dates = NULL,
  locations = NULL,
  demographics = NULL,
  date.defs = "monthly",
  loc.defs = NULL,
  demog.defs = NULL,
  start.date = NULL,
  end.date = NULL
)
```

Arguments

`overlaps`	a data frame that tells whether the i'th record appears on the j'th lists, where n is the total number of sampled elements and p is the number of lists. For example, if the [3,2] entry is 1, then the third element appeared on the second list. If it is zero, then the third element did NOT appear on the second list.
`dates`	record dates, in identical row oder to overlaps. This must be a chron object. Do not include this if you don't want to stratify by time.
`locations`	record locations, though unlike the dates, there is nothing special about the type that would prevent you from using any other variable type to stratify by here. Do not include this unless you want to stratify by the factor you include here.
`demographics`	record demographic variables. Like locations, there is nothing specific to this that requires this be demographic. This should be a factor. Do not incude this unless you want to stratify by this factor.
`date.defs`	how you'd like to stratify by date. This defaults to "monthly". Other options are "weekly", "daily", and "yearly". If you enter an integer (k) instead of one of these options, the data will be stratified into blocks of size k days.
`loc.defs`	How to divide up all of the levels of locations into groups. e.g. if locations has levels A, B, and C, and you'd like to stratify so that A and B are one strata and C is another, input loc.defs = list(g1 = c('A', 'B'), g2 = c('C')). If this is left as NULL, each level will be put into its own stratum.
`demog.defs`	Similar to loc.defs. Same format. Including both just allows you to stratify along two dimensions.
`start.date`	A chron object of one date. This gives the date of earliest record we want to include. If NULL, this defaults to the earliest record in the dataset.
`end.date`	a chron object of one date. This gives the date of the latest record to be included. If NULL, this defaults to the latest record in the dataset. This can only be included if dates are given.

Value

`overlap.counts`	a data frame where each row gives the list intersection counts that can be used in bma.cr

source.counts a data frame that gives the total number of records by each data source and stratum.

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
require(chron)
N <- 10000
overlaps <- data.frame(l1 = rbinom(N, 1, .5), l2 = rbinom(N, 1, .5), l3 = rbinom(N, 1, .5))
dates <- paste(
  rep(2015, N), "-", sample(1:12, N, replace = TRUE), "-",
  sample(1:28, N, replace = TRUE)
)
dates <- chron(dates, format = c(dates = "y-m-d"))
locations <- sample(c("A", "B", "C", "D"), N, replace = TRUE)

# Aggregate only by week:
make.strata(overlaps, dates, date.def = "weekly")

# Aggregate by year and location, where locations are not grouped:
make.strata(overlaps, dates, date.def = "yearly", locations)

# Aggregate by 2 day increments and location, where there are unique location levels
#       A, B, C, and D and locations A and B are in group 1
#       and locations C and D are in group 2.
loc.defs <- list("g1" = c("A", "B"), "g2" = c("C", "D"))
make.strata(overlaps, dates, date.def = 2, locations, loc.defs = loc.defs)
# Aggregate by demographic (sex) only, where sex takes values M, F, A, NA, and U
#       and we would like to group these as M, F, and other.
sex <- sample(c("M", "F", "A", NA, "U"),
  prob = c(.4, .4, .1, .05, .05),
  N, replace = TRUE
)
demog.defs <- list("M" = "M", "F" = "F", "Other" = c("A", NA, "U"))
make.strata(overlaps, demographics = sex, demog.defs = demog.defs)
```

MakeCompMatrix *Component-wise Matrix of Log Marginal Likelihoods*

Description

Calls CompLogML to create a matrix of number of possible components by length(Nmissing) log marginal likelihoods. Calculates the log marginal likehood of each possible marginal table for every value of Nmissing.

Usage

```
MakeCompMatrix(p, delta, Y, Nmissing)
```

Arguments

`p`	Number of lists
`delta`	Prior hyperparameter of the Dirichlet distribution.
`Y`	The `2^k` matrix of list intersection counts.
`Nmissing`	The vector of possible values for the missing cell.

Value

A matrix of log marginal likelihoods.

Author(s)

James Johndrow `<james.johndrow@gmail.com>` and Kristian Lum `<kl@hrdag.org>`

`plotPosteriorN` *Plots Posterior Distribution of Nmissing*

Description

Plots the model averaged posterior distribution of the total number of elements (the solid line) and the contribution to the posterior of each of the models (dotted lines)

Usage

```
plotPosteriorN(weights, N, main = NULL)
```

Arguments

`weights`	The output of BMAfunction.
`N`	N + Nmissing. Or, if you prefer, just Nmissing. The former shows the posterior distribution of the total population size; the latter shows the posterior distribution of the number of missing elements.
`main`	the title of the plot

Value

A plot.

Author(s)

Kristian Lum `<kl@hrdag.org>`

Examples

```
##### 5 list example from M & Y #######

delta <- .5
Y <- c(0, 27, 37, 19, 4, 4, 1, 1, 97, 22, 37, 25, 2, 1, 3, 5,
       83, 36, 34, 18, 3, 5, 0, 2, 30, 5, 23, 8, 0, 3, 0, 2)
Y <- array(Y, dim = c(2, 2, 2, 2, 2))
Nmissing <- 1:300
N <- Nmissing + sum(Y)
data(graphs5)
weights <- bma.cr(Y, Nmissing, delta, graphs5)
plotPosteriorN(weights, N)


##### 3 list example from M & Y #######
Y <- c(0, 60, 49, 4, 247, 112, 142, 12)
Y <- array(Y, dim = c(2, 2, 2))

delta <- 1
a <- 13.14
b <- 55.17


Nmissing <- 1:300
N <- Nmissing + sum(Y)

logprior <- N * log(b) - (N + a) * log(1 + b) + lgamma(N + a) - lgamma(N + 1) - lgamma(a)

data(graphs3)
weights <- bma.cr(Y, Nmissing, delta, graphs3, logprior)
plotPosteriorN(weights, N)
```

`remove.close` *A Helper Function to Tell Which Points Are Near the Boundary of a Circle*

Description

Used in venn3 to tell which of the potential points to be plotted are near the boundary of the circle defned by x, y, and r.

Usage

```
remove.close(ps, x, y, r)
```

Arguments

`ps`	an n x 2 matrix of potential points.

x	the x coordinate of the center of the circle.
y	the y coordinate of the center of the circle.
r	the radius of the circle

Value

inds	tells which points are too close to the edge of the circle.

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
ps <- cbind(runif(100), runif(100))
inds <- dga:::remove.close(ps, .5, .5, .1)
```

---

remove.close.ellipse *A Helper Function to Tell Which Points are Near the Boundary of the Ellipse*

Description

A helper function.

Usage

```
remove.close.ellipse(ps, x, y, a, b, alpha)
```

Arguments

ps	an n x 2 matrix of potential points.
x	the x coordinate of the center of the ellipse.
y	the y coordinate of the center of the ellipse.
a	the x-radius of the ellipse.
b	the y-radius of the ellipse.
alpha	the angle of rotation of the ellipse.

Value

inds	a vector of length nrow(ps) that tells whether each row of ps is near the border of the ellipse defined by x,y,a,b, and alpha.

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
## The function is currently defined as
ps <- cbind(runif(100), runif(100))
inds <- dga:::remove.close.ellipse(ps, .5, .5, .1, .3, 1)
```

---

sfunction *A Helper Function for make.strata.*

Description

This is the simplest function ever. It's just an apply to sum across columns.

Usage

```
sfunction(x)
```

Arguments

x	capture histories, as numbers

Value

apply(x, 2, sum)

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
## The function is currently defined as
sfunction <- function(x) {
  out <- apply(x, 2, sum)
}
```

venn3 *Three List Venn Diagram*

Description

A function that plots a venn diagram of 3 lists. One point is plotted in each region for each record that falls into the corresponding list overlap.

Usage

```
venn3(
  overlap.counts,
  main = NULL,
  num.test.points = 1e+05,
  p.cex = 0.75,
  write_numbers = FALSE,
  t.cex = 1.25,
  cex.main = 1
)
```

Arguments

`overlap.counts`	A vector of length `2^3` that gives the number of records in each overlap in lexicographic order, i.e. 001, 010, 011, 100, etc.
`main`	the title of the graph
`num.test.points`	how many test points to generate as potentials to be plotted in the circles.
`p.cex`	the size of the points to be plotted
`write_numbers`	indicates whether to print the number of points in each region.
`t.cex`	the size of the text to write the numbers.
`cex.main`	the size of the title

Value

a 3-way venn diagram with points inside of each segment representing the number of records on each list overlap.

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
overlap.counts <- rpois(8, 30)
venn3(overlap.counts, main = "example diagram")
```

venn4 *Four List Venn Diagram*

Description

A function that plots a venn diagram of 4 lists. One point is plotted in each region for each record that falls into the corresponding list overlap.

Usage

```
venn4(
  overlap.counts,
  main = NULL,
  num.test.points = 1e+05,
  p.cex = 0.75,
  cex.main = 1
)
```

Arguments

`overlap.counts`	A vector of length `2^4` that gives the number of records in each overlap in lexicographic order, i.e. 0000, 0001, 0010, 0011, 0100, etc.
`main`	the title of the graph
`num.test.points`	how many test points to generate as potentials to be plotted in the circles.
`p.cex`	the size of the points to be plotted
`cex.main`	the size of the title

Value

A venn diagram of the list overlap structure for four lists. Each region of the plot contains points representing each record in that list intersection.

Author(s)

Kristian Lum <kl@hrdag.org>

Examples

```
overlap.counts <- rpois(16, 50)
venn4(overlap.counts, main = "example diagram")
```

Index