Package ‘detect’

March 3, 2016

Type    Package
Title   Analyzing Wildlife Data with Detection Error
Version 0.4-0
Date    2016-03-02
Author  Peter Solymos [cre, aut], Monica Moreno [aut], Subhash R. Lele [aut]
Maintainer Peter Solymos <solymos@ualberta.ca>
Depends R (>= 2.13.0), Formula, stats4, pbapply
Suggests dclone, dcmle
Description Models for analyzing site occupancy and count data models with detection error, including single-visit based models, conditional distance sampling and time-removal models. Package development was supported by the Alberta Biodiversity Monitoring Institute (www.abmi.ca) and the Boreal Avian Modelling Project (borealbirds.ca).
License GPL-2
URL https://github.com/psolymos/detect
BugReports https://github.com/psolymos/detect/issues
LazyLoad yes
LazyData true
NeedsCompilation no
Repository CRAN
Date/Publication 2016-03-03 08:25:21

R topics documented:

detect-package .................................................. 2
AUC ............................................................... 3
bootstrap ....................................................... 3
cmulti ........................................................... 4
convertEDT .................................................... 8
Models for analyzing site occupancy and count data models with detection error, including single-visit based models, conditional distance sampling and time-removal models. Package development was supported by the Alberta Biodiversity Monitoring Institute (www.abmi.ca) and the Boreal Avian Modelling Project (borealbirds.ca).

Details

svocc: single visit occupancy model (Lele et al. 2011, Moreno et al. 2010).

svabu: single visit abundance model based on conditional maximum likelihood (Solymos et al. 2012, Solymos and Lele 2016).

cmulti: conditional multinomial maximum likelihood estimation for removal and (point count) distance sampling, efficient and flexible setup for varying methodologies (Solymos et al. 2013).

Author(s)

Peter Solymos, Monica Moreno, Subhash R Lele
Maintainer: Peter Solymos <solymos@ualberta.ca>

References


Supporting info, including a tutorial for the above paper: http://dcr.r-forge.r-project.org/qpad/


---

### AUC

**AUC ROC plot for fitted models**

**Description**

Area under the receiver-operator (ROC) curve (AUC), and ROC plot methods for fitted models.

**Usage**

```r
AUC(object, ...)  
rocplot(x, ...)
```

**Arguments**

- `object`: a fitted model object
- `x`: a fitted model object
- `...`: other arguments

**Value**

- `AUC` returns AUC value for a model, or a data frame with values for more models.
- `rocplot` returns the values used for the plot invisibly, and as a side effect it draws a graph.

**Author(s)**

Peter Solymos and Monica Moreno

---

### bootstrap

*Do bootstrap and extract bootstrap results*

**Description**

Do bootstrap and extract bootstrap results.

**Usage**

```r
bootstrap(object, ...)  
exttractBOOT(object, ...)
```
Arguments

object a fitted model object
... other arguments

Value

bootstrap performs bootstrap.
extractboot is used to extract bootstrap results.

Author(s)

Peter Solymos

cmulti

Description

Conditional Multinomial Maximum Likelihood Estimation for different sampling methodologies.

Usage

cmulti(formula, data, type = c("rem", "mix", "dis"),
inits = NULL, method = "Nelder-Mead", ...)
cmulti.fit(Y, D, X=NULL, type=c("rem", "mix", "dis"),
inits=NULL, method="Nelder-Mead", ...)
cmulti2.fit(Y, D1, D2, X1=NULL, X2=NULL,
inits=NULL, method="Nelder-Mead", ...)
## S3 method for class 'cmulti'
fitted(object, ...)

Arguments

formula formula, LHS takes 2 matrices in the form of Y | D, RHS is either 1 or some covariates, see Examples.
data data.
type character, one of "rem" (removal sampling, homogeneous singing rates), "mix" (removal sampling, heterogeneous singing rates), "dis" (distance sampling, half-normal detection function for point counts, circular area)
Y this contains the cell counts. cmulti.fit requires that Y is a matrix (observations x intervals), dimensions and pattern in NAs must match that of D. cmulti2.fit requires that Y is a 3-dimensional array (observations x time intervals x distance intervals), dimensions and pattern in NAs must match that of D1 and D2.
D, D1, D2 design matrices, that describe the interval endpoints for the sampling methodology, dimensions must match dimensions of Y.
X, X1, X2  design matrices, X is the matrix with covariates for the removal/distance sampling parameters. X1 is the matrix with covariates for the removal, X2 is the matrix with covariates for the distance sampling parameters.

inits  optional initial values.

method  method for \texttt{optim}.

object  fitted model object.

...  additional options for \texttt{optim}.

Details

Conditional Multinomial Maximum Likelihood Estimation for different sampling methodologies.

Value

An object of class `cmulti`.

Author(s)

Peter Solymos

References


Supporting info, including a tutorial for the above paper: \url{http://dcr.r-forge.r-project.org/qpad/}

Examples

```r
simfun1 <- function(n = 10, phi = 0.1, c=1, tau=0.8, type="rem") {
  if (type=="dis") {
    Dparts <- matrix(c(0.5, 1, NA,
                      0.5, 1, Inf,
                      1, Inf, NA), 3, 3, byrow=TRUE)
    D <- Dparts[sample.int(3, n, replace=TRUE),]
    CP <- 1-exp(-D/tau)^2
  } else {
    Dparts <- matrix(c(5, 10, NA,
                       3, 5, 10,
                       3, 5, NA), 3, 3, byrow=TRUE)
    D <- Dparts[sample.int(3, n, replace=TRUE),]
    CP <- 1-exp(-D*phi)
  }
  k <- ncol(D)
  P <- CP - cbind(0, CP[, -k, drop=FALSE])
  Psum <- rowSums(P, na.rm=TRUE)
  PPSum <- P / Psum
  Pok <- !is.na(PPsum)
```
```r
N <- rpois(n, 10)
Y <- matrix(NA, ncol(PPPsum), nrow(PPPsum))
Ypre <- sapply(1:n, function(i) rmultinom(1, N, PPPsum[i, Pok[i,]]))
Y[t(Pok)] <- unlist(Ypre)
Y <- t(Y)
list(Y=Y, D=D)

n <- 200
x <- rnorm(n)
X <- cbind(1, x)

## removal, constant
vv <- simfun1(n=n, phi=exp(-1.5))
m1 <- cmulti(vv$Y | vv$D ~ 1, type="rem")
coef(m1)

## mixture, constant
vv <- simfun1(n=n, phi=exp(-1.5), c=plogis(0.8))
m2 <- cmulti(vv$Y | vv$D ~ 1, type="mix")
coef(m2)

## dist, constant
vv <- simfun1(n=n, tau=exp(-0.2), type="dis")
m3 <- cmulti(vv$Y | vv$D ~ 1, type="dis")
coef(m3)

## removal, not constant
log.phi <- X %*% c(-2,-1)
vv <- simfun1(n=n, phi=exp(cbind(log.phi, log.phi, log.phi)))
m1 <- cmulti(vv$Y | vv$D ~ x, type="rem")
coef(m1)

## mixture, not constant
logit.c <- X %*% c(-2,1)
vv <- simfun1(n=n, phi=exp(-1.5), c=plogis(cbind(logit.c, logit.c, logit.c)))
m2 <- cmulti(vv$Y | vv$D ~ x, type="mix")
coef(m2)

## dist, not constant
log.tau <- X %*% c(-0.5,-0.2)
vv <- simfun1(n=n, tau=exp(cbind(log.tau, log.tau, log.tau)), type="dis")
m3 <- cmulti(vv$Y | vv$D ~ x, type="dis")
coef(m3)

summary(m3)
coef(m3)
vcov(m3)
AIC(m3)
confint(m3)
logLik(m3)

## fitted values
plot(exp(log.tau), fitted(m3))

## joint removal-distance estimation
## is not different from 2 orthogonal estimations
```
```
simfun12 <- function(n = 10, phi = 0.1, c=1, tau=0.8, type="rem") {
  Flat <- function(x, DIM, dur=TRUE) {
    x <- array(x, DIM)
    if (!dur) {
      x <- aperm(x, c(1,3,2))
    }
    dim(x) <- c(DIM[1], DIM[2]*DIM[3])
    x
  }
  Dparts1 <- matrix(c(5, 10, NA,
                       3, 5, 10,
                       3, 5, NA), 3, 3, byrow=TRUE)
  D1 <- Dparts1[sample.int(3, n, replace=TRUE),]
  CP1 <- 1-c*exp(-D1*phi)
  Dparts2 <- matrix(c(0.5, 1, NA,
                       0.5, 1, Inf,
                       1, Inf, NA), 3, 3, byrow=TRUE)
  D2 <- Dparts2[sample.int(3, n, replace=TRUE),]
  CP2 <- 1-exp(-(D2/tau)^2)
  k1 <- ncol(D1)
  k2 <- ncol(D2)
  DIM <- c(n, k1, k2)
  P1 <- CP1 - cbind(0, CP1[, -k1, drop=FALSE])
  P2 <- CP2 - cbind(0, CP2[, -k2, drop=FALSE])
  Psum1 <- rowSums(P1, na.rm=TRUE)
  Psum2 <- rowSums(P2, na.rm=TRUE)
  Pflat <- Flat(P1, DIM, dur=TRUE) * Flat(P2, DIM, dur=FALSE)
  PsumFlat <- Psum1 * Psum2
  PPsumFlat <- Pflat / PsumFlat
  PokFlat <- !is.na(PPsumFlat)
  N <- rpois(n, 10)
  Yflat <- matrix(NA, ncol(PPsumFlat), nrow(PPsumFlat))
  Ypreflat <- sapply(1:n, function(i) rmultinom(1, N, PPsumFlat[i,PokFlat[i,]]))
  Yflat[t(PokFlat)] <- unlist(Ypreflat)
  Yflat <- t(Yflat)
  Y <- array(Yflat, DIM)
  k1 <- dim(Y)[2]
  k2 <- dim(Y)[3]
  Y1 <- t(sapply(1:n, function(i) {
    count <- rowSums(Y[i,,], na.rm=TRUE)
    nas <- rowSums(is.na(Y[i,,]))
    count[nas == k2] <- NA
    count
  }))
  Y2 <- t(sapply(1:n, function(i) {
    count <- colSums(Y[i,,], na.rm=TRUE)
    nas <- colSums(is.na(Y[i,,]))
    count[nas == k2] <- NA
    count
  }))
  list(Y=Y, D1=D1, D2=D2, Y1=Y1, Y2=Y2)
}
```
## convert EDT

### Description

Conversion between truncated and unlimited effective detection distance (EDR).

### Usage

```r
convertEDR(edr, r, truncated=FALSE)
```

### Arguments

- **edr**: effective detection distance. In same units as `r`.
- **r**: truncation distance (radius of point count). In same units as `edr`.
- **truncated**: logical, see Details.

### Details

- **truncated = FALSE** means that `edr` is unlimited EDR, and the function returns the truncated EDR given `r`.
- **truncated = TRUE** means that `edr` is truncated EDR given `r`, and the function returns the unlimited EDR.
Value

A numeric vector with converted EDR values.

Author(s)

Peter Solymos

References


Supporting info, including a tutorial for the above paper: [http://dcr.r-forge.r-project.org/qpad/](http://dcr.r-forge.r-project.org/qpad/)

Examples

```r
convertedr(1, 0.5, truncated=FALSE)
## should be close to 1
convertedr(convertedr(1, 0.5, truncated=FALSE), 0.5, truncated=TRUE)
```

---

**databu**

*Simulated example for abundance model*

Description

Simulated example for abundance model, see code below.

Usage

data(databu)

Format

A data frame with 1000 observations on the following 11 variables.

- `N` true counts
- `Y` observed counts
- `x1` random variables used as covariates
- `x2` random variables used as covariates
- `x3` random variables used as covariates
- `x4` random variables used as covariates
x5 random variables used as covariates
x6 random variables used as covariates
p probability of detection
lambda mean of the linear predictor
A occupancy
phi zero inflation probabilities

Details
This simulated example corresponds to the Binomial - ZIP model implemented in the function svabu.

Source
Simulated example.

References

Examples
```r
data(databu)
str(databu)
## Not run:
## simulation
n <- 1000
set.seed(1234)
x1 <- runif(n, 0, 1)
x2 <- rnorm(n, 0, 1)
x3 <- runif(n, -1, 1)
x4 <- runif(n, -1, 1)
x5 <- rbinom(n, 1, 0.6)
x6 <- rbinom(n, 1, 0.4)
x7 <- rnorm(n, 0, 1)
X <- model.matrix(~ x1 + x5)
Z <- model.matrix(~ x2 + x5)
Q <- model.matrix(~ x7)
beta <- c(2, -0.8, 0.5)
theta <- c(1, 2, -0.5)
phi <- 0.3
p <- drop(binomial("logit")$linkinv(Z * theta))
lambda <- drop(exp(X * beta))
A <- rbinom(n, 1, 1-phi)
N <- rpois(n, lambda * A)
Y <- rbinom(n, N, p)
databu <- data.frame(N=N, Y=Y, x1, x2, x3, x4, x5, x6, p=p, lambda=lambda, A, phi)

## End(Not run)
```
Simulated example for occupancy model

Description
Simulated example for occupancy model, see code below.

Usage
data(datocc)

Format
A data frame with 1000 observations on the following 6 variables.

- \( Y \) true occupancy
- \( W \) observations
- \( x_1 \) random variables used as covariates
- \( x_2 \) random variables used as covariates
- \( x_3 \) random variables used as covariates
- \( x_4 \) random variables used as covariates
- \( p.occ \) probability of occurrence
- \( p.det \) probability of detection

Details
This simulated example corresponds to the ZI Binomial model implemented in the function `svocc`.

Source
Simulated example.

References

Examples
```r
data(datocc)
str(datocc)
## Not run:
## simulation
n <- 1000
set.seed(1234)
x1 <- runif(n, -1, 1)
x2 <- as.factor(rbinom(n, 1, 0.5))```
hbootindex

Hierarchical bootstrap indices

Description

Generates hierarchical bootstrap indices.

Usage

hbootindex(groups, strata, B = 199)

Arguments

groups  group membership vector.
strata   strata, optional.
B        number of bootstrap iterations.

Details

Resampling with replacement with weights proportional to the number of observations in each of the group level (unique values in groups).

Values of groups within levels (unique values) of strata are resampled independently of other strata levels.

Value

A matrix with bootstrapped indices, number of columns is B + 1. The column is a resample without replacement (random subsets can be selected without further reshuffling). Other elements contain indices according to rules described in Details section (these also randomly reshuffled).

Author(s)

Peter Solymos
### Examples

```r
# equal group sizes
groups <- rep(1:4, each=5)
strata <- rep(1:2, each=10)
hbootindex(groups, strata, 3)

# unequal group sizes
groups <- groups[-c(5,9,10,11)]
strata <- strata[-c(5,9,10,11)]
hbootindex(groups, strata, 3)
```

---

**load_BAM_QPAD**  
*Load BAM QPAD parameter estimates and support functions*

**Description**

Load BAM QPAD parameter estimates and support functions.

**Usage**

```r
load_BAM_QPAD(version)
```

**Arguments**

- `version`  
  version of the BAM QPAD estimates. List of selection is provided if missing.

**Details**

The `load_BAM_QPAD` function in the 'detect' package is deprecated. Use the `load_BAM_QPAD` function 'QPAD' package instead. See [https://github.com/psolymos/QPAD](https://github.com/psolymos/QPAD) for more information.

**Value**

It returns a message.

**Author(s)**

Peter Solymos

**References**


Supporting info, including a tutorial for the above paper: [http://dcr.r-forge.r-project.org/qpad/](http://dcr.r-forge.r-project.org/qpad/)
**Ovenbird abundances**

**Description**

Ovenbird abundances from BBS

**Usage**

```r
data(oven)
```

**Format**

A data frame with 891 observations on the following 11 variables.

- `count` observations
- `route` route id
- `stop` stop id within route
- `pforest` proportion of forest
- `pdecid` proportion of deciduous forest
- `pagri` proportion of agricultural areas
- `long` longitude
- `lat` latitude
- `observ` observer, a factor with levels ARS DW RDW SVW
- `julian` Julian day
- `timeday` time of day

**Source**

BBS

**References**

Bayne unpublished

**Examples**

```r
data(oven)
str(oven)
```
**svabu**

*Single visit N-mixture abundance models*

**Description**

Binomial-Poisson, Binomial-NegBin, Binomial-ZIP, and BinomialZINB models with single visit.

**Usage**

```r
svabu(formula, data, zeroinfl = TRUE, area = 1, N.max = NULL,
       inits, link.det = "logit", link.zif = "logit",
       model = TRUE, x = FALSE, distr = c("P", "NB"), ...)

svabu.fit(Y, X, Z, Q = NULL, zeroinfl = TRUE, area = 1, N.max = NULL,
           inits, link.det = "logit", link.zif = "logit", ...)
svabu_nb.fit(Y, X, Z, Q = NULL, zeroinfl = TRUE, area = 1, N.max = NULL,
             inits, link.det = "logit", link.zif = "logit", ...)

zif(x)
is.present(object, ...)
predictMCMC(object, ...)
svabu.step(object, model, trace = 1, steps = 1000,
           criter = c("AIC", "BIC"), test = FALSE, k = 2, control, ...)
```

**Arguments**

- `formula`: formula of the form `y ~ x | z`, where `y` is a vector of observations, `x` is the set of covariates for the occurrence model, `z` is the set of covariates for the detection model. `x` can further expanded as `x1 + zif(x2)` into terms for the nonzero count data part (`x1`) and the zero inflation component (`zif(x2)`) using the `zif` special.

- `Y, X, Z, Q`: vector of observation, design matrix for abundance model, design matrix for detection and design matrix for zero inflation model.

- `data`: data

- `area`: area

- `N.max`: maximum of true count values (for calculating the integral)

- `zeroinfl`: logical, if the Binomial-ZIP model should be fitted

- `inits`: initial values used by `link{optim}`

- `link.det, link.zif`: link function for the detection and zero inflation parts of the model

- `model`: a logical value indicating whether model frame should be included as a component of the returned value, or true state or detection model

- `x`: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For the function `zif` it is any object to be returned.
object  a fitted object.
trace  info returned during the procedure
steps  max number of steps
criter  criterion to be minimized (cAUC=1-AUC)
test  logical, if decrease in deviance should be tested
k  penalty to be used with AIC
control  controls for optimization, if missing taken from object
distr  character, abundance distribution: "P" for Poisson, "NB" for Negative Binomial.
...  other arguments passed to the functions

Details
See Examples.

The right hand side of the formula must contain at least one continuous (i.e. non discrete/categorical) covariate. This is the necessary condition for the single-visit method to be valid and parameters to be identifiable. See References for more detailed description.

The Binomial-Poisson model is the single visit special case of the N-mixture model proposed by Royle (2004) and explained in Solymos et a. (2012) and Solymos and Lele (2016).

Value
An object of class `svabu`.

Author(s)
Peter Solymos and Subhash Lele

References

Examples

data(databu)

```r
## fit BZIP and BP models
m00 <- svabu(Y ~ x1 + x5 | x2 + x5, databu[1:200,])

## print method
m00
```
## svabu

```
## summary: CMLE
summary(m00)
## coef
coeff(m00)
coeff(m00, model="sta") ## state (abundance)
coeff(m00, model="det") ## detection
coeff(m00, model="zif") ## zero inflation (this is part of the 'true state'!)

## Not run:
## Diagnostics and model comparison

m01 <- svabu(Y ~ x1 + x5 | x2 + x5, databu[1:200,], zeroInfl=FALSE)
## compare estimates (note, zero inflation is on the logit scale!)
cbind(truth=c(2, -0.8, 0.5, 1.2, -0.5, plogis(0.3)),
"B-ZIP"=coeff(m00), "B-P"=c(coeff(m01), NA))

## fitted
plot(fitted(m00), fitted(m01))
abline(0,1)

## compare models
AIC(m00, m01)
BIC(m00, m01)
loglik(m00)
logLik(m01)
## diagnostic plot
plot(m00)
plot(m01)

## Bootstrap

## non parametric bootstrap
## - initial values are the estimates
m02 <- bootstrap(m00, B=25)
attr(m02, "bootstrap")
eXtractBOOT(m02)
summary(m02)
summary(m02, type="cmle")
summary(m02, type="boot")
## vcov
vcov(m02, type="cmle")
vcov(m02, type="boot")
vcov(m02, model="sta")
vcov(m02, model="det")
## confint
confint(m02, type="cmle") ## Wald-type
confint(m02, type="boot") ## quantile based
## parametric bootstrap
simulate(m00, 5)
m03 <- bootstrap(m00, B=5, type="param")
eXtractBOOT(m03)
summary(m03)
```
### Model selection

```r
m04 <- svabu(Y ~ x1 + x5 | x2 + x5 + x3, databu[1:200], phi.boot=0)
m05 <- drop1(m04, model="det")
m05
m06 <- svabu.step(m04, model="det")
summary(m06)
m07 <- update(m04, ~ . | . - x3)
m07
```

### Controls

```r
m00$control
getOption("detect.optim.control")
getOption("detect.optim.method")
options("detect.optim.method"="BFGS")
m08 <- svabu(Y ~ x1 + x5 | x2 + x5, databu[1:100])
m00$control ## but original optim method is retained during model selection and bootstrap
## fitted models can be used to provide initial values
options("detect.optim.method"="Nelder-Mead")
m09 <- svabu(Y ~ x1 + x5 | x2 + x5, databu[1:100], inits=coef(m08))
```

### Ovenbirds dataset

```r
data(oven)
oven <- ovenc
oven[, c(4:8,10:11)][] <- lapply(oven[, c(4:8,10:11)], scale)
moven <- svabu(count ~ pforest | observ + pforest + julian + timeday, ovenc)
summary(moven)
drop1(moven, model="det")
moven2 <- update(moven, ~ . | . - timeday)
summary(moven2)
moven3 <- update(moven2, ~ . | .., zeroinfl=FALSE)
summary(moven3)
BIC(moven, moven2, moven3)
```

### End(Not run)

---

**svocc** | **ZI Binomial model with single visit**

---

**Description**

ZI Binomial model with single visit

**Usage**

```r
svocc(formula, data, link.sta = "cloglog", link.det = "logit", penalized = FALSE,
method = c("optim", "de"), inits,
model = TRUE, x = FALSE, ...)
```
svocc.fit(Y, X, Z, link.sta = "cloglog", link.det = "logit", penalized = FALSE, auc = FALSE, method = c("optim", "dc"), inits, ...) svocc.step(object, model, trace = 1, steps = 1000, criter = c("AIC", "BIC", "cAUC"), test = FALSE, k = 2, control, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>formula of the form ( y \sim x \mid z ), where ( y ) is a vector of observations, ( x ) is the set of covariates for the occurrence model, ( z ) is the set of covariates for the detection model</td>
</tr>
<tr>
<td>Y, X, Z</td>
<td>vector of observation, design matrix for occurrence model, and design matrix for detection model</td>
</tr>
<tr>
<td>data</td>
<td>data</td>
</tr>
<tr>
<td>link.sta, link.det</td>
<td>link function for the occurrence (true state) and detection model</td>
</tr>
<tr>
<td>penalized</td>
<td>logical, if penalized likelihood estimate should be computed</td>
</tr>
<tr>
<td>method</td>
<td>optimization or data cloning to be used as optimization</td>
</tr>
<tr>
<td>inits</td>
<td>initial values</td>
</tr>
<tr>
<td>model</td>
<td>a logical value indicating whether model frame should be included as a component of the returned value, or true state or detection model</td>
</tr>
<tr>
<td>x</td>
<td>logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value</td>
</tr>
<tr>
<td>auc</td>
<td>logical, if AUC should be calculated</td>
</tr>
<tr>
<td>object</td>
<td>a fitted model object</td>
</tr>
<tr>
<td>trace</td>
<td>info returned during the procedure</td>
</tr>
<tr>
<td>steps</td>
<td>max number of steps</td>
</tr>
<tr>
<td>criter</td>
<td>criterion to be minimized (cAUC=1-AUC)</td>
</tr>
<tr>
<td>test</td>
<td>logical, if decrease in deviance should be tested</td>
</tr>
<tr>
<td>k</td>
<td>penalty to be used with AIC</td>
</tr>
<tr>
<td>control</td>
<td>controls for optimization, if missing taken from object</td>
</tr>
<tr>
<td>...</td>
<td>other arguments passed to the functions</td>
</tr>
</tbody>
</table>

Details

See Examples.

The right hand side of the formula must contain at least one continuous (i.e. non discrete/categorical) covariate. This is the necessary condition for the single-visit method to be valid and parameters to be identifiable. See References for more detailed description.

Value

An object of class 'svocc'.
Author(s)

Peter Solymos and Monica Moreno

References


Examples

data(datocc)

```r
## MLE
m00 <- svocc(w ~ x1 | x1 + x3, datocc)
## PMLE
m01 <- svocc(w ~ x1 | x1 + x3, datocc, penalized=TRUE)

## print
m00
## summary
summary(m00)
## coefficients
coef(m00)
## state (occupancy) model estimates
coef(m00, "sta")
## detection model estimates
coef(m00, "det")
## compare estimates
cbind(truth=c(0.6, 0.5, 0.4, -0.5, 0.3),
mle=coef(m00), pmle=coef(m01))

## AIC, BIC
AIC(m00)
BIC(m00)
## log-likelihood
logLik(m00)
## variance-covariance matrix
vcov(m00)
vcov(m00, model="sta")
vcov(m00, model="det")
## confidence intervals
confint(m00)
confint(m00, model="sta")
confint(m00, model="det")

## fitted values
## (conditional probability of occurrence given detection history:
## if W=1, fitted=1,
```
## svocc

```r
## if W=0, fitted=(phi*(1-delta)) / ((1-delta) + phi * (1-delta))
summary(fitted(m00))
## estimated probabilities: (phi*(1-delta)) / ((1-delta) + phi * (1-delta))
summary(m00$estimated.probabilities)
## probability of occurrence (phi)
summary(m00$occurrence.probabilities)
## probability of detection (delta)
summary(m00$occurrence.probabilities)
## Not run:
## model selection
m02 <- svocc(W ~ x1 | x3 + x4, datocc)
m03 <- drop1(m02, model="det")
## dropping one term at a time, resulting change in AIC
m03
## updating the model
m04 <- update(m02, . ~ . | . - x4)
m04
## automatic model selection
## part of the model (sta/det) must be specified
m05 <- svocc.step(m02, model="det")
summary(m05)

## nonparametric bootstrap
m06 <- bootstrap(m01, B=25)
attr(m06, "bootstrap")
extractBOOT(m06)
summary(m06, type="mle")
summary(m06, type="pmle") # no SEs! PMLE!!!
summary(m06, type="boot")
## vcov
vcov(m06, type="mle") # this does not work with PMLE
vcov(m06, type="boot") # this works
## confint
confint(m06, type="boot") # quantile based

## parametric bootstrap
## This is how observations are simulated
head(simulate(m01, 5))
m07 <- bootstrap(m01, B=25, type="param")
extractBOOT(m07)
summary(m07)

data(oven)
oven <- oven
oven[, c(4:8,10:11)][] <- lapply(oven[, c(4:8,10:11)], scale)
oven$count01 <- ifelse(oven$count > 0, 1, 0)
moven <- svocc(count01 ~ pforest | julian + timeday, oven)
summary(moven)
drop1(moven, model="det")
moven2 <- update(moven, . ~ . | . ~ timeday)
summary(moven)
```
BIC(moven, moven2)
AUC(moven, moven2)
rocplot(moven)
rocplot(moven2, col=2, add=TRUE)

## End(Not run)
Index

*Topic **datasets**
  databu, 9
  datocc, 11
  oven, 14
*Topic **hplot**
  AUC, 3
*Topic **htest**
  AUC, 3
    bootstrap, 3
cmulti, 4
  svabu, 15
  svocc, 18
*Topic **misc**
  convertEDR, 8
  load_BAM_QPAD, 13
*Topic **models**
  svabu, 15
  svocc, 18
*Topic **package**
  detect-package, 2
*Topic **utils**
  hbootindex, 12

AUC, 3

bootstrap, 3
cmulti, 2, 4
cmulti2.fit (cmulti), 4
convertEDR (convertEDR), 8
convertEDR, 8
databu, 9
datocc, 11
detect (detect-package), 2
detect-package, 2

extractBOOT (bootstrap), 3
extractMLE (svocc), 18
fitted.cmulti (cmulti), 4
hbootindex, 12
is.present (svabu), 15
load_BAM_QPAD, 13
optim, 5
oven, 14
predictMCMC (svabu), 15
rocplot (AUC), 3
svabu, 2, 10, 15
svabu_nb.fit (svabu), 15
svocc, 2, 11, 18
zif (svabu), 15