Package: extBatchMarking (via r-universe)

September 8, 2024

Type Package

Title Extended Batch Marking Models

Version 1.0.1

Date 2024-01-05

Maintainer Kehinde Olobatuyi <olobatuyikenny@uvic.ca>

Description A system for batch-marking data analysis to estimate survival probabilities, capture probabilities, and enumerate the population abundance for both marked and unmarked individuals. The estimation of only marked individuals can be achieved through the batchMarkOptim() function. Similarly, the combined marked and unmarked can be achieved through the batchMarkUnmarkOptim() function. The algorithm was also implemented for the hidden Markov model encapsulated in batchMarkUnmarkOptim() to estimate the abundance of both marked and unmarked individuals in the population. The package is based on the paper: ``Hidden Markov Models for Extended Batch Data" of Cowen et al. (2017) <doi:10.1111/biom.12701>.

License AGPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

LinkingTo Rcpp, RcppArmadillo

Imports doParallel, foreach, optimbase, Rcpp, parallel

Depends R (>= 4.0)

LazyData true

Suggests testthat (>= 3.0.0), knitr

Config/testthat/edition 3

URL https://github.com/Olobatuyi/extBatchMarking

BugReports https://github.com/Olobatuyi/extBatchMarking/issues

NeedsCompilation yes

Author Kehinde Olobatuyi [aut, cre]
(<https://orcid.org/0000-0002-4635-7895>), Simon Johns [aut],
Matthew RP Parker [aut]
(<https://orcid.org/0000-0003-3021-7959>), Steve Hof [aut],
Laura LE Cowen [aut] (<https://orcid.org/0000-0002-0853-1450>)

Date/Publication 2024-01-10 10:23:04 UTC

Repository https://olobatuyi.r-universe.dev

RemoteUrl https://github.com/cran/extBatchMarking

RemoteRef HEAD

RemoteSha a7459c7983364933b85dc589ae84bcdd8d8c4c7f

Contents

`batchLL` *batchLL function provides the batch marking log-likelihood*

Description

batchLL function provides the batch marking log-likelihood

Usage

```
batchLL(phi, p, R, begin_g, end_g, cores)
```

Arguments

phi	The probability of surviving and remaining in the population between occasions t and t +1, given an individual was alive and in the population at occasion t. This must be a number between 0 and 1.
p	The probability of capture at occasion t. This must be a number between 0 and 1.
R	The number of individuals marked and released at sampling occasion g from batch group g; g = 1,2,...,G. This must be an integer.
begin_g	The beginning of the occasion.
end_g	The end of the occasion.
cores	Number of cores for parallel.

Value

fr returns the log sum of the Hidden Markov Model.

---

batchLogit *batchLogit function*

Description

'batchLogit' provides the number between 0 and 1.

Usage

```
batchLogit(x)
```

Arguments

x	This is an input numerical value i.e double.

Value

Returns a number between 0 and 1.

batchMarkHmmLL *Log-likelihood function for marked model.*

Description

This helps users check whether the function can be optimized at the given initial values before optimizing using batchMarkOptim. After a quick check, if NAN or Inf is returned, the initial values should be revisited.

Usage

```
batchMarkHmmLL(
  par,
  data,
  choiceModel = c("model1", "model2", "model3", "model4"),
  cores
)
```

Arguments

par	Initial values for the parameters to be optimized over.
data	A capture-recapture data matrix or data frame.
choiceModel	This chooses among different models and allows for model selection
cores	The number of cores for parallelization.

Value

Negative Log-likelihood value of the likelihood function

Examples

```
library(extBatchMarking)
# Initial parameter
theta <- c(0, -1)
res1  <- batchMarkHmmLL(par        = theta,
                        data       = WeatherLoach,
                        choiceModel = "model4",
                        cores      = 1)
res1
```

batchMarkOptim *Marked model only.*

Description

batchMarkOptim function provides the batch marking function to be optimized.

Usage

```
batchMarkOptim(
  par = NULL,
  data,
  choiceModel = c("model1", "model2", "model3", "model4"),
  method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B"),
  parallel = FALSE,
  lowerBound = -Inf,
  cores = 1,
  hessian = FALSE,
  control,
  ...
)
```

Arguments

par	Initial values for the parameters to be optimized over.
data	A capture-recapture data matrix or data frame
choiceModel	This chooses among different models and allow for model selection
method	The method to be used. See optim for details.
parallel	Logical. Should the algorithm be run in parallel? This will be implemented in a future version.
lowerBound	Lower bounds on the variables for the "L-BFGS-B" method.
cores	The number of cores for parallelization
hessian	Logical. Should a numerically differentiated Hessian matrix be returned?
control	A list of control parameters. See optim for details.
...	Further arguments to be passed by user which goes into the optim function.

Details

Note that arguments after ... must be matched exactly. batchMarkOptim depends on optim function to optimize the parameters of the marked model only. By default optim performs minimization.

Value

For batchMarkOptim, a list with components:

phi The survival probability and remaining in the population between occasion t and t+1.

p The capture probability at occasion time t.

ll The optimized log-likelihood value of marked model.

hessian The hessian matrix.

AIC The Akaike Information Criteria for model selection.

References

Laura L. E. Cowen, Panagiotis Besbeas, Byron J. T. Morgan, 2017.: Hidden Markov Models for Extended Batch Data, Biometrics, 73, 1321-1331. DOI: 10.1111/biom.12701.

Examples

```
# Load the package
library(extBatchMarking)

# Load the WeatherLoach data from Cowen et al., 2017.
data(WeatherLoach)

# Initial parameter values
theta <- c(0, -1)


mod1 <- batchMarkOptim(
           par         = theta,
           data        = WeatherLoach,
           choiceModel = "model4",
           method      = "BFGS",
           parallel    = FALSE,
           hessian     = TRUE,
           control     = list(trace = 1)
    )

 # Survival probability
 mod1$phi
 # Capture probability
 mod1$p
 # Optimized log-likelihood
 mod1$ll
 # The Hessian matrix
 mod1$hessian
 # The Aikaike Information Criteria
 mod1$AIC



 mod2 <- batchMarkOptim(
```

```
            par         = theta,
            data        = WeatherLoach,
            choiceModel = "model4",
            method      = "L-BFGS-B",
            parallel    = FALSE,
            hessian     = TRUE,
            control     = list(trace = 1))

# Survival probability
mod2$phi
# Capture probability
mod2$p
# Optimized log-likelihood
mod2$ll
# The Hessian matrix
mod2$hessian
# The Akaike Information Criteria
mod2$AIC
```

`batchMarkUnmarkHmmLL` *Log-likelihood function for combined model.*

Description

This helps users check whether the function can be optimized at the given initial values before optimizing using `batchMarkUnmarkOptim`. After a quick check, if `NAN` or `Inf` is returned, the initial values should be revisited.

Usage

```
batchMarkUnmarkHmmLL(
  par,
  data,
  Umax,
  nBins,
  choiceModel = c("model1", "model2", "model3", "model4"),
  cores
)
```

Arguments

`par`	Initial values for the parameters to be optimized over.
`data`	A capture-recapture data matrix or data frame.
`Umax`	The maximum number of the unmarked individuals in the population for capture on any occasion.
`nBins`	The number of bin size into which the matrix will be divided.
`choiceModel`	This chooses among different models and allow for model selection.
`cores`	The number of cores for parallelization.

Value

Negative Log-likelihood value of the likelihood function.

Examples

```
library(extBatchMarking)
theta <- c(0.1, 0.1, 7, -1.5)
res3  <- batchMarkUnmarkHmmLL(par        = theta,
                              data       = WeatherLoach,
                              choiceModel = "model4",
                              Umax       = 1800,
                              nBins      = 20,
                              cores      = 1)
res3
```

`batchMarkUnmarkOptim` *Combined Marked and Unmarked models.*

Description

batchMarkUnmarkOptim function provides the batch marking and unmarked function to be optimized.

Usage

```
batchMarkUnmarkOptim(
  par = NULL,
  data,
  choiceModel = c("model1", "model2", "model3", "model4"),
  method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B"),
  Umax = 1800,
  nBins = 20,
  popSize = c("Horvitz_Thompson", "Model-Based"),
  parallel = FALSE,
  lowerBound = -Inf,
  cores = 1,
  hessian = FALSE,
  control,
  ...
)
```

Arguments

`par`	Initial values for the parameters to be optimized over.
`data`	A capture-recapture data matrix or data frame
`choiceModel`	This chooses among different models and allow for model selection
`method`	The method to be used. See optim for details.

Umax	The maximum number of the unmarked individuals in the population for capture on any occasion.
nBins	The number of bin size into which the matrix will be divided.
popSize	The Horvitz_Thompson method or Model-Based to compute population size.
parallel	Logical. Should the algorithm be run in parallel? This will be implemented in a future version.
lowerBound	Lower bounds on the variables for the "L-BFGS-B" method.
cores	The number of cores for parallelization
hessian	Logical. Should a numerically differentiated Hessian matrix be returned?
control	a list of control parameters. See optim for details.
...	Further arguments to be passed by user which goes into the optim function.

Details

Note that arguments after ... must be matched exactly.

batchMarkUnmarkOptim depends on optim function to optimize the parameters of the combined model. By default optim performs minimization.

Example on Umax and nBins: Umax = 1800 has a matrix of 1801 x 1801 and nBins = 20, reduces the matrix to 90 x 90. This is done in Cowen et al., 2017 to reduce the computing time when dealing with large matrix.

Value

A list of the following optimized parameters will be returned.

phi The survival probability and remaining in the population between occasion t and t+1.

p The capture probability at occasion time t.

ll The optimized log-likelihood value of marked model.

hessian The hessian matrix.

AIC The Akaike Information Criteria for model selection.

lambda Initial mean abundance at occasion t = 1.

gam Recruitment rate of individual into the unmarked population.

M Total number of marked individual in the population.

U Total number of unmarked individuals in the population available for capture at occasion t = 1,..., T.

N Total population size at time t = 1, ..., T.

References

Laura L. E. Cowen, Panagiotis Besbeas, Byron J. T. Morgan, 2017.: Hidden Markov Models for Extended Batch Data, Biometrics, 73, 1321-1331. DOI: 10.1111/biom.12701.

Examples

```
# Load the package
library(extBatchMarking)

# Load the WeatherLoach data from Cowen et al., 2017.
data(WeatherLoach)

# Initial parameter values
theta <- c(0.1, 0.1, 7, -1.5)


mod1 <- batchMarkUnmarkOptim(
           par         = theta,
           data        = WeatherLoach,
           Umax        = 1800,
           nBins       = 20,
           choiceModel = "model4",
           popSize    = "Horvitz_Thompson",
           method      = "CG",
           parallel    = FALSE,
           control     = list(trace = 1))

 # Survival probability
 mod1$phi
 # Capture probability
 mod1$p
 # Optimized log-likelihood
 mod1$ll
 # The Hessian matrix
 mod1$hessian
 # The Aikaike Information Criteria
 mod1$AIC
 # The initial mean abundance
 mod1$lambda
 # Recruitment rate into the population
 mod1$gam
 # The estimated abundance of unmarked animals
 mod1$U
 # The estimated abundance of marked animals
 mod1$M
 # The estimated total abundance of marked and unmarked animals
 mod1$N



mod2 <- batchMarkUnmarkOptim(
           par         = theta,
           data        = WeatherLoach,
           Umax        = 1800,
           nBins       = 20,
           choiceModel = "model4",
           popSize    = "Model-Based",
```

```
            method      = "L-BFGS-B",
            parallel    = FALSE,
            control     = list(trace = 1))

# Survival probability
mod2$phi
# Capture probability
mod2$p
# Optimized log-likelihood
mod2$ll
# The Hessian matrix
mod2$hessian
# The Akaike Information Criteria
mod2$AIC
# The initial mean abundance
mod2$lambda
# Recruitment rate into the population
mod2$gam
# The estimated abundance of unmarked animals
mod2$U
# The estimated abundance of marked animals
mod2$M
# The estimated total abundance of marked and unmarked animals
mod2$N
```

`batchUnmark2Viterbi` *batchUnmark2Viterbi function provides a wrapper for the batchUnmarkViterbi to compute the popuation abundance*

Description

batchUnmark2Viterbi function provides a wrapper for the batchUnmarkViterbi to compute the popuation abundance

Usage

```
batchUnmark2Viterbi(
  par,
  data,
  Umax,
  nBins,
  choiceModel = c("model1", "model2", "model3", "model4")
)
```

Arguments

`par`	Initial values for the parameters to be optimized over.
`data`	A capture-recapture data matrix or data frame.

Umax	The maximum number of the unmarked individuals in the population for capture on any occasion.
nBins	The number of bin size into which the matrix will be divided.
choiceModel	This chooses among different models and allows for model selection

Value

Negative Log-likelihood value of the likelihood function

`batchUnmarkHmmLL` *batchUnmarkHmmLL function provides the unmarked function to be optimized*

Description

batchUnmarkHmmLL function provides the unmarked function to be optimized

Usage

```
batchUnmarkHmmLL(phi, p, lambda, gam, Umax, nBins, u)
```

Arguments

phi	The probability of surviving and remaining in the population between occasions t and t +1, given an individual was alive and in the population at occasion t. This must be a number between 0 and 1.
p	The probability of capture at occasion t. This must be a number between 0 and 1.
lambda	The initial mean abundance (at occasion 1) for the unmarked population.
gam	The recruitment rate into the unmarked population.
Umax	The maximum number of the unmarked individuals in the population for capture on any occasion.
nBins	The number of bin size into which the matrix will be divided.
u	The number of individuals captured at sampling occasion t that were not marked; t = 1,...,T.

Value

Negative Log-likelihood value of the likelihood function

batchUnmarkViterbi *batchUnmarkViterbi function provides the implementation of the Viterbi alogrithm for the unmarked model*

Description

batchUnmarkViterbi function provides the implementation of the Viterbi alogrithm for the unmarked model

Usage

```
batchUnmarkViterbi(phi, p, lambda, gam, Umax, nBins, u)
```

Arguments

phi	The probability of surviving and remaining in the population between occasions t and t +1, given an individual was alive and in the population at occasion t. This must be a number between 0 and 1.
p	The probability of capture at occasion t. This must be a number between 0 and 1.
lambda	the initial mean abundance (at occasion 1) for the unmarked population.
gam	The recruitment rate into the unmarked population
Umax	The maximum number of the unmarked individuals in the population for capture on any occasion.
nBins	The number of bin size into which the matrix will be divided.
u	The number of individuals captured at sampling occasion t that were not marked; t = 1,...,T.

Value

Negative Log-likelihood value of the likelihood function

dbinpois *Convolution of Poisson and Binomial for Batch*

Description

This is the convolution of Poisson and Binomial distributions

Usage

```
dbinpois(z, n, par)
```

Arguments

z	This is the vector of numerical values
n	The nrow of capture-recapture data matrix or data frame
par	This is the vector of parameter values: average from Poisson distribution and probability of success from Binomial distribution

Details

The convolution of Poisson and Binomial distribution helps us to compute the number of individuals that have survived from t-1 to t in the combined model while simultaneously computing the number of individuals recruited into the population at occasion t.

The survival is modeled as Binomial distribution and the recruitment as the Poisson distirubiton

Value

f This is the output of the convolution from the Binomial and Poisson distributions

---

delta_g *initial probability function*

Description

initial probability function

Usage

```
delta_g(R)
```

Arguments

R	The number of individuals marked and released at sampling occasion g from batch group g; g = 1,2,...,G. This must be an integer.

Value

A vector of initial value with 1 at the observed position

gamma_gt — *Transition State Probability 'gamma_gt' computes the transition probability matrix*

Description

Transition State Probability 'gamma_gt' computes the transition probability matrix

Arguments

R	integer number of marked individuals released per occasion
phi	double number. Survival probability of individuals
cores	The number of cores on your machine.

Value

pR Returns the transition matrix

probs — *State-dependent probability function*

Description

'probs' computes the state-dependent transition matrix

Usage

```
probs(r, p, R)
```

Arguments

r	The number of individuals from batch group "g" recaptured at recapture occasion t; g = 1,2,...,G, t = g+1,...,T. This must be an integer.
p	The probability of capture at occasion t. This must be a number between 0 and 1.
R	The number of individuals marked and released at sampling occasion g from batch group g; g = 1,2,...,G. This must be an integer.

Value

PR diagonal matrix of the state-dependent probability.

`WeatherLoach` *Weather Loach data*

Description

Data from marked individuals captured on multiple occasions The weather-loach study was described in detail by Huggin () Different colored batch tags were given to a random sample of unmarked individuals at each occasion.

Usage

```
WeatherLoach
```

Format

'Weather_loach':

A data frame with 10 rows indicating number of captures and 11 columns indicating recaptures

Weather Loach Data

Index