Package: SPAS (via r-universe)

March 8, 2025

Type Package

Title Stratified-Petersen Analysis System

Version 2025.2.1

Date 2025-02-01

LinkingTo TMB, RcppEigen

Imports checkmate, MASS, Matrix, msm, numDeriv, plyr, reshape2, TMB (>= 1.7.15), utils

Description The Stratified-Petersen Analysis System (SPAS) is designed to estimate abundance in two-sample capture-recapture experiments where the capture and recaptures are stratified. This is a generalization of the simple Lincoln-Petersen estimator. Strata may be defined in time or in space or both, and the s strata in which marking takes place may differ from the t strata in which recoveries take place. When s=t, SPAS reduces to the method described by Darroch (1961) <doi:10.2307/2332748>. When s<t, SPAS implements the methods described in Plante, Rivest, and Tremblay (1988) <doi:10.2307/2533994>. Schwarz and Taylor (1998) <doi:10.1139/f97-238> describe the use of SPAS in estimating return of salmon stratified by time and geography. A related package, BTSPAS, deals with temporal stratification where a spline is used to model the distribution of the population over time as it passes the second capture location. This is the R-version of the (now obsolete) standalone Windows program of the same name.

License GPL (>= 2)

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation yes

Author Carl James Schwarz [aut, cre]

Maintainer Carl James Schwarz <cschwarz.stat.sfu.ca@gmail.com>

Repository CRAN

Date/Publication 2025-02-06 07:30:02 UTC

Config/pak/sysreqs libicu-dev

Contents

`SPAS.autopool` *Autopooling a Stratified-Petersen (SP) data set. This function applies pooling rules to pool a SPAS dataset to meeting minimum sparsity requirements .*

Description

Autopooling a Stratified-Petersen (SP) data set. This function applies pooling rules to pool a SPAS dataset to meeting minimum sparsity requirements .

Usage

```
SPAS.autopool(
  rawdata,
  min.released = 100,
  min.inspected = 50,
  min.recaps = 50,
  min.rows = 1,
  min.cols = 1
)
```

Arguments

`rawdata`	An (s+1) x (t+1) of the raw data BEFORE pooling. The s x t upper left matrix is the number of animals released in row stratum i and recovered in column stratum j. Row s+1 contains the total number of UNMARKED animals recovered in column stratum j. Column t+1 contains the number of animals marked in each row stratum but not recovered in any column stratum. The rawdata[s+1, t+1] is not used and can be set to 0 or NA. The sum of the entries in each of the first s rows is then the number of animals marked in each row stratum. The sum of the entries in each of the first t columns is then the number of animals captured (marked and unmarked) in each column stratum. The row/column names of the matrix may be set to identify the entries in the output.

`min.released` Minimum number of releases in a pooled row

`min.inspected` Minimum number of inspections in a pooled column

`min.recaps` Minimum number of recaptures before any rows can be pooled

`min.rows`, `min.cols`
Minimum number or rows and columns after pooling

Details

In many cases, the stratified set of releases and recapture is too sparse (many zeroes) or count are very small. Pooling rows and columns may be needed.

Data needs to be pooled both row wise and column wise if the data are sparse to avoid singularities in the fit. This function automates pooling rows or columns following Schwarz and Taylor (1998).

- All rows that have 0 releases are discarded
- All columns that have 0 recaptures of tagged fish and 0 fish inspected are discarded
- Starting at the first row and working forwards in time, and then working from the final row and working backwards in time, . rows are pooled until a minimum of `min.released` are released. An alternating pooling (from the top, from the bottom, from the top, etc) is used
- Starting at the first column and working forwards in time, . and then working from the final column and working backwards in time, columns are pooled until a minimum of `min.inspected` are inspected. An alternating pooling (from the left, from the right, from the left, etc) is used.
- If the sum of the total recaptures from released fish is <= `min.recaps`, then all rows are pooled (which reduces to a Chapman estimator)

Value

A list with a suggest pooling.

Examples

```
conne.data.csv <- textConnection("
9  ,    21  ,     0  ,    0  ,    0  ,    0  ,   171
0  ,   101  ,    22  ,    1  ,    0  ,    0  ,   763
0  ,     0  ,   128  ,   49  ,    0  ,    0  ,   934
0  ,     0  ,     0  ,   48  ,   12  ,    0  ,   434
0  ,     0  ,     0  ,    0  ,    7  ,    0  ,    49
0  ,     0  ,     0  ,    0  ,    0  ,    0  ,     4
351,  2736  ,  3847  , 1818  ,  543  ,   191 ,     0")
conne.data <- as.matrix(read.csv(conne.data.csv, header=FALSE))
close(conne.data.csv)

SPAS.autopool(conne.data)
```

SPAS.fit.model *Fit a Stratified-Petersen (SP) model using TMB.*

Description

This function fits a Stratified-Petersen (Plante, 1996) to data and specify which rows/columns of the data should be pooled. The number of rows after pooling should be <= number of columns after pooling .

Usage

```
SPAS.fit.model(
  model.id = "Stratified Petersen Estimator",
  rawdata,
  autopool = FALSE,
  row.pool.in = NULL,
  col.pool.in = NULL,
  row.physical.pool = TRUE,
  theta.pool = FALSE,
  CJSpool = FALSE,
  optMethod = c("nlminb"),
  optMethod.control = list(maxit = 50000),
  svd.cutoff = 1e-04,
  chisq.cutoff = 0.1,
  min.released = 100,
  min.inspected = 50,
  min.recaps = 50,
  min.rows = 1,
  min.cols = 1
)
```

Arguments

model.id	Character string identifying the name of the model including any pooling..
rawdata	An (s+1) x (t+1) of the raw data BEFORE pooling. The s x t upper left matrix is the number of animals released in row stratum i and recovered in column stratum j. Row s+1 contains the total number of UNMARKED animals recovered in column stratum j. Column t+1 contains the number of animals marked in each row stratum but not recovered in any column stratum. The rawdata[s+1, t+1] is not used and can be set to 0 or NA. The sum of the entries in each of the first s rows is then the number of animals marked in each row stratum. The sum of the entries in each of the first t columns is then the number of animals captured (marked and unmarked) in each column stratum. The row/column names of the matrix may be set to identify the entries in the output.
autopool	Should the automatic pooling algorithms be used. Give more details here on these rule work.

`row.pool.in`, `col.pool.in`
: Vectors (character/numeric) of length s and t respectively. These identify the rows/columns to be pooled before the analysis is done. The vectors consists of entries where pooling takes place if the entries are the same. For example, if s=4, then row.pool.in = c(1,2,3,4) implies no pooling because all entries are distinct; row.pool.in=c("a","a","b","b") implies that the first two rows will be pooled and the last two rows will be pooled. It is not necessary that row/columns be continuous to be pooled, but this is seldom sensible. A careful choice of pooling labels helps to remember what as done, e.g. row.pool.in=c("123","123","123","4") indicates that the first 3 rows are pooled and the 4th row is not pooled. Character entries ensure that the resulting matrix is sorted properly (e.g. if row.pool.in=c(123,123,123,4), then the same pooling is done, but the matrix rows are sorted rather strangely.

`row.physical.pool`
: Should physical pooling be done (default) or should logical pooling be done. For example, if there are 3 rows in the data matrix and row.pool.in=c(1,1,3), then in physical pooling, the entries in rows 1 and 2 are physically added together to create 2 rows in the data matrix before fitting. Because the data has changed, you cannot compare physical pooling using AIC. In logical pooling, the data matrix is unchanged, but now parameters p1=p2 but the movement parameters for the rest of the matrix are not forced equal.

`theta.pool`, `CJSpool`
: NOT YET IMPLEMENTED. DO NOT CHANGE.

`optMethod`
: What optimization method is used. Defaults is the nlminb() function..

`optMethod.control`
: Control parameters for optimization method. See the documentation on the different optimization methods for details.

`svd.cutoff`
: When finding the variance-covariance matrix, a singular value decomposition is used. This identifies the smallest singular value to retain.

`chisq.cutoff`
: When finding a goodness of fit statistic using (obs-exp)^2/exp, all cell whose Exp < gof.cutoff are ignored to try and remove structural zero cells.

`min.released`
: Minimum number of releases in a pooled row

`min.inspected`
: Minimum number of inspections in a pooled column

`min.recaps`
: Minimum number of recaptures before any rows can be pooled

`min.rows`, `min.cols`
: Minimum number or rows and columns after pooling

Value

A list with many entries. Refer to the vignettes for more details.

Examples

```
conne.data.csv <- textConnection("
9  ,   21  ,    0  ,   0  ,   0  ,   0  ,  171
0  ,  101  ,   22  ,   1  ,   0  ,   0  ,  763
0  ,    0  ,  128  ,  49  ,   0  ,   0  ,  934
0  ,    0  ,    0  ,  48  ,  12  ,   0  ,  434
```

```
0  ,     0  ,     0  ,    0  ,    7  ,    0  ,    49
0  ,     0  ,     0  ,    0  ,    0  ,    0  ,     4
351,  2736  ,  3847  , 1818  ,  543  ,   191 ,     0")
conne.data <- as.matrix(read.csv(conne.data.csv, header=FALSE))
close(conne.data.csv)

mod1 <- SPAS.fit.model(conne.data, model.id="Pooling rows 1/2, 5/6; pooling columns 5/6",
                       row.pool.in=c("12","12","3","4","56","56"),
                       col.pool.in=c(1,2,3,4,56,56))
mod2 <- SPAS.fit.model(conne.data, model.id="Auto pool",
                       autopool=TRUE)
```

`SPAS.print.model` *Print or Extract the results from a fit of a Stratified-Petersen (SP) model when using the TMB optimizer*

Description

This function makes a report/extracts the components of the results of the model fitting .

Usage

```
SPAS.print.model(x, extract = FALSE)
```

Arguments

`x`	A result from the model fitting. See `SPAS.fit.model`
`extract`	Should the function simply return the "printed" components in a list?.

Value

A report to the console or extracted components. Refer to the vignettes.

Examples

```
conne.data.csv <- textConnection("
9  ,    21  ,     0  ,    0  ,    0  ,    0  ,   171
0  ,   101  ,    22  ,    1  ,    0  ,    0  ,   763
0  ,     0  ,   128  ,   49  ,    0  ,    0  ,   934
0  ,     0  ,     0  ,   48  ,   12  ,    0  ,   434
0  ,     0  ,     0  ,    0  ,    7  ,    0  ,    49
0  ,     0  ,     0  ,    0  ,    0  ,    0  ,     4
351,  2736  ,  3847  , 1818  ,  543  ,   191 ,     0")
conne.data <- as.matrix(read.csv(conne.data.csv, header=FALSE))
close(conne.data.csv)

mod1 <- SPAS.fit.model(conne.data, model.id="Pooling rows 1/2, 5/6; pooling columns 5/6",
                       row.pool.in=c("12","12","3","4","56","56"),
                       col.pool.in=c(1,2,3,4,56,56))
```

```
SPAS.print.model(mod1)

out <- SPAS.print.model(mod1, extract=TRUE)
names(out)
out$spas
```

Index