Package 'Petersen'

May 31, 2024

Type Package

Title Estimators for Two-Sample Capture-Recapture Studies

Version 2024.6.1

Date 2024-06-01

Description A comprehensive implementation of Petersen-type estimators
and its many variants for two-sample capture-recapture studies.
A conditional likelihood approach is used that allows
for tag loss; non reporting of tags; reward tags; categorical, geographical and temporal stratification;
partial stratification; reverse capture-recapture;
and continuous variables in modeling the probability of capture.
Many examples from fisheries management are presented.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Imports AICcmodavg, bbmle, BTSPAS, formula.tools, ggplot2, MASS,
Matrix, msm, numDeriv, plyr, reshape2, rlang, SPAS, stats,
stringr, tidyr, utils

Depends R (>= 2.10)

Suggests knitr, rmarkdown, R.rsp

VignetteBuilder knitr, rmarkdown, R.rsp

URL https://github.com/cschwarz-stat-sfu-ca/Petersen

BugReports https://github.com/cschwarz-stat-sfu-ca/Petersen/issues

NeedsCompilation no

Author Carl Schwarz [aut, cre]

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

Repository CRAN

Date/Publication 2024-05-31 06:20:04 UTC

R topics documented:

`cap_hist_to_n_m_u` *Convert capture history data to n, m and u for use in BTSPAS*

Description

Convert capture history data to n, m and u for use in BTSPAS

Usage

```
cap_hist_to_n_m_u(data, sep = "..")
```

Arguments

`data`	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.
`sep`	Separator used between strata in cap_hit

Details

The frequency variable (`freq` in the `data` argument) is the number of animals with the corresponding capture history.

Capture histories (`cap_hist` in the `data` argument) are character values of the format `xx..yy` is a capture_history where `xx` and `yy` are the temporal stratum (e.g., julian week) and `'..'` separates the two temporal strata. If a fish is released in temporal stratum and never captured again, then `yy` is set to 0; if a fish is newly captured in temporal stratum `yy`, then `xx` is set to zero. For example, a capture history of `23..23` indicates animals released in temporal stratum 23 and recaptured in temporal stratum 23; a capture history of `23..00` indicates animals released in temporal stratum 23 and never seen again; a capture history of `00..23` indicates animals newly captured in temporal stratum 23 at the second sampling event.

. In the diagonal case, fish are only recovered in the same temporal stratum. In the non-diagonal case, fish are allowed to move among temporal strata.

It is not necessary to label the temporal strata starting at 1; BTSPAS will treat the smallest value of the temporal strata seen as the first stratum and will interpolate for temporal strata without any data. Temporal strata labels should be numeric, i.e., do NOT use A, B, C etc.

Value

A list with entries for the stratum index, n (number released), m matrix of recoveries in the current, next, etc stratum, and u (number of unmarked fish) captured in this recovery stratum.

Examples

```
data(data_btspas_diag1)
cap_hist_to_n_m_u(data_btspas_diag1)

data(data_btspas_nondiag1)
cap_hist_to_n_m_u(data_btspas_nondiag1)
```

`data_btspas_diag1` *Estimating abundance of outgoing smolt - BTSPAS - diagonal case*

Description

This is the first diagonal case dataset from BTSPAS.

Usage

```
data(data_btspas_diag1)
```

Format

`data_btspas_diag1`**:**

A data frame with many rows and 3 columns

`cap_hist`**.** Capture history of the form 'jweek..jweek' for fish that are recaptured in the same julian week; '0..jweek' for unmarked fish newly captured in that julian week ; 'jweek..0' for fish released in the julian week but never recaptured.

`freq`**.** Number of fish with this history.

`logflow` log(flow) for this julian week

Details

Consider an experiment to estimate the number of outgoing smolts on a small river. The run of smolts extends over several weeks. As smolts migrate, they are captured and marked with individually numbered tags and released at the first capture location using, for example, a fishwheel. The migration continues, and a second fishwheel takes a second sample several kilometers down stream. At the second fishwheel, the captures consist of a mixture of marked (from the first fishwheel) and unmarked fish.

The efficiency of the fishwheels varies over time in response to stream flow, run size passing the wheel and other uncontrollable events. So it is unlikely that the capture probabilities are equal over time at either location, i.e. are heterogeneous over time.

We suppose that we can temporally stratify the data into, for example, weeks, where the capture-probabilities are (mostly) homogeneous at each wheel in each week. Furthermore, suppose that fish captured and marked in each week tend to migrate together so that they are captured in a single subsequent stratum. For example, suppose that in each julian week j, $n1_j$ fish are marked and

released above the rotary screw trap. Of these, $m2_j$ are recaptured. All recaptures take place in the week of release, i.e. the matrix of releases and recoveries is diagonal. The $n1_j$ and $m2_j$ establish the capture efficiency of the second trap in julian week j.

At the same time, $u2_j$ unmarked fish are captured at the screw trap.

Capture-efficiency may be related to flow, so the log(flow) is also recorded.

`data_btspas_nondiag1` *Estimating abundance of salmon - BTSPAS - non-diagonal case*

Description

This is the first non-diagonal case dataset from BTSPAS.

Usage

```
data(data_btspas_nondiag1)
```

Format

`data_btspas_nondiag1`:

A data frame with many rows and 3 columns

`cap_hist`. Capture history of the form 'week1..week2' for fish that are released on week 1 and recaptured on week 2 ; '0..week22' for unmarked fish newly captured in week 2; 'week1..0' for fish released in week 1 but never recaptured.

`freq`. Number of fish with this history.

Details

Incoming sockeye salmon are captured on a first wheel, tagged with color tags that vary by week, and recaptured on an upriver weir. The upriver weir was not in operation for the first few weeks.

`data_kokanee_tagloss` *Capture-recapture on Kokanee in Metolius River with tag loss*

Description

This is the data from Hyun et al (2012). In August and September 2007, the period just before the spawning run, adult kokanee were collected by beach seining in the upper arm of the lake near the confluence with the Metolius River. Fish were tagged with nonpermanent, plastic T-bar anchor tags and then were released back into the lake. Randomly selected fish received single tags of one color, while the other fish received two tags of a second color (i.e., the double tags were identical in color). In late September through October, spawning ground surveys were conducted by 2–3 people walking abreast in a downstream direction (or floating, in sections where the water depth and flow were too great to allow walking). Instead of being physically recaptured, the fish were resighted as they swam freely in the clear, relatively shallow water within the spawning areas of the river. The total number of fish observed with or without a tag (or tags) was recorded for each section, and information on the number and color of tags for each marked fish was also noted.

Usage

```
data(data_kokanee_tagloss)
```

Format

`data_kokanee_tagloss`:

A data frame with many rows and 2 columns

`cap_hist`. Capture history (1000, 1010, 1100, 1110, 1111).

`freq`. Number of fish with this history. Always 1

Details

Because fish were not handled, it is not possible to know which of the double tags were lost, and so only models with equal retention probabilities and non-distinguishable double tags should be fit. Note that the capture history for a lost of 1 indistinguishable tag is 111X rather than 1110 or 1101 (both of which are not allowed in the model with indistinguishable double tags)

---

data_lfc_reverse *Lower Fraser Coho for Reverse Capture-Recapture with geographic stratification.*

Description

This is the data provided by Kaitlyn Dionne, DFO. Arbeider et al (2020) proposed to estimate the run size of Lower Fraser River Coho (LFC) using a geographically stratified reverse-capture method. Briefly, a LFC coho swim upstream, they are sampled near New Westminister, BC, which is downstream from several major rivers up which are large spawning populations. These sample fish are assigned to the spawning population using genetic and other methods. These spawning populations are identified as the Chilliwack Hatchery (denoted *C*), the Lilloet River natural spawning population (denoted *L*), the Nicomen Slough population (denote *N*) and all other population (denoted as *0*). Notice that the sample fish at New Westminister are NOT physically tagged, and population assignment is through genetic and other measures. The upstream migration extends over two months (September to October) and is divided into 3 temporal strata corresponding to *Early* (denoted *1E*), *Peak* (denoted *2P*) and *Late* (denoted *3L*). The digits 1, 2, 3 in front of the codes ensures that the temporal strata are sorted temporally, but this is merely a convenience and does not affect the results. The spawning populations at *C*, *L*, and *N* are estimated by a variety of methods (see Arbeider, et al. 2020). Each of the population estimates also has an estimated (which will be ignored for now).

Usage

```
data(data_lfc_reverse)
```

Format

`data_lfc_reverse` A data frame with many rows and 4 columns

`cap_hist`. Capture history with possible histories as noted below

`freq`. Number of fish with this history.

`SE`. SE of the number of fish with this history. Only available for total escapement to C, L, N

---

data_NorthernPike *Capture-recapture experiment on Northern Pike in Mille Lacs, MN, in 2005.*

Description

Fish were tagged on the spawning grounds and recovered in the summer gillnet assessment. Fish were double tagged, and a tag loss analysis showed that tag loss was negligible. It will be ignored here. Length was measured a both times and didn't not change very much between the two sampling occasions. The value recorded below is the average of the two lengths if both lengths were present. Fish that were not sexed or measured for length are ignored and not included

Usage

```
data(data_NorthernPike)
```

Format

`data_NorthernPike`:

A data frame with many rows and 4 columns

`cap_hist`. Capture history (10, 01, or 11). Note that $n_{10} = n_1 - m_2$; $n_{01} = n_2 - m_2$; and $n_{11} = m2$

`freq`. Number of fish with this history. Always 1

`Sex`. Sex of the fish. M=Male; F=Female

`Length` Length of the fish in inches.

---

data_NorthernPike_tagloss *Capture-recapture experiment on Northern Pike in Mille Lacs, MN, in 2005 with tagloss information.*

Description

Fish were tagged on the spawning grounds and recovered in the summer gillnet assessment. Fish were double tagged and the double tagging information is included here. Length was measured a both times and didn't not change very much between the two sampling occasions. The value recorded below is the average of the two lengths if both lengths were present. Fish that were not sexed or measured for length are ignored and not included

Usage

```
data(data_NorthernPike_tagloss)
```

Format

`data_NorthernPike_tagloss`:

A data frame with many rows and 4 columns

`cap_hist`. Capture history (10, 01, or 11). Note that $n_{10} = n_1 - m_2$; $n_{01} = n_2 - m_2$; and $n_{11} = m2$

`freq`. Number of fish with this history. Always 1

`Sex`. Sex of the fish. M=Male; F=Female

`Length` Length of the fish in inches.

---

data_rodli *Capture-recapture experiment at Rodli Tarn.*

Description

Ricker (1975) gives an example of work by Knut Dahl on estimating the number of brown trout (*Salmo truitta*) in some small Norwegian tarns. Between 100 and 200 trout were caught by seining, marked by removing a fin (an example of a batch mark) and distributed in a systematic fashion around the tarn to encourage mixing. A total of n_1=109 fish were captured, clipped and released, n_2=177 fish were captured at the second occasion, and m_2=57 marked fish were recovered.

Usage

```
data(data_rodli)
```

Format

`data_rodli`:

A data frame with 3 rows and 2 columns:

`cap_hist`. Capture history (10, 01, or 11). Note that $n_{10} = n_1 - m_2$; $n_{01} = n_2 - m_2$; and $n_{11} = m_2$

`freq`. Number of fish with this history

data_sim_reward *Simulated data for reward tags used to estimate reporting rate*

Description

This is simulated data with the parameter values given in details.

Usage

```
data(data_sim_reward)
```

Format

data_sim_reward**:**

A data frame with many rows and 2 columns

cap_hist**.** Capture history (1000, 1010, 0P00, 0P0P, 0010).

freq**.** Number of fish with this history.

Details

```
 data_sim_reward <-LP_TL_simulate(
      dt_type=dt_type,  #  permanent tag
      N=10000,
      cov1=function(N)         {rep(1,N)},
      cov2=function(cov1)      {rep(1,  length(cov1))},
      p1  =function(cov1, cov2){rep(.1, length(cov1))},
      pST =function(cov1, cov2){rep(.75,length(cov1))},
      rho1=function(cov1, cov2){rep(.70,length(cov1))},
      rho2=function(cov1, cov2){rep(1,  length(cov1))},  # permanent second tag
      p2  =function(cov1, cov2){rep(.1, length(cov1))},
      seed=45985, trace=FALSE)
# we don't have fish with both tags
data_sim_reward$cap_hist <- gsub("1P", "0P", data_sim_reward$cap_hist)
```

---

data_sim_tagloss_t2perm

Simulated data for tag loss with second permanent tag.

Description

This is simulated data with the parameter values given in details.

Usage

```
data(data_sim_tagloss_t2perm)
```

Format

data_sim_tagloss_t2perm:

A data frame with many rows and 2 columns

cap_hist. Capture history (1000, 1010, 1P00, 1P0P, 1P1P, 0010).

freq. Number of fish with this history.

Details

```
data_sim_tagloss_t2perm <-LPTL_simulate(
      dt_type="t2perm",         # second permanent
      N=10000,
      cov1=function(N)         {rep(1,N)},
      cov2=function(cov1)      {rep(1,  length(cov1))},
      p1  =function(cov1, cov2){rep(.1, length(cov1))},
      pST =function(cov1, cov2){rep(.25,length(cov1))},
      rho1=function(cov1, cov2){rep(.70,length(cov1))},
      rho2=function(cov1, cov2){rep(1,  length(cov1))}, # permanent tag
      p2  =function(cov1, cov2){rep(.1, length(cov1))},
      seed=234523, trace=FALSE)
```

---

data_sim_tagloss_twoD *Simulated data for tag loss with 2 distinguishable tags.*

Description

This is simulated data with the parameter values given in the description.

Usage

```
data(data_sim_tagloss_twoD)
```

Format

data_sim_tagloss_twoD:

A data frame with many rows and 2 columns

cap_hist. Capture history (1000, 1010, 1100, 1110, 1101, 1111).

freq. Number of fish with this history.

Details

```
data_sim_tagloss_twoD <-LPTL_simulate(
      dt_type="twoD",         # two distinguishable tags
      N=10000,
      cov1=function(N)         {rep(1,N)},
      cov2=function(cov1)      {rep(1,  length(cov1))},
      p1  =function(cov1, cov2){rep(.1, length(cov1))},
```

```
pST =function(cov1, cov2){rep(.25,length(cov1))},
rho1=function(cov1, cov2){rep(.70,length(cov1))},
rho2=function(cov1, cov2){rep(.80,length(cov1))},
p2  =function(cov1, cov2){rep(.1, length(cov1))},
seed=234523, trace=FALSE)
```

`data_spas_harrison` *Estimating abundance of salmon - SPAS - Harrison River*

Description

Incoming sockeye salmon are captured on a first wheel, tagged with color tags that vary by week, and recaptured on several spawning areas.

Usage

```
data(data_spas_harrison)
```

Format

`data_spas harrison`**:**

A data frame with many rows and 2 columns

`cap_hist`**.** Capture history of the form 'week..area' for fish that are released on week and recaptured area ; '0..area' for unmarked fish newly captured in area; 'week..0' for fish released in week but never recaptured.

`freq`**.** Number of fish with this history.

`data_wae_is_long` *Walleye data with incomplete stratification with length covariate*

Description

Data used in Premarathna, W.A.L., Schwarz, C.J., Jones, T.S. (2018) Partial stratification in two-sample capture–recapture experiments. Environmetrics, 29:e2498. https://doi.org/10.1002/env.2498

Usage

```
data(data_wae_is_long)
```

Format

`data_wae_is_long` A data frame with many rows and 3 columns

`cap_hist`**.** Capture history with possible histories as noted below

`freq`**.** Number of fish with this history.

`length` Length of fish (inches)

Details

Fish were tagged on the spawning grounds and recovered in the summer gillnet assessment.

Length was measured a both times and didn't not change very much between the two sampling occasions. The value recorded below is the average of the two lengths if both lengths were present.

Rather than sexing all of the fish, only a sub-sample of unmarked fish is sexed at each sampling occasion. Possible capture histories are then M0, F0, MM, FF, U0, UU, 0M, 0F

`data_wae_is_short` *Walleye data with incomplete stratification with no covariates and condensed*

Description

Data used in Premarathna, W.A.L., Schwarz, C.J., Jones, T.S. (2018) Partial stratification in two-sample capture–recapture experiments. Environmetrics, 29:e2498. https://doi.org/10.1002/env.2498

Usage

```
data(data_wae_is_short)
```

Format

`data_wae_is_short` A data frame with many rows and 2 columns

`cap_hist`**.** Capture history with possible histories as noted below

`freq`**.** Number of fish with this history.

Details

Data is slightly different from that in paper above because some fish did not have length measured and so were drop from data_wae_is_long and this is the condensed version of data_wae_is_long.

Fish were tagged on the spawning grounds and recovered in the summer gillnet assessment.

Rather than sexing all of the fish, only a sub-sample of unmarked fish is sexed at each sampling occasion. Possible capture histories are then M0, F0, MM, FF, U0, UU, 0M, 0F

data_yukon_reverse *Yukon River data used for Reverse Capture-Recapture example.*

Description

Data from Hamazaki, T. and DeCovich, N. (2014). Application of the Genetic Mark–Recapture Technique for Run Size Estimation of Yukon River Chinook Salmon. North American Journal of Fisheries Management, 34, 276-286. DOI: 10.1080/02755947.2013.869283 This is the data from the 2011 data in Table 2 of the above paper. Estimated that total escapement to Canada (plus harvest) was 66,225 (SE 1574) Estimated that proportion of stock that was Canadian was .34644 (SE .030) We converted this into a "sample size" and number of fish with Canadian genetics that gave the same SE.

Usage

```
data(data_yukon_reverse)
```

Format

`data_yukon_reverse` A data frame with many rows and 4 columns

`cap_hist`. Capture history with possible histories as noted below

`freq`. Number of fish with this history.

`SE`. SE of the number of fish with this history

fit_classes **LP_fit**, **LP_IS_fit**, **LP_SPAS_cit**, **CL_fit**, **LP_BTSPAS_fit_Diag**, **LP_BTSPAS_fit_NonDiag**, **LP_CL_fit** *classes.*

Description

We assign a "class" (one of the classes above) to the results from one of the fitting methods. This class designation is **only used to ensure that estimation routines have the correct type of fit when finding estimates of abundance**, and **model averaging only considers models of comparable class** when creating the model averaging results of abundance.

Usage

```
fit_classes()
```

Details

The structure of an object of the above classes is roughly comparable across classes. The object should be a list with the following objects

- **summary** A data frame with the model for the parameters of the model; the conditional log-likelihood; the number of parameters; the number of parameters, and method used to fit the model
- **data** A data frame with the raw data used in the fit
- **fit** Results of the fit including the estimates, SE, vcov, etc.
- **datetime** Date and time the fit was done

Other objects may also be included in the list.

After a fit performed, estimates of **ABUNDANCE** are extracted using the **xxx_est()** function corresponding to the **xxx_fit()** function used to estimate parameters. This separation occurs because

- abundance is a derived estimate; the fits are based on conditional likelihood (on the observed data) and the abundance parameter does not appear in the conditional likelihood. Abundance is usually estimated using a variation of a Horvitz-Thompson estimator.
- you can obtain estimates of overall abundance, subsets of the population (e.g., sex or other strata) from the same fit, and so you don't need to do several fits to get several estimates of abundance.

Value

NOTHING. This is not a function, but only documents how I use "classes".

`logit` *Logit and anti-logit function.*

Description

Compute the logit or anti-logit.

Usage

```
logit(p)

expit(theta)
```

Arguments

`p`	probability between 0 and 1.
`theta`	logit between -infinity and +infinity

Value

Computed logit or anti-logit

Author(s)

C.J.Schwarz <cschwarz.stat.sfu.ca@gmail.com>

Examples

```
##---- compute the logit and its inverse
logitp <- logit(.3)
p <- expit(-.84)
```

LP_AICc *Create an AIC table comparing multiple LP fits*

Description

This will take a series of LP fits and computes the usual AICc table and model weights

Usage

```
LP_AICc(...)
```

Arguments

...	Series of LP fits

Value

An data frame with an AICc table and model weights etc

Examples

```
data(data_rodli)
mt <- Petersen::LP_fit(data=data_rodli, p_model=~..time)
m0 <- Petersen::LP_fit(data=data_rodli, p_model=~1)
Petersen::LP_AICc(m0,mt)
```

LP_BTSPAS_est *Extract estimates of abundance after BTSPAS fit*

Description

This will take a previous fit and return estimates of abundance.

Usage

```
LP_BTSPAS_est(LP_BTSPAS_fit, parm = "Ntot", conf_level = 0.95, trace = FALSE)
```

Arguments

LP_BTSPAS_fit	A result of an call to fitting at BTSPAS object.
parm	Which parameter from the BTSPAS fix is to be extracted?
conf_level	The expected coverage for confidence intervals on N.
trace	If trace flag is set in call when estimating functions

Value

An list object of class *LP_BTSPAS_est* with the following elements

- **summary** A data frame with the estimates of abundance, SE, and CI
- **datetime** Date and time the fit was done

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

Examples

```
# NOTE. To keep execution time to a small value as required by CRAN
# I've made a very small example.
# Additionally, I've set the number of MCMC chains, iterations, burning, simulation to save to
# small values. Proper mixing may not have occurred yet.
# When using this routine, you likely want to the use the default values
# for these MCMC parameters.

data(data_btspas_diag1)
# extract the strata of interest
temp<- cbind(data_btspas_diag1,
             split_cap_hist( data_btspas_diag1$cap_hist,
                             sep="..", make.numeric=TRUE))
# only use data up to week 10 to keep example small
temp <- temp[ temp$t1 %in% 0:10 & temp$t2 %in% 0:10,]

fit <- Petersen::LP_BTSPAS_fit_Diag(
```

```
  temp,
  p_model=~1,
  InitialSeed=23943242,
  # the number of chains and iterations are too small to be useful
  # they are set to a small number to pare execution time to <5 seconds for an example
  n.chains=2, n.iter=20000, n.burnin=1000, n.sims=100,
  quietly=TRUE
)
fit$summary

# now get the estimates of abundance
est <-  Petersen::LP_BTSPAS_est (fit)
est$summary
```

LP_BTSPAS_fit_Diag *Wrapper (*_fit) to call the Time Stratified Petersen Estimator with Diagonal Entries function in BTSPAS.*

Description

Takes the data structure as described below, and uses Bayesian methods to fit a fit a spline through the population numbers and a hierarchical model for the trap efficiencies over time. An MCMC object is also created with samples from the posterior.

Usage

```
LP_BTSPAS_fit_Diag(
  data,
  p_model = ~1,
  p_model_cov = NULL,
  jump.after = NULL,
  logitP.fixed = NULL,
  logitP.fixed.values = NULL,
  InitialSeed = ceiling(stats::runif(1, min = 0, max = 1e+06)),
  n.chains = 3,
  n.iter = 2e+05,
  n.burnin = 1e+05,
  n.sims = 2000,
  trace = FALSE,
  remove_MCMC_files = TRUE,
  quietly = FALSE
)
```

Arguments

data	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.
p_model	Model for the captured probabilities. This can reference other variables in the data frame, plus a special reserved term ..time to indicate a time dependence in the capture probabilities. For example, p_model=~1 would indicate that the capture probabilities are equal across the sampling events; p_model=~..time would indicate that the capture probabilities vary by sampling events; p_model=~sex*..time would indicate that the capture probabilities vary across all combination of sampling events (..time) and a stratification variable (sex). The sex variable also needs to be in the data frame. For some models (e.g., tag loss models), the ..time variable cannot be used because the conditional models (on being captured at the second event) end up having only have one capture probability (e.g., only for event 1) because of the conditioning process.
p_model_cov	Data frame with covariates for the model for prob capture at second sampling event. If this data frame is given, it requires one line for each of the temporal strata at the second sampling event (even if missing in the data that has the capture histories) with one variable being ..time to represent the second temporal stratum.
jump.after	A numeric vector with elements belonging to time. In some cases, the spline fitting the population numbers should be allowed to jump. For example, the population size could take a jump when hatchery released fish suddenly arrive at the trap. The jumps occur AFTER the strata listed in this argument.
logitP.fixed	A numeric vector (could be null) of the time strata where the logit(P) would be fixed. Typically, this is used when the capture rates for some strata are 0 and logit(P) is set to -10 for these strata. The fixed values are given in logitP.fixed.values
logitP.fixed.values	A numerical vector (could be null) of the fixed values for logit(P) at strata given by logitP.fixed. Typically this is used when certain strata have a 0 capture rate and the fixed value is set to -10 which on the logit scale gives p_i essentially 0. Don't specify values such as -50 because numerical problems could occur when this is converted to the 0-1 scale.
InitialSeed	Numeric value used to initialize the random numbers used in the MCMC iterations.
n.chains	Number of chains to fit in the MCMC
n.iter	Total number of iterations
n.burnin	Number of burnin iterations
n.sims	Total number of simulations to keep in output (implies a thinning)
trace	Internal tracing flag.

`remove_MCMC_files`
: Should the temporary MCMC files (init.txt, data.text, model.txt, CODA*txt) removed after the fit.

`quietly`
: Suppress all console messages that occur during the fit. This includes the progress bar when a model that requires MCMC is fit (*LP_BTSPAS_fit_Diag* and *LP_BTSPAS_fit_NonDiag*), or a trace of the likelihood during the fit (*LP_SPAS_fit*).

Details

Use the `Petersen::LP_BTSPAS_fit_NonDiag` function for cases where recaptures take place outside the stratum of release.

The frequency variable (`freq` in the `data` argument) is the number of animals with the corresponding capture history.

Capture histories (`cap_hist` in the `data` argument) are character values of the format `xx..yy` is a capture_history where `xx` and `yy` are the temporal stratum (e.g., julian week) and `'..'` separates the two temporal strata. If a fish is released in temporal stratum and never captured again, then `yy` is set to 0; if a fish is newly captured in temporal stratum `yy`, then `xx` is set to zero. For example, a capture history of `23..23` indicates animals released in temporal stratum 23 and recaptured in temporal stratum 23; a capture history of `23..00` indicates animals released in temporal stratum 23 and never seen again; a capture history of `00..23` indicates animals newly captured in temporal stratum 23 at the second sampling event.

In the diagonal case, no fish should move between temporal strata.

It is not necessary to label the temporal strata starting at 1; BTSPAS will treat the smallest value of the temporal strata seen as the first stratum and will interpolate for temporal strata without any data. Temporal strata labels should be numeric, i.e., do NOT use A, B, C etc.

Value

An list object of class *LP_BTSPAS_fit_Diag* with the following elements

- **summary** A data frame with the information on the number of observations in the fit
- **data** Data used in the fit
- **p_model**, **p_model_cov** Information on modelling the capture probabilities at the second occasion
- **fit** n MCMC object with samples from the posterior distribution. A series of graphs and text file are also created with summary information. Refer to the BTSPAS package for more details.
- **datetime** Date and time the fit was done

References

Bonner, S. J. and Schwarz, C. J. (2021). BTSPAS: Bayesian Time Stratified Petersen Analysis System.R package version 2021.11.2.

Bonner, S. J., & Schwarz, C. J. (2011). Smoothing population size estimates for Time-Stratified Mark-Recapture experiments Using Bayesian P-Splines. Biometrics, 67, 1498-1507. doi:10.1111/j.15410420.2011.01599.x

Examples

```
# NOTE. To keep execution time to a small value as required by CRAN
# I've made a very small example.
# Additionally, I've set the number of MCMC chains, iterations, burning, simulation to save to
# small values. Proper mixing may not have occurred yet.
# When using this routine, you likely want to the use the default values
# for these MCMC parameters.

data(data_btspas_diag1)
# extract the strata of interest
temp<- cbind(data_btspas_diag1,
             split_cap_hist( data_btspas_diag1$cap_hist,
                             sep="..", make.numeric=TRUE))
# only use data up to week 10 to keep example small
temp <- temp[ temp$t1 %in% 0:10 & temp$t2 %in% 0:10,]

fit <- Petersen::LP_BTSPAS_fit_Diag(
  temp,
  p_model=~1,
  InitialSeed=23943242,
  # the number of chains and iterations are too small to be useful
  # they are set to a small number to pare execution time to <5 seconds for an example
  n.chains=2, n.iter=20000, n.burnin=1000, n.sims=100,
  quietly=TRUE
)
fit$summary

# now get the estimates of abundance
est <-  Petersen::LP_BTSPAS_est (fit)
est$summary
```

LP_BTSPAS_fit_NonDiag *Wrapper (*_fit) to call the Time Stratified Petersen Estimator with NON-Diagonal Entries function in BTSPAS.*

Description

Takes the data structure as described below, and uses Bayesian methods to fit a fit a spline through the population numbers and a hierarchical model for the trap efficiency over time. An MCMC object is also created with samples from the posterior.

Usage

```
LP_BTSPAS_fit_NonDiag(
  data,
  p_model = ~1,
```

```
  p_model_cov = NULL,
  jump.after = NULL,
  logitP.fixed = NULL,
  logitP.fixed.values = NULL,
  InitialSeed = ceiling(stats::runif(1, min = 0, max = 1e+06)),
  n.chains = 3,
  n.iter = 2e+05,
  n.burnin = 1e+05,
  n.sims = 2000,
  trace = FALSE,
  remove_MCMC_files = TRUE,
  quietly = FALSE
)
```

Arguments

`data`	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.
`p_model`	Model for the captured probabilities. This can reference other variables in the data frame, plus a special reserved term `..time` to indicate a time dependence in the capture probabilities. For example, `p_model=~1` would indicate that the capture probabilities are equal across the sampling events; `p_model=~..time` would indicate that the capture probabilities vary by sampling events; `p_model=~sex*..time` would indicate that the capture probabilities vary across all combination of sampling events (`..time`) and a stratification variable (`sex`). The `sex` variable also needs to be in the data frame. For some models (e.g., tag loss models), the `..time` variable cannot be used because the conditional models (on being captured at the second event) end up having only have one capture probability (e.g., only for event 1) because of the conditioning process.
`p_model_cov`	Data frame with covariates for the model for prob capture at second sampling event. If this data frame is given, it requires one line for each of the temporal strata at the second sampling event (even if missing in the `data` that has the capture histories) with one variable being `..time` to represent the second temporal stratum.
`jump.after`	A numeric vector with elements belonging to `time`. In some cases, the spline fitting the population numbers should be allowed to jump. For example, the population size could take a jump when hatchery released fish suddenly arrive at the trap. The jumps occur AFTER the strata listed in this argument.
`logitP.fixed`	A numeric vector (could be null) of the time strata where the logit(P) would be fixed. Typically, this is used when the capture rates for some strata are 0 and logit(P) is set to -10 for these strata. The fixed values are given in `logitP.fixed.values`

`logitP.fixed.values`
: A numerical vector (could be null) of the fixed values for logit(P) at strata given by logitP.fixed. Typically this is used when certain strata have a 0 capture rate and the fixed value is set to -10 which on the logit scale gives p_i essentially 0. Don't specify values such as -50 because numerical problems could occur when this is converted to the 0-1 scale.

`InitialSeed`
: Numeric value used to initialize the random numbers used in the MCMC iterations.

`n.chains`
: Number of chains to fit in the MCMC

`n.iter`
: Total number of iterations

`n.burnin`
: Number of burnin iterations

`n.sims`
: Total number of simulations to keep in output (implies a thinning)

`trace`
: Internal tracing flag.

`remove_MCMC_files`
: Should the temporary MCMC files (init.txt, data.text, model.txt, CODA*txt) removed after the fit.

`quietly`
: Suppress all console messages that occur during the fit. This includes the progress bar when a model that requires MCMC is fit (*LP_BTSPAS_fit_Diag* and *LP_BTSPAS_fit_NonDiag*), or a trace of the likelihood during the fit (*LP_SPAS_fit*).

Details

Use the `Petersen::LP_BTSPAS_fit_Diag` function for cases where recaptures take place in a single stratum (diagonal case).

The frequency variable (`freq` in the `data` argument) is the number of animals with the corresponding capture history.

Capture histories (`cap_hist` in the `data` argument) are character values of the format `xx..yy` is a capture_history where `xx` and `yy` are the temporal stratum (e.g., julian week) and `'..'` separates the two temporal strata. If a fish is released in temporal stratum and never captured again, then `yy` is set to 0; if a fish is newly captured in temporal stratum `yy`, then `xx` is set to zero. For example, a capture history of `23..23` indicates animals released in temporal stratum 23 and recaptured in temporal stratum 23; a capture history of `23..00` indicates animals released in temporal stratum 23 and never seen again; a capture history of `00..23` indicates animals newly captured in temporal stratum 23 at the second sampling event.

In the non-diagonal case, fish are allowed to move among temporal strata.

It is not necessary to label the temporal strata starting at 1; BTSPAS will treat the smallest value of the temporal strata seen as the first stratum and will interpolate for temporal strata without any data. Temporal strata labels should be numeric, i.e., do NOT use A, B, C etc.

Value

An list object of class *LP_BTSPAS_fit_Diag* with the following elements

- **summary** A data frame with the information on the number of observations in the fit
- **data** Data used in the fit

- **p_model**, **p_model_cov** Information on modelling the capture probabilities at the second occasion
- **fit** n MCMC object with samples from the posterior distribution. A series of graphs and text file are also created with summary information. Refer to the BTSPAS package for more details.
- **datetime** Date and time the fit was done

References

Bonner, S. J. and Schwarz, C. J. (2021). BTSPAS: Bayesian Time Stratified Petersen Analysis System.R package version 2021.11.2.

Bonner, S. J., & Schwarz, C. J. (2011). Smoothing population size estimates for Time-Stratified Mark-Recapture experiments Using Bayesian P-Splines. Biometrics, 67, 1498-1507. doi:10.1111/j.15410420.2011.01599.x

Examples

```
# NOTE. To keep execution time to a small value as required by CRAN
# I've made a very small example.
# Additionally, I've set the number of MCMC chains, iterations, burning, simulation to save to
# small values. Proper mixing may not have occurred yet.
# When using this routine, you likely want to the use the default values
# for these MCMC parameters.
data(data_btspas_nondiag1)
temp<- cbind(data_btspas_nondiag1,
             split_cap_hist( data_btspas_nondiag1$cap_hist,
                             sep="..", make.numeric=TRUE))
xtabs(~t1, data=temp)

# only use data up to week 10 to keep example small
temp <- temp[ temp$t1 %in% c(0, 27:32) & temp$t2 %in% c(0, 27:32),]

fit <- Petersen::LP_BTSPAS_fit_NonDiag(
  temp,
  p_model=~1,
  InitialSeed=23943242,
  # the number of chains and iterations are too small to be useful
  # they are set to a small number to pare execution time to <5 seconds for an example
  n.chains=2, n.iter=20000, n.burnin=1000, n.sims=100,
  quietly=TRUE
)
fit$summary

# now get the estimates of abundance
est <-  Petersen::LP_BTSPAS_est (fit)
est$summary
```

`LP_CL_fit` *Fit the Chen-Lloyd model to estimate abundance using a non-parametric smoother for a covariates*

Description

This will take a data frame of capture histories, frequencies, and a covariates and will do a non-parametric smoother for the detection probabilities as a function of the covariates and use this to estimate the population size.

Usage

```
LP_CL_fit(
  data,
  covariate,
 centers = hist(data[, covariate, drop = TRUE], breaks = "Sturges", plot = FALSE)$mids,
  h1 = (centers[2] - centers[1]) * 0.75,
  h2 = (centers[2] - centers[1]) * 0.75,
  conf_level = 0.95
)
```

Arguments

`data`	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.
`covariate`	Name of continuous covariate that influences capture probabilities at each event
`centers`	Centers of bins to group the covariates. We suggest no more than 30 bins in total with fewer bins with smaller sample sizes. Of course with smaller sample sizes, a simple stratified estimator may be easier to use.
`h1, h2`	Standard deviation of normal kernel for first sampling event. This should be between 1/2 and the 1.5x the bin width. Larger values imply more smoothing. Smaller values imply less smoothing.
`conf_level`	The expected coverage for confidence intervals on N.

Details

The frequency variable (`freq` in the `data` argument) is the number of animals with the corresponding capture history.

Capture histories (`cap_hist` in the `data` argument) are character values of length 2.

- **10** Animals tagged but never seen again.
- **11** Animals tagged and recaptured and tag present at event 2.
- **01** Animals captured at event 2 that appear to be untagged.

Value

An list object of class *LP_CL_fit* with abundance estimates and other information with the following elements

- **summary** A data frame with the estimates of abundance, SE, and CI
- **fit** Details on the Chen and Lloyd fit including the smoothed estimates of catchability, estimates abundance by category classes, estimates of total abundance, plots of the estimated abundance curve and catchability curves, etc.
- **datetime** Date and time the fit was done

References

SX Chen, CJ Lloyd (2000). A nonparametric approach to the analysis of two-stage mark-recapture experiments. Biometrika, 87, 633–649. doi:10.1093/biomet/87.3.633.

Examples

```
library(Petersen)
data(data_NorthernPike)
res <- LP_CL_fit(data_NorthernPike, "length")
res$summary
```

LP_est *Estimate abundance after the LP conditional likelihood fit.*

Description

This will take a previous fit and return estimates of abundance. The population abundance is estimated using a Horvitz-Thompson type estimator and the user can request abundance estimates for sub-sets of the population.

Usage

```
LP_est(LP_fit, N_hat = ~1, conf_level = 0.95, trace = FALSE)
```

Arguments

LP_fit	A result of an LP_fit() call.
N_hat	A formula requesting which abundance estimates should be formed. The formula are expanded against the data frame to determine which records form part of the abundance estimate. The formula is evaluated against the data frame used in the fit using the model.matrix() function, and each column of the model matrix is used to form an estimate. Some familiarity on how model.matrix() generates the model matrix of coefficients used in the expansion is needed. For example N_hat=~1 creates a model matrix with 1 column (representing the intercept) and so requests abundance

over the entire population; Specifying N_hat=~-1+Sex creates a model matrix with 2 columns (one for each sex) consisting of 0/1 depending if that row of the data frame is M/F. Hence, two abundance estimates (one for each sex) is computed. On the other hand, N_hat=Sex generates a model matrix where the first column is all 1's, and a second column which is 0/1 depending if the row in the data frame is the "second" sex. Hence, this will request the overall abundance (over both sexes) and the estimate of abundance for the second sex.

In addition to the variables in the data frame, special variables include ..EF to allow access to the expansion factor so you can request a "truncated" Horvitz-Thompson estimator using N_hat=~-1+I(as.numeric(..EF<1000)) to only use those animals with expansion factors less than 1000 in forming the estimate.

conf_level — The expected coverage for confidence intervals on N.

trace — If trace flag is set in call when estimating functions

Value

An list object with abundance estimates and other information with the following elements

- **summary** Data frame with abundance estimates, their SE, and CIs as requested
- **detail** List with many components, including the rawdata, model fitting information, observed and expected values, residual plot, etc
- **datetime** Date and time the estimation was done from the fit.

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

Examples

```
# fit a simple Petersen model and get the estimated abundance
data(data_rodli)
fit <- Petersen::LP_fit(data=data_rodli, p_model=~..time)
fit$summary
# Now to get the estimated abundance
est <- Petersen::LP_est(fit, N_hat=~1)
est$summary

# repeat the fit with the Chapman correction
# we add an additional animal with history 11
rodli.chapman <- plyr::rbind.fill(data_rodli,
                                  data.frame(cap_hist="11",
                                             freq=1,
                                             comment="Added for Chapman"))

rodli.chapman
fit.chapman <- Petersen::LP_fit(data=rodli.chapman, p_model=~..time)
fit.chapman$summary
# Now to get the estimated abundance
est.chapman <- Petersen::LP_est(fit.chapman, N_hat=~1)
est.chapman$summary
```

```
# Example of simple stratification (by sex)
data(data_NorthernPike)
nop.red <- plyr::ddply(data_NorthernPike, c("cap_hist","Sex"), plyr::summarize,
                       freq=sum(freq))
nop.red # reduced capture history to speed execution time of example

# Fit the various models
nop.fit.sex.time   <- Petersen::LP_fit(nop.red, p_model=~-1+Sex:..time)
nop.fit.sex.time$summary

# estimate of overall abundance
nop.est.ALL   <- Petersen::LP_est(nop.fit.sex.time, N=~1)
nop.est.ALL$summary

# estimate of abundance for each sex
nop.est.by.sex <- Petersen::LP_est(nop.fit.sex.time, N=~-1+Sex)
nop.est.by.sex$summary


# Refer to vignettes for example using continuous variable (e.g. length) to model catchability
```

LP_est_adjust *Estimate abundance after empirical adjustments for various factors.*

Description

This will take a previous fit and return estimates of abundance after making various empirical adjustments

Usage

```
LP_est_adjust(
  N_hat,
  N_hat_SE,
  conf_level = 0.95,
  tag.retention.est = 1,
  tag.retention.se = 0,
  tag.reporting.est = 1,
  tag.reporting.se = 0,
  n1.adjust.est = 1,
  n1.adjust.se = 0,
  n2.adjust.est = 1,
  n2.adjust.se = 0,
  m2.adjust.est = 1,
  m2.adjust.se = 0,
  n.sim = 10000,
  trace = FALSE
)
```

Arguments

N_hat	Estimate of N that will be adjusted
N_hat_SE	SE of the N_hat
conf_level	The expected coverage for confidence intervals on N.
tag.retention.est	Estimated tag retention probability
tag.retention.se	Estimated SE of tag retention probability
tag.reporting.est	Estimated tag reporting probability
tag.reporting.se	Estimated SE of tag reporting probability
n1.adjust.est	Adjustment to "n1". This should typically be a ratio of new n1 to old n1
n1.adjust.se	Adjustment to "n1" uncertainty
n2.adjust.est	Adjustment to "n2" This should typically be a ratio of new n2 to old n2
n2.adjust.se	Adjustment to "n2" uncertainty
m2.adjust.est	Adjustment to "m2" This should typically be a ratio of new m2 to old m2
m2.adjust.se	Adjustment to "m2" uncertainty
n.sim	Number of simulation runs to make
trace	If trace flag is set in call when estimating functions

Details

The estimate and SE are converted to a beta distribution for adjustment factors between 0 and 1 with equivalent mean and SD as the estimate and se. The estimate and se are used in normal distribution for adjustment factors for n1, n2, and m2. These adjustment factors are then simulated a large number of times and then multiplied together to get the mean and sd of all adjustments applied together. Then the abundance is simulated (on the log scale), the product taken, and the mean, sd, ci estimated directly.

Value

An list object with a summary data frame and a data frame with the adjustment factors with the following objects **summary** A data frame with the adjusted abundance estimates, SE, and CI **adjustment** a data frame showing the adjustment factors applied for tag retention, tag reporting, n1 n2 or m2. **datetime** Date and time the adjustment was done

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

Examples

```
data(data_rodli)
rodli.fit <- Petersen::LP_fit(data=data_rodli, p_model=~..time)
rodli.est <- Petersen::LP_est(rodli.fit)
res <- Petersen::LP_est_adjust(rodli.est$summary$N_hat, rodli.est$summary$N_hat_SE,
          tag.retention.est=.90, tag.retention.se=.05)
res$summary
```

LP_fit *Fit a Lincoln-Petersen Model using conditional likelihood*

Description

This will take a data frame of capture histories, frequencies, and additional covariates (e.g., strata and/or continuous covariates) and the model for the capture probabilities and will use conditional likelihood (Huggins, 1989) to fit the model. The population abundance is estimated using a Horvitz-Thompson type estimator and the user can request abundance estimates for sub-sets of the population.

Usage

```
LP_fit(data, p_model = ~..time, p_beta.start = NULL, trace = FALSE)
```

Arguments

`data`	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.
`p_model`	Model for the captured probabilities. This can reference other variables in the data frame, plus a special reserved term `..time` to indicate a time dependence in the capture probabilities. For example, `p_model=~1` would indicate that the capture probabilities are equal across the sampling events; `p_model=~..time` would indicate that the capture probabilities vary by sampling events; `p_model=~sex*..time` would indicate that the capture probabilities vary across all combination of sampling events (`..time`) and a stratification variable (`sex`). The `sex` variable also needs to be in the data frame. For some models (e.g., tag loss models), the `..time` variable cannot be used because the conditional models (on being captured at the second event) end up having only have one capture probability (e.g., only for event 1) because of the conditioning process.
`p_beta.start`	Initial values for call to optimization routine for the beta parameters (on the logit scale). The values will be replicated to match the number of initial beta parameters needed. Some care is needed here!
`trace`	If trace flag is set in call when estimating functions

Details

The frequency variable (`freq` in the `data` argument) is the number of animals with the corresponding capture history.

Capture histories (`cap_hist` in the `data` argument) are character values of length 2.

- **10** Animals tagged but never seen again.
- **11** Animals tagged and recaptured and tag present at event 2.
- **01** Animals captured at event 2 that appear to be untagged.

Value

An list object of class *LP_fit* with abundance estimates and other information with the following elements

- **summary** A data frame with the model for the capture probabilities; the conditional log-likelihood; the number of parameters; the number of parameters, and method used to fit the model
- **data** A data frame with the raw data used in the fit
- **fit** Results of the fit from the optimizer
- **datetime** Date and time the fit was done

After the fit is done, use the *LP_est()* function to get estimates of abundance.

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

References

Huggins, R. M. 1989. On the Statistical Analysis of Capture Experiments. Biometrika 76: 133–40.

Examples

```
# fit a simple Petersen model and get the estimated abundance
data(data_rodli)
fit <- Petersen::LP_fit(data=data_rodli, p_model=~..time)
fit$summary
# Now to get the estimated abundance
est <- Petersen::LP_est(fit, N_hat=~1)
est$summary

# repeat the fit with the Chapman correction
# we add an additional animal with history 11
rodli.chapman <- plyr::rbind.fill(data_rodli,
                                  data.frame(cap_hist="11",
                                             freq=1,
                                             comment="Added for Chapman"))

rodli.chapman
```

```
fit.chapman <- Petersen::LP_fit(data=rodli.chapman, p_model=~..time)
fit.chapman$summary
# Now to get the estimated abundance
est.chapman <- Petersen::LP_est(fit.chapman, N_hat=~1)
est.chapman$summary


# Example of simple stratification (by sex)
data(data_NorthernPike)
nop.red <- plyr::ddply(data_NorthernPike, c("cap_hist","Sex"), plyr::summarize,
                       freq=sum(freq))
nop.red # reduced capture history to speed execution time of example

# Fit the various models
nop.fit.sex.time   <- Petersen::LP_fit(nop.red, p_model=~-1+Sex:..time)
nop.fit.sex.time$summary

# estimate of overall abundance
nop.est.ALL   <- Petersen::LP_est(nop.fit.sex.time, N=~1)
nop.est.ALL$summary

# estimate of abundance for each sex
nop.est.by.sex <- Petersen::LP_est(nop.fit.sex.time, N=~-1+Sex)
nop.est.by.sex$summary


# Refer to vignettes for example using continuous variable (e.g. length) to model catchability
```

`LP_for_rev_fit` *Fit a combined FORWARD and REVERSE simple Lincoln-Petersen Model using pseudo-likelihood*

Description

EXPERIMENTAL. This will take a data frame of capture histories, frequencies, and additional covariates (e.g., strata and/or continuous covariates) for a simple forward Petersen estimate plus estimates of escapement and associated stock proportions with SE for backwards estimation. DO NOT USE YET.

Usage

```
LP_for_rev_fit(
  data,
  E,
  E.SE,
  G,
  G.SE,
  min.G = 0.01,
  n.boot = 100,
```

```
  trace = FALSE
)
```

Arguments

data	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.
E	Escapement at one or more terminal areas. E, E.SE, G, G.SE must all have the same length
E.SE	SE of the estimates of escapement
G	Estimated proportion of the stock at the first capture location estimated using GSI and other methods
G.SE	SE of the estimated stock proportion.
min.G	Miniumum acceptable stock proportion during the bootstrap estimation of uncertainty
n.boot	Number of bootstrap samples used to estimate the uncertainty
trace	Should intermediate tracing be enabled (e.g. browser() stops)

Details

The frequency variable (`freq` in the `data` argument) is the number of animals with the corresponding capture history.

Capture histories (`cap_hist` in the `data` argument) are character values of length 2.

- **10** Animals tagged but never seen again.
- **11** Animals tagged and recaptured and tag present at event 2.
- **01** Animals captured at event 2 that appear to be untagged.

A pseudo-likelihood is constructed consisting of the usual likelihood for a forward capture recapture and marginal likelihoods for each of the escapement (E) and stock proportions (G) point estimates. I have not integrated over the uncertainty in G and E.

Value

An list object of class *LP_for_rev_est* with abundance estimates and measures of uncertainty

- **summary** A data frame with the pseudo-likelihood value, abundance estimates and SE
- **data** A data frame with the raw data used in the fit
- **fit** Results of the fit from the optimizer
- **datetime** Date and time the fit was done

Unlike other routine, it is not necessary to use a *XX_est()* function to get estimates of abundance.

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

Examples

```
# Example of combined forward and reverse MR

# get some data
n1 <-  500
n2 <- 1500
m2 <-  150

f.data <- data.frame(cap_hist=c("10","11","01"), freq=c(n1 - m2,  m2,  n2 - m2))
f.data

E = 1500
E.SE = 150
G = .2
G.SE = .05

res <- LP_for_rev_fit(data=f.data,
                      E=E,
                      E.SE=E.SE,
                      G=G,
                      G.SE=G.SE)
```

LP_IS_est *Estimate abundance after the LP_IS conditional likelihood fit.*

Description

This will take a previous fit and return estimates of abundance. The population abundance is estimated using a Horvitz-Thompson type estimator and the user can request abundance estimates for sub-sets of the population

Usage

```
LP_IS_est(LP_IS_fit, N_hat = ~1, conf_level = 0.95, trace = FALSE)
```

Arguments

LP_IS_fit	A result of an LP_IS_fit() call.
N_hat	A formula requesting which abundance estimates should be formed. The formula are expanded against the data frame to determine which records form part of the abundance estimate. The formula is evaluated against the data frame used in the fit using the model.matrix() function, and each column of the model matrix is used to form an estimate.

Some familiarity on how `model.matrix()` generates the model matrix of coefficients used in the expansion is needed. For example `N_hat=~1` creates a model matrix with 1 column (representing the intercept) and so requests abundance over the entire population; Specifying `N_hat=~-1+Sex` creates a model matrix with 2 columns (one for each sex) consisting of 0/1 depending if that row of the data frame is M/F. Hence, two abundance estimates (one for each sex) is computed. On the other hand, `N_hat=Sex` generates a model matrix where the first column is all 1's, and a second column which is 0/1 depending if the row in the data frame is the "second" sex. Hence, this will request the overall abundance (over both sexes) and the estimate of abundance for the second sex.

In addition to the variables in the `data` frame, special variables include `..EF` to allow access to the expansion factor so you can request a "truncated" Horvitz-Thompson estimator using `N_hat=~-1+I(as.numeric(..EF<1000))` to only use those animals with expansion factors less than 1000 in forming the estimate.

`conf_level` The expected coverage for confidence intervals on N.

`trace` If trace flag is set in call when estimating functions

Value

An list object with abundance estimates and other information with the following elements

- **summary** Data frame with abundance estimates, their SE, and CIs as requested
- **detail** List with many components, including the rawdata, model fitting information, observed and expected values, residual plot, etc
- **datetime** Date and time the estimation was done from the fit.

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

Examples

```
data(data_wae_is_short)
fit <- Petersen::LP_IS_fit(data=data_wae_is_short, p_model=~..time)
fit$summary
est <- LP_IS_est(fit, N_hat=~1)
est$summary
```

LP_IS_fit *Fit a Lincoln-Petersen Model with incomplete stratification*

Description

In some LP studies, stratification is only done on a random sample of unmarked fish, e.g., only a sample of fish is sexed. Is is known as incomplete stratification. This is a wrapper to the published code for the case of stratification by a discrete covariate. At the moment, no other covariates are allowed, but see the published code.

Usage

```
LP_IS_fit(
  data,
  p_model,
  theta_model = ~-1 + ..time,
  lambda_model = ~-1 + ..cat,
  logit_p_offset = 0,
  logit_theta_offset = 0,
  logit_lambda_offset = 0,
  cat.unknown = "U",
  p_beta.start = NULL,
  trace = FALSE,
  control.optim = list(trace = 0, maxit = 1000)
)
```

Arguments

`data`	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting. At the moment, you are not allowed to use these covariates to used in the modeling process, but see the published code for more details.
`p_model`	Model for the captured probabilities. This can reference other variables in the data frame, plus a special reserved term `..time` to indicate a time dependence in the capture probabilities. For example, `p_model=~1` would indicate that the capture probabilities are equal across the sampling events; `p_model=~..time` would indicate that the capture probabilities vary by sampling events; `p_model=~sex*..time` would indicate that the capture probabilities vary across all combination of sampling events (`..time`) and a stratification variable (`sex`). The `sex` variable also needs to be in the data frame. For some models (e.g., tag loss models), the `..time` variable cannot be used because the conditional models (on being captured at the second event) end up having only have one capture probability (e.g., only for event 1) because of the conditioning process.
`theta_model`	Model for theta (sampling fraction). Usually, this is set to be different for the two sampling occasions, but you can constrain this to have equal sampling fractions at both occasions.
`lambda_model`	Model for lambda category proportions. Usually this is set to different for the categories but you can constrain this with a null matrix and the `logit_lamba_offset` parameter
`logit_p_offset`	Used to fix capture probabilities at known values (seldom useful). Logit(p)=p_design %*% beta_p + logit_p_offset.
`logit_theta_offset`	Used to fix sampling fractions at known values (seldom useful). logit(theta) = theta_design %*% beta_theta + logit_theta_offset

`logit_lambda_offset`
: Used to fix the sex ratio as a known value (e.g. .50) using logit(lambda) = lambda_design %*% beta_lambda + logit_lambda_offset. Set the design matrix to a matrix with all zeros. Notice that because the lambda proportions must sum to 1, only specify an offset matrix that is number of categories -1.

`cat.unknown`
: Value of character used to indicate the unknown stratum in the capture histories. Currently, this is fixed to "U" regardless of what is specified.

`p_beta.start`
: Initial values for call to optimization routine for the beta parameters (on the logit scale). The values will be replicated to match the number of initial beta parameters needed. Some care is needed here!

`trace`
: If trace flag is set in call when estimating functions

`control.optim`
: Control values passed to optim() optimizer.

Details

The frequency variable (`freq` in the `data` argument) is the number of animals with the corresponding capture history.

Capture histories (`cap_hist` in the `data` argument) are character values of length 2. The strata values are single character values with "U" typically representing a fish not measured for the stratification variable. For example, consider the case where only a sample of unmarked fish are examined for sex (M or F). Possible capture histories are:

- **M0** Animals tagged and sexed as male but never seen again.
- **MM** Animals tagged and sexed as male and recaptured and tag present at event 2.
- **0M** Animals captured at event 2 that appears to be untagged and was sexed as male.
- **F0** Animals tagged and sexed as female but never seen again.
- **FF** Animals tagged and sexed as female and recaptured and tag present at event 2.
- **0F** Animals captured at event 2 that appears to be untagged and was sexed as female.
- **U0** Animals tagged and not sexed but never seen again.
- **UU** Animals tagged and not sexed and recaptured and tag present at event 2.
- **0U** Animals captured at event 2 that appears to be untagged and was not sexed.

Capture histories such as **UF** or **UM** are not allowed since only UNTAGGED animals are examined and sexed. Similarly, capture histories such as **FM** or **MF** are not allowed.

Value

An list object of class *LP_IS_fit* with abundance estimates and other information with the following elements

- **summary** A data frame with the model for the capture probabilities, the sampling fractions at each capture occasion, and the category proportions; the conditional log-likelihood; the number of parameters; the number of parameters, and method used to fit the model
- **data** A data frame with the raw data used in the fit
- **fit** Results of the fit including the estimates, SE, vcov, etc.
- **fit.call** Arguments used in the fit
- **datetime** Date and time the fit was done

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

References

Premarathna, W.A.L., Schwarz, C.J., Jones, T.S. (2018) Partial stratification in two-sample capture–recapture experiments. Environmetrics, 29:e2498. doi:10.1002/env.2498

Examples

```
data(data_wae_is_short)
fit <- Petersen::LP_IS_fit(data=data_wae_is_short, p_model=~..time)
fit$summary
est <- LP_IS_est(fit, N_hat=~1)
est$summary
```

`LP_IS_print` *Print the results from a fit a Lincoln-Petersen Model with incomplete stratification*

Description

Print the results from a fit a Lincoln-Petersen Model with incomplete stratification

Usage

```
LP_IS_print(IS.results)
```

Arguments

`IS.results`	Results from fitting an incomplete stratification model

Value

A nicely formatted report showing the results of the fit.

- Model information (summary of arguments, model name, negative log-likelihood, number of parameters, AICc)
- Raw data used in the fit (history, frequency,categories)
- Initial values used in the optimization of the likelihood for the parameters
- Design matrix and offset values for the parameters
- Maximum likelihood estimates for the parameters and estimated abundance by category
- SE for the above
- Observed and expected counts for each capture history
- Residual plot constructed from the previous observed and expected counts

Examples

```
data(data_wae_is_short)
res <- Petersen::LP_IS_fit(data=data_wae_is_short, p_model=~-1 + ..cat:..time)
LP_IS_print(res)
```

`LP_modavg` *Create an table of individual estimates and the model averaged values*

Description

This will take a series of LP fits and computes the model averages for each set of N_hat

Usage

```
LP_modavg(..., N_hat = ~1, conf_level = 0.95)
```

Arguments

`...`	Series of LP fits
`N_hat`	A formula requesting which abundance estimates should be formed. The formula are expanded against the data frame to determine which records form part of the abundance estimate. The formula is evaluated against the `data` frame used in the fit using the `model.matrix()` function, and each column of the model matrix is used to form an estimate. Some familiarity on how `model.matrix()` generates the model matrix of coefficients used in the expansion is needed. For example `N_hat=~1` creates a model matrix with 1 column (representing the intercept) and so requests abundance over the entire population; Specifying `N_hat=~-1+Sex` creates a model matrix with 2 columns (one for each sex) consisting of 0/1 depending if that row of the data frame is M/F. Hence, two abundance estimates (one for each sex) is computed. On the other hand, `N_hat=Sex` generates a model matrix where the first column is all 1's, and a second column which is 0/1 depending if the row in the data frame is the "second" sex. Hence, this will request the overall abundance (over both sexes) and the estimate of abundance for the second sex. In addition to the variables in the `data` frame, special variables include `..EF` to allow access to the expansion factor so you can request a "truncated" Horvitz-Thompson estimator using `N_hat=~-1+I(as.numeric(..EF<1000))` to only use those animals with expansion factors less than 1000 in forming the estimate.
`conf_level`	The expected coverage for confidence intervals on N.

Value

An data frame with model averaged values for abundance

Examples

```
data(data_rodli)
mt <- Petersen::LP_fit(data=data_rodli, p_model=~..time)
m0 <- Petersen::LP_fit(data=data_rodli, p_model=~1)
Petersen::LP_modavg(m0,mt)
```

LP_SPAS_est *Extract estimates of abundance after SPAS fit*

Description

This will take a previous fit and return estimates of abundance.

Usage

```
LP_SPAS_est(LP_SPAS_fit, conf_level = 0.95, trace = FALSE)
```

Arguments

LP_SPAS_fit	A result of an call to fitting at SPAS object.
conf_level	The expected coverage for confidence intervals on N.
trace	If trace flag is set in call when estimating functions

Value

An list object with abundance estimates and other information with the following elements

- **summary** Data frame with abundance estimates, their SE, and CIs as requested
- **datetime** Date and time the estimation was done from the fit.

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

Examples

```
data(data_spas_harrison)
fit <- Petersen::LP_SPAS_fit(data=data_spas_harrison,
                             model.id="Pooling rows 5/6",
                             row.pool.in=c(1,2,3,4,56,56),
                             col.pool.in=c(1,2,3,4,5,6))
fit$summary
est <- Petersen::LP_SPAS_est(fit)
est$summary
```

LP_SPAS_fit *Fit a Stratified-Petersen SPAS model.*

Description

This function is a wrapper to fits a SPAS model(Schwarz, 2023; Schwarz and Taylor, 1998). Consult the SPAS package for more details.

Usage

```
LP_SPAS_fit(
  data,
  model.id = "Base model",
  autopool = FALSE,
  row.pool.in = NULL,
  col.pool.in = NULL,
  min.released = 100,
  min.inspected = 50,
  min.recaps = 50,
  min.rows = 1,
  min.cols = 1,
  quietly = FALSE
)
```

Arguments

`data` Data frame containing the variables:

- **cap_hist** Capture history (see details below)
- **freq** Number of times this capture history was observed

plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.

`model.id` Character string identifying the name of the model.

`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.

`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
`quietly`	Suppress all console messages that occur during the fit. This includes the progress bar when a model that requires MCMC is fit (*LP_BTSPAS_fit_Diag* and *LP_BTSPAS_fit_NonDiag*), or a trace of the likelihood during the fit (*LP_SPAS_fit*).

Value

An list object of class *LP_SPAS_fit* with abundance estimates and other information with the following elements

- **summary** A data frame with the model for the capture probabilities; the conditional log-likelihood; the number of parameters; the number of parameters, condition factor of the data matrix, and method used to fit the model
- **data** A data frame with the raw data used in the fit
- **fit** Results of the fit including the estimates, SE, vcov, etc.
- **row.pool.in**, **col.pool.in**, **autopool** Arguments used in the fit to indicate row, column, or automatic pooling used in the fit.
- **datetime** Date and time the fit was done

After the fit is complete, use the *LP_SPAS_est()* function to extract the estimates, and the SPAS::SPAS.print.model() function to get a nicely formatted report on the fit.

References

Schwarz CJ (2023). *SPAS: Stratified-Petersen Analysis System*. R package version 2023.3.31, https://CRAN.R-project.org/package=SPAS.

Schwarz, C. J. and Taylor, C. G. (1998). The use of the stratified-Petersen estimator in fisheries management: estimating the number of pink salmon (Oncorhynchus gorbuscha) that spawn in the Fraser River. Canadian Journal of Fisheries and Aquatic Sciences 55, 281-297. https://doi.org/10.1139/f97-238

Examples

```
data(data_spas_harrison)

fit <- Petersen::LP_SPAS_fit(data=data_spas_harrison,
                             model.id="Pooling rows 5/6",
                             row.pool.in=c(1,2,3,4,56,56),
                             col.pool.in=c(1,2,3,4,5,6),quietly=TRUE)
fit$summary
est <- Petersen::LP_SPAS_est(fit)
est$summary

# make a nice report using the SPAS package functions
```

```
SPAS::SPAS.print.model(fit$fit)
```

`LP_summary_stats` *Compute summary statistics from the capture histories*

Description

This function takes the capture histories and computes n_1, n_2 and m_2.

Usage

```
LP_summary_stats(data)
```

Arguments

`data`	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.

Value

Summary statistics of *n1* (number observed at first occasion), *n2* (number observed at second occasion), and *m2* number recaptured in second occasion.

Examples

```
data(data_rodli)
LP_summary_stats(data_rodli)
```

`LP_test_equal_mf` *Test for equal marked fractions in LP experiment*

Description

This function takes the capture histories and stratification variable and computes the test for equal marked fractions.

Usage

```
LP_test_equal_mf(data, strat_var, do.fisher.test = FALSE)
```

Arguments

data	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.
strat_var	Variable in the dataframe that serves as a stratification variable for contingency table tests of equal marked fraction or equal recapture probability
do.fisher.test	Do a fisher test?

Value

List containing the contingency table and the chi-square test and fisher-exact test

Examples

```
data(data_NorthernPike)
LP_test_equal_mf(data_NorthernPike, "Sex")
```

LP_test_equal_recap *Test for equal recapture probability in LP experiment*

Description

This function takes the capture histories and stratification variable and computes the test for equal recapture probabilities.

Usage

```
LP_test_equal_recap(data, strat_var, do.fisher.test = FALSE)
```

Arguments

data	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.
strat_var	Variable in the dataframe that serves as a stratification variable for contingency table tests of equal marked fraction or equal recapture probability
do.fisher.test	Should a fisher exact test be done?

Value

List containing the contingency table and the chi-square test and fisher-exact test

Examples

```
data(data_NorthernPike)
LP_test_equal_recap(data_NorthernPike, "Sex")
```

`LP_TL_est` *Estimate abundance after the LP_TL (tag loss) conditional likelihood fit.*

Description

This will take a previous fit and return estimates of abundance. The population abundance is estimated using a Horvitz-Thompson type estimator based only on p1 and the user can request abundance estimates for sub-sets of the population.

Usage

```
LP_TL_est(LP_TL_fit, N_hat = ~1, conf_level = 0.95, trace = FALSE)
```

Arguments

`LP_TL_fit`	A result of an LP_TL_fit() call.
`N_hat`	A formula requesting which abundance estimates should be formed. The formula are expanded against the data frame to determine which records form part of the abundance estimate. The formula is evaluated against the `data` frame used in the fit using the `model.matrix()` function, and each column of the model matrix is used to form an estimate. Some familiarity on how `model.matrix()` generates the model matrix of coefficients used in the expansion is needed. For example `N_hat=~1` creates a model matrix with 1 column (representing the intercept) and so requests abundance over the entire population; Specifying `N_hat=~-1+Sex` creates a model matrix with 2 columns (one for each sex) consisting of 0/1 depending if that row of the data frame is M/F. Hence, two abundance estimates (one for each sex) is computed. On the other hand, `N_hat=Sex` generates a model matrix where the first column is all 1's, and a second column which is 0/1 depending if the row in the data frame is the "second" sex. Hence, this will request the overall abundance (over both sexes) and the estimate of abundance for the second sex. In addition to the variables in the `data` frame, special variables include `..EF` to allow access to the expansion factor so you can request a "truncated" Horvitz-Thompson estimator using `N_hat=~-1+I(as.numeric(..EF<1000))` to only use those animals with expansion factors less than 1000 in forming the estimate.
`conf_level`	The expected coverage for confidence intervals on N.
`trace`	If trace flag is set in call when estimating functions

Value

An list object with abundance estimates and other information with the following elements

- **summary** Data frame with abundance estimates, their SE, and CIs as requested
- **detail** List with many components, including the rawdata, model fitting information, etc
- **datetime** Date and time the estimation was done from the fit.

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

Examples

```
data(data_kokanee_tagloss)
fit <- Petersen::LP_TL_fit(data=data_kokanee_tagloss, p_model=~1, rho_model=~1, dt_type="notD")
fit$summary
est <- Petersen::LP_TL_est(fit, N_hat=~1)
est$summary
```

`LP_TL_fit` *Fit a Lincoln-Petersen Model with Tag Loss using conditional likelihood*

Description

This will take a data frame of capture histories, frequencies, and additional covariates (e.g., strata and/or continuous covariates) and the model for p1 and the tag retention probabilities and will use conditional likelihood (conditional on capture at time 2) to fit the model. The population abundance is estimated using a Horvitz-Thompson type estimator and the user can request abundance estimates for sub-sets of the population. Refer to references and appendices in vignettes for more details.

Usage

```
LP_TL_fit(
  data,
  dt_type = NULL,
  p_model,
  rho_model,
  all_beta.start = NULL,
  trace = FALSE
)
```

Arguments

data	Data frame containing the variables: • **cap_hist** Capture history (see details below) • **freq** Number of times this capture history was observed plus any other covariates (e.g. discrete strata and/or continuous covariates) to be used in the model fitting.
dt_type	Double Tag type. Valid values are notD, twoD, and t2perm for two indistinguishable tags; two distinguishable tags, when the second tag is a permanent tag and cannot be lost, respectively.
p_model	Model for the captured probabilities. This can reference other variables in the data frame, plus a special reserved term ..time to indicate a time dependence in the capture probabilities. For example, p_model=~1 would indicate that the capture probabilities are equal across the sampling events; p_model=~..time would indicate that the capture probabilities vary by sampling events; p_model=~sex*..time would indicate that the capture probabilities vary across all combination of sampling events (..time) and a stratification variable (sex). The sex variable also needs to be in the data frame. For some models (e.g., tag loss models), the ..time variable cannot be used because the conditional models (on being captured at the second event) end up having only have one capture probability (e.g., only for event 1) because of the conditioning process.
rho_model	Model for retention probabilities
all_beta.start	Initial values for call to optimization routine for the beta parameters (on the logit scale). The values will be replicated to match the number of initial beta parameters needed. Some care is needed here since the parameter order are for the p1 probabilities and then for the rho probabilities
trace	If trace flag is set in call when estimating functions

Details

The frequency variable (freq in the data argument) is the number of animals with the corresponding capture history.

Capture histories (cap_hist in the data argument) are character values of length 4.

If the tag loss model is two indistinguishable tags (dt_type="notD"), then valid capture histories are:

- **1100** Animals double tagged but never seen again.
- **111X** Animals double tagged, but only 1 tag was present when animal recaptured at second event.
- **1111** Animals double tagged and both tags present when animal recaptured at second event.
- **1000** Animals single tagged and never seen again.
- **0100** Animals single tagged and never seen again.
- **1010** Animals single tagged and recaptured with the single tag.
- **0101** Animals single tagged and recaptured with the single tag.

- **0010** Animals APPARENTLY captured for the first time at event 2. This includes animals that are newly captured, plus fish that were tagged and lost all their tags, and were captured again

If the tag loss model is two distinguishable tags (`dt_type="twoD"`), then valid capture histories are the same as above except the history `111X` is replaced by:

- **1110** Animals double tagged, but only the first of the double tags applied was present when animal recaptured at event 2,
- **1101** Animals double tagged, but only the second of the double tags applied was present when animal recaptured at event 2.

If the second tag is a permanent batch mark (`dt_type="t2perm"`), then valid capture histories are:

- **1P00** Animals double tagged but never seen again.
- **1P0P** Animals double tagged,but non-permanent tag missing when animal recaptured at second event.
- **1P1P** Animals double tagged and both tags present when animal recaptured at second event.
- **1000** Animals single tagged and never seen again.
- **0P00** Animals single tagged with a permanent batch mark only and never seen again.
- **1010** Animals single tagged and recaptured with the single tag.
- **0P0P** Animals single tagged with the permanent batch mark and recaptured with the permanent tag.
- **0010** Animals APPARENTLY captured for the first time at event 2. This includes animals that are newly captured, plus fish that were tagged and lost all their tags, and were captured again

Value

An list object of class *LP_TL_fit-notD* or *LP_TL_fit-twoD*, or *LP_TL_fit-t2per* (depending on the type of double tag) with abundance estimates and other information with the following elements

- **summary** A data frame with the model for the capture probabilities, and tag retention probabilities; the conditional log-likelihood; the number of parameters; the number of parameters, and method used to fit the model
- **data** A data frame with the raw data used in the fit
- **fit** Results of the fit including the estimates, SE, vcov, etc.
- **datetime** Date and time the fit was done

After the fit is complete, use the *LP_TL_est()* function to obtain estimates.

Author(s)

Schwarz, C. J. <cschwarz.stat.sfu.ca@gmail.com>.

References

Seber, G. A. F., and R. Felton. (1981). Tag Loss and the Petersen Mark-Recapture Experiment. Biometrika 68, 211–19.

Hyun, S.-Y., Reynolds.J.H., and Galbreath, P.F. (2012). Accounting for Tag Loss and Its Uncertainty in a Mark–Recapture Study with a Mixture of Single and Double Tags. Transactions of the American Fisheries Society, 141, 11-25 http://dx.doi.org/10.1080/00028487.2011.639263

Examples

```
data(data_kokanee_tagloss)
fit <- Petersen::LP_TL_fit(data=data_kokanee_tagloss, p_model=~1, rho_model=~1, dt_type="notD")
fit$summary
est <- Petersen::LP_TL_est(fit, N_hat=~1)
est$summary
```

`LP_TL_simulate` *Simulate data from a Lincoln-Petersen Model with Tag Loss*

Description

This function creates simulated capture histories for the Lincoln-Petersen model with tag loss.

Usage

```
LP_TL_simulate(
  dt_type = NULL,
  N = 1000,
  cov1 = function(N) {
     rep(1, N)
 },
  cov2 = function(cov1) {
     rep(1, length(cov1))
 },
  p1 = function(cov1, cov2) {
     rep(0.1, length(cov1))
 },
  pST = function(cov1, cov2) {
     rep(0.5, length(cov1))
 },
  pST.1 = function(cov1, cov2) {
     rep(1, length(cov1))
 },
  rho1 = function(cov1, cov2) {
     rep(0.8, length(cov1))
 },
  rho2 = function(cov1, cov2) {
     rep(0.8, length(cov1))
 },
  p2 = function(cov1, cov2) {
     rep(0.1, length(cov1))
 },
  seed = round(1e+08 * runif(1)),
  trace = FALSE
)
```

Arguments

dt_type	Double Tag type. Valid values are notD, twoD, and t2perm for two indistinguishable tags; two distinguishable tags, when the second tag is a permanent tag and cannot be lost, respectively.
N	Population size
cov1	Function to generate first covariate for each member of population as function of N
cov2	Function to generate second covariate for each member of population as function of cov1.
p1	Function to generate P(capture) at event 1 for each member of population as function of cov1,cov2.
pST	Function to generate P(single tag) if captured at event 1 as function of cov1,cov2.
pST.1	Function to generate p(apply single tag to first position at event 1) as function of cov1,cov2.
rho1	Function to generate P(tag1 retained) as function of cov1,cov2.
rho2	Function to generate P(tag2 retained) as function of cov1,cov2.
p2	Function to generate P(capture) at event 2 for each member of population as function of cov1,cov2.
seed	Initial value of random seed
trace	Trace flag to help debug if things fail.

Details

The cov1 function takes the value N and returns N covariate values. For example these could be simulated length, or sex of each fish. The cov2 function takes the cov1 values and generates a second covariate. Two covariates should be sufficient for most capture-recapture simulations. If generating continuous covariates, you should round the covariate to about 100 distinct values to speed up your simulation.

The remaining functions take the two covariate values and generate capture probabilities, single tag probabilities, placing tags on fish, and tag retention probabilities. These should all be in the range of 0 to 1.

After generating capture histories for the entire population, animals never seen are "discarded" and the data set is compress to unique combinations of the two covariates and the capture history with the frequency variable set accordingly.

Value

Data frame with observed capture histories

Examples

```
sim_data <-LP_TL_simulate(
      dt_type="t2perm",  # permanent tag
      N=1000,
```

```
      cov1=function(N)         {rep(1,N)},
      cov2=function(cov1)      {rep(1,  length(cov1))},
      p1  =function(cov1, cov2){rep(.1, length(cov1))},
      pST =function(cov1, cov2){rep(.25,length(cov1))},
      rho1=function(cov1, cov2){rep(.70,length(cov1))},
      rho2=function(cov1, cov2){rep(1,  length(cov1))},  # permanent second tag
      p2  =function(cov1, cov2){rep(.1, length(cov1))},
      seed=round(1000000*runif(1)))
sim_data
```

`split_cap_hist` *Split a vector of capture histories into a matrix with one column for each occasion*

Description

Split a vector of capture histories into a matrix with one column for each occasion

Usage

```
split_cap_hist(cap_hist, sep = "", n = 2, prefix = "t", make.numeric = FALSE)
```

Arguments

`cap_hist`	A vector of capture histories.
`sep`	What separates the individual history values
`n`	Number of sampling events in each history
`prefix`	Prefix for labeling columns of matrix
`make.numeric`	Change the expanded columns to numeric from character?

Details

@template data.cap_hist

Value

A matrix of capture histories with 1 column per sampling event

Examples

```
# standard 2 character capture histor
data(data_rodli)
Petersen::split_cap_hist(data_rodli$cap_hist)

# history vector with ".." separating the fields
test <- c("1..1","1..0")
split_cap_hist(test, sep=stringr::fixed(".."))
```

Index