Package ‘pistar’

September 27, 2013

Type Package

Title Rudas, Clogg and Lindsay Mixture Index of Fit

Version 0.5.1

Date 2013-09-27

Author Juraj Medzihorsky <juraj.medzihorsky@gmail.com>

Maintainer Juraj Medzihorsky <juraj.medzihorsky@gmail.com>

Depends methods

Description Functions for estimating the Rudas, Clogg and Lindsay mixture index of fit.

License GPL (>= 2)

R topics documented:

`pistar-package` *Rudas, Clogg, and Lindsay Mixture Index of Fit*

Description

Functions for the Rudas, Clogg and Lindsay π^* mixture index of fit.

Details

Package:	pistar
Type:	Package
Version:	0.5.1
Date:	2013-09-27
License:	GPL (>= 2)

Functions for the Rudas, Clogg and Lindsay π^* mixture index of fit.

Note

The author would like to thank Tamas Rudas for invaluable advice in the development of this package.

Work on the package was supported in part by Central European University grant BPF/5182/20123/R.

Author(s)

Juraj Medzihorsky <`juraj.medzihorsky@gmail.com`>

`Fienberg1980a` *Citation Practices in Two Operations Research Journals*

Description

Citation practices in two operations reserch journals from Fienberg (1980a) analyzed by Rudas (2002) using the mixture index of fit.

Reprinted by permission of The Applied Probability Trust. First published in Fienberg, S. E. (1980a) Using loglinear models to analyze cross-classified categorical data, *The Mathematical Scientist*, 5, 13-30. Copyright (c) Applied Probability Trust 1980.

Usage

```
data(Fienberg1980a)
```

Format

A 2-dimensional 2-by-4 array cross-tabulating observations of 336 papers in two operations research journals, *Management Science* (MS) and *Operations Research* (OR), from their 1969 and 1970 issues. Each papers is classified into one of four categories based on whether it refers to other papers from these two journals.

The variables and their levels are as follows:

No	Name	Levels
1	citing	MS, OR
2	cited	none, MS, OR, both

Source

Fienberg, S. E. (1980a) Using loglinear models to analyze cross-classified categorical data, *The Mathematical Scientist*, 5, 13-30.

References

Fienberg, S. E. (1980a) Using loglinear models to analyze cross-classified categorical data, *The Mathematical Scientist*, 5, 13-30.

Rudas, T. (2002) 'A Latent Class Approach to Measuring the Fit of a Statistical Model' in Hagenaars, J. A. and McCutcheon, A. L. (eds.) Applied Latent Class Analysis. Cambridge University Press. 345-365.

Examples

```
data(Fienberg1980a)
Fienberg1980a
```

`freq.table` *Frequency Table of Integers in a Vector*

Description

`freq.table` is used to obtain a frequency table of values from a numeric vector. Unlike `table` it includes the frequency of each integer value on the interval from the smallest to the largest observed value, and thus can contain also zeros.

Usage

```
freq.table(x)
```

Arguments

`x`	a numeric vector.

Details

`freq.table` first coerces the input vector into type `integer` and then creates a table of frequencies of all integers on the closed interval from the smallest to the largest inputed integer.

Value

A named vector of frequencies.

Author(s)

Juraj Medzihorsky

See Also

`table`

Examples

```
set.seed(1989)
y <- c(rpois(1e1, 3), rpois(1e1, 1e1))
freq.table(y)
# compare with table()
table(y)
```

`pearson2pistar` *Pearson to $\pi\hat{}*$ Conversion*

Description

`pearson2pistar` is used to convert Pearson product-moment correlation coefficient into the π^* mixture index of fit using the relationship between them described by Rudas, Clogg, and Lindsay (1994).

Usage

```
pearson2pistar(coeff)
```

Arguments

`coeff`	value of Pearson product-moment correlation coefficient to be converted.

Details

The relationship between π^* and Pearson ρ is:

$$\pi^* = 1 - \sqrt{\frac{1 - |\rho|}{1 + |\rho|}}$$

Value

Value of π^* corresponding to the supplied value of Pearson product-moment correlation coefficient.

Author(s)

Juraj Medzihorsky

References

Rudas, T., Clogg, C. C., Lindsay, B. G. (1994) A New Index of Fit Based on Mixture Methods for the Analysis of Contingency Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 56, No. 4, 623-639.

See Also

`cor` `pistar.bvn`

Examples

```
pearson2pistar(0.5)

curve(pearson2pistar, from=-1,
      xlab=expression(rho),
      ylab=expression(pi^symbol('*')))
```

`piplot.ct` *Pre-analysis Plot for* `pistar.ct`

Description

`piplot.ct` is used to inform the choice of the interval on which to look for π^* with `pistar.ct`. It plots log-likelihood ratio statistic values of a two-point mixture of a user-supplied model and an unrestricted component fit to a contingency table on a specified number of equally-spaced points on a supplied interval.

Usage

```
piplot.ct(fn, data, ..., from = .Machine$double.neg.eps^0.25,
           to = 1 - .Machine$double.neg.eps^0.25,  n = 10, draw = TRUE,
           color = "black", zero_line = TRUE, add = FALSE, values = FALSE)
```

Arguments

`fn`	a user-supplied function that inputs a contingency table of observed values and outputs a contingency table of predicted values. The function must output a named list with the contingency table of predicted values named `'fit'`.
`data`	a contingency table.
`...`	further arguments passed to the user-supplied function.
`from`	numeric: lower bound of the interval of out-of-model proportions to be explored.
`to`	numeric: upper bound of the interval of out-of-model proportions to be explored.
`n`	numeric: number of equally-spaced points from the interval of out-of-model proportions to be explored.
`draw`	logical: draw a plot or lines?

`color` color of the line.

`zero_line` logical: plot a horizontal line at 0?

`add` logical: add the line to a plot?

`values` logical: return a dataframe with explored out-of-model proportions and their corresponding log-likelihood ratio statistics?

Details

Developed from John M. Grego's `clr.plot`

Value

A plot. If `values = TRUE` returns also an object of class 'PiplotCT' with the following slots:

`pi` numeric vector of explored out-of-model proportions

`lr` numeric vector of corresponding log-likelihood ratio statistics

`lr_plus_eps` `lr` plus the value of `eps`

Author(s)

Juraj Medzihorsky, developed from John M. Grego's `clr.plot`

References

Grego, J.M. `clr.plot` function available at `http://www.stat.sc.edu/~grego/courses/stat770/CLR.txt`

See Also

`pistar.ct` `rcl.em`

Examples

```
data(Fienberg1980a)

mf <- function(data) loglin(data, list(1, 2), fit=TRUE, print=FALSE)

a <- piplot.ct(fn=mf, data=Fienberg1980a, values=TRUE)

a

plot(a, color='red')
```

PiplotCT-class *Class* "PiplotCT"

Objects from the Class

Objects can be created by calls of the form `new("PiplotCT", ...)`.

Slots

`call`: Object of class `"language"` ~~

`pi`: Object of class `"numeric"` ~~

`lr`: Object of class `"numeric"` ~~

`lr_plus_eps`: Object of class `"numeric"` ~~

Methods

plot `signature(x = "PiplotCT")`: ...

print `signature(x = "PiplotCT")`: ...

show `signature(object = "PiplotCT")`: ...

Examples

```
showClass("PiplotCT")
```

pistar *The Mixture Index of Fit*

Description

`pistar` is a wrapper function for all functions estimating the Rudas-Clogg-Lindsay π^* mixture index of fit from package **pistar**.

Usage

```
pistar(proc, ...)
```

Arguments

`proc`	`"uv"` for `pistar.uv`
	`"ct"` for `pistar.ct`
	`"ll"` for `pistar.ll`
	`"2by2"` for `pistar.2by2`
	`"mvn"` for `pistar.mvn`
	`"bvn"` for `pistar.bvn`

`...`	arguments passed to the function.

Value

An object of class "Pistar".

Author(s)

Juraj Medzihorsky

Examples

```
#  create data in a 2-by-2 table
H <- matrix(rpois(4, 20), ncol=2)

# find pi* for independence (i.e. odds ratio of 1)
h <- pistar(proc='2by2', data=H, alpha=1, jack=FALSE)

h
```

Pistar-class   *Class* "Pistar"

Objects from the Class

Objects can be created by calls of the form `new("Pistar", ...)`.

Slots

`call`**:** Object of class "language" ~~

`pistar`**:** Object of class "list" ~~

`pred`**:** Object of class "list" ~~

`param`**:** Object of class "list" ~~

Methods

print `signature(x = "Pistar")`: ...

show `signature(object = "Pistar")`: ...

summary `signature(object = "Pistar")`: ...

Examples

```
showClass("Pistar")
```

pistar.2by2 *The Mixture Index of Fit for Odds Ratios in 2-by-2 Tables*

Description

`pistar.2by2` is used to estimate π^* for a given cross-product ratio (i.e. odds ratio) in a 2-by-2 table using the method devised by Clogg, Rudas, and Xi (1995).

Usage

```
pistar.2by2(data, alpha = 1, jack = FALSE, verbose = TRUE)
```

Arguments

`data`	a 2-by-2 contingency table.
`alpha`	numeric: cross-product ratio
`jack`	logical: perform jackknife?
`verbose`	logical: print during estimation?

Value

Object of `class "Pistar"`, `"PistarCT"`, and `"Pistar2by2"` with the following slots:

`call` the matched call.

`pistar` a list of estimated values of the mixture index of fit.

- **est** for the supplied data and odds ratio.
- **jack** vector of values from jackknife.

`pred` a list of predicted values with three items:

- **model** the model component multiplied by $(1 - \pi^*)$
- **unres** the unrestricted component multiplied by π^*
- **combi** the two-point mixture, i.e. $(1 - \pi^*)M + \pi^*U$

`data` the supplied data.

`param` a list of with a single item

- **est** numeric: the supplied value of cross-product ratio.

References

Clogg, C. C., Rudas, T., & Xi, L. (1995). A new index of structure for the analysis of models for mobility tables and other cross-classifications. *Sociological Methodology*, 197-222.

Rudas. T., Zwick. R., (1997) Estimating the Importance of Differential Item Functioning. *Journal of Educational and Behavioral Statistics*, Vol. 22, No. 1 (Spring, 1997), pp. 31-45

Examples

```
H <- matrix((1:4)*1e1, byrow=TRUE, ncol=2)

h <- pistar.2by2(H, alpha=1, jack=TRUE)

h

str(h)

plot(h)

summary(h)
```

`pistar.bvn` *The Mixture Index of Fit for Bivariate Normal Independence*

Description

`pistar.bvn` is used to estimate the π^* mixture index of fit for independence in a bivariate normal distribution using the relationship between π^* and Pearson correlation coefficient.

Usage

```
pistar.bvn(x, y, conf = 0.95, na.action, alt = "two.sided", verbose = TRUE)
```

Arguments

`x`	a numeric vector containing the first variable.
`y`	a numeric vector containing the second variable.
`conf`	confidence level.
`na.action`	passed to `cor.test`.
`alt`	alternative hypothesis, determines the shape of the confidence interval; passed to `cor.test`.
`verbose`	logical: print during estimation?

Value

Object of `class "Pistar"`, and `"PistarBVN"` with the following slots:

`call`	the matched call.
`pistar`	a list of estimated values of the mixture index of fit: **est** for the supplied data. **jack** vector of values from jackknife.
`pred`	not yet implemented.
`data`	a `data.frame` with the supplied data.
`param`	not yet implemented.
`interval`	confidence interval for pi*.
`conf`	confidence level.
`alt`	the supplied alternative hypothesis.

Author(s)

Juraj Medzihorsky

References

Rudas, T., Clogg, C. C., Lindsay, B. G. (1994) A New Index of Fit Based on Mixture Methods for the Analysis of Contingency Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 56, No. 4, 623-639.

See Also

`cor.test` `pearson2pistar`

Examples

```
# simlate data
set.seed(1989)
n <- 1e2
a <- rnorm(n)
b <- rnorm(n)

# find pi*
o <- pistar.bvn(x=a, y=b)

o
```

`pistar.ct` *The Mixture Index of Fit for a Contingency Table*

Description

`pistar.ct` is used to find the value of the π^* for any user-supplied model fit to a contingency table. The only requirements are (1) that the model inputs only a contingency table with non-negative continuous cell values, and (2) outputs a named list in which the predicted values are a contingency table named `"fit"`. Optionally parameter estimates of interest can be outputed as a vector named `"param"` in the output list.

π^* for the model of interest is estimated using the algorithm of Rudas, Clogg, and Lindsay (1994). Standard errors for π^* and any other parameter estimates of interest can be obtained by jackknife as proposed by Dayton (2003).

Usage

```
pistar.ct(data, fn, from = .Machine$double.neg.eps^0.25,
          to = 1 - .Machine$double.neg.eps^0.25, jack = FALSE,
          method = "uniroot", u_iter = 1e3, zeta = 1,
          lr_eps = .Machine$double.neg.eps^0.25,
          max_dif = .Machine$double.neg.eps^0.5, chi_stat = 0, verbose = TRUE)
```

Arguments

`data` a contingency table.

`fn` a user supplied function that estimates the model of interests. Must input only the observed values as a contingency table containging non-negative continuous cell values. Must output the predicted values as a contingency table as item named `"fit"` in a named list. Optionally can output parameter estimates of interest as a vector named `"param"` in the output named list.

`from` numeric: lower bound of the interval of out-of-model proportions to be explored.

`to` numeric: upper bound of the interval of out-of-model proportions to be explored.

`jack` logical: perform jackknife?

`method` character: method with to look for π^*. `"uniroot"` uses `uniroot`, can be expected to be faster `"split"` uses a simple binary search from `rcl.s`.

`u_iter` maximum number of iterations for method `uniroot` if method `"uniroot"`.

`zeta` weighing constant; default is 1. The EM algorithm might crash due to very low cell values and in such case increasing the zeta might help.

`lr_eps` penalty for finding π^*, the largest small positive number that can be still considered practically indistinguishable from 0.

`max_dif` largest acceptable difference, passed to `rcl.em` and `rcl.s`.

`chi_stat` χ^2 statistic penalty; default 0. Supply a different e.g. if you want to find the lower endpoint of a one-sided confidence interval for π^*.

`verbose` logical: print during estimation?

Details

The EM algorithm implemented here was proposed by Rudas, Clogg and Lindsay (1994). The jackknife procedure was proposed by Dayton (2003). The function is developed from J.M.Grego's `clr` and `clr.root` functions.

Value

Object of `class` `"Pistar"`, `"PistarCT"`, and `"PistarRCL"` with the following slots:

`call` the matched call.

`pistar` a list of estimated values of the mixture index of fit.

est for the supplied data.

jack vector of values from jackknife.

`pred` a list of predicted values with three items:

model the model component multiplied by $(1 - \pi^*)$

unres the unrestricted component multiplied by π^*

combi the two-point mixture, i.e. $(1 - \pi^*)M + \pi^*U$

`data` an `array` with the supplied data.

`param` a list of requested estimates of the parameters of interest of the model fit to an **unscaled** model density, i.e. to M and **not** $(1 - \pi)M$.

est the estimated values.

jack from each jackknife replication.

`llrs` a list of values of log-likelihood ratio statistics

est for the supplied data.

jack vector of values from each jackknife.

iter — a list of the numbers of iterations of either uniroot or rcl.s

est for the supplied data.

jack from each jackknife replication.

Author(s)

Juraj Medzihorsky

Developed from J.M.Grego's clr and clr.root functions.

References

Dayton, C. M. (2003) Applications and computational strategies for the two-point mixture index of fit. *British Journal of Mathematical & Statistical Psychology*, 56, 1-13.

Grego, J. M. clr and clr.root functions available at http://www.stat.sc.edu/~grego/courses/stat770/CLR.txt

Rudas, T., Clogg, C. C., Lindsay, B. G. (1994) A New Index of Fit Based on Mixture Methods for the Analysis of Contingency Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 56, No. 4, 623-639.

Rudas, T. (2002) 'A Latent Class Approach to Measuring the Fit of a Statistical Model' in Hagenaars, J. A. and McCutcheon, A. L. (eds.) Applied Latent Class Analysis. Cambridge University Press. 345-365.

See Also

piplot.ct rcl.em rcl.s

Examples

```
# load data
data(Fienberg1980a)

# define a function: log-linear model of independence in a
# 2-way table
mf <- function(x){
loglin(table=x, margin=list(1,2), fit=TRUE, print=FALSE)
}

# find pi*
p <- pistar(proc='ct', data=Fienberg1980a, fn=mf, jack=FALSE)

p

summary(p)

plot(p)
```

pistar.ll *The Mixture Index of Fit for Log-linear Models*

Description

pistar.ll is used to find the value of the π^* index of fit for any log-linear model estimated with loglin; π^* is estimated using the algorithm of Rudas, Clogg, and Lindsay (1994). Standard errors for π^* and any other estimates of parameters of interest can be obtained by jackknife as proposed by Dayton (2003).

Usage

```
pistar.ll(data, margin = list(1, 2), start = rep(1, length(data)), eps = 0.1,
          iter = 1e3, param = TRUE, print = FALSE,
          from = .Machine$double.neg.eps^0.25,
          to = 1 - .Machine$double.neg.eps^0.25, jack = FALSE,
          lr_eps = .Machine$double.neg.eps^0.25,
          max_dif = .Machine$double.neg.eps^0.5, chi_stat = 0, u_iter = 1e3,
          tol = .Machine$double.eps^0.25, verbose = TRUE)
```

Arguments

data	a contingency table.
margin	passed to loglin.
start	start argument of loglin.
eps	eps argument of loglin.
iter	maximum number of iterations for loglin.
param	logical: return parameter estimates?
print	logical: should loglin print during fitting?
from	numeric: lower bound of the interval of out-of-model proportions to be explored.
to	numeric: upper bound of the interval of out-of-model proportions to be explored.
jack	logical: perform jackknife?
lr_eps	penalty for finding π^*, the largest small positive number that can be still considered practically indistinguishable from 0.
max_dif	largest acceptable difference, passed to rcl.em.
chi_stat	χ^2 statistic penalty; default 0. Supply a different e.g. if you want to find the lower endpoint of a one-sided confidence interval for π^*.
u_iter	maximum number of iterations for method uniroot.
tol	tolerance passed to loglin.
verbose	logical: print during estimation?

Details

This is a version of the algorithm implemented in pistar.ct for log-linear models that is speed-optimized.

Value

Object of class "Pistar", "PistarCT", , "PistarRCL", and "PistarLL" with the following slots:

call	the matched call.
pistar	a list of estimated values of the mixture index of fit. **est** for the supplied data. **jack** vector of values from jackknife.
pred	a list of predicted values with three items: **model** the model component multiplied by $(1 - \pi^*)$ **unres** the unrestricted component multiplied by π^* **combi** the two-point mixture, i.e. $(1 - \pi^*)M + \pi^* U$
data	the supplied data.
param	a list of requested estimates of the parameters of interest of the model fit to an **unscaled** model density, i.e. to M and **not** $(1 - \pi)M$. **est** the estimated values. **jack** from each jackknife replication.
llrs	a list of values of log-likelihood ratio statistics **est** for the supplied data. **jack** vector of values from each jackknife.
iter	a list of the numbers of iterations of either uniroot or rcl.s **est** for the supplied data. **jack** from each jackknife replication.

Author(s)

Juraj Medzihorsky

Developed from J.M.Grego's clr and clr.root functions.

References

Dayton, C. M. (2003) Applications and computational strategies for the two-point mixture index of fit. *British Journal of Mathematical & Statistical Psychology*, 56, 1-13.

Grego, J. M. clr and clr.root functions available at http://www.stat.sc.edu/~grego/courses/stat770/CLR.txt

Rudas, T., Clogg, C. C., Lindsay, B. G. (1994) A New Index of Fit Based on Mixture Methods for the Analysis of Contingency Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 56, No. 4, 623-639.

Rudas, T. (2002) 'A Latent Class Approach to Measuring the Fit of a Statistical Model' in Hagenaars, J. A. and McCutcheon, A. L. (eds.) Applied Latent Class Analysis. Cambridge University Press. 345-365.

See Also

pistar.ct loglin

Examples

```
data(HairEyeColor)

# check if the data is an "array"
is(HairEyeColor, "array")

# it is not, so it first needs to be converted:
HEC <- array(HairEyeColor,
 dim=dim(HairEyeColor),
 dimnames=dimnames(HairEyeColor))

# find pi* for independence in a 3-way table
p <- pistar(proc='ll', data=HEC, margin=list(1, 2, 3), jack=FALSE)

p

summary(p)

# plot does not work for n-way tables if n > 2
# plot(p)


# create data
H <- matrix((1:4)*1e1, byrow=TRUE, ncol=2)

# find pi* and model parameter estimates and perform jackknife
h <- pistar(proc='ll', data=H, margin=list(1, 2), param=TRUE, jack=TRUE)

h

summary(h)
```

---

pistar.mvn *The Mixture Index of Fit for Multivariate Normal Independence*

Description

pistar.mvn is used to estimate the value of the π^* mixture index of fit for independence in multivariate normal distribution using the procedure of Knott (2005). Standard errors can be obtained using jackknife as proposed by Dayton (2003).

Usage

```
pistar.mvn(data, cor_matrix = FALSE, max_dif = .Machine$double.neg.eps^0.5,
            jack = FALSE, seed = 1989, lag = c(5, 10), verbose = TRUE)
```

Arguments

data	a matrix, or a correlation matrix.
cor_matrix	logical: is the supplied data a correlation matrix? If TRUE jackknife cannot be performed.

`max_dif` numeric: maximal acceptable difference between selected iterations for the convergence diagnostic. See ‘Details’.

`jack` logical: perform jackknife?

`seed` seed for random number generation.

`lag` parameters of the convergence diagnostic, see ‘Details’.

`verbose` logical: print during estimation?

Details

The function was developed from code published by Knott (2005).

A simple convergence diagnostic was added to Knott’s (2005) procedure. The absolute values of the differences between the value of π^* at the current iteration and between `lag` iterations and stops the iterations as successful if all the differences are smaller than the constant supplied as `max_dif` argument. To check if the algorithm has converged to a global or a local minimum check `test` in the output and restart the procedure with a different `seed` if needed (see Knott 2005 for more details).

Value

Object of `class "Pistar"`, and `"PistarMVN"` with the following components:

`call` the matched call.

`pistar` a list of estimated values of the mixture index of fit:

- **est** for the supplied data.
- **jack** vector of values from jackknife.

`pred` not yet implemented.

`data` the supplied data.

`param` not yet implemented.

`trace` a list of traces from the iterations:

- **est** vector from estimation with supplied data.
- **jack** list of traces from jackknife.

`iter` a list of numbers of the iterations:

- **est** from estimation with supplied data.
- **jack** a list of traces from jackknife.

`test` a test statistic to evaluate if the procedure has converged to a local or global optimum. See Knott (2005) for more details.

- **est** Vector from estimation with supplied data.
- **jack** List of vectors from jackknife.

Author(s)

Juraj Medzihorsky

Developed from code published by Knott (2005).

References

Dayton, C. M. (2003) Applications and computational strategies for the two-point mixture index of fit. *British Journal of Mathematical & Statistical Psychology*, 56, 1-13.

Knott, M. (2005) A measure of independence for a multivariate normal distribution and some connections with factor analysis, *Journal of Multivariate Analysis*, 96, 374-383.

Examples

```
# simulate data
set.seed(1989)
n <- 1e2
A <- cbind(rnorm(n), rnorm(n))

# find pi*
a <- pistar(proc=’mvn’, data=A, jack=FALSE)

a

summary(a)

plot(a)
```

`pistar.uv` *The Mixture Index of Fit for Univariate Distributions*

Description

`pistar.uv` is used to estimate the π^* index of fit for any user-supplied univariate distribution. The user must supply a probability mass or density function that inputs the data as the first argument and the parameters as the next arguments. See ‘Details’ for the estimation procedures. Standard errors available via jackknife as suggested by Dayton (2003).

Usage

```
pistar.uv(data, dfn, n_par = NULL, inits = NULL, discrete = FALSE,
          freq = FALSE, lower = NULL, upper = NULL, jack = FALSE,
          method = "Nelder-Mead", control = list(maxit = 2000),
          verbose = TRUE, npk = 1e3, eps = .Machine$double.neg.eps^0.5)
```

Arguments

`data` a vector or a frequency table.

`dfn` function: probability mass or density function that inputs the data as the first argument and the parameters as the arguments that immediately follow it.

`n_par` numeric: number of parameters. Either `n_par` or `inits` must be supplied. If only `n_par` is supplied initial values are generated internally and might not always be suitable.

`inits` a vector or list of initial values of parameters supplied to `optim`. If named the parameter names are preserved in the output.

`discrete` logical: is the distribution discrete?

`freq` logical: is the supplied data a frequency table? Relevant only if `discrete` is `TRUE`.

`lower` numeric: a vector of lower bounds for parameters.

`upper` numeric: a vector of upper bounds for parameters.

`jack` logical: perform jackknife?

`method` method argument for optim. Default "Brent" for mono-parameter functions and "Nelder-Mead" for multi-parameter functions. See `optim` for details on methods.

`control` list supplied to `optim`, see `optim`.

`verbose` logical: print during estimation?

`npk` an integer indicating the number of points for `density`; used if `discrete = FALSE`.

`eps` numeric: the smallest number practically indistinguishable from 0. Used only if `discrete = TRUE`.

Details

The general procedure for discrete and continuous distributions is the same: a general purpose optimization method is used to find such values of the parameters of the supplied distribution that minimize the following quantity: 1 minus the inverse of the ratio of the model and the observed density at the point of their supports where this ratio is highest. This quantity is π^*.

The procedure for discrete distributions differs from the one for the continuous distributions in the method used to obtain the observed density. In the discrete case the observed frequencies are used for the observed density. In the continuous case a kernel density is estimated using `density` with gaussian kernel.

Value

Object of `class "Pistar"`, and `"PistarUV"` and depending on the `discrete` argument of the function either `"PistarDUV"` or `"PistarCUV"` with the following slots:

`call` the matched call.

`pistar` a list of estimated values of the mixture index of fit.

est for the supplied data.

jack vector of values from jackknife.

`pred` if `discrete = TRUE` a list of predicted values with three items:

model the model component multiplied by $(1 - \pi^*)$

unres the unrestricted component multiplied by π^*

combi the two-point mixture, i.e. $(1 - \pi^*)M + \pi^* U$

if `discrete = FALSE` the list also contains three components with the same names, but they contain the values of the scaled densities at `npk` (i.e. by default `1e3`) points.

`data` the supplied data.

`param` a list of parameter estimates of interest:

est the estimated values.

jack from each jackknife replication.

`meth` `method` of `optim` used.

`conv` a list of integer codes from `optim` that indicate convergence of the optimization algorithm. Any value that is not 0 suggests problems. See `optim` for details.

est from estimation with the supplied data.

jack from jackknife replications.

`mess` a list of messages pased from `optim`. See `optim` for details.

est From the main estimation.

jack From jackknife replications.

Note

The application of the mixture index of fit for discrete distributions was proposed by Dayton (2003).

Author(s)

Juraj Medzihorsky

References

Dayton, C. M. (2003) Applications and computational strategies for the two-point mixture index of fit. *British Journal of Mathematical & Statistical Psychology*, 56, 1-13.

See Also

optim density freq.table

Examples

```
# (1) discrete
# simulate data
set.seed(1989)
e <- c(rpois(1e3, 2), rpois(2e2, 5))

# make a frequency table
te <- freq.table(e)

# define a funcion for a slice from Poisson
md <- function(x, l, lo=0, up=5){
z <- dpois(x, l)
z[x<lo] <- 0
z[x>up] <- 0
z <- z/sum(z)
return(z)
}

# find pi*
pe <- pistar(proc='uv', data=te, dfn=md, n_par=1,
 discrete=TRUE, freq=TRUE, jack=FALSE)

pe

summary(pe)

plot(pe)


# (2) continuous
# simulate data
set.seed(1989)
y <- c(rnorm(1e2, 0, 2), runif(2e1, -1, 1))

# find pi* and parameters for normal dist.
py <- pistar(proc='uv', data=y, dfn=dnorm, n_par=2, discrete=FALSE,
 jack=FALSE)

py
```

```
summary(py)

plot(py)
```

Pistar2by2-class *Class* "Pistar2by2"

Objects from the Class

Objects can be created by calls of the form `new("Pistar2by2", ...)`.

Slots

`data`: Object of class `"array"` ~~

`call`: Object of class `"language"` ~~

`pistar`: Object of class `"list"` ~~

`pred`: Object of class `"list"` ~~

`param`: Object of class `"list"` ~~

Extends

Class `"PistarCT"`, directly. Class `"Pistar"`, by class "PistarCT", distance 2.

Methods

No methods defined with class "Pistar2by2" in the signature.

Examples

```
showClass("Pistar2by2")
```

PistarBVN-class *Class* "PistarBVN"

Objects from the Class

Objects can be created by calls of the form `new("PistarBVN", ...)`.

Slots

`data`: Object of class `"data.frame"` ~~

`interval`: Object of class `"numeric"` ~~

`conf`: Object of class `"numeric"` ~~

`alt`: Object of class `"character"` ~~

`call`: Object of class `"language"` ~~

`pistar`: Object of class `"list"` ~~

`pred`: Object of class `"list"` ~~

`param`: Object of class `"list"` ~~

Extends

Class "Pistar", directly.

Methods

plot `signature(x = "PistarBVN")`: ...

Examples

```
showClass("PistarBVN")
```

PistarCT-class *Class* "PistarCT"

Usage

```
## S4 method for signature 'PistarCT'
plot(x, y, ...)
```

Arguments

x	object of class PistarCT
y	ignored
...	optional

Objects from the Class

Objects can be created by calls of the form `new("PistarCT", ...)` or `PistarCT(...)`.

Slots

`data`: Object of class "array" ~~

`call`: Object of class "language" ~~

`pistar`: Object of class "list" ~~

`pred`: Object of class "list" ~~

`param`: Object of class "list" ~~

Extends

Class "Pistar", directly.

Methods

plot `signature(x = "PistarCT")`: ...

Author(s)

Juraj Medzihorsky

Examples

```
showClass("PistarCT")
```

PistarCUV-class *Class* "PistarCUV"

Usage

```
## S4 method for signature 'PistarCUV'
plot(x, model_col = 'blue', unres_col = 'grey', combi_col = 'black', lty = 1, lwd = 1,
pos = 'topright', bty = 'n', ... )
```

Arguments

x	object of class "PistarCUV".
model_col	color for the model component.
unres_col	color for the unrestricted component.
combi_col	color for the predicted values from the two-point mixture.
lty	line type.
lwd	line width.
pos	position of the legend, see legend.
bty	box type of the legend, see legend.
...	optional arguments passed further to plot.

Objects from the Class

Objects can be created by calls of the form new("PistarCUV", ...) or PistarCUV(...).

Slots

data: Object of class "numeric" ~~

meth: Object of class "character" ~~

conv: Object of class "list" ~~

mess: Object of class "list" ~~

call: Object of class "language" ~~

pistar: Object of class "list" ~~

pred: Object of class "list" ~~

param: Object of class "list" ~~

Extends

Class "PistarUV", directly. Class "Pistar", by class "PistarUV", distance 2.

Methods

plot signature(x = "PistarCUV"): ...

Examples

```
showClass("PistarCUV")
```

PistarDUV-class *Class* "PistarDUV"

Usage

```
## S4 method for signature 'PistarDUV'
plot(x, model_col = 'blue', unres_col = 'grey', combi_col = 'black', pos = 'topright', bty = 'n'
```

Arguments

x	object of class "PistarDUV".
model_col	color for the model component.
unres_col	color for the unrestricted component.
combi_col	color for the predicted values from the two-point mixture.
pos	position of the legend, see legend.
bty	box type of the legend, see legend.
...	optional arguments passed further to plot.

Objects from the Class

Objects can be created by calls of the form `new("PistarDUV", ...)` or `PistarDUV(...)`.

Slots

`data`: Object of class "table" ~~

`meth`: Object of class "character" ~~

`conv`: Object of class "list" ~~

`mess`: Object of class "list" ~~

`call`: Object of class "language" ~~

`pistar`: Object of class "list" ~~

`pred`: Object of class "list" ~~

`param`: Object of class "list" ~~

Extends

Class "PistarUV", directly. Class "Pistar", by class "PistarUV", distance 2.

Methods

plot `signature(x = "PistarDUV")`: ...

Examples

```
showClass("PistarDUV")
```

PistarLL-class *Class* "PistarLL"

Objects from the Class

Objects can be created by calls of the form `new("PistarLL", ...)`.

Slots

`llrs`: Object of class `"list"` ~~

`iter`: Object of class `"list"` ~~

`data`: Object of class `"array"` ~~

`call`: Object of class `"language"` ~~

`pistar`: Object of class `"list"` ~~

`pred`: Object of class `"list"` ~~

`param`: Object of class `"list"` ~~

Extends

Class `"PistarRCL"`, directly. Class `"PistarCT"`, by class "PistarRCL", distance 2. Class `"Pistar"`, by class "PistarRCL", distance 3.

Methods

No methods defined with class "PistarLL" in the signature.

Examples

```
showClass("PistarLL")
```

PistarMVN-class *Class* "PistarMVN"

Usage

```
## S4 method for signature 'PistarMVN'
plot(x, lty = 1, lwd = 1, col = 'black', ... )
```

Arguments

`x`	object of class `"PistarMVN"`.
`lty`	line type.
`lwd`	line width.
`col`	color of the line.
`...`	optional arguments passed further.

Objects from the Class

Objects can be created by calls of the form `new("PistarMVN", ...)` or `PistarMVN()`.

Slots

`data`: Object of class `"data.frame"` ~~

`trace`: Object of class `"list"` ~~

`iter`: Object of class `"list"` ~~

`test`: Object of class `"list"` ~~

`call`: Object of class `"language"` ~~

`pistar`: Object of class `"list"` ~~

`pred`: Object of class `"list"` ~~

`param`: Object of class `"list"` ~~

Extends

Class `"Pistar"`, directly.

Methods

plot `signature(x = "PistarMVN")`: ...

Examples

```
showClass("PistarMVN")
```

PistarRCL-class *Class* "PistarRCL"

Objects from the Class

Objects can be created by calls of the form `new("PistarRCL", ...)`.

Slots

`llrs`: Object of class `"list"` ~~

`iter`: Object of class `"list"` ~~

`data`: Object of class `"array"` ~~

`call`: Object of class `"language"` ~~

`pistar`: Object of class `"list"` ~~

`pred`: Object of class `"list"` ~~

`param`: Object of class `"list"` ~~

Extends

Class `"PistarCT"`, directly. Class `"Pistar"`, by class "PistarCT", distance 2.

Methods

No methods defined with class "PistarRCL" in the signature.

Examples

```
showClass("PistarRCL")
```

`PistarUV-class` *Class* `"PistarUV"`

Objects from the Class

Objects can be created by calls of the form `new("PistarUV", ...)`.

Slots

`meth`: Object of class `"character"` ~~

`conv`: Object of class `"list"` ~~

`mess`: Object of class `"list"` ~~

`call`: Object of class `"language"` ~~

`pistar`: Object of class `"list"` ~~

`pred`: Object of class `"list"` ~~

`param`: Object of class `"list"` ~~

Extends

Class `"Pistar"`, directly.

Methods

No methods defined with class "PistarUV" in the signature.

Examples

```
showClass("PistarUV")
```

pool.jack *Jackknife Standard Errors and Confidence Intervals*

Description

`pool.jack` is used to obtain standard errors and confidence intervals from the output of jackknife procedures.

Usage

```
pool.jack(data, ct = TRUE, estimate, jack_est, side = NULL, conf = 0.95,
          lower = -Inf, upper = Inf, bias = FALSE)
```

Arguments

`data`	the data.
`ct`	logical: is the data a contingency table?
`estimate`	the value of the estimate.
`jack_est`	the vector of estimates from jackknife replications.
`side`	`NULL` for a two-sided c.i., `"lower"` or `"upper"` for a one-sided interval. `"auto"` selects `"lower"` if the estimate is larger than 0 and `"upper"` if it is smaller than 0.
`conf`	confidence level expressed as a number between 0 and 1. e.g. for a 95% confidence interval supply 0.95
`lower`	lowest possible value for one-sided c.i.
`upper`	largest possible value for one-sided c.i.
`bias`	logical: apply bias correction? Bias correction currently not implemented.

Value

A named list with the following components:

`se`	standard error.
`low`	lower endpoint of c.i.
`upp`	upper endpoint of c.i.
`conf`	confidence level of the c.i.
`side`	`"lower"` and `"upper"` for one-sided c.i., `"both"` for a two-sided c.i.
`bias`	implementation of bias correction not finished.

Author(s)

Juraj Medzihorsky

References

Efron, Bradley, and Robert Tibshirani. (1993). An introduction to the bootstrap. Vol. 57. CRC press.

rcl.em *Rudas-Clogg-Lindsay EM Algorithm*

Description

rcl.em is used to fit a two-point mixture composed of a user-supplied model of interest and an unrestricted distribution fit to a contingency table with supplied mixing proportions using the Rudas-Clogg-Lindsay (1994) EM algorithm.

Usage

```
rcl.em(pi_out, FNEM, data, max_dif = .Machine$double.neg.eps^0.5,
       zeta = 1, lr_only = TRUE, chi_stat = 0,
       lr_eps = .Machine$double.neg.eps^0.25)
```

Arguments

pi_out	out-of-model proportion, i.e. the mixing weight of the unrestricted component
FNEM	user-supplied function that estimates the model of interests. Must input only the observed values as a contingency table. Must output the predicted values as a contingency table as item named "fit" in a named list. Optionally can output parameter estimates of interest as a vector named "param" in the outputed named list.
data	a contingency table.
max_dif	largest acceptable difference, i.e. the largest number practically indistinguishable from 0.
zeta	weighing constant; default is 1. The EM algorithm might crash due to very low cell values; in such case increasing the zeta might help.
lr_only	logical: return only the value of the log-likelihood ratio statistic?
chi_stat	χ^2 statistic penalty; default 0. Supply a different value e.g. if you want to find the lower endpoint of a one-sided confidence interval for π^*.
lr_eps	penalty for finding π^*, the largest small positive number that can be still considered practically indistinguishable from 0.

Value

A named list with the following components: (if lr_only is TRUE then the list contains only the "lr" component)

pi_out	the out-of-model proportion, i.e. the mixing weight of the unrestricted component
param	a vector of the estimated parameter values fit to an **unscaled** model density, i.e. to M and **not** $(1-\pi)M$.
lr	general contingency table log-likelihood ratio statistic for the two-point mixture.
model	scaled density of predicted values following the model of interest, i.e. $(1-\pi)M$.
unrestricted	Scaled density of predicted values following unrestricted component, i.e. πU
predicted	values predicted by the two-point mixture, i.e. $(1-\pi)M+\pi U$.

Author(s)

Juraj Medzihorsky

Developed from J.M.Grego's functions, see 'References'

References

Rudas, T., Clogg, C. C., Lindsay, B. G. (1994) A New Index of Fit Based on Mixture Methods for the Analysis of Contingency Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 56, No. 4, 623-639.

Grego, J. M. `clr` and `clr.root` functions available at http://www.stat.sc.edu/~grego/courses/stat770/CLR.txt

See Also

`pistar.ct`

`rcl.s` *Simple Line-Splitting Function*

Description

'rcl.s' is used to find the input values under which a user-supplied function outputs a specified value by searching on a user-specified interval

Usage

```
rcl.s(FNS, ..., y_goal = .Machine$double.neg.eps^0.5,
      x_lo = .Machine$double.neg.eps^0.25,
      x_up = 1 - .Machine$double.neg.eps^0.25,
      s_tol = .Machine$double.neg.eps^0.5,
      s_mit = 1e2, trace_plot = FALSE)
```

Arguments

`FNS`	user-supplied function. Must input only a single number and output only a single number.
`...`	arguments passed to the user-supplied function.
`y_goal`	the goal output value.
`x_lo`	the lower endpoint of the interval for the input value.
`x_up`	the upper endpoint of the interval for the input value.
`s_tol`	tolerance value for convergence diagnostic. If the absolute value of the difference between the output value at the current iteration and the goal output value is less than this tolerance value the algorithm terminates successfully.
`s_mit`	maximum number of iterations.
`trace_plot`	logical: return also a traceplot?

Value

A dataframe with the following columns:

`pi`	input values.
`lr`	output values.

If `trace_plot = TRUE` also a plot.

Author(s)

Juraj Medzihorsky

`summary.Pistar` *Summarizing* `"Pistar"` *Objects*

Description

`summary` method for `class "Pistar"`

Usage

```
## S4 method for signature 'Pistar'
summary(object, conf = 0.95, pi_side = NULL, par_side = NULL,
        lower = NULL, upper = NULL, bias = FALSE, ...)

## S4 method for signature 'SummaryPistar'
print(x, digits = 2, ...)
```

Arguments

`object`	an object of `class "pistar"`
`x`	an object of `class "summary.pistar"`
`digits`	integer indicating the number of decimal places to print
`conf`	Confidence level expressed as a number between 0 and 1. e.g. for a 95% confidence interval supply 0.95
`pi_side`	Sidedness of c.i. for π^*. `NULL` for a two-sided c.i., `"lower"` or `"upper"` for a one-sided interval. `"auto"` selects `"lower"` if the estimate is larger than 0 and `"upper"` if it is smaller than 0.
`par_side`	Sidedness of c.i. for reported estimates of parameters other than π^*. `NULL` for a two-sided c.i., `"lower"` or `"upper"` for a one-sided interval. `"auto"` selects `"lower"` if the estimate is larger than 0 and `"upper"` if it is smaller than 0.
`lower`	lowest possible value for one-sided c.i.
`upper`	largest possible value for one-sided c.i.
`bias`	logical: Apply bias correction? Bias correction currently not implemented.
`...`	arguments to be passed to methods; not used

Value

Object of class "SummaryPistar", a list with the following slots

`oldcall` the matched call inherited from the input object of class "Pistar"

`pred` inherited from the input object of class "Pistar"

`est` A `data.frame` with the parameter estimates, optionally also with standard errors, lower and upper endpoints of c.i., sidedness of the c.i., and bias (the last is not currently fully implemented)

Author(s)

Juraj Medzihorsky

Examples

```
# create data:
H <- matrix((1:4)*1e1, byrow=TRUE, ncol=2)

# pi* for independence in a 2-by-2 table
h <- pistar(proc='2by2', data=H, alpha=1, jack=TRUE)

# print 'pistar' object
h

# summarize 'pistar' object
s <- summary(h)

# print 'summary.pistar' object
s

# print the 'summary.pistar' object to 4 decimal places
print(s, digits=4)

# compare the structure of the objects
str(h)
str(s)
```

SummaryPistar-class *Class* "SummaryPistar"

Objects from the Class

Objects can be created by calls of the form `new("SummaryPistar", ...)` or `SummaryPistar`.

Slots

`oldcall`: Object of class "language" ~~

`pred`: Object of class "list" ~~

`est`: Object of class "data.frame" ~~

Methods

- **print** `signature(x = "SummaryPistar")`: ...
- **show** `signature(object = "SummaryPistar")`: ...

Examples

```
showClass("SummaryPistar")
```

Index