Package 'fitPS'

August 27, 2026

Type Package

Title Fit Probability Models to Forensic Survey Data

Version 1.1.4

Description Fits probability models to P- and S-type count data arising from forensic surveys of clothing for the background presence of glass, paint, and related trace material. Built-in models include zeta, zero-inflated zeta, and logarithmic distributions, with a public extension interface for additional models. Inference is available by maximum likelihood, parametric Bayesian methods, the ordinary nonparametric bootstrap, and Rubin's Bayesian Bootstrap. The clothing-survey setting is described by Coulson, Buckleton, Gummer, and Triggs (2001) <doi:10.1016/S1355-0306(01)71847-3>.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports cubature, doParallel, dplyr, foreach, Hmisc, iterators, knitr, ks, methods, pbapply, Rdpack, readxl, VGAM

RdMacros Rdpack

URL https://github.com/jmcurran/fitPS

BugReports https://github.com/jmcurran/fitPS/issues

Suggests rmarkdown, sp, testthat (>= 3.0.0), xtable

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author James Curran [aut, cre]

Maintainer James Curran <j.curran@auckland.ac.nz>

Repository CRAN

Date/Publication 2026-08-27 03:10:02 UTC

Contents

==.psData . . . 3
add . . . 4
as.data.frame.psData . . . 4
bootCI . . . 5
bootstrapFit . . . 7
bootstrapProbs . . . 9
compareSurveys . . . 10
compareSurveysLRT . . . 12
confint.psFit . . . 13
credint . . . 14
deviance.psFit . . . 16
DIC . . . 16
fit . . . 17
fitDist . . . 19
fitlogDist . . . 22
fitted.psFit . . . 23
fitted.psPosterior . . . 24
fitZIDist . . . 25
logarithmicModel . . . 28
logLik.psFit . . . 28
makePrior . . . 29
makePSData . . . 30
mean.psData . . . 31
modelBayesControl . . . 32
modelFromUnconstrained . . . 33
modelLogJacobian . . . 34
modelLogLikelihood . . . 35
modelLogPrior . . . 35
modelMleControl . . . 36
modelObservationData . . . 37
modelParameterNames . . . 37
modelProbabilities . . . 38
modelToUnconstrained . . . 39
plot.psBootstrap . . . 40
plot.psFit . . . 41
plot.psPosterior . . . 42
plotPosterior . . . 43
plotUncertainty . . . 45
posteriorInflation . . . 47
posteriorProbs . . . 48
predict.psFit . . . 49
print.psBayesianBootstrap . . . 50
print.psBootstrap . . . 51
print.psData . . . 52
print.psFit . . . 52
print.psPosterior . . . 53

probfun 53
psModel 54
Psurveys 55
readData 56
rzeta 57
rZIzeta 57
Ssurveys 58
summary.psBayesianBootstrap 59
summary.psBootstrap 60
summary.psFit 61
summary.psPosterior 61
supportedPosteriorEngines 62
var 63
var.psData 63

Index **65**

`==.psData` *S3 method for objects of class* `psData`

Description

Tests to see if two objects of class `psData` are equal. That is their `type` is the same, and the data contained in `data` is the same. See `readData` for a description of the `psData` class.

Usage

```
## S3 method for class 'psData'
lhs == rhs
```

Arguments

`lhs`	an object of class `psData`.
`rhs`	an object of class `psData`.

Details

NOTE: the `notes` member variable is ignored in this function as it is unlikely that a user would want to see if the notes are the same.

Value

TRUE if the two objects are equal

Examples

```
p = readData(system.file("extdata", "p.xlsx", package = "fitPS"))
p1 = makePSData(n = 0:2, count = c(98, 1, 1), type = "P")
p2 = makePSData(n = 0:2, count = c(97, 2, 1), type = "P")
p == p1 ## TRUE
p == p2 ## FALSE
p1 == p2 ## FALSE
```

---

add *Add data to a psData object*

Description

Add one or more new observations to an existing clothing survey object.

Usage

```
add(x, newData)
```

Arguments

x	an object of class psData—see readData for details.
newData	either a vector, matrix or data.frame containing the new data. If a vector or magtrix is supplied then it must be either of length or have two columns. If a data.frame is supplied then the columns must be labelled "n" and "rn". The new data MUST NOT contain values that already exist in x$n

Value

an object of class pSData

Examples

```
add(Ssurveys$lau, c(11, 1))
```

---

as.data.frame.psData *Converts an object of class* psData *to a* data.frame

---

Description

Converts an object of class psData—see readData—to a data.frame that can be used with in functions in other packages such as vglm to fit more complicated models.

Usage

```
## S3 method for class 'psData'
as.data.frame(x, ...)
```

Arguments

x	an object of class psData—see readDatafor more details.
...	any other arguments passed to data.frame.

Details

If x is a psData object of type "P", i.e. it relates to numbers of groups of glass, then a data.frame with a single variable count will be return where count = rep(x$data$n + 1, x$data$rn). The counts have one added to them because the zeta distribution requires that the counts are greater than or equal to one. If x is a psData object of type "P", i.e. it relates to group sizes, then a data.frame with a single variable count will be return where count = rep(x$data$n, x$data$rn).

Value

a data.frame with a single variable count. The number of rows in the data.frame is equal to sum(x$data$rn).

Examples

```
p = readData(system.file("extdata", "p.xlsx", package = "fitPS"))
p.df = as.data.frame(p)
table(p.df$count)
p$data
```

---

bootCI *Legacy bootstrap confidence intervals or regions*

Description

‘bootCI()‘ is a deprecated compatibility interface. New code should fit the model with [fit()] using ‘method = "bootstrap"‘ and use [plotUncertainty()] for parameter-uncertainty intervals or regions. The historical return structure is retained for existing code.

Usage

```
bootCI(x, ...)

## Default S3 method:
bootCI(
  x,
  level = 0.95,
  B = 2000,
  model = c("zeta", "ziz"),
  returnBootValues = FALSE,
  silent = FALSE,
  plot = FALSE,
  parallel = TRUE,
```

```
  progressBar = FALSE,
  pbopts = list(type = "txt"),
  ...
)

## S3 method for class 'psData'
bootCI(x, ...)

## S3 method for class 'psFit'
bootCI(x, ...)
```

Arguments

x	a object either of class psData—see readData for more details—or of class psFit.
...	other arguments.
level	the confidence level required—restricted to [0.75, 1). This may be a vector, in which case multiple intervals, or confidence regions will be returned.
B	the number of bootstrap samples to take.
model	which model to fit to the data, either "zeta" or "ziz". Maybe abbreviated to "z" and "zi". Default is "zeta".
returnBootValues	if TRUE then the vector (or data.frame) of bootstrapped values is returned. This can be useful for debugging or understanding the results. Default is FALSE.
silent	if TRUE, then no output will be displayed whilst the bootstrapping is being undertaken. plot if TRUE then the contours for the confidence region will be plotted. This only works if model = "ziz". It is ignored otherwise. parallel if TRUE then the bootstrapping is performed in parallel.
plot	if TRUE and model == "ziz", then a plot of the bootstrapped values will be produced and confidence contour lines will be drawn for each value in level.
parallel	if TRUE, then the package will attempt to use multiple cores to speed up computation.
progressBar	if TRUE, then progress bars will be displayed to show progress on the bootstrapping.
pbopts	a list of arguments for the pboptions function that affect the progress bars. Ignored if progressBar = FALSE.

Details

This compatibility function uses bootstrapping to compute a confidence interval for the shape parameter in the case of the zeta model and a confidence region in the case of the zero-inflated zeta model. A smoothed bootstrap approach is taken rather than a simple percentile method. The kernel density estimation is performed by the ks package using a smoothed cross-validated bandwidth selection procedure. Two-dimensional bootstrap regions use an unconstrained KDE. Earlier fitPS releases requested positive = TRUE from ks::kde(), but that boundary transformation can materially distort the orientation of the joint bootstrap replicate density even when the fitted parameter cloud lies well inside its parameter bounds.

Value

If `returnBootVals == TRUE` then the results are returned in a list with elements named `ci` and `bootVals` for the zeta model and `confRegion` and `bootVals` for the zero-inflated zeta model. The structure of `ci` and `confregion` is described below. If `model == "zeta"`, then either a `vector` or a `data.frame` with elements/columns named `"lower"` and `"upper"` representing the lower and upper bounds of the confidence interval(s). Multiple bounds are returned in a `data.frame` when `level` has more than one value. If `model == "ziz "`, then a list with length equal to the length of `level` is returned. The name of each element in the list is the level with list has a single element named `"95%"`. It is possible for there to be multiple contours for the confidence region for a given `level`. If there is only one contour for each value of `level`, then each element of the list consists of a `list` with elements named `pi` and `shape` which specify the coordinates of the contour(s) for that level. There is a third element named `level` which gives the height of the kernel density estimate at that contour. If there are multiple contours for a given value of `level` then each list element is a list of lists with the structure given above (`level`, `pi`, and `shape`). NOTE: it is quite possible that there are multiple contours for a given height. If you want a way of thinking about this consider a mountain range with two mountains of equal height. If you draw the contours for (almost) any elevation, then you would expect to capture a region from each mountain.

Methods (by class)

- `bootCI(default)`: Bootstrap confidence intervals or regions
- `bootCI(psData)`: Bootstrap confidence intervals or regions
- `bootCI(psFit)`: Bootstrap confidence intervals or regions

Examples

```
if (interactive()) {
  data(Psurveys)
  roux = Psurveys$roux
  bootFit = fit(
    roux,
    model = zizModel(),
    method = "bootstrap",
    B = 200,
    seed = 123,
    silent = TRUE,
    parallel = FALSE
  )
  plotUncertainty(bootFit, level = c(0.80, 0.95))
}
```

`bootstrapFit` *Compute a bootstrap distribution for a fitted fitPS model*

Description

‘bootstrapFit()‘ is a deprecated compatibility wrapper. New code should use [fit()] with ‘method = "bootstrap"‘. The wrapper computes and attaches a nonparametric bootstrap distribution to a maximum-likelihood ‘psFit‘ object.

Usage

```
bootstrapFit(object, ...)

## S3 method for class 'psFit'
bootstrapFit(
  object,
  B = 2000,
  level = 0.95,
  seed = NULL,
  silent = FALSE,
  parallel = TRUE,
  progressBar = FALSE,
  pbopts = list(type = "txt"),
  ...
)
```

Arguments

object	A maximum likelihood psFit object.
...	Additional arguments passed to methods.
B	Number of bootstrap replicates.
level	Confidence level used for percentile intervals.
seed	Optional random seed for reproducible resampling.
silent	Logical; suppress progress messages when TRUE.
parallel	Logical; use parallel fitting when TRUE.
progressBar	Logical; display a progress bar when TRUE.
pbopts	Options passed to pbapply::pboptions() when progress bars are requested.

Details

The bootstrap is an observation-level nonparametric bootstrap. Each successful parameter replicate is transformed into the corresponding P or S probabilities before probability summaries are calculated. Consequently, the reported bootstrap mean probabilities are averages of transformed bootstrap replicates, not probabilities evaluated at average bootstrap parameter values.

Value

The input psFit object with a psBootstrap object attached as bootstrap.

Examples

```
if (interactive()) {
data(Psurveys)
fit = fit(
  Psurveys$roux,
  model = zizModel(),
  method = "bootstrap",
```

```
  nterms = 4,
  B = 20,
  seed = 123,
  silent = TRUE,
  parallel = FALSE
)
bootstrapProbs(fit)
}
```

`bootstrapProbs` *Extract bootstrap probability summaries*

Description

Extract bootstrap summaries for fitted P or S probabilities from a frequentist fitPS model with an attached bootstrap distribution or directly from its `psBootstrap` object.

Usage

```
bootstrapProbs(object, ...)

## S3 method for class 'psFit'
bootstrapProbs(object, n = NULL, ...)

## S3 method for class 'psBootstrap'
bootstrapProbs(object, n = NULL, ...)
```

Arguments

`object`	A `psFit` object with an attached `psBootstrap` object, or a `psBootstrap` object.
`...`	Additional arguments passed to methods.
`n`	`NULL`, a single number of leading probabilities to return, or a vector of P or S indices to return.

Details

The `estimate` column is the bootstrap mean of the probability across successful bootstrap replicates. It is not a Bayesian posterior mean and is generally not identical to the plug-in probability evaluated at the MLE. The reported bounds are percentile bootstrap confidence intervals.

Value

A data frame containing bootstrap probability summaries.

compareSurveys *Compare two surveys on the basis of their shape parameters*

Description

Compare two surveys on the basis of their shape parameters

Usage

```
compareSurveys(x, ...)

## Default S3 method:
compareSurveys(
  x,
  y,
  xname = NULL,
  yname = NULL,
  alternative = c("two.sided", "less", "greater"),
  null.value = 0,
  print = TRUE,
  ...
)

## S3 method for class 'psData'
compareSurveys(x, y, ...)

## S3 method for class 'psFit'
compareSurveys(x, y, ...)

compare.surveys(x, ...)

comp.survs(x, ...)
```

Arguments

x	either an object of class psData—see readData or an object of class psFit—see fitDist.
y	either an object of class psData—see readData or an object of class psFit—see fitDist.
xname	an optional name for the first survey object.
yname	an optional name for the second survey object.
alternative	one of "two.sided", "less", or "greater", depending on the type of hypothesis test you wish to carry out. These may be replaced by single letter (or more) abbreviations.
null.value	the true value of the difference in the shape parameters under the null hypothesis.

print	if TRUE then the function will print summary output to the screen. This lets output be suppressed in situations where the user wants the function to run silently.
...	further arguments to be passed to or from methods.

Details

This function **only** works for the zeta distribution. It does not work for the zero-inflated zeta distribution. If the results from fitting ZIZ models are passed to this function, then it will ignore the zero-inflated part and simply refit a zeta model.

There is very little reason for null.value to be set to be anything other than 0. However it has been included for flexibility.

alternative = "greater" is the alternative that x has a larger shape parameter than y. alternative = "less" is the alternative that x has a smaller shape parameter than y.

Value

The function returns a list of class "htest" with the following elements:

statistic – the test statistic.

p.value – the P-value associated with the estimate.

estimate – the estimated difference in the shape parameters.

null.value – the specified hypothesized value of the difference in shape parameters—0 by default.

stderr – the standard error of the difference.

alternative – a character string describing the alternative hypothesis.

method – a character string describing the method.

data.name – a character string with the names of the two input data sets separated by " and ".

Methods (by class)

- compareSurveys(default): Compare two surveys on the basis of their shape parameters
- compareSurveys(psData): Compare two surveys on the basis of their shape parameters
- compareSurveys(psFit): Compare two surveys on the basis of their shape parameters

Functions

- compare.surveys(): Compare two surveys on the basis of their shape parameters
- comp.survs(): Compare two surveys on the basis of their shape parameters

Examples

```
data(Psurveys)
lau = Psurveys$lau
jackson = Psurveys$jackson
compareSurveys(lau, jackson)

## Example with fitted objects - note the function just refits the models
```

```
fit.lau = fitDist(lau)
fit.jackson = fitDist(jackson)
compareSurveys(fit.lau, fit.jackson)

## Example with a bigger difference
compareSurveys(Psurveys$roux, lau)
```

`compareSurveysLRT` *Compare two or more surveys on the basis of their shape parameters using a Likelihood Ratio Test*

Description

Compare two or more surveys on the basis of their shape parameters using a Likelihood Ratio Test

Usage

```
compareSurveysLRT(...)
```

Arguments

`...`	two or more objects of class `"psData"`—see `readData`.

Details

This function **only** works for the zeta distribution. The function carries out a likelihood ratio test (LRT) to test the null hypothesis

$$H_0 : \alpha_1 = \alpha_2 = \ldots = \alpha_K$$

versus the alternative

$$H_1 : \alpha_i \neq \alpha_j \text{ for some } i \neq j \in \{1, \ldots, K\},$$

where α_i is the shape parameter for the zeta distribution of the i^{th} survey.

Value

The function returns a `list` of class `"htest"` with the following elements:

`statistic` – the test statistic.

`parameter` – the degrees of freedom for the test

`p.value` – the P-value associated with the estimate.

`method` – a character string describing the method hypothesis.

`data.name` – the names of the data sets used in the test

Examples

```
data(Psurveys)
lau = Psurveys$lau
jackson = Psurveys$jackson
compareSurveysLRT(lau, jackson)

## Example with three surveys
roux = Psurveys$roux
compareSurveysLRT(lau, jackson, roux)
```

`confint.psFit` *S3 confint method for objects of class psFit*

Description

S3 confint method for objects of class psFit

Usage

```
## S3 method for class 'psFit'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	An object of class `psFit` returned by `fit()`.
`parm`	added for compatibility. Should be left empty as it is ignored.
`level`	the confidence level required—restricted to [0.75, 1)
`...`	in theory other parameters to be passed to `confint`, but in reality passed as extra parameters to the internal function `plZIZ`.

Details

NOTE: the method for ZIZ model is a little computationally intensive and possibly (almost certainly) unstable.

Value

if the zeta model is used (i.e `object` comes from a call to `fit(..., model = zetaModel())`), then a list with two items: `wald` and `prof` containing the Wald and profile likelihood confidence intervals respectively for the shape parameter of the fitted zeta distribution is returned. In general these should be relatively close to each other. These values use the zeta distribution shape parameter and must satisfy `shape > 1`. If a zero-inflated zeta model is used then a list of a confidence regions is returned with and element for each value of `level`. The confidence regions are `data.frames` with variables `pi` and `shape` which can be used with `lines` or `polygon` to draw a the confidence region.

Examples

```
data(Psurveys)
roux = Psurveys$roux
mleFit = fit(roux, model = zetaModel())
confint(mleFit)

if (interactive()) {
  zizFit = fit(roux, model = zizModel())
  confint(zizFit, level = c(0.80, 0.95))
  plotUncertainty(zizFit, level = c(0.80, 0.95))
}
```

`credint` *Bayesian credible intervals or regions*

Description

Extract one-dimensional credible intervals or two-dimensional credible regions from a fitted Bayesian uncertainty representation. Parametric Bayesian fits reuse the posterior representation stored by [fit()], while Rubin Bayesian Bootstrap objects reuse their stored weighted-fit parameter replicates.

Usage

```
credint(
  psFit,
  level = 0.95,
  plot = FALSE,
  silent = FALSE,
  parameters = NULL,
  nGrid = 401,
  ...
)

credInt(
  psFit,
  level = 0.95,
  plot = FALSE,
  silent = FALSE,
  parameters = NULL,
  nGrid = 401,
  ...
)
```

Arguments

`psFit`	A Bayesian ‘psFit‘ object or a ‘psBayesianBootstrap‘ object.

level	One or more credible probability levels strictly between zero and one.
plot	Logical; if 'TRUE', draw the corresponding uncertainty display with [plotUncertainty()] after extracting the interval or region.
silent	Logical; retained for backward compatibility. If 'FALSE', a short progress message is printed for two-dimensional regions.
parameters	Optional character vector naming one or two parameters. By default all model parameters are used when there are at most two.
nGrid	Number of grid points used when a one-dimensional posterior density must be evaluated for interval extraction.
...	Additional graphical arguments passed to [plotUncertainty()] when 'plot = TRUE'.

Details

'credint()' is the Bayesian numerical extractor corresponding to the visual [plotUncertainty()] interface. It does not rerun posterior fitting merely to obtain an interval or region.

For one-dimensional numerical posteriors, credible limits are interpolated from the stored cumulative posterior representation. For MCMC posteriors, stored draws are used. Importance-sampling regions preserve their stored importance weights, Laplace regions use the stored Gaussian covariance approximation, and two-dimensional numerical posteriors use the stored posterior grid and quadrature mass. Rubin Bayesian Bootstrap regions are derived from the stored weighted-fit parameter replicates.

The returned intervals are equal-tailed in one dimension. Sample-based two-dimensional regions use probability-content KDE contours; numerical posterior grids use highest-density probability-content contours; Laplace regions use the corresponding Gaussian probability ellipse.

Value

For one parameter, a named numeric vector for one requested level or a data frame of lower and upper limits for multiple levels. For two parameters, a named list of credible-region contour coordinates. Each contour records its probability content and density threshold where one is available.

Functions

- `credInt()`: Bayesian credible intervals or regions

Examples

```
if (interactive()) {
  data(Psurveys)
  bayesFit = fit(
    Psurveys$roux,
    model = zizModel(),
    method = "bayes",
    bayesOptions = list(posteriorMethod = "numerical")
  )
  credint(bayesFit, level = c(0.80, 0.95))
  plotUncertainty(bayesFit, level = c(0.80, 0.95))
}
```

deviance.psFit — *Extract deviance from a fitted fitPS model*

Description

Extract deviance from a fitted fitPS model

Usage

```
## S3 method for class 'psFit'
deviance(object, ...)
```

Arguments

object	An object of class ‘psFit‘.
...	Additional arguments retained for S3 compatibility.

Value

Scalar model deviance at the fitted representative parameter value.

DIC — *Compute the deviance information criterion for a Bayesian fitPS model*

Description

DIC is computed as ‘2 * E[D(theta) | y] - D(E[theta | y])‘, where ‘D(theta) = -2 log L(theta)‘. Numerical integration, MCMC draws, and importance weights are taken from the common posterior representation.

Usage

```
DIC(object, ...)
```

Arguments

object	A Bayesian object of class ‘psFit‘.
...	Additional arguments reserved for future controls.

Details

Laplace fits currently do not retain enough posterior integration information for a stable DIC calculation and therefore fail explicitly rather than using an unrelated approximation silently.

Value

A scalar numeric DIC value with ‘pD‘, ‘Dbar‘, and ‘Dhat‘ attributes.

`fit` *Fit a fitPS model*

Description

Fits a ‘psModel‘ object to forensic P- or S-survey data using maximum likelihood, parametric Bayesian inference, the ordinary nonparametric bootstrap, or Rubin’s Bayesian Bootstrap. ‘fit()‘ is the canonical public entry point for model fitting and uncertainty procedures.

Usage

```
fit(
  x,
  model,
  nterms = 10,
 method = c("mle", "bayes", "bootstrap", "bayesianBootstrap", "integrate", "numerical",
    "mcmc", "laplace", "importance"),
  prior,
  bayesOptions = NULL,
  B = 2000,
  level = 0.95,
  seed = NULL,
  silent = FALSE,
  parallel = TRUE,
  progressBar = FALSE,
  pbopts = list(type = "txt"),
  ...
)
```

Arguments

`x`	An object of class ‘psData‘.
`model`	A model descriptor inheriting from ‘psModel‘, such as an object returned by ‘zetaModel()‘, ‘zizModel()‘, or ‘logarithmicModel()‘.
`nterms`	Number of fitted P/S probability terms to retain.
`method`	Inferential method. Use ‘"mle"‘ for maximum likelihood, ‘"bayes"‘ for parametric Bayesian inference, ‘"bootstrap"‘ for the observation-level nonparametric bootstrap, or ‘"bayesianBootstrap"‘ for Rubin’s Bayesian Bootstrap. Legacy Bayesian engine names remain accepted and are normalised to ‘"bayes"‘.
`prior`	Optional prior used for parametric Bayesian fitting. For external Bayesian models this may be any model-specific object understood by ‘modelLogPrior()‘.
`bayesOptions`	Optional parametric Bayesian fitting controls.
`B`	Number of bootstrap or Bayesian Bootstrap replicates.
`level`	Confidence or equal-tail interval level used by bootstrap methods.
`seed`	Optional random-number seed used by bootstrap methods.

silent	Logical; suppress ordinary bootstrap progress messages when ‘TRUE‘.
parallel	Logical; use parallel ordinary-bootstrap fitting when ‘TRUE‘.
progressBar	Logical; display an ordinary-bootstrap progress bar when ‘TRUE‘.
pbopts	Options passed to ‘pbapply::pboptions()‘ when an ordinary bootstrap progress bar is requested.
...	Additional controls passed to the model-specific fitting path.

Value

For ‘method = "mle"‘ or ‘"bayes"‘, an object of class ‘psFit‘. For ‘method = "bootstrap"‘, a maximum-likelihood ‘psFit‘ with a ‘psBootstrap‘ object attached as ‘bootstrap‘. For ‘method = "bayesianBootstrap"‘, a ‘psBayesianBootstrap‘ object.

References

Curran JM, Buzzini P, Trejos T (2024). “Estimating probability terms for the background presence of glass when considering activity in forensic casework.” *Forensic Science International*, **364**, 112221. doi:10.1016/j.forsciint.2024.112221.

Efron B (1979). “Bootstrap methods: Another look at the jackknife.” *The Annals of Statistics*, **7**(1), 1–26. doi:10.1214/aos/1176344552.

Rubin DB (1981). “The Bayesian bootstrap.” *The Annals of Statistics*, **9**(1), 130–134. doi:10.1214/aos/1176345338.

Examples

```
data(Psurveys)
zetaFit = fit(Psurveys$roux, model = zetaModel())
logFit = fit(Psurveys$roux, model = logarithmicModel())
if (interactive()) {
  bootFit = fit(
    Psurveys$roux,
    model = zizModel(),
    method = "bootstrap",
    B = 20,
    seed = 123,
    silent = TRUE,
    parallel = FALSE
  )
  bayesBoot = fit(
    Psurveys$roux,
    model = zetaModel(),
    method = "bayesianBootstrap",
    B = 20,
    seed = 123
  )
}
```

fitDist *Fit a Zeta Distribution to Forensic Data*

Description

This function uses maximum likelihood estimation (MLE), or Bayesian estimation (MCMC), to estimate the shape parameter of a zeta distribution from a set of observed counts for either the number of groups/sources of forensically interesting material (mostly glass or paint) recovered from clothing, or the number of fragments/particles in each group. This, in turn, allows the estimation of the P and S probabilities, as described by Evett and Buckleton (1990), which are used in computing the likelihood ratio (LR) for activity level propositions. The data arise from clothing surveys. The general method is described in Coulson et al. (2001), although poor typesetting and a lack of defined terms make it hard to follow. This package improves on the estimation in that linear interpolation is not required, and standard numerical optimisation is used instead. The zeta distribution has probability mass function

$$p(k) = \frac{k^{-s}}{\zeta(s)}$$

where $\zeta(s)$ is the Riemann zeta function. Coulson et al. (2001) did not have an easy way to rapidly compute this quantity, hence their use of linear interpolation.

Usage

```
fitDist(
  x,
  nterms = 10,
 method = c("mle", "bayes", "integrate", "numerical", "mcmc", "laplace", "importance"),
  prior,
  bayesOptions = NULL,
  ...
)

fitdist(
  x,
  nterms = 10,
 method = c("mle", "bayes", "integrate", "numerical", "mcmc", "laplace", "importance"),
  prior,
  bayesOptions = NULL,
  ...
)
```

Arguments

x	an object of type psData, usually obtained from readData.
nterms	the number of terms to compute the probability distribution for.
method	primary fitting method. Use "mle" for maximum likelihood estimation or "bayes" for Bayesian estimation. Legacy Bayesian aliases "integrate", "numerical",

and "mcmc" are accepted with a deprecation warning and translated to method = "bayes" with the corresponding bayesOptions$posteriorMethod.

prior — optional prior object used by the Bayesian methods. This is retained for backward compatibility. New code should usually pass priors through bayesOptions. If omitted, makePrior() is used.

bayesOptions — optional list controlling Bayesian fitting. The posteriorMethod element selects "numerical", "mcmc", "laplace", or "importance". The default is "numerical". The prior element may contain a prior object returned by makePrior().

... — other arguments that control the estimation methods. If method == "mle", then the user can provide an optional argument start which is the starting value for the numerical optimisation. If this is not provided, then start = 1 by default. If you specify your own starting value, it must satisfy shape > 1.

If method == "bayes", then there are five optional parameters (which, despite the documentation, are actually case-insensitive):

shape0 – The initial value of the shape parameter. The default is 2.

a – The lower bound for the default uniform prior on $\log(\text{shape} - 1)$. The default is -2.

b – The upper bound for the default uniform prior on $\log(\text{shape} - 1)$. The default is +2.

nIter – The number of samples to save from the chain. Must be greater than zero, and ideally greater than 1000.

nBurnIn – The number of samples to discard from the chain. Must be greater than zero. **NOTE**: the sampler runs for nIter + nBurnIn iterations, so you do not need to factor this number into your number of samples, nIter.

silent – A logical variable which allows the user to get a progress bar if they want. TRUE by default.

Details

The function returns an object of class psFit. Core fitted values are stored at the top level, while Bayesian uncertainty is stored in the attached psPosterior object. Important components include:

psData – an object of class psData–see readData,

fit – method-specific point-estimate information,

shape – the maximum likelihood estimate, or the posterior mean, of the shape parameter,

var.shape – the maximum likelihood estimate, or posterior estimate, of the variance of the shape parameter,

fitted – a named vector containing the first nterms of the fitted distribution.

model – set to "zeta" for this model.

method – the method of estimation used, either "mle" or "bayes".

posterior – for Bayesian fits, a psPosterior object containing parameter summaries, probability summaries, diagnostics, and the engine-specific representation,

chain – if method == "bayes", then this element will contain the Markov Chain from the sampler, that is, hopefully a sample from the posterior density of the shape parameter. If method == "mle", then this element does not exist.

The output can be used in a variety of ways. If the interest is just in the shape parameter estimate, then the shape member of the psFit object contains this information. It is also displayed along with a number of fitted probabilities by the print.psFit method. The fitted object can also be plotted using the plot method plot.psFit, and to create a probability function with probfun. The shape value stored in the fitted object is the zeta distribution shape parameter and must satisfy shape > 1.

This function implements both maximum likelihood estimation (MLE) and Bayesian estimation. Both modes of estimation require additional information such as starting values and parameters for priors. Please read the documentation for the ... argument closely because it explains what you can change and what the default values are.

Bayesian estimation uses the prior returned by makePrior and the posterior engine selected by bayesOptions$posteriorMethod. Plain zeta currently supports "numerical" and "mcmc". Both engines return the same psPosterior contract, including posterior parameter summaries and posterior summaries of the fitted P or S probabilities.

Value

an object of class psFit–see Details.

Functions

- fitdist(): Fit a Zeta Distribution to Forensic Data

Deprecated

'fitDist()' is retained as a compatibility wrapper. New code should use 'fit(x, model = zetaModel())'.

References

Coulson, S. A., Buckleton, J. S., Gummer, A. B., and Triggs, C.M., "Glass on clothing and shoes of members of the general population and people suspected of breaking crimes", Science & Justice 2001: 41(1): 39–48.

Evett, I. W. and Buckleton, J. S., "The interpretation of glass evidence. A practical approach", Journal of the Forensic Science Society 1990: 30(4): 215–223.

See Also

plot.psFit, print.psFit, probfun.

Examples

```
p = readData(system.file("extdata", "p.xlsx", package = "fitPS"))
fit = fitDist(p)
fit

## Compare to the Bayesian estimates
fit2 = fitDist(p, method = "bayes")
fit2

fit3 = fitDist(
  p,
```

```
  method = "bayes",
  bayesOptions = list(posteriorMethod = "numerical")
)
fit3
```

`fitlogDist` *Fit a logarithmic distribution to forensic data*

Description

Fits the logarithmic-series distribution to P- or S-survey data using maximum likelihood or the common fitPS Bayesian posterior architecture. The logarithmic model has probability mass function

$$p(k) = -\frac{\pi^k}{k \log(1-\pi)}, \quad 0 < \pi < 1.$$

Usage

```
fitlogDist(
  x,
  nterms = 10,
  method = c("mle", "bayes"),
  prior,
  bayesOptions = NULL,
  ...
)

fitLogdist(
  x,
  nterms = 10,
  method = c("mle", "bayes"),
  prior,
  bayesOptions = NULL,
  ...
)

fitlogdist(
  x,
  nterms = 10,
  method = c("mle", "bayes"),
  prior,
  bayesOptions = NULL,
  ...
)
```

Arguments

x	An object of class ‘psData‘.
nterms	Number of fitted P/S probability terms to retain.
method	Fitting method, either ‘"mle"‘ or ‘"bayes"‘.
prior	Optional ‘psPrior‘ used for Bayesian fitting. When omitted, a uniform prior on ‘(0.001, 0.999)‘ is used.
bayesOptions	Optional Bayesian controls. The logarithmic model supports ‘"numerical"‘ and ‘"mcmc"‘ posterior engines.
...	Additional fitting controls. For MLE, ‘start‘ is an optional initial value in ‘(0, 1)‘. For MCMC, ‘pi0‘, ‘nIter‘, ‘nBurnIn‘, and ‘silent‘ are passed to the logarithmic MCMC model method.

Value

An object of class ‘psFit‘.

Deprecated

‘fitlogDist()‘ is retained as a compatibility wrapper. New code should use ‘fit(x, model = logarithmicModel())‘.

Examples

```
data(Psurveys)
fit = fitlogDist(Psurveys$roux)
fit
```

---

fitted.psFit *S3 fitted method for an object of class* psFit

Description

S3 fitted method for an object of class psFit

Usage

```
## S3 method for class 'psFit'
fitted(
  object,
  n = NULL,
  type = c("plugIn", "posteriorMean", "bootstrapMean"),
  ...
)
```

Arguments

object	an object of class psFit, usually from fitDist or fitZIDist.
n	This parameter is NULL by default. If it is not NULL then it must be either the number of fitted terms to be returned, or a vector containing the desired fitted values.
type	The fitted-value definition. ‘"plugIn"‘ preserves the existing behaviour and evaluates probabilities at fitted parameter values. ‘"posteriorMean"‘ returns posterior mean probabilities for Bayesian fits. ‘"bootstrapMean"‘ returns bootstrap mean probabilities when a bootstrap distribution has been attached to the fit.
...	other arguments passed to fitted—not used.

Value

a named vector of fitted probabilities

---

fitted.psPosterior *Extract posterior mean probabilities*

Description

Extract posterior mean probabilities

Usage

```
## S3 method for class 'psPosterior'
fitted(object, ...)
```

Arguments

object	An object of class ‘psPosterior‘.
...	Additional arguments retained for S3 compatibility.

Value

A named numeric vector of posterior mean probabilities.

`fitZIDist` *Fit a Zero-Inflated Zeta Distribution to Forensic Data*

Description

This function uses maximum likelihood estimation (MLE) or Bayesian estimation to estimate the mixing parameter and the shape parameter of a zero-inflated zeta distribution from a set of observed counts for either the number of groups/sources of forensically interesting material (mostly glass or paint) recovered from clothing, or the number of fragments/particles in each group. This, in turn, allows the estimation of the P and S probabilities, as described by Evett and Buckleton (1990), which are used in computing the likelihood ratio (LR) for activity level propositions. The data arise from clothing surveys. The zero-inflated zeta distribution has probability mass function

$$p(k) = \begin{cases} \pi + \frac{(1-\pi)}{\zeta(s)} & , k = 0, \\ \frac{(1-\pi)k^{-s}}{\zeta(s)} & , k = 1, 2, \ldots \end{cases}$$

where $\zeta(s)$ is the Riemann zeta function.

Usage

```
fitZIDist(
  x,
  nterms = 10,
 method = c("mle", "bayes", "integrate", "numerical", "mcmc", "laplace", "importance"),
  prior,
  bayesOptions = NULL,
  ...
)

fitZIdist(
  x,
  nterms = 10,
 method = c("mle", "bayes", "integrate", "numerical", "mcmc", "laplace", "importance"),
  prior,
  bayesOptions = NULL,
  ...
)

fitzidist(
  x,
  nterms = 10,
 method = c("mle", "bayes", "integrate", "numerical", "mcmc", "laplace", "importance"),
  prior,
  bayesOptions = NULL,
  ...
)
```

Arguments

x	an object of type psData, usually obtained from readData.
nterms	the number of terms to compute the probability distribution for.
method	primary fitting method. Use "mle" for maximum likelihood estimation or "bayes" for Bayesian estimation. Legacy Bayesian aliases "integrate", "numerical", "mcmc", "laplace", and "importance" are accepted with a deprecation warning and translated to method = "bayes" with the corresponding bayesOptions$posteriorMethod.
prior	optional prior object used by Bayesian posterior approximation methods where applicable. This is retained for consistency with fitDist(); new code should usually pass priors through bayesOptions.
bayesOptions	optional list controlling Bayesian fitting. The posteriorMethod element selects "numerical", "mcmc", "laplace", or "importance". The default is "numerical". The prior element may contain a prior object returned by makePrior().
...	other arguments that control the estimation methods. If method == "mle", then the user can provide an optional argument start which is the starting value for the numerical optimisation. If this is not provided, then start = c(0.5, 2) by default. If you specify your own starting value, keep the mixing parameter greater than 0.5 and use shape > 1. If method == "bayes", engine-specific controls can be supplied through .... Common MCMC controls include: theta0 – The initial values of the mixing parameter and shape parameter. The default is c(0.5, 2). shape1 – The first shape parameter for the beta prior on the mixing distribution, Beta(shape1, shape2). The default is 1. shape2 – The second shape parameter for the beta prior on the mixing distribution, Beta(shape1, shape2). The default is 1. nIter – The number of samples to save from the chain. Must be greater than zero, and ideally greater than 1000. nBurnIn – The number of samples to discard from the chain. Must be greater than zero. **NOTE**: the sampler runs for nIter + nBurnIn iterations, so you do not need to factor this number into your number of samples, nIter. silent – A logical variable which allows the user to get a progress bar if they want. TRUE by default.

Details

The function returns an object of class psFit. Core fitted values are stored at the top level, while Bayesian uncertainty is stored in the attached psPosterior object. Important components include:

psData – an object of class psData–see readData,

fit – method-specific point-estimate information,

pi - the maximum likelihood estimate, or the posterior mean, of the mixing parameter,

shape – the maximum likelihood estimate, or the posterior mean, of the shape parameter,

var.cov – for maximum-likelihood fits, the estimated variance-covariance matrix for the parameters,

`fitted` – a named `vector` containing the first `nterms` of the fitted distribution.

`model` – set to `"ziz"` for this model,

`method` – the method of estimation used, either `"mle"` or `"bayes"`,

`posterior` – for Bayesian fits, an object of class `psPosterior` containing posterior parameter summaries, posterior probability summaries, and the engine-specific posterior representation,

The output can be used in a variety of ways. If the interest is just in the mixing and shape parameter estimates, then the `pi` and `shape` members of the `psFit` object contain this information. It is also displayed along with a number of fitted probabilities by the `print.psFit` method. The fitted object can also be plotted using the plot method `plot.psFit`, and to create a probability function with `probfun`. The `shape` value stored in the fitted object is the zeta distribution shape parameter and must satisfy `shape > 1`.

This function implements both maximum likelihood estimation (MLE) and Bayesian estimation. Both modes of estimation require additional information such as starting values and parameters for priors. Please read the documentation for the `...` argument closely because it explains what you can change and what the default values are.

Bayesian zero-inflated zeta estimation is selected with `method = "bayes"`. The posterior approximation is selected with `bayesOptions$posteriorMethod`. The default, `"numerical"`, uses deterministic two-dimensional grid integration over `pi` and `shape`. The legacy Metropolis-Hastings sampler remains available with `bayesOptions = list(posteriorMethod = "mcmc")`. The prior for the mixing proportion is Beta(shape1, shape2), and the shape prior is supplied by `bayesOptions$prior` or `prior`. If no shape prior is supplied, `makePrior()` is used. Bayesian fits store a `psPosterior` object in `fit$posterior`; use `posteriorProbs()` for posterior means and credible intervals of the derived P or S probabilities. The four posterior engines are `"numerical"`, `"mcmc"`, `"laplace"`, and `"importance"`.

Value

an object of class `psFit`–see Details.

Deprecated

'fitZIDist()' is retained as a compatibility wrapper. New code should use 'fit(x, model = zizModel())'.

References

Evett, I. W. and Buckleton, J. S., "The interpretation of glass evidence. A practical approach", Journal of the Forensic Science Society 1990: 30(4): 215–223.

See Also

`plot.psFit`, `print.psFit`, `probfun`, `posteriorProbs`, `bootstrapProbs`.

Examples

```
data(Psurveys)
roux = Psurveys$roux
fit = fitZIDist(roux)
```

```
fit
```

`logarithmicModel` *Construct built-in fitPS model descriptors*

Description

These constructors create model objects for the built-in fitPS distributions. They can be supplied directly to [fit()]. The lower-level third-party model constructor and extension generics are introduced separately after the public contract has been validated.

Usage

```
logarithmicModel()

zetaModel()

zizModel()
```

Value

An object inheriting from ‘psModel‘ and the concrete built-in model class.

`logLik.psFit` *Extract the maximised log-likelihood from a fitPS model fit*

Description

Returns the model log-likelihood evaluated at the maximum-likelihood estimate. The returned ‘logLik‘ object includes the model parameter count and number of observations, allowing [stats::AIC()] and [stats::BIC()] to use the shared fitPS model-comparison contract.

Usage

```
## S3 method for class 'psFit'
logLik(object, ...)
```

Arguments

`object`	An object of class ‘psFit‘.
`...`	Additional arguments retained for S3 compatibility.

Details

Parametric Bayesian fits are rejected because their stored representative parameter values are posterior summaries rather than maximum-likelihood estimates. Bootstrap fits retain the underlying maximum-likelihood point fit and therefore remain valid for AIC and BIC calculations.

Value

An object of class ‘logLik‘ with ‘df‘ and ‘nobs‘ attributes.

`makePrior` *Define a prior density*

Description

Construct a prior that can be used in the `fitdist` function.

Usage

```
makePrior(family = c("loguniform", "uniform", "custom"), range, logd)
```

Arguments

`family`	One of `"loguniform"`, `"uniform"` or `"custom"`.
`range`	Optionally the range for which the prior density is evaluated. It is zero outside of this range. Loguniform priors retain the zeta shape > 1 domain; uniform and custom priors may use another finite ordered parameter range.
`logd`	Optionally (required when `family="custom"`.) a function that evaluates the log density of the prior inside the range.

Details

The default is a LogUniform[-2, 2] prior on `shape - 1`. Equivalently, the prior range on the fitPS standard zeta shape parameter is `1 + exp(c(-2, 2))`.

Value

an object of type `psPrior`

See Also

readData

Examples

```
## With default parameters, the prior will be LogUniform[-2, 2]
## on shape - 1, so the prior support is above shape = 1.
p1 <- makePrior()

# plot the prior density
xPlot <- seq(from = 1.01, to = 10, length = 100)
plot(xPlot, exp(p1$logd(xPlot)), type = "l")

# Alternatively, a Uniform[a, b] prior can be used on standard shape
p2 <- makePrior(family = "uniform", range = c(1.01, 10))
```

```
plot(xPlot, exp(p2$logd(xPlot)), type = "l")

# A custom prior needs the log density function to be specified
# We define an exponential prior with rate = 1 on shape - 1
logdexp <- function(x) dexp(x - 1, rate = 1 / 10, log = TRUE)
p3 <- makePrior(family = "custom", range = c(1.01, 10), logd = logdexp)

plot(xPlot, exp(p3$logd(xPlot)), type = "l")
```

`makePSData` *Create a survey data set manually*

Description

Create a survey data set from the command line rather than reading data in from a file. This function is likely to be only useful where there are a very small number of group sizes, or sizes of groups of glass.

Usage

```
makePSData(n, count = NULL, type = c("P", "S"), notes = NULL)

makeData(n, count = NULL, type = c("P", "S"), notes = NULL)

createPSData(n, count = NULL, type = c("P", "S"), notes = NULL)
```

Arguments

`n`	Either the number of groups of glass or the size of different groups of glass, or a `vector` of observed groups of glass, or group sizes. See details for a longer explanation.
`count`	Either the number of people in the survey sample who had n groups of glass on their clothing, or the number of people who had a group of glass of size n.
`type`	either `"P"` or `"S"`
`notes`	a `bibentry` or a character string which allows extra information about the data to be stored, such as the source, or reference. `NULL` by default.

Details

If `count` is `NULL`, then it is assumed that `n` consists of actual observed group sizes or numbers of groups of glass found on a survey of N individuals. That is, one could provide `n = rep(0:1, 98, 1)` or `n = 0:1, count = c(98, 1)`. The former is more useful when performing simulation studies.

Value

an object of type `psData`—see `readData` for more details.

See Also

readData

Examples

```
## recreate the data read in the readData example
p1 = makePSData(n = c(0, 1, 2), count = c(98, 1, 1), type = "P")
s1 = makePSData(n = 1:3, count = c(1, 1, 1), type = "S")
p1
s1
```

`mean.psData` *An S3 method for computing the mean of clothing survey for the number of groups or size of groups*

Description

An S3 method for computing the mean of clothing survey for the number of groups or size of groups

Usage

```
## S3 method for class 'psData'
mean(x, ...)
```

Arguments

`x`	an object of class `psData`—`readData` for more details.
`...`	other arguments which are passed to `sum`

Value

the mean of the data. If there are r_i observations of the value n_i then the mean is given by

$$\sum_i \frac{r_i \times n_i}{\sum_i r_i}$$

.

Examples

```
data(Psurveys)
mean(Psurveys$roux)
```

`modelBayesControl` *Return Bayesian fitting controls supplied by a model*

Description

This generic exposes model-owned Bayesian controls such as natural-scale starting values. Engine-owned controls such as iteration counts, tolerances, and random seeds remain arguments to the posterior engine.

Usage

```
modelBayesControl(model, x, engine, prior, ...)

## S3 method for class 'psModel'
modelBayesControl(model, x, engine, prior, ...)

## S3 method for class 'zetaModel'
modelBayesControl(model, x, engine, prior, ...)

## S3 method for class 'logarithmicModel'
modelBayesControl(model, x, engine, prior, ...)

## S3 method for class 'zizModel'
modelBayesControl(model, x, engine, prior, start = c(pi = 0.5, shape = 2), ...)
```

Arguments

`model`	A ‘psModel‘ object.
`x`	An object of class ‘psData‘.
`engine`	A posterior-engine object or engine name.
`prior`	A model-specific prior specification.
`...`	Additional model-specific control inputs.
`start`	Optional named numeric natural-scale starting values used by models that support caller-supplied Bayesian starts. Ignored by models that do not use this argument.

Value

A named list containing model-specific Bayesian controls. Generic engines may require a named numeric ‘start‘ component. One-dimensional numerical fitting also uses named numeric ‘lower‘ and ‘upper‘ components.

`modelFromUnconstrained` *Transform unconstrained coordinates to natural model parameters*

Description

This is the inverse of [modelToUnconstrained()]. Generic MCMC calls it for every proposed unconstrained state before evaluating the likelihood and prior, both of which remain defined on the model's natural parameter scale.

Usage

```
modelFromUnconstrained(model, unconstrained, ...)

## S3 method for class 'psModel'
modelFromUnconstrained(model, unconstrained, ...)

## S3 method for class 'zetaModel'
modelFromUnconstrained(model, unconstrained, ...)

## S3 method for class 'logarithmicModel'
modelFromUnconstrained(model, unconstrained, ...)

## S3 method for class 'zizModel'
modelFromUnconstrained(model, unconstrained, ...)
```

Arguments

`model`	A ‘psModel‘ object.
`unconstrained`	Named numeric coordinates on the unconstrained scale.
`...`	Additional model-specific transformation inputs.

Details

The transformation must be one-to-one with [modelToUnconstrained()] over the Bayesian parameter region, preserve the names and order returned by [modelParameterNames()], and return valid natural-scale parameters whenever its input is finite and valid for the chosen transformation. The default ‘psModel‘ method is the identity transformation.

For example, if ‘modelToUnconstrained()‘ uses ‘log(theta)‘ for a positive parameter, this method uses ‘exp(z)‘. If it uses ‘qlogis(p)‘ for a probability, this method uses ‘plogis(z)‘.

Value

A named numeric vector of natural-scale model parameters, with names and order matching ‘modelParameterNames(model)‘.

See Also

[modelToUnconstrained()], [modelLogJacobian()]

`modelLogJacobian` *Evaluate the log-Jacobian for the unconstrained parameter transformation*

Description

When MCMC samples an unconstrained vector ‘z‘ but the model prior is defined for natural parameters ‘theta = modelFromUnconstrained(model, z)‘, the posterior density on the sampled scale contains the change-of-variables factor

Usage

```
modelLogJacobian(model, unconstrained, ...)

## S3 method for class 'psModel'
modelLogJacobian(model, unconstrained, ...)

## S3 method for class 'zetaModel'
modelLogJacobian(model, unconstrained, ...)

## S3 method for class 'logarithmicModel'
modelLogJacobian(model, unconstrained, ...)

## S3 method for class 'zizModel'
modelLogJacobian(model, unconstrained, ...)
```

Arguments

`model`	A ‘psModel‘ object.
`unconstrained`	Named numeric coordinates on the unconstrained scale.
`...`	Additional model-specific transformation inputs.

Details

‘log |det(d theta / d z)|‘.

This generic returns exactly that quantity: the log absolute determinant of the Jacobian of the transformation **from unconstrained coordinates back to natural parameters**. It is not the Jacobian of [modelToUnconstrained()]. Generic MCMC adds this value to the natural-scale log likelihood and log prior.

For an identity transformation the value is zero. For ‘theta = exp(z)‘ it is ‘z‘. For ‘p = plogis(z)‘ it is ‘log(p) + log1p(-p)‘. For several independent component-wise transformations, return the sum of their log-Jacobian terms; for a genuinely coupled transformation, return the log absolute determinant of the full inverse-transformation Jacobian matrix.

Value

One finite numeric value equal to ‘log |det(d theta / d z)|‘ for the inverse transformation.

See Also

[modelToUnconstrained()], [modelFromUnconstrained()]

`modelLogLikelihood` *Evaluate a model log likelihood through the shared model protocol.*

Description

Concrete likelihood methods implement the model-specific mathematics behind the shared interface. The base method fails explicitly so an unsupported model cannot silently use the wrong likelihood.

Usage

```
modelLogLikelihood(model, parameters, data, ...)
```

Arguments

`model`	A ‘psModel‘ object.
`parameters`	Named model parameters.
`data`	Model-specific observation data.
`...`	Additional model-specific inputs.

Value

A numeric log likelihood when implemented by a concrete model.

`modelLogPrior` *Evaluate a model prior on the natural parameter scale*

Description

External Bayesian models implement this generic to provide the complete log-prior density for a named natural-scale parameter vector. The prior object is deliberately model-defined so multi-parameter and correlated priors do not have to use the legacy one-dimensional ‘psPrior‘ structure.

Usage

```
modelLogPrior(model, parameters, prior, ...)

## S3 method for class 'psModel'
modelLogPrior(model, parameters, prior, ...)

## S3 method for class 'zetaModel'
modelLogPrior(model, parameters, prior, ...)

## S3 method for class 'logarithmicModel'
modelLogPrior(model, parameters, prior, ...)

## S3 method for class 'zizModel'
modelLogPrior(model, parameters, prior, shape1 = 1, shape2 = 1, ...)
```

Arguments

model	A ‘psModel‘ object.
parameters	Named natural-scale model parameters.
prior	A model-specific prior specification.
...	Additional model-specific prior inputs.
shape1, shape2	Positive beta-prior shape parameters used by models that place a beta prior on a probability parameter. Ignored by models that do not use these arguments.

Value

One numeric log-prior density value, or ‘-Inf‘ outside prior support.

---

modelMleControl *Return generic maximum-likelihood controls for a model*

Description

External models that use fitPS-owned numerical optimisation may store their starting values and bounds in [psModel()] or override this generic.

Usage

```
modelMleControl(model, x, ...)

## S3 method for class 'psModel'
modelMleControl(model, x, ...)
```

Arguments

model	A ‘psModel‘ object.
x	An object of class ‘psData‘.
...	Additional model-specific fitting controls.

Value

A list with named numeric ‘start‘, ‘lower‘, and ‘upper‘ components.

---

modelObservationData *Convert fitPS data to the observation support required by a model.*

Description

External methods are responsible for mapping the P/S survey labels onto the model's natural support. For example, a zero-based distribution may use P0, P1, ... directly but map S1, S2, ... to support values 0, 1, ... so the probability sequence is shifted without truncation or renormalisation.

Usage

```
modelObservationData(model, x, ...)
```

Arguments

model	A ‘psModel‘ object.
x	An object of class ‘psData‘.
...	Additional arguments reserved for model methods.

Value

Model-specific observation data.

---

modelParameterNames *Return the natural parameter names declared by a fitPS model.*

Description

Return the natural parameter names declared by a fitPS model.

Usage

```
modelParameterNames(model, ...)
```

Arguments

model	A ‘psModel‘ object.
...	Additional arguments reserved for model methods.

Value

Character vector of parameter names.

---

modelProbabilities *Evaluate fitted P/S probabilities through a model descriptor.*

Description

Evaluate fitted P/S probabilities through a model descriptor.

Usage

```
modelProbabilities(model, parameters, n, type, ...)
```

Arguments

model	A ‘psModel‘ object.
parameters	Named model parameters.
n	Requested P/S probability indices. These are survey labels, not necessarily the model’s natural support values; concrete methods should perform any required support shift before evaluating probabilities.
type	Survey type, either ‘"P"‘ or ‘"S"‘.
...	Additional arguments reserved for model methods.

Value

Numeric matrix of fitted P/S probabilities.

`modelToUnconstrained` *Transform natural model parameters to unconstrained coordinates*

Description

Generic MCMC proposes parameter values on an unconstrained Euclidean scale. This avoids proposals that repeatedly leave a model's natural parameter space, such as negative values for a positive rate or values outside ‘(0, 1)‘ for a probability. External models with constrained parameters implement this generic to map their named natural-scale parameter vector to a named vector whose components may vary over the whole real line.

Usage

```
modelToUnconstrained(model, parameters, ...)

## S3 method for class 'psModel'
modelToUnconstrained(model, parameters, ...)

## S3 method for class 'zetaModel'
modelToUnconstrained(model, parameters, ...)

## S3 method for class 'logarithmicModel'
modelToUnconstrained(model, parameters, ...)

## S3 method for class 'zizModel'
modelToUnconstrained(model, parameters, ...)
```

Arguments

`model`	A ‘psModel‘ object.
`parameters`	Named numeric model parameters on their natural scale.
`...`	Additional model-specific transformation inputs.

Details

The transformation must be one-to-one with [modelFromUnconstrained()] over the parameter region used for Bayesian fitting, preserve the names and order returned by [modelParameterNames()], and return finite values for valid interior natural-scale parameters. The default ‘psModel‘ method is the identity transformation, which is appropriate when every natural parameter is already unconstrained.

Common choices include ‘log(theta)‘ for a positive parameter ‘theta‘ and ‘qlogis(p)‘ for a probability ‘p‘. If several parameters are transformed, the returned vector contains one unconstrained coordinate for each natural parameter.

Value

A named numeric vector of unconstrained coordinates, with names and order matching ‘modelParameterNames(model)‘.

See Also

[modelFromUnconstrained()], [modelLogJacobian()]

`plot.psBootstrap` *Plot bootstrap probability summaries*

Description

Plot bootstrap mean probabilities and percentile confidence intervals for the fitted P or S probabilities stored in a `psBootstrap` object.

Usage

```
## S3 method for class 'psBootstrap'
plot(
  x,
  n = NULL,
  showInterval = TRUE,
  ylim = NULL,
  xlab = "Probability term",
  ylab = "Bootstrap probability",
  main = "Bootstrap probability summaries",
  pch = 19,
  ...
)
```

Arguments

`x`	An object of class `psBootstrap`.
`n`	`NULL`, a single number of leading probabilities to plot, or a vector of P or S indices to plot.
`showInterval`	Logical; if `TRUE`, draw the stored percentile confidence intervals.
`ylim`	Optional numeric vector giving the y-axis limits.
`xlab`, `ylab`, `main`	Optional plot labels.
`pch`	Plotting symbol used for bootstrap means.
`...`	Additional graphical arguments passed to `plot`.

Value

Invisibly returns the plotted bootstrap probability summary data frame.

Examples

```
if (interactive()) {
data(Psurveys)
fit = fit(
  Psurveys$roux,
  model = zizModel(),
  method = "bootstrap",
  nterms = 4,
  B = 20,
  seed = 123,
  silent = TRUE,
  parallel = FALSE
)
plot(fit$bootstrap, n = 4)

}
```

`plot.psFit` *S3 plot method for an object of class* `psFit`

Description

S3 plot method for an object of class `psFit`

Usage

```
## S3 method for class 'psFit'
plot(
  x,
  ylim = c(0, 1),
  conf = FALSE,
  conf.level = 0.95,
  ci.type = c("wald", "prof"),
  log.scale = FALSE,
  ...
)
```

Arguments

`x`	an object of class `psFit`, usually from `fitDist` or `fitZIDist`.
`ylim`	the limits of the y-axis.
`conf`	if `TRUE`, and the model is the the zeta model (as opposed to the zero-inflated zeta (ZIZ), then confidence intervals (based on the standard error of the shape parameter) are drawn on the plot. If the ZIZ model has been used, then this is ignored.
`conf.level`	the confidence level for the confidence intervals. Must be between 0.75 and 0.99.

ci.type	Specifies the type of confidence interval. If conf == TRUE, then then ci.type can be either "wald" "prof" (or an abbreviation), depending on whether the Wald interval or the profile likelihood interval should be used. Note that these are intervals on the shape parameter and not the density heights. Therefore the intervals around the probabilities should not really be thought of as confidence intervals but rather something more similar to a "sensitivity" interval.
log.scale	if TRUE the y-axis is changed to a logarithmic (base 10) axis.
...	other arguments passed to plot.

Value

No return value, called for side effects

Examples

```
p = readData(system.file("extdata", "p.xlsx", package = "fitPS"))
fit = fitDist(p)
plot(fit)

## An example with Wald generated intervals
plot(fit, conf = TRUE)

plot(fit, conf = TRUE, ci.type = "p")
```

---

plot.psPosterior *Plot posterior probability summaries*

Description

Plot posterior means and equal-tailed credible intervals for the fitted P or S probabilities stored in a psPosterior object.

Usage

```
## S3 method for class 'psPosterior'
plot(
  x,
  n = NULL,
  showInterval = TRUE,
  ylim = NULL,
  xlab = "Probability term",
  ylab = "Posterior probability",
  main = "Posterior probability summaries",
  pch = 19,
  ...
)
```

Arguments

x	An object of class psPosterior.
n	NULL, a single number of leading probabilities to plot, or a vector of P or S indices to plot.
showInterval	Logical; if TRUE, draw the stored credible intervals.
ylim	Optional numeric vector giving the y-axis limits.
xlab, ylab, main	Optional plot labels.
pch	Plotting symbol used for posterior means.
...	Additional graphical arguments passed to plot.

Details

The points are posterior mean probabilities. The vertical intervals are the equal-tailed credible intervals stored in the posterior object. This method does not change plot.psFit(), which continues to display fitted model probabilities and observed proportions.

Value

Invisibly returns the plotted posterior probability summary data frame.

Examples

```
if (interactive()) {
  data(Psurveys)
  fit = fit(
    Psurveys$roux,
    model = zizModel(),
    method = "bayes",
    bayesOptions = list(posteriorMethod = "numerical")
  )
  plot(fit$posterior, n = 5)
}
```

---

plotPosterior *Plot a posterior density for a fitted power-series model*

Description

Plot the marginal posterior density for a parameter in a Bayesian ‘psFit‘ object. The posterior density is estimated from stored MCMC samples when they are available. For numerical integration fits, the stored posterior density function is evaluated on a grid.

Usage

```
plotPosterior(
  object,
  parameter = "shape",
  level = 0.95,
  showEstimate = TRUE,
  showInterval = TRUE,
  nGrid = 512,
  xlab = NULL,
  ylab = "Posterior density",
  main = NULL,
  ...
)
```

Arguments

object	an object of class psFit, usually returned by fit with method = "bayes".
parameter	character; the posterior parameter to plot. The default is "shape". Zero-inflated Bayesian fits also support "pi".
level	numeric; credible level for the interval, if displayed.
showEstimate	logical; if TRUE, draw a vertical line at the posterior point estimate stored in the fitted object.
showInterval	logical; if TRUE, draw vertical lines for the equal-tail credible interval.
nGrid	integer; number of grid points used when a stored posterior density function is evaluated directly.
xlab, ylab, main	optional plot labels.
...	other graphical arguments passed to plot.

Details

This function intentionally does not overload plot.psFit, which continues to plot fitted probabilities. For MCMC fits, the density is estimated from stored posterior samples. For numerical Bayesian fits, the stored marginal posterior density is evaluated on an automatically chosen grid. The hatched credible region follows the posterior density between the equal-tail interval endpoints.

Value

Invisibly returns a data frame containing the plotted posterior density grid. The return value has attributes named "estimate" and "interval" when those values are available.

Examples

```
if (interactive()) {
  data(Psurveys)
  roux = Psurveys$roux
  bayesFit = fit(roux, model = zizModel(), method = "bayes")
  plotPosterior(bayesFit, parameter = "pi")
```

```
}
```

`plotUncertainty` *Plot parameter uncertainty for a fitPS inferential result*

Description

Draw a common one- or two-parameter uncertainty display while preserving the statistical interpretation of the fitted method. Maximum-likelihood fits use profile-likelihood confidence intervals or regions, ordinary bootstrap fits use the stored bootstrap parameter distribution, parametric Bayesian fits use the stored posterior representation, and Rubin Bayesian Bootstrap fits use the stored weighted-fit parameter distribution.

Usage

```
plotUncertainty(object, ...)

## S3 method for class 'psFit'
plotUncertainty(
  object,
  level = c(0.8, 0.95),
  parameters = NULL,
  showPoints = NULL,
  nGrid = 401,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ...
)

## S3 method for class 'psBootstrap'
plotUncertainty(
  object,
  level = c(0.8, 0.95),
  parameters = NULL,
  showPoints = TRUE,
  nGrid = 401,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ...
)

## S3 method for class 'psBayesianBootstrap'
plotUncertainty(
  object,
```

```
  level = c(0.8, 0.95),
  parameters = NULL,
  showPoints = TRUE,
  nGrid = 401,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ...
)
```

Arguments

`object`	A ‘psFit‘, ‘psBootstrap‘, or ‘psBayesianBootstrap‘ object.
`...`	Additional graphical arguments passed to the initial plot call.
`level`	One or more probability levels strictly between zero and one.
`parameters`	Optional character vector naming one or two parameters to display. By default all available parameters are used when there are at most two.
`showPoints`	Logical or ‘NULL‘; for sample-based two-dimensional displays, show the stored parameter replicates behind the contours. When ‘NULL‘, ordinary and Rubin Bayesian Bootstrap displays show their realizations by default, while parametric Bayesian displays do not.
`nGrid`	Number of points used for one-dimensional likelihood/density displays.
`xlab`, `ylab`, `main`	Optional plot labels.

Details

The visual grammar is deliberately consistent across inferential methods, but the regions are not interchangeable. MLE regions are profile-likelihood confidence regions. Ordinary bootstrap regions are smoothed confidence regions based on the bootstrap estimator distribution. Parametric Bayesian regions are posterior credible regions. Rubin Bayesian Bootstrap regions describe the distribution of weighted maximum-likelihood fits induced by Bayesian Bootstrap weights.

For two-dimensional numerical Bayesian fits, fitPS reuses the stored posterior summary grid. Probability-content contours are obtained by ordering the stored grid points by posterior density and cumulatively summing their stored quadrature mass; plotting therefore does not rerun adaptive cubature. For MCMC and bootstrap representations, contours are obtained from the stored parameter draws using kernel density estimation. Ordinary and Rubin Bayesian Bootstrap contours use an unconstrained bivariate KDE so that boundary transformations do not distort the observed joint replicate geometry. Importance-sampling posteriors retain their sample weights for weighted KDE contours, while Laplace fits use Gaussian contours implied by the stored local covariance approximation.

Value

Invisibly returns a list describing the plotted uncertainty representation, including its inferential interpretation and interval or contour coordinates. Sample-based two-dimensional results also include empirical containment of the stored realizations for each contour.

Examples

```
if (interactive()) {
  data(Psurveys)
  roux = Psurveys$roux

  mleFit = fit(roux, model = zizModel())
  plotUncertainty(mleFit, level = c(0.80, 0.95))

  bayesFit = fit(
    roux,
    model = zizModel(),
    method = "bayes",
    bayesOptions = list(posteriorMethod = "numerical")
  )
  plotUncertainty(bayesFit, level = c(0.80, 0.95))
}
```

`posteriorInflation` *Posterior probability of practically negligible inflation*

Description

Calculate the posterior probability that the zero/one-inflation parameter ‘pi‘ is smaller than a user-specified practical threshold.

Usage

```
posteriorInflation(object, ...)

## S3 method for class 'psFit'
posteriorInflation(object, epsilon = 0.01, ...)

## S3 method for class 'psPosterior'
posteriorInflation(object, epsilon = 0.01, ...)
```

Arguments

`object`	A Bayesian zero-inflated ‘psFit‘ object or its associated ‘psPosterior‘ object.
`...`	Additional arguments passed to methods.
`epsilon`	A number strictly between 0 and 1 defining the largest inflation probability regarded as practically negligible.

Details

With the continuous beta prior used for ‘pi‘, the posterior probability of exactly ‘pi = 0‘ is zero. A more practically useful diagnostic is therefore ‘Pr(pi < epsilon | data)‘, where ‘epsilon‘ is chosen to represent an inflation effect small enough to be negligible for the application.

The calculation respects the stored posterior engine. Numerical fits integrate the marginal posterior grid, MCMC fits use retained draws, importance fits retain their importance weights, and Laplace fits evaluate the implied Gaussian approximation on the logit scale.

Value

A data frame with the threshold and posterior probabilities below and at or above that threshold.

Examples

```
data(Psurveys)
roux = Psurveys$roux
if (interactive()) {
  fit = fitZIDist(
    roux,
    method = "bayes",
    bayesOptions = list(posteriorMethod = "numerical")
  )
  posteriorInflation(fit, epsilon = 0.01)
}
```

---

posteriorProbs *Extract posterior probability summaries*

Description

Extract posterior summaries for fitted P or S probabilities from a Bayesian fitPS model or from its associated posterior object.

Usage

```
posteriorProbs(object, ...)

## S3 method for class 'psFit'
posteriorProbs(object, n = NULL, ...)

## S3 method for class 'psPosterior'
posteriorProbs(object, n = NULL, ...)
```

Arguments

object	A Bayesian ‘psFit‘ object or a ‘psPosterior‘ object.
...	Additional arguments passed to methods.
n	‘NULL‘, a single number of leading probabilities to return, or a vector of P or S indices to return.

Details

The ‘estimate‘ column is the posterior mean of each derived P or S probability. It is generally different from evaluating the probability at posterior mean parameter values. Numerical fits integrate over the weighted joint posterior grid; MCMC fits transform retained draws; importance fits retain importance weights; and Laplace fits transform seeded draws from the unconstrained Gaussian approximation. The reported bounds are equal-tailed credible intervals at the level stored in the ‘psPosterior‘ object.

Value

A data frame containing posterior probability summaries.

Examples

```
data(Psurveys)
roux = Psurveys$roux
if (interactive()) {
  fit = fitZIDist(
    roux,
    nterms = 6,
    method = "bayes",
    bayesOptions = list(posteriorMethod = "numerical")
  )
  posteriorProbs(fit, n = 6)
}
```

---

predict.psFit *S3 predict method for an object of class* psFit

---

Description

S3 predict method for an object of class psFit

Usage

```
## S3 method for class 'psFit'
predict(
  object,
  newdata,
```

```
  type = c("plugIn", "posteriorMean", "bootstrapMean"),
  interval = c("none", "prof", "wald", "credible", "percentile"),
  level = 0.95,
  ...
)
```

Arguments

object	an object of class psFit, usually from fitDist or fitZIDist.
newdata	an optional vector of integers at which to calculate $\Pr(X = x)$.
type	The probability definition. "plugIn" evaluates the probability model at fitted parameter values. "posteriorMean" returns posterior mean probabilities for Bayesian fits. "bootstrapMean" returns bootstrap mean probabilities when a bootstrap distribution has been attached to the fit.
interval	Interval type. Existing "prof" and "wald" intervals are retained for plug-in zeta predictions. "credible" returns stored equal-tailed credible intervals for posterior mean predictions. "percentile" returns stored percentile bootstrap confidence intervals for bootstrap mean predictions.
level	the interval level. For posterior and bootstrap predictions, the requested level must match the level stored in the corresponding uncertainty object. If omitted, the stored level is used.
...	other arguments passed to predict—not used.

Value

either a named vector of probabilities, or a data.frame with columns predicted, lower, and upper and row names showing the requested P or S terms.

Examples

```
data(Psurveys)
roux = Psurveys$roux
fit = fitDist(roux)
predict(fit, interval = "prof")
```

---

print.psBayesianBootstrap

Print a fitPS Bayesian Bootstrap object

Description

Print a fitPS Bayesian Bootstrap object

Usage

```
## S3 method for class 'psBayesianBootstrap'
print(x, nterms = NULL, ...)
```

Arguments

x	An object of class ‘psBayesianBootstrap‘.
nterms	Optional number of leading probability summaries to print. By default, at most the first 10 are shown.
...	Additional arguments passed to ‘print.data.frame()‘.

Value

The Bayesian Bootstrap object, invisibly.

---

print.psBootstrap *Print a fitPS bootstrap object*

Description

Print a fitPS bootstrap object

Usage

```
## S3 method for class 'psBootstrap'
print(x, nterms = NULL, ...)
```

Arguments

x	An object of class ‘psBootstrap‘.
nterms	Optional number of leading probability summaries to print. By default, at most the first 10 are shown.
...	Additional arguments passed to ‘print.data.frame()‘.

Value

The bootstrap object, invisibly.

print.psData *S3 print method for an object of class* psData

Description

S3 print method for an object of class psData

Usage

```
## S3 method for class 'psData'
print(x, ...)
```

Arguments

x	an object of class psData, usually from readData or makePSData
...	other arguments passed to print

Value

No return value, called for side effects

print.psFit *S3 print method for an object of class* psFit

Description

S3 print method for an object of class psFit

Usage

```
## S3 method for class 'psFit'
print(x, nterms = NULL, ...)
```

Arguments

x	An object of class psFit, usually returned by [fit()] or one of the deprecated compatibility fitters.
nterms	Number of probability terms to print. If NULL, print the terms already stored in the fitted object, capped at 10 for posterior and bootstrap summaries.
...	Other arguments passed to delegated print methods.

Value

No return value, called for side effects.

print.psPosterior *Print a fitPS posterior object*

Description

Print a fitPS posterior object

Usage

```
## S3 method for class 'psPosterior'
print(x, nterms = NULL, ...)
```

Arguments

x	An object of class ‘psPosterior‘.
nterms	Optional number of leading probability summaries to print. By default, at most the first 10 are shown.
...	Additional arguments passed to ‘print.data.frame()‘.

Value

The posterior object, invisibly.

probfun *Plug-in probability functions*

Description

Creates a probability function that evaluates P or S terms through the model contract at the fitted parameter values stored in a ‘psFit‘ object.

Usage

```
probfun(psFitobj)
```

Arguments

psFitobj	An object of class ‘psFit‘.

Value

A function that can calculate plug-in P or S terms.

Examples

```
p = readData(system.file("extdata", "p.xlsx", package = "fitPS"))
fit = fitDist(p)
P = probfun(fit)
P(0:5)
```

---

psModel *Construct a public fitPS model descriptor*

Description

Creates a model descriptor that can be defined outside fitPS and passed to [fit()]. External model classes supply model-specific mathematics through the exported S3 generics documented here. For simple maximum-likelihood models, fitPS can own the numerical optimisation when starting values and parameter bounds are supplied.

Usage

```
psModel(
  model,
  parameterNames,
  subclass,
  supportedEngines = character(),
  mleStart = NULL,
  mleLower = NULL,
  mleUpper = NULL
)
```

Arguments

model	Non-empty model identifier stored in fitted objects.
parameterNames	Unique natural-scale parameter names.
subclass	Concrete S3 subclass for external method dispatch.
supportedEngines	Character vector of supported Bayesian posterior engines. MLE-only models may use ‘character()‘.
mleStart	Optional named numeric starting values for generic MLE.
mleLower	Optional named numeric lower bounds. Missing bounds default to ‘-Inf‘.
mleUpper	Optional named numeric upper bounds. Missing bounds default to ‘Inf‘.

Value

An object inheriting from ‘psModel‘ and ‘subclass‘.

Examples

```
model = psModel(
  model = "example",
  parameterNames = "theta",
  subclass = "exampleModel",
  mleStart = c(theta = 1),
  mleLower = c(theta = 0)
)
```

`Psurveys` *Number of Groups of Glass Data*

Description

Count data from six different surveys looking at the number of sources/groups of glass found on the upper surfaces of clothing taken from the general public.

Usage

```
data(Psurveys)
```

Format

A `list` with nine objects of class `psData`—see `readData` for more details. The elements of the list are named: `coulson`, `jackson`, `lau`, `lewis.all`, `lewis.clothing`, `lewis.shoes`, `pettard`, `ross`, and `roux`, corresponding to the lead author in each of the references given below. `lau`, `pettard`, and `ross` were taken from Coulson et al. (2001) rather than the original source. The three objects starting with `lewis` represent the combined data (`all`), the groups of glass found on the outer clothing (`clothing`), and the groups of glass found on shoes/footwear (`shoes`).

Source

Coulson, S. A., Buckleton, J. S., Gummer, A. B., and Triggs, C. M. (2001) doi:10.1016/S1355-0306(01)718473 Glass on clothing and shoes of members of the general population and people suspected of breaking crimes, Science & Justice, 41(1):39–48.

References

Lau L, Beveridge AD, Callowhill BC, Conners N, Foster K, Groves RJ, Ohashi KN, Sumner AM, Wong H (1997). "The Frequency of Occurrence of Paint and Glass on the Clothing of High School Students." *Canadian Society of Forensic Science Journal*, **30**(4), 233–240. doi:10.1080/00085030.1997.10757103.

Lewis AD, Alexander LC, Ovide O, Duffett O, Curran JM, Buzzini P, Trejos T (2023). "A study on the occurrence of glass and paint across various cities in the United States—Part I: Background presence of glass in the general population." *Forensic Chemistry*, **34**, 100497. doi:10.1016/j.forc.2023.100497.

Petterd CI, McCallum I, Bradford L, Brinch K, Stewart S (1998). "Glass particles in the clothing of the general population in Canberra—a survey." In *Proceedings of the 14th International Symposium on the Forensic Sciences*.

Ross P, Nguyen H (1998). "A survey of clothing for the presence of glass fragments." In *Proceedings of the 14th International Symposium on the Forensic Sciences*.

Roux C, Kirk R, Benson S, Van Haren T, Petterd CI (2001). "Glass particles in footwear of members of the public in south-eastern Australia—a survey." *Forensic Science International*, **116**(2), 149–156. doi:10.1016/S03790738(00)003558.

Jackson F, Maynard P, Cavanagh-Steer K, Dusting T, Roux C (2013). "A survey of glass found on the headwear and head hair of a random population vs. people working with glass." *Forensic Science International*, **226**(1), 125–131. doi:10.1016/j.forsciint.2012.12.017.

`readData` *Read count data from file*

Description

Reads observed counts of either the number of groups or the size of the groups. The file must have only two columns. One of the columns must be labelled P or S and the other count. It does not matter if the column names are in upper case or not. The P column can have labels 0, 1, 2, ... representing the observation of 0, 1, 2, or more groups. The corresponding count column should contain a positive (non-zero) count for each number of groups. Similarly, if the file contains S counts, then the S column can contain labels 1, 2, ... representing the observation of 1, 2, ... fragments in a group. Note that zeros are neither allowed, or useful, in the file as they both simply result in log-likelihood terms of zero, and therefore make no difference.

Usage

```
readData(fileName, notes = NULL, ...)
```

Arguments

`fileName`	the name of the file to be read. Must be either a modern (xlsx) Excel file or a csv file.
`notes`	any additional information about the data, such as the source or a reference.
`...`	any additional parameters which will be passed to either `read_excel` or `read.csv` depending on the extension of your input file.

Value

an object of class `psData` which is a list containing member variables:

`type` – either `"P"` or `"S"`

`data` – a `data.frame` which contains columns `n` and `rn`, representing the number of groups/fragments, and the number of times that was seen, respectively.

`notes` — either a `bibentry` or a character string which allows extra information about the data to be stored, such as the source, or reference.

Examples

```
p = readData(system.file("extdata", "p.xlsx", package = "fitPS"))
p
s = readData(system.file("extdata", "s.xlsx", package = "fitPS"))
s
```

`rzeta` *Generate random variates from a zeta distribution*

Description

Generate random variates from a zeta distribution

Usage

```
rzeta(n, shape)
```

Arguments

`n`	Same as `Poisson`.
`shape`	The standard zeta shape parameter, greater than 1. See `rzeta`.

`rZIzeta` *Generate zero inflated zeta random variates*

Description

Generate zero inflated zeta random variates

Usage

```
rZIzeta(n, pi = 0.5, shape = 2, offset = 0)

rzizeta(n, pi = 0.5, shape = 2, offset = 0)

rzizeta(n, pi = 0.5, shape = 2, offset = 0)
```

Arguments

`n`	the number of observations.
`pi`	the mixing parameter for the zero-inflated zeta model—must be in (0, 1).
`shape`	the shape parameter for the zero-inflated zeta. Must be greater than 1.
`offset`	the zeta distribution returns random variates that are greater than, or equal to one. If the offset is greater than 0, then the distribution is anchored on (has minimum value of) `1 - offset`.

Details

Technically this function returns values from the one-inflated zeta distribution. However, if `offset` is greater than zero (and typically we expect it to be 1), then the minimium random variate value is 1 - `offset`. We chose the name "zero-inflated zeta" as more people are familiar with zero-inflated models.

Value

a vector of random variates from a zero-inflated zeta model

Examples

```
data(Psurveys)
roux = Psurveys$roux
fit.zi = fitZIDist(roux)
x = rZIzeta(n = sum(roux$data$rn), pi = fit.zi$pi, shape = fit.zi$shape)
table(x)
```

Ssurveys *Size of Groups of Glass Data*

Description

Count data from six different surveys looking at the number of sources/groups of glass found on the upper surfaces of clothing taken from the general public.

Usage

```
data(Psurveys)
```

Format

A `list` with five objects of class `psData`—see `readData` for more details. The elements of the list are named: `jackson`, `lau`, `pettard`, `ross`, and `roux`, corresponding to the lead author in each of the references given below. `lau`, `pettard`, and `ross` were taken from Coulson et al. (2001) rather than the original source.

Source

Coulson, S. A., Buckleton, J. S., Gummer, A. B., and Triggs, C. M. (2001) doi:10.1016/S1355-0306(01)718473 Glass on clothing and shoes of members of the general population and people suspected of breaking crimes, Science & Justice, 41(1):39–48.

References

Lau L, Beveridge AD, Callowhill BC, Conners N, Foster K, Groves RJ, Ohashi KN, Sumner AM, Wong H (1997). "The Frequency of Occurrence of Paint and Glass on the Clothing of High School Students." *Canadian Society of Forensic Science Journal*, **30**(4), 233–240. doi:10.1080/00085030.1997.10757103.

Petterd CI, McCallum I, Bradford L, Brinch K, Stewart S (1998). "Glass particles in the clothing of the general population in Canberra—a survey." In *Proceedings of the 14th International Symposium on the Forensic Sciences*.

Ross P, Nguyen H (1998). "A survey of clothing for the presence of glass fragments." In *Proceedings of the 14th International Symposium on the Forensic Sciences*.

Coulson SA, Buckleton JS, Gummer AB, Triggs CM (2001). "Glass on clothing and shoes of members of the general population and people suspected of breaking crimes." *Science & Justice*, **41**(1), 39–48. doi:10.1016/S13550306(01)718473.

Roux C, Kirk R, Benson S, Van Haren T, Petterd CI (2001). "Glass particles in footwear of members of the public in south-eastern Australia—a survey." *Forensic Science International*, **116**(2), 149–156. doi:10.1016/S03790738(00)003558.

Jackson F, Maynard P, Cavanagh-Steer K, Dusting T, Roux C (2013). "A survey of glass found on the headwear and head hair of a random population vs. people working with glass." *Forensic Science International*, **226**(1), 125–131. doi:10.1016/j.forsciint.2012.12.017.

`summary.psBayesianBootstrap` *Summarise a fitPS Bayesian Bootstrap object*

Description

Summarise a fitPS Bayesian Bootstrap object

Usage

```
## S3 method for class 'psBayesianBootstrap'
summary(object, nterms = NULL, ...)

## S3 method for class 'summary.psBayesianBootstrap'
print(x, ...)
```

Arguments

`object`	An object of class 'psBayesianBootstrap'.
`nterms`	Optional number of leading probability summaries to include. If 'NULL', all stored summaries are included.
`...`	Additional arguments retained for S3 compatibility.
`x`	An object of class 'summary.psBayesianBootstrap'.

Value

An object of class ‘summary.psBayesianBootstrap‘.

Methods (by generic)

- `print(summary.psBayesianBootstrap)`: Print a summarized fitPS Bayesian Bootstrap object.

---

summary.psBootstrap *Summarise a fitPS bootstrap object*

Description

Summarise a fitPS bootstrap object

Usage

```
## S3 method for class 'psBootstrap'
summary(object, nterms = NULL, ...)

## S3 method for class 'summary.psBootstrap'
print(x, ...)
```

Arguments

`object`	An object of class ‘psBootstrap‘.
`nterms`	Optional number of leading probability summaries to include. If ‘NULL‘, all stored summaries are included.
`...`	Additional arguments retained for S3 compatibility.
`x`	An object of class ‘summary.psBootstrap‘.

Value

An object of class ‘summary.psBootstrap‘.

Methods (by generic)

- `print(summary.psBootstrap)`: Print a summarized fitPS bootstrap object.

`summary.psFit` *S3 summary method for an object of class* `psFit`

Description

S3 summary method for an object of class `psFit`

Usage

```
## S3 method for class 'psFit'
summary(object, nterms = NULL, ...)
```

Arguments

`object`	An object of class `psFit`, usually returned by [fit()] or one of the deprecated compatibility fitters.
`nterms`	Optional number of leading posterior or bootstrap probability summaries to include when an uncertainty object is attached.
`...`	Other arguments passed to delegated summary methods.

Details

Bayesian fits delegate to the attached `psPosterior` object. For maximum-likelihood fits, built-in zeta, ZIZ, and logarithmic presentation is preserved. Other models use the public model contract to report their fitted natural-scale parameters and, when available, standard errors obtained from the fitted covariance matrix. If a `psBootstrap` object is attached, its parameter and probability summaries are printed afterwards.

Value

For Bayesian fits, a `summary.psPosterior` object. For maximum-likelihood fits, invisibly returns the coefficient matrix.

`summary.psPosterior` *Summarise a fitPS posterior object*

Description

Summarise a fitPS posterior object

Usage

```
## S3 method for class 'psPosterior'
summary(object, nterms = NULL, inflationEpsilon = 0.01, ...)

## S3 method for class 'summary.psPosterior'
print(x, ...)
```

Arguments

`object`	An object of class ‘psPosterior‘.
`nterms`	Optional number of leading probability summaries to include. If ‘NULL‘, all stored summaries are included.
`inflationEpsilon`	Practical threshold for the inflation parameter. For zero-inflated posteriors, the summary reports ‘Pr(pi < inflationEpsilon | data)‘. The default is 0.01.
`...`	Additional arguments retained for S3 compatibility.
`x`	An object of class ‘summary.psPosterior‘.

Value

An object of class ‘summary.psPosterior‘.

Methods (by generic)

- `print(summary.psPosterior)`: Print a summarized fitPS posterior object.

`supportedPosteriorEngines` *Return posterior engines supported by a fitPS model.*

Description

Return posterior engines supported by a fitPS model.

Usage

```
supportedPosteriorEngines(model, ...)
```

Arguments

`model`	A ‘psModel‘ object.
`...`	Additional arguments reserved for model methods.

Value

Character vector of posterior-engine identifiers.

var *Variance generic*

Description

Variance generic

Usage

```
var(x, ...)

## Default S3 method:
var(x, ...)
```

Arguments

x	an object for which we want to compute the sample variance.
...	Any additional arguments to be passed to var.

Methods (by class)

- var(default): Default variance method for non-psData objects.

var.psData *An S3 method for computing the variance of clothing survey for the number of groups or size of groups*

Description

An S3 method for computing the variance of clothing survey for the number of groups or size of groups

Usage

```
## S3 method for class 'psData'
var(x, ...)
```

Arguments

x	an object of class psData—readData for more details.
...	other arguments which are passed to sum

Value

the mean of the data. If there are r_i observations of the value n_i then the variance is computed by $\mathrm{E}[X^2] - \mathrm{E}[X]^2$, where $\mathrm{E}[X]$ is computed using

$$\sum_i \frac{r_i \times n_i}{\sum_i r_i}$$

, and $\mathrm{E}[X^2]$ is computed by

$$\sum_i \frac{r_i \times n_i^2}{\sum_i r_i}$$

. We realise that the computational formula, $\mathrm{E}[X^2] - \mathrm{E}[X]^2$, is usually not regarded as computationally stable, but the magnitude of the numbers involved is such that, that this is not likely to cause an issue.

Examples

```
data(Psurveys)
var(Psurveys$roux)
```

Index

∗ **datasets**
    Psurveys, 55
    Ssurveys, 58
==.psData, 3

add, 4
as.data.frame.psData, 4

bibentry, *30*, *56*
bootCI, 5
bootstrapFit, 7
bootstrapProbs, 9, *27*

comp.survs *(*compareSurveys*)*, 10
compare.surveys *(*compareSurveys*)*, 10
compareSurveys, 10
compareSurveysLRT, 12
confint.psFit, 13
createPSData *(*makePSData*)*, 30
credInt *(*credint*)*, 14
credint, 14

deviance.psFit, 16
DIC, 16

fit, 17, *44*
fitDist, *10*, 19, *24*, *41*, *50*
fitdist, *29*
fitdist *(*fitDist*)*, 19
fitLogdist *(*fitlogDist*)*, 22
fitlogDist, 22
fitlogdist *(*fitlogDist*)*, 22
fitted.psFit, 23
fitted.psPosterior, 24
fitZIDist, *24*, 25, *41*, *50*
fitZIdist *(*fitZIDist*)*, 25
fitzidist *(*fitZIDist*)*, 25

lines, *13*
logarithmicModel, 28
logLik.psFit, 28

makeData *(*makePSData*)*, 30
makePrior, *21*, 29
makePSData, 30, *52*
mean.psData, 31
modelBayesControl, 32
modelFromUnconstrained, 33
modelLogJacobian, 34
modelLogLikelihood, 35
modelLogPrior, 35
modelMleControl, 36
modelObservationData, 37
modelParameterNames, 37
modelProbabilities, 38
modelToUnconstrained, 39

pboptions, *6*
plot, *44*
plot.psBootstrap, 40
plot.psFit, *21*, *27*, 41, *44*
plot.psPosterior, 42
plotPosterior, 43
plotUncertainty, 45
Poisson, *57*
polygon, *13*
posteriorInflation, 47
posteriorProbs, *27*, 48
predict.psFit, 49
print.psBayesianBootstrap, 50
print.psBootstrap, 51
print.psData, 52
print.psFit, *21*, *27*, 52
print.psPosterior, 53
print.summary.psBayesianBootstrap
        *(*summary.psBayesianBootstrap*)*, 59
print.summary.psBootstrap
        *(*summary.psBootstrap*)*, 60
print.summary.psPosterior
        *(*summary.psPosterior*)*, 61
probfun, *21*, *27*, 53

psModel, 54
Psurveys, 55

readData, *3–6*, *10*, *12*, *19*, *20*, *26*, *30*, *31*, *52*, *55*, 56, *58*, *63*
rzeta, *57*, 57
rZIzeta, 57
rzizeta *(*rZIzeta*)*, 57

Ssurveys, 58
sum, *31*, *63*
summary.psBayesianBootstrap, 59
summary.psBootstrap, 60
summary.psFit, 61
summary.psPosterior, 61
supportedPosteriorEngines, 62

var, 63
var.psData, 63
vglm, *4*

zetaModel *(*logarithmicModel*)*, 28
zizModel *(*logarithmicModel*)*, 28