

Package ‘fitPS’

June 10, 2026

Type Package

Title Fit Zeta Distributions to Forensic Data

Version 1.0.6

Description Fits Zeta distributions (discrete power laws) to data that arises from forensic surveys of clothing on the presence of glass and paint in various populations. The general method is described to some extent in Coulson, S.A., Buckleton, J.S., Gummer, A.B., and Triggs, C.M. (2001) <[doi:10.1016/S1355-0306\(01\)71847-3](https://doi.org/10.1016/S1355-0306(01)71847-3)>, although the implementation differs.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

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

RdMacros Rdpack

RoxygenNote 7.3.3

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

NeedsCompilation no

Author James Curran [aut, cre]

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

Repository CRAN

Date/Publication 2026-06-10 21:40:02 UTC

Contents

==.psData	2
add	3
as.data.frame.psData	4
bootCI	5
compareSurveys	7
compareSurveysLRT	9
confint.psFit	10
credint	11
fitDist	13
fitted.psFit	16
fitZIDist	16
logLik.psFit	19
makePrior	20
makePSData	21
mean.psData	22
plot.psFit	23
plotPosterior	24
predict.psFit	25
print.psData	26
print.psFit	27
probfun	27
Psurveys	28
readData	29
rzeta	30
rZizeta	30
Ssurveys	31
summary.psFit	32
var	33
var.psData	33
Index	35

==.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 matrix 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$1au, 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 [readData](#) for 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	<i>Bootstrap confidence intervals or regions</i>
--------	--

Description

Use bootstrapping to generate confidence intervals, or confidence regions in the case of the zero-inflated model.

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 <code>psData</code> —see readData for more details—or of class <code>psFit</code> .
...	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 <code>data.frame</code>) of bootstrapped values is returned. This can be useful for debugging or understanding the results. Default is FALSE.

<code>silent</code>	if TRUE, then no output will be displayed whilst the bootstrapping is being undertaken. <code>plot</code> if TRUE then the contours for the confidence region will be plotted. This only works if <code>model = "ziz"</code> . It is ignored otherwise. <code>parallel</code> if TRUE then the bootstrapping is performed in parallel.
<code>plot</code>	if TRUE and <code>model == "ziz"</code> , then a plot of the bootstrapped values will be produced and confidence contour lines will be drawn for each value in <code>level</code> .
<code>parallel</code>	if TRUE, then the package will attempt to use multiple cores to speed up computation.
<code>progressBar</code>	if TRUE, then progress bars will be displayed to show progress on the bootstrapping.
<code>pbopts</code>	a list of arguments for the <code>pboptions</code> function that affect the progress bars. Ignored if <code>progressBar = FALSE</code> .

Details

This 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.

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

```
## Not run:
data(Psurveys)
roux = Psurveys$roux
confRegion = bootCI(roux, model = "ziz", parallel = FALSE, plot = TRUE)

## This will not work unless you have the sp package installed
## Count how many of the points lie within the 95% confidence region
lapply(confRegion, function(cr){
  table(sp::point.in.polygon(fit$pi,fit$shape, cr$pi, cr$shape))
. })

## End(Not run)
```

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:

<code>statistic</code>	– the test statistic.
<code>p.value</code>	– the P-value associated with the estimate.
<code>estimate</code>	– the estimated difference in the shape parameters.
<code>null.value</code>	– the specified hypothesized value of the difference in shape parameters—0 by default.
<code>stderr</code>	– the standard error of the difference.
<code>alternative</code>	– a character string describing the alternative hypothesis.
<code>method</code>	– a character string describing the method.
<code>data.name</code>	– 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)
```

<code>compareSurveysLRT</code>	<i>Compare two or more surveys on the basis of their shape parameters using a Likelihood Ratio Test</i>
--------------------------------	---

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

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

versus the alternative

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

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—see fitDist for more details
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 p1ZIZ.

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 `fitDist`), 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 $\text{shape} > 1$. If a zero-inflated zeta model is used (i.e. object comes from a call to `fitZIDist`) then list of a confidence regions is returned with an 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
fit = fitDist(roux)
confint(fit)

## Not run:
fit.zi = fitZIDist(roux)
cr = confint(fit.zi, level = c(0.80, 0.95))
plot(cr[["0.95"]], type = "l")
polygon(cr[["0.8"]])

## End(Not run)

## Not run:
fit.zi = fitZIDist(roux, method = "bayes")
cr = confint(fit.zi, level = c(0.80, 0.95))
plot(cr[["0.95"]], type = "l")
polygon(cr[["0.8"]])

## End(Not run)
```

credint

Bayesian credible intervals or regions

Description

Use kernel density estimation to generate credible intervals, or credible regions in the case of the zero-inflated model.

Usage

```
credint(psFit, level = 0.95, plot = FALSE, silent = FALSE, ...)

credInt(psFit, level = 0.95, plot = FALSE, silent = FALSE, ...)
```

Arguments

<code>psFit</code>	a object of class <code>psFit</code> .
<code>level</code>	the credible level required—restricted to $[0.75, 1)$. This may be a vector, in which case multiple intervals, or credible regions will be returned.
<code>plot</code>	if TRUE and <code>model == "ziz"</code> , then a plot of the bootstrapped values will be produced and confidence contour lines will be drawn for each value in <code>level</code> .
<code>silent</code>	if TRUE, then no output will be displayed whilst the the kernel density estimation is being undertaken.
<code>...</code>	other arguments fed to <code>plot</code> . If <code>plot == FALSE</code> , then these will be ignored

Details

This function uses kernel density estimation to compute a Bayesian credible interval for the shape parameter in the case of the zeta model and a credible region in the case of the zero-inflated zeta model. A smoothing 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.

Value

If `psData$model == "zeta"`, then either a vector or a `data.frame` with elements/columns named "lower" and "upper" representing the lower and upper bounds of the confidence interval(s). Multiple bounds are returned in a `data.frame` when `level` has more than one value. If `psData$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.

Functions

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

Examples

```
## Not run:
data(Psurveys)
roux = Psurveys$roux
fit = fitzidist(roux, method == "bayes")
credRegion = credint(roux, plot = TRUE)

## This will not work unless you have the sp package installed
```

```
## Count how many of the points lie within the 95% confidence region
lapply(credRegion, function(cr){
  table(sp::point.in.polygon(fit$pi,fit$shape, cr$pi, cr$shape))
. })

## End(Not run)
```

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(shape - 1). The default is -2. b – The upper bound for the default uniform prior on log(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 which is a list containing seven or eight elements:

psData	– an object of class psData—see readData ,
fit	– the fitted object from optim ,
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".

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 $\text{shape} > 1$.

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.

Currently the Bayesian estimation is done using the prior returned by `makePrior`. By default this is a Uniform[a, b] prior on $\log(\text{shape} - 1)$, so the prior support always has $\text{shape} > 1$. This may become more flexible in the future. Similarly, the estimation is done using a simple Metropolis-Hastings sampler. It might be more efficient to sample through adaptive rejection sampling, but it is unclear whether it is worth the effort.

Value

an object of class psFit—see Details.

Functions

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

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.

Evetts, 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

```

 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, ...)

```

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 return, or, a vector containing the desired fitted values.
...	other arguments passed to fitted—not used.

Value

a named vector of fitted probabilities

 fitZIDist

Fit a Zero-Inflated Zeta Distribution to Forensic Data

Description

This function uses maximum likelihood estimation (MLE) or Bayesian estimation (MCMC) 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

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

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

<code>x</code>	an object of type <code>psData</code> , usually obtained from readData .
<code>nterms</code>	the number of terms to compute the probability distribution for.
<code>method</code>	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 <code>method = "bayes"</code> with the corresponding <code>bayesOptions\$posteriorMethod</code> .
<code>prior</code>	optional prior object used by Bayesian posterior approximation methods where applicable. This is retained for consistency with <code>fitDist()</code> ; new code should usually pass priors through <code>bayesOptions</code> .
<code>bayesOptions</code>	optional list controlling Bayesian fitting. The <code>posteriorMethod</code> element selects "numerical", "mcmc", "laplace", or "importance". The default is "numerical". The <code>prior</code> element may contain a prior object returned by <code>makePrior()</code> .
<code>...</code>	other arguments that control the estimation methods. If <code>method == "mle"</code> , then the user can provide an optional argument <code>start</code> which is the starting value for the numerical optimisation. If this is not provided, then <code>start = c(0.5, 2)</code> by default. If you specify your own starting value, keep the mixing parameter greater than 0.5 and use <code>shape > 1</code> .

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

- `theta0` – The initial values of the mixing parameter and shape parameter. The default is `c(0.5, 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`.
- `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` which is a list containing eight or nine elements:

- `psData` – an object of class `psData`—see [readData](#),
- `fit` – the fitted object from [optim](#),
- `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` – the estimated (posterior) 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"`,
- `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 mixing parameter and 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 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$posteriorMethod`. The default, "numerical", uses deterministic two-dimensional grid integration over π and shape. The legacy Metropolis-Hastings sampler remains available with `bayesOptions = list(posteriorMethod = "mcmc")`. The prior for the mixing proportion is `Beta(shape1, shape2)`, and the shape prior is supplied by `bayesOptions$prior` or `prior`. If no shape prior is supplied, `makePrior()` is used.

Value

an object of class `psFit`—see Details.

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](#).

Examples

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

logLik.psFit

Extract the log-likelihood from a zeta model fit

Description

Returns the maximised log-likelihood for a fitted zeta model.

Usage

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

Arguments

`object` An object of class "psFit".
`...` Additional arguments passed to methods. Currently ignored.

Details

This method allows generic functions such as `[stats::AIC()]` and `[stats::BIC()]` to work with objects of class "psFit".

Value

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

makePrior	<i>Define a prior density</i>
-----------	-------------------------------

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. For zeta models this range is on the standard shape scale, where shape > 1.
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 $n = \text{rep}(0:1, 98, 1)$ or $n = 0:1$, $\text{count} = \text{c}(98, 1)$. 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)
```

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

<code>x</code>	an object of class psFit, usually from <code>fitDist</code> or <code>fitZIDist</code> .
<code>ylim</code>	the limits of the y-axis.
<code>conf</code>	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.
<code>conf.level</code>	the confidence level for the confidence intervals. Must be between 0.75 and 0.99.
<code>ci.type</code>	Specifies the type of confidence interval. If <code>conf == TRUE</code> , then then <code>ci.type</code> 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.
<code>log.scale</code>	if TRUE the <i>y</i> -axis is changed to a logarithmic (base 10) axis.
<code>...</code>	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")
```

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 from fitDist or fitZIDist .
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 `object$chain`. For numerical integration fits from `fitDist(..., method = "integrate")`, the stored posterior density function in `object$pdf` is evaluated on an automatically chosen grid.

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

```
## Not run:
data(Psurveys)
roux = Psurveys$roux
fit = fitDist(roux, method = "bayes")
plotPosterior(fit)

## End(Not run)
```

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,
  interval = c("none", "prof", "wald"),
  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 $\Pr(X = x)$.

interval	either "none", "prof", or "wald" and can be abbreviated. If "prof" or "wald" AND the zeta model has been used, then an interval, based on the bounds of a 100 * level confidence interval for the shape parameter, is given for each predicted probability. The interval is provided based on either a profile likelihood or a Wald confidence interval for shape, and therefore cannot really be regarded as a confidence interval for the probabilities. The intervals might be more sensibly regarded as a measure of how sensitive the probabilities are to the choice of shape parameter. NOTE: this parameter is ignored if the Zero-inflated (ZIZ) model has been used.
level	the level of a confidence interval. Ignored if interval == "none".
...	other arguments passed to predict—not used

Value

either a named vector of fitted probabilities, or a data.frame with columns predicted, lower, and upper and the row names set to show what terms are being calculated

Examples

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

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	<i>S3 print method for an object of class psFit</i>
-------------	---

Description

S3 print method for an object of class psFit

Usage

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

Arguments

x	an object of class psFit, usually from fitDist or #' fitZIDist .
...	other arguments passed to print.

Value

No return value, called for side effects.

probfun	<i>Probability Functions</i>
---------	------------------------------

Description

Creates a probability function that allows the computation of any P or S term.

Usage

```
probfun(psFitobj)
```

Arguments

psFitobj	an object of class psFit—see fitDist and fitZIDist .
----------	--

Value

a function that can be used to calculate any P or S term.

Examples

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

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](https://doi.org/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, Connors 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](https://doi.org/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](https://doi.org/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](https://doi.org/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	<i>Generate random variates from a zeta distribution</i>
-------	--

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	<i>Generate zero inflated zeta random variates</i>
---------	--

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 minimum random variate value is $1 - \text{offset}$. 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](https://doi.org/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.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, ...)
```

Arguments

object	an object of class psFit, usually from <code>fitDist</code> or <code>fitZIDist</code>
...	other arguments passed to summary

Details

Experimental because I am unsure if it is useful. If object is a zero-inflated zeta fitted object, then the function carries out a likelihood ratio test for the value of pi. Currently not implemented for the logarithmic distribution because we are currently not interested in the logarithmic distribution.

Value

No return value, called for side effects

var	<i>Variance generic</i>
-----	-------------------------

Description

Variance generic

Usage

```
var(x, ...)
```

Arguments

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

var.psData	<i>An S3 method for computing the variance of clothing survey for the number of groups or size of groups</i>
------------	--

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 $E[X^2] - E[X]^2$, where $E[X]$ is computed using

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

, and $E[X^2]$ is computed by

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

. We realise that the computational formula, $E[X^2] - E[X]^2$, 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, [28](#)
 - Ssurveys, [31](#)
- ==.psData, [2](#)
- add, [3](#)
- as.data.frame.psData, [4](#)

- bibentry, [21, 29](#)
- bootCI, [5](#)

- comp.survs (compareSurveys), [7](#)
- compare.surveys (compareSurveys), [7](#)
- compareSurveys, [7](#)
- compareSurveysLRT, [9](#)
- confint.psFit, [10](#)
- createPSData (makePSData), [21](#)
- credInt (credint), [11](#)
- credint, [11](#)

- fitDist, [8, 11, 13, 16, 23–25, 27, 32](#)
- fitdist, [20](#)
- fitdist (fitDist), [13](#)
- fitted.psFit, [16](#)
- fitZIDist, [11, 16, 16, 23–25, 27, 32](#)
- fitZIdist (fitZIDist), [16](#)
- fitzidist (fitZIDist), [16](#)

- lines, [11](#)
- logLik.psFit, [19](#)

- makeData (makePSData), [21](#)
- makePrior, [15, 20](#)
- makePSData, [21, 26](#)
- mean.psData, [22](#)

- optim, [14, 18](#)

- pboptions, [6](#)
- plot, [24](#)
- plot.psFit, [15, 18, 19, 23, 25](#)

- plotPosterior, [24](#)
- Poisson, [30](#)
- polygon, [11](#)
- predict.psFit, [25](#)
- print.psData, [26](#)
- print.psFit, [15, 18, 19, 27](#)
- probfun, [15, 18, 19, 27](#)
- Psurveys, [28](#)

- readData, [2–5, 8, 9, 14, 17, 18, 21, 22, 26, 28, 29, 31, 33](#)
- rzeta, [30, 30](#)
- rZIZeta, [30](#)
- rzizeta (rZIZeta), [30](#)

- Ssurveys, [31](#)
- sum, [22, 33](#)
- summary.psFit, [32](#)

- var, [33](#)
- var.psData, [33](#)
- vglm, [4](#)