



**University of
Zurich**^{UZH}

**Zurich Open Repository and
Archive**

University of Zurich
Main Library
Strickhofstrasse 39
CH-8057 Zurich
www.zora.uzh.ch

Year: 2013

Package 'prevalence'

Devleesschauwer, Brecht ; Torgerson, Paul R ; Charlier, Johannes ; Levecke, Bruno ; Praet, Nicolas ; Dorny, Pierre ; Berkvens, Dirk ; Speybroeck, Niko

Abstract: Tools for prevalence assessment studies. IMPORTANT: the truePrev functions in the prevalence package call on JAGS (Just Another Gibbs Sampler), which therefore has to be available on the user's system. JAGS can be downloaded from <http://mcmc-jags.sourceforge.net/>

Posted at the Zurich Open Repository and Archive, University of Zurich
ZORA URL: <https://doi.org/10.5167/uzh-89061>
Scientific Publication in Electronic Form
Published Version



The following work is licensed under a Software: GNU General Public License, version 2.0 (GPL-2.0).

Originally published at:

Devleesschauwer, Brecht; Torgerson, Paul R; Charlier, Johannes; Levecke, Bruno; Praet, Nicolas; Dorny, Pierre; Berkvens, Dirk; Speybroeck, Niko (2013). Package 'prevalence'. On Line: The Comprehensive R Archive Network.

Package ‘prevalence’

September 22, 2013

Type Package

Title The prevalence package

Version 0.2.0

Date 2013-09-22

Author Brecht Devleesschauwer [aut, cre], Paul Torgerson [aut], Johannes Charlier [aut], Bruno Levecke [aut], Nicolas Praet [aut], Pierre Dorny [aut], Dirk Berkvens [aut], Niko Speybroeck [aut]

Maintainer Brecht Devleesschauwer <Brecht.Devleesschauwer@UGent.be>

BugReports <https://github.com/brechtdev/prevalence/issues>

Description

Tools for prevalence assessment studies. IMPORTANT: the truePrev functions in the prevalence package call on JAGS (Just Another Gibbs Sampler), which therefore has to be available on the user's system. JAGS can be downloaded from <http://mcmc-jags.sourceforge.net/>

Depends R (>= 3.0.0), rjags, coda, methods

SystemRequirements jags (>= 3.2.0)

License GPL (>= 2)

URL <http://users.ugent.be/~bdvleess/R/prevalence/>

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2013-09-22 22:50:23

R topics documented:

prevalence-package	2
betaExpert	3
betaPERT	5
definition	7
plot-methods	9
plot-methods-coda	10
prev-class	10
print-methods	12
propCI	13
show-methods	15
summary-methods	15
truePrev	16
truePrevMulti	18
truePrevPools	24
Index	27

prevalence-package *Tools for prevalence assessment studies*

Description

The **prevalence** package provides Frequentist and Bayesian methods useful in prevalence assessment studies.

Details

Package: prevalence
 Type: Package
 Version: 0.2.0
 Date: 2013-09-22
 License: GNU >= 2

Available functions in the **prevalence** package:

<code>truePrev</code>	Estimate TP from AP obtained by testing individual samples with a single test
<code>truePrevMulti</code>	Estimate TP from AP obtained by testing individual samples with multiple tests
<code>truePrevPools</code>	Estimate TP from AP obtained by testing pooled samples
<code>propCI</code>	Derive confidence intervals for a prevalence estimate
<code>betaPERT</code>	Calculate the parameters of a Beta-PERT distribution
<code>betaExpert</code>	Calculate the parameters of a Beta distribution based on expert opinion

IMPORTANT: the `truePrev` functions in the **prevalence** package call on **JAGS (Just Another Gibbs Sampler)**, through the **rjags** package. Therefore, JAGS has to be installed on the user's

system.

JAGS can be downloaded from <http://sourceforge.net/projects/mcmc-jags/>

Author(s)

Creator, Maintainer

<Brecht.Devleeschauwer@UGent.be>

Contributors

Paul Torgerson, Johannes Charlier, Bruno Levecke, Nicolas Praet, Pierre Dorny, Dirk Berkvens, Niko Speybroeck

betaExpert	<i>Calculate the parameters of a Beta distribution based on expert information</i>
------------	------------------------------------------------------------------------------------

Description

The `betaExpert` function fits a (standard) Beta distribution to expert opinion. The expert provides information on a best-guess estimate (mode or mean), and an uncertainty range:

- The parameter value is with $100 \cdot p\%$ certainty greater than lower
- The parameter value is with $100 \cdot p\%$ certainty smaller than upper
- The parameter value lies with $100 \cdot p\%$ in between lower and upper

Usage

```
betaExpert(best, lower, upper, p = 0.95, method = "mode")
```

```
## S3 method for class 'betaExpert'
print(x, conf.level = .95, ...)
```

Arguments

<code>best</code>	Best-guess estimate; see argument method
<code>lower</code>	Lower uncertainty limit
<code>upper</code>	Upper uncertainty limit
<code>p</code>	Expert's certainty level
<code>method</code>	Does best-guess estimate correspond to the mode or to the mean? Defaults to mode
<code>x</code>	Object of class <code>betaExpert</code>
<code>conf.level</code>	Confidence level used in printing quantiles of resulting Beta distribution
<code>...</code>	Other arguments to pass to function <code>print</code>

Details

The methodology behind the `betaExpert` function is presented by Branscum et al. (2005) and implemented in the *BetaBuster* software.

The parameters of a standard Beta distribution are calculated based on a best-guess estimate and a $100(p)\%$ uncertainty range, defined by a lower and/or upper limit. The `betaExpert` function uses minimization (`optimize`) to derive α and β from this best guess and lower and/or upper limit. The resulting distribution is a standard 2-parameter Beta distribution: $\text{Beta}(\alpha, \beta)$.

Value

A list of class "betaExpert":

alpha	Parameter α (shape1) of the Beta distribution
beta	Parameter β (shape2) of the Beta distribution

The print method for "betaExpert" additionally calculates the mean, median, mode, variance and range of the corresponding Beta distribution.

Author(s)

<Brecht.Devleeschauwer@UGent.be>

References

Branscum AJ, Gardner IA, Johnson WO (2005) Estimation of diagnostic-test sensitivity and specificity through Bayesian modeling. *Prev Vet Med* **68**:145-163.
[BetaBuster homepage](#)

See Also

Package **riskDistributions**, which provides a collection of functions for fitting distributions to given data or by known quantiles.

[betaPERT](#), for modelling a generalized Beta distribution based on expert opinion

Examples

```
## Most likely value (mode) is 90%
## Expert states with 95% certainty that true value is larger than 70%
betaExpert(best = 0.90, lower = 0.75, p = 0.95)

## Most likely value (mode) is 0%
## Expert states with 95% certainty that true value is smaller than 40%
betaExpert(best = 0, upper = 0.40, p = 0.95)

## Most likely value (mode) is 80%
## Expert states with 90% certainty that true value lies in between 40% and 90%
betaExpert(best = 0.80, lower = 0.40, upper = 0.90, p = 0.90)
```

```
## Mean value is assumed to be 80%
## Expert states with 90% certainty that true value lies in between 40% and 90%
betaExpert(best = 0.80, lower = 0.40, upper = 0.90, p = 0.90, method = "mean")
```

betaPERT

Calculate the parameters of a Beta-PERT distribution

Description

The Beta-PERT methodology allows to parametrize a generalized Beta distribution based on expert opinion regarding a pessimistic estimate (minimum value), a most likely estimate (mode), and an optimistic estimate (maximum value). The betaPERT function incorporates two methods of calculating the parameters of a Beta-PERT distribution, designated "Classic" and "Vose".

Usage

```
betaPERT(a, m, b, k = 4, method = "classic", p = 0.95, plot = FALSE)
```

Arguments

a	Pessimistic estimate (Minimum value)
m	Most likely estimate (Mode)
b	Optimistic estimate (Maximum value)
k	Scale parameter
method	"Classic" or "Vose"; see details below
p	Confidence level for printing confidence intervals
plot	Logical flag: should resulting distribution be plotted?

Details

The Beta-PERT methodology was developed in the context of Program Evaluation and Review Technique (PERT). Based on a pessimistic estimate (minimum value), a most likely estimate (mode), and an optimistic estimate (maximum value), typically derived through expert elicitation, the parameters of a Beta distribution can be calculated. The Beta-PERT distribution is used in stochastic modeling and risk assessment studies to reflect uncertainty regarding specific parameters.

Different methods exist in literature for defining the parameters of a Beta distribution based on PERT. The two most common methods are included in the betaPERT function:

Classic: The standard formulas for mean, standard deviation, α and β , are as follows:

$$mean = \frac{a + k * m + b}{k + 2}$$

$$sd = \frac{b - a}{k + 2}$$

$$\alpha = \frac{mean - a}{b - a} * \left\{ (mean - a) * \frac{b - mean}{sd^2} - 1 \right\}$$

$$\beta = \alpha * \frac{b - mean}{mean - a}$$

The resulting distribution is a 4-parameter Beta distribution: Beta(α , β , a, b).

Vose: Vose (2000) describes a different formula for α :

$$(mean - a) * \frac{2 * m - a - b}{(m - mean) * (b - a)}$$

Mean and β are calculated using the standard formulas; as for the classical PERT, the resulting distribution is a 4-parameter Beta distribution: Beta(α , β , a, b).

Note: If $m = mean$, α is calculated as $1 + k/2$, in accordance with the **mc2d** package (see 'Note').

Value

A list of class "betaPERT":

alpha	Parameter α (shape1) of the Beta distribution
beta	Parameter β (shape2) of the Beta distribution
a	Pessimistic estimate (Minimum value)
m	Most likely estimate (Mode)
b	Optimistic estimate (Maximum value)
method	Applied method
p	Confidence level for printing confidence intervals

Available generic functions for class "betaPERT" are print and plot.

Note

The **mc2d** package provides the probability density function, cumulative distribution function, quantile function and random number generation function for the PERT distribution, parametrized by the "Vose" method.

Author(s)

<Brecht.Devleeschauwer@UGent.be>

References

Classic: Malcolm DG, Roseboom JH, Clark CE, Fazar W (1959) Application of a technique for research and development program evaluation. *Oper Res* 7(5):646-669.

Vose: David Vose. *Risk analysis, a quantitative guide, 2nd edition*. Wiley and Sons, 2000.
PERT distribution in ModelRisk (Vose software)

See Also

[betaExpert](#), for modelling a standard Beta distribution based on expert opinion

Examples

```
## The value of a parameter of interest is believed to lie between 0 and 50
## The most likely value is believed to be 10

# Classical PERT
betaPERT(a = 0, m = 10, b = 50)

# Vose parametrization
betaPERT(a = 0, m = 10, b = 50, method = "Vose")
```

definition

Definition of truePrevMulti model

Description

These two utility functions can help to specify function [truePrevMulti](#).

Usage

```
definition_x(n)
definition_theta(n)
```

Arguments

n Number of tests

Details

Function [truePrevMulti](#) estimates true prevalence from individual samples tested with n tests, using the approach of Berkvens et al. (2006).

The vector of apparent tests results, x, must contain the number of samples corresponding to each combination of test results. The model assumes that the first value corresponds to the number of samples that tested negative on all tests and that the last value corresponds to the number of samples that tested positive on all tests.

The prior in the multinomial model consists of a vector theta, which holds values for the true prevalence (TP), the sensitivity and specificity of the first test (SE1, SP1), and the conditional dependencies between the results of the subsequent tests and the preceding one(s).

Author(s)

<Brecht.Devleeschauwer@UGent.be>

References

- Berkvens D, Speybroeck N, Praet N, Adel A, Lesaffre E (2006) *Estimating disease prevalence in a Bayesian framework using probabilistic constraints*. *Epidemiology* 17; 145-153

See Also

[truePrevMulti](#)

Examples

```
## how is a 2-test model defined?

definition_x(2)
# Definition of the apparent test results, 'x', for 2 tests:
# x[1] : T1-,T2-
# x[2] : T1-,T2+
# x[3] : T1+,T2-
# x[4] : T1+,T2+

definition_theta(2)
# Definition of the prior, 'theta', for 2 tests:
# theta[1] : P(D+) = TP
# theta[2] : P(T1+|D+) = SE1
# theta[3] : P(T1-|D-) = SP1
# theta[4] : P(T2+|D+,T1+)
# theta[5] : P(T2+|D-,T1-)
# theta[6] : P(T2-|D-,T1-)
# theta[7] : P(T2-|D-,T1+)

## how is a 3-test model defined?

definition_x(3)
# Definition of the apparent test results, 'x', for 3 tests:
# x[1] : T1-,T2-,T3-
# x[2] : T1-,T2-,T3+
# x[3] : T1-,T2+,T3-
# x[4] : T1-,T2+,T3+
# x[5] : T1+,T2-,T3-
# x[6] : T1+,T2-,T3+
# x[7] : T1+,T2+,T3-
# x[8] : T1+,T2+,T3+

definition_theta(3)
# Definition of the prior, 'theta', for 3 tests:
# theta[1] : P(D+) = TP
# theta[2] : P(T1+|D+) = SE1
# theta[3] : P(T1-|D-) = SP1
# theta[4] : P(T2+|D+,T1+)
# theta[5] : P(T2+|D-,T1-)
# theta[6] : P(T2-|D-,T1-)
# theta[7] : P(T2-|D-,T1+)
# theta[8] : P(T3+|D+,T1+,T2+)
```

```
# theta[9] : P(T3+|D+,T1+,T2-)
# theta[10] : P(T3+|D+,T1-,T2+)
# theta[11] : P(T3+|D+,T1-,T2-)
# theta[12] : P(T3-|D-,T1-,T2-)
# theta[13] : P(T3-|D-,T1-,T2+)
# theta[14] : P(T3-|D-,T1+,T2-)
# theta[15] : P(T3-|D-,T1+,T2+)
```

plot-methods

Methods for Function plot in Package prevalence

Description

Plot objects of class prev

Usage

```
## S4 method for signature 'prev,ANY'
plot(x, y = NULL, ...)
```

Arguments

x	An object of class prev
y	Which parameter to plot? defaults to NULL
...	Other arguments to pass to the plot function

Methods

signature(x = "prev", y = "ANY") Show [density](#), [trace](#), [Brooks-Gelman-Rubin](#) and [autocorrelation](#) plots.

See Also

[prev-class](#)
[densplot-methods](#), [traceplot-methods](#), [gelman.plot-methods](#), [autocorr.plot-methods](#)

plot-methods-coda *Plotting functions from package **coda***

Description

Different plotting functions from package **coda** have been made available as method to class `prev`

Usage

```
## S4 method for signature 'prev'
densplot(x, exclude_fixed = TRUE, ...)

## S4 method for signature 'prev'
traceplot(x, exclude_fixed = TRUE, ...)

## S4 method for signature 'prev'
autocorr.plot(x, exclude_fixed = TRUE, chain = 1, ...)
```

Arguments

<code>x</code>	An object of class <code>prev</code>
<code>exclude_fixed</code>	Should fixed parameters be excluded from plotting? defaults to TRUE
<code>chain</code>	Which chain to plot in <code>autocorr.plot</code> ; defaults to 1
<code>...</code>	Other arguments to pass to the specific plot function.

Methods

`signature(x = "prev")` Show [density](#), [trace](#), [Brooks-Gelman-Rubin](#) and [autocorrelation](#) plots.

See Also

[prev-class](#)
[plot-methods](#)
[densplot](#), [traceplot](#), [gelman.plot](#), [autocorr.plot](#)

prev-class *Class "prev"*

Description

The "prev" class represents output from Bayesian true prevalence estimation models.

Objects from the Class

Objects of class "prev" are created by [truePrev](#), [truePrevMulti](#) and [truePrevPools](#).

Slots

Objects of class "prev" contain the following four slots:

par: A list of input parameters

model: The fitted Bayesian model, in BUGS language (class "prevModel")

mcmc: A list, with one element per chain, of the simulated true prevalences (and sensitivities and specificities in the case of [truePrevMulti](#))

diagnostics: A list with elements for the Deviance Information Criterion (\$DIC), the Brooks-Gelman-Rubin statistic (\$BGR), and in the case of [truePrevMulti](#), the Bayes-P statistic (\$bayesP)

Author(s)

<Brecht.Devleeschauwer@UGent.be>

See Also

[truePrev](#), [truePrevMulti](#), [truePrevPools](#)
[show-methods](#), [print-methods](#), [summary-methods](#), [plot-methods](#), [plot-methods-coda](#)

Examples

```
## Taenia solium cysticercosis in Nepal
SE <- list(dist = "uniform", min = 0.60, max = 1.00)
SP <- list(dist = "uniform", min = 0.75, max = 1.00)
TP <- truePrev(x = 142, n = 742, SE = SE, SP = SP)

## Summarize estimates per chain
summary(TP)

## Diagnostic plots
par(mfrow = c(2, 2))
plot(TP)

## Generic plots from package coda
par(mfrow = c(1, 1))
densplot(TP)
traceplot(TP)
gelman.plot(TP)
autocorr.plot(TP)

## Use 'str()' to see the structure of object TP
str(TP)

## Every slot can be accessed using the '@' operator
TP@par      # input parameters
TP@model    # fitted model
TP@mcmc     # simulated TP (and SE, SP)
TP@diagnostics # DIC and BGR (and bayesP)
```

```
## TP@mcmc inherits from class 'mcmc.list' in coda package
## List all available methods for this class
methods(class = "mcmc.list")
## List all available functions in coda package
library(help = "coda")
```

print-methods

Methods for Function print in Package prevalence

Description

Print objects of class `prev`

Usage

```
## S4 method for signature 'prev'
print(x, conf.level, dig = 3, ...)
```

Arguments

<code>x</code>	An object of class <code>prev</code>
<code>conf.level</code>	Confidence level to be used in credibility interval
<code>dig</code>	Number of decimal digits to print
<code>...</code>	Other arguments to pass to the <code>print</code> function

Methods

`signature(x = "prev")` Print mean, median, mode and credibility interval of estimated true prevalence (and sensitivities and specificities, for `prev` objects created by `truePrevMulti`). In addition, print [Brooks-Gelman-Rubin statistic](#) and corresponding upper confidence limit. BGR values substantially above 1 indicate lack of convergence. For `prev` objects created by `truePrevMulti`, the Bayes-P statistic is also printed. Bayes-P should be as close to 0.5 as possible.

See Also

[prev-class](#)
[gelman.diag](#)

propCI *Calculate confidence intervals for prevalences and other proportions*

Description

The propCI function calculates five types of confidence intervals for proportions:

- Wald interval (= Normal approximation interval, asymptotic interval)
- Agresti-Coull interval (= adjusted Wald interval)
- Exact interval (= Clopper-Pearson interval)
- Jeffreys interval (= Bayesian interval)
- Wilson score interval

Usage

```
propCI(x, n, method = "all", level = 0.95, sortby = "level")
```

Arguments

x	Number of successes (positive samples)
n	Number of trials (sample size)
method	Confidence interval calculation method; see details
level	Confidence level for confidence intervals
sortby	Sort results by "level" or "method"

Details

Five methods are available for calculating confidence intervals. For convenience, synonyms are allowed. Please refer to the PDF version of the manual for proper formatting of the below formulas.

"agresti.coull", "agresti-coull", "ac"

$$\tilde{n} = n + z_{1-\frac{\alpha}{2}}^2$$

$$\tilde{p} = \frac{1}{\tilde{n}} \left(x + \frac{1}{2} z_{1-\frac{\alpha}{2}}^2 \right)$$

$$\tilde{p} \pm z_{1-\frac{\alpha}{2}} \sqrt{\frac{\tilde{p}(1-\tilde{p})}{\tilde{n}}}$$

"exact", "clopper-pearson", "cp"

$$\left(\text{Beta}\left(\frac{\alpha}{2}; x, n - x + 1\right), \text{Beta}\left(1 - \frac{\alpha}{2}; x + 1, n - x\right) \right)$$

"jeffreys", "bayes"

$$\left(\text{Beta}\left(\frac{\alpha}{2}; x + 0.5, n - x + 0.5\right), \text{Beta}\left(1 - \frac{\alpha}{2}; x + 0.5, n - x + 0.5\right) \right)$$

"wald", "asymptotic", "normal"

$$p \pm z_{1-\frac{\alpha}{2}} \sqrt{\frac{p(1-p)}{n}}$$

"wilson"

$$\frac{p + \frac{z_{1-\frac{\alpha}{2}}^2}{2n} \pm z_{1-\frac{\alpha}{2}} \sqrt{\frac{p(1-p)}{n} + \frac{z_{1-\frac{\alpha}{2}}^2}{4n^2}}}{1 + \frac{z_{1-\frac{\alpha}{2}}^2}{n}}$$

Value

Data frame with seven columns:

x	Number of successes (positive samples)
n	Number of trials (sample size)
p	Proportion of successes (prevalence)
method	Confidence interval calculation method
level	Confidence level
lower	Lower confidence limit
upper	Upper confidence limit

Note

In case the observed prevalence equals 0% (ie, x == 0), an upper one-sided confidence interval is returned. In case the observed prevalence equals 100% (ie, x == n), a lower one-sided confidence interval is returned. In all other cases, two-sided confidence intervals are returned.

Author(s)

<Brecht.Devleeschauwer@UGent.be>

Examples

```
## All methods, 95% confidence intervals
propCI(x = 142, n = 742)

## Wald-type 90%, 95% and 99% confidence intervals
propCI(x = 142, n = 742, method = "wald", level = c(0.90, 0.95, 0.99))
```

show-methods

Methods for Function show in Package prevalence

Description

Show objects of class prev

Usage

```
## S4 method for signature 'prev'  
show(object)
```

Arguments

object An object of class prev

Methods

signature(object = "prev") Corresponds to print(object, object@par\$conf.level)

See Also

[prev-class](#)

summary-methods

Methods for Function summary in Package prevalence

Description

Summarize objects of class prev

Usage

```
## S4 method for signature 'prev'  
summary(object, conf.level)
```

Arguments

object An object of class prev
conf.level Confidence level to be used in credibility intervals

Methods

signature(object = "prev") Obtain mean, median, mode, standard deviation, variance, credibility interval and number of samples for each chain separately and for all chains combined.

See Also[prev-class](#)

`truePrev`*Estimate true prevalence from individuals samples*

Description

Bayesian estimation of true prevalence from apparent prevalence obtained by testing *individual* samples.

Usage

```
truePrev(x, n, SE = 1, SP = 1, prior = c(1, 1), conf.level = 0.95,
         nchains = 2, burnin = 5000, update = 10000,
         verbose = FALSE, plot = FALSE)
```

Arguments

<code>x</code>	The apparent number of positive samples
<code>n</code>	The sample size
<code>SE, SP</code>	The prior distribution for sensitivity (SE) and specificity (SP); see 'Details' below for specification of these distributions
<code>prior</code>	The parameters of the prior Beta distribution for true prevalence; defaults to <code>c(1, 1)</code>
<code>conf.level</code>	The confidence level used in the construction of the confidence interval; defaults to 0.95
<code>nchains</code>	The number of chains used in the estimation process; 'n' must be ≥ 2
<code>burnin</code>	The number of discarded model iterations; defaults to 5000
<code>update</code>	The number of withheld model iterations; defaults to 10000
<code>verbose</code>	Logical flag, indicating if JAGS process output should be printed to the R console; defaults to FALSE
<code>plot</code>	Logical flag, indicating if diagnostic plots should be given; defaults to FALSE

Details

`truePrev` calls on **JAGS/rjags** to estimate the true prevalence from the apparent prevalence in a Bayesian framework. The default model, in BUGS language, is given below. To see the actual fitted model, see the model slot of the [prev-object](#).

```

model {
  x ~ dbin(AP, n)
  AP <- SE * TP + (1 - SP) * (1 - TP)
  # SE ~ user-defined (see below)
  # SP ~ user-defined (see below)
  TP ~ dbeta(prior[1], prior[2])
}

```

The test sensitivity (SE) and specificity (SP) can be specified, independently, as one of "fixed", "uniform", "beta", "pert", or "beta-expert", with "fixed" as the default.

Distribution parameters can be specified in a *named* list() as follows:

- **Fixed:** list(dist = "fixed", par)
- **Uniform:** list(dist = "uniform", min, max)
- **Beta:** list(dist = "beta", alpha, beta)
- **Beta-PERT:** list(dist = "pert", method, a, m, b, k)
 'method' must be "Classic" or "Vose";
 'a' denotes the pessimistic (minimum) estimate, 'm' the most likely estimate, and 'b' the optimistic (maximum) estimate;
 'k' denotes the scale parameter.
 See [betaPERT](#) for more information on Beta-PERT parametrization.
- **Beta-Expert:** list(dist = "beta-expert", mode, mean, lower, upper, p)
 'mode' denotes the most likely estimate, 'mean' the mean estimate;
 'lower' denotes the lower bound, 'upper' the upper bound;
 'p' denotes the confidence level of the expert.
 Only mode or mean should be specified; lower and upper can be specified together or alone.
 See [betaExpert](#) for more information on Beta-Expert parametrization.

For Uniform, Beta and Beta-PERT distributions, BUGS-style short-hand notation is also allowed:

- **Uniform:** ~dunif(min, max)
- **Beta:** ~dbeta(alpha, beta)
- **Beta-PERT:** ~dpert(min, mode, max)

Value

An object of class [prev](#).

Note

Markov chain Monte Carlo sampling in truePrev is performed by **JAGS (Just Another Gibbs Sampler)** through the [rjags](#) package. JAGS can be downloaded from <http://sourceforge.net/projects/mcmc-jags/>.

Author(s)

<Brecht.Devleeschauwer@UGent.be>

References

- Speybroeck N, Devleesschauwer B, Joseph L, Berkvens D (2012) *Misclassification errors in prevalence estimation: Bayesian handling with care*. International Journal of Public Health
- Online Shiny application: <http://users.ugent.be/~bdvleess/R/prevalence/shiny/>

See Also

coda for various functions that can be applied to the `prev@mcmc.list` object
truePrevMulti: estimate true prevalence from apparent prevalence obtained by testing *individual* samples with multiple tests
truePrevPools: estimate true prevalence from apparent prevalence obtained by testing *pooled* samples
betaPERT: calculate the parameters of a Beta-PERT distribution
betaExpert: calculate the parameters of a Beta distribution based on expert opinion

Examples

```
## Taenia solium cysticercosis in Nepal
## 142 positives out of 742 pigs sampled

## Model SE and SP based on literature data
## Sensitivity ranges uniformly between 60% and 100%
## Specificity ranges uniformly between 75% and 100%
#> BUGS-style:
truePrev(x = 142, n = 742,
         SE = ~dunif(0.60, 1.00), SP = ~dunif(0.75, 1.00))

#> list-style:
SE <- list(dist = "uniform", min = 0.60, max = 1.00)
SP <- list(dist = "uniform", min = 0.75, max = 1.00)
truePrev(x = 142, n = 742, SE = SE, SP = SP)

## Model SE and SP based on expert opinions
## Sensitivity lies in between 60% and 100%; most likely value is 90%
## Specificity is with 95% confidence larger than 75%; most likely value is 90%
SE <- list(dist = "pert", a = 0.60, m = 0.90, b = 1.00)
SP <- list(dist = "beta-expert", mode = 0.90, lower = 0.75, p = 0.95)
truePrev(x = 142, n = 742, SE = SE, SP = SP)

## Model SE and SP as fixed values (each 90%)
truePrev(x = 142, n = 742, SE = 0.90, SP = 0.90)
```

Description

Bayesian estimation of true prevalence from apparent prevalence obtained by applying *multiple* tests to *individual* samples. truePrevMulti implements the approach described by Berkvens et al. (2006), which uses a multinomial distribution to model observed test results, and in which conditional dependence between tests is modelled through conditional probabilities.

Usage

```
truePrevMulti(x, n, prior,
              conf.level = 0.95, nchains = 2, burnin = 5000, update = 10000,
              verbose = FALSE, plot = FALSE)
```

Arguments

x	Vector of apparent test results; see 'Details' below
n	The total sample size
prior	The prior distribution for theta; see 'Details' below
conf.level	The confidence level used in the construction of the confidence interval; defaults to 0.95
nchains	The number of chains used in the estimation process; must be ≥ 2
burnin	The number of discarded model iterations; defaults to 5000
update	The number of withheld model iterations; defaults to 10000
verbose	Logical flag, indicating if JAGS process output should be printed to the R console; defaults to FALSE
plot	Logical flag, indicating if diagnostic plots should be given; defaults to FALSE

Details

truePrevMulti calls on **JAGS/rjags** to estimate the true prevalence from the apparent prevalence in a Bayesian framework. truePrevMulti fits a multinomial model to the apparent test results obtained by testing individual samples with a given number of tests. To see the actual fitted model, see the model slot of the [prev](#)-object.

The vector of apparent tests results, x, must contain the number of samples corresponding to each combination of test results. To see how this vector is defined for the number of tests n at hand, use [definition_x](#).

The prior in the multinomial model consists of a vector theta, which holds values for the true prevalence (TP), the sensitivity and specificity of the first test (SE1, SP1), and the conditional dependencies between the results of the subsequent tests and the preceding one(s). To see how this vector is defined for the number of tests n at hand, use [definition_theta](#).

The values of prior can be specified in two ways, referred to as BUGS-style and list-style, respectively. See also below for some examples.

For BUGS-style specification, the values of theta should be given between curly brackets (i.e., {}), separated by line breaks. theta values can be specified to be deterministic (i.e., fixed), using the <- operator, or stochastic, using the ~ operator. In the latter case, the following distributions can be used:

- **Uniform:** dunif(min, max)
- **Beta:** dbeta(alpha, beta)
- **Beta-PERT:** dpert(min, mode, max)

Alternatively, theta values can be specified in a *named list()* as follows:

- **Fixed:** list(dist = "fixed", par)
- **Uniform:** list(dist = "uniform", min, max)
- **Beta:** list(dist = "beta", alpha, beta)
- **Beta-PERT:** list(dist = "pert", method, a, m, b, k)
'method' must be "Classic" or "Vose";
'a' denotes the pessimistic (minimum) estimate, 'm' the most likely estimate, and 'b' the optimistic (maximum) estimate;
'k' denotes the scale parameter.
See [betaPERT](#) for more information on Beta-PERT parametrization.
- **Beta-Expert:** list(dist = "beta-expert", mode, mean, lower, upper, p)
'mode' denotes the most likely estimate, 'mean' the mean estimate;
'lower' denotes the lower bound, 'upper' the upper bound;
'p' denotes the confidence level of the expert.
Only mode or mean should be specified; lower and upper can be specified together or alone.
See [betaExpert](#) for more information on Beta-Expert parametrization.

Value

An object of class [prev](#).

Note

Markov chain Monte Carlo sampling in truePrevMulti is performed by **JAGS (Just Another Gibbs Sampler)** through the [rjags](#) package. JAGS can be downloaded from <http://sourceforge.net/projects/mcmc-jags/>.

Author(s)

<Brecht.Devleeschauwer@UGent.be>

References

- Berkvens D, Speybroeck N, Praet N, Adel A, Lesaffre E (2006) *Estimating disease prevalence in a Bayesian framework using probabilistic constraints*. Epidemiology 17; 145-153
- Habib I, Sampers I, Uyttendaele M, De Zutter L, Berkvens D (2008) *A Bayesian modelling framework to estimate Campylobacter prevalence and culture methods sensitivity: application to a chicken meat survey in Belgium*. Journal of Applied Microbiology 105; 2002-2008

- Geurden T, Berkvens D, Casaert S, Vercruyssen J, Claerebout E (2008) *A Bayesian evaluation of three diagnostic assays for the detection of Giardia duodenalis in symptomatic and asymptomatic dogs*. Veterinary Parasitology 157; 14-20

See Also

[definition_x](#): how to define the vector of apparent test results x

[definition_theta](#): how to define the vector of theta values in prior

[coda](#) for various functions that can be applied to the `prev@mcmc` object

[truePrev](#): estimate true prevalence from apparent prevalence obtained by testing *individual* samples with a single test

[truePrevPools](#): estimate true prevalence from apparent prevalence obtained by testing *pooled* samples

[betaPERT](#): calculate the parameters of a Beta-PERT distribution

[betaExpert](#): calculate the parameters of a Beta distribution based on expert opinion

Examples

```
## Not run:
## ===== ##
## 2-TEST EXAMPLE: Campylobacter ##
## ----- ##
## Two tests were performed on 656 chicken meat samples ##
## -> T1 = enrichment culture ##
## -> T2 = direct plating ##
## The following assumption were made: ##
## -> TP is larger than 45% and smaller than 80% ##
## -> SE1 must lie within 24% and 50% ##
## -> SP1 and SP2 both equal 100% ##
## -> beta(30, 12) describes P(T2+|D+,T1+) ##
## The following results were obtained: ##
## -> 341 samples T1-,T2- ##
## -> 156 samples T1-,T2+ ##
## -> 46 samples T1+,T2- ##
## -> 113 samples T1+,T2+ ##
## ===== ##

## how is the 2-test model defined?
definition_x(2)
definition_theta(2)

## fit campylobacter 2-test model
campy <-
truePrevMulti(
  x = c(341, 156, 46, 113),
  n = 656,
  prior = {
    theta[1] ~ dunif(0.45, 0.80)
    theta[2] ~ dunif(0.24, 0.50)
    theta[3] <- 1
```

```

    theta[4] ~ dbeta(30, 12)
    theta[5] ~ dbeta(1, 1)
    theta[6] <- 1
    theta[7] <- 1
  }
)

## fit same model using 'list-style'
campy <-
truePrevMulti(
  x = c(341, 156, 46, 113),
  n = 656,
  prior =
    list(
      theta1 = list(dist = "uniform", min = 0.45, max = 0.80),
      theta2 = list(dist = "uniform", min = 0.24, max = 0.50),
      theta3 = 1,
      theta4 = list(dist = "beta", alpha = 30, beta = 12),
      theta5 = list(dist = "beta", alpha = 1, beta = 1),
      theta6 = 1,
      theta7 = 1
    )
)

## show model results
campy

## explore model structure
str(campy)          # overall structure
str(campy@par)     # structure of slot 'par'
str(campy@mcmc)    # structure of slot 'mcmc'
campy@model        # fitted model
campy@diagnostics # DIC, BGR and Bayes-P values

## standard methods
print(campy)
summary(campy)
par(mfrow = c(2, 2))
plot(campy)        # shows plots of TP by default
plot(campy, "SE1") # same plots for SE1
plot(campy, "SE2") # same plots for SE2

## coda plots of TP, SE1, SE2
par(mfrow = c(1, 3))
densplot(campy, col = "red")
traceplot(campy)
gelman.plot(campy)
autocorr.plot(campy)

## ===== ##
## 3-TEST EXAMPLE: Giardia ##
## ----- ##

```

```

## Three tests were performed on stools from 272 dogs ##
## -> T1 = immunofluorescence assay ##
## -> T2 = direct microscopy ##
## -> T3 = SNAP immunochromatography ##
## The following assumption were made: ##
## -> TP is smaller than 20% ##
## -> SE1 must be higher than 80% ##
## -> SP1 must be higher than 90% ##
## The following results were obtained: ##
## -> 220 samples T1-,T2-,T3- ##
## -> 3 samples T1-,T2-,T3+ ##
## -> 14 samples T1-,T2+,T3- ##
## -> 1 sample T1-,T2+,T3+ ##
## -> 12 samples T1+,T2-,T3- ##
## -> 12 samples T1+,T2-,T3+ ##
## -> 4 samples T1+,T2+,T3- ##
## -> 6 samples T1+,T2+,T3+ ##
## ===== ##

## how is the 3-test model defined?
definition_x(3)
definition_theta(3)

## fit giardia 3-test model
giardia <-
truePrevMulti(
  x = c(220, 3, 14, 1, 12, 12, 4, 6),
  n = 272,
  prior = {
    theta[1] ~ dunif(0.00, 0.20)
    theta[2] ~ dunif(0.90, 1.00)
    theta[3] ~ dunif(0.80, 1.00)
    theta[4] ~ dbeta(1, 1)
    theta[5] ~ dbeta(1, 1)
    theta[6] ~ dbeta(1, 1)
    theta[7] ~ dbeta(1, 1)
    theta[8] ~ dbeta(1, 1)
    theta[9] ~ dbeta(1, 1)
    theta[10] ~ dbeta(1, 1)
    theta[11] ~ dbeta(1, 1)
    theta[12] ~ dbeta(1, 1)
    theta[13] ~ dbeta(1, 1)
    theta[14] ~ dbeta(1, 1)
    theta[15] ~ dbeta(1, 1)
  }
)

## show model results
giardia

## coda densplots
par(mfcol = c(2, 4))
densplot(giardia, col = "red")

```



```
## End(Not run)
```

```
truePrevPools      Estimate true prevalence from pooled samples
```

Description

Bayesian estimation of true prevalence from apparent prevalence obtained by testing *pooled* samples.

Usage

```
truePrevPools(x, n, SE = 1, SP = 1, prior = c(1, 1), conf.level = 0.95,
              nchains = 2, burnin = 5000, update = 10000,
              verbose = FALSE, plot = FALSE)
```

Arguments

x	The vector of indicator variables, indicating whether a pool was positive ("1") or negative ("0")
n	The vector of pool sizes
SE, SP	The prior distribution for sensitivity (SE) and specificity (SP); see 'Details' below for specification of these distributions
prior	The parameters of the prior Beta distribution for true prevalence; defaults to c(1, 1)
conf.level	The confidence level used in the construction of the confidence interval; defaults to 0.95
nchains	The number of chains used in the estimation process; 'n' must be ≥ 2
burnin	The number of discarded model iterations; defaults to 5000
update	The number of withheld model iterations; defaults to 10000
verbose	Logical flag, indicating if JAGS process output should be printed to the R console; defaults to FALSE
plot	Logical flag, indicating if diagnostic plots should be given; defaults to FALSE

Details

truePrevPools calls on **JAGS/rjags** to estimate the true prevalence from the apparent prevalence in a Bayesian framework. The default model, in BUGS language, is given below. To see the actual fitted model, see the model slot of the `prev`-object.

```

model {
  for (i in 1:N) {
    x[i] ~ dbern(AP[i])
    AP[i] <- SEpool[i] * (1 - pow(1 - TP, n[i])) + (1 - SPpool[i]) * pow(1 - TP, n[i])
    SEpool[i] <- 1 - (pow(1 - SE, n[i] * TP) * pow(SP, n[i] * (1 - TP)))
    SPpool[i] <- pow(SP, n[i])
  }
  # SE ~ user-defined (see below)
  # SP ~ user-defined (see below)
  TP ~ dbeta(prior[1], prior[2])
}

```

The test sensitivity (SE) and specificity (SP) can be specified by the user, independently, as one of "fixed", "uniform", "beta", "pert", or "beta-expert", with "fixed" as the default. Note that SE and SP must correspond to the test characteristics for testing individual samples; truePrevPools will calculate SEpool and SPpool, the sensitivity and specificity for testing pooled samples, based on Boelaert et al. (2000).

Distribution parameters can be specified in a *named* list() as follows:

- **Fixed:** list(dist = "fixed", par)
- **Uniform:** list(dist = "uniform", min, max)
- **Beta:** list(dist = "beta", alpha, beta)
- **PERT:** list(dist = "pert", method, a, m, b, k)
'method' must be "Classic" or "Vose";
'a' denotes the pessimistic (minimum) estimate, 'm' the most likely estimate, and 'b' the optimistic (maximum) estimate;
'k' denotes the scale parameter.
See [betaPERT](#) for more information on Beta-PERT parametrization.
- **Beta-Expert:** list(dist = "beta-expert", mode, mean, lower, upper, p)
'mode' denotes the most likely estimate, 'mean' the mean estimate;
'lower' denotes the lower bound, 'upper' the upper bound;
'p' denotes the confidence level of the expert.
Only mode or mean should be specified; lower and upper can be specified together or alone.
See [betaExpert](#) for more information on Beta-Expert parametrization.

For Uniform, Beta and Beta-PERT distributions, BUGS-style short-hand notation is also allowed:

- **Uniform:** ~dunif(min, max)
- **Beta:** ~dbeta(alpha, beta)
- **Beta-PERT:** ~dpert(min, mode, max)

Value

An object of class [prev](#).

Note

Markov chain Monte Carlo sampling in truePrevPools is performed by **JAGS (Just Another Gibbs Sampler)** through the **rjags** package. JAGS can be downloaded from <http://sourceforge.net/projects/mcmc-jags/>.

Author(s)

<Brecht.Devleeschauwer@UGent.be>

References

- Speybroeck N, Williams CJ, Lafia KB, Devleeschauwer B, Berkvens D (2012) *Estimating the prevalence of infections in vector populations using pools of samples*. Medical and Veterinary Entomology 26; 361-371
- Boelaert F, Walravens K, Biront P, Vermeersch JP, Berkvens D, Godfroid J (2000) *Prevalence of paratuberculosis (John's disease) in the Belgian cattle population*. Veterinary Microbiology 77; 269-281

See Also

coda for various functions that can be applied to the prev@mcmc.list object
truePrev: estimate true prevalence from apparent prevalence obtained by testing *individual* samples with a single test
truePrevMulti: estimate true prevalence from apparent prevalence obtained by testing *individual* samples with multiple tests
betaPERT: calculate the parameters of a Beta-PERT distribution
betaExpert: calculate the parameters of a Beta distribution based on expert opinion

Examples

```
## Sandflies in Aurabani, Nepal, 2007
pool_results <- c(0, 0, 0, 0, 0, 0, 0, 0, 1, 0)
pool_sizes <- c(2, 1, 6, 10, 1, 7, 1, 4, 1, 3)

## Sensitivity ranges uniformly between 60% and 95%
## Specificity is considered to be 100%

#> BUGS-style:
truePrevPools(x = pool_results, n = pool_sizes,
              SE = ~dunif(0.60, 0.95), SP = 1)

#> list-style:
SE <- list(dist = "uniform", min = 0.60, max = 0.95)
truePrevPools(x = pool_results, n = pool_sizes,
              SE = SE, SP = 1)
```

Index

- *Topic **Expert**
 - betaExpert, 3
 - betaPERT, 5
- *Topic **PERT**
 - betaPERT, 5
- *Topic **classes**
 - prev-class, 10
- *Topic **confidence interval**
 - propCI, 13
- *Topic **methods**
 - plot-methods, 9
 - plot-methods-coda, 10
 - print-methods, 12
 - show-methods, 15
 - summary-methods, 15
- *Topic **package**
 - prevalence-package, 2
- *Topic **prevalence**
 - propCI, 13

- autocorr.plot, 10
- autocorr.plot,prev-method (plot-methods-coda), 10
- autocorr.plot-methods (plot-methods-coda), 10
- autocorrelation, 9, 10

- betaExpert, 2, 3, 3, 4, 7, 17, 18, 20, 21, 25, 26
- betaPERT, 2, 4, 5, 17, 18, 20, 21, 25, 26
- Brooks-Gelman-Rubin, 10
- Brooks-Gelman-Rubin statistic, 12

- coda, 18, 21, 26

- definition, 7
- definition_theta, 19, 21
- definition_theta (definition), 7
- definition_x, 19, 21
- definition_x (definition), 7
- density, 9, 10

- densplot, 10
- densplot,prev-method (plot-methods-coda), 10
- densplot-methods (plot-methods-coda), 10

- gelman.diag, 12
- gelman.plot, 10
- gelman.plot,prev-method (plot-methods-coda), 10
- gelman.plot-methods (plot-methods-coda), 10

- optimize, 4

- plot,prev,ANY-method (plot-methods), 9
- plot,prev-method (plot-methods), 9
- plot-methods, 9
- plot-methods-coda, 10
- plot.betaPERT (betaPERT), 5
- prev, 16, 17, 19, 20, 24, 25
- prev-class, 10
- prevalence (prevalence-package), 2
- prevalence-package, 2
- print,prev-method (print-methods), 12
- print-methods, 12
- print.betaExpert (betaExpert), 3
- print.betaPERT (betaPERT), 5
- propCI, 2, 13

- rjags, 2, 16, 17, 19, 20, 24, 26

- show,prev-method (show-methods), 15
- show-methods, 15
- summary,prev-method (summary-methods), 15
- summary-methods, 15

- trace, 9, 10
- traceplot, 10
- traceplot,prev-method (plot-methods-coda), 10

traceplot-methods (plot-methods-coda),
10
truePrev, 2, 10, 11, 16, 21, 26
truePrevMulti, 2, 7, 8, 10–12, 18, 18, 26
truePrevPools, 2, 10, 11, 18, 21, 24