Package ‘bootComb’

October 12, 2022

Type Package

Title Combine Parameter Estimates via Parametric Bootstrap

Version 1.1.2

Description
Propagate uncertainty from several estimates when combining these estimates via a function. This is done by using the parametric bootstrap to simulate values from the distribution of each estimate to build up an empirical distribution of the combined parameter. Finally either the percentile method is used or the highest density interval is chosen to derive a confidence interval for the combined parameter with the desired coverage. Gaussian copulas are used for when parameters are assumed to be dependent / correlated. References: Davison and Hinkley (1997,ISBN:0-521-57471-4) for the parametric bootstrap and percentile method, Gelman et al. (2014,ISBN:978-1-4398-4095-5) for the highest density interval, Stockdale et al. (2020)<doi:10.1016/j.jhep.2020.04.008> for an example of combining conditional prevalences.

License GPL-3

Encoding UTF-8

Imports MASS (>= 7.3.54)

Suggests HDInterval (>= 0.2.2)

RoxygenNote 7.1.1

NeedsCompilation no

Author Marc Henrion [aut, cre] (<https://orcid.org/0000-0003-1242-839X>)

Maintainer Marc Henrion <mhenrion@mlw.mw>

Repository CRAN

Date/Publication 2022-01-30 19:40:02 UTC

R topics documented:

`adjPrevSensSpec` *Adjust a prevalence point estimate for a given assay sensitivity and specificity.*

Description

Given a reported prevalence estimate from an imperfect assay with known sensitivity and specificity, this function will adjust the prevalence point estimate for the assay sensitivity and specificity.

Usage

```
adjPrevSensSpec(prevEst, sens, spec, replaceImpossibleValues = FALSE)
```

Arguments

`prevEst`	The reported prevalence point estimate.
`sens`	The known assay sensitivity.
`spec`	The known assay specificity.
`replaceImpossibleValues`	Logical; not all combinations of prevalence, sensitivity and specificity are possible and it can be that the adjusted prevalence is <0 or >1, so if this parameter is set to TRUE, values below 0 are set to 0, values above 1 to 1. Default to FALSE.

Value

A vector of the same length as prevEst, returning the adjusted prevalence estimates.

See Also

adjPrevSensSpecCI, ssBetaPars, optim, dbeta

Examples

```
adjPrevSensSpec(prevEst=0.16,sens=0.90,spec=0.95)
```

---

adjPrevSensSpecCI — *Adjust a prevalence point estimate and confidence interval for a given assay sensitivity and specificity (also known only imprecisely).*

Description

This function takes as input a prevalence confidence interval, a sensitivity confidence interval and a specificity confidence interval and returns a confidence interval with the desired coverage of the adjusted prevalence. Optionally the point estimates of prevalence, sensitivity and specificity can also be specified and, if so, these will be returned together with the confidence interval. This function will automatically replace impossible point estimate values with 0 (if estimate <0) or 1 (if estimate >1) and also update the lower, repsectively upper confidence interval limit in this case.

Usage

```
adjPrevSensSpecCI(
  prevCI,
  sensCI,
  specCI,
  N = 1e+06,
  method = "hdi",
  alpha = 0.05,
  Sigma = NULL,
  doPlot = FALSE,
  prev = NULL,
  sens = NULL,
  spec = NULL,
  ylim = NULL,
  returnBootVals = FALSE,
  seed = NULL
)
```

Arguments

prevCI	A vector of length 2 giving the lower and upper bounds of the confidence interval for the prevalence estimate.
sensCI	A vector of length 2 giving the lower and upper bounds of the confidence interval for the assay sensitivity estimate.

`specCI` A vector of length 2 giving the lower and upper bounds of the confidence interval for the assay specificity estimate.

`N` A (large) integer giving the number of parametric bootstrap samples to take. Defaults to 1e6.

`method` The method uses to derive a confidence interval from the empirical distribution of the combined parameter. Needs to be one of 'hdi' (default; computes the highest density interval) or 'quantile (uses quantiles to derive the confidence interval).

`alpha` The desired confidence level; i.e. the returned confidence interval will have coverage 1-alpha.

`Sigma` Set to NULL if parameters are assumed to be independent (the default). If specified, this needs to be a valid 3x3 covariance matrix for a multivariate normal distribution with variances equal to 1 for all variables (in other words, this really is a correlation matrix).

`doPlot` Logical; indicates whether a graph should be produced showing the input estimated distributions for the prevalence, sensitivity and specificity estimates and the resulting empirical distribution of the adjusted prevalence together with the reported confidence interval. Defaults to FALSE.

`prev` Optional; if not NULL, and parameters `sens` and `spec` are also not NULL, then an adjusted point estimate will also be calculated.

`sens` Optional; if not NULL, and parameters `prev` and `spec` are also not NULL, then an adjusted point estimate will also be calculated.

`spec` Optional; if not NULL, and parameters `prev` and `sens` are also not NULL, then an adjusted point estimate will also be calculated.

`ylim` Optional; a vector of length 2, giving the vertical limits for the top panel of the produced plot. Only used if `doPlot` is set to `TRUE`.

`returnBootVals` Logical; if TRUE then the parameter values computed from the bootstrapped input parameter values will be returned; values for the individual parameters will be reported as a second list element; defaults to FALSE.

`seed` If desired a random seed can be specified so that the same results can be reproduced (this gets passed to function `bootComb`).

Value

A list object with 4 elements:

`estimate` The adjusted prevalence point estimate (only non-NULL if `prev`, `sens` and `spec` are specified).

`conf.int` The confidence interval for the adjusted prevalence.

`bootstrapValues` A vector containing the bootstrapped adjusted prevalence values from the bootstrap samples of the input parameters. (Only non-NULL if `returnBootVals` is set to TRUE.)

`bootstrapValuesInput`
A list where each element is the vector of the bootstrapped values for the corresponding input parameters (prevalence, sensitivity, specificity). This can be useful to check the dependence structure that was specified. (Only non-NULL if `returnBootVals` is set to TRUE.)

See Also

`bootComb`, `adjPrevSensSpec`, `identifyBetaPars`, `dbeta`, `hdi`

Examples

```
adjPrevSensSpecCI(
  prevCI=binom.test(x=84,n=500)$conf.int,
  sensCI=binom.test(x=238,n=270)$conf.int,
  specCI=binom.test(x=82,n=88)$conf.int,
  doPlot=TRUE,
  prev=84/500,
  sens=238/270,
  spec=82/88)
```

`bootComb` *Combine parameter estimates via bootstrap*

Description

This package propagates uncertainty from several estimates when combining these estimates via a function. It does this by using the parametric bootstrap to simulate values from the distribution of each estimate to build up an empirical distribution of the combined parameter. Finally either the percentile method is used or the highest density interval is chosen to derive a confidence interval for the combined parameter with the desired coverage.

Usage

```
bootComb(
  distList,
  combFun,
  N = 1e+06,
  distributions = NULL,
  qLowVect = NULL,
  qUppVect = NULL,
  alphaVect = 0.05,
  Sigma = NULL,
  method = "quantile",
  coverage = 0.95,
  doPlot = FALSE,
  legPos = "topright",
```

```
  returnBootVals = FALSE,
  validRange = NULL,
  seed = NULL
)
```

Arguments

- `distList` — If `Sigma` is set to NULL, this is a list object where each element of the list is a sampling function for a probability distribution function (i.e. like rnorm, rbeta, ...). If `Sigma` is specified, then this needs to be a list of quantile functions for the distributions for each parameter.
- `combFun` — The function to combine the different estimates to a new parameter. Needs to take a single list as input argument, one element of the list for each estimate. This list input argument needs to be a list of same length as distList.
- `N` — The number of bootstrap samples to take. Defaults to 1e6.
- `distributions` — Alternatively to specifying `distlist`, the parameters `distributions`, `qLowVect`, `qUppVect` and (optionally) `alphaVect` can be specified. The first 3 of these need t be either all specified and be vectors of the same length or all set to NULL. The `distributions` parameter needs to be a vector specifying the names of the distributions for each parameter (one of "beta", "exponential", "gamma", "normal", "Poisson" or "NegativeBinomial").
- `qLowVect` — Alternatively to specifying `distlist`, the parameters `distributions`, `qLowVect`, `qUppVect` and (optionally) `alphaVect` can be specified. The first 3 of these need t be either all specified and be vectors of the same length or all set to NULL. The `qLowVect` parameter needs to be a vector specifying the lower confidence interval limits for each parameter.
- `qUppVect` — Alternatively to specifying `distlist`, the parameters `distributions`, `qLowVect`, `qUppVect` and (optionally) `alphaVect` can be specified. The first 3 of these need t be either all specified and be vectors of the same length or all set to NULL. The `qUppVect` parameter needs to be a vector specifying the upper confidence interval limits for each parameter.
- `alphaVect` — Alternatively to specifying `distlist`, the parameters `distributions`, `qLowVect`, `qUppVect` and (optionally) `alphaVect` can be specified. The first 3 of these need t be either all specified and be vectors of the same length or all set to NULL. The `alphaVect` parameter needs to be a vector specifying the alpha level (i.e. 1 minus the coverage) of each confidence interval. Can be specified as a single number if the same for all parameters. Defaults to 0.05.
- `Sigma` — Set to NULL if parameters are assumed to be independent (the default). If specified, this needs to be a valid covariance matrix for a multivariate normal distribution with variances equal to 1 for all variables (in other words, this really is a correlation matrix).
- `method` — The method uses to derive a confidence interval from the empirical distribution of the combined parameter.Needs to be one of 'quantile' (default; uses the percentile method to derive the confidence interval) or hdi' (computes the highest density interval).
- `coverage` — The desired coverage of the resulting confidence interval.Defaults to 0.95.

`doPlot`	Logical; indicates whether a graph should be produced showing the input distributions and the resulting empirical distribution of the combined estimate together with the reported confidence interval. Defaults to FALSE.
`legPos`	Legend position (only used if doPlot==TRUE); either NULL (no legend) or one of "top", "topleft", "topright", "bottom", "bottomleft", "bottomright" "left", "right", "center".
`returnBootVals`	Logical; if TRUE then the parameter values computed from the bootstrapped input parameter values will be returned; values for the individual parameters will be reported as a second list element; defaults to FALSE.
`validRange`	Optional; if not NULL, a vector of length 2 giving the range within which the values obtained from the bootstrapped input parameters must lie; values outside this range will be discarded. Behaviour that results in the need for this option arises when parameters are not independent. Use with caution.
`seed`	If desired a random seed can be specified so that the same results can be reproduced.

Value

A list with 3 elements:

`conf.int`	A vector of length 2 giving the lower and upper limits of the computed confidence interval.
`bootstrapValues`	A vector containing the computed / combined parameter values from the bootstrap samples of the input parameters. (Only non-NULL if `returnBootVals` is set to TRUE.)
`bootstrapValuesInput`	A list where each element is the vector of the bootstrapped values for the corresponding input parameter. This can be useful to check the dependence structure that was specified. (Only non-NULL if `returnBootVals` is set to TRUE.)

See Also

hdi

Examples

```
## Example 1 - product of 2 probability parameters for which only the 95% CIs are reported
dist1<-getBetaFromCI(qLow=0.4,qUpp=0.6,alpha=0.05)
dist2<-getBetaFromCI(qLow=0.7,qUpp=0.9,alpha=0.05)
distListEx<-list(dist1$r,dist2$r)
combFunEx<-function(pars){pars[[1]]*pars[[2]]}
bootComb(distList=distListEx,
         combFun=combFunEx,
         doPlot=TRUE,
         method="hdi",
        N=1e5, # reduced from N=1e6 so that it runs quicker; larger values => more accurate
         seed=352)
```

```
# Alternatively, the same example can be run in just 2 lines of code:
combFunEx<-function(pars){pars[[1]]*pars[[2]]}
bootComb(distributions=c("beta","beta"),
         qLowVect=c(0.4,0.7),
         qUppVect=c(0.6,0.9),
         combFun=combFunEx,
         doPlot=TRUE,
         method="hdi",
       N=1e5, # reduced from N=1e6 so that it runs quicker; larger values => more accurate
         seed=352)

## Example 2 - sum of 3 Gaussian distributions
dist1<-function(n){rnorm(n,mean=5,sd=3)}
dist2<-function(n){rnorm(n,mean=2,sd=2)}
dist3<-function(n){rnorm(n,mean=1,sd=0.5)}
distListEx<-list(dist1,dist2,dist3)
combFunEx<-function(pars){pars[[1]]+pars[[2]]+pars[[3]]}
bootComb(distList=distListEx,combFun=combFunEx,doPlot=TRUE,method="quantile")

# Compare with theoretical result:
exactCI<-qnorm(c(0.025,0.975),mean=5+2+1,sd=sqrt(3^2+2^2+0.5^2))
print(exactCI)
x<-seq(-10,30,length=1e3)
y<-dnorm(x,mean=5+2+1,sd=sqrt(3^2+2^2+0.5^2))
lines(x,y,col="red")
abline(v=exactCI[1],col="red",lty=3)
abline(v=exactCI[2],col="red",lty=3)

## Example 3 - same as Example 1 but assuming the 2 parameters to be dependent / correlated
combFunEx<-function(pars){pars[[1]]*pars[[2]]}
bootComb(distributions=c("beta","beta"),
         qLowVect=c(0.4,0.7),
         qUppVect=c(0.6,0.9),
         Sigma=matrix(byrow=TRUE,ncol=2,c(1,0.5,0.5,1)),
         combFun=combFunEx,
         doPlot=TRUE,
         method="hdi",
       N=1e5, # reduced from N=1e6 so that it runs quicker; larger values => more accurate
         seed=352)
```

`getBetaFromCI` *Find the best-fit beta distribution for a given confidence interval for a probability parameter.*

Description

Finds the best-fit beta distribution for a given confidence interval for a probability parameter; returns the corresponding density, distribution, quantile and sampling functions.

Usage

```
getBetaFromCI(qLow, qUpp, alpha = 0.05, initPars = c(50, 50), maxiter = 1000)
```

Arguments

qLow	The observed lower quantile.
qUpp	The observed upper quantile.
alpha	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
initPars	A vector of length 2 giving the initial parameter values to start the optimisation; defaults to c(50,50).
maxiter	Maximum number of iterations for optim. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A list with 5 elements:

r	The sampling function.
d	The density function.
p	The distribution function.
q	The quantile function.
pars	A vector of length 2 giving the two shape parameters for the best-fit beta distribution (shape1 and shape2 as in rbeta, dbeta, pbeta, qbeta).

See Also

identifyBetaPars, optim, dbeta

Examples

```
b<-getBetaFromCI(qLow=0.1167,qUpp=0.1636,initPars=c(200,800))
print(b$pars) # the fitted parameter values
b$r(10) # 10 random values from the fitted beta distribution
b$d(0.15) # the probability density at x=0.15 for the fitted beta distribution
b$p(0.15) # the cumulative density at x=0.15 for the fitted beta distribution
b$q(c(0.25,0.5,0.75)) # the 25th, 50th (median) and 75th percentiles of the fitted distribution
x<-seq(0,1,length=1e3)
y<-b$d(x)
plot(x,y,type="l",xlab="",ylab="density") # density plot for the fitted beta distribution
```

getExpFromCI *Find the best-fit exponential distribution for a given confidence interval.*

Description

Finds the best-fit exponential distribution for a given confidence interval; returns the corresponding density, distribution, quantile and sampling functions.

Usage

```
getExpFromCI(qLow, qUpp, alpha = 0.05, initPars = 1, maxiter = 1000)
```

Arguments

`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
`initPars`	A single number giving the initial rate parameter value to start the optimisation; defaults to 1.
`maxiter`	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A list with 5 elements:

`r`	The sampling function.
`d`	The density function.
`p`	The distribution function.
`q`	The quantile function.
`pars`	A single number giving the rate parameter for the best-fit exponential distribution (`rate` as in `rexp`, `dexp`, `pexp`, `qexp`).

See Also

`identifyExpPars`, `optim`, `dexp`

Examples

```
n<-getExpFromCI(qLow=0.01,qUpp=1.75)
print(n$pars) # the fitted rate parameter value
n$r(10) # 10 random values from the fitted exponential distribution
n$d(2) # the probability density at x=2 for the exponential distribution
n$p(1.5) # the cumulative density at x=1.5 for the fitted exponential distribution
n$q(c(0.25,0.5,0.75)) # the 25th, 50th (median) and 75th percentiles of the fitted distribution
```

```
x<-seq(0,5,length=1e3)
y<-n$d(x)
plot(x,y,type="l",xlab="",ylab="density") # density plot for the fitted exponential distribution
```

---

getGammaFromCI *Find the best-fit gamma distribution for a given confidence interval.*

Description

Finds the best-fit gamma distribution for a given confidence interval; returns the corresponding density, distribution, quantile and sampling functions.

Usage

```
getGammaFromCI(qLow, qUpp, alpha = 0.05, initPars = c(1, 1), maxiter = 1000)
```

Arguments

qLow	The observed lower quantile.
qUpp	The observed upper quantile.
alpha	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
initPars	A vector of length 2 giving the initial parameter values (shape & rate) to start the optimisation; defaults to c(1,1).
maxiter	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A list with 5 elements:

r	The sampling function.
d	The density function.
p	The distribution function.
q	The quantile function.
pars	A vector of length 2 giving the shape and rate for the best-fit gamma distribution (`shape` and `rate` as in `rgamma`, `dgamma`, `pgamma`, `qgamma`).

See Also

`identifyGammaPars`, `optim`, `dgamma`

Examples

```
n<-getGammaFromCI(qLow=0.82,qUpp=5.14)
print(n$pars) # the fitted parameter values (shape & rate)
n$r(10) # 10 random values from the fitted gamma distribution
n$d(6) # the probability density at x=6 for the gamma distribution
n$p(2) # the cumulative density at x=2 for the fitted gamma distribution
n$q(c(0.25,0.5,0.75)) # the 25th, 50th (median) and 75th percentiles of the fitted distribution
x<-seq(0,8,length=1e3)
y<-n$d(x)
plot(x,y,type="l",xlab="",ylab="density") # density plot for the fitted gamma distribution
```

---

getNegBinFromCI — *Find the best-fit negative binomial distribution for a given confidence interval.*

Description

Finds the best-fit negative binomial distribution for a given confidence interval; returns the corresponding probability mass, distribution, quantile and sampling functions. The use of this function within the bootComb package is limited: this is a discrete distribution but since users provide confidence intervals, the corresponding parameters will be best approximated by continuous distributions.

Usage

```
getNegBinFromCI(
  qLow,
  qUpp,
  alpha = 0.05,
  initPars = c(10, 0.5),
  maxiter = 1000
)
```

Arguments

`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
`initPars`	A vector of length 2 giving the initial parameter values (size & prob) to start the optimisation; defaults to c(10,0.5).
`maxiter`	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A list with 5 elements:

r	The sampling function.
d	The probability mass function.
p	The distribution function.
q	The quantile function.
pars	A vector of length 2 giving the mean and standard deviation for the best-fit negatove binomial distribution (size and prob as in rnbinom, dnbinom, pnbinom, qnbinom).

See Also

identifyNegBinPars, optim, dnbinom

Examples

```
n<-getNegBinFromCI(qLow=1.96,qUpp=19.12)
print(n$pars) # the fitted parameter values (size & prob)
n$r(10) # 10 random values from the fitted negative binomial distribution
n$d(8) # the probability mass at x=8 for the negative binomial distribution
n$p(12) # the cumulative probability at x=12 for the fitted negative binomial distribution
n$q(c(0.25,0.5,0.75)) # the 25th, 50th (median) and 75th percentiles of the fitted distribution
x<-0:30
y<-n$d(x)
barplot(height=y,names.arg=x,xlab="",ylab="probability mass") # bar plot of the fitted neg. bin. pmf
```

---

getNormFromCI *Find the best-fit normal / Gaussian distribution for a given confidence interval.*

Description

Finds the best-fit normal distribution for a given confidence interval; returns the corresponding density, distribution, quantile and sampling functions.

Usage

```
getNormFromCI(qLow, qUpp, alpha = 0.05, initPars = c(0, 1), maxiter = 1000)
```

Arguments

- `qLow` The observed lower quantile.
- `qUpp` The observed upper quantile.
- `alpha` The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
- `initPars` A vector of length 2 giving the initial parameter values (mean & sd) to start the optimisation; defaults to c(0,1).
- `maxiter` Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A list with 5 elements:

- `r` The sampling function.
- `d` The density function.
- `p` The distribution function.
- `q` The quantile function.
- `pars` A vector of length 2 giving the mean and standard deviation for the best-fit normal distribution (`mean` and `sd` as in `rnorm`, `dnorm`, `pnorm`, `qnorm`).

See Also

`identifyNormPars`, `optim`, `dnorm`

Examples

```
n<-getNormFromCI(qLow=1.08,qUpp=8.92)
print(n$pars) # the fitted parameter values (mean & sd)
n$r(10) # 10 random values from the fitted normal distribution
n$d(6) # the probability density at x=6 for the normal distribution
n$p(4.25) # the cumulative density at x=4.25 for the fitted normal distribution
n$q(c(0.25,0.5,0.75)) # the 25th, 50th (median) and 75th percentiles of the fitted distribution
x<-seq(0,10,length=1e3)
y<-n$d(x)
plot(x,y,type="l",xlab="",ylab="density") # density plot for the fitted normal distribution
```

`getPoisFromCI` *Find the best-fit Poisson distribution for a given confidence interval.*

Description

Finds the best-fit Poisson distribution for a given confidence interval; returns the corresponding probability mass, distribution, quantile and sampling functions. The use of this function within the bootComb package is limited: this is a discrete distribution but since users provide confidence intervals, the corresponding parameters will be best approximated by continuous distributions.

Usage

```
getPoisFromCI(qLow, qUpp, alpha = 0.05, initPars = 5, maxiter = 1000)
```

Arguments

qLow	The observed lower quantile.
qUpp	The observed upper quantile.
alpha	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
initPars	A vector of length 1 giving the initial parameter value (rate parameter) to start the optimisation; defaults to 5.
maxiter	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A list with 5 elements:

r	The sampling function.
d	The probability mass function.
p	The distribution function.
q	The quantile function.
pars	A single number giving the rate parameter for the best-fit Poisson distribution (`lambda` as in `rpois`, `dpois`, `ppois`, `qpois`).

See Also

`identifyPoisPars`, `optim`, `dpois`

Examples

```
n<-getPoisFromCI(qLow=9,qUpp=22)
print(n$par) # the fitted parameter value (lambda)
n$r(10) # 10 random values from the fitted Poisson distribution
n$d(6) # the probability mass at x=6 for the Poisson distribution
n$p(7) # the cumulative probability at x=7 for the fitted Poisson distribution
n$q(c(0.25,0.5,0.75)) # the 25th, 50th (median) and 75th percentiles of the fitted distribution
x<-0:40
y<-n$d(x)
barplot(height=y,names.arg=x,xlab="",ylab="probability mass") # bar plot of the fitted Poisson pmf
```

identifyBetaPars *Determine the parameters of the best-fit beta distribution for a given confidence interval for a probability parameter.*

Description

Finds the best-fit beta distribution parameters for a given confidence interval for a probability parameter and returns the shape1, shape2 parameters.

Usage

```
identifyBetaPars(
  qLow,
  qUpp,
  alpha = 0.05,
  initPars = c(50, 50),
  maxiter = 1000
)
```

Arguments

`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
`initPars`	A vector of length 2 giving the initial parameter values to start the optimisation; defaults to c(50,50).
`maxiter`	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A vector of length 2 giving the 2 parameters shape1 and shape1 for use with rbeta/dbeta/pbeta/qbeta.

See Also

`ssBetaPars`, `optim`, `dbeta`

`identifyExpPars` *Determine the parameters of the best-fit exponential distribution for a given confidence interval.*

Description

Finds the best-fit exponential distribution parameter for a given confidence interval and returns the rate parameter.

Usage

```
identifyExpPars(qLow, qUpp, alpha = 0.05, initPars = 1, maxiter = 1000)
```

Arguments

`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
`initPars`	A single number giving the initial parameter value to start the optimisation; defaults to 1.
`maxiter`	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A single number giving the rate parameter for use with rexp/dexp/pexp/qexp.

See Also

`ssExpPars`, `optim`, `dexp`

`identifyGammaPars` *Determine the parameters of the best-fit gamma distribution for a given confidence interval.*

Description

Finds the best-fit gamma distribution parameters for a given confidence interval and returns the shape, rate parameters.

Usage

```
identifyGammaPars(qLow, qUpp, alpha = 0.05, initPars = c(1, 1), maxiter = 1000)
```

Arguments

`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
`initPars`	A vector of length 2 giving the initial parameter values to start the optimisation; defaults to c(1,1).
`maxiter`	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A vector of length 2 giving the 2 parameters shape and rate for use with rgamma/dgamma/pgamma/qgamma.

See Also

`ssGammaPars`, `optim`, `dgamma`

`identifyNegBinPars` *Determine the parameters of the best-fit negative binomial distribution for a given confidence interval.*

Description

Finds the best-fit negative binomial distribution parameters for a given confidence interval and returns the size, prob parameters.

Usage

```
identifyNegBinPars(
  qLow,
  qUpp,
  alpha = 0.05,
  initPars = c(10, 0.5),
  maxiter = 1000
)
```

Arguments

`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
`initPars`	A vector of length 2 giving the initial parameter values to start the optimisation; defaults to c(10,0.5).
`maxiter`	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A vector of length 2 giving the 2 parameters size and prob for use with rnbinom/dnbinom/pnbinom/qnbinom.

See Also

`ssNegBinPars`, `optim`, `dnbinom`

`identifyNormPars` *Determine the parameters of the best-fit normal / Gaussian distribution for a given confidence interval.*

Description

Finds the best-fit normal distribution parameters for a given confidence interval and returns the mean and sd parameters.

Usage

```
identifyNormPars(qLow, qUpp, alpha = 0.05, initPars = c(0, 1), maxiter = 1000)
```

Arguments

`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
`initPars`	A vector of length 2 giving the initial parameter values to start the optimisation; defaults to c(50,50).
`maxiter`	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A vector of length 2 giving the 2 parameters mean and sd for use with rnorm/dnorm/pnorm/qnorm.

See Also

`ssNormPars`, `optim`, `dnorm`

`identifyPoisPars` *Determine the parameters of the best-fit Poisson distribution for a given confidence interval.*

Description

Finds the best-fit Poisson distribution parameters for a given confidence interval and returns the rate parameter.

Usage

```
identifyPoisPars(qLow, qUpp, alpha = 0.05, initPars = 5, maxiter = 1000)
```

Arguments

`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
`initPars`	A single number > 0, giving the initial parameter value to start the optimisation; defaults to 5.
`maxiter`	Maximum number of iterations for `optim`. Defaults to 1e3. Set to higher values if convergence problems are reported.

Value

A single number giving the rate parameter for use with rpois/dpois/ppois/qpois.

See Also

`ssPoisPars`, `optim`, `dpois`

`simScenPrevSensSpec` *Simulation scenario for adjusting a prevalence for sensitivity and specificity.*

Description

This is a simulation to compute the coverage of the confidence interval returned by bootComb() in the case of adjusting a prevalence estimate for estimates of sensitivity and specificity.

Usage

```
simScenPrevSensSpec(
  B = 1000,
  p,
  sens,
  spec,
  nExp,
  nExpSens,
  nExpSpec,
  alpha = 0.05,
  assumeSensSpecExact = FALSE
)
```

Arguments

B	The number of simulations to run. Defaults to 1e3.
p	The true value of the prevalence parameter.
sens	The true value of the assay sensitivity parameter.
spec	The true value of the assay specificity parameter
nExp	The size of each simulated experiment to estimate p.
nExpSens	The size of each simulated experiment to estimate sens.
nExpSpec	The size of each simulated experiment to estimate spec.
alpha	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.
assumeSensSpecExact	Logical; indicates whether coverage should also be computed for the situation where sensitivity and specificity are assumed to be known exactly. Defaults to FALSE.

Value

A list with 2 or 4 elements, depending whether `assumeSensSpecExact` is set to FALSE or TRUE:

estimate	A single number, the proportion of simulations for which the confidence interval contained the true prevalence parameter value.
conf.int	A confidence interval of coverage 1-alpha for the coverage estimate.
estimate.sensSpecExact	Returned only if `assumeSensSpecExact` is set to TRUE. A single number, the proportion of simulations for which the confidence interval, derived assuming sensitivity and specificity are known exactly, contained the true prevalence parameter value.
conf.int.sensSpecExact	Returned only if `assumeSensSpecExact` is set to TRUE. A confidence interval of coverage 1-alpha for the coverage estimate in the scenario where sensitivity and specificity are assumed to be known exactly.

Examples

```
simScenPrevSensSpec(p=0.15,sens=0.85,spec=0.90,nExp=300,nExpSens=600,nExpSpec=400,B=100)
  # B value only for convenience here
  # Increase B to 1e3 or 1e4 (be aware this may run for some time).
```

---

simScenProductTwoPrevs *Simulation scenario for the product of two prevlaence estimates.*

Description

This is a simulation to compute the coverage of the confidence interval returned by bootComb() in the case of the product of 2 probability parameter estimates.

Usage

```
simScenProductTwoPrevs(B = 1000, p1, p2, nExp1, nExp2, alpha = 0.05)
```

Arguments

B	The number of simulations to run. Defaults to 1e3.
p1	The true value of the first probability parameter.
p2	The true value of the second probability parameter.
nExp1	The size of each simulated experiment to estimate `p1`.
nExp2	TThe size of each simulated experiment to estimate `p2`.
alpha	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.

Value

A list with 2 elements:

estimate	A single number, the proportion of simulations for which the confidence interval contained the true parameter value.
conf.int	A 95% confidence interval for the coverage estimate.

Examples

```
simScenProductTwoPrevs(p1=0.35,p2=0.2,nExp1=100,nExp2=1000,B=100)
  # B value only for convenience here
  # Increase B to 1e3 or 1e4 (be aware this may run for some time).
```

`ssBetaPars` *Compute the sum of squares between the theoretical and observed quantiles of a beta distribution.*

Description

This is a helper function that compute the sum of squares between two theoretical and observed quantiles of a beta distribution (typically the lower and upper bounds of a confidence interval). This function is for internal use to find the best-fit beta distribution for a given confidence interval.

Usage

```
ssBetaPars(abPars, qLow, qUpp, alpha = 0.05)
```

Arguments

`abPars`	The shape1 and shape2 parameters of the theoretical beta distribution.
`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.

Value

A single number, the sum of squares.

See Also

`identifyBetaPars`, `optim`, `qbeta`

`ssExpPars` *Compute the sum of squares between the theoretical and observed quantiles of an exponential distribution.*

Description

This is a helper function that compute the sum of squares between two theoretical and observed quantiles of an exponential distribution (typically the lower and upper bounds of a confidence interval). This function is for internal use to find the best-fit exponential distribution for a given confidence interval.

Usage

```
ssExpPars(ratePar, qLow, qUpp, alpha = 0.05)
```

Arguments

`ratePar`	The rate parameter of the theoretical exponential distribution.
`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.

Value

A single number, the sum of squares.

See Also

`identifyExpPars`, `optim`, `qexp`

`ssGammaPars` *Compute the sum of squares between the theoretical and observed quantiles of a gamma distribution.*

Description

This is a helper function that compute the sum of squares between two theoretical and observed quantiles of a gamma distribution (typically the lower and upper bounds of a confidence interval). This function is for internal use to find the best-fit gamma distribution for a given confidence interval.

Usage

```
ssGammaPars(shapeRatePars, qLow, qUpp, alpha = 0.05)
```

Arguments

`shapeRatePars`	The shape and rate parameters of the theoretical gamma distribution.
`qLow`	The observed lower quantile.
`qUpp`	The observed upper quantile.
`alpha`	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.

Value

A single number, the sum of squares.

See Also

`identifyGammaPars`, `optim`, `qgamma`

ssNegBinPars

Compute the sum of squares between the theoretical and observed quantiles of a negative binomial distribution.

Description

This is a helper function that compute the sum of squares between two theoretical and observed quantiles of a negative binomial distribution (typically the lower and upper bounds of a confidence interval). This function is for internal use to find the best-fit negative binomial distribution for a given confidence interval.

Usage

```
ssNegBinPars(sizeProbPars, qLow, qUpp, alpha = 0.05)
```

Arguments

sizeProbPars	The size and prob parameters of the theoretical negative binomial distribution.
qLow	The observed lower quantile.
qUpp	The observed upper quantile.
alpha	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.

Value

A single number, the sum of squares.

See Also

identifyNegBinPars, optim, qnbinom

ssNormPars

Compute the sum of squares between the theoretical and observed quantiles of a normal / Gaussian distribution.

Description

This is a helper function that compute the sum of squares between two theoretical and observed quantiles of a normal distribution (typically the lower and upper bounds of a confidence interval). This function is for internal use to find the best-fit normal distribution for a given confidence interval.

Usage

```
ssNormPars(muSigPars, qLow, qUpp, alpha = 0.05)
```

Arguments

muSigPars	The mean and standard deviation parameters of the theoretical normal distribution.
qLow	The observed lower quantile.
qUpp	The observed upper quantile.
alpha	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.

Value

A single number, the sum of squares.

See Also

identifyNormPars, optim, qnorm

---

ssPoisPars — *Compute the sum of squares between the theoretical and observed quantiles of a Poisson distribution.*

Description

This is a helper function that compute the sum of squares between two theoretical and observed quantiles of a normal distribution (typically the lower and upper bounds of a confidence interval). This function is for internal use to find the best-fit normal distribution for a given confidence interval.

Usage

```
ssPoisPars(poisPar, qLow, qUpp, alpha = 0.05)
```

Arguments

poisPar	The rate parameter of the theoretical Poisson distribution.
qLow	The observed lower quantile.
qUpp	The observed upper quantile.
alpha	The confidence level; i.e. the desired coverage is 1-alpha. Defaults to 0.05.

Value

A single number, the sum of squares.

See Also

identifyPoisPars, optim, qpois

Index