Package 'bootLR'

May 8, 2026

Type Package

Title Bootstrapped Confidence Intervals for (Negative) Likelihood Ratio Tests

Version 1.0.2

Date 2019-01-27

Author Keith A. Marill and Ari B. Friedman

Maintainer Ari B. Friedman <abfriedman@gmail.com>

Description Computes appropriate confidence intervals for the likelihood ratio tests commonly used in medicine/epidemiology, using the method of Marill et al. (2015) <doi:10.1177/0962280215592907>. It is particularly useful when the sensitivity or specificity in the sample is 100%. Note that this does not perform the test on nested models--for that, see 'epicalc::lrtest'.

License LGPL-2.1

LazyData TRUE

Imports boot, stats, binom

Suggests testthat

Collate 'bootLR.R'

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-01 15:13:30 UTC

Contents

BayesianLR.test 2
bca 4
confusionStatistics 4
diagCI 5
drawMaxedOut 6
medianConsistentlyOne 7
print.diagCI 8

print.lrtest 8
run.BayesianLR.test 9
sequentialGridSearch 10

Index **11**

`BayesianLR.test` *Compute the (positive/negative) likelihood ratio with appropriate, bootstrapped confidence intervals*

Description

Compute the (positive/negative) likelihood ratio with appropriate, bootstrapped confidence intervals. A standard bootstrapping approach is used for sensitivity and specificity, results are combined, and then 95 For the case where sensitivity or specificity equals zero or one, an appropriate bootstrap sample is generated and then used in subsequent computations.

Usage

```
BayesianLR.test(truePos, totalDzPos, trueNeg, totalDzNeg, R = 10^4,
  nBSave = 50, verbose = FALSE, parameters = list(shrink = 5, tol =
  5e-04, nEach = 80), maxTries = 20, ci.width = 0.95,
  consistentQuantile = 0.5, ...)
```

Arguments

`truePos`	The number of true positive tests.
`totalDzPos`	The total number of positives ("sick") in the population.
`trueNeg`	The number of true negatives in the population.
`totalDzNeg`	The total number of negatives ("well") in the population.
`R`	The number of replications in each round of the bootstrap (has been tested at 10,000 or greater).
`nBSave`	The number of times to re-bootstrap the statistic and then average at the end to obtain confidence intervals (has been tested at 50).
`verbose`	Whether to display internal operations as they happen.
`parameters`	List of control parameters (shrink, tol, nEach) for sequential search.
`maxTries`	Each time a run fails, BayesianLR.test will back off on the parameters and try again. maxTries specifies the number of times to try before giving up. If you can't get it to converge, try setting this higher.
`ci.width`	Changing this parameter results in different properties than have been tested and is not recommended. The width of the confidence interval used by boot.ci (not necessarily the same as the width of the CI produced by the algorithm overall)

consistentQuantile
: Changing this parameter results in different properties than have been tested and is not recommended. Defaults to 0.5, i.e. the median. Finds the lowest probability for which the random draws are likely to be consistently one, where consistently is defined by this value (i.e. at .5, a simple majority of the time is enough for consistency).

...
: Arguments to pass along to boot.ci for the BCa confidence intervals.

Details

If the denominator is 0, calculations are inverted until the final result.

Value

An object of class lrtest.

Note

You'll either need a fast computer or substantial patience for certain combinations of inputs.

Examples

```
## Not run:
blrt <- BayesianLR.test( truePos=100, totalDzPos=100, trueNeg=60, totalDzNeg=100 )
blrt
summary(blrt)

BayesianLR.test( truePos=98, totalDzPos=100, trueNeg=60, totalDzNeg=100 )
BayesianLR.test( truePos=60, totalDzPos=100, trueNeg=100, totalDzNeg=100 )
BayesianLR.test( truePos=60, totalDzPos=100, trueNeg=99, totalDzNeg=100 )

# Note the argument names are not necessary if you specify them in the proper order:
BayesianLR.test( 60, 100, 50, 50 )

# You can specify R= to increase/decrease the number of bootstrap replications
BayesianLR.test( 60, 100, 50, 50, R=10000 )

# You can change the number of digits that are printed
print.lrtest( BayesianLR.test( 500, 500, 300, 500 ), digits = 4 )

# Or extract the results yourself
model.blrt1 <- BayesianLR.test( 500, 500, 300, 500 )
unclass( model.blrt1 )
model.blrt1$statistics
model.blrt1$posLR.ci

# If the model doesn't converge, you can alter the search parameters
BayesianLR.test( 500, 500, 300, 500, parameters=list(shrink=4,tol=.001,nEach=150), maxTries = 50 )

### Statistician-only options
# These change the way the model works.
# It is not recommended to alter these, as this will alter the statistical properties of the test
```

```
# in ways that have not been validated.
# Change number of bootstrap replications
BayesianLR.test( 500, 500, 300, 500, R = 5*10^4 )
# Change number of times to average the confidence interval limits at the end
BayesianLR.test( 500, 500, 300, 500, nBSave = 100 )
# Change the criteria from median being consistent 0 or 1 to some other quantile
BayesianLR.test( 500, 500, 300, 500, consistentQuantile = .53 )

## End(Not run)
```

`bca` *Internal function to analyze LR bootstrap finding median, and standard and BCa percentile 95 To obtain bca CI on a non-boot result, use a dummy boot. and replace t and t0 with the results of interest.*

Description

Internal function to analyze LR bootstrap finding median, and standard and BCa percentile 95 To obtain bca CI on a non-boot result, use a dummy boot. and replace t and t0 with the results of interest.

Usage

```
bca(t, t0, ...)
```

Arguments

`t`	The vector to obtain a BCa bootstrap for (e.g. nlr).
`t0`	The central value of the vector (e.g. the).
`...`	Pass-alongs to boot.ci.

`confusionStatistics` *Compute sensitivity, specificity, positive likelihood ratio, negative likelihood ratio for a single 2x2 table*

Description

Compute sensitivity, specificity, positive likelihood ratio, negative likelihood ratio for a single 2x2 table

Usage

```
confusionStatistics(truePos, totalDzPos, trueNeg, totalDzNeg)
```

Arguments

truePos	The number of true positive tests.
totalDzPos	The total number of positives ("sick") in the population.
trueNeg	The number of true negatives in the population.
totalDzNeg	The total number of negatives ("well") in the population.

Value

A one-row matrix containing sensitivity, specificity, posLR, negLR results.

References

Deeks JJ, Altman DG. BMJ. 2004 July 17; 329(7458): 168-169.

Examples

```
## Not run:
confusionStatistics( 25, 50, 45, 75 )

## End(Not run)
```

---

diagCI — *Compute values and confidence intervals for sensitivity, specificity, positive likelihood ratio, negative likelihood ratio for a single 2x2 table*

Description

Compute values and confidence intervals for sensitivity, specificity, positive likelihood ratio, negative likelihood ratio for a single 2x2 table

Usage

```
diagCI(truePos, totalDzPos, trueNeg, totalDzNeg,
  calcLRCI = "BayesianLR.test", alpha = 0.05, binomMethod = "wilson",
  ...)
```

Arguments

truePos	The number of true positive tests.
totalDzPos	The total number of positives ("sick") in the population.
trueNeg	The number of true negatives in the population.
totalDzNeg	The total number of negatives ("well") in the population.
calcLRCI	Method to use to calculate the LR CI: "BayesianLR.test" "none" or "analytic"
alpha	The alpha for the width of the confidence interval (defaults to alpha = 0.05 for a 95 percent CI)

binomMethod	The method to be passed to binom.confint to calculate confidence intervals of proportions (sensitivity, etc.). See help("binom.confint") and the Newcombe article referenced below.
...	Arguments to pass to Bayesian.LRtest.

Value

A matrix containing sensitivity, specificity, posLR, negLR results and their confidence intervals

References

Deeks JJ, Altman DG. BMJ. 2004 July 17; 329(7458): 168-169. Newcombe RG. Statist Med. 1998; 17(857-872).

Examples

```
## Not run:
diagCI( 25, 50, 45, 75 )
diagCI( truePos = c(25, 30), totalDzPos = c( 50, 55 ), trueNeg = c(5, 35), totalDzNeg = c(60,65) )

## End(Not run)
```

---

drawMaxedOut *Internal function to draw a set of sensitivities or specificities This is intended for the case where testPos == totalDzPos or testNeg == totalDzNeg.*

Description

Internal function to draw a set of sensitivities or specificities This is intended for the case where testPos == totalDzPos or testNeg == totalDzNeg.

Usage

```
drawMaxedOut(n, R, consistentQuantile = 0.5, verbose,
  parameters = list(shrink = 5, tol = 5e-04, nEach = 80),
  method = "deterministic")
```

Arguments

n	The total number of positives/negatives in the population.
R	is the number of replications in each round of the bootstrap (has been tested at 10,000 or greater).
consistentQuantile	Defaults to 0.5, i.e. the median. Finds the lowest probability for which the random draws are likely to be consistently one, where consistently is defined by this value (i.e. at .5, a simple majority of the time is enough for consistency)

verbose	Whether to display internal operations as they happen.
parameters	List of control parameters (shrink, tol, nEach) for sequential search.
method	Either "deterministic" or "search". The former is faster and more accurate. Thanks to an anonymous reviewer for pointing out the utility of the binomial distribution in solving this problem.

`medianConsistentlyOne` *Find the lowest population probability whose median is consistently one This is the lowest estimate for Sens that is consistently (over 5 runs) most likely to yield a sample estimate that is all 1's (e.g. 100/100, etc.).*

Description

Find the lowest population probability whose median is consistently one This is the lowest estimate for Sens that is consistently (over 5 runs) most likely to yield a sample estimate that is all 1's (e.g. 100/100, etc.).

Usage

```
medianConsistentlyOne(pr, size, R, nConsistentRuns = 5, warn = TRUE,
  consistentQuantile = 0.5)
```

Arguments

pr	Probability input.
size	Number of trials.
R	number of bootstrap replications.
nConsistentRuns	Number of runs that all have to be identical to return TRUE.
warn	Warn if searching outside of the range c(0,1).
consistentQuantile	Defaults to 0.5 (the median). Change if we want to use a different criterion for consistency than the median

Value

Boolean of length one (TRUE or FALSE).

Examples

```
## Not run:
prs <- seq(.990,.995,.0001)
bools <- sapply( prs, medianConsistentlyOne, size=truePos, R=R )
data.frame( prs, bools )

## End(Not run)
```

`print.diagCI` *Prints results from diagCI As is typical for R, this is run automatically when you type in an object name, and is typically not run directly by the end-user.*

Description

Prints results from diagCI As is typical for R, this is run automatically when you type in an object name, and is typically not run directly by the end-user.

Usage

```
## S3 method for class 'diagCI'
print(x, digits = 3, ...)
```

Arguments

`x`	The diagCI object created by diagCI()
`digits`	Number of digits to round to
`...`	Pass-alongs (currently ignored).

Value

Returns x unaltered.

Examples

```
## Not run:
diagCI( 25, 50, 45, 75 )

## End(Not run)
```

`print.lrtest` *Prints results from the BayesianLR.test As is typical for R, this is run automatically when you type in an object name, and is typically not run directly by the end-user.*

Description

Prints results from the BayesianLR.test As is typical for R, this is run automatically when you type in an object name, and is typically not run directly by the end-user.

Usage

```
## S3 method for class 'lrtest'
print(x, digits = 3, ...)
```

Arguments

x	The lrtest object created by BayesianLR.test.
digits	Number of digits to round to for display purposes
...	Pass-alongs (currently ignored).

Value

Returns x unaltered.

Examples

```
## Not run:
print.lrtest( BayesianLR.test( 500, 500, 300, 500 ), digits = 4 )

## End(Not run)
```

---

run.BayesianLR.test *The actual function that runs the test. BayesianLR.test is a wrapper that runs this with ever-looser tolerances.*

Description

The actual function that runs the test. BayesianLR.test is a wrapper that runs this with ever-looser tolerances.

Usage

```
run.BayesianLR.test(truePos, totalDzPos, trueNeg, totalDzNeg, R = 10^4,
  verbose = FALSE, parameters = list(shrink = 5, tol = 5e-04, nEach =
  80), ci.width = 0.95, consistentQuantile = 0.5, ...)
```

Arguments

truePos	The number of true positive tests.
totalDzPos	The total number of positives ("sick") in the population.
trueNeg	The number of true negatives in the population.
totalDzNeg	The total number of negatives ("well") in the population.
R	is the number of replications in each round of the bootstrap (has been tested at 10,000 or greater).
verbose	Whether to display internal operations as they happen.
parameters	List of control parameters (shrink, tol, nEach) for sequential grid search.
ci.width	The width of the confidence interval used by boot.ci (not necessarily the same as the width of the CI produced by the algorithm overall; if this parameter is changed, results are not tested)

consistentQuantile
: Defaults to 0.5, i.e. the median. Finds the lowest probability for which the random draws are likely to be consistently one, where consistently is defined by this value (i.e. at .5, a simple majority of the time is enough for consistency). Changing this parameter results in different properties than have been tested and is not recommended.

...
: Arguments to pass along to boot.ci for the BCa confidence intervals.

Value

An object of class lrtest.

`sequentialGridSearch` *Optimize a function returning a single numeric value subject to a boolean constraint Utilizes a naive recursive grid search.*

Description

Optimize a function returning a single numeric value subject to a boolean constraint Utilizes a naive recursive grid search.

Usage

```
sequentialGridSearch(f, constraint, bounds, nEach = 40, shrink = 10,
  tol = .Machine$double.eps^0.5, verbose = FALSE, ...)
```

Arguments

f	Function to be minimized: takes a single numeric value and returns a single numeric value.
constraint	Function of a single variable returning a single boolean value (must be TRUE to be at the optimum).
bounds	A numeric vector of length two which are the upper and lower bounds of the input to try.
nEach	Number of points n each round of grid searching to use.
shrink	Factor indicating how much (1/shrink) to narrow the search width by each round; highly recommended that shrink is at least half the size of nEach.
tol	The tolerance (epsilon).
verbose	Whether to display verbose output.
...	Arguments to pass along to constraint.

Value

The optimized input value (numeric).

Index

BayesianLR.test, 2
bca, 4

confusionStatistics, 4

diagCI, 5
drawMaxedOut, 6

medianConsistentlyOne, 7

print.diagCI, 8
print.lrtest, 8

run.BayesianLR.test, 9

sequentialGridSearch, 10