Package ‘bootLR’

February 19, 2015

Type Package
Title Bootstrapped confidence intervals for (negative) likelihood ratio tests
Version 0.9
Date 2014-5-26
Author Keith A. Marill and Ari B. Friedman
Maintainer Ari B. Friedman <abfriedman@gmail.com>
Description This package computes appropriate confidence intervals for the likelihood ratio tests commonly used in medicine/epidemiology. 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
Suggests testthat
Collate 'bootLR.R'
NeedsCompilation no
Repository CRAN
Date/Publication 2014-05-27 06:42:46

R topics documented:

BayesianLR.test ......................................................... 2
bca ................................................................. 3
confusionStatistics ........................................................... 3
drawMaxedOut ............................................................ 4
medianConsistentlyOne ...................................................... 5
print.lrtest ............................................................... 6
run.BayesianLR.test ....................................................... 6
sequentialGridSearch ...................................................... 7

Index 9
BayesianLR.test

**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 are determined. For the case where sensitivity or specificity equals zero or one, an appropriate bootstrap sample is generated and then used in subsequent computations.

**Usage**

```r
BayesianLR.test(truePos, totalDzPos, trueNeg, totalDzNeg,
R = 5 * 10^4, verbose = FALSE,
parameters = list(shrink = 5, tol = 5e-04, nEach = 80),
maxTries = 20, ...)
```

**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 50,000 or greater).
- `verbose`: Whether to display internal operations as they happen.
- `parameters`: List of control parameters (shrink, tol, nEach) for sequential grid 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.
- `...`: 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**

This algorithm utilizes a sequential grid search. You’ll either need a fast computer or substantial patience for certain combinations of inputs.
bca

Examples
bler <- BayesianLR.test( truePos=100, totalDzPos=100, trueNeg=60, totalDzNeg=100 )
bler
summary(bler)
## Not run:
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 the number of bootstrap replications
BayesianLR.test( 60, 100, 50, 50, R=10000 )

## 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 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


Examples

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

## 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, verbose, parameters = list(shrink = 5, tol = 5e-04, nEach = 80))

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 50,000 or greater).
verbose  Whether to display internal operations as they happen.
parameters  List of control parameters (shrink, tol, nEach) for sequential grid search.
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)

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).

Value
Boolean of length one (TRUE or FALSE).

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

## 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

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

### Arguments

- `x`  
The lrtest object created by BayesianLR.test.
- `...`  
  Pass-alongs (currently ignored).

### Value

Returns x unaltered.

---

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

### Description

The actual function that does the running (BayesianLR.test is now a wrapper that runs this with ever-looser tolerances)

### Usage

```r
run.BayesianLR.test(truePos, totalDzPos, trueNeg, totalDzNeg, 
                      R = 5 * 10^4, verbose = FALSE, 
                      parameters = list(shrink = 5, tol = 5e-04, nEach = 80), 
                      ...)  
```
sequentialGridSearch

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>truePos</td>
<td>The number of true positive tests.</td>
</tr>
<tr>
<td>totalDzPos</td>
<td>The total number of positives (&quot;sick&quot;) in the population.</td>
</tr>
<tr>
<td>trueNeg</td>
<td>The number of true negatives in the population.</td>
</tr>
<tr>
<td>totalDzNeg</td>
<td>The total number of negatives (&quot;well&quot;) in the population.</td>
</tr>
<tr>
<td>R</td>
<td>Number of replications in each round of the bootstrap (has been tested at 50,000 or greater).</td>
</tr>
<tr>
<td>verbose</td>
<td>Whether to display internal operations as they happen.</td>
</tr>
<tr>
<td>parameters</td>
<td>List of control parameters (shrink, tol, nEach) for sequential grid search.</td>
</tr>
<tr>
<td>...</td>
<td>Arguments to pass along to boot.ci for the BCa confidence intervals.</td>
</tr>
</tbody>
</table>

**Value**

An object of class lrtest.

---

**Description**

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

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>Function to be minimized: takes a single numeric value and returns a single numeric value.</td>
</tr>
<tr>
<td>constraint</td>
<td>Function of a single variable returning a single boolean value (must be TRUE to be at the optimum).</td>
</tr>
<tr>
<td>bounds</td>
<td>A numeric vector of length two which are the upper and lower bounds of the input to try.</td>
</tr>
<tr>
<td>nEach</td>
<td>Number of points n each round of grid searching to use.</td>
</tr>
<tr>
<td>shrink</td>
<td>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.</td>
</tr>
<tr>
<td>tol</td>
<td>The tolerance (epsilon).</td>
</tr>
<tr>
<td>verbose</td>
<td>Whether to display verbose output.</td>
</tr>
<tr>
<td>...</td>
<td>Arguments to pass along to constraint.</td>
</tr>
</tbody>
</table>
Value

The optimized input value (numeric).
Index

BayesianLR.test, 2
bca, 3
confusionStatistics, 3
drawMaxedOut, 4
medianConsistentlyOne, 5
print.lrtest, 6
run.BayesianLR.test, 6
sequentialGridSearch, 7