Package 'betafunctions'

June 10, 2020

Type Package

Title Functions for Working with Two- And Four-Parameter Beta Probability Distributions

Version 1.2.0

Author Haakon Haakstad

Maintainer Haakon Haakstad <h.t.haakstad@cemo.uio.no>

Description Package providing a number of functions for working with the Two- and Four-parameter Beta distributions, including alternative parameterizations and calculation of moments. Includes functions for estimating classification accuracy, diagnostic performance and consistency, using what's known as the Livingston and Lewis approach in the educational-measurement literature as the base method.
Livingston and Lewis (1995) <doi:10.1111/j.1745-3984.1995.tb00462.x>.
Hanson (1991) <https://files.eric.ed.gov/fulltext/ED344945.pdf>.
Glas, Lijmer, Prins, Bonsel and Bossuyt (2003) <doi:10.1016/S0895-4356(03)00177-X>.

License CC0

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-10 21:30:02 UTC

R topics documented:

`AMS` *Alpha Shape Parameter Given Mean and Variance of a Standard Beta PDD.*

Description

Calculates the Alpha value required to produce a Standard (two-parameter) Beta probability density distribution with defined mean and variance or standard deviation.

Usage

```
AMS(mean, var, sd = NULL)
```

Arguments

`mean`	The mean of the target Standard Beta probability density distribution.
`var`	The variance of the target Standard Beta probability density distribution.
`sd`	The standard deviation of the target Standard Beta probability density distribution.

Value

A numeric value representing the required value for the Alpha shape-parameter in order to produce a Standard Beta probability density distribution with the target mean and variance.

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0, rescaled to proportion
# of maximum.
set.seed(1234)
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3)) / 100
hist(testdata, xlim = c(0, 100))

# To find the alpha shape-parameter of a Standard (two-parameter) Beta
# distribution with the same mean and variance as the observed-score
# distribution using AMS():
AMS(mean(testdata), var(testdata))
```

AUC *Area Under the ROC Curve.*

Description

Given a vector of false-positive rates and a vector of true-positive rates, calculate the area under the Receiver Operator Characteristic (ROC) curve.

Usage

```
AUC(FPR, TPR)
```

Arguments

FPR	Vector of False-Positive Rates.
TPR	Vector of True-Positive Rates.

Value

A value representing the area under the ROC curve.

Note

Script originally retrieved and modified from https://blog.revolutionanalytics.com/2016/11/calculating-auc.html.

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0.
set.seed(1234)
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3))
hist(testdata, xlim = c(0, 100))

# Suppose the cutoff value for attaining a pass is 50 items correct, and
# that the reliability of this test was estimated to 0.7. To calculate the
```

```
# necessary (x, y) coordinates to compute the area under the curve statistic
# one can use the LL.ROC() function with the argument
# raw.out = TRUE.
coords <- LL.ROC(x = testdata, reliability = .7, truecut = 50, min = 0,
max = 100, raw.out = TRUE)

# To calculate and retrieve the Area Under the Curve (AUC) with the AUC()
# function, feed it the raw coordinates calculated above.
AUC(coords[, "FPR"], coords[, "TPR"])
```

`Beta.4p.fit` *Method of Moment Estimates of Shape- and Location Parameters of the Four-Parameter Beta Distribution.*

Description

An implementation of the method of moments estimation of four-parameter beta distribution parameters presented by Hanson (1991). Given a string of values, calculates the shape- and location parameters required to produce a four-parameter beta distribution with the same mean, variance, skewness and kurtosis (i.e., the first four moments) as the observed-score distribution.

Usage

```
Beta.4p.fit(scores)
```

Arguments

`scores`	A vector of values to which the four-parameter beta distribution is to be fitted.

Value

A list of parameter-values required to produce a four-parameter beta distribution with the same first four moments as the observed distribution.

References

Hanson, Bradley A. (1991). Method of Moments Estimates for the Four-Parameter Beta Compound Binomial Model and the Calculation of Classification Consistency Indexes.American College Testing Research Report Series.

Lord, Frederic M. (1965). A Strong True-Score Theory, With Applications. Psychometrika, 30(3).

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0.
set.seed(1234)
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3))
hist(testdata, xlim = c(0, 100))
```

```
# To fit and retrieve the parameters for a four-parameter beta distribution
# to the observed-score distribution using Beta.4p.fit():
Beta.4p.fit(testdata)
```

`Beta.gfx.poly.cdf` *Coordinate Generation for Marking an Area Under the Curve for the Beta Cumulative Probability Density Distribution.*

Description

Plotting tool, producing a two-column matrix with values of y corresponding to locations on x. Useful for shading areas under the curve when tracing the line for the Standard Beta cumulative probability function.

Usage

```
Beta.gfx.poly.cdf(from, to, by, alpha, beta, l = 0, u = 1)
```

Arguments

`from`	The point of the x-axis from where to start producing y-density values.
`to`	The point of the x-axis to where y-density values are to be produced.
`by`	The resolution (or spacing) at which to produce y-density values.
`alpha`	The Alpha shape-parameter value for the Standard Beta cumulative probability distribution.
`beta`	The Beta shape-parameter fort he Standard Beta cumulative probability distribution.
`l`	The lower-bound location parameter of the Beta distribution.
`u`	The upper-bound location parameter of the Beta distribution.

Value

A two-column matrix with cumulative probability-values of y to plot against corresponding location values of x.

Examples

```
# To box in an area under a four-parameter Beta cumulative distribution with
# location parameters l = .25 and u = 75, and shape parameters
# alpha = 5 and beta = 3, from .4 to .6:
plot(NULL, xlim = c(0, 1), ylim = c(0, 1))
coords <- Beta.gfx.poly.cdf(from = .4, to = .6, by = .001, alpha = 5,
beta = 3, l = .25, u = .75)
polygon(coords)
```

`Beta.gfx.poly.pdf` *Coordinate Generation for Marking an Area Under the Curve for the Beta Probability Density Distribution.*

Description

Plotting tool, producing a two-column matrix with values of `y` corresponding to locations on `x`. Useful for shading areas under the curve when tracing the line for the Standard Beta probability density function.

Usage

```
Beta.gfx.poly.pdf(from, to, by, alpha, beta, l = 0, u = 1)
```

Arguments

`from`	The point of the `x`-axis from where to start producing `y`-density values.
`to`	The point of the x-axis to where y-density values are to be produced.
`by`	The resolution (or spacing) at which to produce y-density values.
`alpha`	The Alpha shape-parameter value for the Standard Beta probability density distribution.
`beta`	The Beta shape-parameter fort he Standard Beta probability density distribution.
`l`	The lower-bound location parameter of the Beta distribution.
`u`	The upper-bound location parameter of the Beta distribution.

Value

A two-column matrix with density-values of y to plot against corresponding location values of `x`.

Examples

```
# To box in an area under a four-parameter beta distribution with location
# parameters l = .25 and u = .75, and shape parameters
# alpha = 5 and beta = 3, from .4 to .6:
plot(NULL, xlim = c(0, 1), ylim = c(0, 7))
coords <- Beta.gfx.poly.pdf(from = .4, to = .6, by = .001, alpha = 5,
beta = 3, l = .25, u = .75)
polygon(coords)
```

`Beta.gfx.poly.qdf` *Coordinate Generation for Marking an Area Under the Curve for the Beta Quantile Density Distribution.*

Description

Plotting tool, producing a two-column matrix with values of `y` corresponding to locations on `x`. Useful for shading areas under the curve when tracing the line for the Standard Beta probability quantile function.

Usage

```
Beta.gfx.poly.qdf(from, to, by, alpha, beta, l = 0, u = 1)
```

Arguments

`from`	The point of the `x`-axis from where to start producing `y`-quantile values.
`to`	The point of the `x`-axis to where `y`-quantile values are to be produced.
`by`	The resolution (or spacing) at which to produce `y`-density values.
`alpha`	The Alpha shape-parameter value for the Standard Beta probability distribution.
`beta`	The Beta shape-parameter for the Standard Beta probability distribution.
`l`	The lower-bound location parameter of the Beta distribution.
`u`	The upper-bound location parameter of the Beta distribution.

Value

A two-column matrix with quantile-values of `y` to plot against corresponding location values of `x`.

Examples

```
# To box in an area under a four-parameter beta quantile distribution with
# location parameters l = .25 and u = 75, and shape parameters
# alpha = 5 and beta = 3, from .4 to .6:
plot(NULL, xlim = c(0, 1), ylim = c(0, 1))
coords <- Beta.gfx.poly.qdf(from = .4, to = .6, by = .001, alpha = 5,
beta = 3, l = .25, u = .75)
polygon(coords)
```

betamoments *Compute Moments of Two-to-Four Parameter Beta Probability Density Distributions.*

Description

Computes Raw, Central, or Standardized moment properties of defined Standard Beta probability density distributions.

Usage

```
betamoments(
  a,
  b,
  l = 0,
  u = 1,
  types = c("raw", "central", "standardized"),
  orders = 4
)
```

Arguments

a	The Alpha shape parameter of the PDD.
b	The Beta shape parameter of the PDD.
l	The first (lower) location parameter of a four-parameter distribution.
u	The second (upper) location parameter of a four-parameter distribution.
types	A character vector determining which moment-types are to be calculated. Permissible values are "raw", "central", and "standardized".
orders	The number of moment-orders to be calculated for each of the moment-types.

Value

A list of moment types, each a list of moment orders.

References

Hanson, B. A (1991). Method of Moments Estimates for the Four-Parameter Beta Compound Binomial Model and the Calculation of Classification Consistency Indexes. American College Testing Research Report Series.

Examples

```
# Assume some variable follows a four-parameter beta distribution with
# location parameters l = 0.25 and u = .75, and shape
# parameters a = 5 and b = 3. To compute the first four
# raw, central, and standardized moments of this distrubution using
```

```
# betamoments():
betamoments(a = 5, b = 3, l = .25, u = .75,
types = c("raw", "central", "standardized"), orders = 4)
```

BMS *Beta Shape Parameter Given Mean and Variance of a Standard Beta PDD.*

Description

Calculates the Beta value required to produce a Standard (two-parameter) Beta probability density distribution with defined mean and variance or standard deviation.

Usage

```
BMS(mean, var, sd = NULL)
```

Arguments

mean	The mean of the target Standard Beta probability density distribution.
var	The variance of the target Standard Beta probability density distribution.
sd	The standard deviation of the target Standard Beta probability density distribution.

Value

A numeric value representing the required value for the Beta shape-parameter in order to produce a Standard Beta probability density distribution with the target mean and variance.

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0, rescaled to proportion
# of maximum.
set.seed(1234)
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3)) / 100
hist(testdata, xlim = c(0, 100))

# To find the beta shape-parameter of a Standard (two-parameter) Beta
# distribution with the same mean and variance as the observed-score
# distribution using BMS():
BMS(mean(testdata), var(testdata))
```

caStats *Classification Accuracy Statistics.*

Description

Provides a set of statistics often used for conveying information regarding the certainty of classifications based on tests.

Usage

```
caStats(tp, tn, fp, fn)
```

Arguments

tp	The frequency or rate of true-positive classifications.
tn	The frequency or rate of true-negative classifications.
fp	The frequency or rate of false-positive classifications.
fn	The frequency or rate of false-negative classifications.

Value

A list of diagnostic performance statistics based on true/false positive/negative statistics. Specifically, the sensitivity, specificity, positive likelihood ratio (LR.pos), negative likelihood ratio (LR.neg), positive predictive value (PPV), negative predictive value (NPV), Youden's J. (Youden.J), and Accuracy.

References

Glas et al. (2003). The Diagnostic Odds Ratio: A Single Indicator of Test Performance, Journal of Clinical Epidemiology, 1129-1135, 56(11). doi: 10.1016/S0895-4356(03)00177-X

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0.
set.seed(1234)
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3))
hist(testdata, xlim = c(0, 100))

# Suppose the cutoff value for attaining a pass is 50 items correct, and
# that the reliability of this test was estimated to 0.7. First, compute the
# estimated confusion matrix using LL.CA():
cmat <- LL.CA(x = testdata, reliability = .7, cut = 50, min = 0,
max = 100)$confusionmatrix

# To estimate and retrieve diagnostic performance statistics using caStats(),
# feed it the appropriate entries of the confusion matrix.
caStats(tp = cmat["True", "Fail"], tn = cmat["True", "Pass"],
fp = cmat["False", "Fail"], fn = cmat["False", "Pass"])
```

cba *Calculate Cronbach's Alpha from supplied variables.*

Description

Calculates Cronbach's Alpha, a very commonly used index for assessing the reliability / internal consistency of a sum-score. Often interpreted as the mean correlation across all possible split-half alternate forms of the test.

Usage

```
cba(x)
```

Arguments

x	A data-frame or matrix of numerical values where rows are across-items within-respondent observation vectors, and columns are within-item across-respondents observation vectors.

Value

Cronbach's Alpha for the sum-score of supplied variables.

Note

Missing values are treated by passing `na.rm = TRUE` to the `var` function call.

Be aware that this function does not issue a warning if there are negative correlations between variables in the supplied data-set.

References

Cronbach, L.J. (1951). Coefficient alpha and the internal structure of tests. Psychometrika 16, 297–334. doi: 10.1007/BF02310555

Examples

```
# Generate some fictional data. Say 100 students take a 50-item long test
# where all items are equally difficult.
set.seed(1234)
p.success <- rBeta.4P(100, .25, .75, 5, 3)
for (i in 1:50) {
  if (i == 1) {
    rawdata <- matrix(nrow = 100, ncol = 50)
  }
  rawdata[, i] <- rbinom(100, 1, p.success)
}
# To calculate Cronbach's Alpha for this test:
cba(rawdata)
```

ccStats *Classification Consistency Statistics.*

Description

Provides a set of statistics often used for conveying information regarding the consistency of classifications based on tests.

Usage

```
ccStats(ii, ij, ji, jj)
```

Arguments

ii	The frequency or rate of consistent classifications into category "i".
ij	The frequency or rate of inconsistent classifications into categories "i" and "j".
ji	The frequency or rate of inconsistent classifications into categories "j" and "i".
jj	The frequency or rate of consistent classifications into category "j".

Value

A list of classification consistency statistics. Specifically, the coefficient of consistent classification (p), the coefficient of consistent classification by chance (p_c), and Cohen's Kappa coefficient.

References

Hanson, Bradley A. (1991). Method of Moments Estimates for the Four-Parameter Beta Compound Binomial Model and the Calculation of Classification Consistency Indexes. American College Testing.

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0.
set.seed(1234)
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3))
hist(testdata, xlim = c(0, 100))

# Suppose the cutoff value for attaining a pass is 50 items correct, and
# that the reliability of this test was estimated to 0.7. First, compute the
# estimated consistency matrix using LL.CA():
cmat <- LL.CA(x = testdata, reliability = .7, cut = 50, min = 0,
max = 100)$consistencymatrix

# To estimate and retrieve diagnostic performance statistics using caStats(),
# feed it the appropriate entries of the consistency matrix.
ccStats(ii = cmat["i", "i"], ij = cmat["i", "j"],
ji = cmat["j", "i"], jj = cmat["j", "j"])
```

dBeta.4P — *Probability Density under the Four-Parameter Beta PDD.*

Description

Gives the density at desired values of x under the Four-Parameter Beta PDD.

Usage

```
dBeta.4P(x, l, u, alpha, beta)
```

Arguments

x	Value of x.
l	The first (lower) location parameter.
u	The second (upper) location parameter.
alpha	The first shape parameter.
beta	The second shape parameter.

Value

The value for the probability density at specified values of x.

Examples

```
# Assume some variable follows a four-parameter beta distribution with
# location parameters l = 0.25 and u = .75, and shape
# parameters alpha = 5 and beta = 3. To compute the
# probability density at a specific point of the distribution (e.g., .5)
# using dBeta.4P():
dBeta.4P(x = .5, l = .25, u = .75, alpha = 5, beta = 3)
```

dBeta.pBeta — *An implementation of the Beta-density Compound Cumulative-Beta Distribution.*

Description

The Beta Compound Beta distribution: The product of the four-parameter Beta probability density function and the beta cumulative probability function. Used in the Livingston and Lewis approach to classification accuracy and consistency, the output can be interpreted as the population density of passing scores produced at "x" (a value of true-score).

Usage

```
dBeta.pBeta(x, l, u, a, b, n, c, lower.tail = FALSE)
```

Arguments

x	x-axis input for which p (proportion or probability) is to be computed.
l	The lower-bound of the four-parameter Beta distribution.
u	The upper-bound of the four-parameter Beta distribution.
a	The alpha shape-parameter of the Beta density distribution.
b	The beta shape-parameter of the Beta density distribution.
n	The number of trials for the Beta cumulative probability distribution.
c	The "true-cut" (proportion) of on the Beta cumulative probability distribution.
lower.tail	Logical. Whether to compute the lower or upper tail of the Beta cumulative probability distribution. Default is FALSE (i.e., upper tail).

References

Hanson, Bradley A. (1991). Method of Moments Estimates for the Four-Parameter Beta Compound Binomial Model and the Calculation of Classification Consistency Indexes.American College Testing Research Report Series.

Livingston, Samuel A. and Lewis, Charles. (1995). Estimating the Consistency and Accuracy of Classifications Based on Test Scores. Journal of Educational Measurement, 32(2).

Lord, Frederic M. (1965). A Strong True-Score Theory, With Applications. Psychometrika, 30(3).

Examples

```
# Given a four-parameter Beta distribution with parameters l = 0.25, u = 0.75,
# alpha = 5, and beta = 3, and a Beta error distribution with number of
# trials (n) = 10 and a cutoff-point (c) at 50% correct (i.e., proportion correct
# of 0.5), the population density of passing scores produced at true-score
# (x) = 0.5 can be calculated as:
dBeta.pBeta(x = 0.5, l = 0.25, u = 0.75, a = 5, b = 3, n = 10, c = 0.5)

# Conversely, the density of failing scores produced at x can be calculated
# by passing the additional argument "lower.tail = TRUE" to the function.
# That is:
dBeta.pBeta(x = 0.5, l = 0.25, u = 0.75, a = 5, b = 3, n = 10, c = 0.5, lower.tail = TRUE)

# By integration, the population proportion of (e.g.) passing scores in some
# region of the true-score distribution (e.g. between 0.25 and 0.5) can be
# calculated as:
integrate(function(x) { dBeta.pBeta(x, 0.25, .75, 5, 3, 10, 0.5) }, lower = 0.25, upper = 0.5)
```

---

dBeta.pBinom *An implementation of the Beta-density Compound Cumulative-Binomial Distribution.*

Description

The Beta Compound Binomial distribution: The product of the four-parameter Beta probability density function and the binomial cumulative probability mass function. Used in the Livingston and Lewis approach to classification accuracy and consistency, the output can be interpreted as the population density of passing scores produced at "x" (a value of true-score).

Usage

```
dBeta.pBinom(x, l, u, a, b, n, c, lower.tail = FALSE)
```

Arguments

x	x-axis input for which p (proportion or probability) is to be computed.
l	The lower-bound of the four-parameter Beta distribution.
u	The upper-bound of the four-parameter Beta distribution.
a	The alpha shape-parameter of the Beta distribution.
b	The beta shape-parameter of the Beta distribution.
n	The number of trials for the Binomial distribution.
c	The "true-cut" (proportion) of on the Binomial distribution.
lower.tail	Logical. Whether to compute the lower or upper tail of the Binomial distribution. Default is FALSE (i.e., upper tail).

References

Hanson, Bradley A. (1991). Method of Moments Estimates for the Four-Parameter Beta Compound Binomial Model and the Calculation of Classification Consistency Indexes.American College Testing Research Report Series.

Livingston, Samuel A. and Lewis, Charles. (1995). Estimating the Consistency and Accuracy of Classifications Based on Test Scores. Journal of Educational Measurement, 32(2).

Lord, Frederic M. (1965). A Strong True-Score Theory, With Applications. Psychometrika, 30(3).

Examples

```
# Given a four-parameter Beta distribution with parameters l = 0.25, u = 0.75,
# alpha = 5, and beta = 3, and a Binomial error distribution with number of
# trials (n) = 10 and a cutoff-point (c) at 50% correct (i.e., proportion correct
# of 0.5), the population density of passing scores produced at true-score
# (x) = 0 can be calculated as:
dBeta.pBinom(x = 0.5, l = 0.25, u = 0.75, a = 5, b = 3, n = 10, c = 0.5)

# Conversely, the density of failing scores produced at x can be calculated
# by passing the additional argument "lower.tail = TRUE" to the function.
# That is:
dBeta.pBinom(x = 0.5, l = 0.25, u = 0.75, a = 5, b = 3, n = 10, c = 0.5, lower.tail = TRUE)

#By integration, the population proportion of (e.g.) passing scores in some
#region of the true-score distribution (e.g. between 0.25 and 0.5) can be
```

```
#calculated as:
integrate(function(x) { dBeta.pBinom(x, 0.25, .75, 5, 3, 10, 0.5) }, lower = 0.25, upper = 0.5)
```

---

dBetaMS *Density Under a Specific Point of the Standard Beta PDD with Specific Mean and Variance or Standard Deviation.*

Description

Calculates the density under specific points of the Standard Beta probability density distribution with defined mean and variance or standard deviation.

Usage

```
dBetaMS(x, mean, var = NULL, sd = NULL)
```

Arguments

x	A specific point on the x-axis of the Standard Beta PDD.
mean	The mean of the target Standard Beta probability density distribution.
var	The variance of the target Standard Beta probability density distribution.
sd	The standard deviation of the target Standard Beta probability density distribution.

Value

A numeric value representing the required value for the Beta Shape-parameter in order to produce a Standard Beta probability density distribution with the target mean and variance.

Examples

```
# To compute the density at a specific point (e.g., .5) along the Standard
# (two-parameter) PDD with mean of .6 and variance of .04:
dBetaMS(x = .5, mean =.6, var = .04)
```

ETL *Livingston and Lewis' "Effective Test Length".*

Description

According to Livingston and Lewis (1995), "The effective test length corresponding to a test score is the number of discrete, dichotomously scored, locally independent, equally difficult items required to produce a total score of the same reliability."

Usage

```
ETL(mean, variance, l = 0, u = 1, reliability)
```

Arguments

mean	The mean of the observed-score distribution.
variance	The variance of the observed-score distribution.
l	The lower-bound of the observed-score distribution. Default is 0 (assuming observed scores represent proportions).
u	The upper-bound of the observed-score distribution. Default is 1 (assuming observed scores represent proportions).
reliability	The reliability of the observed scores (proportion of observed-score distribution variance shared with true-score distribution).

Value

An estimate of the effective length of a test, given the stability of the observations it produces.

References

Livingston, Samuel A. and Lewis, Charles. (1995). Estimating the Consistency and Accuracy of Classifications Based on Test Scores. Journal of Educational Measurement, 32(2).

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0.
set.seed(1234)
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3))
hist(testdata, xlim = c(0, 100))

# Suppose the reliability of this test was estimated to 0.7. To estimate and
# retrieve the effective test length using ETL():
ETL(mean = mean(testdata), variance = var(testdata), l = 0, u = 100,
reliability = .7)
```

LL.CA *An Implementation of the Livingston and Lewis (1995) Approach to Estimate Classification Consistency and Accuracy based on Observed Test Scores and Test Reliability.*

Description

An implementation of what has been come to be known as the "Livingston and Lewis approach" to classification consistency and accuracy, which by employing a compound beta-binomial distribution assumes that true-scores conform to the four-parameter beta distribution, and errors of measurement to the binomial distribution. Under these assumptions, the expected classification consistency and accuracy of tests can be estimated from observed outcomes and test reliability.

Usage

```
LL.CA(
  x = NULL,
  reliability,
  cut,
  min = 0,
  max = 1,
  error.model = "binomial",
  truecut = NULL,
  output = c("accuracy", "consistency"),
  override = FALSE,
  grainsize = NULL
)
```

Arguments

x	A vector of observed scores for which a beta-distribution is to be fitted, or a list of pre-defined true-score distribution parameter values. If a list is provided, the list entries must be named after the parameters: `l` and `u` for the location parameters, and `alpha` and `beta` for the shape parameters.
reliability	The observed-score squared correlation (i.e., proportion of shared variance) with the true-score.
cut	The cutoff value for classifying observations into pass or fail categories.
min	The minimum value possible to attain on the test. Default is 0 (assuming `x` represent proportions).
max	The maximum value possible to attain on the test. Default is 1 (assuming `x` represent proportions).
error.model	The probability distribution to be used for producing the sampling distributions at different points of the true-score scale. Options are `binomial` and `beta`. The binomial distribution is discrete, and is the distribution used originally by Livingston and Lewis. Use of the binomial distribution involves a rounding of the

effective test length to the nearest integer value. The Beta distribution is continuous, and does not involve rounding of the effective test length.

`truecut` Optional specification of a "true" cutoff. Useful for producing ROC curves.

`output` Character vector indicating which types of statistics (i.e, accuracy and/or consistency) are to be computed and included in the output. Permissible values are `"accuracy"` and `"consistency"`.

`override` Logical value indicating whether to override the automatic default to the two-parameter Beta true-score distribution if the four-parameter fitting procedure produces impermissible parameter estimates. Default is `FALSE`.

`grainsize` Outdated and inert. Maintained for compatibility. Will be removed completely in future update.

Value

A list containing the estimated parameters necessary for the approach (i.e., the effective test-length and the beta distribution parameters), the confusion matrix containing estimated proportions of true/false pass/fail categorizations for a test, diagnostic performance statistics, and / or a classification consistency matrix and indices. Accuracy output includes a confusion matrix and diagnostic performance indices, and consistency output includes a consistency matrix and consistency indices `p` (expected proportion of agreement between two independent test administrations), `p_c` (proportion of agreement on two independent administrations expected by chance alone), and `Kappa` (Cohen's Kappa).

Note

It should be noted that this implementation differs from the original articulation of Livingston and Lewis (1995) in some respects. First, the procedure includes a number of diagnostic performance (accuracy) indices which the original procedure enables but that were not included. Second, the possibility of employing a two-parameter Beta error distribution in place of the binomial error distribution is not part of the original procedure. Third, the way consistency is calculated differs substantially from the original articulation of the procedure, which made use of a split-half approach. Rather, this implementation uses the approach to calculating classification consistency outlined by Hanson (1991).

References

Livingston, Samuel A. and Lewis, Charles. (1995). Estimating the Consistency and Accuracy of Classifications Based on Test Scores. Journal of Educational Measurement, 32(2).

Hanson, Bradley A. (1991). Method of Moments Estimates for the Four-Parameter Beta Compound Binomial Model and the Calculation of Classification Consistency Indexes. American College Testing.

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0.
set.seed(1234)
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3))
```

```
hist(testdata, xlim = c(0, 100))

# Suppose the cutoff value for attaining a pass is 50 items correct, and
# that the reliability of this test was estimated to 0.7. To estimate and
# retrieve the estimated parameters, confusion matrix, consistency and
# accuracy statistics using LL.CA():
LL.CA(x = testdata, reliability = .7, cut = 50, min = 0, max = 100)

# Alternatively to supplying scores to which a true-score distribution is
# to be fit, a list with true-score distribution parameter values can be
# supplied manually, foregoing the need for actual data. The list entries
# must be named. "l" is the lower-bound and "u" the upper bound location
# parameters of the true-score distribution, and "alpha" and "beta" the
# shape parameters.
trueparams <- list("l" = 0.25, "u" = 0.75, "alpha" = 5, "beta" = 3)
LL.CA(x = trueparams, reliability = .7, cut = 50, min = 0, max = 100)
```

LL.ROC *ROC curves for the Livingston and Lewis approach.*

Description

Generate a ROC curve plotting the false-positive rate against the true-positive rate at different cut-off values across the observed proportion-score scale.

Usage

```
LL.ROC(
  x = NULL,
  reliability,
  min = 0,
  max = 1,
  truecut,
  AUC = FALSE,
  maxJ = FALSE,
  raw.out = FALSE
)
```

Arguments

x	A vector of observed results.
reliability	The reliability coefficient of the test.
min	The minimum possible value to attain on the observed-score scale. Default is 0 (assuming x represent proportions).
max	The maximum possible value to attain on the observed-score scale. Default is 1 (assuming x represent proportions).
truecut	The true point along the x-scale that marks the categorization-threshold.

AUC	Calculate and include the area under the curve? Default is FALSE.
maxJ	Mark the point along the curve where Youden's J statistic is maximized? Default is FALSE.
raw.out	Give raw coordinates as output rather than plot? Default is FALSE.

Value

A plot tracing the ROC curve for the test, or matrix of coordinates if raw.out is TRUE.

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0.
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3))
hist(testdata, xlim = c(0, 100))

# Suppose the cutoff value for attaining a pass is 50 items correct, and
# that the reliability of this test was estimated to 0.7. To produce a plot
# with an ROC curve using LL.ROC(), along with the AUC statistics and the
# points at which Youden's J. is maximized:
LL.ROC(x = testdata, reliability = .7, truecut = 50, min = 0, max = 100,
AUC = TRUE, maxJ = TRUE)
```

MLA *Most Likely True Alpha Value Given Observed Outcome.*

Description

Given a fitted Standard (two-parameter) Beta Distribution, return the alpha shape-parameter value where the observed mean becomes the mode.

Usage

```
MLA(a, b, x = NULL, n = NULL)
```

Arguments

a	Observed alpha-parameter value for fitted Standard Beta PDD.
b	Observed beta-parameter value for fitted Standard Beta PDD.
x	Observed proportion-correct outcome.
n	Test-length.

Value

The Alpha shape-parameter value for the Standard Beta probability density distribution where the observed mean is the expected mode.

Examples

```
# Assuming a prior Standard (two-parameter) Beta distribution is fit, which
# yield an alpha parameter of 10 and a beta parameter of 8, calculate the
# true-alpha parameter most likely to have produced the observations:
MLA(a = 10, b = 8)
```

MLB *Most Likely True Beta Value Given Observed Outcome.*

Description

Assuming a prior standard (two-parameter) Beta Distribution, return the beta shape-parameter value where the observed mean becomes the mode.

Usage

```
MLB(a, b, x = NULL, n = NULL)
```

Arguments

a	Observed alpha-parameter value for fitted Standard Beta PDD.
b	Observed beta-parameter value for fitted Standard Beta PDD.
x	Observed proportion-correct outcome.
n	Test-length.

Value

The Beta shape-parameter value for the Standard Beta probability density distribution where the observed mean is the expected mode.

Examples

```
# Assuming a prior Standard (two-parameter) Beta distribution is fit, which
# yield an alpha parameter of 10 and a beta parameter of 8, calculate the
# true-beta parameter most likely to have produced the observations:
MLB(a = 10, b = 8)
```

MLM — *Most Likely Mean of the Standard Beta PDD, Given that the Observation is Considered the Most Likely Observation of the Standard Beta PDD (i.e., Mode).*

Description

Assuming a prior Standard (two-parameter) Beta Distribution, returns the expected mean of the distribution under the assumption that the observed value is the most likely value of the distribution.

Usage

```
MLM(a, b, x = NULL, n = NULL)
```

Arguments

a	Observed alpha value for fitted Standard Beta PDD.
b	Observed beta value for fitted Standard Beta PDD.
x	Observed proportion-correct outcome.
n	Test-length.

Value

The expected mean of the Standard Beta probability density distribution, for which the observed mean is the most likely value.

Examples

```
# Assuming a prior Standard (two-parameter) Beta distribution is fit, which
# yield an alpha parameter of 10 and a beta parameter of 8, calculate the
# true-mean most likely to have produced the observations:
MLM(a = 10, b = 8)
```

observedmoments — *Compute Moments of Observed Value Distribution.*

Description

Computes Raw, Central, or Standardized moment properties of a vector of observed scores.

Usage

```
observedmoments(
  x,
  type = c("raw", "central", "standardized"),
  orders = 4,
  correct = TRUE
)
```

Arguments

x	A vector of values, the distribution of which moments are to be calculated.
type	A character vector determining which moment-types are to be calculated. Permissible values are "raw", "central", and "standardized".
orders	The number of moment-orders to be calculated for each of the moment-types.
correct	Whether to include bias correction in estimation of orders. Default is TRUE.

Value

A list of moment types, each a list of moment orders.

Examples

```
# Generate some fictional data. Say, 100 individuals take a test with a
# maximum score of 100 and a minimum score of 0.
set.seed(1234)
testdata <- rbinom(100, 100, rBeta.4P(100, .25, .75, 5, 3))
hist(testdata, xlim = c(0, 100))

# To compute the first four raw, central, and standardized moments for this
# distribution of observed scores using observedmoments():
observedmoments(x = testdata, type = c("raw", "central", "standardized"),
orders = 4, correct = TRUE)
```

---

pBeta.4P — *Cumulative Probability Function under the Four-Parameter Beta Probability Density Distribution.*

Description

Function for calculating the proportion of observations up to a specifiable quantile under the Four-Parameter Beta Distribution.

Usage

```
pBeta.4P(q, l, u, alpha, beta, lt = TRUE)
```

Arguments

q	The quantile or a vector of quantiles for which the proportion is to be calculated.
l	The first (lower) location parameter.
u	The second (upper) location parameter.
alpha	The first shape parameter.
beta	The second shape parameter.
lt	Whether the proportion to be calculated is to be under the lower or upper tail. Default is TRUE (lower tail).

Value

A vector of proportions of observations falling under specified quantiles under the four-parameter beta distribution.

Examples

```
# Assume some variable follows a four-parameter beta distribution with
# location parameters l = 0.25 and u = .75, and shape
# parameters alpha = 5 and beta = 3. To compute the
# cumulative probability at a specific point of the distribution (e.g., .5)
# using pBeta.4P():
pBeta.4P(q = .5, l = .25, u = .75, alpha = 5, beta = 3)
```

---

pBetaMS *Probability of Some Specific Observation under the Standard Beta PDD with Specific Mean and Variance.*

Description

Calculates the probability of some specific observation falling under a specified interval ([0, x] or [x, 1]) under the Standard Beta probability density distribution with defined mean and variance or standard deviation.

Usage

```
pBetaMS(q, mean, var = NULL, sd = NULL, lt = TRUE)
```

Arguments

q	A specific point on the x-axis of the Standard Beta probability density distribution with a defined mean and variance.
mean	The mean of the target Standard Beta probability density distribution.
var	The variance of the target Standard Beta probability density distribution.
sd	The standard deviation of the target Standard Beta probability density distribution.
lt	Whether the density that should be considered is between the lower-end (i.e., [0 -> x]) or the higher-end of the distribution (i.e., [x -> 1]).

Value

A value representing the probability of a random draw from the Standard Beta probability density distribution with a defined mean and variance being from one of two defined intervals (i.e., [0 -> x] or [x -> 1]).

Examples

```
# To compute the proportion of the density under the lower-end tail of a
# point along the Standard (two-parameter) PDD (e.g., .5) with mean of .6
# and variance of .04:
pBetaMS(q = .5, mean = .6, var = .04)
```

---

qBeta.4P *Quantile Given Probability Under the Four-Parameter Beta Distribution.*

Description

Function for calculating the quantile (i.e., value of x) for a given proportion (i.e., the value of y) under the Four-Parameter Beta Distribution.

Usage

```
qBeta.4P(p, l, u, alpha, beta, lt = TRUE)
```

Arguments

p	A vector (or single value) of proportions or probabilities for which the corresponding value of x (i.e., the quantiles) are to be calculated.
l	The first (lower) location parameter.
u	The second (upper) location parameter.
alpha	The first shape parameter.
beta	The second shape parameter.
lt	Logical. Whether the quantile(s) to be calculated is to be under the lower or upper tail. Default is TRUE (lower tail).

Value

A vector of quantiles for specified probabilities or proportions of observations under the four-parameter beta distribution.

Examples

```
# Assume some variable follows a four-parameter beta distribution with
# location parameters l = 0.25 and u = .75, and shape
# parameters alpha = 5 and beta = 3. To compute the
# quantile at a specific point of the distribution (e.g., .5)
# using qBeta.4P():
qBeta.4P(p = .5, l = .25, u = .75, alpha = 5, beta = 3)
```

qBetaMS *Quantile Containing Specific Proportion of the Distribution, Given a Specific Probability of the Standard Beta PDD with Specific Mean and Variance or Standard Deviation.*

Description

Calculates the quantile corresponding to a specific probability of some observation falling within the [0, x] (`lt = TRUE`) or [x, 1] (`lt = FALSE`) interval under the Standard Beta probability density distribution with defined mean and variance or standard deviation.

Usage

```
qBetaMS(p, mean, var = NULL, sd = NULL, lt = TRUE)
```

Arguments

`p`	A value of probability marking the point of the Y-axis to correspond to the X-axis.
`mean`	The mean of the target Standard Beta probability density distribution.
`var`	The variance of the target Standard Beta probability density distribution.
`sd`	The standard deviation of the target Standard Beta probability density distribution.
`lt`	Logical. Specifies which end of the tail for which to calculate quantile. Default is `TRUE` (meaning, find q for lower tail.)

Value

A numeric value representing the quantile for which the specified proportion of observations fall within.

Examples

```
# To compute the quantile at a specific point (e.g., .5) along the Standard
# (two-parameter) PDD with mean of .6 and variance of .04:
qBetaMS(p = .5, mean =.6, var = .04)
```

`rBeta.4P` *Random Number Generation under the Four-Parameter Beta Probability Density Distribution.*

Description

Function for generating random numbers from a specified Four-Parameter Beta Distribution.

Usage

```
rBeta.4P(n, l, u, alpha, beta)
```

Arguments

`n`	Number of draws.
`l`	The first (lower) location parameter.
`u`	The second (upper) location parameter.
`alpha`	The first shape parameter.
`beta`	The second shape parameter.

Value

A vector with length `n` of random values drawn from the Four-Parameter Beta Distribution.

Examples

```
# Assume some variable follows a four-parameter beta distribution with
# location parameters l = 0.25 and u = .75, and shape
# parameters alpha = 5 and beta = 3. To draw a random
# value from this distribution using rBeta.4P():
rBeta.4P(n = 1, l = .25, u = .75, alpha = 5, beta = 3)
```

`rBetaMS` *Random Draw from the Standard Beta PDD With Specific Mean and Variance.*

Description

Draws random samples of observations from the Standard Beta probability density distribution with defined mean and variance.

Usage

```
rBetaMS(n, mean, var = NULL, sd = NULL)
```

Arguments

n	Number of observations to be drawn from under the Standard Beta PDD.
mean	The mean of the target Standard Beta probability density distribution.
var	The variance of the target Standard Beta probability density distribution.
sd	The standard deviation of the target Standard probability density distribution.

Value

A vector of length n, each value representing a random draw from the Standard Beta probability density distribution with defined mean and variance.

Index